

# CIRCUIT CELLAR<sup>®</sup> INK<sup>®</sup>

THE COMPUTER APPLICATIONS JOURNAL

#83 June 1997

## COMMUNICATIONS

Identifying DSP Techniques Graphically

Creating a PC-Telephone Interface

Setting Up On- and Off-Hook Caller ID

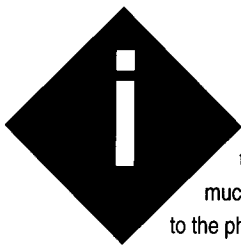
EPC: Exploring the Flash Translation Layer



\$3.95 U.S.  
\$4.95 Canada

# TASK MANAGER

## Life's Little Mysteries



think one of the great ironies in my life is how much I enjoy every toy to come along that connects to the phone line, and how much I dislike talking on the telephone. Give me E-mail, give me faxes, but ask me to call

you back and you might as well shove bamboo sticks under my nails.

At last count, I have eight devices connected to our phone line at home, and only three of those are telephones. Between the modems, HCS interface, answering machine, and caller-ID box, I can tell you who called, when they called, and how often they called. I can check the house while I'm away, remotely turn things on and off, and find out when the mailman came. Just don't ask me to talk on the darn thing.

I carry a cell phone with me wherever I go, just in case someone needs to contact me or an emergency comes up. (Only my wife knows the number, though.) And, I've even caught myself wishing I had a pager like the alphanumeric one my wife has for her job. But, I don't really need one. I just think they're neat. (However, many who carry them despise the things. Maybe I'm lucky.)

I think it's therefore appropriate that so many of our Communications theme articles this month deal with wireless and telephone communications.

In our first article, David Tweed starts looking at what it takes to receive and decode timecode signals transmitted from Canada. By concentrating on DSP design techniques without going into the heavy-duty math, he presents the project in a more understandable way.

Dave Ryan and Asher Hazanchuk dial us into what's involved with decoding caller ID both on and off hook. On-hook decoding is easy. Off-hook support isn't.

Next, Chris Sakkas presents a project I'd love to get my hands on: a PC-based voice mail system. He does everything using high-integration chips and software, keeping it inexpensive and very flexible.

Finally, Art Sobel continues his presentation of the ARM7500 with a look at some firmware device drivers to support it.

In our columns, Hugh Anglin starts a two-part MicroSeries on machine vision by exploring some of the fundamental issues surrounding any such design, Jeff automates a mechanical toy Ciarcia-style, and Tom checks out the latest DSP from Texas Instruments.

The topic of the month in *EPC* is flash memory, and Raz Dan begins by explaining flash file systems and why FTL is the up-and-coming favorite. Rick Lehrbaum decides to embed a PC-based application using flash. But to do so, he first must survey the flash methods available. Finally, Fred Eady sets up a dog-and-pony show based on National's NS486SXF.

editor@circellar.com

# CIRCUIT CELLAR®

THE COMPUTER APPLICATIONS JOURNAL

**EDITORIAL DIRECTOR/PUBLISHER**  
Steve Ciarcia

**ASSOCIATE PUBLISHER**  
Sue Hodge

**EDITOR-IN-CHIEF**  
Ken Davidson

**CIRCULATION MANAGER**  
Rose Mansella

**MANAGING EDITOR**  
Janice Hughes

**CIRCULATION CONSULTANT**  
John S. Treworgy

**TECHNICAL EDITOR**  
Elizabeth Laurençot

**BUSINESS MANAGER**  
Jeannette Walters

**ENGINEERING STAFF**  
Jeff Bachiochi

**ADVERTISING COORDINATOR**  
Dan Gorsky

**WEST COAST EDITOR**  
Tom Cantrell

**CONTRIBUTING EDITORS**  
Rick Lehrbaum  
Fred Eady

**NEW PRODUCTS EDITOR**  
Harv Weiner

**ART DIRECTOR**  
KC Zienka

**PRODUCTION STAFF**  
John Gorsky  
James Soussounis

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

Direct subscription orders and subscription related questions to Circuit Cellar INK Subscriptions, P.O. Box 698, Holmes, PA 19043-9613 or call (800) 269-6301. POSTMASTER: Please send address changes to Circuit Cellar INK, Circulation Dept., P.O. Box 698, Holmes, PA 19043-9613.

Cover photograph © 1997 Barbara Swenson  
PRINTED IN THE UNITED STATES

For information on authorized reprints of articles, contact Jeannette Walters (860) 875-2199.

## HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

**NORTHEAST & MID-ATLANTIC**

**Barbara Best**  
(561) 694-2044  
Fax: (561) 694-2051  
B.Best-Hajar@worldnet.att.net

**MIDWEST & SOUTHEAST**

**Christa Collins**  
(954) 966-3939  
Fax: (954) 985-8457  
HajarChrista@worldnet.att.net

**WEST COAST**

**Barbara Jones & Shelley Rainey**  
(714) 540-3554  
Fax: (714) 540-7103  
shelley.hajar@worldnet.att.net

Circuit Cellar BBS—24 Hrs., 2400/9600/14.4k bps, 8 bits, no parity, 1 stop bit, (860) 871-1988. For information, mail to [info@circellar.com](mailto:info@circellar.com). World Wide Web: [www.circellar.com](http://www.circellar.com)

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 responsibility for the safe and proper function of reader-assembled projects based upon or from plans, descriptions, or information published in *Circuit Cellar INK®*.

Entire contents copyright © 1997 by Circuit Cellar Incorporated. All rights reserved. Circuit Cellar INK is a registered trademark of Circuit Cellar Inc. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

- 12 **DSP-Based Canadian Timecode Receiver**  
Part 1: Identifying DSP Techniques  
David Tweed
- 20 **On- and Off-Hook Caller ID Using DSP**  
*Dave Ryan & Asher Hazanchuk*
- 26 **PC Telephone Interface**  
*Chris Sakkas*
- 30 **Embedding the ARM7500**  
Part 2: Programming an Embedded Computer  
*Art Sobel*
- 66  **MicroSeries**  
Machine Vision  
Part 1: Industrial Inspection  
*Hugh Anglin*
- 74  **From the Bench**  
It Can't Be A Robot  
Part 1: There are No Arms and Legs!  
*Jeff Bachiochi*
- 78  **Silicon Update**  
High-Velocity DSP  
*Tom Cantrell*

- Task Manager** 2  
Ken Davidson
- Life's Little Mysteries**
- New Product News** 6  
edited by Harv Weiner
- Advertiser's index** 65
- Priority Interrupt** 96  
Steve Ciarcia  
A Winning Proposition

# INSIDE ISSUE 83

**EMBEDDED PC**

- 40 **Nouveau PC**
- 45 **An In-Depth Look at FTL**  
*Raz Dan*
- 53 **PCQ PC/104 Quarter**  
To ROM or NOT to ROM  
That is the Question  
*Rick Lehrbaum*
- 60 **APC Applied PCs**  
Right on Cue  
National Presents Slimline 'x86  
*Fred Eady*

[www.circellar.com](http://www.circellar.com)



# NEW PRODUCT NEWS

Edited by Harv Weiner

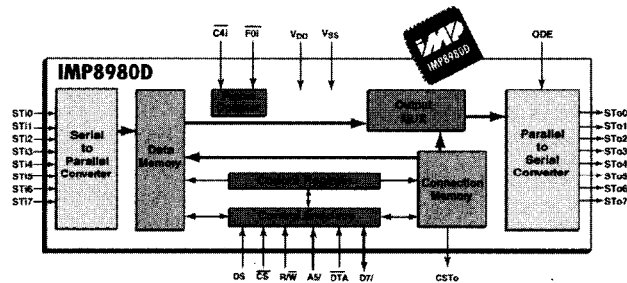
## CROSSPOINT DATA SWITCH IC

IMP has announced a digital crosspoint switch IC, accommodating 256 x 256 channels. The **IMP8980D** is a CMOS device that switches digital datastreams such as pulse code modulated (PCM) voice, video, or data signals. It establishes a path between any input and output over its internal ST-Bus (Serial Telecom Bus). Uses include digital exchange, PBX, and central-office applications.

To support 256 channels, the **IMP8980D** has eight each ST-Bus I/O pins. Via time-division multiplexing, the component-level 2048-kbps ST-Bus supports 32 logical data channels at 64 kbps at each device I/O pin. ST-Bus bit rate is divided into 8000 frames with 32 channels per frame.

In the Message mode, the system microcontroller can pass data onto an output channel. In the nonblocking Switching mode, the output can specify its input-channel data source. Multiple outputs can share an input, which is useful in message-broadcast applications.

A system microprocessor makes switched connections, writes data to output channels, and can receive data from input channels. In addition, the system micro-



controller can concurrently read input-channel data and write data to ST-Bus channel outputs. Large logical-switch structures are possible since the **IMP8980D** can set outputs into a high-impedance state on a per-channel basis.

Pricing for the 44-pin PLCC **IMP8980DP** and 40-pin DIP **IMP8980DE** starts at \$7.70 in quantity.

IMP, Inc.  
2830 N. First St.  
San Jose, CA **95134-2071**  
(408) 432-9100 • Fax: (408) 434-0335  
www.impweb.com

#501

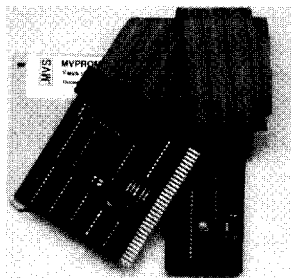
## 386 SBC \$83



OEM (1K) PRICE INCLUDES:  
- 5 SER (8250 USART)  
- 3 PAR (32 BITS MAX)  
- 32K RAM. EXP 64M  
- STANDARD PC BUS  
- LCD. KBD PORT  
- BATT. BACK. RTC  
- IRQ0-15 (8259 X2)  
- 0237 DMA 8253 TMR  
- BUILT-IN LED DISP.  
- UP TO 8 MEG ROM  
- CMOS NVRAM  
USE TURBO C, BASIC, MASM RUNS DOS AND WINDOWS EVAL KIT \$295

## \$95 SINGLE PIECE PRICE UNIVERSAL PROGRAMMER

-DOES 8 MEG EPROMS  
-CMOS, EE, FLASH, NVRAM  
- EASIER TO USE THAN MOST  
- POWERFUL SCRIPT ABILITY  
- MICROCONT. ADAPTERS  
- PLCC, MINI-DIP ADAPTERS  
- SUPER FAST ALGORITHMS



OTHER PRODUCTS:  
8088 SINGLE BOARD COMPUTER ..... OEM \$27 ... • 95  
PC FLASH/ROM DISKS 128K-16M) ..... 21 ..... 75  
16 BIT 16 CHAN ADC-DAC CARD ..... 55 ..... 195  
WATCHDOG (REBOOTS PC ON HANGUP) ..... 27 ..... 95

• EVAL KITS INCLUDE MANUAL BRACKET AND SOFTWARE.  
5 YR LIMITED WARRANTY  
FREE SHIPPING  
HRS: MON-FRI 10AM-6PM EST

**MVS** MVS BOX 850  
MERRIMACK, NH  
(508) 792 9507

## Contacting Circuit Cellar

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

Mail: Letters to the Editor may be sent to:  
Editor,  
Circuit Cellar *INK*,  
4 Park St.,  
Vernon, CT 06066.

Phone: Direct all subscription inquiries to (800) 269-6301.  
Contact our editorial offices at (860) 875-2199.

Fax: All faxes may be sent to (860) 871-0411.

BBS: All of our editors and regular authors frequent the Circuit Cellar BBS and are available to answer questions. Call (860) 871-1988 with your modem (300-14.4k bps, 8N1).

Internet: Letters to the editor may be sent to [editor@circellar.com](mailto:editor@circellar.com). Send new subscription orders, renewals, and address changes to [subscribe@circellar.com](mailto:subscribe@circellar.com). Be sure to include your complete mailing address and return E-mail address in all correspondence. Author E-mail addresses (when available) may be found at the end of each article. For more information, send E-mail to [info@circellar.com](mailto:info@circellar.com).

WWW: Point your browser to [www.circellar.com](http://www.circellar.com).

# NEW PRODUCT NEWS

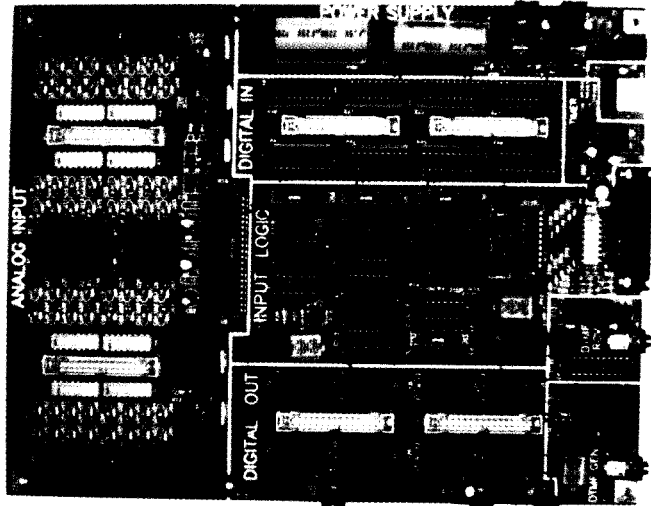
## LOW COST I/O KIT

A low-cost I/O kit, **IO/U**, is available from Take Control. The double-sided PC board supports 64 analog inputs, 64 digital inputs, and 64 digital outputs. As well, it supports a DTMF decoder and generator, IR amplifier, watchdog timer, power supplies, and a high-speed parallel interface that plugs into a bidirectional PC printer port.

Applications include robotics, home automation, weather logging, data acquisition, operator interface, ham repeater/remote base controller, and antenna tracker.

The board features a 12-bit ADC (Maxim's MAX180). Its open collector digital-output relay drivers can sink 150 mA, and all TTL-level digital inputs include pull-up resistors. The unit's modular design enables the user to build just the needed sections. All analog and digital I/O uses 34-pin IDC cables.

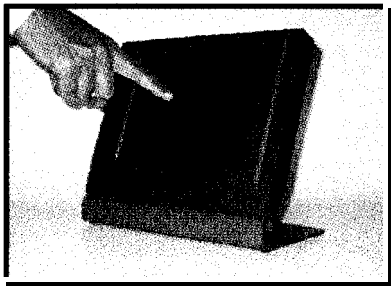
Prices start at \$79 for the bare board, instruction manual, and software (Turbo C source-code and BASIC port-driver examples). A complete kit, including all parts and a wall transformer, is available. Cables, enclosure, shipping, and sales tax are not included.



Take Control, Inc.  
280 Church St. • Clayton, GA 30525-1473  
(706) 782-9848 • Fax: (706) 782-2277  
[www.takecontrol.com](http://www.takecontrol.com)

#502

## Touch The Future



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



**EARTH**

Computer Technologies

Lowest Prices on Earth!

27101 Aliso Creek Rd - #154 - Aliso Viejo - CA - 92656  
Ph: 714-448-9368 - Fax: 714-448-9316  
Email: [oemsales@flat-panel.com](mailto:oemsales@flat-panel.com)  
FREE CATALOG available at <http://www.flat-panel.com>

#105

## Quality Products Shipped Today!

### TraxMaker PCB Design Software

- Create PCB layouts with this virtual reality software!

Part No.	Description	Price
139301	TraxMaker.....	Special \$289.95

### WinBOARD PCB Layout Software

- Choose from over 700 module footprints with surface mounts, or create your own designs

Part No.	Description	Price
137605	WinBOARD PCB CAD ..	\$224.95
137592	WinDRAFT Schem CAD	224.95

### Velleman Computer Interface Board Kit

- More Velleman kits available!

- Pass through parallel connection
- 16 I/Os with opto coupler
- Analog outputs: (8) 6-bit (64 steps), (1) 8-bit (256 steps)
- Analog Inputs: (4) 8-bit (256 steps)

Part No.	Description	Price
128928	Interface brd..	Special \$139.95

### PARALLAX Basic Stamp@ Rev. D Kit

- Additional Parallax products available!

Part No.	Description	Price
140089	Basic Stamp Kit	\$79.95

### 8031 Embedded Applications Proto PC Board

Part No.	Description	Price
119546	Prototype PC board	\$99.95

### AXIOM Low Cost A/D Board

- 16 S.E. analog inputs with 12-bit resolution

Part No.	Product No.	Price
136688	AX5210 .....	Special \$229.95

### E(E)PROM Programmer

- Programs 16Kbits to 512Kbits EPROMs
- Programming speeds/ algorithms: normal, intelligent, and quick pulse

Part No.	Description	Price
101400	E(E)PROM programmer ..	\$129.95

## JAMECO

ELECTRONIC COMPONENTS  
COMPUTER PRODUCTS

©1997 Jameco 6/97

Ordering Hours:  
5:30am - 5:30pm PST

1355 Shoreway Road  
Belmont, CA 94002-4100

FAX: 1-800-237-6948 (Domestic)

FAX: 415-592-2503 (International)

E-mail: [info@jameco.com](mailto:info@jameco.com)

<http://www.jameco.com>



Mention  
V.I.P.# 6C7

Call 1-800-831-4242 to order today!

#106

# NEW PRODUCT NEWS

## INFRARED TRANSCIVER

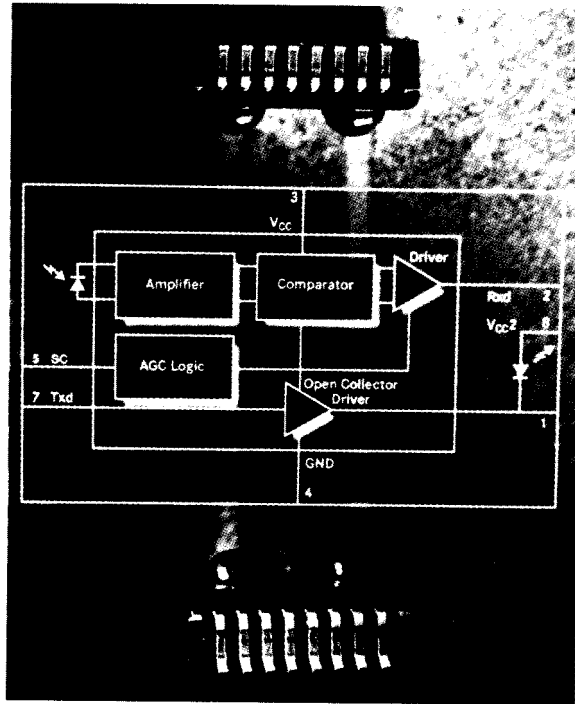
The **TFDT6000** is a multimode integrated IR transceiver module for data-communication systems. The transceiver supports all IrDA speeds up to 4 Mbps, HP-SIR, and Sharp ASK modes. Integrated into this tiny package are a photodiode, IR LED, and analog IC. A current-limiting resistor in series and a  $V_{CC}$  bypass capacitor are the only external components required to implement a complete transceiver.

The transceiver uses a complete differential design for superior interference rejection. It features 5-V operation and low power consumption. By integrating the receiver's preamplifier and the transmitter's driver stage, the TFDT6000 transceiver combines the functions of two ICs and eliminates a large number of external components. A typical discrete implementation requires up to nine separate components.

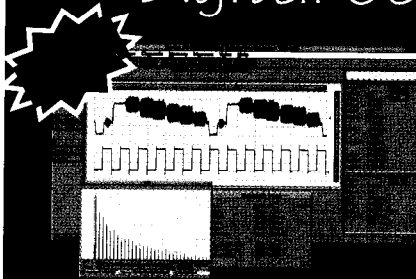
The transceiver is offered in a surface-mount epoxy resin package measuring 0.52" x 0.30" with a height of 0.23". High-volume pricing is \$4.50 each.

**Temec Semiconductors**  
 2201 Laurelwood Rd. • Santa Clara, CA 95054-1 595  
 (408) 567-8220 • Fax: (408) 567-8995

#503



## Digital Oscilloscopes



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

**\$499**

For \$499 you get the model DSO-2042 scope, Probe, Interface cable, AC Adapter and Windows and DOS Software.

## Logic Analyzers



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

Digital DSO (4 Channels) \$1099  
 DSO-2042 (2 Channels) \$499  
 All prices include Probe and software

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

16 Channel Logic Analyzer	\$1799
32 Channel Logic Analyzer	\$2499
64 Channel Logic Analyzer	\$3499
128 Channel Logic Analyzer	\$4999
256 Channel Logic Analyzer	\$7999

All prices include PC and Software  
 All prices include 1 year warranty  
 OEM and Factory Direct options

**Link Instruments (201) 808-8990**

369 Passaic Ave • Suite 100 • Fairfield, NJ 07004 • Fax (201) 808-8786

Web: <http://www.LinkInstruments.com/ci6> Email: [Sales@LinkInstruments.com](mailto:Sales@LinkInstruments.com)

# NEW PRODUCT NEWS

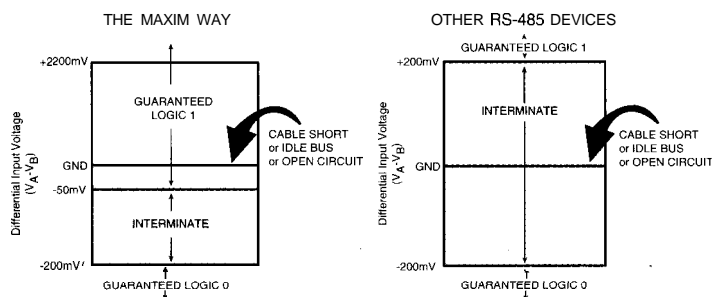
## RS-485/RS-422 TRANSCEIVERS

Each item in the MAX3080-MAX3089 family of high-speed RS-485/RS-422 communications transceivers includes one driver and one receiver. The devices feature fail-safe circuitry, guaranteeing a logic-high receiver output when the receiver inputs are open or shorted. Thus, the receiver output is a logic-high if all transmitters on a terminated bus are disabled (high impedance).

The MAX3080, '81, and '82 feature reduced slew-rate drivers that minimize EMI and reflections caused by improperly terminated cables, enabling error-free data-transmission rates up to 115 kbps. The MAX3083, '84, and '85 offer higher driver output slew-rate limits, allowing transmit speeds up to 500 kbps. The MAX3086, '87, and '88 driver slew rates are unlimited, so transmit speeds up to 10 Mbps are possible. The MAX3089 slew rate can be 115 kbps, 500 kbps, or 10 Mbps by driving a selector pin with a single tristate driver.

All devices have a  $\frac{1}{R}$ -unit-load receiver input impedance that enables up to 256 transceivers on the bus. Driver outputs are short-circuit-current limited and pro-

FAIL-SAFE OUTPUT GUARANTEES LOGIC 1 DURING SHORT OR OPEN CIRCUIT



ected by thermal shutdown circuitry that puts them in a high-impedance state to avoid excessive power dissipation.

The devices come in 8- and 14-pin plastic DIP and SO packages. Prices start at \$1.25 in quantity.

Maxim Integrated Products  
120 San Gabriel Dr.  
Sunnyvale, CA 94086  
(408) 737-7600 • Fax: (408) 737-7194  
[www.maxim-ic.com](http://www.maxim-ic.com)

#504

## OVER/UNDER VOLTAGE PROTECTOR

The "Smart" Protector Type 6 (SPPC-6) PC board controls an offcard solid-state relay to disconnect a load if the AC power-line voltage exceeds programmed limits. The nominal line voltage is set via an oncard DIP switch. High and low voltage limits are proportional to the programmed voltage (i.e., 110-140 V when set for 125-V operation, and 95-125 V with a 110-V line). Power available for the controlled relay is 6 mA max., so a solid-state relay must be used. Load current depends on the relay rating.

A Microchip PIC 16C71 microprocessor, powered by a rechargeable 100-mAh NiCd battery, monitors the AC power-line voltage. If the voltage exceeds limits, the

relay opens and the load disconnects. The circuit automatically resets itself and reconnects the load after 80 s when the line voltage returns within limits. An on-card circuit trickle charges the NiCd battery.

The 8-bit ADC output (proportional to monitored voltage) is broadcast as a serial RS-232 signal to enable display and logging. A two-wire interface is used, and handshaking with the receiver is not needed. Sample MS-DOS software is supplied.

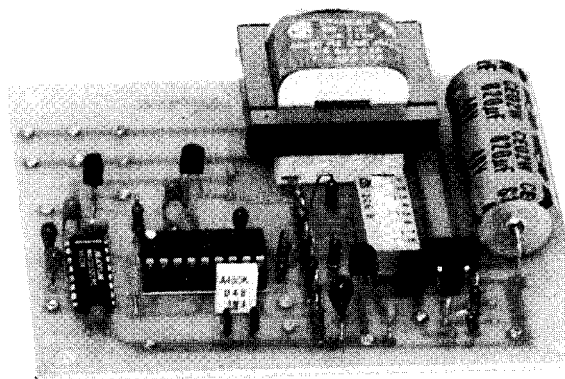
The user can select the Protector response during a power outage. If a DIP switch is off, the microprocessor enters sleep mode to conserve battery power, but it continues to monitor the AC line. When the switch is on, the microprocessor continues to broadcast the voltage (0, in this case) over the RS-232 line. This feature is useful when outage and restore times need to be logged but battery current is -30% higher. When power returns, reset is automatic.

A built-in test circuit simulates an out-of-limits line voltage with a single-pole, normally open push-button switch.

The SPPC-6 sells for \$42.

TDL Electronics  
5260 Cochise Trail • Las Cruces, NM 88012-9736  
(505) 382-8175 • Fax: (505) 382-8810

#505



## FEATURES

12

DSP-Based Canadian Timecode Receiver

20

On- and Off-Hook Caller ID Using DSP

26

PC Telephone Interface

30

Embedding the ARM7500

# DSP-Based Canadian Timecode Receiver

## Part 1: Identifying DSP Techniques

Many people get too caught up in the mathematical details of DSP. To prevent this, David takes a graphical approach. He introduces an app and its high-level design, identifying the necessary DSP techniques involved.

## FEATURE ARTICLE

David Tweed

a

lot has been written recently about digital signal processing, especially since the advent of low-cost general-purpose DSP chips like the Texas Instruments TMS320 series, the Motorola DSP56000, and the Analog Devices ADSP-2101 family. Digital filtering and spectral analysis have been covered as well as high-level application topics such as speech, music, image, and video compression.

But, with the nuts and bolts of finite impulse response (FIR) versus infinite impulse response (IIR) filters, or correlation functions, or discrete Fourier transform (DFT) versus fast Fourier transform (FFT), many people get lost in the details and mathematics.

In this two-part series, I want you to gain a more intuitive feel for these topics. So, I skip (most of) the math, and concepts are presented graphically. I also discuss the practical tradeoffs associated with using these techniques in a real application.

Part 1 introduces the application and walks through the high-level design to identify the necessary DSP techniques. I examine two techniques—cross-correlation and FIR filtering—in detail.

In Part 2, I discuss the Fourier Transform and real-world issues that arise



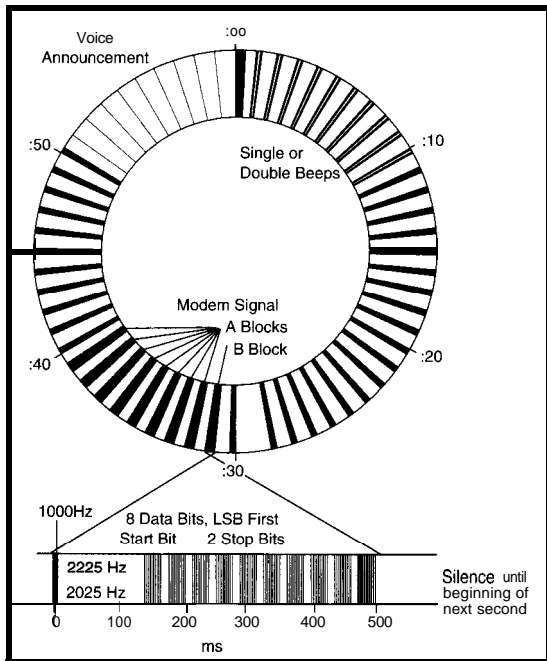


Figure 1—CHU's signal repeats each minute. Seconds :31–:39 contain a Bell 103-compatible FSK modem signal between the second ticks.

when signals don't resemble textbook examples. To wrap up, I show how to use direct digital synthesis to create a timebase independent of the CPU clock.

## THE APPLICATION

It's fairly well-known that station WWV in Boulder, Colorado (and WWVH in Hawaii) broadcasts time signals that can be received over most of North America. These signals contain components that can be decoded with relatively simple hardware to keep a clock synchronized to the international Universal Coordinated Time (UTC).

For some years, Heath offered a kit—the Model GC1000—that took advantage of this. Unfortunately, in New England, WWV's signals are weak and fading at best. Plus, they're often non-existent for large segments of the day.

It's less well-known that Ottawa, Canada's CHU broadcasts a similar time signal that covers New England fairly well. It also can be decoded to automatically set a clock.

This signal's structure is quite different from those of WWV and WWVH. So, other techniques are necessary to extract the relevant information.

I designed a software-based CHU time-signal decoder that runs on a common DSP development board. It uses an ordinary shortwave receiver's audio output and produces an RS-232

ASCII output to set and/or display the time.

While this application is a little contrived, it's a good base for discussing DSP. And, it demonstrates how far we can push the performance envelope in terms of accuracy and tolerance to noise and fading.

## CHU SIGNAL

CHU's time signal can be found on 3.330, 7.335, and 14.670 MHz. It's an AM-compatible full-carrier single-sideband signal, containing 1000-Hz beeps, voice announcements, and a 300-bps modem signal. Figure 1 shows how the components fit together.

As you can see, the heavy lines of the figure represent the 1000-Hz tone. It comes in 500 ms at the top of the minute, 300 ms or double tones as indicated, and 10-ms ticks when a voice announcement or modem signal is needed.

The announcement alternates between the station ID and time in English followed by the time in French (on even minutes) and the station ID and time in French followed by the time in English (on odd minutes).

At the top of each hour, the :00 tone is extra long, and there is no tone for seconds :01–:09. The 300-bps modem signal shown at the bottom of Figure 1 is Bell 103 compatible, using 2225 Hz for mark and 2025 Hz for space.

Each data burst begins immediately after the 10-ms tick with 123.3 ms of 2225 Hz, representing a binary 1 or idle state. It's followed

by ten 8-bit bytes of data, each framed with a start bit of 0 and two stop bits of 1.

With either of the two types of data blocks (A or B), the data with its start and stop bits requires 110 bit times (i.e., 366.7 ms) to trans-

mit. The last stop bit ends exactly 500 ms into the second and is followed by another 10 ms of 2225 Hz to avoid false overrun of the stop bits. The remainder of each second is silent.

Each data block contains 5 bytes of data (divided into ten 4-bit nibbles), followed by 5 redundancy bytes. The A-format redundancy bytes are exactly like the data bytes. The B-format redundancy bytes are exactly inverted (1's complement, NOT, XOR 0xFF, etc.) from the data bytes.

Figure 2 shows the two types of blocks as received by a CPU. Once the data is in memory and the redundancy bytes checked, swap the least and most significant nibbles in each byte.

In the A block, the 6 is a constant, DDD is the day of the year, and hh:mm:ss is the UTC time of day (at the beginning of the current second). Each nibble is a BCD digit.

In the B block, X is a bitwise field, and D is the absolute value of DUT in tenths of a second. YYYY is the Gregorian year, and TT is the difference between TAI and UTC.

The A nibble flags Canadian Daylight Time (this nibble's contents are currently undocumented). The B nibble is a serial number that increments when the B-block format changes.

A B block transmits once per minute, at second :3 1. An A block transmits during seconds :32–:39.

DUT is a signed number representing the difference between UTC (atomic time) and UT1 (astronomical time). It varies in a complex way because of slight variations in the earth's rotation rate. When it reaches  $\pm 0.7$  s, a leap second is added to or deleted from UTC, usually at the next new year.

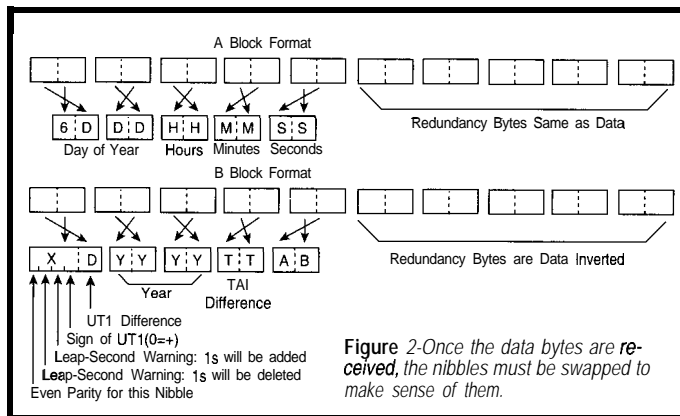


Figure 2—Once the data bytes are received, the nibbles must be swapped to make sense of them.

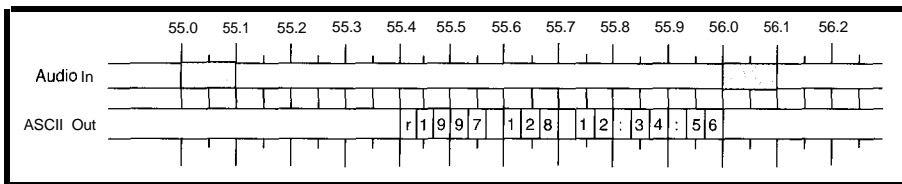


Figure 3—The ASCII output of the receiver ends just as the corresponding second begins.

## A BETTER MOUSETRAP

Suppose you want to build a clock that sets itself to the CHU signal like the Heathkit clock does to the WWV signal. And, you want to see how precise you can get this signal.

The Heathkit clock guarantees 10-ms accuracy when its Hi Spec light is on, but I think submillisecond accuracy is possible. I want a lot of information out of the audio signal despite its noisiness.

Under most conditions, CHU offers a stronger signal than WWV to New England. However, it's still subject to severe fading.

The clock should provide continuous output regardless of the radio signal's condition, while keeping the best possible accuracy. That's why I didn't just use a \$15 modem.

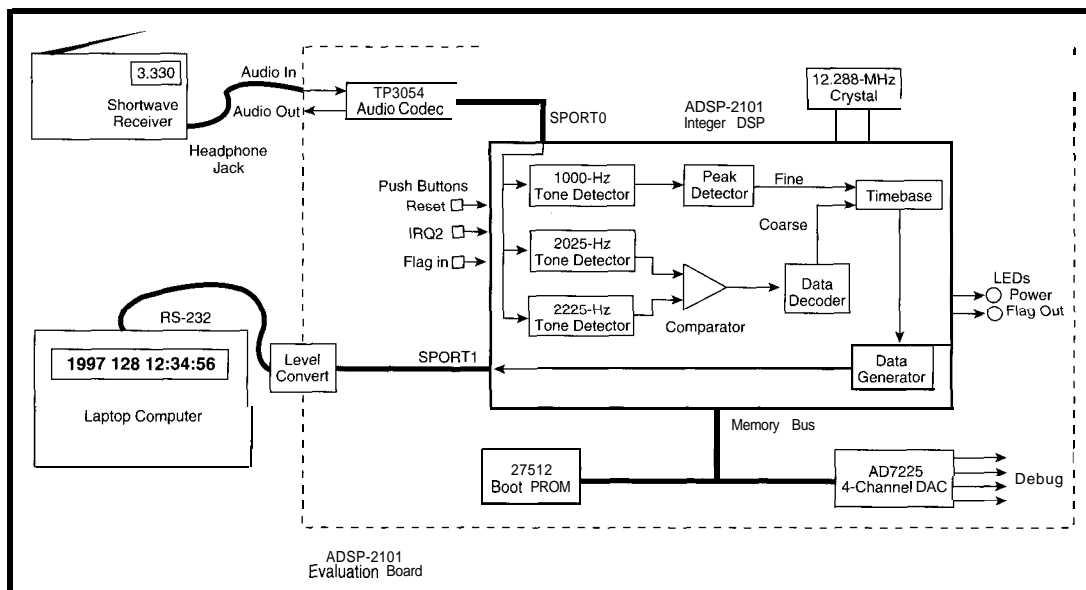
## FUNCTIONAL SPECIFICATION

I wanted to generate an RS-232 output that gives the time of day as an ASCII string every second, based on the signal received from CHU.

Since the signal isn't always available, the product needs a local timebase. I wanted to avoid RF, so I used the audio output of a shortwave receiver.

The audio input is from the headphone jack of a general-coverage shortwave receiver, which gives a 1-V<sub>RMS</sub> signal. The DSP evaluation board's audio input should accept this directly.

Figure 4—The complete time receiver has a radio, the DSP eval board, and a computer or terminal to display the time. The ADSP-2101 EZ-Lab eval board includes the DSP, a boot PROM, a voice-grade audio codec, and a four-channel DAC. The DSP outline offers a tentative software-dataflow diagram.



## RS-232 OUTPUT

The output signal is RS-232C, using ASCII characters in an 8N1 configuration. The data rate would range between 300 and 9600 bps.

Using C print f () notation, the output string is:

```
\r%4d %3d %2d:%02d:%02d
```

where the individual fields are year, day, hours, minutes, and seconds UTC (using 24-h notation). \r represents a bare CR.

When observed on a screen or emulator, the time display updates in place onscreen, leaving the cursor at the end of the string between updates.

This string has a fixed length of 18 bytes and is transmitted so the last byte ends at the time represented by the string (see Figure 3). The screen appears in sync with the audio, but I started transmitting the string 18 character times before the represented time.

## ACCURACY

The local timebase should be as accurate as possible within the limits imposed by the radio link and receiver. The 1000-Hz tones give a basic 1-pps

(pulse per second) indication. Depending on how accurately I identify the tones' start and stop transitions, I can set the local timebase to within a few milliseconds.

By discriminating individual cycles of the 1000 Hz, I can get it to around 1 ms. And, if I can accurately measure the tone's relative phase angle, I might get 0.1-ms or less error.

However, the radio-path length between Ottawa and eastern Massachusetts is ~700 km. And, it can vary by ~10% as the ionosphere varies in height and reflectivity.

At 300,000 km/s, the path delay is  $-2.2 \pm 0.2$  ms. So, the accuracy goal should be ~1.0-ms maximum instantaneous error.

## TOP-DOWN DESIGN

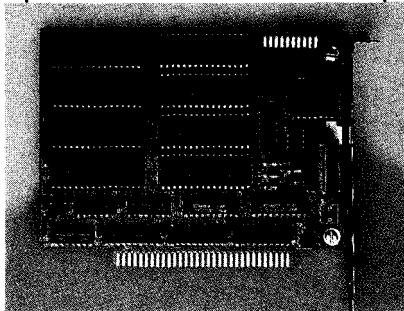
Once the product's task is set, consider which technologies to use.

I need to decode audio tones at 1000, 2025, and 2225 Hz. I also need a local timebase to generate ASCII output messages which synchronizes with CHU's signal when it is available.

While analog filters along with PLL (phase-locked loop) circuits handle tone decoding and the local timebase, they are rather inflexible for trying different algorithms or if the functional requirements change. Also, getting everything to work together optimally is a complex calibration process.

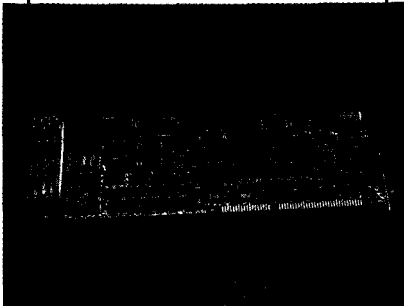
To demonstrate DSP techniques with an off-the-shelf evaluation board, I chose an all-software implementation.

**VMAX®**  
**QUALITY PRODUCTS**  
**RESPONSIBLE SERVICE**  
**RELIABLE DELIVERY**



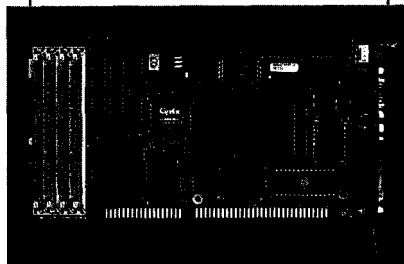
**ENHANCED SOLID STATE DRIVE — \$105\***

4M Total, Either Drive Bootable  
 1/2 Card 2 Disk Emulator  
 Flash System Software Included  
 FLASH & SRAM Customs too



**486 SLAVE PC — \$898\***

Add up to 4 Boards to One Host PC  
 Fast Data Transfer and I/O  
 PC-104 Port, IDE & Floppy Control  
 Independent Processors on One Bus  
 No Special Compilers Needed



**486 66MHz SINGLE CARD COMPUTER — \$335\***

Up to 2.5MegFlash/Sram drive  
 Compact-XT height 1/2 card size  
 Industry Standard PC-104 port  
 L2 cache to 64K—DRAM to 16Meg  
 Dual IDE/Floppy connectors

All Tempustech VMAX® products are  
 PC Bus Compatible. Made in the  
 U.S.A., 30 Day Money Back Guarantee  
 \*Qty 1, Qty breaks start at 5 pieces.

**TEMPUSTECH, INC.**

TEL: (800) 634-0701

FAX: (941) 643-4981

E-Mail: [cpusales@tempustech.com](mailto:cpusales@tempustech.com)

I-Net: [www.tempustech.com](http://www.tempustech.com)

Fax for 295 Airport Road  
 fast response! Naples, FL 34104

Figure 4 shows the complete hardware of the time receiver. It comprises a Realistic DX-380 receiver, Analog Devices' EZ-Lab board for the ADSP-2101, and a TRS-80 Model 100 laptop.

Within the dotted line is the block diagram of the DSP evaluation board. It includes an audio codec for the A/D conversion. An RS-232 level converter on serial port 1 (SPORT1) generates the correct voltages for the output signal.

The four-channel DAC connects to an oscilloscope for algorithm development and debugging. There, it graphically indicates the DSP's real-time activity.

The software takes in 8000 audio samples per second—more than sufficient to handle the bandwidth. It generates ASCII output messages as well.

In between, it detects tones and decodes CHU signal's data. Using this information, it establishes a local timebase relative to the CPU's crystal. The timebase then drives the output-message generator.

Figure 4 illustrates the required components and how they interact. I fully develop this diagram after discussing possible techniques for tone detection and establishing a timebase.

**TONE DETECTION**

From this diagram, you see that tone detection plays a major role in the receiver's operation. Because I want high accuracy, it's important to determine the existence or nonexistence of tones and to find when they begin and end—down to a single cycle or less.

Many people believe this is what FFTs are for. But, the FFT is most useful when you're looking for one or more tones but don't know their frequency. It's overkill when looking for a tone at a particular frequency, and it isn't particularly good at locating a tone's start and stop edges.

A Fourier Transform (FT) converts a block of numbers representing signal samples in time into the signal's frequency components for that period. It can't tell you whether a given component was there for the whole block of time or only part of it.

The best you can do is see whether the component is present in one block but not another. This limits time resolution to the FFT's block size.

So, you use small sample blocks to get good resolution. But, there's a tradeoff. The number of frequency bins at the FFT's output is proportional to the number of time samples at the input.

For a given sample rate, each bin's size grows as the number of bins goes down, so it's harder to discriminate among frequencies that are close together. Thus, you need large sample blocks to get good frequency resolution.

Obviously, you can't have good time and frequency resolutions simultaneously with an ordinary FFT. A different operation—cross-correlation—helps locate signals in time.

**CROSS-CORRELATION**

Suppose you have two signals. One is a template for a simple 2-cycle tone

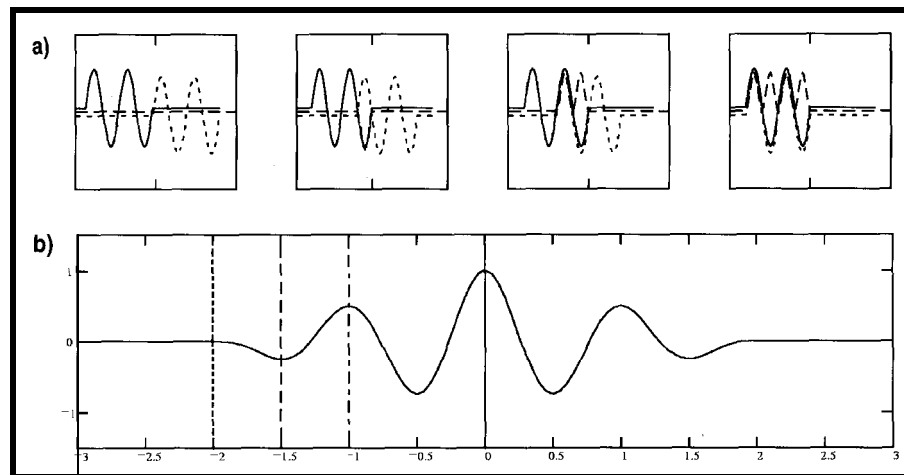
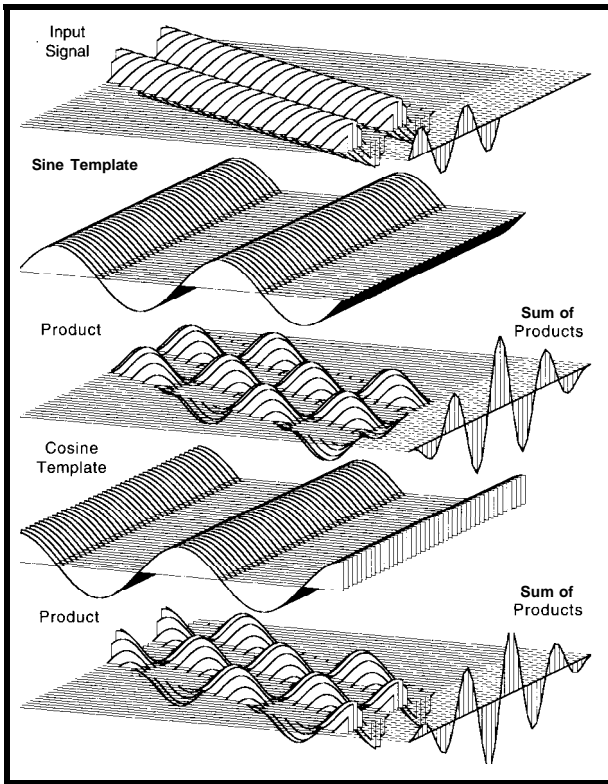


Figure 5a—These graphs show trial alignments of an incoming signal (dotted line) with a template (solid line). The dashed line shows the point-by-point multiplication of the two functions. Integrating this line over time yields a single point in the overall cross-correlation function. b—This graph includes markers for the four trial alignments.



**Figure 6**—Repeated cross-correlation trials can be represented as a three-dimensional structure. Redoing the process using a cosine template enables the extraction of phase-angle information.

burst. Does the other signal contain this tone burst?

The signals are both functions of time. So, line the template up with the unknown signal at various offsets in time to see how they match up.

Figure 5a shows several such trials. The solid line represents the template function. The incoming signal is shown as a dotted line at various offsets ( $A_t$ ).

The matching is done via a point-by-point multiplication of the two function values. See the result in the dashed line.

Note, when either function is zero, the result is zero. If both functions are positive or both are negative, the result is positive. If the signs are opposite, the result is negative.

I boil this down to a single number for each trial by adding up (integrating) the individual multiplication results. If the result is zero or near zero, the signals are uncorrelated. If the result is negative, there is negative cor-

relation between the signals, and if the result is positive, there is positive correlation.

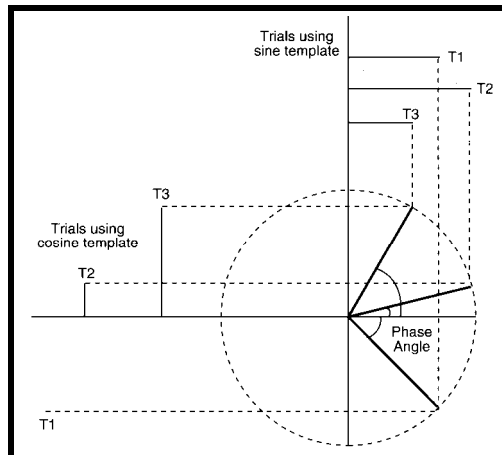
By making many trials at various values of  $A_t$  and generating a correlation value for each, I can graph these values as a function of  $A_t$ . Figure 5b has vertical markers showing values for the trial alignments. The fourth trial in Figure 5a shows perfect alignment at a  $A_t$  value of 0, corresponding to the highest peak in Figure 5b.

Figure 6 shows this process differently. Here,  $t$  (time) varies left to right, and  $A_t$  from front to back. The top section gives the input signal, shifting left to right as  $A_t$  varies.

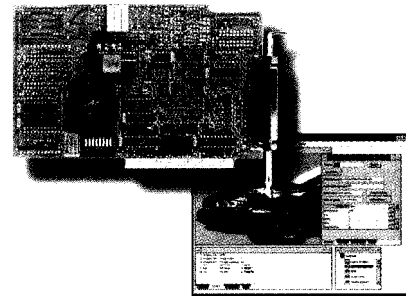
The second section shows the template function, which doesn't change with  $A_t$ . The middle section represents the point-by-point multiplication of the first two sections. Each layer is a different trial alignment of the input signal with the template.

Integrating the middle section left to right (i.e., over time) gives a single value for each trial, representing the value of the cross-correlation. Together, these points represent the cross-correlation function of  $A_t$ .

In effect, this integration "projects" the surface onto the two-dimensional graph shown running front to back at



**Figure 7**—Combining the results from the sine and cosine analyses allows the phase angle difference to be calculated at each trial.



## Lowest Cost Data Acquisition

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

Just check out the specs:

### Lowest Cost

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

**\$195**

### High Speed

**5508LC**  
8 channels 12-bit A/D,  
100KHz, DMA

**\$245**

### Multi-Function DMA

**5516DMA**  
16 channels 12-bit A/D,  
DMA, 16 digital I/O

**\$295**

### High Resolution

**5500HR**  
16 channels 16-bit A/D,  
DMA, 8 digital I/O

**\$595**

learn more:

voice 800-648-6589

fax 617-938-6553

web [www.adac.com](http://www.adac.com)

email [info@adac.com](mailto:info@adac.com)

**ADAC**

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

the right. A single trial alignment is represented as a slice parallel to the paper surface, and it represents a single value on the final graph.

## DISCRETE TIME

I've been cheating a bit. In these graphs, I pretended the template and input functions are continuous with respect to time. Actually, they're sequences of numbers representing samples of the continuous functions.

Therefore, I can't arbitrarily make many trial alignments between the template and input functions. The best I can do is generate one cross-correlation value for each input sample processed.

I have a second, related problem. Since the clock taking samples isn't synchronized to the clock generating the signal at the transmitter, I can't count on a sample occurring at the peak of the cross-correlation function.

However, I can compensate for these issues. Consider what happens if I take a second cross-correlation using a cosine wave as the template, which is another way of describing a sine wave shifted by 90° (a quarter of a wavelength).

The bottom sections of Figure 6 show the same analysis with a cosine template. I can take the results from

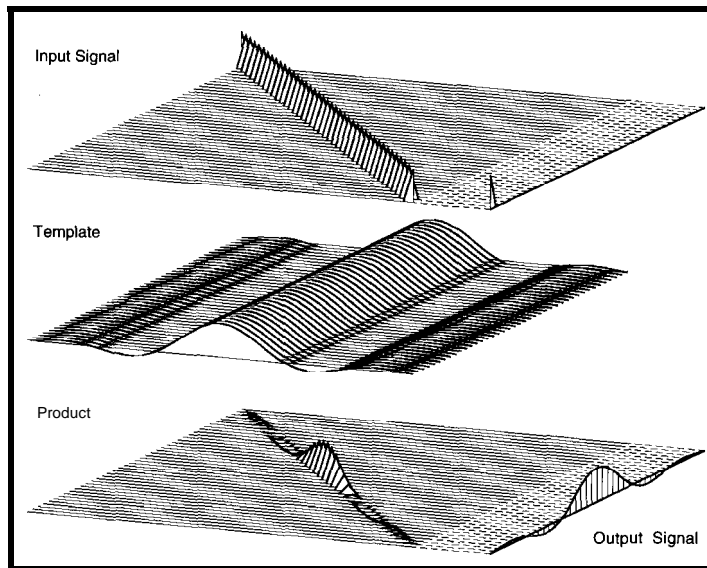


Figure 8—An input signal consisting of a single nonzero sample reads out the template function (FIR coefficients) in sequence.

near the centers of both analyses and plot the result of the sine correlation against the result of the cosine correlation (see Figure 7).

The resulting points lie on a circle. If I draw a line from each point to the circle's center, its angle relative to the x-axis represents the phase angle of the input signal with respect to the cosine template.

I get a numerical value for this phase angle by taking the arctangent of the ratio of the two results. This value can have any resolution and may represent fractions of the sample period.

So, if the true peak of the cross-correlation function falls between two actual samples, I get a negative phase angle as part of the answer for the first calculation but a positive phase angle

at the next calculation. It's easy to do a linear interpolation between these two angles to calculate the exact moment the phase angle went through 0.

## FIR FILTER

Now for something completely different. The Finite Impulse Response (FIR) filter is an algorithm commonly used on DSPs because of its predictable characteristics and nice, regular structure.

Treated as a black box, it takes in a sequence of numbers representing a signal's samples and out-

puts a new sequence of numbers representing the filtered version of the input signal.

Internally, the FIR filter is implemented as a series of registers that hold the input sample and copies of previous input samples. As each sample arrives, the oldest sample is discarded.

The whole set of samples is multiplied by a set of numbers (the filter's coefficients), the products are summed, and this sum becomes the current output sample. This process repeats at the sample rate.

The filter's coefficients are the same as its impulse response. Consider what happens if all the registers contain 0 and a sample of 1 arrives, followed by more 0 samples, which is the discrete-time version of an impulse function.

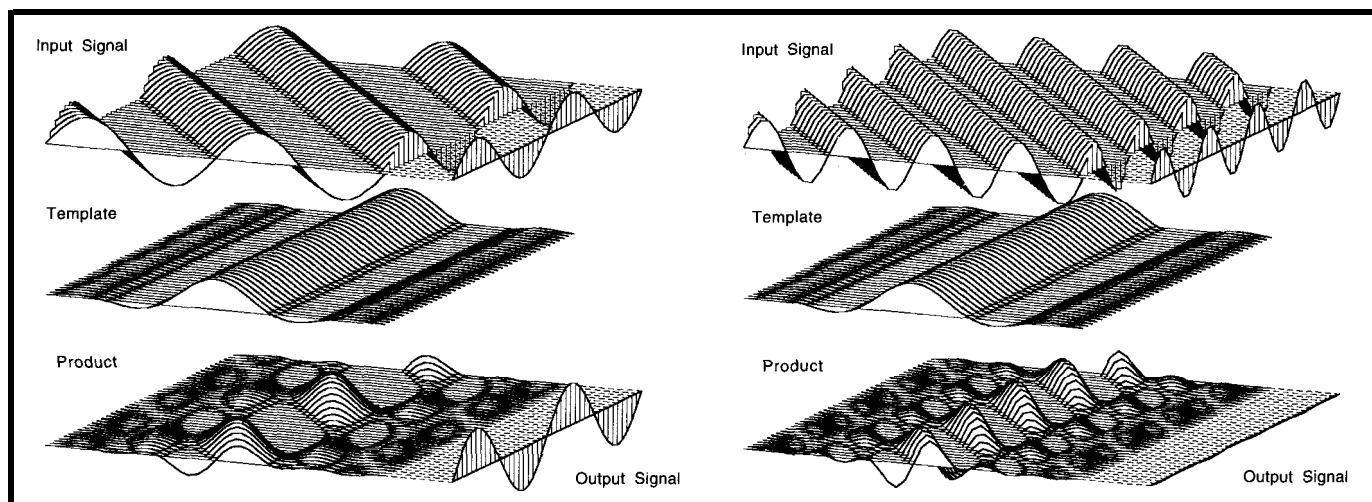


Figure 9—The low-pass FIR filter has a large output for a signal below its cutoff frequency but a tiny output for a signal above it.



As the 1 propagates through the registers, it is multiplied once by each coefficient in sequence. All other coefficients are multiplied by 0. The sequence of output samples, representing the filter's response to the impulse stimulus, matches the sequence of coefficients exactly.

As Figure 8 shows, the only difference between the FIR filter and cross-correlation function is terminology. "Template function" is now "impulse response" or "filter coefficients." And, what I called "At" is now the registers holding the older input samples.

In effect, the output of the FIR filter is a signal that, from moment to moment, tells how well the input sample matches or correlates with the impulse response. Therefore, you'll sometimes see the term "matched filter" used in certain signal-processing applications.

The coefficients in Figure 8 implement a low-pass filter. Figure 9 shows what's going on within the filter for signals both below and above the cut-off frequency.

For the signal above the cutoff frequency, the outcome of the multiplication step has nearly equal amounts of positive and negative results, giving almost total cancellation and a very small output signal.

It isn't obvious why this set of coefficients implements a low-pass filter. The math shows that the frequency response is the FT of the impulse response.

## UPCOMING

In Part 2, I return to the FT and look at building a local copy of the UTC timebase. I also cover the details of implementing the algorithms discussed.

One set of software tone detectors demodulates the FSK data to coarsely set the timebase, and another fine-tunes the setting based on a 1000-Hz burst. ☐

*David Tweed has been developing real-time software for microprocessors for more than 18 years, starting with the 8008 in 1976. He currently designs equipment to carry high-quality audio and wide-bandwidth data over digital telephone services such as T1 and ISDN. You may reach him at dave.tweed@circellar.com.*

## SOFTWARE

MathCad graphics for this article are on the Circuit Cellar Web site.

## REFERENCES

Radio station CHU, [www.nrc.ca/inms/whatetime.html](http://www.nrc.ca/inms/whatetime.html).  
Radio station WWV/WWVH, [www.boulder.nist.gov/timefreq](http://www.boulder.nist.gov/timefreq).  
D.L. Mills, **Gadget Box PPS Level Converter and CHU Modem**, [www.sibson.com/SysAdmin/ntpdoc/gadget.html](http://www.sibson.com/SysAdmin/ntpdoc/gadget.html).

## SOURCES

**MathCAD**  
MathSoft, Inc.  
101 Main St.  
Cambridge, MA 02142-1521  
(617) 577-1017  
Fax: (617) 577-8829  
[www.mathsoft.com](http://www.mathsoft.com)  
TRS-80 Model 100  
Andy Diller's Web 100 Main Page  
[www.clark.net/pub/m100/pages/comp/m100/m100Main.html](http://www.clark.net/pub/m100/pages/comp/m100/m100Main.html)

**ADSP-2101**, ADSP-2181, EZ-Lab, EZ-Lab Lite  
Analog Devices  
One Technology Way  
Norwood, MA 02062-9 106  
(617) 329-4700  
Fax: (617) 329-1241  
[www.analog.com](http://www.analog.com)

DSP56000  
Motorola  
MS OE314  
6501 William Cannon Dr. W  
Austin, TX 78735-8598  
(512) 891-2030  
Fax: (512) 891-3877  
[www.mot.com/SPS/DSP/products](http://www.mot.com/SPS/DSP/products)

TMS320 series  
Texas Instruments, Inc.  
34 Forest St., MS 14-01  
Attleboro, MA 02703  
(508) 699-5269  
Fax: (508) 699-5200  
[www.ti.com](http://www.ti.com)

401 Very Useful  
402 Moderately Useful  
403 Not Useful

## If you need I/O...

### Add these numbers up:

**80C552** a '51 Compatible Micro  
40 Bits of Digital I/O  
8 Channels of 10 Bit A/D  
3 Serial Ports (RS-232 or 422/485)  
2 Pulse Width Modulation Outputs  
6 Capture/Compare Inputs  
1 Real Time Clock  
64K bytes Static RAM  
1+ UVPRAM Socket  
5 12 bytes of Serial EEPROM  
1 Watchdog  
1 Power Fail Interrupt  
1 On-Board Power Regulation

### It adds up to real I/O power!

That's our popular OEM **552SBC-40**, priced at just \$299 in single quantities. Not enough I/O? There is an expansion bus, too! Too much I/O? We'll create a version just for your needs, and pass the savings on to you! Development is easy, using our Development Board: The **552SBC-50** Development board with ROM Monitor for \$349.

### New Versions of the 8031SBC

Our popular 803 1 SBC can now be shipped with your favorite 8051 family processor. Models include **80C5 1 FA**, **DS80C320**, **80C550**, **80C652**, **80C154**, **80C851** and more. Call for pricing today!

### Truly Low-cost In-Circuit Emulator

The DryICE Plus is a low-cost alternative to conventional ICE products. Load, single step, interrogate, disasm, execute to breakpoint. Total price for the base unit with most pods is a low \$448. Call for brochure, or **World Wide Web** at [www.hte.com](http://www.hte.com).

Call for your custom product needs. Quick Response.

**HTE** HiTech Equipment Corp.  
9672 Via Excelencia  
San Diego, CA 92126  
[Fax: (619) 530-1458]

Since 1983

(619) 566-1 892



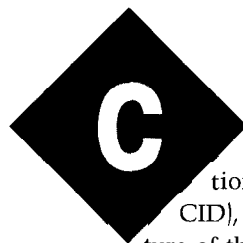
Internet e-mail: [info@hte.com](mailto:info@hte.com)  
World Wide Web: [www.hte.com](http://www.hte.com)

# FEATURE ARTICLE

Dave Ryan  
& Asher Hazanchuk

## On- and Off-Hook Caller ID Using DSP

Phone companies have already made on-hook caller ID popular. But, you can also quite easily gain this feature off-hook. After reviewing caller ID basics, Dave and Asher show us how to implement it using DSP.



aller identification, (caller ID or CID), is an added feature of the telephone system that visually indicates who is calling. The display, usually a custom LCD with 2-4 lines of information, might look like:

```
08:16AM    8/18    Call#1
408-370-8504    Dave Ryan
```

Before picking up the phone, you can identify the caller. Unwanted calls can therefore be filtered out.

Two primary messaging formats are supported. SDMF (single data message format) displays the date, time, and phone number of the calling party. In addition to displaying these features, MDMF (multiple data message format) gives the caller's name as it appears in the telephone book. This information arrives via two methods of delivery—on- or off-hook.

On-hook delivery transmits information between the first and second rings of the incoming call. This method is widely implemented in analog systems and is commercially available.

Off-hook delivery, is also called SCWID [spontaneous call waiting with caller ID] or CIDCW (caller ID with call waiting). When a third party tries to connect with two parties already engaged with each other, information is only transmitted if an acknowledgment is received from the party to be interrupted. This method is not commercially available.

In addition to the various call-waiting signals transmitted from the SPCS (stored program control system), a special CAS (customer premises equipment alerting signal) is also sent. The basic data is transmitted using FSK (continuous phase binary frequency shift keying).

### ON-HOOK DELIVERY

This fairly simple system only requires circuitry to detect the ringing current, demodulate the FSK signal,

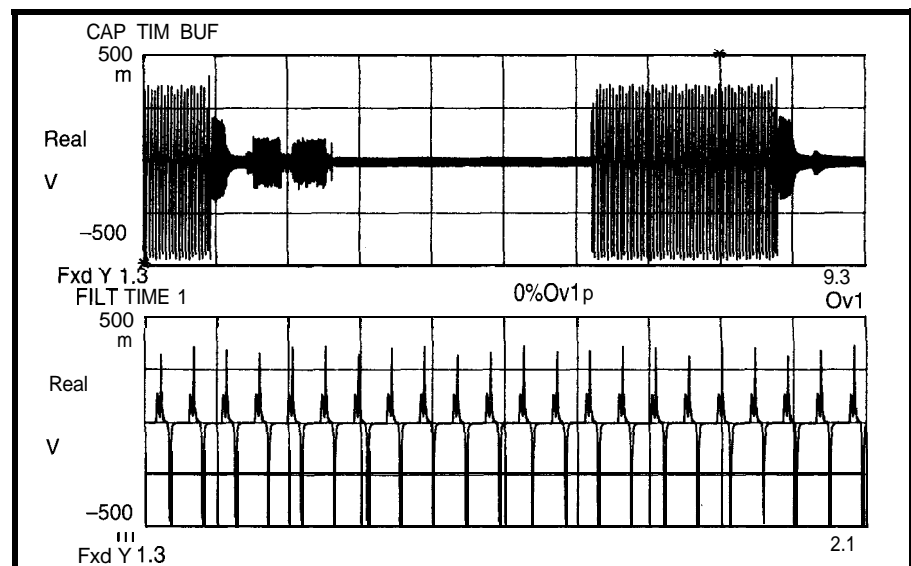


Figure 1—The high-amplitude, low-frequency signal is the ringing voltage. The FSK data-transmission signal is the short burst of low-amplitude, high-frequency signal that appears between the first and second rings.

and display the resulting data.

Figure 1 shows the delivery of FSK data sandwiched between the first and second rings. The larger amplitude, lower frequency waveforms at the beginning and end of CAP TIM BUF (Captured Time Buffer) are the ringing pulses.

FILT TIME 1 (Filtered Time 1) shows the ringing pulses in greater detail. The smaller amplitude, higher frequency waveform is the FSK data.

A somewhat idealized simulation of the data is shown in Figure 2a. The data alternates between 1, 0, 1, and 0. The power spectral density plot shows this signal's frequency content in the frequency domain.

Of course, real-world data is never as clean idealized situation. Figure 2b shows actual received data.

It's easy to see that the amplitudes of the high and low-frequency segments are quite different. In addition, noise is superimposed on the signal, most noticeably on the peaks and troughs.

### SDMF AND MDMF

Although SDMF displays only the date, time, and phone number, MDMF give the caller's name as well. In fact, via MDMF, any ASCII data may be transmitted.

Figure 3 shows a simplified overview of the MDMF. The channel seizure is a series of alternating 1s and 0s that are only supplied in the on-hook case. Off-hook as, data transmission starts with the mark signal, which is a series of 1s.

Parameter words are not limited to one message. There may be many parameter messages, each consisting of a parameter type, length, and word.

Just to complicate matters, optional mark signals may be sent between frames. At the end of every transmission is a checksum we describe in detail later. Notice that the parameter-data length can vary.

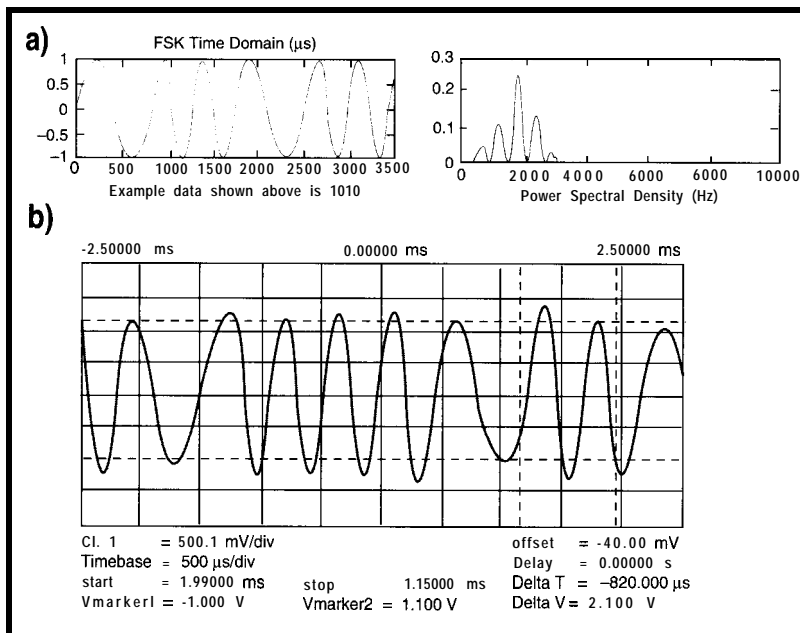


Figure 2a—A Matlab simulation shows idealized FSK and its corresponding power spectral density in the frequency domain. b—An actual FSK transmission caught using a storage scope shows some of the impairments that occur in the real world (e.g., over- and undershoots).

Figure 4 illustrates an on-hook solution. An FSK band-pass filter filters signals, and the FSK demodulator converts the analog signal into binary data.

The SDMF/MDMF section removes the start and stop bits and determines the messaging format. Data is stored in SRAM or displayed on the LCD.

### LCD

The display is usually a small LCD capable of showing the caller's date, time, telephone number, and name. It usually has enough memory to store 30-99 calls.

The system is usually battery powered since the time of system operation is generally limited to the time between the first and second ring. Once the call is answered, the system may be put in power-down or standby mode.

### WHY DSP?

Digital signal processing isn't necessary for on-hook operation. Relatively simple and cost-contained analog solutions exist. DSP makes much more

sense for off-hook operation.

The difficulty arises in accurately detecting the special CAS tone in the presence of VOX. The chip must avoid inadvertent detection due to the similarity with speech (the Talk Off problem).

This type of system hasn't been widely implemented in analog solutions primarily because implementing a cost-contained, manufacturable, and robust solution is difficult.

With digital filters, the manufacturing

difficulties associated with using critically matched components [e.g., resistors, capacitors, inductors, etc.] are largely avoided. In addition, the solution may now be made adaptive.

Variant implementations then become simply a matter of software upgrades. Of course, there are tradeoffs. A/D conversion must be supported with its ancillary requirements, and so must D/A conversion. However, usually, a DSP solution seems far superior.

### BUILDING A CALLER-ID SYSTEM

The simplest way to get caller ID is to purchase a ready-made evaluation board complete with firmware. However, it's certainly possible to write the software and build the hardware.

While building the hardware is reasonably straightforward, software development is a little more complex. You'll definitely need some firmware development tools (e.g., an emulator, assembler, linker, and debugger).

You can see the system in Figures 5 and 6. The system blocks for it are:

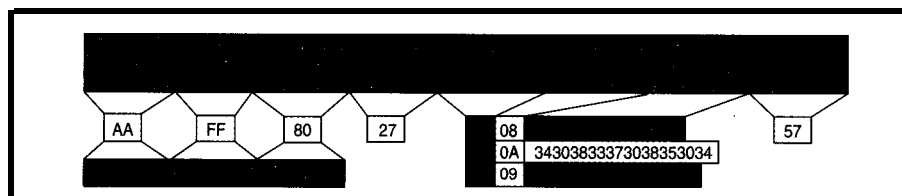


Figure 3—Here, you see the digital data overview as well as its relation to the overall messaging or packet structure.

- phone-line interface-includes the transformer and components that isolate the caller-ID circuits from the line (protects against damage from the ring high-voltage signal) and the on-/off-hook relay
- ring-detect circuit-gives digital input (RING\_DET signal) to the DSP to indicate rings on the phone line
- caller-ID gain control-controls signal gain coming from the phone-line interface to the codec analog input. The DSP enables this path by the CID\_CNT control signal.
- codec-acts as the DSP analog front end. The codec data format is 8-bit PCM  $\mu$ law. The DSP controls the sampling rate by the FS1 signal and serial shifts by SCLK signal. The DSP receives serial data from RXD and transmits serial data to TXD.
- hybrid-The DSP sends the CAS acknowledge via the codec and the hybrid back to the phone-line interface. The DSP enables this path by using the HO\_CNT control signal.

The operation is relatively simple and should be possible anywhere you can subscribe to caller ID.

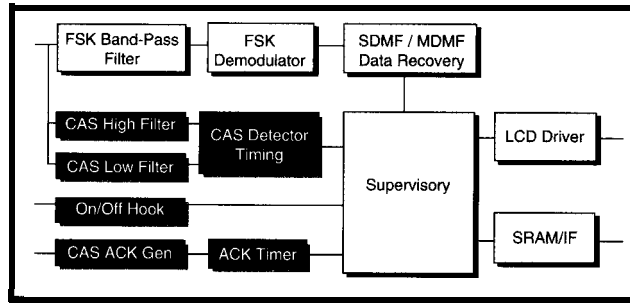


Figure 4—These are the primary software functioning blocks with their rough interconnection. Black areas illustrate the sections relevant to off-hook connection.

After power is applied and the reset button pushed, the LCD should display “ready”. Check FSK levels on TP3. The FSK signal’s amplitude should be  $-3 V_{pp}$  when the FSK data is being received between the first and second ring. Adjust R3 1, if necessary. Then, read the LCD for the information.

### OFF-HOOK DATA

Figure 7 shows the delivery of FSK data in the off-hook mode.

The larger amplitude, lower frequency waveforms at the beginning of CAP TIM BUF are the call-waiting and CAS tones. After a gap, the FSK data is seen.

During this gap, the DSP generates by an ACK. This ACK is not shown, as the DSO was connected to the receive side. FILT TIME shows the call waiting and CAS tone in greater detail.

Comparing the on- and off-hook sections of Figure 4, we see many differences. In addition to the modules used in on-hook selection, there’s a CAS filter for the high- and low- band portions of the CAS signal, and special CAS detector timing.

Once the CAS tone is detected, an acknowledgment must be returned via a DTMF

generator. It’s also necessary to determine if the system is on or off hook.

Operation in the off-hook mode is not as simple, due to the extra communication involved.

Connect P2 to line 2 or a CIDCW simulator, if available. A simulator is mandatory for additional development, since it involves a least 3 lines.

Again, a scope TP3 to check FSK levels. On line 2, you should hear a call-waiting tone followed by the special CAS tone. If all is in order, the module detects the tone, ACK is sent, and the SPCS or simulator transmits data.

### FSK DEMODULATION

A software FSK demodulation function is integrated into the DSP as you see in Figure 4. The FSK frequencies are  $1200 \pm 12$  and  $2200 \pm 22$  Hz. After all

the protocol and hand-shaking complete, the data is sent using FSK.

This means there are no phase discontinuities and only two frequencies involved in the FSK signal. The lower frequency 1200 Hz represents a mark (logic 1), and the higher 2200 Hz represents a space (logic 0).

There is no parity or error checking beyond checking a checksum sent at the end of transmission. A start bit (0) and a stop bit (1) added to each 8-bit transmitted word.

The transmission rate is 1200 bps and demodulation is simi-

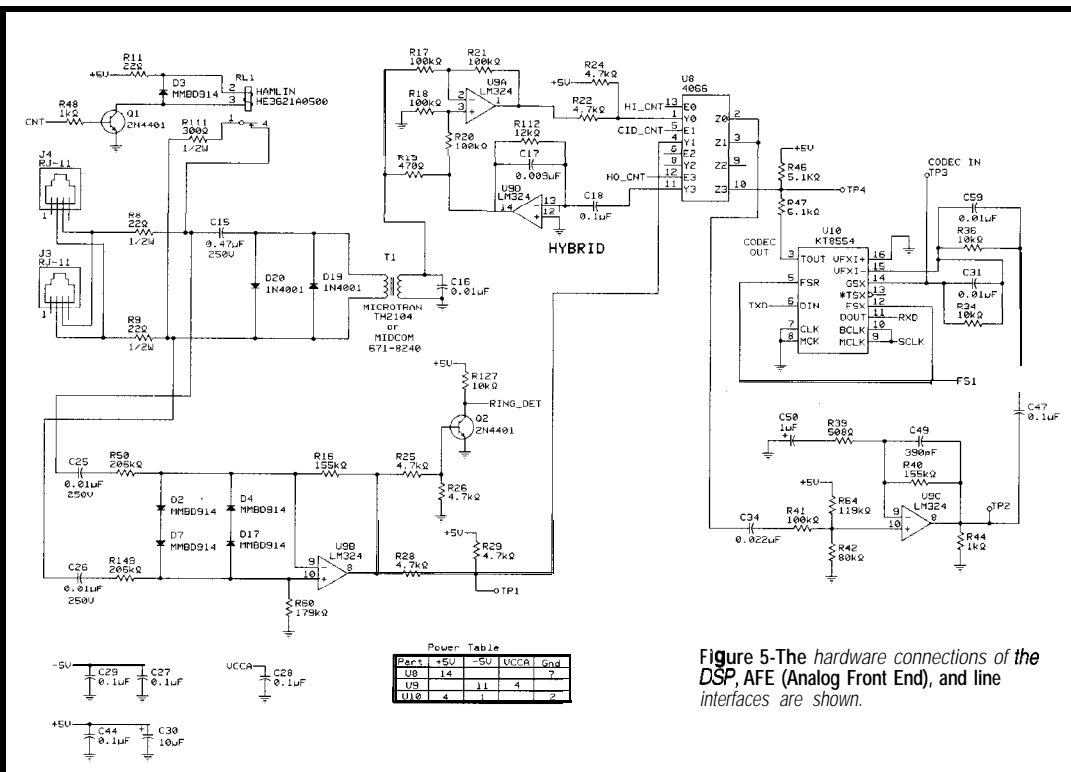


Figure 5—The hardware connections of the DSP, AFE (Analog Front End), and line interfaces are shown.

lar to a standard low baud-rate V.21/Bell 103 modem.

## DATA RECOVERY

After FSK demodulation, the obvious concerns are the data format (see Table 1) and how to decode it.

The message type is 80% (128 or MDMF). Data therefore sent as parameter words where the parameter type and length are binary and the calling name and number are ASCII.

The last word of SMDFs or MDMFs is the checksum. The checksum is the 2's complement of the modulo 256 sum of the binary representation of all other words in the message including

message type and length as well as the parameter type and length.

Remove the start and stop bits. To obtain the 2's complement, XOR with %FF. If you use Table 1 and hex calculations, the checksum is:

$$CS = \text{XOR} (\text{MOD}(\text{sum}(80, 27, 01, 08, \dots, 20, 52, 79, 61, 6E), 100), \text{FF})$$

$$CS = \text{XOR} (\text{MOD}(7A8, 100), \text{FF})$$

$$CS = \text{XOR} (A8, \text{FF})$$

$$CS = \%57$$

In a practical application, the calculation is less cumbersome due to the natural modulo 256 nature of a byte.

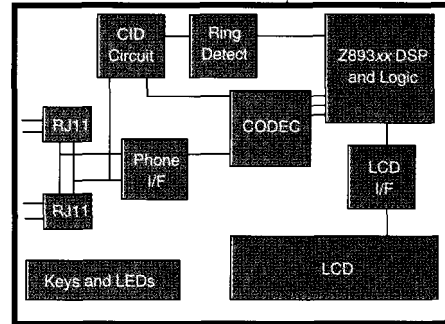


Figure 6—The Z893xx DSP is shown with its primary links. The main route for DSP connection is the CODEC or A/D and D/A connection through which the FSK data is received and the DTMF tones are sent back to the central office or exchange.

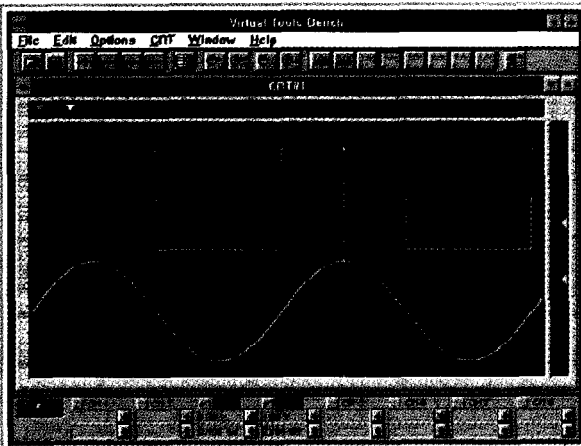
Since there's no error correction, the practical application of the checksum is to compare the received checksum

# Parallel port connected and ready to go...



Our Digital Sampling Oscilloscopes have 20 or 40 MHz maximum sampling rates with 8-bit resolution. Both have 32 Kbytes of storage; 7 sampling depths; 24 sampling rates; 6 input voltage ranges; and multiple trigger options.

20 MI-k **\$199/\$249** 40MHZ

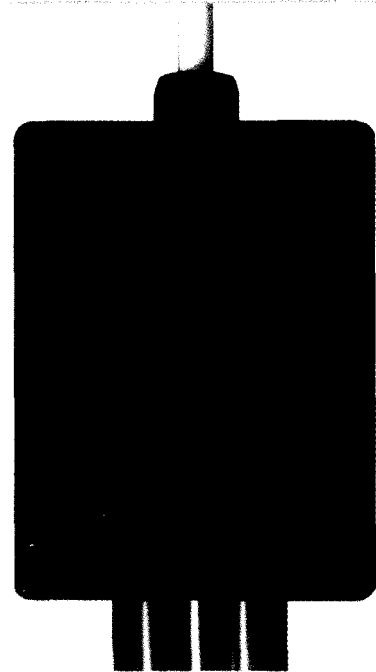


Our Virtual Tools Bench software detects connected devices and installs appropriate control surfaces. Features include 4 data display screens with rapid zooming and panning. Supports up to 8 devices with trigger interconnects and chaining.

**FREE** at website

Our Logic Analyzer has 16 channels with 3.3V/5V compatible logic inputs. The maximum sampling rate is 40 MHz with 8 internal clock rates and an external clock input with + or - going slope. The internal trigger setup allows bits to be low, high or disabled. An external logic level trigger is provided as well as the ability to trigger from our DSO. The internal storage is 32 Kbytes with 7 sampling depths and 3 trigger position options. Units can be chained for larger data widths.

**\$199**



**VIRTUAL TOOLS, Inc.**  
(619) 940-0259 FAX (619) 940-1427

VISA MASTERCARD <http://www.virtualtools.com>



with the calculated one. If they don't agree, then the data is bad and should generally not be displayed.

### CAS DETECTION

A software CAS-detection function is integrated into the DSP as shown in Figure 4. It distinguishes the periodic nature of the CAS tones from the aperiodic nature of voiced VOX.

CAS frequencies are  $2130 \pm 5\%$  Hz and  $2750 \pm 5\%$  Hz, making it a DTMF signal. However, the CAS frequencies are quite distinctly beyond the range of normal signaling DTMF frequencies.

The signal is first filtered with CAS high-filter  $2750 \pm 5\%$  Hz and also CAS low-filter  $2130 \pm 5\%$  Hz. The resultant outputs are rectified and tested for minimum amplitude requirements.

If requirements are met for both frequencies, a timer checks for CAS duration. For detection, the amplitude must constantly exceed minimum requirements for a period longer than a predetermined gating limit.

The CAS-detector ISR services the CAS detection (see Listing 1). This portion first saves the accumulator and status register, and then, the data is retrieved from the codec.

The codec register ext6 is double buffered, which means that there are really two ext6 registers-extb-0 and ext6-1. The assembler only reads ext6, which is why we have an apparently redundant load of ext6 to push the data out.

Next, the filters are called. Since the biquad structure is used, the filters are called three times to give a net sixth-order filter.

This process repeats twice—once for each filter. The final portion of the casint ISR restores the accumulator and status register.

The basic biquad structure filters the core. This portion of the code is structured so the tap updates and actual filter calculations are performed within the biquad subroutine.

The first section of this code updates the input taps or delays. The newest sample of

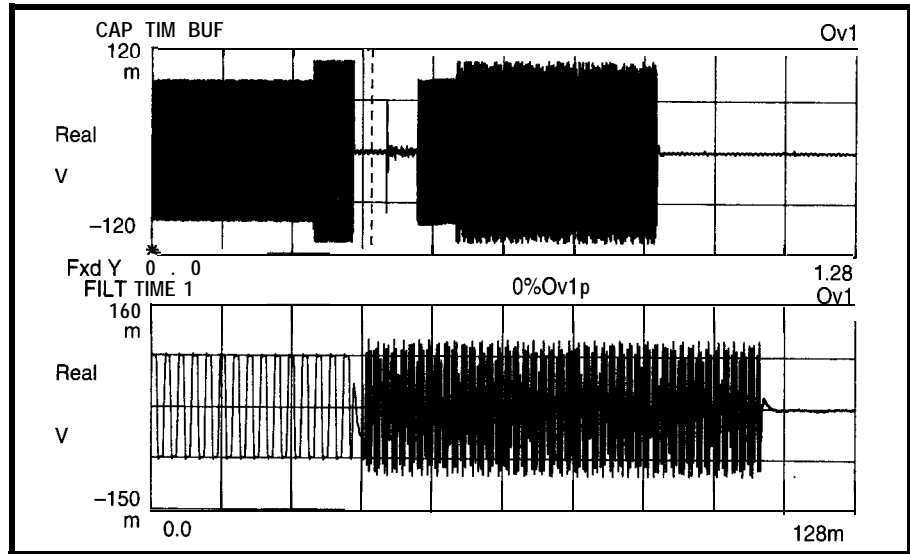


Figure 7-In the off-hook mode, the CAS signals the availability of the caller-ID information.

data  $x(n)$  or data at time  $n$  replaces the older  $x(n)$ , which is saved as  $x(n-1)$  or the current sample delayed by 1 (i.e.,  $n-1$ ). This process repeats for all taps.

Autoincrement performs filter computations (coeff \* sample). Also, a single-cycle M PY A (multiply and accumulate) instruction is used.

The fundamental equation is:

$$y(n) = b_0 * x(n) + B_1 * x(n-1) + b_2 * x(n-2) + a_1 * y(n-1) + a_2 * y(n-2)$$

where  $b_i$  and  $a_i$  are the filter coefficients and  $x(n)$  is the current sample.  $x(n-1)$  is the previous sample, and  $x(n-2)$  is the second previous sample.  $y(n)$  is the current output,  $y(n-1)$  is the previous output, and  $y(n-2)$  is the second previous output.

The last code section updates the output taps or delays in time. The newest output  $y(n)$  (i.e., output data at time  $n$ ) replaces the older  $y(n)$  and is saved as  $y(n-1)$  or the current output delayed by 1 (i.e.,  $n-1$ ).  $y(n-1)$  is saved as  $y(n-2)$ . The process is repeated for all output taps.

### DISPLAY DRIVERS

Our display is a off-the-shelf dot-matrix LCD with 16 characters x 4 lines. It is logically organized with 2 lines of 32 characters which overrun. It can display any ASCII character and many other characters.

The low-level drivers and controller are mounted on the LCD module. You just need a relatively simple high-level software driver to instruct the LCD which character to display and where to place it.

The DSP bit bangs the ASCII data to the LCD controller using the DSP's external data bus. The LCD is a relatively slow device, far slower than normal DSP operations, so updating the LCD presents minimal overhead to the DSP.

Bit (MSB-LSB)	ASCII/HEX%	Stop	7	6	5	4	3	2	1	0	Start
Message Type	/ 80	1	1	0	0	0	0	0	0	0	0
Message Length	/ 27	1	0	0	1	0	0	1	1	1	0
Parameter Type	/ 1	1	0	0	0	0	0	0	0	1	0
Parameter Length	/ 8	1	0	0	0	0	1	0	0	0	0
Month	0 / 30	1	0	0	1	1	0	0	0	0	0
Day	6 / 36	1	0	0	1	1	0	1	1	0	0
Hour	2 / 32	1	0	0	1	1	0	0	1	0	0
Minute	3 / 33	1	0	0	1	1	0	0	1	1	0
Parameter Type	/ 3	1	0	0	0	0	0	0	1	1	0
Parameter Length	/ A	1	0	0	0	1	0	1	0	1	0
DN 4083708504	4 / 34	1	0	0	1	1	0	1	0	0	0
Parameter Type	/ 8	1	0	0	1	1	0	0	0	0	0
Parameter Length	/ 8	1	0	0	1	1	1	0	0	0	0
DN 4083708504	3 / 33	1	0	0	1	1	0	0	1	1	0
Parameter Type	/ 7	1	0	0	1	1	0	1	1	1	0
Parameter Length	/ 0	1	0	0	1	1	0	0	0	0	0
DN 4083708504	8 / 38	1	0	0	1	1	1	0	0	0	0
Parameter Type	/ 5	1	0	0	1	1	0	1	0	1	0
Parameter Length	/ 3	1	0	0	1	1	0	0	0	0	0
DN 4083708504	4 / 74	1	0	0	1	1	0	1	0	0	0
Parameter Type	/ 9	1	0	0	0	0	0	1	1	1	0
Parameter Length	/ A	1	0	0	0	1	0	1	0	1	0
CN Dave Ryan	D / 44	1	0	1	0	0	0	1	0	0	0
Parameter Type	/ a	1	0	1	1	0	0	0	0	1	0
Parameter Length	/ v	1	0	1	1	0	0	1	1	0	0
DN 4083708504	e / 65	1	0	1	1	0	0	1	0	1	0
Parameter Type	/ Space	1	0	1	0	0	0	0	0	0	0
Parameter Length	/ R	1	0	1	0	1	0	0	1	0	0
DN 4083708504	/ y	1	0	1	0	1	0	0	1	0	0
Parameter Type	/ a	1	0	1	1	1	0	0	1	0	0
Parameter Length	/ n	1	0	1	1	0	1	1	1	0	0
Checksum	/ 57	1	0	1	0	1	0	1	1	1	0

Table 1—This traces a call by Dave at 0816 AM on June 23. If you follow vertically down the second column, you can see the individual elements of this transaction byte by byte.

Listing 1—The base filter's biquad routine may be used for a great variety of filters. Of course, the filter coefficients must be computed in each case. Note the single-instruction multiply and accumulate capability.

```

;F1 Sixth-order triple biquad IIR filter
ld a,#X11 ;load Acc with address (X11) for input samples
ld p0:0,a ;point to input sample
ld a,#BF1 ;address (BF1) for filter coefficients
ld p0:1,a ;point to filter coefficients
;read new input sample x(n)
ld ext6,a ;push new u-law input sample ext6-2 to ext6-1
ld a,ext6 ;load u-law data to accumulator x(n)
Ulaw ;u-law result is 14-bit sign magnitude number
sll a ;shift left logical, multiply by 2
sll a ;shift left logical, multiply by 2
ld x,a ;x = new data
ld tempst,a ;store temporary input storage
call biquad ;perform standard biquad
call biquad ;perform standard biquad
call biquad ;perform standard biquad
; save output filter response y(n)
ld outgbf1,a ;store third-stage output
; ld ext6,a ;Codecl output y(n) auto hardware u-law
endi:
ret ;return from Codec ISR
biquad:

;perform filter computations (coeff * sample) using autoincrement
;y = b0*X(n) + B1*X(n-1) + b2*X(n-2) + a1*Y(n-1) + a2*Y(n-2)
;Input New Sample is in x, Output is in Acc
;update input sample buffer
ld y,@p0:0 ;y saves old (n) p0:0 points at (n)
ld @p0:0+,x ;(n) = new sample p0:0 points at (n)
ld x,@p0:0 ;x saves old (n-1) p0:0 points at (n-1)
ld @p0:0+,y ;(n-1) = (n)
ld y,@p0:0 ;y saves old (n-2) p0:0 points at (n-2)
ld @p0:0,x ;(n-2) = (n-1)
ld a,p0:0 ;p0:0 points at (n-2)
sub a,#%2 ;decrement acc
ld p0:0,a ;p0:0 points at (n)
;y = b0*X(n) + B1*X(n-1) + b2*X(n-2) + a1*Y(n-1) + a2*Y(n-2)
mld @p0:1+,@p0:0+,on ;A= 0 P2 = (b0 * X11) X11 = X(n)
mpya @p0:1+,@p0:0+,on ;b1 * X12 X12 = X(n-1)
mpya @p0:1+,@p0:0+,on ;b2 * X13 X13 = X(n-2)
mpya @p0:1+,@p0:0+,on ;a1 * Y11 Y11 = Y(n-1)
mpya @p0:1+,@p0:0+,on ;a2 * Y12 Y12 = Y(n-2)
add a,p ;add result of last multiply to Acc.
sll a ;scale back if divide by 2 on coefficients
ld x,a ;return result in x
;update output buffer
ld a,p0:0 ;p0:0 points at Y12+1
sub a,#%2 ;decrement acc
ld p0:0,a ;p0:0 points at Y11(n-1)
ld y,@p0:0 ;y saves old (n-1)
ld @p0:0+,x ;Y11 = new result
ld @p0:0+,y ;Y12 = Y11 = Y(n-2)
; output filter response y(n)
ld a,x ;store stage output
ret ;return from biquads

```

Communication is done via a specialized series of LCD instructions. Once the LCD is initialized, the data is transmitted.

## CALL PROGRESS

Call progress is sequential. A ring must be detected first. Once the call is established, only one of two events can happen—either a call interrupt occurs or does not occur.

When the SDMF or MDMF data is received, it should be displayed. A state machine takes care of the logical progress of the call. At the end of the call, a disconnection occurs and the entire cycle repeats.

## MEMORY STORAGE

Again, due to the multitasking features common to DSPs, normal call progress, especially on-/off-hook moni-

toring, system supervision, memory for calls received, and display tasks, can be handled by a single DSP.

For the sake of simplicity, we didn't add memory storage to this demo. The external bus addressing capability enables this feature to be easily added.

## TAKE IT FURTHER

The system described is elemental. Many value-added features are possible (e.g., ring only on certain callers). Such features are easily added as controller functions.

Just let your imagination lead. □

Dave Ryan is a systems engineer in Zilog's data communications. He works on their next-generation 50-MIPS fixed-point processor-class device. You may reach Dave at [drya@zilog.com](mailto:drya@zilog.com).

Asher Hazanchuk works in SDP system engineering and applications at Zilog. He has 15 years of DSP experience in image processing and compression, digital answering machines, cell phones, caller ID, magnetic-stripe readers, and DSP architectures.

## SOFTWARE

The complete code for this article is available on the Circuit Cellar Web site.

## REFERENCES

- J.D. Gibson, *Principles of Digital and Analog Communications*, MacMillan, New York, NY, 1993.
- Bellcore, Technical Reference TR-NWT-000031 and NWT-001188.
- Bellcore, Generic Requirements GR-30-CORE.
- Bellcore, Special Reports SR-TSV-002578 and SR-TSV-002476.

## SOURCE

**Z89CID00ZCO**  
Zilog  
2 10 Hacienda Ave.  
Campbell, CA 95008-6600  
(408) 370-8000  
Fax: (408) 370-8056

## I R S

404 Very Useful  
405 Moderately Useful  
406 Not Useful

# FEATURE ARTICLE

Chris Sakkas

## PC Telephone Interface

Chris shows you how to make a low-cost expansion card with a complete telephone interface. His software enables you to create applications for voice messaging, call processing, outbound calling, and home automation.



Applying computer control over the telephone can produce fascinating applications beyond simple voice mail. Computer telephony also includes complete interactive voice-response systems, call processing, autoattendants, and more.

As well, computer telephony integration can lead to interesting applications involving remote access to computer control and home-automation systems.

In this project, a low-cost ISA expansion card serves as a complete telephone interface. It records and plays back messages, decodes touchtones, dials, and handles switch-hook control.

I also discuss software for developing a nine-mailbox voice-mail system. This software-hardware combination is a useful base for creating applications for voice messaging, call processing, home automation, and more.

### CONCEPT

Figure 1 outlines the hardware design, showing the telephone input to the card as well as the I/O and functional relationship of individual items.

The Data Access Arrangement takes telephone input and passes it to a summing amplifier to mix the signal with the microphone input. This input is amplified to a level ready to sample via the preamplifier and a second amplifier.

The signal is fed through the anti-aliasing low-pass filter and sampled by the ADC. Since the phone system bandwidth is limited to 4 kHz, the sampling frequency must be at least 8 kHz to satisfy the Nyquist sampling-frequency theorem. The CPU gets this data byte via the ISA interface.

After the DAC converts PC data to analog form, the signal is fed into a reconstruction filter and then mixed with the DTMF transmitter output. An audio amplifier amplifies the signal into levels capable of driving a speaker.

### SPECIFICATIONS

The hardware needed to handle 8-bit A/D and D/A conversions, as well as DTMF tone decoding and transmission. It had to be able to sample a 4-kHz signal, and its data storage rate was limited to at most 8 kbps.

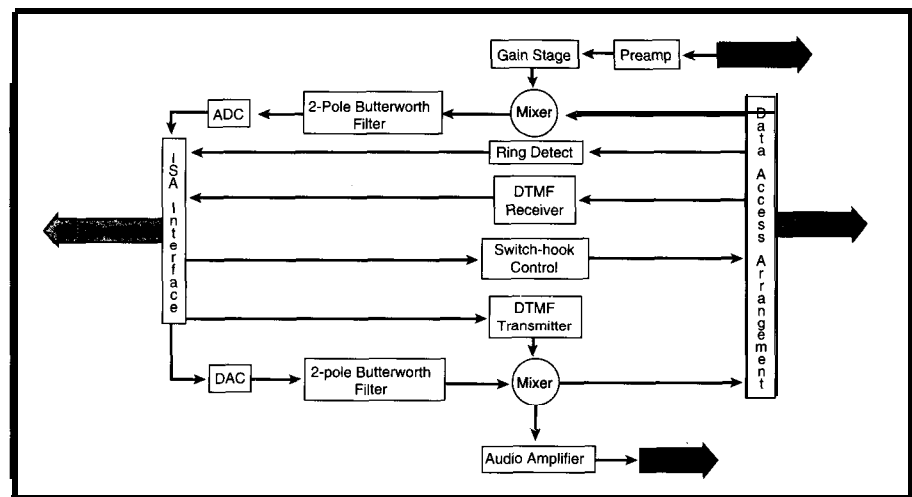


Figure 1—The input and output relationship of all subsystems is shown. Many of the subsystems were implemented in single monolithic devices.

As well, it needed an **RJ-11** phone-line connection and a user-selectable port address. Finally, it had to satisfy FCC Part 68 requirements.

To minimize components, complexity, and cost and maximize the hardware's flexibility, I chose highly integrated components to handle the interface logic, A/D and D/A conversions, DTMF decoding and transmission, and telephone-line interfacing.

## HARDWARE

The Analog Devices AD7569 8-bit Analog I/O system provides fast A/D and D/A conversion in a small, low-cost 24-pin package. It has a minimal

bus interface, 2- $\mu$ s conversion time, and single supply voltage, which accepts several ranges of input voltages.

The Teltone M-8888 DTMF transceiver handles DTMF tone decoding and transmission. This 20-pin package provides easy interfacing with a microprocessor and has a call-progress mode. (It works with a single supply voltage.)

The Xecom XE0068 Data Access Arrangement provides TTL-level ring detection and switch-hook control. The internal Automatic Gain Control (AGC) circuit optimizes transmit levels and maintains a small package size.

This device provides a legal, low-cost interface to the phone system with

its FCC Part 68 registration. The registration transfers to the end application.

Figure 2 shows the schematic for the interface card. A 74LS688 8-bit comparator is used as a decoder for the board. When an address corresponding to the card's base port address is detected, the enable of the 74LS245 octal bus transceiver is selected so the data-bus contents can be accessed.

The 74LS244 also buffers between the bus and hardware, the I/O read and write lines, and the two least significant bits of the address bus. These bits are needed for decoding which of the four port addresses is to be used for hardware access.

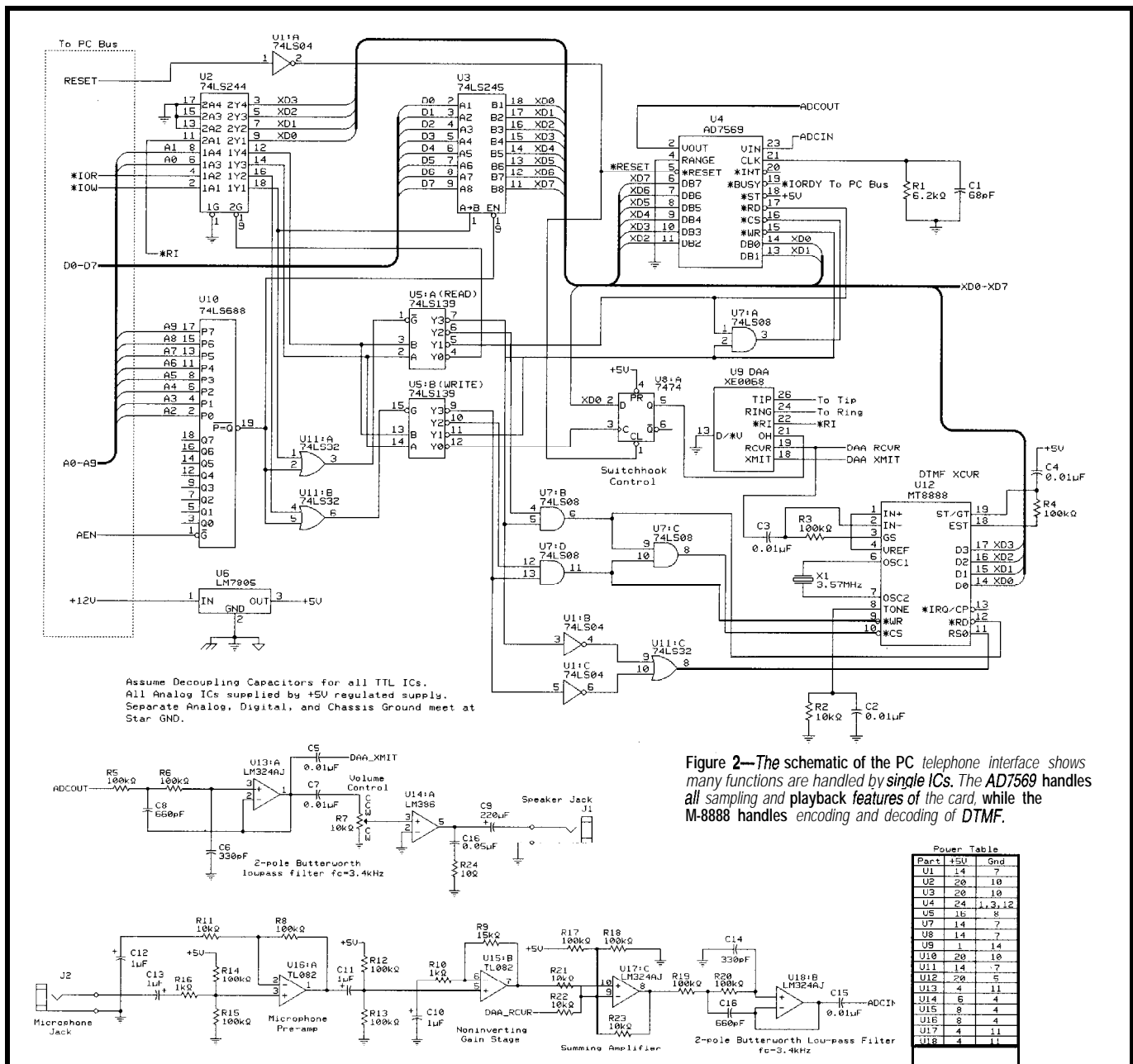


Figure 2—The schematic of the PC telephone interface shows many functions are handled by single ICs. The AD7569 handles all sampling and playback features of the card, while the M-8888 handles encoding and decoding of DTMF.

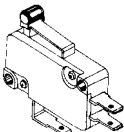
Part	+5V	Gnd
U1	14	7
U2	20	10
U3	20	10
U4	24	1,3,12
U5	16	8
U7	14	7
U8	14	7
U9	1	14
U10	20	10
U11	14	7
U12	20	5
U13	4	11
U14	6	4
U15	8	4
U16	8	4
U17	4	11
U18	4	11

# ALL ELECTRONICS

C O R P O R A T I O N

## CHERRY D43 SNAP-ACTION SWITCH

S.P.D.T. snap-action switch with roller positioned above switch actuator. Rated 5 amps @ 125/250 vac. Switch body: 1.1" x 0.63" x 0.375". Solder or qc terminals. UL and CSA listed.



**\$1.25**  
each

CAT# SMS-150  
10 for \$10.00

## 470 UF, 450 VOLT SNAP-IN CAPACITOR

Nichicon LGQ2W471 MHSC  
1.375" diameter x 2" high.  
0.4" lead spacing.



**\$4.50**  
each

CAT# EC-4745  
10 for \$40.00

## SMALL NEODYMIUM RARE EARTH MAGNETS

Semi-circular, irregularly-shaped magnets. Shiny finish with a polarity marking dot. 0.93" long x 0.3" x 0.07" thick. Powerful for their size.

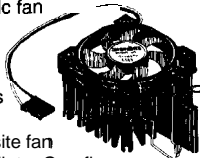


**\$1.00**  
each

CAT# MAG-30  
160 PCS. - \$100.00

## 12 Vdc CPU FAN

Enertron# NX586-02Z-20  
7 blade, mini 12 vdc fan on a heatsink. Assembly is 2" square X 1.22" high. Heatsink has a 0.87" square flat area on side opposite fan from which fins radiate. One fin extends 0.63" beyond the others. Includes two mounting clips.



**\$6.50**  
each

CAT # CF-40

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

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

CALL, WRITE  
FAX or E-MAIL  
for our FREE

**96 Page  
CATALOG**  
Outside the U.S.A.  
send \$3.00 postage.

MAIL ORDERS TO:  
ALL ELECTRONICS  
CORPORATION  
P.O. Box 567  
Van Nuys, CA 91401  
FAX (818)781-2653

E-Mail [allcorp@allcorp.com](mailto:allcorp@allcorp.com)  
Internet - <http://www.allcorp.com>

#114

Port	Bit	Read	Write
\$300	0	Ring detect (0 = ring)	Hook switch (0 = on, 1 = off)
\$301	O-7	ADC read	DAC write
\$302	O-3	Read DTMF receiver	Write to DTMF transmitter
\$303	O-3	Read DTMF status register bit	Write DTMF control register

Table I-Four PC I/O ports are used for the card. Additional functionality can be added to the card and controlled via the first port address if needed.

The 74LS139 consists of two 2-to-4 decoders, each supplied with A0 and A1 of the address bus. The appropriate portion for reading or writing is enabled, depending on the status of the \*IOR and \*IOW bus lines.

The base address of \$300 (hex) is used, but any nonconflicting address is possible. Changing the address means pins 16 and 18 of the 74LS688 should be tied high. The rest should be low for this addressing example. Table 1 lists the port addresses and functions.

Depending on the action taken by the 74LS139, the appropriate component is enabled-either the AD7569 for read or write or the M-8888 for read, write, or register selection.

It can also read the contents of the XE0068's Ring Indicator pin via the second half of the 74LS244. Or, it can toggle the hook switch by changing the contents of the 74LS744 D-type flip-flop [which acts as a I-bit register].

The AD7569 converts data when it's selected and the RD (read) pin is strobed. The IC activates its \*BUSY line, which is connected to \*IORDY on the PC bus. This action extends the read's bus cycle if necessary to the amount needed for a read to occur.

Due to the relatively low sampling frequency, I didn't use precise timing circuitry. All timing was done via the PC's programmable interval timer.

In a PC-compatible system, this timer has three different channels, and channel zero is the system clock-tick timer. The ROM BIOS programs the timer to generate an interrupt 08 at a frequency of 18.2 times per second.

For most systems, however, this frequency can be reprogrammed to occur at a much greater rate, making it more useful for this project. Software can reprogram the timer and still maintain the proper call to other service routines 18.2 times per second.

I chose a microphone preamplifier based on a noninverting amplifier

using one-half of a TL082 operational amplifier. The op-amp is biased to operate from a single 5-V supply, as are all other op-amps in this design.

The preamp provides a 20-dB gain. This low-level signal is amplified again by the second-half of the TL082 op-amp configured as another noninverting amplifier with a gain of 23.5 dB.

I chose National Semiconductor's TL082 JFET input operational amplifier for its high input impedance, low noise voltage, and low input bias current. These features make it ideal for converting a microvolt signal to a millivolt signal.

A summing amplifier mixes outputs from the microphone amplification circuitry and the DAA's receiver, using one-fourth of an LM324 op-amp. The summing amplifier's output is applied to a two-pole Butterworth low-pass filter before entering the ADC.

A second summing amplifier mixes the DAC and DTMF outputs. A two-pole Butterworth low-pass filter acts as a reconstruction filter for this signal.

Both filters in this design are identical and are based on the popular unity-gain Sallen and Key configuration. I use National Semiconductor's LM324 quad op-amp for both since it is low cost and has four op-amps per package. The filters were designed for a 3.4-kHz cut-off frequency, appropriate for filtering out aliasing signal elements for this application.

The reconstruction filter output is applied to the DAA's transmit pin and the input of the LM386 audio amplifier. The LM386 amplifier provides adequate audio amplification in a low-cost monolithic package. The audio output connects to a jack on the back of the card.

## SOFTWARE

Voice data is managed by a message structure, `mess_t` (see Web site for source code). It stores a pointer to the



actual voice message data, the number of data bytes, an indicator of whether the data is in memory or stored on disk, a message description, and a filename.

An enumerated type, `bitstatus`, is defined with on and off for Boolean control. The software has functions that can be integrated into other programs to incorporate telephone support. The routines are divided into telephone-control, message record and playback, and DTMF functions.

Telephone-control functions include `WaitForRing`, `HookSwitch`, and `RingDetect`. `WaitForRing` waits for an incoming number of rings based on the variable `count`. Control goes to the calling function after the specified ring number is encountered.

`HookSwitch` simply controls the telephone's hook-switch status, either on or off. `RingDetect` returns on if a ring is detected and off otherwise.

`ReadFile`, a message record and play-back function, reads a specified filename into memory for playback. `Play-MessageandRecordMessage` expect a `message` structure to be passed to begin playback or recording. The PC's programmable interval timer is used for machine-independent timing.

DTMF initialization is performed via `DTMFInit`. `DTMFReceive` and `DTMFTransmit` read or place the DTMF character in the M-8888 buffer. `DTMF-Transmit` must be setup with a call to the `DTMFToneBurst` function.

With these functions, I developed a small voice-mail application for nine mailboxes. After a main greeting and the individual voice message for each mailbox is played, the user may record a message at the tone.

The main greeting is contained in the file `GREET - MN`. Mailbox greetings are contained in `GREET x`, where `x` denotes the mailbox number. A received message is stored in `MESSAGE - x`. This entire application was coded in -60 lines of C code.

Several example programs show further potential uses of the card. These programs can record to a file, playback a file, and act as a telephone dialer.

## INTERFACING IDEAS

The PC-telephone interface provides an easy way to interface a PC-

compatible computer to the telephone network. Other more sophisticated applications can be developed, including many beyond typical computer-telephony applications.

A home-automation or other computer-controlled system can be modified to receive commands and deliver reports remotely. With added circuitry, a complete amateur-radio repeater controller can be created with voice and sophisticated computer control. □

*Chris Sakkas is president of ITU Technologies, a company specializing in development tools for microcontrollers. You may reach him via E-mail at [chris@itutech.com](mailto:chris@itutech.com) or by telephone at (513) 574-7523.*

## SOFTWARE

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

## SOURCES

### AD7569

Analog Devices  
One Technology Way  
Norwood, MA 02062-9 106  
(617) 329-4700  
Fax: (617) 329-1241

### M-8888

Telton Corp.  
22121 20th Ave. SE  
Bothell, WA 98021  
(206) 487-1515  
Fax: (206) 487-2288

### XE0068

Xecom  
374 Turquoise St.  
Milpitas, CA 95035  
(408) 945-6640  
Fax: (408) 942-1346

### TL082, LM324, LM386

National Semiconductor  
P.O. Box 58090  
Santa Clara, CA 95052-8090  
(408) 721-5000  
Fax: (408) 739-9803

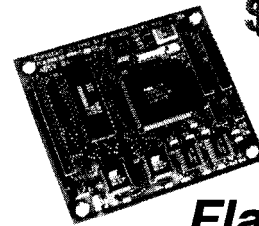
## IRS

407 Very Useful  
408 Moderately Useful  
409 Not Useful

## EMBEDDED DOS CONTROLLERS AT 8051 PRICES

Use Your PC Development Tools!

No MORE CRASH & BURN EPROM  
Technology



**\$195**  
Qty 1 Price

## Flashlite

DOS Single Board Computer

with 572 k FLASH Memory disk drive

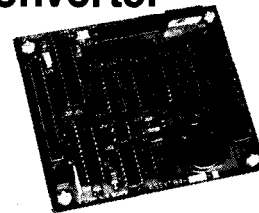
- ✓ 10 Mhz/8 Mhz CPU
  - ✓ 2 Timers
  - ✓ 512 k bytes RAM
  - ✓ 4 Interrupt Lines
  - ✓ 512 k/256 k FLASH
  - ✓ 8 Analog Inputs
  - ✓ 2 Serial Ports
  - ✓ X-Modem File Transfer
  - ✓ 24 Parallel I/O Lines
- INCLUDES DOS & Utilities

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

## A/D Converter

**\$95**

Qty 1 Price



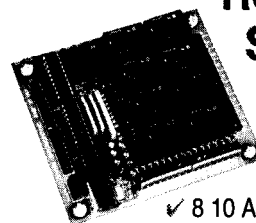
- ✓ 8 Channels, 12 Bits
- ✓ 6  $\mu$ s. Conversion Time
- ✓ Clock/Calendar Option
- ✓ Includes Drivers & Apps.

GET YOUR EMBEDDED CONTROLLER  
PROJECT RUNNING FAST!  
WITH THESE ACCESSORIES

## Relay I/O

**\$139**

Qty 1 Price



- ✓ 8 10 Amp Relays
- 4 8 Opto-Isolated Inputs

## JK microsystems

Cost Effective Controllers for Industry

TO ORDER (510) 2364151

FAX (510) 236-2999—email: [jkmicro@dsp.com](mailto:jkmicro@dsp.com)  
Visit our WEB site—[www.dsp.com/jkmicra](http://www.dsp.com/jkmicra)  
1275 Yuba Ave., San Pablo, CA 94806

#115

# FEATURE ARTICLE

Art Sobel

## Embedding the ARM7500

### Part 2: Programming an Embedded Computer

Building on last month's introduction of the ARM7500, Art moves from chip architecture and development boards to firmware. He wants to make sure we have the instruction set that makes it zip!

**O**ne ARM7500 is exceedingly complicated, having similar resources to a typical PC's CPU and motherboard logic. After building the development board, my first task was porting the C-Demon, a ROM-based monitor used in other ARM development boards.

After the C-Demon was working, each major chip section needed drivers. These drivers are wrapped up in the console test program.

The ARM7500's CPU and MEMC memory, I/O, and VIDC video/sound controllers were conserved from the original Acorn computer, keeping the original OS and user software somewhat compatible.

From a programmer's view, the ARM7500 functional blocks are separate sets of registers incorporated in the memory map as shown in Figure 1.

As Table 1 shows, the IOC handles internal peripherals like keyboard and mouse control, 11 general-purpose I/O pins, video flyback, and two 16-bit timers. It

also controls six sets of interrupt-control registers, four single-slope ADCs for the joystick interface, memory and I/O timing, as well as ROM and DRAM width.

Lastly, the IOC has registers controlling the clocks. The CPU clock can be turned off, or the whole chip can have clocks suspended.

The external clock can also be controlled. In stopped mode, an external clock/calendar restarts the chip by grounding one of two special interrupt pins.

The DMA channels were historically part of the MEMC. The basic DMA channels are retained in the ARM7500 (see Table 2).

The myriad video-timing registers, pixel control, and clock control, as well as the analog sound clock and steering register are placed in the VIDC functional block (see Table 3).

#### C-DEMON PORTING

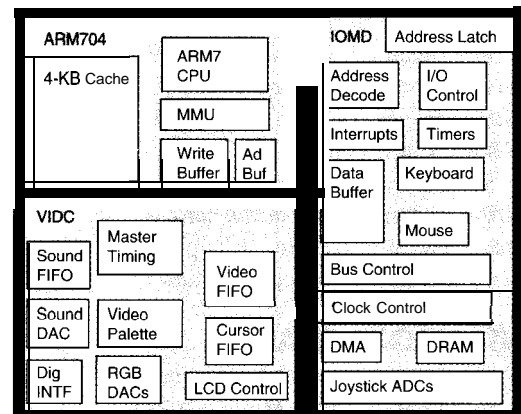
Demon initializes the ARM and peripheral registers, builds a compatible memory map for monitor variables, and starts communication with the host.

The ARM7500's ROM controller resets to 16-bit mode. I chose a 32-bit-wide ROM for the onboard software. This switch was a bit tricky as the ARM program counter nearly always fetches two instructions ahead of the execution unit.

In 16-bit mode, the memory controller accesses the low and then the high 16 bits and presents the assembled 32-bit word to the instruction unit. In Listing 1 (Level 0 code), the first 14 entries have the upper 16 bits zeroed.

The ROM start-up code is hand assembled since the rest of the code is

Figure 1—Acorn's former discrete chipset, CPU, IOC, MEMC, and VIDC are preserved in the layout of the ARM7500.



Name	Address	Size	Read	Write	Description	Name	Address	Size	Read	Write	Description
IOCR	00	8	IOCR	IOCR	I/O Pin Ctrl	VIDCAUX	6C	8	VIDAUX	VIDAUX	Video AuxCtrl
KBDAT	04	8	KBDATIN	KBDATOUT	Keyboard Data	IRQSTD	70	8	IRQ StatusD	---	IRQD Stat
KBDCR	08	8	KBDCR	KBDCR	Keyboard Stat and Ctrl	IRQREQD	74	8	IRQ ReqD	---	IRQD Req
IOPINS	0c	---	IOPINS	IOPINEN	8 Open-Drain I/O Pins	IRQMSKD	78	8	IRQ MaskD	IRQ MaskD	IRQD Mask
IRQSTA	10	8	IRQ StatusA	---	IRQA Stat	ROMCRO	80	8	ROM Con 0	ROM Con 0	ROM 0 Timing Ctrl
IRQRQA	14	8	IRQ ReqA	IRQ clear A	IRQA Req	ROMCR1	84	8	ROM Con 1	ROM Con 1	ROM 1 Timing Ctrl
IRQMSKA	18	8	IRQ MaskA	IRQ MaskA	IRQA Mask	RESV	88	8	---	---	---
IDLEMD	1C	1	---	Enter IDLE MODE	CPU Idle Cmd	RFSHCR	8C	8	Refresh CR	Refresh CR	DRAM Refresh Ctrl
IRQSTB	20	8	IRQ StatusB	---	IRQB Stat	ID0	94	8	Chip ID L byte	---	---
IRQRQB	24	8	IRQ ReqB	---	IRQB Req	ID1	98	8	Chip ID H byte	---	---
IRQMSKB	28	8	IRQ MaSK B	IRQ Mask B	IRQB Mask	VERSION	9C	8	Chip Version	---	---
STOPMD	2C	1	---	Enter STOP MODE	Clock Stop Cmd	MSDAT	A8	8	MSDATIN	MSDATOUT	Mouse Data
FIQST	30	8	FIQ Status	---	FIQ Stat	MSCR	AC	8	MSCR	MSCR	Mouse Ctrl and Stat
FIQRQ	34	8	FIQ Req	---	FIQ Req	reserved B0-BC					
FIQMSK	38	8	FIQ MaSK	---	FIQ Mask	IOTCR	c4	8	IO Timing	IO Timing	IO Timing Ctrl
CLKCTL	3C	8	CLKCTL	CLKCTL	Clock Ctrl	ECTCR	C8	8	Ext IO Timing	Ext IO Timing	EASICS Timing
T0LOW	40	8	T0countL	T0LatchL	Tmr 0 Latch Data Low	ASTCR	c c	8	ASTCR	ASTCR	Ext MEMC Timing
T0HIGH	44	8	T0countH	T0LatchH	Tmr 0 Latch Data High	DRAMWID	D 0	8	DRAMWID	DRAMWID	DRAM Width
T0GO	48	0	---	T0Go Command	Tmr 0 Start	SELFREF	D4	8	SELFREF	SELFREF	Self Refresh Ctrl
T0LAT	4c	0	---	T0Latch	Tmr 0 Latch Cmd	JOYICR	E0	8	JOYICR	JOYICR	Joystick Int Ctrl
T1LOW	50	8	T1countL	T1LatchL	Tmr 1 Latch Data Low	JOYSR	E4	8	JOYSR	---	Joystick Stat
T1HIGH	54	8	T1 CountH	T1LatchH	Tmr 1 Latch Data High	JOYCC	E8	8	JOYCC	JOYCC	Joystick Ctrl
T1GO	58	0	---	T1Go	Tmr 1 Start	JOYCNT0	EC	16	JOYCNT0	---	Joystick Ctrl 0
T1LAT	5c	0	---	T1 Latch	Tmr 1 Latch Cmd	JOYICR1	F0	16	JOYICR1	---	Joystick Ctrl 1
IRQSTC	60	8	IRQ StatusC	---	IRQC Stat	JOYICR2	F4	16	JOYICR2	---	Joystick Ctrl 2
IRQRQC	64	8	IRQ ReqC	---	IRQC Req	JOYICR3	F8	16	JOYICR3	---	Joystick Ctrl 3
IRQMSKC	68	8	IRQ MaskC	IRQ MaskC	IRQC Mask	reserved FC-17C					

Table 1--The I/O control processor registers manage the keyboard, mouse, interrupts, timer, joystick, and memory-control functions.

in normal 32-bit format. The code loads an immediate value for the internal I/O controller address and the new ROM-controller value, and then loads it into the ROM controller.

After this code, the PC is directly loaded with 0. Although the ROM controller's `store` instruction is written before the jump, it executes afterwards. The start-up code is reinterpreted in 32-bit mode as a series of NOPs.

The next trick is to remap the memory using the MMU (see Table 4). I took advantage of the ROM being mapped to 0 and also mapped to 0x20000000, since the physical memory map repeats on 512-MB boundaries.

The program jumps to the higher ROM location and initializes the MMU page-table pointer to a precalculated primary page table at the end of ROM. The cache remains off so the RAM size may be determined.

Since the actual RAM is smaller than the huge space allotted (64 MB in each bank), the physical RAM repeats several times. The RAM size is found by detecting the rollover to RAM address 0 when its size is exceeded.

When cache is enabled, figuring out RAM size becomes a problem. The cache tag thinks address 0 is still valid even though it's overwritten from a higher address.

After RAM size is found and the stacks for the various ARM operation modes are initialized, the cache is enabled.

The next task is programming the internal registers for the interrupts, timer, and other functions (see Table 1).

The interrupts differ greatly from the previous ARM600 PID port. The ARM7500 has five IRQ (normal interrupt) and one FIQ (fast interrupt) registers. Thus, the processor reads up to five 8-bit registers to find out which interrupt caused an IRQ and one 8-bit register to locate the FIQ source. Each interrupt request register is read and each bit is examined for the first set bit.

This bit's location indexes into a table of interrupt routines. Despite such complexities, the ARM7500 can handle the interrupt routine much faster than

a protected-mode 'x86 processor with all its hardware interrupt-assist logic.

The ARM7500 has two 16-bit timers operating at 2 MHz (with a 32-MHz I/O clock). To produce an -IOO-Hz continuous interrupt every 10 ms, the 2 MHz is divided by 20,000.

Communications with the host are accomplished via the serial port on the PC I/O Combo chip (FDC37C665) or Ethernet (SMC91C94).

The I/O Combo's serial port is 16C550 compatible, so the code used in the previous PID board works but at a different address.

The next steps to start the Demon are common to all versions. You set up data structures in low-memory RAM, check ROM for the correct checksum, and send a banner message to the host.

Name	Address	Size	Read	Write	Description
SDOCURA	180	32	SDO 0 Current A	SDO 0 Current A	Ch A Current
SDOENDA	184	32	SDO 0 End A	SDO 0 End A	Ch A End
SDOCURB	168	32	SDO 0 Current B	SDO 0 Current B	Ch B Current
SDOENDB	18C	32	SDO 0 End B	SDO 0 End B	Ch B End
SDOCR	190	8	SDO 0 Control	SDO 0 Control	Sound Control
SDOST	194	8	SDO 0 Status	---	Sound Status
CURSCUR	1C0	32	Curs Current	---	curs current
CURSINIT	1C4	32	Curs Init	---	Curs Init
VIDCURB	1C8	32	VIDEO Current	VIDEO Current B	---
VIDCURA	1D0	32	VIDEO Current	VIDEO Current A	---
VIDEND	1D4	32	VIDEO End	VIDEO End	---
VIDSTART	1D8	32	VIDEO Start	VIDEO Start	---
VIDINITA	1DC	32	VIDEO INIT	VIDEO INIT A	---
VIDCR	1E0	8	VIDEO Control	VIDEO Control	---
VIDINITB	1E8	32	VIDEO INIT	VIDEO INIT B	---
DMAST	1F0	8	DMA Status	---	---
DMARQ	1F4	8	DMA IRQ Req	DMA IRQ Req	---
DMAMSK	1F8	8	DMA IRQ Mask	DMA IRQ Mask	---

Table 2--ARM7500 DMA registers are loaded according to their state diagram in Figure 4.

## CONSOLE TEST PROGRAM

The console test program checks the functions of the RC7500 and ARM7500, as well as the additional onboard logic. Source code for all tests helps you get a feel for the software drivers (see Listing 2). When the console program is run, Figure 2 appears onscreen.

## RAW VIDEO

Getting video comes first. Without a functional display, no progress is possible.

The ARM7500 video registers are in Table 3. The display

**Listing 1**—ARM 7500 start-up code ROM is switched from 16-bit mode to 32-bit mode. The complete listing is available on Circuit Cellar INK's Web site.

```

ENTRY                ; Entry point to C-Demon ROM world
ROMStart             ; Symbol for first location in ROM
MOV pc,#0x3C
DCD 0x0000F000      ;NEVER Op code
DCD 0x0000B632
DCD 0x0000E3A0      ;MOV R11, 0x03200000 ;point at register base
DCD 0x00000002
DCD 0x0000E3A0      ;MOV R0, #&02 ;32b,slow,155ns,no burst
DCD 0x00000080
DCD 0x0000E5CB      ;STRB R0, [R11,0x80]
                    ;Program ROMCRO & switch mode to 32 bit
DCD 0x0000F000
DCD 0x0000E3A0      ;MOV PC,#0
DCD 0x00000000
DCD 0x00000000
DCD 0x00000000      ;NOP
DCD 0x00000000      ;NOP
DCD 0x00000000      ;NOP
ROMReset            ; our exported entry point @ 3c
                    ; check processor mode on startup.
                    ; MMU control register is written as zero by reset.
                    ; This means little-endian, early abort, 26-bit data, 26-bit program.
                    ; write buffer off, cache off, alignment fault off, and MMU off world.
                    ; Configure CPU as 32-bit address and data, plus little-endian
                    ; operation.
MRS r14,CPSR        ; get current processor mode
TST r14,#0x10        ; 26- or 32-bit mode?
MOVEQ r14,#0x30      ; 32-bit address & data, little-endian
MCREQ MMUCP,0,r14,MMUControlReg,c0
MRS r14,CPSR
; Force bits to explicitly select 32-bit mode, enter 32-bit SVC mode
BIC r14,r14,#ModeMask
ORR r14,r14,#SVCmode
MSR CPSR,r14
; Now executing in 32-bit mode, with MMU suitably set
; must set the memory map.
LDR pc,=gotoROM      ; update PC with address within ROM
; We are now executing from the alt ROM location. 0x2000 0000
gotoROM

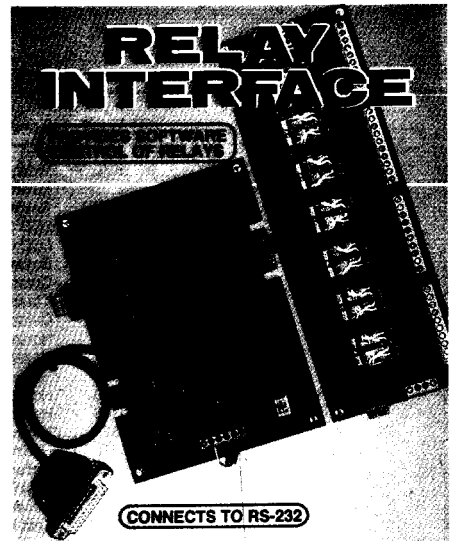
```

**Listing 2**—This initialization sequence starts the console test program.

```

main(void)
{
    int x;
    vidc_init();          /* Initialize VIDC */
    active_win = (window_t *) &conwin;
    wininit(active_win);
    font8_buf = (unsigned char *) ((vuptr) FONT8_BASE);
    set_font8(font8_buf, FONT8_BSIZE, active_win->color);
    printf("Video Initialized\n");
    if (kbdinit()) {      /* Initialize keyboard */
        printf("Keyboard initialization failed!\n");
        return(1);
    }
    printf("Keyboard Initialized\n");
    mouse_trap_init();    /* Initialize mouse */
    printf("Mouse Initialized\n");
    cons_start(active_win); /* Start console process */
    fill_vbuf(cur_video_bufp, BLACK, HSIZE, VSIZE); /* Reset
    video */
    vidc_clear();
    vidc_disable();
    vidc_dma_disable();
    return(0);
}

```

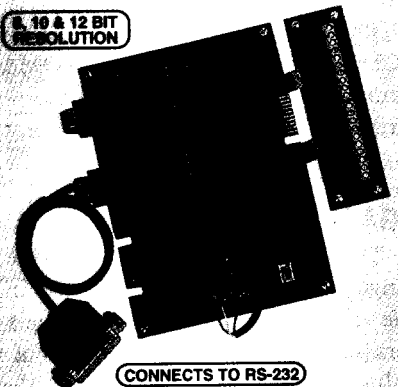


CONNECTS TO RS-232

**AR-16 RELAY INTERFACE (16 channel).....\$ 69.95**  
 Two 8 channel (TTL level) outputs are provided for connection to relay cards or other devices (expandable to 128 relays using EX-16 expansion cards). A variety of relay cards and relays are stocked. Call for more info.  
**AR-2 RELAY INTERFACE (2 relays, 10 amp)....\$ 44.95**  
**RD-8 REED RELAY CARD (8 relays, 10 VA)....\$ 48.95**  
**RD-9 RELAY CARD (10 amp SPDT, 277 VAC)....\$ 69.95**

## ANALOG TO DIGITAL

8, 10 & 12 BIT RESOLUTION



CONNECTS TO RS-232

**ADC-16 A/D CONVERTER\* (16 channel/8 bit)....\$ 99.95**  
**ADC-8G A/D CONVERTER\* (8 channel/10 bit) \$124.95**  
 Input voltage, amperage, pressure, energy usage, light, joystick and a wide variety of other types of analog signals. RS-422/RS-485 available (lengths to 4,000'). Call for info on other A/D configurations and 12 bit converters (terminal block and cable sold separately). Includes Data Acquisition software for Windows 95 or 3.1  
**ADC-8E TEMPERATURE INTERFACE\* (8 ch)....\$ 139.95**  
 Includes term. block & 8 temp. sensors (-40° to 146° F)  
**STA-8 DIGITAL INTERFACE\* (8 channel).....\$ 99.95**  
 Input on/off status of relays, switches, HVAC equipment, security devices, keypads, and other devices.  
**PS-4 PORT SELECTOR (4 channels RS-422)....\$ 79.95**  
 Converts an RS-232 port into 4 selectable RS-422 ports  
**GD-422 (RS-232 to RS-422 converter).....\$ 39.95**

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

**\*FULL TECHNICAL SUPPORT**...provided over the telephone by our staff. Technical reference & disk including test software & programming examples in TurboBasic, GW Basic, Visual Basic, Visual C++ Turbo C, Assembly and others are provided.

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

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

**FREE INFORMATION PACKET**...use our 800 number, fax or E-mail to order, or visit our Internet on-line catalog  
 URL: <http://www.eeel.com>  
 Technical Support (614) 464-4470

**24 HOUR ORDER LINE (800) 642-7714**  
 The Standard American Express COD

Internet E-mail: [eesf1@eesl.com](mailto:eesf1@eesl.com)  
 International & Domestic FAX: (614) 464-4470  
 For information, technical support & orders  
**ENERGY CONTROL SYSTEMS, INC.**  
 390 South Main Street, Suite 604  
 Columbus, Ohio 43215-5493

Register	Data	VGA Value	Description	Register	Data	VGA Value	Description
VP	0XXXXXXX		Video Palette	VCR	9000XXXX	90000207	Vertical
VPAR	10000000		Palette address	VSWR	9100xxxx	91000003	Vertical
	2xxxxxxx		Resewed	VBSR	9200XXXX	9200001 E	Vertical Border Start Register
LORO	30000000		LCD Offset Register 0	VDSR	9300XXXX	9300001 E	Vertical Display Start Register
LOR1	31000000		LCD Offset Register 1	VDER	9400xxxx	940001 FE	Vertical Display End Register
BCR	4xxxxxxx	40000000	Border Color	VBER	9500XXXX	94000207	
CPC1	5xxxxxxx		Cursor Palette Color 1	VCSR	9600XXXX		Vertical
CPC2	6XXXXXXX		Cursor Palette Color 2	VCER	9700XXXX		Vertical
CPC3	7xxxxxxx		Cursor Palette Color 3		9800XXXX-9C00XXXX		
HCR	8000XXXX	80000336	Horizontal Cycle Register	SIR[0-7]	A000000X-A700000X		
HSWR	8100XXXX	8100002C	Horizontal Sync Width Register	SFR	B000000X		
HBSR	8200XXXX	82000072	Horizontal Border Start Register	SCR	B1 00000X		
HDSR	8300XXXX	83000080	Horizontal Display Start Register				
HDER	8400XXXX	84000300	Horizontal Display End Register	EXR	C00xxxxx		
HCSR	8500XXXX	85000324	Horizontal Border End Register	FSR	D000XXXX		
	8600XXXX	8600007f	Horizontal Cursor Start Register	FSR	E00XXXXX		
	8700XXXX		Resewed	FSR	F000XXXX		
	8800XXXX		Test Register				
	8900XXXX		Resewed				
	8C00XXXX		Test Register				

Table 3—ARM7500 video control registers and values control the screen as shown in Figure 3.

data presented to the video DACs is furnished by two DMA channels—one for video and one for cursor. These channels provide start (VIDSTART) and stop (VIDEND) addresses for defining a circular display buffer as well as a VIDINITA (and VIDINTB for dual-scan LCDs) for initializing the video DMA pointer after vertical flyback. The circular display is useful when operating in a full-screen terminal mode.

### VGA MODE SETUP

The ARM7500 provides for a wide range of programming possibilities in the values placed into the video registers (see Figure 3). Only a few sets of values are useful, however.

Since the ARM7500 uses the main memory for the CPU and screen, the two functions interact. So, there's a limit to the usable screen size and pixel depth before the CPU gets starved.

The typical VGA screen of 640 x 480 x 8 bits at 60 frames per second (i.e., 307,200 bytes per screen and a display memory bandwidth of 18 MBps) reduces the raw CPU performance by about 20% (49,000/38,000 dhrystones).

First, the VIDCLK is set up by programming the FREQCON register to generate 28.18 MHz.

The FREQCON register is an external 74HCT377 directly connected to a Chrontel CH9294 clock chip generating the VIDCLK to the 7500.

The VIDC register values are calculated from a direct description of the screen parameters

and written into the VIDC (see Figure 4).

The palette and the DMA channels for the screen and cursor are programmed, the video buffer cleared, and the cursor data area initialized. Before the screen can be used, the vertical flyback interrupt is initialized, and the DMA is programmed and enabled.

The VGA now shows a blank screen. To write onscreen, use a drawing library that includes some simple routines for painting characters and graphical primitives (e.g., line drawing and screen fills).

The 8 x 8 screen font for the diagnostics is similar to the fonts of a typical PC. To write a character onscreen, the proper font is located and expanded so each bit is represented by a byte.

### KEYBOARD INTERFACE

The ARM7500 plugs into a standard AT- or PC/2-type keyboard. Two open-collector lines, Kdata and Kclock, provide communication to this important device. An internal serial-to-parallel register and a simple sequencer provide the interface control.

Unlike the original PC serial keyboard, the AT keyboard has a reverse mode letting the computer program

keyboard functions by switching in three scan-code sets. VLSI chose scan-code set 3 because of its regularity and Ed Nisley's recommendation and code (ff 64. z i p in 1995 downloads).

In scan-code set 3, each key has a unique 8-bit make code. The break code is an FO byte followed by the key's make code.

The keyboard driver tracks the keyboard's state from the make and break data of the modifier keys. It uses this information to modify the key data when inserting it into the key buffer.

When the keyboard is initialized, the scan mode is set up and the software key buffer is zeroed. An interrupt routine attaches to the keyboard interrupt that reads the key codes, interprets them, and manages the keyboard state and keystroke buffer.

A routine expecting keyboard input calls a function that extracts a keystroke from this buffer.

### MOUSE INTERFACE

The mouse-interface hardware is the same as the keyboard interface but at a different address.

Of course, an interrupt handling mouse data differs. The mouse regu-

larly sends a 3-byte burst of overflow, sign, and button data-Ax and Ay.

This interrupt handler keeps a set of current values. If the information changes, it is put in a circular buffer of 16 sets of 4 integers—mouse and button states, and x and y positions.

```

HELP MENU
clear Clear screen
di <address> Disassemble instructions at <address>
dump <address> Dump memory address in hex at <address>
bounce Bouncing line
road Palette test
mouse Mouse test
fdtest Floppy test
hdtest Hard disk test
showpal Show palette
sound Sound test

```

Figure 2—This initial '7500 diagnostic screen display is incorporated into the console test program.

A program accessing mouse data gets its information from this buffer. If the read and write pointers are equal, a call to read the mouse buffer returns a -1.

The mouse test program displays a cursor onscreen that follows the movements of the mouse. If a mouse key is pressed, the cursor leaves a colored line.

## SOUND INTERFACE

The ARM7500 has an internal 8-bit companding DAC that can be steered to left and right channels. It also supports standard 16-bit stereo DACs. Some drivers (e.g., the dual sound DMA channels) are the same for either choice.

The nature of the data stored is quite different, however. Since I chose external digital DACs, the driver was written to support this interface.

When generating a 44.1-kHz sample rate, the clock to the DAC must be -1.4112 MHz (32 x 44.1 kHz). Dividing 32 MHz by 22 yields 1.455 MHz or 3% high (about a 1/2 tone error).

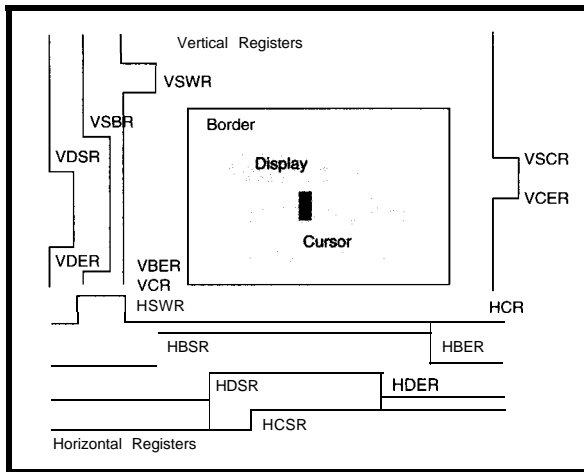


Figure 3—The VIDC registers easily relate to the video functions.

The sound channel has dual DMA pointers, enabling continuous sound data. Use Figure 4 to run the sound DMA. When the diagram calls to Write A, write the DMA-channel pointers SNDCURA and SNDENDA.

An LMC1982 between the DAC output and the input of the stereo power amplifier controls volume and tone. It is programmed by sending a serial bitstream with an open-drain

data line and a serial clock. Data is then strobed into the device.

To operate the sound channel, the sound frequency divider and control registers are set up for the sound type being played. The sound DMA irq routine is installed, the buffers zeroed, and the driver message queues initialized.

Four sound buffers help keep up with the sound DMA channel and are initially loaded with adjusted sound data. In the diagnostic, the sound data is 8-bit data expanded to 32 bits.

Thus, it takes 1024 bytes of input data to fill the playable buffer with 4096. As each DMA buffer is used up, the DMA loads a new buffer address, setting the interrupt.

The interrupt routine also sets a flag and updates pointers. When the last buffer is loaded, the DMA overruns. To reset the DMA int, the next DMA channel is programmed. When all the internal sound sample data is played, control returns to the diagnostic.

# Good, Fast, Low Cost: PICK 2

With PromICE, You CAN Have It All!

High performance memory emulation and debugging:

- Stable and reliable on today's embedded systems.
- New faster access speeds now standard.
- The best connection solutions for **TSOP**, PSOP and **PLCC** chips.
- Expanded Virtual UART support for industry-standard debuggers.

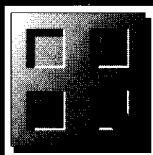
Ultra-Fast code downloads reduce development time:

- New high-speed download support for Windows NT
- 90 **KBytes/Second** over a PC parallel port.
- Low-cost Ethernet support for UNIX systems.

New lower Prices for 1997:

- 128 **KByte PromICE** now just \$495.
- Source-level debugging systems at a fraction of an **ICE's** cost.

Today's PromICE really is "better, faster, less expensive". Call 1.800.PROMICE (776-6423) for more information, or visit us at [www.gei.com](http://www.gei.com).



Voice: (614) 899-7878 • Fax: (614) 899-7888 • email: [info@gei.com](mailto:info@gei.com)  
921 Eastwind Drive • Suite 122 • Westerville, Ohio 43081 • USA  
US and Canadian Sales: 1-800-PROMICE (776-6423)

## FLOPPY INTERFACE

The floppy attaches to a standard I/O Combo chip (FDC37C-663 or 665). It's programmed exactly as in a PC but at different addresses. In particular, the successive addresses are on word boundaries, so 8- and 16-bit operations are supported.

The ARM7500 has a special chip select (COMBOCS at physical address 0x03010xxx) to select this part as well as one that mimics the floppy DMA Acknowledge (CDACK at address 0x03012000).

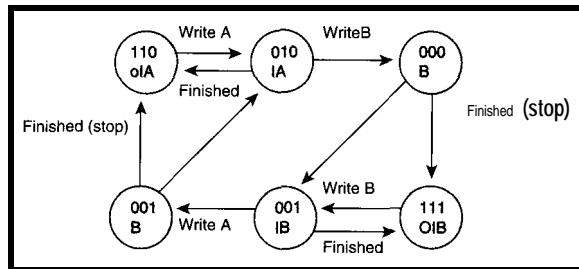


Figure 4-The sound DMA has dual data pointers. The program interacts with the hardware to maintain continuous sound output.

## IDE INTERFACE

The RC7500 sports two IDE connections-IDE1 connects to PC Combo and is located within its address space,

and a separate IDE2. Both have separate address spaces for the Western Digital Hard Disk register set and the extra floppy registers for the hard disk.

## DOES THE SHOE FIT?

Obviously, this information merely scratches the surface of what the ARM7500 is all about.

If your application is along the lines of an Internet appliance, medical instrumentation (e.g., EKG display), and GPS or airport display, the '7500 is a chip you should check out. ☐

Art Sobel is the hardware applications manager of embedded products at VLSI Technology. He has spent 24 years designing disk-drive electronics and controllers, laser interferometers and printer controllers, many controller chips, and speech synthesizers. You may reach Art at [sobel@sanjose.vlsi.com](mailto:sobel@sanjose.vlsi.com).

# LOGICAL

Versatility in device programmers DOS, Windows 3.1  
Windows 95 Windows NT Compatible



Start programming devices today with the lowest cost and highest performance CERTIFIED programmers. Enjoy a no hassle user interface for ALL versions of Windows and DOS. Works with any PC of any speed without a hitch. Device libraries added in less 2 hours to our Web customer support section. Unique programming head options for gang programming most microcontrollers and memory devices. Direct Docking Gang TSOP, QFP, PLCC, DIP... programming heads. Evaluate a unit today with 100% satisfaction guaranteed or YOUR MONEY BACK!

(no penalties or restocking fees if unit is returned).

**Gang any Device, in any Socket Type  
with PC based or Stand Alone unit**

Call Today in USA 800-331-7766  
303-733-6868 or Visit our Home Page:

[www.logicaldevices.com](http://www.logicaldevices.com)

## SOFTWARE

75demon.zip and 75diag.zip offer complete source code at [www.circellar.com](http://www.circellar.com). RiscBSD and GNU cross-development software is at [www.ph.kcl.ac.uk/~amb/riscbsd/docs.html](http://www.ph.kcl.ac.uk/~amb/riscbsd/docs.html) or <ftp://netbsd.org>.

## SOURCES

ARM7500 Spec Sheet  
[www.arm.com/Pro+Peripherals/ASSPs/7500](http://www.arm.com/Pro+Peripherals/ASSPs/7500)

RC7500, ARM7500 chips  
VLSI Technology  
18375 S. River Pkwy.  
Tempe, AZ 85284  
(602) 752-6630  
Fax: (602) 752-6001

ARM7500 chips  
Cirrus Logic, Inc.  
3100 W. Warren Ave.  
Fremont, CA 94538  
(510) 623-8300  
Fax: (510) 226-2180  
[www.cirrus.com/prodtech/ov.netmobile/ps7500.html](http://www.cirrus.com/prodtech/ov.netmobile/ps7500.html)

## IRS

410 Very Useful  
411 Moderately Useful  
412 Not Useful



40

How to Buy

49

An In-depth Look at  
Razafindralandy

53

To ROM or Not to ROM  
Rajesh Chhabra

60

Rising Star  
Regional Present: Slimline  
Razafindralandy



Photo courtesy of M-Systems, Inc.

## SINGLE-BOARD COMPUTER

The **DI05330**, an industrial computing board for process- and motion-control applications, is available as a stand-alone board or powered by the company's **CardIO** credit-card sized PC/AT motherboard. It offers operation at the full industrial temperature range, (-40 to +85°C), a 4" x 7" footprint, access to industry-standard software (DOS or Windows preinstalled, if requested) and onboard interfaces. Targeted embedded applications include factory floor automation, hand-held instruments, and test equipment.

**Onboard** features include serial and parallel ports, interfaces for graphics, hard and floppy disk drives, as well as mouse and keyboard controllers. It has up to 4-MB DRAM and resident BIOS in 256-KB flash memory. Also included are an LCD interface with backlight control circuitry, programmable watchdog timer, four analog-input channels, and a power-management controller.

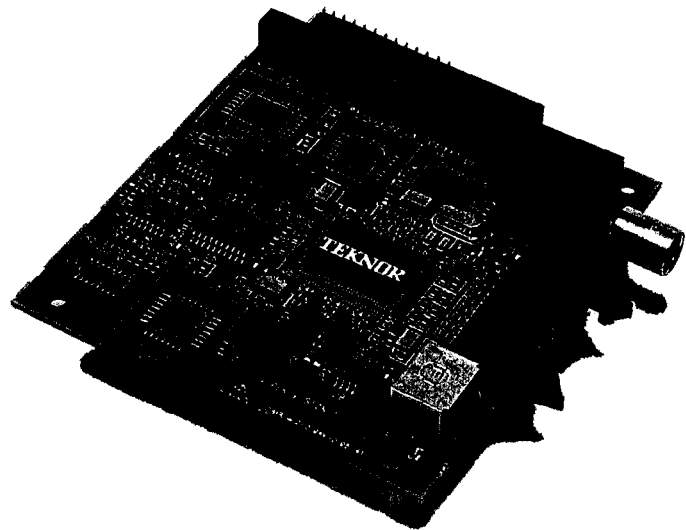
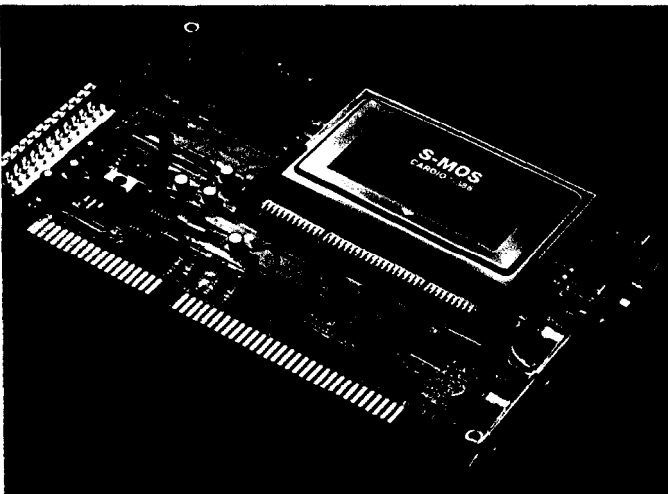
The **CardIO** is a fully functional PC/AT motherboard, available in '486 and '486DX4 platforms with speeds up to 100 MHz and onboard memory of up to 16 MB.

The DI05330 and **CardIO** are sold separately. The DI05330 sells for less than \$300 in quantity. The **CardIO** starts at \$800 in quantity. Pricing for evaluation kits starts at \$300, depending on components.

### S-MOS Systems

150 River Oaks Pkwy.  
San Jose, CA 95134-1951  
(408) 922-0200 • Fax: (408) 922-0238  
[www.smos.com](http://www.smos.com)

#506



## REAL-TIME VIDEO INTERFACE MODULE

The VIPer Vision TEK-380 interfaces automatically to various video standards to accommodate noise-free video-display and capture applications. Designed to complement the company's VIPer SBCs, the card features up to six composite or three S-video inputs, NTSC, PAL, and SECAM compatibility, hue saturation, brightness and contrast control, real-time image resizing, and onscreen positioning in a PC/104 form factor. Typical applications include automated shop floor equipment, surveillance systems, personal identification systems, in-vehicle readers and scanners, and electronic kiosks.

Other features include CCIR or square pixels for easier image processing, linear zooming with interpolation for smoother edges, full cropping control prior to capture, and the ability to save captured images to disk. The card uses the VIPer industrial SBC's internal video circuitry to produce full real-time video without burdening the system bus for additional bandwidth. Thus, the entire system can run at maximum capacity at all times.

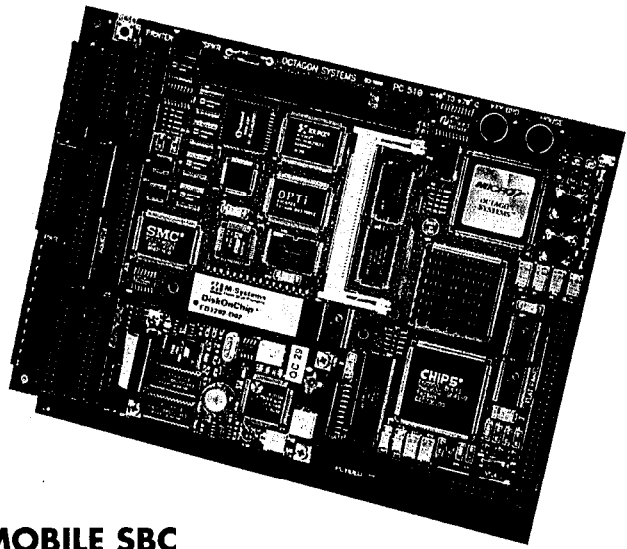
The VIPer Vision TEK-380 comes standard with one BNC connector for composite video, one 4-pin miniDIN for S-video input, and one 14-pin header to handle multiple inputs. Video output is via a 26-socket header which interfaces to a VIPer industrial board. Both standard and customdesigned software drivers are provided. The unit sells for \$395.

Teknor Industrial Computers, inc.  
7900 Glades Rd.  
**Boca Raton, FL 33434**  
(407) 883-6191  
Fax: (407) 883-6690

#507

# Nouveau PC

edited by Harv Weiner



**MOBILE SBC**

The PC-510 single-board computer combines a 133-MHz 586 processor, six serial ports, a GPS interface, advanced video, and 48 lines of DIO on a 5.75" x 8" form-factor board. It is designed for rugged mobile communications, data acquisition, and industrial control applications, and it features an MTBF of 13 years. The PC-510 supports LCD and Et flat-panel displays, The on-card 65550 video chip acts as a graphics accelerator to support

real-time video. Because the video circuitry operates on the Local bus at full processor speed, high-performance programs like Windows execute very rapidly. As well, 2 MB of video RAM is provided to accommodate a high-resolution display monitor. Power-management functionality is also included. The board also includes a PC/104 interface, IEEE 1284 multifunctional parallel port, floppy- and hard-drive interfaces, keyboard, speaker and mouse ports, watch-dog timer, real-time clock, 2-MB flash disk, and 1 MB of EDO DRAM (expandable to 33 MB).

The PC-510 contains DOS 6.22 in ROM, as well as diagnostic software to test and verify on-card I/O and memory functions. DOS applications can be stored directly in the resident flash memory, eliminating the need for a hard drive. The card also supports other operating systems, such as Windows, Windows 95, Windows NT, and QNX.

The PC-510 can operate either in stand-alone mode, or it can be expanded via its PC/104 connector. The unit sells for \$995 in small quantities.

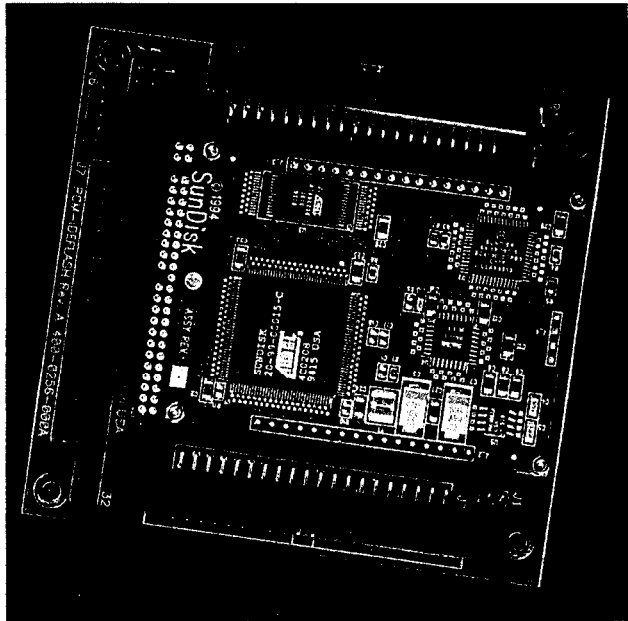
**Octagon Systems**  
 65 10 W. 91 st Ave. • Westminster, CO 80030  
 (303) 430-1500 • Fax: (303) 426-8126 **#508**

**MASS-STORAGE MODULE**

The **PCM-IDEFLASH-0** is designed for embedded systems requiring low power, high shock and vibrational resistance, instant access to data, and full compatibility with rotational disk drives. Its PC/104 form factor provides up to 84 MB of formatted flash disk storage to replace conventional disk drives in harsh environments. Applications include program and data storage for data collection and logging, diagnostics, process variables, and setpoints.

The board provides solid mechanical and electrical mounting, permitting a user to install a 1.8" FlashDrive and cable it to a host computer's IDE interface. The drive plugs into a 2-mm connector and fastens securely to the PC board. Since it appears as a standard IDE interface, no special software drivers or utilities are required. The FlashDrive products are 100% compatible with DOS, DOS applications, and other operating systems supporting IDE disk drives. It also operates with QNX, OS/9000, Lynx, and other real-time embedded OSs that interface to IDE drives.

The PCM-IDEFLASH-0 comes in an ADP-FLASH version if PC/104 stack mounting is not desired. It has a 4-pin power connector rather than PC/104 connectors. The PCM-IDEFLASH-0 sells for \$50.



**WinSystems, Inc.** • 715 Stadium Dr. • Arlington, TX 76011  
 (817) 274-7553 • Fax: (817) 548-1358 • [www.winsystems.com](http://www.winsystems.com) **#509**

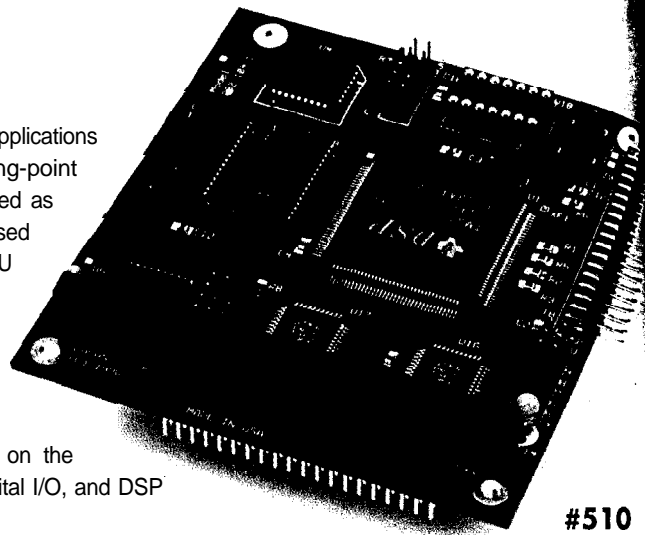
*Nouveau* **EMBEDDED PC**

DSP BOARD

The Model **C32-104** is designed for embedded applications requiring the computational and I/O capabilities of a floating-point DSP, as well as for DSP algorithm development. It can be operated as a PC/104-bus expansion board or a stand-alone unit, or it can be used to control other boards via the PC/104 bus in a system without a host CPU board. This last mode permits the creation of a low-cost embedded DSP computer with the unit performing functions normally done by the 80386 (or higher) CPU board. These functions include the controlling of PC/104 video, RS-232 serial port, and analog I/O boards.

The unit is based on the Texas Instruments TMS320C32 floating-point DSP operating at 50 MHz, for up to 50-MFLOPS performance. Included on the board are 256 KB of zero-wait-state SRAM, 512 KB of flash memory, digital I/O, and DSP serial port expansion.

A DSP software-development package containing an assembler, debugger, application examples, and flash-memory programming utilities is included with the board. Price including software is \$499 in small quantities.



#510

Dalanco **Spry** • 89 **Westland** Ave. • Rochester, NY 14618  
 (716) 473-3610 • Fax: (716) 271-8380 • [www.vivanet.com/~dalanco](http://www.vivanet.com/~dalanco)

# Nouveau IPC

## PC COMPATIBLE!

### SINGLE BOARD COMPUTERS

Just connect a keyboard, monitor/LCD, a disk drive and you're ready to run. Or forget the drive and boot directly from a Flash disk. Add PC/104 Modules for Fax/Modem, SCSI, Ethernet, Digital/Analog I/O, and PCMCIA. Great for Point Of Sale and Web Browsers/Servers. Prices start at \$200.00 Qty. 1.



- \* Wide CPU Selection: 386SX, 486DX, DX2, DX4, 586, Pentium.
- \* All SBCs have Real Time Clock, Serial, Parallel, IDE, and Floppy.
- \* On Board Watchdog Timer.
- \* BIOS with Power Saving Green Mode.
- \* Wide Bus Selection: PC/104, ISA, PCI.
- \* 10.4" TFT super bright LCD Panel Kits.
- \* Hardware and Cable kits included for most boards.

1985-1997  
 OVER  
**12**  
 YEARS OF  
 SINGLE BOARD  
 SOLUTIONS

## EMAC, inc.

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

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 575  
 MVDISK2 144m \$150  
 MVDISK3 576m \$195 **\$75**

Quantity discounts!

## DOS IN ROM!



## \$95 EPROM PROGRAMMER

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

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

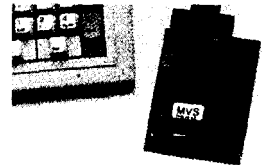
FREE SHIPPING IN U.S.

5 YEAR LIMITED WARRANTY



Box 850  
 Merrimack, NH  
 (508) 792 8507

## 8088 SINGLE BOARD COMPUTER





The fact that FTL can use the native OS's file system makes this solution stand out from all others. Also, it's significant that FTL implements a fully read/writable disk.

Many programs let you use a flash memory array as a Write Once Read Many (WORM) device. But, they require special utilities to update the disk image, which is usually a slow process.

Microsoft's FFS I and II drivers were an early attempt to provide hard-disk emulation. But, these solutions replace the standard file system that's part of the OS. So, standard OS disk management and diagnostic utilities can't be used on a flash disk managed by an FFS driver.

In addition, the FFS II linked-list approach is plagued with performance and reliability problems. The medium is easily corrupted by power failures or other events that interrupt a write cycle.

The FTL specification, coauthored by M-Systems and SCM, provides a uniform, high-performance, and robust solution for

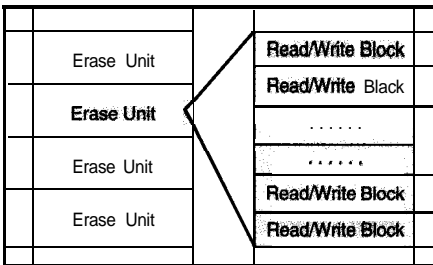
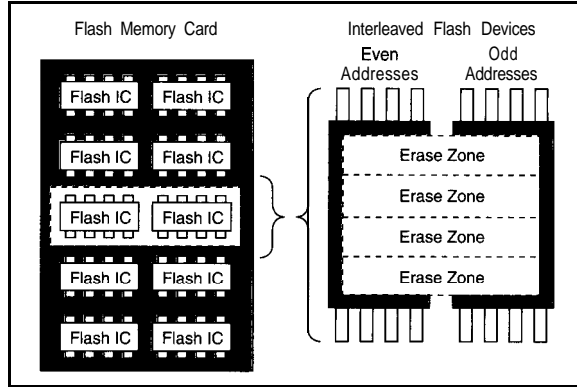


Figure 3: Each erase unit is divided into evenly sized blocks, each about 5 12 bytes long.

Figure 2: Erase units are composed of individual erase zones. For example, two Intel 28F008 chips connected in parallel yield a 128-KB erase unit.



working with flash PC Cards. The IP rights associated with the patent granted to M-Systems were released into the public domain for designs using linear-flash PC Cards and Miniature Cards. A variety of companies and user groups can therefore provide their own FTL implementations.

### WORKING WITH FLASH MEMORY

Flash memory offers some attractive features for mass data storage. They are nonvolatile—the data is retained indefinitely without any power to the flash components. No back-up batteries are needed.

Flash is low cost compared to other battery-backed solid-state solutions. It consumes low power and takes up little space, so it's ideal for mobile and hand-held applications. Also, it's solid state, so it can work in harsh, rugged environments where mechanical disks are unsuitable.

Managing data on flash memory is complicated, however, by the inherent constraints of the flash technology—most importantly, the nonrewritability of data. With flash, you can't write over existing data without a slow erase cycle.

In many flash components, erase blocks are large (typically 64-KB blocks), further

complicating data updates. The blocks are much larger than the units of data (usually 512-byte sectors) stored on the medium.

Unless handled properly, this problem complicates data updates. Larger blocks of unrelated data must be rewritten to update a single sector.

### LIMITATIONS OF FLASH

The number of times flash can be written and erased is limited and depends on the specific flash technology. Typically, it's about 1 million times per block.

A region of flash close to its cycling limit usually displays sporadic write failures that become more frequent. Eventually, the sector is no longer erasable or writable.

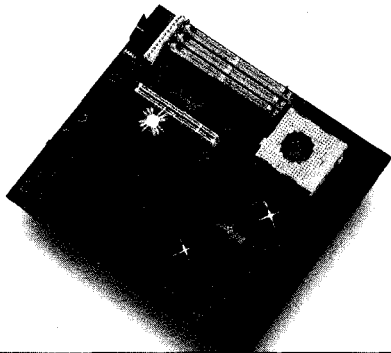
Flash cells can be accidentally overprogrammed or overerased by incorrect programming. When this occurs, the flash usually won't respond to programming for a period of time or it responds very slowly. This condition can usually be reversed, but the cell's life is shortened.

### FTL Definitions

- Block**—This sector-sized (5 12 byte) unit of data stores information, control or data, on the medium. It is a subdivision of an erase unit.
- Block Allocation Map (BAM)**—This FTL control structure stores Block Allocation Information (BAI) about blocks on the medium. It includes a 32-bit entry for each block on the medium.
- Erase Unit (EU)**—This area of the flash medium is handled as a single erasable unit by FTL, although it may contain one or more erase zones. This area is determined by the hardware configuration of the flash and is identified during media formatting.
- Erase Unit Header (EUH)**—This header contains information specific to the erase unit and global information about the entire FTL partition.
- Erase Zone**—This area of flash must be erased as a single unit due to the characteristics of the flash chip.
- Logical Address**—This address is based on accessing the medium in logical Erase Unit order.
- Logical Erase Unit Number (LogicalEUN)**—This logical number is assigned to an erase unit by the FTL. FTL assigns logical numbers to erase units in order to remap the ordering of the physical medium and simplify recovering superseded areas.
- Partition**—This region of the flash medium is dedicated for a specific use. The medium can contain a partition at the beginning that contains binary information, an FTL-handled partition located after the first partition, and a final partition for storing additional binary information. Once the medium is formatted, the physical starting address of each partition remains fixed.
- Physical Address**—This address is based on accessing the medium in Physical Erase Unit order [i.e., the hardware address of a location in the flash array].

- Physical Erase Unit Number (PhysicalEUN)**—This number is given to an erase unit based on its location on the physical medium. This unchanging number is implied by the erase unit's position. The erase unit at the beginning of an FTL partition is known as the first physical erase unit. If the partition begins at physical address zero, the first physical erase unit is number zero.
- Reclaim**—Also known as garbage collection, this procedure recovers blocks that were deleted or contain superseded data for reuse. It also preserves valid data within the erase unit being reclaimed.
- Read/Write Block**—see Block.
- Replacement Page**—This alternate VBM page contains entries that override the values in the original VBM being replaced.
- Transfer Unit**—This erase unit is reserved for storing read/write blocks of valid data from an erase unit being reclaimed. Transfer units are not included in the formatted size of the FTL partition presented to the host file system.
- Virtual Address**—This address is recorded in a read/write block's allocation information (BAI), representing where the stored data appears in the virtual image presented to the host system. It is calculated by multiplying the virtual block number (e.g., the sector number) by the block size (5 12 bytes).
- Virtual Block**—This unit of information is used by the host file system above FTL for reading and writing data to the medium. It's usually called a sector when dealing with file systems.
- Virtual Block Map (VBM)**—This array of 32-bit entries maps a virtual block number to a logical address on the medium.
- Virtual Page Map (VPM)**—This structure maps the locations of VBM pages. It is never stored on the medium. Instead, it is stored in the host's system memory and rebuilt every time a new card is inserted or power is cycled.

. 7 ISA slots  
 • 486 support



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

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

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

**DFI**<sup>®</sup>

## FLASH IS COMPLICATED

Because of the nonrewritability of flash, data must be organized via flash file systems. Damaging or corrupting the data (i.e., the low-level format) may mean user data can no longer be accessed.

Since an accidental or deliberate power failure, often due to prematurely removing the flash card, is possible anytime, writing to the flash must be done in a way that ensures no loss of existing data.

Even if no flash-hardware errors occur, the data and recording format must be coherent at all stages of writing. Also, manufacturers use incompatible programming algorithms to control the flash.

## FLASH FILE SYSTEM FUNCTION

A flash file system is a software driver that makes flash memory emulate a disk drive. It lets the developer use a common, well-understood mechanism to store data on a nonvolatile solid-state medium. The resulting flash disks may be interchanged with mechanical disk drives, adding flexibility to the design and debug process.

A well-written flash file system emulates a disk so transparently that the user and system cannot functionally distinguish it from a mechanical disk drive. However, it must perform low-level operations to accomplish this as well as overcome the constraints of flash components.

I examine these operations in detail, but these features include:

- mapping OS model (disk sectors) to physical model (flash blocks) as in Figure 1
- managing the mapping tables
- maintaining flash erase operations in the background to optimize performance
- wear-leveling for increased flash-media endurance

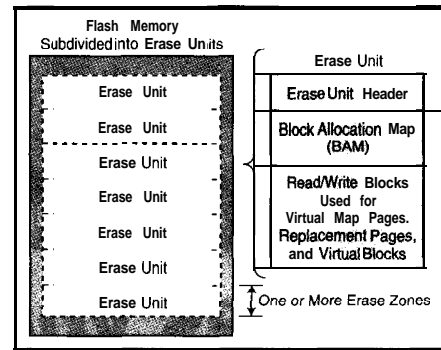


Figure 4: Each erase unit has an erase-unit header (EUH) that contains information about the specific unit as well as information about the entire medium.

- detecting errors for mapping bad or worn-out flash blocks
- protecting existing data and directory structures for reliability
- implementing different programming algorithms for specific flash components

Before getting into the structure of the FTL data, look at some terms of the specification (see the sidebar "FTL Definitions").

## FTL DATA STRUCTURES

An OS file system (not a flash file system) randomly updates any block on the system's storage medium. But, unless flash is in its erased state, it cannot accept new data.

FTL delivers this capability to higher level software layers by remapping requests to write blocks to unallocated or free areas of the medium and invalidating the area that previously contained the block's data. It also records where the remapped block is placed for subsequent read accesses.

In effect, FTL presents a virtual-block storage device to the higher level software layers. Virtual block size can be determined when the storage medium is formatted, but it's normally set to 5 12 bytes to emulate the standard hard-disk sector size.

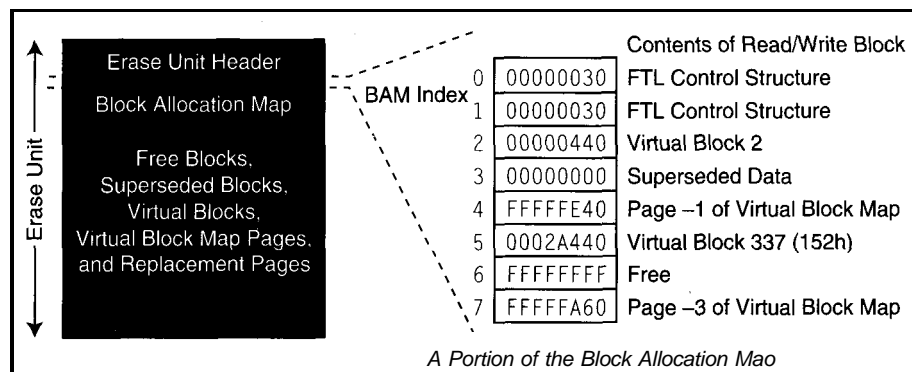


Figure 5: In this particular block allocation map (BAM), the read/write block size is 5 12 bytes. Also, the FTL partition does not store checksums, CRCs, or ECCs for the virtual-block data.



# The PC/104 Motion Control Experts

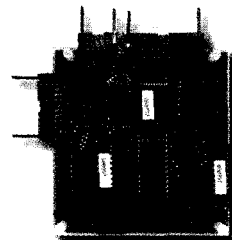
Need motion control within your PC1104 application? Overwhelmed by the number of products & vendors out there? Looking for a motion control specialist instead of just another PC/104 vendor with one motion controller?



Model 5912 Encoder Interface

Look no further. With more than a decade of experience supporting the motion control needs of OEM customers, Tech 80's family of

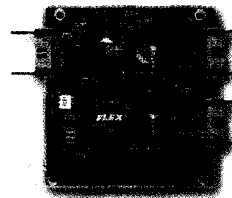
PC1104 modules can meet the encoder interfacing and servo & stepper control demands of your embedded application.



Model 5928 Servo Controller

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

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



Model 5936 Stepper Controller

For more information on our PC/104 motion control products or to speak with a sales engineer regarding your current project,

please contact us at 800/545-2980 or visit us at:

[www.tech80.com/cc/pc104.html](http://www.tech80.com/cc/pc104.html)



The People in Control of Motion™

Minneapolis, Minnesota USA  
800/545-2980 • 612/542-9545 • 612/542-9785 (fax)  
[www.tech80.com](http://www.tech80.com) • [info@tech80.com](mailto:info@tech80.com)

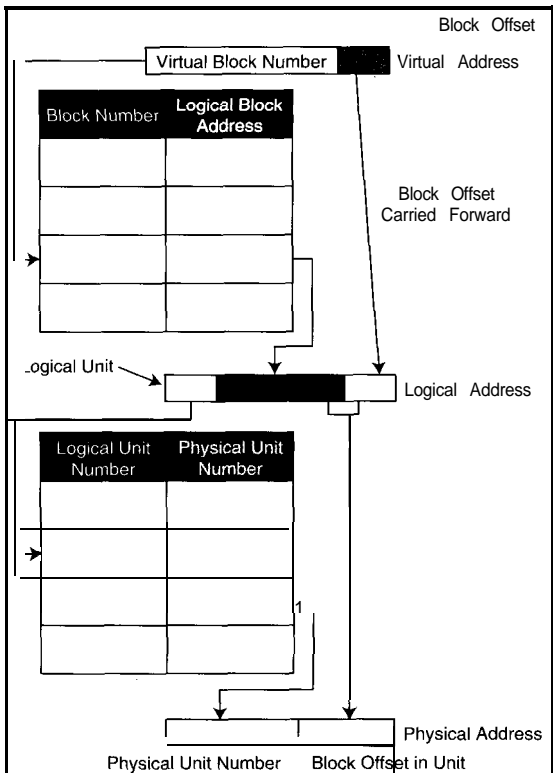


Figure 6: The mapping functions can be defined as a set of table-lookup functions based on the number of the sector that the operating system wants to access.

information about the state of the entire flash medium. All EUHs on the medium are identical except for their LogicalEUN field values.

## TRACKING THE BLOCKS

Each erase unit contains allocation data for the unit's read/write blocks. For each read/write block, a four-byte value (i.e., the block-allocation information (BAI)) tracks the block's current state.

A read/write block in an erase unit can be in only one of four states-free, deleted, bad, or allocated. It can also have one of four types of information-FTL control structures, virtual-block data, virtual-block map pages, or replacement pages.

The encoded BAI tracks the block's content and its state. The block allocation map (BAM) is normally stored immediately after the EUH. Some flash contains hidden areas that store this data instead of using main storage space. The type of storage area, main or hidden, is defined in the EUH.

Figure 5 shows the contents of a BAM. Each map entry describes the contents of a corresponding read/write block. This example uses a 512-byte block size, and the FTL partition doesn't store checksums, CRCs, or ECCs for virtual-block data.

The first two read/write blocks store the EUH and BAM. The third (bytes 1024-1535 of the erase unit) holds data for the third virtual block used by higher level software layers. The next block (bytes 1536-2047) holds superseded data, and the block following it has a virtual-block map page.

## VIRTUAL-BLOCK MAP

The BAM tables carry enough information for the FTL algorithm to track the virtual blocks and control structures on the medium. However, the algorithm required to track the locations would be slow since it would rescan the entire medium every time it looked for the location of a virtual block.

Instead, FTL incorporates an additional map on the medium called the virtual-block map (VBM). The VBM comprises an array

## ERASE ZONES AND UNITS

Depending on its technology, each flash IC on a PC Card is divided into one or more erase zones of equal size. Each erase zone is the minimum contiguous area that can be erased in a single operation.

If 8-bit devices are interleaved to provide 16-bit storage, the corresponding physical zones on two adjacent devices combine as a single erase zone. One device gives even addresses, and the other, the odd ones.

An erase unit is a multiple of one or more contiguous erase zones. Its size is set when the medium is formatted (see Figure 2).

For example, if the flash components are Intel 28F008 FlashFile chips, every chip has sixteen 64-KB erase zones. If two chips are connected on a 16-bit data bus, the erase-unit size is 128 KB.

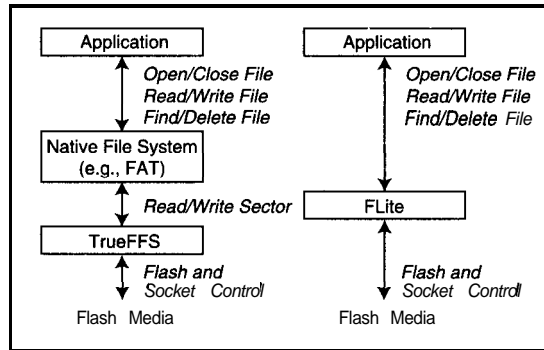
## ERASE-UNIT HEADER

FTL divides every erase unit into one or more equally sized read/write blocks (see Figure 3). Each read/write block is the same size as a virtual block (or DOS data sector) used by the host file system.

As shown in Figure 4, each erase unit contains an erase unit header (EUH) normally located at the beginning of the erase unit (i.e., offset zero). However, it can be mapped to a different offset.

The EUH contains specific information about its erase unit as well as global

Figure 7: **TrueFFS** and **FLite** have different interfaces with their application but maintain the same low-level functionality.



of four-byte entries, each corresponding to a virtual block and containing its logical address. FTL uses the virtual-block number from the host as an index into the VBM.

As virtual blocks get assigned to physical blocks, the appropriate entry updates in the VBM, which is on the medium. So, when an entry is updated to show it has moved, the physical block containing the VBM needs to be changed, too. This situation would become a performance issue if it wasn't addressed properly, as I'll discuss.

The VBM storage space is allocated when the medium is formatted. A mechanism differentiates between blocks containing data and those holding VBM pages. Blocks with VBM information are treated as virtual blocks with negative numbers, while virtual blocks have positive numbers.

REPLACEMENT PAGE

Replacement pages solve any performance degradation caused by updating the VBM for every virtual block that changes physical location on the medium. As its name implies, a replacement page contains alternate VBM entries for virtual blocks.

VIRTUAL PAGE MAP

The virtual page map (VPM), the last map generated by FTL, is never held on the medium. Instead, it is reconstructed from the VBM contents, which are always on the medium, every time the system is powered up or a new card is inserted.

Each entry in the VPM tracks the location of the appropriate VBM page. Since the VBM is stored on the medium, these pages move around as they are updated.

The VPM provides the entry point into the VBM. Without the VPM, every access to the medium requires a complete media scan to find the appropriate VBM.

GETTING A PHYSICAL ADDRESS

Figure 6 shows how a virtual-block number gets translated into a physical address with the required data. It shows some of the key features of the FTL algorithm. All virtual

blocks need the same number of address translations to get to the physical address (i.e., accessing two translation maps).

Because of pointer arithmetic and structures sized as powers of two, the arithmetic involves very fast simple shift-left and shift-right operations. Information in the maps can always be reconstructed from redundant information on the medium.

RECLAIMING UNITS AND BLOCKS

The ability to reclaim superseded blocks enables FTL to provide a solution that works as a writable disk. When the medium is formatted, at least one erase unit is set aside (i.e., a transfer unit). When the medium no longer has any free blocks, a garbage-collection cycle is executed.

The transfer unit is always held in an erased state, ready to receive data. During this cycle, the medium is scanned for an erase unit that has garbage blocks.

When a unit is found, blocks with good data are transferred to the transfer unit. To show that it now contains valid data, the transfer unit's EUH is updated. The logical-EUN field from the unit that had the garbage data is copied and placed in the transfer unit's EUH.

Once the data is in the transfer unit, the old unit is erased and it becomes the new transfer unit. When data was copied into the transfer unit, the blocks containing garbage data were not copied. Thus, when it becomes a new unit, it has blocks free to accept new data.

IMPLEMENTATION

FTL is a specification for a set of data structures that provides a mechanism for tracking the location of data on flash media. However, this specification doesn't address all the implementation issues involved in providing a fully functional driver

that can provide a robust solution that interfaces with an OS or application.

Let's look at TrueFFS and FLite, which have been recognized as the leading implementations of the FTL specification.

TrueFFS AND FLite

M-Systems has two different forms of FTL available to developers for use with flash memory components or cards.

TrueFFS is a driver intended for an OS with an existing file system. It typically comes in a binary format compatible with the specific OS and is available for many standard OSs (e.g., DOS, Windows 3.x, Windows 95, Windows CE, QNX, VxWorks, pSOS+, etc.).

FLite, which stands for FAT/FTL Lite, is a version of TrueFFS targeted to applications with no built-in file system (i.e., they need a file system and the FTL data format). It's customizable and provided with sourcecode.

FLite includes DOS FAT file functions, so an application lacking native file-system capability can write files directly to the flash media in a DOS-compatible file format. The combination of DOS FAT file and FTL compatibility ensures easy data interchange between a personal PC and an embedded system.

This capability may be particularly important when using removable flash media (e.g., PC Cards and Miniature Cards). Both FLite and TrueFFS use the same FTL algorithms and technology.

Where does TrueFFS fit into the system? TrueFFS and FLite are generally used as block device drivers which sit under the

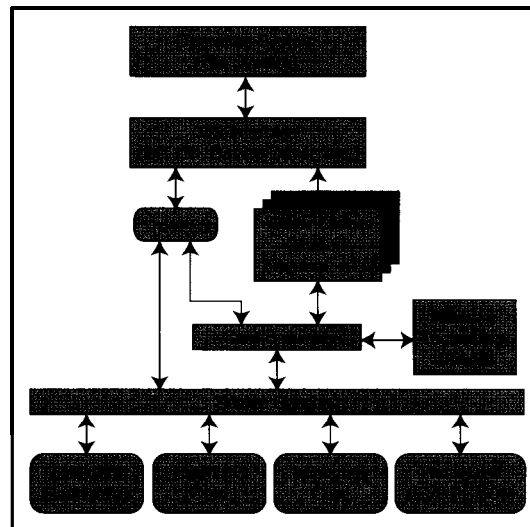
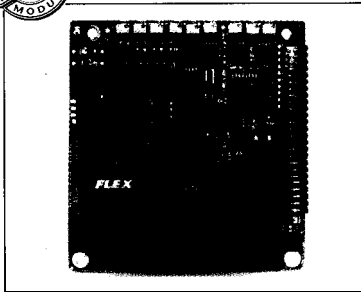


Figure 8: The **TrueFFS driver** is part of a software stack that provides a data path from the flash media to the OS.



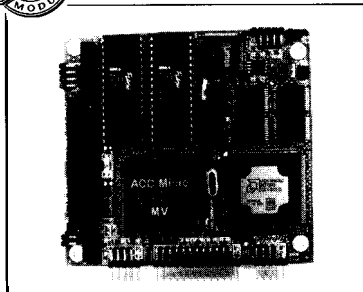
# Sets The Pace In PC/104 Data Acquisition

Scan 16 Channels...  
Any Sequence...  
Any Gain...  
500 kHz



DM6420 500 kHz Analog I/O Module  
with Channel-Gain Table and FIFO

With Companion  
Am5x86™ 133 MHz  
PC/104 cpuModules



The CMV586DX133 offers  
versatile embedded functionality

Our PC/104 and ISA Bus  
Product Lines Feature  
Intelligent DAS Cards With  
Embedded PC and DSP,  
Analog and Digital I/O, CPU,  
Shared Memory, SVGA, PCMCIA,  
CAN Bus and GPS Modules

**rttd** Real Time Devices USA, Inc.  
200 Innovation Boulevard  
State College, PA 16804-0906 USA  
Tel: 1(814) 234-8087 • Fax: 1(814) 234-5218  
URL: www.rtdusa.com  
E-Mail: sales@rtdusa.com

RTD Scandinavia Oy  
Helsinki, Finland  
Tel: 358-9-346-4538  
Fax: 358-9-346-4539

RTD is a founder of the PC/104 Consortium

native file system of the OS (see Figure 7).  
In the case of Ftite, there may not be any file  
system. So, Ftite optionally provides one.

Block device drivers provide:

- compatibility with many file systems
- transparency in operation
- compatibility with all file and disk utilities for the OS
- compactness

In Figure 8, you see how TrueFFS fits into an OS structure. It may interface to a number of different flash storage devices.

These may be removable flash cards, an onboard resident flash array, or a separate flash-disk board (e.g., ISA- or PC/I O&bus boards). In any case, TrueFFS remains unchanged, and the specific interface is dealt with in a socket-services layer under TrueFFS.

In some cases, a Card Services Software manager arbitrates the socket's operation. This often happens when a system has multiple cards (e.g., flash, modems, LAN, etc.), and a specific version of TrueFFS is necessary.

TrueFFS includes standard memory technology drivers (MTDs) for common flash devices used for flash-disk applications. Although these include devices from Intel, AMD, and Samsung and their compatible devices, there may be a need to add more external MTDs to support other devices.

External MTDs are usually added as plug-in drivers via a standard interface to the Card Services layer of the software. With Ftite, the MTDs are more often added via source code and then implemented as a single monolithic driver.

## REFINED ENOUGH?

FTL is a robust, industry-standard, proven flash data format standard that has been widely used and adopted. It fully emulates a hard disk, making flash easy to read and

low-cost, low-power, rugged, reliable storage medium of choice for hand-held, portable, embedded computer applications. It's been used in OSs and is available from third-party suppliers for all other platforms.

TrueFFS and Ftite are the leading implementations of FTL. They both incorporate the FTC data format standard to ensure interoperability across diverse platforms.

Over the past six years, they have been refined in different operating architectures

PDA's,

TrueFFS-based type of a flash-EPC

RazDgn is the customizing manager for M-Systems. He is currently responsible for custom applications like technical support, and system integration line. He is currently a BSEE from Tel-Aviv University. You may reach him at [raz@ccm.msyscal.com](mailto:raz@ccm.msyscal.com).

## SOURCES

FTL Specification, PC Card Standard, Media Storage Formats Specification  
PCMCIA

2635 N. 1st St.  
San Jose, CA 95 134  
(408) 433.2273  
Fox: (408) 433.9558

## TrueFFS, FLite

M-Systems, Inc.  
4655 Old Ironsides Dr.  
Santa Clara, CA 95054  
(408) 654-5820  
Fax: (408) 654-9 107  
[info@ccm.msyscal.com](mailto:info@ccm.msyscal.com)  
[www.m-sys.com](http://www.m-sys.com)

## SwapFTL (V.3.0 and up)

SCM Microsystems, Inc.  
13 1 Albright Way  
Los Gatos, CA 95030  
(408) 370.4888  
Fax: (408) 370.4880  
[pccard@scmmicro.com](mailto:pccard@scmmicro.com)  
[www.scmmicro.com](http://www.scmmicro.com)

## CardWizard, CardWorks

SystemSoft, Inc.  
2 Vision Dr.  
Natick, MA 01760  
(508) 65 1-0088  
Fax: (508) 65 1-8 188  
[wizard@systemsoft.com](mailto:wizard@systemsoft.com)  
[www.systemsoft.com](http://www.systemsoft.com)

## PCM+

Phoenix Technologies Ltd.  
2575 McCabe Way  
Irvine, CA 92714  
(7 14) 440-8000  
Fax: (714) 440.8300  
[pcmplus@piltid.com](mailto:pcmplus@piltid.com)  
[www.piltid.com](http://www.piltid.com)

## TrueFFS

Annasoft  
1 1838 Bernardo Plaza Ct.  
San Diego, CA 92 128-24 14  
(619) 673.0870  
Fax: (619) 673-1 432  
[info@annabooks.com](mailto:info@annabooks.com)  
[www.annabooks.com](http://www.annabooks.com)

## IRS

413 Very Useful  
4 14 Moderately Useful  
4 15 Not Useful

## PC/104 QUARTER

Rick Lehrbaum

## To ROM or Not to ROM

## That is the Question

*We all take for granted the three minutes of boot-up time for desktop computers. But, we'd never tolerate such performance in a task-specific system. Rick looks at ways to gain instant response from an embedded PC/104 computer.*

An embedded system should operate like an appliance. When you turn it on, you don't want to wait half a minute for it to boot its operating software from a hard disk drive. It has to perform its function instantly.

Such systems, if they contain microcomputers, normally load their software instantly from ROM (or EPROM), not disk drives.

There are other reasons why you may not want an embedded system to use a conventional disk drive. In applications with critical data-integrity requirements, "soft" read/write errors are unacceptable.

And, disk drives don't work over wide temperature ranges, so they're usually limited to indoor, temperature-controlled environments. Shock and vibration are also problems. Size, power consumption, and heat generation can also be reasons to avoid disk drives in embedded systems.

So, it may seem like a good idea to ROM your embedded-PC

application rather than operating it out of a disk drive.

You're probably set to hear all about ROMing a PC/104-based application. Do you expect "Rick's tips" on splitting software into independent code and data blocks that go in separate ROM and RAM devices?

Well, this month, my mission is to make the case why you really don't want to ROM

a PC/104 application. Fooled ya! Here's my pitch....

**KEEP AN EYE ON #1**

You may have used microcontrollers on other projects, and you probably ROMed your application. But, just because that's how it's done on a microcontroller doesn't mean it's the right way for an embedded PC.

Of course, I assume you want to harness the full potential of PC compatibility. While I'm sure you have a whole slew of reasons for using an embedded PC, I'll bet software tops your list.

Want to simplify and enhance your embedded project through the vast storehouse of PC OS, driver, and development software? If so, remember: to reap the benefits of PC compatibility, stay PC compatible!

The PC was designed as a disk operating system (DOS) machine. PC software is always

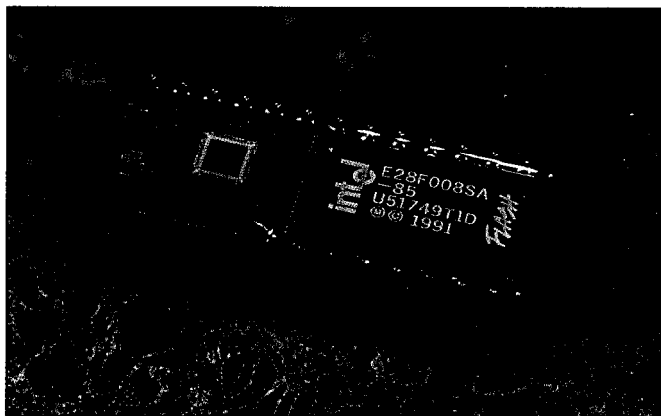


Photo 1: The tiny **DiskOnChip** from M-Systems squeezes up to 12 MB of flash memory into the same package as a 32-pin EPROM.

transferred from disk into system DRAM, where it runs. (Even the BIOS is generally "shadowed" in DRAM for faster execution.)

So, if you ROM your embedded application (and perhaps even DOS), you abandon the PC "standard." Don't be surprised when you lose access to tons of off-the-shelf device drivers, function libraries, and application programs that rely on a DOS environment.

If you insist on ROMing your PC/104 application like a microcontroller, be ready for the traditional microcontroller development headaches and limitations.

### EMULATION = FLATTERY?

Instead of ROMing the application, use a solid state disk (SSD), which "emulates" normal disk drives.

With SSD, a software driver transforms accesses to a normal disk drive into accesses to some form of memory. It's like a RAM disk, except that an SSD is typically used as a boot drive.

Since nearly every PC program makes disk accesses via DOS or BIOS functions, the system can't tell the difference between the SSD and a real disk drive. Therefore, SSD-based embedded-system development doesn't require special expertise, as long as you take advantage of one of the readily available forms of plug-and-play SSD.

You can develop your application on a PC with normal disk drives. You don't need to write ROMable code. You don't even have to know how DOS organizes or uses the PC's memory space!

Simply develop and test your application using your favorite OS, programming

language, and other software tools just like it's going to run on a conventional disk drive. Once you're satisfied, transfer the application to the SSD.

This procedure depends on what type of SSD you're using, but it's normally fast and easy. After transferring the application, remove the normal drive and reboot.

The system should boot and run from SSD. That's it! SSD converts "software" to "firmware" instantly and painlessly.

### MAKING AN SSD

There are quite a few SSD approaches. Many PC/104 embedded PCs (e.g., Ampro's CoreModule and little Board products) have onboard sockets built into the BIOS where you can plug in SSD devices and driver support to emulate a bootable A: or C: drive.

You can also use PCMCIA cards as SSDs. Some SSD drives look and act like ordinary IDE or SCSI disk drives, but they're based on nonvolatile memory, not magnetic media.

In general, you need a nonvolatile memory device and an appropriate SSD software driver. Before examining some of the SSD options available, let's review current technologies and interface architectures.

### TECHNOLOGY OPTIONS

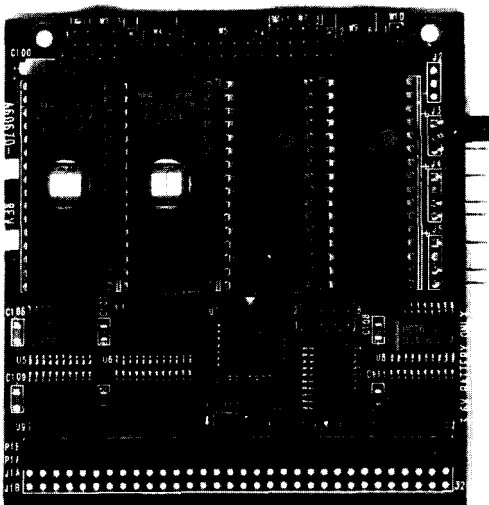
There are three main choices of memory-SSDs—EPROM, SRAM, and flash.

As an SSD technology, EPROMs are, of course, usable only as read-only SSD drives. They can't help you write data into an SSD during system operation.

Since EPROMs generally can't be erased and reprogrammed while they're plugged into the target embedded PC, you have to program them beforehand. Obviously, it's a nuisance (and, sometimes, expensive) to update an embedded system's software in the field.

On the other hand, the per-unit cost of EPROM SSD can't be beat. So, when in-system writability isn't required and cost is critical, an EPROM-based SSD may be just the ticket.

**photo 3: Ampro's SSD PC/104 module lets you mix-and match your choice of EPROM, SRAM, and flash-memory chips. Back-up circuitry converts SRAMs into NVRAM.**



**Photo 2: SanDisk's FlashDrive looks and acts like a miniature (1.8") IDE hard disk, so it's easy to use. Note the IDE interface—a high-density, 2-mm, 44-pin connector.**

Using nonvolatile RAM (NVRAM) as an SSD is probably the easiest approach. It's simple, requiring one or more RAM chips (usually 32-pin DIP), a nonvolatile controller, and a back-up battery.

Since they're fully read-write, NVRAM SSDs can be programmed directly within the target embedded PC and used like ordinary read-write disk drives (provided you have the right SSD driver software).

If you design NVRAM sockets directly into your application, you have to deal with making an SRAM nonvolatile. You can buy the necessary logic in a single tiny chip, add a battery, and look it up.

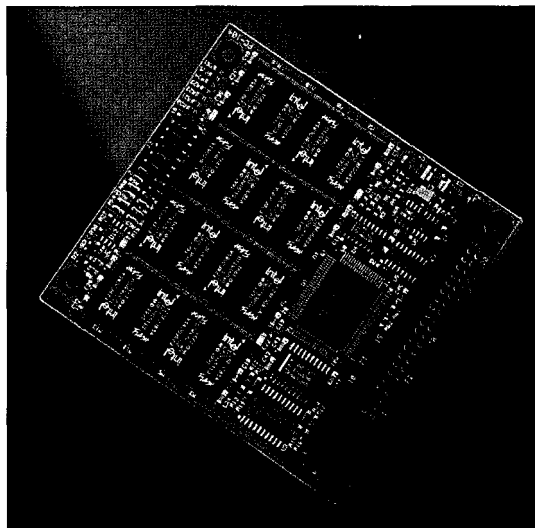
Don't underestimate the technology in that little chip! It's critical that the SRAM be protected from accidental write strobes during system power cycles and that its power be properly switched to and from the back-up battery at the right time.

It's easier to get NVRAM from special 32-pin SRAMs with built-in back-up batteries and control logic. They're available from Dallas Semiconductor, Benchmark, and others. One company even makes a device with a replaceable, snap-on battery.

While NVRAM has the advantage of read-write simplicity, it has a couple of disadvantages. One is cost. SRAM is the most expensive form of memory and can be many times as expensive as EPROM.

Another problem is temperature. Batteries have a limited operating temperature range, potentially excluding an embedded PC's NVRAM SSD from certain applications. And, there are environments where

**Photo 4:**  
M-Systems offers up to 32 MB of flash memory on this PC/104 SSD module. The product's support software works with DOS, Windows, QNX, and VxWorks.



batteries aren't allowed due to their corrosive (sometimes explosive) chemicals.

Of course, batteries don't last forever. Systems with NVRAM SSDs eventually need their batteries replaced, which can be inconvenient and costly. It also means expensive system down time and loss of valuable data.

So, if NVRAMs have all these problems and EPROMs are read-only, can any other memory technology work well as an SSD? That brings us to..

## FLASH MEMORY

Flash is closely related to EPROM. But, some clever semiconductor scientists figured out how to use quantum effects to make an EPROM that can be erased and reprogrammed electrically.

"Isn't that an EEPROM?" you might ask. Not exactly.

EEPROM was the first form of electrically erasable/programmable ROM. But it was more expensive than SRAM! Flash is only slightly more expensive than EPROM.

Unfortunately, flash isn't as easy to erase and reprogram as SRAM. But given its low cost, so what if it takes a little extra effort!

Flash-device erasing and programming require careful attention to details. Data and write control signals must sequence just right or the data neither records nor programs fully into the memory cells.

Also, flash wears out. It only lasts a specified number of erase/write cycles. Fortunately, it's rated for hundreds of thousands of cycles, and there are ways to manage its lifecycle.

One method is to reread the data after writing to a flash location to verify that it was written successfully. As the location wears out, you might need to write the data several times to get it to program successfully. Eventually, it fails completely, but it does extend the life of flash for a while.

If a flash device needs to be written many times, it's critical that the writes be

evenly distributed or it may wear out prematurely. It's like rotating your car tires. This process--called "wear leveling"--is a critical function of flash support software.

As if this wasn't enough, flash doesn't let you rewrite a single location. You must erase and rewrite some minimum block size. Blocksize has been steadily shrinking as flash technology evolves, so it's disappearing as a key issue. But, it used to be an entire chip, which caused some interesting SSD implementation challenges.

Now you track "clean" and "dirty" blocks. New data is always written into clean blocks. And, blocks with data that's no longer current are marked "dirty."

After a while, a flash device can be full of dirty blocks even if it's nearly empty from a DOS perspective. When this happens, the flash-management software consolidates the good data, making new clean blocks, in a process called "garbage collection."

Flash-management software carefully maintains tables of clean and dirty blocks.

Doing this, while maintaining the good data's integrity, is tricky. Don't try it at home!

Fortunately, several sources of off-the-shelf flash file system (FFS) software do a good job of making flash work reliably. A popular one is TFFS from M-Systems.

## NAND VERSUS NOR

The flash memory I've been talking about is NOR. There's a new development in flash called NAND. The two names refer to how the logic inside the devices is structured.

I won't try to explain the internal differences (OK, so I don't really know!). But, I want to point out a couple of key functional issues that affect how they're used.

NOR flash is accessed a lot like EPROM or SRAM, except for the restrictions I mentioned. You put an address on its address pins, and you read or write data on its data pins. It can even plug into an EPROM socket.

NAND flash is accessed in more of a serial datastream manner. In this sense, it acts a little more like a disk drive.

NAND flash was developed as a disk-like storage medium for digital cameras and hand-held computers. So, it's no surprise that it's quicker to program, has conveniently small erase-block sizes, and boasts a high erase/write endurance.

Now for the bad news. NAND is just as tricky as NOR, but for other reasons. For one thing, you don't talk to it like an SRAM or EPROM. You need special circuitry to interface it to the system.

Another problem: NAND devices come with defects, just like hard disks. You need to test for and map out the defects.

As with disk drives, it's useful to include error detection and correction using CRC logic. You need a special controller chip--

Device	System Interface	Size (in.)	Max. Cap.	Sustained Read Rate	Sustained Write Rate
EPROM chip	32-pin DIP	1.8 x 0.6 x 0.3	1 MB	fast	(read only)
NVRAM module	32-pin DIP	1.8 x 0.6 x 0.4	512 KB	fast	fast
NOR flash chip	32-pin DIP	1.8 x 0.6 x 0.3	512 KB	fast	(read only)
DiskOnChip, NOR	32-pin DIP	1.5 x 0.6 x 0.3	2MB	fast	slow
DiskOnChip, NAND	32-pin DIP	1.8 x 0.75 x 0.3	12 MB	fast	medium
1.8" IDE Flash drive	IDE	3.0 x 2.0 x 0.4	240 MB	medium	medium
CompactFlash	IDE	1.4 x 1.7 x 0.2	20 MB	medium	medium
PC/104 Flash Disk	PC/104	3.6 x 3.8 x 0.6	32 MB	fast	slow
MiniModule/SSD + EPROMs	PC/104	3.6 x 3.8 x 0.8	4MB	fast	(read only)
MiniModule/SSD + SRAMs	PC/104	3.6 x 3.8 x 0.6	2MB	fast	fast
PCMCIA-ATA	PCMCIA	3.4 x 2.1 x 0.1	300 MB	medium	medium
PCMCIA linear flash	PCMCIA	3.4 x 2.1 x 0.1	64 MB	medium	medium
PCMCIA linear NVRAM	PCMCIA	3.4 x 2.1 x 0.1	64 MB	fast	fast

Table 1: There are four main SSD approaches used in PC/104 systems--chip-like modules plugged into 32-pin DIP sockets, drive-like modules connected to an IDE interface, specialized PC/104 modules full of flash, EPROM, or NVRAM devices, and cards plugged into PCMCIA slots.

**Photo 5: PCMCIA fits neatly into PC/104 applications and supports all the key SSD approaches, including linear flash (shown), ATA-flash, and NVRAM. Ampro's PC/104 adapter (also shown) has slots for two PCMCIA cards.**

and software-to effectively use NAND flash chips as SSDs.

By now, I've probably scared you so badly you're ready to turn the clock back 20 years and return to ROM-based micro-controllers. But, don't despair!

There are quite a few **easy-to-use, plug-and-play SSD** solutions ready to serve your PC/104 embedded-PC needs.

### BYTE-WIDE DEVICES

Most PC/104 embedded PCs include one or more **32-pin byte-wide memory-device** sockets. These were originally intended for simple DIP or PLCC EPROM and SRAM chips.

Using these devices, the capacity of a 32-pin socket is limited to 1 MB for EPROMs and 5 12 KB for SRAMs. Depending on the SSD driver, multiple byte-wide sockets can combine into a single DOS drive letter for larger SSD capacity.

As **32-pin NOR flash** surfaced, it's been supported like an EPROM, except that it can be reprogrammed—usually on a full device basis—inside the system.

The simple byte-wide SSD is pretty limited in capacity (comparable to a floppy). Although a few PC/104 applications run out of one or two simple byte-wide chips, storage requirements have exploded with CPU performance and memory availability.

A nice solution to the limitations of the simple **32-pin** byte-wide SSD is provided by M-Systems' DiskOnChip flash module, shown in Photo 1. Although it's the same size as a **28-** or 32-pin DIP EPROM, this compact device has up to 2 MB of NOR, or up to 12 MB of NAND, flash. It contains the necessary circuitry to look just like a simple DIP EPROM to the PC/104 CPU.

DiskOnChip comes complete with **TFFS** and other support software for formatting, operation, and maintenance. Future capacities are expected to reach 72 MB.

The original NOR version of DiskOnChip (DOC 1000) had relatively slow write-cycle time and long garbage-collection **latencies**. However, the new higher capacity



version (DOC2000) benefits from the fast write cycles and small erase block sizes of NAND technology.

### DRIVE-LIKE MODULES

Several companies now offer small size (1.3" and 1.8" form factor) SanDisk flash drives that look precisely like small IDE hard disk drives (see Photo 2). They have the same physical footprint, mounting holes, interface connectors, and functional interface as their magnetic media counterparts.

With IDE flash, a microcontroller handles all flash-management functions (e.g., wear leveling), so no special drivers are needed. IDE flash was pioneered and popularized by SanDisk (formerly SunDisk) and is available in **4-300-MB** capacities, with higher capacities on the way.

To use these tiny IDE flash drives, just install and operate them like normal IDE drives. Nothing's simpler!

One possible catch: you need an IDE interface in your system. However, most PC/104 CPUs now include IDE interfaces free.

One major advantage: IDE flash drives are OS independent. All operating systems provide IDE hard disk support, and flash-management is handled by the drive. You can replace the drive without worrying about having the proper driver for the specific flash technology.

### PC/104 SSD MODULES

Why not put EPROM, NVRAM, or flash SSD devices on a dedi-

cated PC/104 module? This option, too, is readily available in a variety of formats.

PC/104 SSD modules come with four or more 32-pin byte-wide sockets for individual plug-in EPROM, SRAM, or flash (e.g., Ampro's MiniModule in Photo 3). They're also available with soldered on NOR flash for up to **32-MB** SSD capacity (e.g., the M-Systems PC/104 Flash Disk in Photo 4).

### PCMCIA MEMORY CARDS

Last, but certainly not least, is PCMCIA (see Photo 5). This well-known standard offers a broad range of SSD capabilities.

In fact, every memory technology I mentioned (i.e., NVRAM, EPROM, various kinds of flash) is available on PCMCIA cards, which are beginning to be called "PC Cards," by the way.

As a plus, PCMCIA cards are removable. They can be inserted and removed while the system is running, just like floppy disks. They're useful for storing data, loading parameters, and updating firmware.

Since PCMCIA cards are popular for laptop PCs, they are sold by all major computer retailers at competitive prices.

But PCMCIA cards require a special PCMCIA card slot. You can't just plug them into a byte-wide memory socket or cable them to an IDE interface. Instead, you need a PCMCIA controller or interface module, adding cost and complexity.

If you don't need them as removable media, check out a DiskOnChip or an IDE flash drive. On the other hand, you may need NVRAM for its speed and truly unlimited **rewritability**, and PCMCIA is the best



Photo 6: Although the jury's **still** out on which **memory-card** format will be the **long-term** favorite for digital cameras, these tiny SanDisk Compactflash cards certainly meet the needs of PC/104 applications requiring highly compact **and** removable **flash-memory** modules.



way to contain high capacity, reasonably priced NVRAM.

Incidentally, PCMCIA offers two different flash-card configurations.

SanDisk's PCMCIA-ATA is functionally identical to an IDE flash drive, except it's accessed through a PCMCIA card slot instead of an IDE interface.

From a command-set perspective, it provides the identical system-level interface as IDE. It's even possible to create a passive adapter between PCMCIA-ATA flash cards and standard IDE interfaces, eliminating the need for a PCMCIA controller.

Like IDE flash drives, each PCMCIA-ATA card has an internal controller to handle flash-management and IDE command-set functions. (ATA means AT attachment interface. It's just another name for IDE.)

The other PCMCIA flash approach corresponds to that used in the DiskOnChip (see Photo 5). It is commonly referred to as "linear flash" because the flash is indirectly accessible (via bank switching) to the system CPU as blocks of linear memory. The required bank-selection logic is located within the PCMCIA interface controller.

Although linear-flash PCMCIA cards don't bear the burden of an internal controller, this is also their biggest shortcoming.

With no internal controller to automatically manage flash-memory wear-leveling and erase/write functions, these tasks must be handled by the system CPU, resulting in reduced real-time performance and creating a degree of OS dependence.

Also, when you change cards, you must ensure that your embedded system has the right driver software to properly handle the card's internal flash technology.

## NEWS FLASH

We can't leave the subject of SSD for PC/104 applications without looking at the latest developments—solid-state storage for digital cameras.

Have you noticed ads for filmless digital cameras? Photography's going digital!

Whether this is good or bad for photographers, I can't say. But for PC/104 embedded systems, it means SSD media—especially flash—is about to become less expensive and more widely available.

Unfortunately, there isn't a consensus on exactly what tiny memory-card standard will prevail. Sound familiar?

So far, SanDisk's CompactFlash has made the most inroads. Shown in Photo 6,

it's essentially a shrunken version of PCMCIA-ATA.

Like its big brother, CompactFlash has an IDE-like functional interface. It's relatively easy to convert from an IDE hard drive interface to a CompactFlash card socket.

While this sounds like a dream come true for embedded systems needing miniature removable SSD cards, CompactFlash and two other tiny memory-card standards are fiercely battling for dominance in the digital-camera market.

For sure, you'll be hearing a lot about new flash-memory alternatives to disk drives.

## PUTTING IT TOGETHER

As you can see, PC/104 system designers have many options. To help you evaluate the alternatives, take a look at Table 1.

I hope you're feeling more enlightened—rather than more confused—than before. If not, don't worry. It'll probably all come to you—in a flash! PCQ/EPC

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

## SOURCES

**MiniModule/SSD**  
Ampro Computers, Inc.  
990 Almanor Ave.  
Sunnyvale, CA 94086  
(408) 522-2100  
Fax: (408) 720-1305  
[www.ampro.com](http://www.ampro.com)

DiskOnChip, TFFS  
M-Systems, Inc.  
4655 Old Ironsides Dr.  
Santa Clara, CA 95054  
(408) 6545820  
Fax: (408) 654-9107  
[info@ccm.msyscal.com](mailto:info@ccm.msyscal.com)  
[www.m-sys.com](http://www.m-sys.com)

**PCMCIA-ATA, FlashDisk, CompactFlash**  
SanDisk Corp.  
140 Caspion Ct.  
Sunnyvale, CA 94089  
(408) 542-0500  
Fax: (408) 542-0503

**MiniatureCard**  
Intel Corp.  
500 W. Chandler Blvd.  
Chandler, AZ 85226-3699  
(602) 5548080  
Fax: (602) 5547436  
[www.intel.com](http://www.intel.com)

## IRS

- 4 16 Very Useful
- 4 17 Moderately Useful
- 4 18 Not Useful

IF YOU DO

# FUNCTIONAL TESTS

YOU NEED

## HOT SWAPPING ELECTRONIC EXTENDERS

For PCI, ISA, VXI, EISA, VESA, Micro Channel & NuBus. Custom Designs available.

Electronic ISA Extender / PC Mini Extender



Electronic PCI Extender / PCI Mini Extender



- Insert/Remove Cards With PC Power On!
- Save Time Testing And Developing Cards
- Save Wear On Your PC From Rebooting
- Adjustable Overcurrent Sensing Circuitry
- NO Fuses, All Electronic For Reliability
- Single Switch Operation W/Auto RESET
- Optional Software Control Of All Features
- Breadboard Area For Custom Circuitry

And More...

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

Passive PCI Extender

Passive EISA Extender



Passive MC32 Extender

Passive ISA Extender



## AZ-COM, INC.

3343 VINCENT Rd., #D  
PLEASANT HILL, CA 94523 U.S.A.

TEL: 510-947-1000

FAX: 510-947-1900

24-Hour Fax on Demand:

510-947-1000 Ext. 7

E-Mail: [sales@az-com.com](mailto:sales@az-com.com)

VISIT OUR HOME PAGE AT:

<http://az-com.com/>

## Applied PCs

Fred Eady

## Right on Cue

## National Presents Slimline 'x86

Fred measures and stores voltages at specified times using National's *NS486SXF*. He shows how to get the ADC data into the '486 for processing, store the data to *EEPROM*, and use *Vetra's Reverse Pipe* for keyboard access.

It's almost time. I tense up as I await my cue. We're on the air.

"Hi, I'm Fred Eady. Welcome to the Circuit Cellar Florida Room. Today's special guests are National Semiconductor's *NS486SXF* and *Vetra Systems' Reverse Pipe*." (Huge applause from the audience.)

I turn to greet my guests, and they're machines! A PC board filled with all sorts of components and a rather tiny black box laden with I/O connectors sit by my desk!

"Oh, no!" I think. "I don't have any software to make them talk!" (We're on a talkshow, you know.)

I frantically call to the set's best boy, "Get my embedded development software and that *VIPer806* out here quick!"

I yell to the set director, Mark, "Roll a couple commercials back to back. That'll give me time to get these things hooked up! Did you bring that little x-ytable? We may need it for fill."

I turn around quickly, drop the mic, and trip over the cord. As I fall on my face in front of a live studio audience, I mumble, "Oh, @\*#\$!"

Blahnn. Blahnn. Blahnn. Wake up, sleepy head. I slap the alarm button and think about how I hate that noise. Boy, another weird dream. Better get up and get with it. I've got an article to write.

Most guys sleep soundly and dream of beautiful women, wealth, and fame. Not me. My *Vanna* is a piece of embedded silicon dressed in sexy software lace. My wealth? It's firmware that's either gone up in a flash or stored in a spinning magnetic vault. Fame? Well, it's the 15 minutes or so a month from those of you that follow my adventures in *INK's* Florida Room.

But since this offering started out as a talk show, let's meet the guests.

NATIONAL'S **NS486SXF**

Normally, when you think of National Semiconductor, you think components. You know-regulators, logic ICs, things like that.

Sure, there's the National COP8 series of microcontrollers, but National never really spelled "embedded" like you and I do.

For the next few minutes, we're going to put the new *NS486SXF* evaluation board to work.

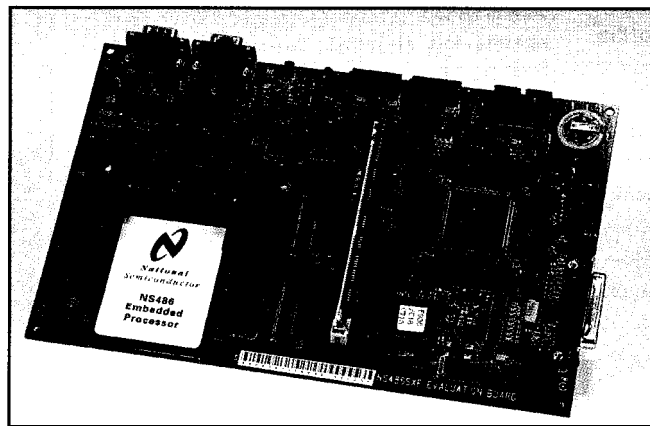


Photo 1: This beauty can turn any embedded programmer's head. Note the abundance of header pins surrounding the **NS486SXF**.

The heart of the NS486SXF eval board shown in Photo 1 is, of course, the NS486SXF silicon. The NS486SXF is an embedded controller based on the Intel '486 32-bit processor.

Unlike its big brother, the NS486SXF fits into most embedded environments. With its power-management ability and strong peripheral set, it was born to be embedded.

This little guy can run most RTOSs, including QNX. It uses standard +5-V power and incorporates all the peripherals we embedded types can't do without.

Although the processor speed is limited to 25 MHz, there are some special embedded features that may come in handy for your applications. One of those is the ability to reconfigure unused peripheral pins for task-dependent purposes. There's also an IEEE-compliant parallel port.

National describes the NS486SXF as a "system on a chip." Hardware functionality for most embedded applications can be found in its core. It's petite, fitting nicely into many embedded tight spots.

But, it gains this slimness by sacrificing some functionality. It differs from the standard Intel part in that real-mode, virtual-memory, and floating-point support aren't there. The lack of real-mode support implies that any conversion of older 8086-based embedded apps will need some touching up.

If your potential application needs to crunch numbers, be ready to take out your code checkbook and write some big ones. The only floating point you'll find will be in your software.

While the checkbook's out, write one for setting up onboard peripherals too. They're handy but code expensive. I spent a great deal of time tweaking bits in various registers to get the simplified code you see in Listing 1.

The bottom line: the NS486SXF is a *slimline* version of its Intel brother and is ideally suited for particular types of embedded applications. Figure 1 offers a simplified block diagram of the NS486SXF.

The evaluation board includes flash, DRAM, UARTs, PCMCIA, IR, and a real-time clock laid out and ready to use. There's even a diskette with "it really works" example code to exercise all the system service elements or to include in your own project.

All you need to get started is Bill's or Borland's C. To help you along, the folks at National include all the C header file definitions for the NS486SXF register set with the evaluation kit. Sample OSs from Microtec, WindRiver, and Phar Lap are also there.

Oh, yeah. The documentation—5.298"! If you don't have room on the shelf, conserve space by viewing it on National's Web site.

### THE MONOLOGUE

As a guy that uses the soldering iron as much as the keyboard, I find myself poking probes here and there to test voltages or

but there's an extra serial port. That's where the Reverse Pipe comes in.

As you see in Listing 1, a bit of C code, the flip of a DIP switch, and bap! It's a keyboard interface, compliments of the serial port.

Vetra's VIP-345 Reverse Pipe converts standard PC-keyboard keystrokes to ASCII codes. It can also be programmed to pass nontranslated voltage levels corresponding to PC-keyboard scan codes.

A keyboard lets me write menu code so I don't have to *hardcode* every test sequence. When you use the Pipe in your NS486SXF application, be sure to use a null modem between it and the NS486SXF board.

This app is all about measuring voltages. The NS486SXF eval board isn't ADC equipped, so that's got to be done externally. I don't need high resolution or speed, but it's gotta be cheap, so the National ADC0809 suffices.

Once the voltages are determined, I have to store them. That's easy. I'll use serial EEPROM.

The NS486SXF has a set of pins that can be configured for a Microwire/Access.bus master interface. And, Microchip offers a Microwire-compatible EEPROM in the 94LC66.

The plan is coming together. So far, I can measure and store my voltages as well as keyboard-alter my test sequences, thanks to the Pipe. I also need to provide TTL-level I/O lines to retrieve voltages and control the prototype's logic.

The ADC0809 is an 8-bit device that won't operate without an NS486SXF 8-bit I/O port, three NS486SXF-generated address lines, a start-conversion line, an address latch-enable line, and a nominal 400-kHz NS486SXF-generated clock signal. To effect the ADC0809 subsystem, I get these resources from the NS486SXF.

Reconfigurable I/O to the rescue! I assign NS486SXF I/O pins for the address and control lines as needed. As for the input port, I use the ECP in "PS/2" mode—a fancy way of saying "standard parallel port with bidirectional data capability." I can reconfigure the ECP port as bidirectional I/O pins, but why waste a perfectly good 25-pin connector already on the eval board?

The NS486SXF is loaded with three 8254-compatible timer-counters. It really

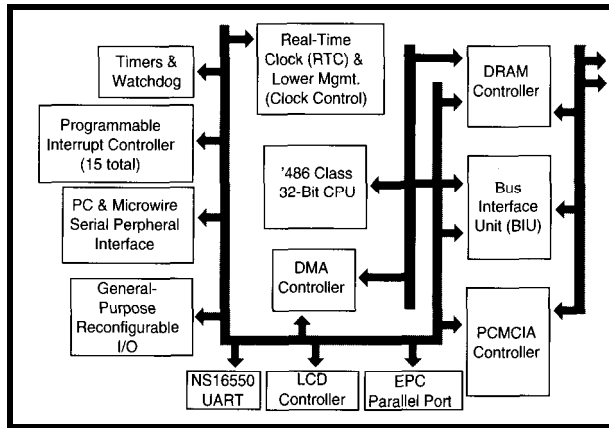


Figure 1: This baby's *all* that and a bag of chips!

look at waveforms on most of my little creations.

Well, this particular collection of solder globs requires logic levels to be applied to certain points in the circuitry and subsequently change voltage levels at other specified points. Since this group of parts and pieces is a prototype, the whole process is looking like manual labor to me.

To add to my misery, the voltage levels must be recorded and loaded into onboard nonvolatile memory for tracking and identification. That implies multiple boards, multiple voltages, and multiple headaches.

I don't think so. Smells like an NS486SXF application to me!

### THE LINEUP

The key to most successful covert operations is to know your enemy and bring the right weapons. Today's weapons are combinations of various black boxes.

In this application, the Vetra Reverse Pipe is one of those highly effective weapon components. The NS486SXF evaluation board has no native keyboard interface,

## 80251 Embedded Controllers



**MMT-188B**  
**MMT-Z180**  
**MMT-HC11**  
**MMT-196**  
**MMT-31**  
**MMT-251**  
**MMT-PIC44**  
**MMT-251 & OTHERS**  
 \$199.00 (100 Quantity)

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

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

#133


## AMX The Real-Time Multitasking Kernel

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

**NEW:** *KwikLook* AMX

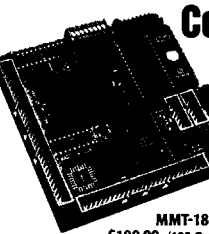
- Compact, ROMable, fast interrupt response
- Preemptive, priority based task scheduler
- Mailbox, semaphore, resource, event, list, buffer and memory managers
- Configuration Builder utility
- Comprehensive documentation
- No royalties, source code included

For a sample of *KwikLook* and description of AMX,  
 Phone: (604) 734-2796  
 Fax: (604) 734-8114  
 E-mail: [amxsales@kadak.com](mailto:amxsales@kadak.com)  
 Web: <http://www.kadak.com>


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

#134

## Low Cost Embedded Controllers



**MMT-188B**  
**MMT-Z180**  
**MMT-HC11**  
**MMT-196**  
**MMT-31**  
**MMT-251**  
**MMT-PIC44**  
**MMT-188B & OTHERS**  
 \$199.00 (100 Quantity)

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

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

#135

shows its embedded roots here. The timer-counter pins, including the gates, are physically accessible! Boom! There's our 400-kHz ADC0809 clocksource. The hardware particulars can be gleaned from Figure 2.

### BEHIND THE SCENES

With the problem defined and hardware resources in place, let's bring the project alive module by module. I'll start with the ECP peripheral baseaddressed at 0x0278.

All ECP functionality is self-contained in the NS486SXF. ECP port operation is controlled via the contents of the NS486SXF parallel-port I/O control registers, which are mapped in I/O space for easy access.

Of the six possible ECP modes of operation, I chose PS/2-compatible mode. I enabled it by setting bit 5 in the Extended Control Register (ECR) at location 0x067A.

The ECR's three highorder bits determine which mode the port operates in. For PS/2 mode, the mask is 001. The remaining ECR bits twiddle with the IRQ, FIFO, and DMA elements. Since I'm not using the other ECP modes, these bits are don't cares.

Once the port mode is set, the only thing left is set the port data direction. Since the ECP is used for input only, I set bit 5 of the Device Control Register (DCR) located at address 0x027A to enable the ECP data I/O pins as inputs.

If I needed bidirectional capability, I could toggle the state of bit 5 in the DCR using ASMOP (A Simple Matter Of Programming). That way, I could determine whether the ECP data pins were inputs or outputs.

The ECP is ready to roll. If the 0x0278 address looks familiar, it's because the NS486SXF architecture tries to retain standard PC I/O addressing where possible.

### INTERVIEWING A-T-O-D

Now I need three address lines and on ALE (Address Latch Enable) line for the ADC0809. These lines are part of an onboard multiplexer arrangement selecting one of eight analog inputs. The ADC0809 also requires a start conversion (SC) pulse I can piggyback onto the ALE line.

The analog-input port address is clocked into the ADC0809 on the rising edge of the

**Listing 1:** Like Barry Manilow says, "I write the wde *that* makes the whole thing sing."

```

int main0 {
  int i;
  i_inp(ECP_ECR);           // turn on PS/2 mode
  i | 0x20;
  _outp(ECP_ECR,i);
  i_inp(ECP_DCR);          // set ECP direction bit
  i | 0x20;
  _outp(ECP_DCR,i);
  i_inp(RIO_CONTROL);      // disable LCD/PCMCIA functions
  i | 0xc0;                 // and steal pins 48-54 and 68-79
  _outp(RIO_CONTROL,i);
  i | 0xFF;                 // set stolen pins for output
  _outp(RIO_DD_BYTE2,i);
  _outp(RIO_DD_BYTE3,i);   // write bit masks to RIO_DD_BYTE2
  i_inp(BIU_CONTROL1);     // enable PIT and Microwire
  i | 0x88;
  _outp(BIU_CONTROL1,i);
  _outp(PIT_CONTROL,0x5E); // initialize PIT
  _outp(PIT_COUNT1,0x04);
  _outp(PIT_CLOCK,0x02);
  _outp(PIT_TICR,0xEA);
  _outp(TWI_CONTROL,0xFC); // initialize Microwire
  _outp(TWI_U_CONTROL,0xF2);
  i_inp(UART_MSC);
  i & 0xFB;
  _outp(UART_MSC,i);
  _outp(UART_CLOCK,27);    // set clock to 25 MHz
  i_inp(UART_CLOCK);
  // Initialize UART and baud rate here...
  do {
    o // read char from Pipe and print to PC
    i _inp(0x3FD);
    while((i&0x01)!0x01);
    i_inp(0x3F8);
    printf("0x%x\n",i);
  } return 0;

```

SC pulse, and the ADC is initialized at that time. The conversion begins on the falling edge of the SC pulse.

Since I'm not using the NS486SXF LCD or PCMCIA peripherals, I can reconfigure their pins as the ADC0809 multiplexer address and latch lines. The Reconfigurable I/O (RIO) Control Register lives at 0xEFC0.

NS486SXF reconfigurable peripherals include four CS (chip select) lines, two UART lines, the ECP port (eight lines), the LCD interface (seven lines), and the PCMCIA interface (eight lines). By setting bits 6 and 7 of the RIO Control Register, I can disable the LCD and PCMCIA functionality, freeing 15 pins for general-purpose I/O.

As Figure 2 shows, I assigned NS486SXF pins 68, 69, and 70 as address lines, with pin 71 acting as the ALE and SC pulse generator. All these pins are outputs, and their function is defined in the Data Direction Register (DDR).

The DDR is 32 bits wide and resides at locations 0xEFC4-0xEFC7. The datasheet lays out what pins correspond with what DDR bits, so trust me. I chose the right ones, although I thought it odd that 1 made a pin an output and 0 signified an input.

Writing 1s to a standard parallel-port control register's (0x027A) lower nibble puts the port's pins in input mode. Most single-chip controllers use the "1 is input" scheme, too. To me, 1 denotes high-impedance inputs and 0 is round for outputs.

The Data Port Out Register at locations 0xEFCC-0xEFCE holds the output values of each reconfigured pin. When the I/O write signals are valid, this register's contents are output to the corresponding pins. Using NS486SXF RIO and the C outp mnemonic makes short work of the NS486SXF multiplexer support.

Next task is to generate a nominal 400-kHz clock for the ADC0809. I used the NS486SXF Programmable Interval Timer (PIT) to generate this pulse train.

Programming the PIT is accomplished by manipulating I/O ports at locations 0x0040-0x0045. Implementing the PIT is identical to the 8254-like devices found on run-of-the-mill desktops.

Getting a square wave at the correct frequency is no problem. First, I ensure the PIT is accessible by enabling it via the Bus Interface Unit (BIU) Control Registers 1 and 2. This task is done by writing a 1 to bit 3 of the BIU Control Register 1 at address 0xEF00.

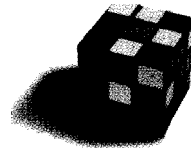
Next, since the counters come up with random junk, I have to program the counter I want to use. I use Counter 1 at I/O address 0x004 1.

The bit mask shown in Listing 1 selects Counter 1 (bits 7 and 6), loads an 8-bit count word (bits 5 and 4), sets squarewave mode (bits 3, 2, and 1), and sets up Counter 1 as a 16-bit binary counter (bit 0).

This byte is written to the Control Word Register at address 0x0043. After entering the control byte, a 16-bit count value is loaded at address 0x0040 and is the count value for Counter 1.

Finally, I make sure the Timer Clock Register at 0x0045 is set to divide the selected internal clock source by 16 and the Timer I/O Control Register (0x0044) lets the clock pulses escape via the Timer 1 out pin. Wham-o! I get a square wave at pin 56 on the NS486SXF that's real close to 400 kHz (~389 kHz).

The only A/D loose end left is the ADC0809 EOC output. No problem. I'll ignore it. Conversion takes place in ~100 μs, so I give it ample time in the final code to do its thing. I'm in no hurry.



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

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

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

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

**Paradigm**

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

**1-800-537-5043**

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



Figure 2: Not a single glue part. All the decoding and clocking are done by the **NS486SXF's** firmware and system service elements.

## MICROWIRE'S PERSONAL SIDE

Now that we can acquire voltage data, we must be able to put it away. The NS486-SXF Microwire interface can support the standard three-wire serial interface. Lots of goodies can play with Microwire, including Microchip's 93LC66 serial EEPROM.

Microwire is a three-wire synchronous serial bus. The serial input pin (SI) receives synchronous data transfers from Microwire-compatible peripherals. SI is NS486SXF pin 42.

Conversely, NS486SXF pin41 (i.e., the SO, serial output pin) drives data to Microwire clients. Since it's a bus-oriented protocol, several devices can be present on Microwire's bus. So, master and slave modes can be implemented, depending on how you use a particular Microwire device.

For now, the NS486SXF will be the Microwire bus master, with the Microchip 93LC66 acting as the slave. This arrangement means the NS486SXF supplies the Microwire clock (SCLK) at its pin 43. I'm not concerned with addressing details—I'm only driving a single slave from a single master.

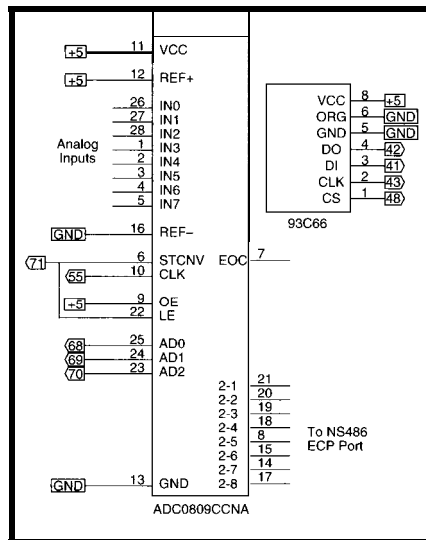
The NS486SXF SIO (serial I/O) register is an 8-bit shift register that transmits and receives data from the Microwire interface. Data is shifted out through the SO pin, most significant bit first.

Similarly, incoming data is shifted into the SIO register via the SI pin, implying that both transmit and receive functions are left shifts within the register. The NS486SXF SIO register is I/O mapped at 0x0051h.

The NS486SXF Microwire interface performs send and receive operations at the same time. Input data is sampled on the rising edge of SCLK, and data is driven to the output pin on the falling edge. In the case of the internal NS486SXF Microwire interface, it's always an 8-bit transfer.

Here's how I activate the NS486SXF Microwire interface. I set the master Microwire enable bit in the BIU (0xEF00 bit 7) and enable its interface via the Microwire/Access.bus Control Register (MACON). Setting MACON bit 2 at 0x0050 does this.

As the register's name implies, the NS486SXF Microwire interface can also



be used as an Access.bus interface. Bit 1 in the MACON lets me choose my mode of transportation. I clear bit 1 to select Microwire's interface. The MACON's remaining bits fiddle with the interface clock frequency.

Since I'm in no hurry, I set the clock well below the 93LC66 datasheet guidelines. I can always tune it later.

The next step tickles bits within the Microwire Control Register (uWCON) at address 0x0052. Bit 2 must be clear to allow the Microwire rising- and falling-edge data transfers.

Setting this bit produces the opposite effect. Microwire master mode is selected by setting uWCON bit 1. The interface shares its pins with some of the UART's modem control signals. Bit 2 in the Modem Signal Control Register at 0xEF71 must be cleared to activate the Microwire signals.

When I'm ready to plug data into the EEPROM, bit 3 of the uWCON sets off the process. This BUSY bit starts a transfer cycle and serves as the shift-register busy flag.

Reading and writing the 93LC66 in 8-bit mode uses 20 SCLK cycles. Also, it must be erase/write enabled via a 12-clock command sequence before accessing storage.

The NS486SXF Microwire interface can only do one 8-bit transfer per Microwire cycle. So, how can I command the EEPROM and get data, too, using an 8-bit cycle?

The 93LC66 data packet has a start bit, a 2-bit opcode, a 9-bit address/command field, and 8 bits of data. Add up the bits. That's where the 20 SCLKs come from.

The erase/write enable (EWEN) and erase/write disable (EWDS) commands are formatted the same way with no data at the end of the packet. The 93LC66 start

bit is detected when CS (Chip Select) and DI (Data In) are both high with respect to a rising SCLK edge.

So, I pad the high-order nibble of the first Microwire transfer cycle with 0s. The start bit is detected in bit time 5 of the first 8-bit transfer, and the rest of the packet's 16 SCLKs transfer the instruction and data just as the 93LC66 likes to see it.

## THANKS FOR TUNING IN

The NS486SXF offers modularity that can't be found in its Intel '486 big brother. But, it takes a lot of code to replace the number crunching of a math coprocessor.

But, if your application can live without some of the comforts of home, the NS486-SXF project you end up with won't be complicated...it'll be embedded. APC.EPC

*Fred Eady has over 19 years' experience as a systems engineer. He has worked with computers and communication systems large and small, simple and complex. His forte is embedded-systems design and communications. Fred may be reached at edtp@ddi.digital.net.*

## REFERENCES

- National Semiconductor, *NS486SXF Embedded Microprocessor Programmer's Guide*, 1996.
- National Semiconductor, *NS486SXF Embedded Microprocessor Evaluation Board Manual*, 1996.
- Phar Lap Software, *ETSLite User's Guide for the NS486SXF Evaluation Board*, 1996.
- National Semiconductor, *National Semiconductor Linear Applications Handbook*, AN-247, 1991.
- Microchip Technology, *Serial EEPROM Handbook*, 1994.
- Microchip Technology, *Non-Volatile Memory Products Data Book*, 1995-1996.

## SOURCES

**NS486SXF**, Microwire  
National Semiconductor Corp.  
2900 Semiconductor Dr.  
Santa Clara, CA 95052.8090  
(408) 721-5000  
Fax: (408) 739.9803  
ns486@arodbor.nsc.com

## 94LC66

Microchip Technology, Inc  
2355 W. Chandler Blvd.  
Chandler, AZ 85224.6199  
(602) 786-7200  
Fax: (602) 786.7277  
www.microchip.com

Reverse Pipe  
Vetra Systems Corp.  
275-J Marcus Blvd.  
Hauppauge, NY 11787  
(516) 434-3185  
Fax: (516) 434-3516  
eemoline.com/vetrasyst

## IRS

- 419 Very Useful
- 420 Moderately Useful
- 421 Not Useful

## DEPARTMENTS

66 MicroSeries

74 From the Bench

78 Silicon Update

## MICRO SERIES

Hugh Anglin

# Machine Vision

## Industrial Inspection

Part  
1  
of  
2

Ever hear of  
Inspector  
Gadget?  
He'd love to

learn how Hugh  
blends a computer,  
camera, sensors, and  
RTOS software to  
make a small-scale  
vision inspection  
system.



My friend was part  
of an engineering  
team installing a newly  
developed inspection sys-

tem in a manufacturing plant.

A status lamp indicating normal operation refused to turn on. Despite having an in-circuit emulator, software debugging tools, and an oscilloscope, they couldn't find the problem.

After several frustrating hours, a plant electrician stopped in to drawl, "Mebe de bubs gawn." An embarrassed engineer held up the evidence—a failed light-bulb filament.

Funny as it sounds, this story is typical. Given so many complex interactions in the system, we naturally suspect failures in the timing, special hardware, or software. Too often, we overlook the obvious.

There's an important distinction between using machine vision as a technology and developing an inspection machine.

An inspection system is a complete, integrated quality-control tool that uses various sensing methods (e.g., machine vision) to solve manufacturing problems. To be useful, machine vision requires integration into an overall system.

If there are ways to solve a problem without using vision, I tend to evaluate them first. Sometimes, a different

sensing technology provides a simple and elegant solution.

But, certain inspection problems are best solved with a camera-based system. And, vision is increasingly making its way into manufacturing.

Photo 1 shows the Insight 100, a turn-key commercialized inspection system. It inspects closures (i.e., bottle caps) for the pharmaceutical and beverage industries.

In Photo 2, the system is rejecting a defective child-resistant cap with a liner that wasn't correctly punched.

Taking basic video-capture technology and making it an integrated inspection system is a costly and involved development process. Success requires talent in a number of areas—mechanical, optical, electronic hardware, software, mathematics, and algorithm development (and don't forget the light bulb).

And, it doesn't impress the end user if the system has the latest VLSI hardware or DSPs when the user interface requires the skills of a rocket scientist. But, a user-friendly system also won't succeed if it doesn't solve the problems.

## SOLVE A REAL PROBLEM

Only by fully understanding the problem and applying complex technology simply and intuitively can you create inspection systems that are useful to the factory floor operator.

When inspection problems aren't well defined, there's a tendency to build extremely flexible systems that solve almost any problem. Conversely, if a system is too easy to use and narrowly focused, it limits market potential.

A balance between flexibility and ease of use must be achieved. Even with careful problem definition, I now expect some changes while designing an inspection system.

New or hidden requirements often materialize. Improvements in the manufacturing process may demand higher speeds. Or, color or material variations needed by the customer may cause changes to optics and algorithms.

Also, new product designs may appear. More inspection data may be

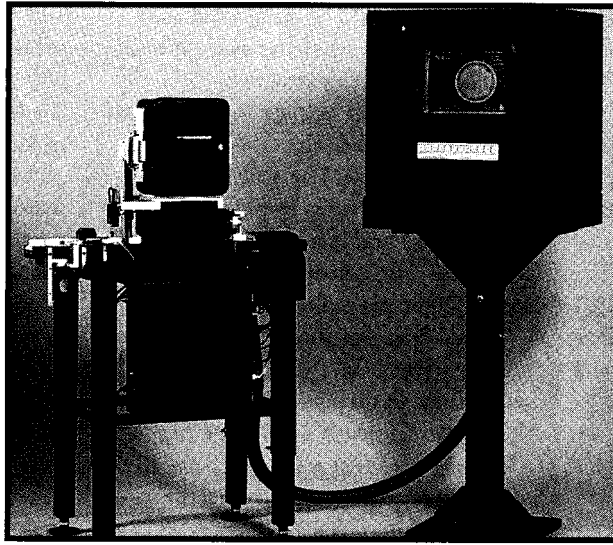


Photo 1—The Insight Control Systems model 100 is a turn-key inspection system used for high-speed closure (i.e., cap) inspection.

needed to help solve process problems. And, new defects may become an issue.

## USING INSPECTION SYSTEMS

Inspection systems can be used in complementary ways.

When sorting, they try to eliminate defective products from the manufacturing process. This task is especially important in high-speed applications where human visual inspection isn't well suited. Even in slower processes, people get tired, bored, or distracted and can be very subjective.

Automatic inspection systems also provide vital data for understanding and improving the manufacturing process. Even in a simple configuration, an alarm from the inspection system can halt the process and notify the operator that a defect limit is exceeded.

In a more advanced configuration, the inspection system interfaces electronically with the manufacturing machinery to automatically control the process directly. So, an inspection system that verifies label placement on a box can tell the manufacturing equipment to adjust the position.

You can also interface all inspection systems to a data-collection server over a network, making remote data collection, analysis, and reporting functions available plant-wide. A phone-line interface to the server's modem offers remote diagnostics, software upgrades, and problem resolution between the inspection-system vendor and the plant.

## A SIMPLE MODEL

In this series, I discuss the requirements, design trade-offs, and problems typically encountered in developing a machine-vision inspection system.

Figure 1 depicts an inexpensive vision system that uses a PC with a PCI frame grabber, camera, and strobe. A separate SBC tracks inspected parts.

An elementary system using these components offers real-time inspection at moderate speeds. It's not a full-blown inspection system since I won't address many important details [e.g., mechanical handling, packaging, and internationalization of software).

I focus on system architecture and integration of software and electronic hardware. I chose MS-DOS and Borland C to program this simple model, but I normally use QNX's 32-bit RTOS. I strongly advise using a real-time, multi-tasking OS for serious development.

This system sets up the camera, tracking, registration, inspection zones, and a few rudimentary image-inspection steps to detect defects on a washer.

This system sets up the camera, tracking, registration, inspection zones, and a few rudimentary image-inspection steps to detect defects on a washer.

## INSPECTION STEPS

Oversimplifying somewhat, inspection consists of performing these steps in sequence:

- sense the presence of the part to be inspected (i.e., "part in place")
- track the part until it's in front of the camera, and then issue a trigger signal to the frame grabber, which coordinates a strobe flash with image acquisition. Using a strobe and shielding the camera from ambient light eliminates motion blur.
- analyze the image to locate the inspected part and determine regions of interest (ROI) for inspection
- execute inspection algorithms in each region
- track a defective part to the reject point and remove it from the conveyor via a mechanical flipper or blast of air
- update inspection counters, display an image of the defective part with



the defects noted, and update process-control interfaces

Most steps must operate asynchronously with respect to the others. For full performance, acquisition, processing, and display must be able to overlap in time.

As for software, this description is just the tip of the iceberg. About 70% of the software goes into developing an interface that enables the user to easily configure and maintain the system.

## SOFTWARE REQUIREMENTS

A number of software modules should be included in an inspection system. Let's discuss each of them.

Camera setup offers a user interface to camera and frame-grabber controls (e.g., gain, reference values, digital filtering, etc.). You should be able to acquire images as they pass in front of the camera or continually acquire images of a static part by flashing a strobe at a constant rate (autostrobe mode).

A tracking-setup module lets the user set the correct timing for image acquisition and rejection. This menu must be fully interactive so the user can tell when the image acquisition and reject timing is right.

By differentiating its features from background clutter, registration setup trains the system how to locate the part and find inspection regions relative to the registration point(s).

Inspection setup lets the user establish sensitivity levels for each inspection algorithm. It must be interactive, showing pass/fail status on a test image.

Job management provides storage for sets of set-up parameters related to a certain product type. Image file management loads and stores filed images.

Run-time inspection setup lets the user fine-tune inspection sensitivities while the machine inspects parts online. The run-time screen displays a set of counters and computations showing inspection speed, number of inspections and rejects, failure rate, and a breakdown of failures by each inspection test.

This screen should support several views of inspected parts. Viewing each part as it goes by is of limited use. The images update too fast. However, dis-

play mode should be supported as it can indicate whether the system fails to reject defective parts.

A more useful display mode-freeze on reject-updates the display when parts are rejected. The ability to only view rejects provides important process information and is a valuable tool for detecting false rejects (something that should have passed inspection).

Photo 3 shows the run-time screen for the Insight 100 running in 256 x 240 resolution. In display all rejects mode, the system displays a beverage cap with a break in the seal area (i.e., a nonfill or void) caused by uneven distribution of the liner material.

A more advanced method of freeze on reject lets the user freeze on one specific inspection step. For instance, the option can be selected to only view rejects when the nonfill tool fails.

Finally, the software may require internationalization. If you're designing a commercial inspection system, it must support the local language. I make systems bilingual, so two languages are loaded into the software at startup—one for the plant and the other for field service technicians.

## STARVING FOR BANDWIDTH

One of the biggest problems in machine vision is having enough bandwidth for acquiring, processing, and displaying images in real time. Ideally, the system handles all three tasks simultaneously while tracking parts, managing user interaction, and updating process-control interfaces.

Although line speeds vary across industries, it's common for systems I design to operate in the ranges of 400-2500 parts per minute (ppm). When inspecting at 600 ppm, there are -100 ms between parts.

This speed is approximate since parts may not be evenly spaced, causing inspection speeds to burst occasionally. The system needs enough extra performance capacity to handle bursts without missing inspections or going south.

Frequently used image resolutions for high-speed industrial inspection are 256 x 240 pixels

and 512 x 480 pixels, with 8 bits per pixel to allow 256 shades of gray. At 600 ppm using 512 x 480 resolution, 2.5 MBps of bandwidth is used per full image operation.

In the past, standard computer buses weren't able to handle this load. Most machine-vision manufacturers designed special hardware with dedicated image buses, but they were typically large, proprietary, and expensive.

In recent years, more vision systems have been available commercially, and costs decreased as systems migrated to PCs. Newer buses (e.g., PCI) are fast, but they still fall short for heavy-duty vision applications unless the load is divided carefully among several processing components.

PCI transfer rates vary between PC manufacturers, so be careful if you depend solely on PCI bandwidth for system performance. Total bandwidth is not a sufficient metric. Instead, the evaluation must consider simultaneous availability of bandwidth for acquisition, processing, and display.

## VISION-SYSTEM ARCHITECTURES

Several types of systems are available in the PC-based vision market.

A PC and frame-grabber system is mainly suitable where inspection rates are low or the processing requirement is light. Some frame grabbers have onboard caching and enough intelligence to capture an image from a hardware trigger asynchronously without host CPU interaction.

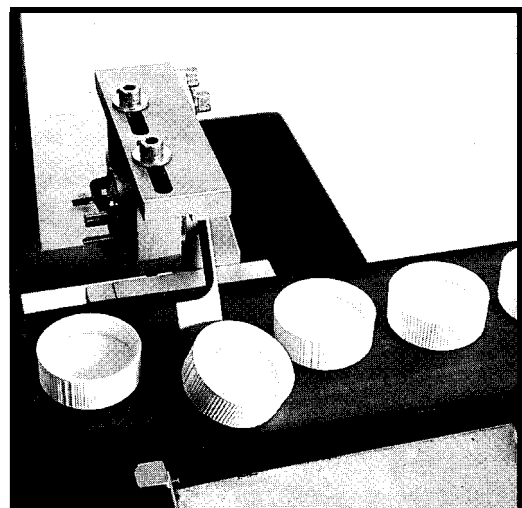


Photo 2—Defective closures must be eliminated from production. The cap being rejected has a flaw in the cut of the liner material.

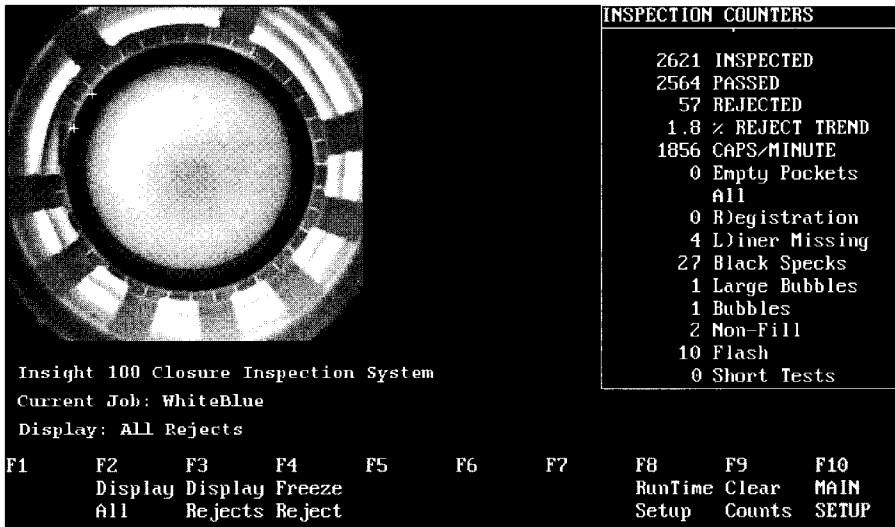


Photo 3—The run-time screen should show process statistics, as well as enabling the operator to select the display mode, clear production counters, and modify the inspection sensitivity while the system continues to inspect.

If more than one camera is used, caching and asynchronous operation are critical. Except for very slow-speed applications, the only suitable PC bus is a high-speed one like PCI.

In PCs with coprocessors (e.g., DSP, RISC, CISC) and frame grabbers, some of the newer processors (e.g., the TI 'C80) are extremely powerful pixel bangers. But, you may have to write highly optimized assembly code.

This multiprocessing architecture clears up some bottlenecks, but it can keep you reaching for the aspirin. Coordinating processes across multiple processors and designing in robust error recovery is complicated. Processor dissimilarities between the host and coprocessor (e.g., byte ordering and data alignment) need to be considered.

A PC with special hardware and a frame grabber normally offers a limited set of very fast image-processing functions. Additional analysis or processing may be necessary on the host PC. A high-speed bus to the special hardware and frame buffers is a big advantage.

Of course, I'm still looking for the ideal vision engine that balances hardware- and software-based processing.

ASICs can be used for image-processing functions requiring fast, repetitive neighborhood processing (e.g., histograms, convolutions, morphology, etc.). A high-speed processor performs intelligent postprocessing to arrive at a pass/fail decision.

The PC, running under an RTOS, serves as the system controller, handling the user interface and the comm link between the tracking and processing sections. A high-speed bus connects the PC and vision processors.

## LIGHTING AND OPTICS

Finding the right lighting technique is the first step in evaluating the use of machine vision in any application. Most real-time image-analysis software requires the inspected part to be illuminated so that any defects cause a contrast, color, or other change.

In more difficult inspections, more than one lighting technique is needed for 100% defect detection. Defects visible with one technique disappear with another. Look for a more generalized lighting technique or use multiple cameras and optical assemblies.

The topic of lighting and optics is huge. Cognex's minicourse proffers a fundamental exposure to the subject and useful course notes. In fact, they offer an excellent one-week course in

machine-vision fundamentals for engineers with a strong grasp of C.

Three types of light sources are commonly used—LEDs, strobes, and various constant light sources. I cover their pros and cons in Part 3. My example system uses a commercial Xenon strobe with a fiber-optic light ring and diffuser.

## CAMERAS

Camera technology is advancing rapidly, resulting in a wide range of new capabilities in a smarter, smaller, and faster package. They're also confusing, due to a lack of standardization both in function and terminology.

A few years ago, I found myself refereeing between my camera and frame-grabber suppliers. After the discussion circled a few times, it became apparent that we were suffering the Tower of Babel syndrome.

They were arguing over a subtle timing issue related to even and odd fields of video. But, one supplier numbered the fields 0 and 1, while the other used 1 and 2. Once terminology was resolved, the technical issues were, too.

For very high-speed applications, use a camera with a high-speed random-reset capability. Be careful you understand the timing and side effects when using special camera modes.

Some cameras accumulate ambient light while waiting for a reset pulse, which can cause blooming in the image. Other cameras advertising random-reset capability require a considerable time delay to reset.

## DISPLAY

An inspection system's display often poses difficult technical issues, since it must display both images and the user interface.

You could use two monitors—one for images and the other for the user interface. However, the system packaging tends to become too bulky, and real estate is often at a premium.

Using two small monitors sounds good until you check prices. If you need multiple cameras later, do you add a monitor for each?

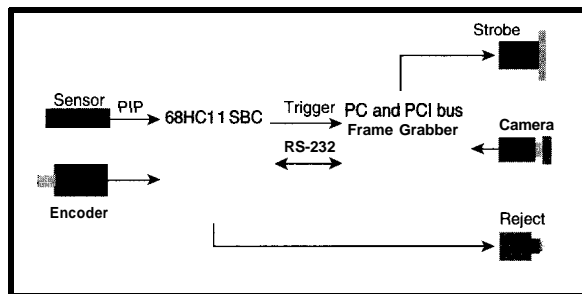


Figure 1—Here, you see the major components and interconnections in a simple inspection system.

# EMBEDDED BIOS

## Listen to What our Customers Say

"We're impressed by the documentation and the readability of the code." - M. Ryan

"We are very pleased with the General Software BIOS and look forward to working with you to bring our product to market." - R. Levaro

Embedded BIOS is well-structured and documented, and technical support at General Software is excellent. - J. Toivanen

"I am sure we made the right decision to buy our BIOS from General Software." - P. Filton

"Embedded BIOS is really the product for embedded PC designs. You were absolutely right!" - J. Josse

## The BIOS for Consumer Electronics

### Why You Should Choose Embedded BIOS, Too

- BIOS, DOS, Flash Disk With One low Royalty
- Instant Boot, Console Redirection, & Much More
- Expert Support with Guaranteed Response Time
- We Work Closely With Acer, AMD, Intel, & RadiSys to Deliver you a Proven, Tested, Feature-Packed BIOS
- Millions of Units Already licensed

### BIOS Adaptation Kit Includes:

- Complete Source Code
- Binary Configuration Program
- Quick Start + Over 600 Pages of Printed Documentation

**FOR DETAILS, CALL 1-800-850-5755**



General Software, Inc.

320 • 108th Ave. N.E., Suite 400 • Bellevue, WA 98004  
Tel: 206.454.5755 • Fax: 206.454.5744 • Sales: 800.850.5755  
http://www.gensw.com/general • E-Mail: general@gensw.com

© 1997 General Software, Inc. General Software, the GS Logo, Embedded BIOS and Embedded DOS are trademarks of General Software. All rights reserved.

#139

## Standard RS-485 Network

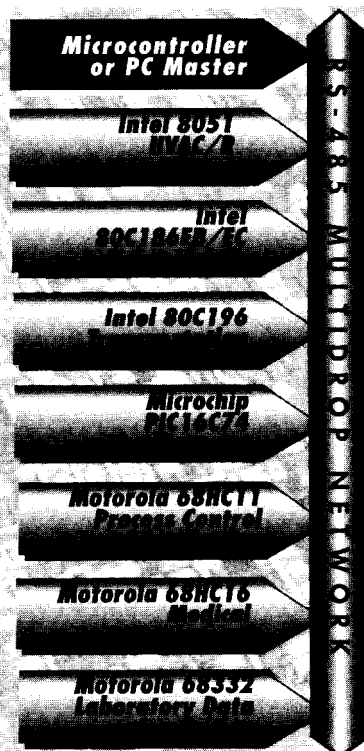
With Cimetrics' 9-Bit  $\mu$ LAN you can link together up to 250 of the most popular 8- and 16-bit microcontrollers (8051, 80C196, 80C186EB/EC, 68HC11, 68HC16, 68332, PIC16C74).

### The Q-Bit $\mu$ LAN is:

- ▶ **Fast** - A high speed (62.5k baud) multidrop master/ slave RS-485 network
- ▶ **Flexible** - Compatible with your microcontrollers
- ▶ **Reliable** - Robust 16-bit CRC and sequence number error checking
- ▶ **Efficient** - Low microcontroller resource requirements (uses your chip's built-in serial port)
- ▶ **Friendly** - Simple-to-use C and assembly language software libraries, with demonstration programs
- ▶ **Complete** - Includes network software, network monitor, and RS-485 hardware
- ▶ **Standard** - The 9-Bit  $\mu$ LAN is an asynchronous adaptation of IEEE 1118



55 Temple Place • Boston, MA 02111-1300 • Ph 617.350.7550 • Fx 617.350.7552



A dedicated image monitor may also have a downside if the vision hardware doesn't provide a non-destructive graphics overlay. You then end up drawing directly to the image buffer, which is a disadvantage if you need to reuse that image. Of course, software may repair screen damage.

Single-monitor displays also pose some design challenges, since the image and user interface share a screen.

Some vision systems have direct overlays over the image. Others use a dedicated window for image display and arrange the user interface around it. In windowed systems, though, the display must run in very high-resolution graphics mode, requiring a large monitor.

Whatever technology you choose, it should display images with graphics notations to mark defects in real time with little performance penalty. Otherwise, you'll have to make significant compromises in system performance to work around the deficiency.

And, avoid display methods that subsample the inspected image. Small defects can be hidden, leading the operator to incorrectly assume that the system has a problem with false rejects.

Now that you have the background, you're ready for Part 2. I'll cover a complete tracking system using a Motorola 68HC11 SBC.

*Hugh Anglin is a systems engineer with experience in real-time and embedded systems, process control, and machine vision. You may reach him by E-mail at [hanglin@insightcontrol.com](mailto:hanglin@insightcontrol.com) or by phone at (918) 3422248.*

## SOURCE

**Lighting and Optics Workbook**  
Cognex Corp.

One Vision Dr.

Natick, MA 01760-2059

(508) 650-3105

Fax: (508) 650-3332

## IRS

422 Very Useful

423 Moderately Useful

424 Not Useful

#140

# It Can't Be A Robot

## Part 1: There are No Arms and Legs!

Robotics has been riddled with too many preconceptions. Over the next few columns, Jeff intends to look at some of the areas that make robots tick. This month, he concentrates on getting the motors going smoothly.

## FROM THE BENCH

Jeff Bachiochi

“W

here do you think you're going?”

“Be-dop be-doop.”

“Well, I'm not going that

way. It's too rocky. What makes you think there are any settlements that way anyhow!”

“Thwerp, biddy biddy ba-werp.”

“Don't get technical with me. I've had just about enough of you. Go that way. You'll be sandlogged within a day, you nearsighted scrap pile.”

It's not the kind of conversation that comes to mind when we think of computers communicating. It's more like an act from the Comedy Club circuit rather than from a protocol (C3PO) and astromech droid (R2D2) lost in the Jundland desert on Tatooine.

I cheer George Lucas and those who pioneer the existence of robotics from Gort through Data. Our technology may not be on the same plane as our

dreams and fantasies, but those dreams and fantasies drive technology forward.

One reason robotics is so popular is because it touches on so many fields—motion, sensing, power, and intelligence. Improvement in one area can dramatically alter other fields.

### JUST A TOY

Don't tell my wife, Beverly, but I like flipping through catalogs. Not the Walter Drake or Harriet Carter stuff she reads, but good stuff like Mondo-tronics and Edmund Scientific.

I keep my eyes peeled for unusual items. Tons of robot kits saturate these catalogs. Thing is, most kits only perform a specific function: follow a line, move toward light, hug the wall, avoid falling off a table top. As teaching tools, these kits have carved out quite a niche.

Toys, however, are for fun. Although there's some truth to the saying “the bigger the boy, the bigger the toy,” I believe the toy's cost is not what makes it delightful.

Tamiya, a Japanese company, has an impressive line of motorized toy vehicles. I was a bit apprehensive about spending \$40 when I had no idea of its quality. I was even more frustrated to learn the kit was discontinued.

Scurrying back through the catalogs, I found an alternative from Mondo-tronics. Photo 1 shows the parts of the Power Shovel/Dozer kit.

What's so impressive about it? The three electric motors have preassem-



Photo 1—Tamiya supplies a collage of parts made from wood, metal, plastic, rubber-whatever fits the job.

bled gear boxes giving a good mix of torque and speed. The independently controlled rubber tracks make moving over small obstacles a breeze.

By rotating the tracks in opposite directions, you get a tight turning radius. Using only two 1.5-V D cells, you gain motor control through joystick-type switches at the end of a 3' umbilical cord. The major holes are predrilled. It would be easy to assemble—even for an 8-year-old.

## EDUCATIONAL PLATFORM

Some might suggest I'm copping out. Surely, I should design it from scratch.

Essentially, I agree, but I want to spend the time I have controlling the beast, not fabricating one. So, I'm going to use a known quantity and add motion, sensing, and maybe even a wee bit of intelligence.

Constructing the Dozer took less than 2 h. After taking it on a few spins, I dug out the multimeter and measured the current draw of each motor.

I measured -0.5-A continuous running current with peaks of -0.75 A. Using the seat-of-the-pants 2x rule, I searched for motor drivers that could handle 1 A continuous. I wanted the parts to be accessible.

National's LM18293 jumped out of the databook, and it's available from Digi-Key. It was on National's Web page, so I knew it wasn't on death row.

This single device has quad push-pull drivers. It can be used to form two H-bridges, one for either motorized tread. With an H-bridge, the motor can run in both directions without needing a bipolar supply. The only thing missing was internal protection diodes.

Figure 1 illustrates how I used the '18293. Each H-bridge is formed with a pair of push-pull amplifiers, each having two inputs and sharing an enable.

To be configured as an H-bridge, the inputs must be driven by opposing logic. If the inputs are both driven high or low, there's no potential across the motor. An '04 inverter kept the inputs opposed.

The motor drive IC was shown to have webbed legs on pins 4/5 and 12/13, but the parts I received didn't have them. These ground pins are beefed up to also give heatsinking for the chip.

The drop across each driver is about 1.5 V, so I needed -5 V just to get 1.5 V across the motor. Not terribly efficient, but at \$5, it's at least cost effective.

This device uses transistor junctions. The more expensive parts (e.g., an

LMD18200) use MOSFETs so the drops are considerably less. But, they are a single H-bridge device and cost ~\$20. If you substitute a pair of these, they'll cost as much as the motorized platform.

## MOTION

To control the motor driver, I used a micro. To keep costs low and the programming environment friendly, I used PicStic 1. The reprogrammable flash memory let me change BASIC (or assembler) programs easily—my form of experimental nirvana.

I've used PicStics a lot lately. It's a friendly device for those who always wanted to play around with micros but didn't dare to, given entry-level costs.

Whenever possible, I define the upper two bits of the I/O port as serial output (bit 7) and serial input (bit 6), even if the project doesn't require serial communication of any kind.

It's a useful debugging feature, and it enables me to use the same 5-wire networking connection on all my projects. The same connections can then reprogram the micro even while buried deep within the bowels of a project.

The open-collector mode of the serial communications protocol permits simple networking. The se r i n

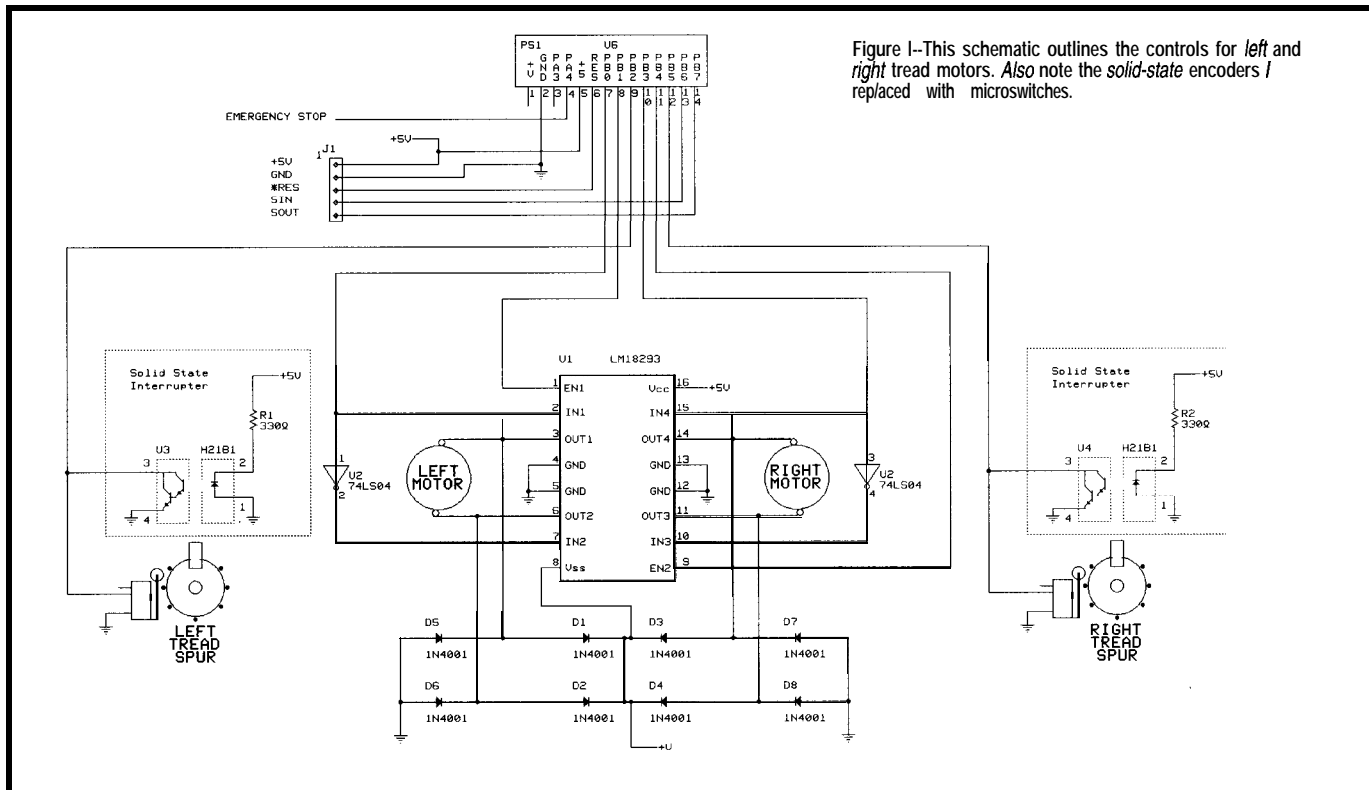


Figure 1—This schematic outlines the controls for left and right tread motors. Also note the solid-state encoders I replaced with microswitches.

statement ignores all communication until a particular character sequence is recognized, so you can keep other micros hanging on the same bus from interfering with private conversations.

I use the capital letter M as a single addressing character followed by one of four 2-byte commands-forward { Fx), backward (Bx), left turn(Lx), and right turn (Rx), where x=1-255 counts (0 being continuous). A count is a specific unit of distance measurement.

The only difference between moving forward/backward and turning is that the treads move in opposite directions instead of moving in the same direction. The base rotates about its center in a tight radius, making the platform highly maneuverable.

## SYNCHRONICITY

The twin DC motors that independently move the left and right treads run at different speeds depending on friction and load presented to each motor. Therefore, starting and stopping them together doesn't assure that both treads move the same distance.

Although the treads can independently slip in relation to one another, it's helpful to keep the dual drives in sync. To do this, you track the distance traveled by each drive train, perhaps by using shaft encoders. However, this vehicle has components that lend themselves to tracking distance.

The front wheels have three equally spaced holes that can be used to count wheel rotation (distance) by one-third or about a linear inch between hole rotations. The rear wheels have teeth that engage the plastic track. The gear teeth are spaced about every 0.25" and provide better resolution.

Remember when floppy disk drives were open framed and you could see their inner workings? The head movement was usually initialized to track zero by moving the head back and forth and sensing when a plastic vane on the head carriage slipped into an optical interrupter. This interrupter was made from an IR transmitter/receiver pair aimed at one another across a short gap.

The same IR pair can be positioned over the rear wheel's teeth so the rotating teeth break the IR beam. By tracking the number of times the beam is

Listing 1 -- This *PicStic1* BASIC program queries for a motor command (F,B,R,L) and a count. The left and right tread motors will then operate appropriately for the desired command.

```

symbol fwd = 1      : symbol bwr = 0      : symbol go   = 1
symbol halt = 0     : symbol open = 1     : symbol closed = 0
symbol ldir = pin0  : symbol len  = pin1  : symbol lsen  = pin2
symbol rdir = pin3  : symbol ren  = pin4  : symbol rsen  = pin5
symbol cnt  = b0    : symbol mode = b1   : symbol lmode = b2
symbol rmode = b3  : symbol emstop = b4

start:  poke $81,$7f
        pin0=0 : pin1=0 : pin3=0 : pin4=0 ' logic 0 on pins 0,1,3,4
        dir0=1 : dir1=1 : dir2=0
        dir3=1 : dir4=1 : dir5=0
        ldir=fwd : len=go
start1: if lsen=open then start.1
        len=halt : rdir=fwd : ren=go
start2: if rsen=1 then start2
        ren=halt
loop:   serin 6,n9600,("M"),b1,#b0
        SEROUT 7,N9600,("M",B1,#B0,13,10)' respond w th cmd
        if b1=$46 or b1=$66 then F      branch to fwd if 'F' or 'f'
        if b1=$42 or b1=$62 then B      branch to bwd if 'B' or 'b'
        if b1=$52 or b1=$72 then R      branch to rt if 'R' or 'r'
        if b1=$4C or b1=$6C then L      branch to lt if 'L' or 'l'
        got0 loop
F:      ldir=fwd : rdir=fwd
        got0 loop1
B:      ldir=bwr : rdir=bwr
        got0 loop1
R:      ldir=fwd : rdir=bwr
        got0 loop1
L:      ldir=bwr : rdir=fwd
        got0 loop1
loop1:  lmode=0 : rmode=0
        if cnt=0 then loop3
        if cnt<>1 then loop2
        got0 loop
loop2:  cnt=cnt-1
loop3:  len=go : ren=go
loop4:  peek $05,emstop
        emstop = emstop & $10
        if emstop = 0 then allstop
        if lmode>1 and rmode>1 then loop1
        if len=go then Ll
        got0 loop5
Ll:     if lsen=lmode then loop5
        pause 1
        lmode=lmode+1
        if lmode<2 then loop5
        len=halt
loop5:  if ren=go then Rl
        got0 loop4
Rl:     if rsen=rmode then loop4
        pause 1
        rmode=rmode+1
        if rmode<2 then loop4
        ren=halt
        got0 loop4
allstop: len=halt : ren=halt
        got0 loop

```

broken, we can calculate the distance the tread moves.

Now, this all sounds good on paper, but I ran into a little snag. I couldn't get the IR sensors to sense the gear teeth.

The orange plastic used in the gears passed IR like it wasn't there. To use

this method, I would have to paint the gear's teeth. But, I was worried about the paint scratching off, so I discarded the IR sensors for a mechanical switch.

I picked up a couple microswitches with levers. Not only did the lever give a mechanical advantage, but an idler



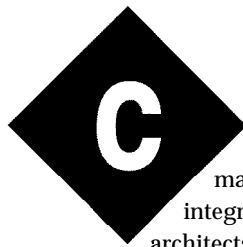
Tom Cantrell

## High-Velocity DSP



Tom's view:  
bit by bit,  
we've  
crept  
ahead

through boosting clock rate and memory, but to make major leaps, we need a paradigm shift. Enter DSPs. They offer the optimal combination of data and signal processing.



Compelled by the march of silicon integration, computer architects are doing their best to find a way through a maze of rocks and hard places.

Instruction-level parallelism-how much there is, how to find it, and how to exploit it-is a key area of interest. Another is the chip-level equivalent of convergence, blurring the distinction between ICs processing data and those processing signals (see M.R. Smith's "To DSP Or Not To DSP," *INK* 28).

For the most part, microprocessor gurus have had a pretty easy go of it. They've gotten away with brute forcing (thanks to nearly free transistors) more mileage out of old mainframe ideas. The modern generation of pipelined, superscalar, speculative semiSIMD CPUs is the result.

But now, other than boosting clock rate and on-chip cache/memory size, it's getting tough to squeeze more MIPS out of evolutionary designs. Pressure is building for an architectural paradigm shift.

Meanwhile, as computers take on the challenges of multimedia, designers look for solutions with the optimal combination of data and signal processing. Perhaps the time is right to take a closer look at one of the newer concepts-VLIW (Very Long Instruction Word).

A number of chips have toyed with the idea, but so far, it's remained little more than a lab curiosity. Now, the concept is getting a big push from TI in the form of their new 'C6x series of DSPs featuring a VLIW architecture they call VelociTI.

### SHADES OF MICROCODE

Like most ideas in computing, VLIW isn't brand new. It's just newer than most of the rest.

The original concepts harken back to the days of microcode (remember how CISCs used to work?). The challenge is to transform vertical microcode into a faster horizontal format with separate fields for each functional unit.

Hennessy and Patterson [1] relate the early history of VLIW as embodied in research and commercial machines such as those offered by Floating Point Systems, Cydrome, Multiflow, and companies you've likely never heard of.

The reason you've never heard of them is that these machines, and other '80s vintage CPUs with VLIW-esque features (e.g., the Intel '860 and experimental MIPS prototypes), never obtained much commercial success.

Some argue this proves the VLIW concept is just another example of a bad idea whose time has come, but I suspect it's not that simple. Perhaps a combination of at-the-time immature and constrained technology along with the end of the Cold War (sapping the market for performance-at-any-price crunchers) was more to blame.

While main CPU architects remain skeptical, the VLIW approach has found favor in the niche of chips known as multimedia accelerators from the likes of Chromatic and Trimedia. Though the jury is still out, these chips seem well on their way to rehabilitating the VLIW concept. Needless to say, the latest blessing from TI is both significant and timely.

### MISSION IMPOSSIBLE

That old joke about RISC standing for Relegate the Impossible Stuff to the Compiler might better be said about VLIW.

From high altitude, the problem is rather simple. The goal: execute as many instructions per clock as possible.



So, CPU operations are scheduled to fully exploit opportunities for parallel operation.

Unfortunately, a variety of dependencies and constraints get in the way. For instance, you can't read a variable before it's written, and you can't demand  $n$  functional units when the chip only has  $n-1$ .

How best to schedule instructions subject to these constraints is where the arguments arise. Conventional superscalar CPU wisdom calls for a bunch of complex, ugly hardware to dynamically examine and reorder instructions at runtime. The good news is such a chip can handle old binaries, although a recompile is usually necessary for top performance.

By contrast, VLIWs rely on static, or compile-time, scheduling to organize instructions most efficiently ahead of time. Instructions that can execute in parallel are lined up arm to arm for

digestion by the multiple functional units in one big gulp.

Reasonable observers can disagree on whether scheduling dynamically (hardware) or statically (compiler) makes more sense. For instance, runtime hardware can adapt to conditional branch behavior, but a static scheme must commit one way or the other.

On the other hand, dynamic scheduling can only deal with a small window of instructions. Static scheduling can examine the entire program. Runtime optimization incurs a silicon penalty for each chip shipped, whereas static schemes only pay a compile-time penalty, presuming such a complex compiler can get beyond beta.

In fact, the overall trend seems to be to combine the two schemes. By making both the chip and compiler smarter, we can let each do what it does best.

One key question keeps popping up. Just how much instruction-level paral-

lelism (ILP) is to be found? The answer: It depends.

For example, Hennessy and Patterson examined traces of SPEC92 benchmarks and found large amounts of ILP (17.9-150.1 instructions per cycle). However, since this is based on hindsight (actual program traces), it models a perfect machine with its infinite resources (registers and function units).

It has foolproof branch prediction and a full program-size reorder buffer. Also, it has no aliases, referring to the situation, as with a C pointer, where it's next to impossible to determine if a data dependency exists.

Even the most java-jolted architect knows such a machine is a look-ahead pipedream. Hennessy and Patterson perform a similar analysis with a real CPU (PowerPC 620). Although it's theoretically capable of issuing 4 instructions per cycle, it barely averages 1.3. Ouch!

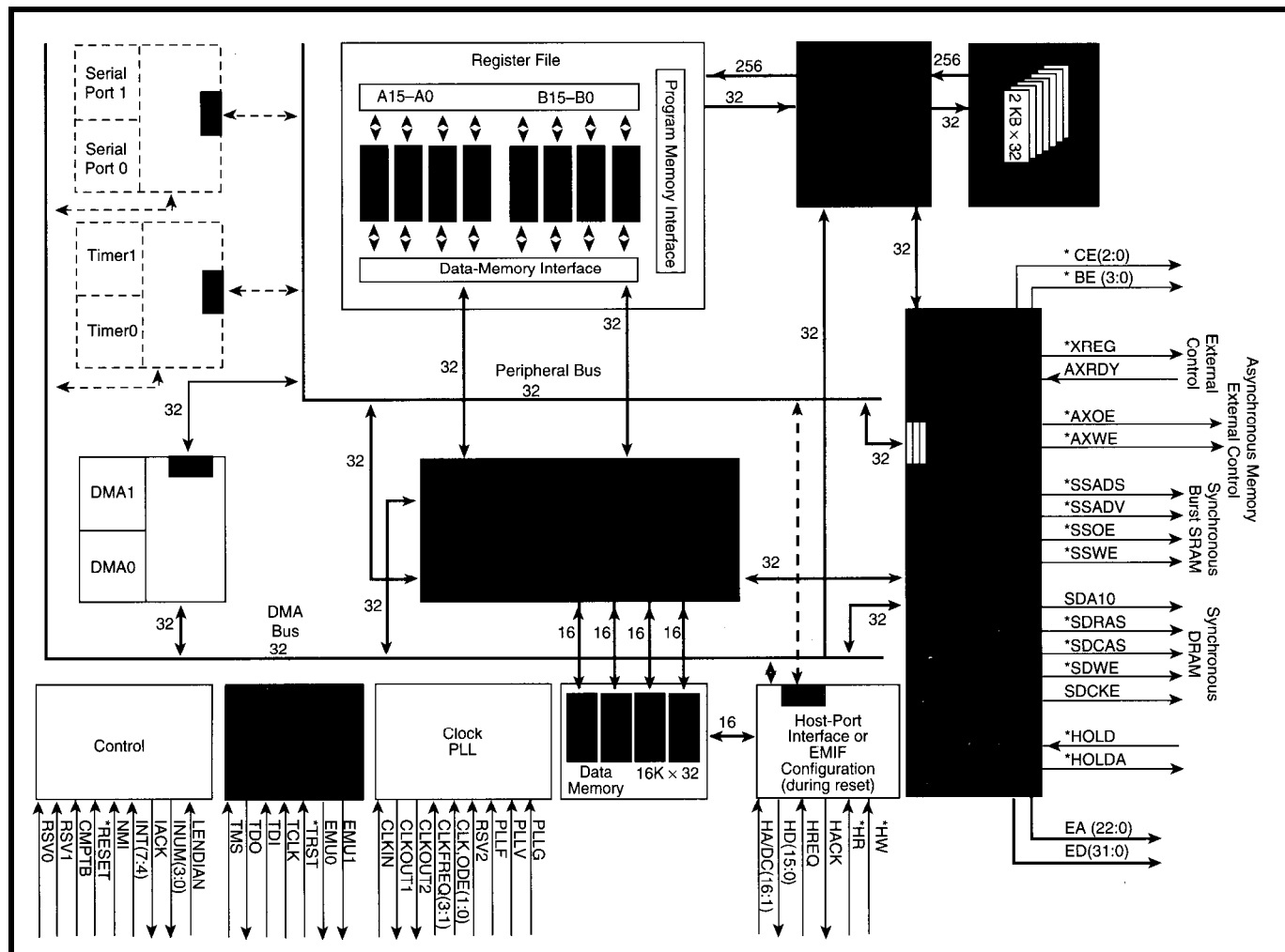


Figure 1—The radical architecture of the 'C6201 represents a major blessing of-and commitment to-the VLIW concept by TI

However, one key point (and hope) to note is that different kinds of programs exhibit more or less ILP. In particular, vector loops (e.g., vector add, dot product, etc.) are relatively more parallelizable, and such routines are at the core of signal processing.

I've spent a lot of time up front looking at the big picture in the hope of making the motivation behind VLIW a little easier to understand. Needless to say, the TI chips aren't your father's CPU.

## V8 PUNCH

"There's no substitute for cubic inches" is a hot-rod maxim that applies well to the first chip in the 'C6x series—the 320C6201. The chip combines a whopping 8 functional units with 1 Mb of on-chip zero-wait-state SRAM and a bunch of glue logic, all running at up to 200 MHz (see Figure 1).

Featuring the silicon equivalent of multiport fuel injection, each functional unit gets its own 32-bit fetch bus. It makes for a 256-bit "instruction," which is what the VL in VLIW is all about.

The on-chip SRAM is split in half, with 2k 256-bit instructions joined by 16 K x 32 data RAM. The instruction memory can be reconfigured to operate as a direct-mapped cache.

SIMD-like partitioning of the data-memory interface enables simultaneous transactions to different banks. Registers and off-chip data memory are all byte, half-word, and word addressable.

It's possible to store into the program memory 32 bits at a time with the STP instruction. But, there's no way [and since it's RAM, no need) to load from program memory.

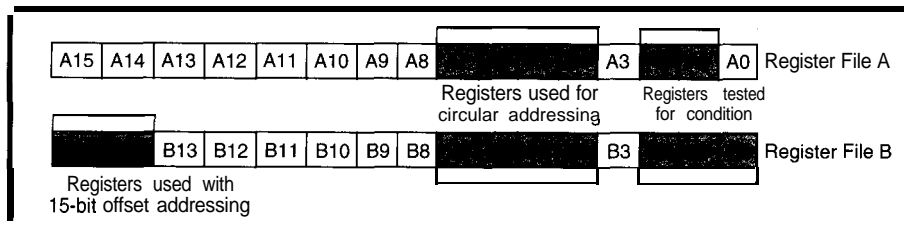


Figure 2—Each cluster has a complement of 16 Kx 32-bit registers. All registers are general purpose! though certain ones support alternative functions, including circular addressing, long branch offsets, and conditional execution.

Note that switching the program RAM from memory to cache mode invalidates the cache. However, it is possible to freeze the cache anytime, locking the contents for fast and deterministic access.

Much as a V8 is conceptually two straight-4s stuck together, the C6201 is actually a pair of 128-bit VLIWs. Each has four function units, 16 K x 32-bit registers, and a 32-bit data bus.

Like a crankshaft, cross paths link clusters, enabling function units on one side to access registers on the other. Similarly, a register on one side generates an address for loads/stores on the other.

As shown in Table 1, each function unit is responsible for part of the instruction set. It's an interesting hybrid of three-operand load/store RISC spiced up with DSP features.

The latter includes circular addressing (i.e., software FIFO), saturated math (results top and bottom out, rather than overflow), 40-bit calculation headroom [using a pair of registers), and so on. Barrel shifters support a variety of one-clock bit field operations, including shifts, searches, and extracts.

Other features not particularly related to DSP functions are interesting nonetheless. For instance, the traditional concept of conditional

branches based on PSW flags is completely discarded.

Instead, the CPU has conditional execution for every instruction based on the contents (zero or nonzero) of certain registers. This works in concert with CMP instructions that compare two source registers and put a 0 or 1 in a destination register accordingly.

Thus, there aren't any conditional branch (B) instructions per se. But like any other instruction, branches can be made conditional. Besides the expected register and displacement options, the B I R P and B N R P variants act as returns from maskable and nonmaskable interrupts, respectively.

Like some other RISCs (MIPS comes to mind), the 'C6201 does little in response to interrupts except store the return address one-level deep on chip and mask further interrupts. Any nesting, dynamic priority, or other fancy interrupt pretensions are left completely up to software.

As you see in Figure 2, other registers play special roles for circular addressing and long (15 vs. 5 bit) offset addressing. Circular addressing mode is enabled with a control register that also specifies block size (powers of two between 2<sup>1</sup> and 2<sup>32</sup> bytes).

Subsequently, add and subtract operations on the affected registers (whether via an explicit ADD and SUB or a load/store address increment and decrement) are calculated in a modulo manner (i.e., they wrap around once the block size is exceeded).

## NOP IN MY BACKYARD

Table 2 shows the 'C6201 "pipeline," though that term is a bit misleading.

The front of the pipeline is a single 256-bit-wide datapath that grabs fetch packets from memory. In the middle, the long fetch packet splits into 32-bit

.L Unit	.M Unit	.S Unit	.D Unit
ABS	NORM	MPY	ADD
ADD	MPT	SMPY	ADDK
AND	OR		EXTU
CMPEQ	SADD		SHL
CMPGT	SAT		SHR
CMPGTU	SSUB		SHRU
CMPLT	SUB		SSHL
CMPLTU	SUBC		STP*
LMBD	XOR		SUB
MV	XOR		SUB2
NEG			XOR
			ZERO
			ZERO

Table 1—The 'C6201 relies on four types of functional units to handle the entire instruction set. Two such groups (each sometimes referred to as a "cluster") compose the total of eight function units. Single-asterisk instructions only apply to the .S2 components, and double asterisks apply to .D2.

Listing 1 a-A typical DSP loop written in C translates easily into serial assembler (b).c—Optimization starts by scheduling around resource constraints and filling delay slots. d-The loop is unrolled to handle two array elements per iteration and expose more parallelism. e—Finally, software pipelining packs the entire loop into one 256-bit VLIW instruction. The asterisks show how each unit works on a different iteration of the loop, eliminating dependencies and every single NOP.

```

a) int dotp(short a[], short b[]){
    int sum = 0, i;
    for (i=0; i<100; i++) sum += a[i] * b[i];
    return(sum);
}

b)          MVK   .S1  100, A1    ; set up loop counter
          ZERO  .L1  A7          ; zero out accumulator
LOOP:      LDH   .D1  *A4++,A2    ; load ai from memory
          LDH   .D1  *A3++,A5    ; load bi from memory
          NOP   4                ; delay slots for LDH
          MPY   .M1  A2,A5,A6    ; ai * bi
          NOP   1                ; delay slot for MPY
          ADD   .L1  A6,A7,A7    ; sum += (ai * bi)
          SUB   .S1  A1,1,A1     ; decrement loop counter
[A1]      B     .S2  LOOP        ; branch to loop
          NOP   5                ; delay slots for branch
;Branch occurs here

c)          MVK   .S1  100, A1    ; set up loop counter
          ZERO  .L1  A7          ; zero out accumulator
LOOP:      LDH   .D1  *A4++,A2    ; load ai from memory
          LDH   .D2  *B4++,B2    ; load bi from memory
          SUB   .S1  A1,1,A1     ; decrement loop counter
[A1]      B     .S2  LOOP        ; branch to loop
          NOP   2                ; delay slots for LDH
          MPY   .M1X A2,B2,A6    ; ai * bi
          NOP   1                ; delay slots for mpy
          ADD   .L1  A6,A7,A7    ; sum += (ai * bi)
;Branch occurs here

d)          MVK   .S1  50,A1     ; set up loop counter
          ZERO  .L1  A7          ; zero out sum0 accumulator
          ZERO  .L2  B7          ; zero out sum1 accumulator
LOOP:      LDW   .D1  *A4++,A2    ; load ai & ai+1 from memory
          LDW   .D2  *B4++,B2    ; load bi & bi+1 from memory
          SUB   .S1  A1,1,A1     ; decrement loop counter
          B     .S1  LOOP        ; branch to loop
          NOP   2                ;
          MPY   .M1X A2,B2,A6    ; ai * bi
          MPYH  .M2X A2,B2,B6    ; ai+1 * bi+1
          NOP   1                ;
          ADD   .L1  A6,A7,A7    ; sum0 += (ai * bi)
          ADD   .L2  B6,B7,B7    ; sum1 += (ai+1 * bi+1)
; Branch occurs here
          ADD   .L1X A7,B7,A4    ; sum = sum0 + sum1

          B     .S2  LOOP        ; branch to loop
          MVK   .S1  51,A1     ; set up loop counter
          B     .S2  LOOP        ; * branch to loop
          B     .S2  LOOP        ; ** branch to loop
          ZERO  .L1  A7          ; zero out sum0 accumulator
          ZERO  .L2  B7          ; zero out sum1 accumulator
          B     .S2  LOOP        ; *** branch to loop
          ZERO  .L1  A6          ; zero out ADD input
          ZERO  .L2  B6          ; zero out ADD input
          B     .S2  LOOP        ; **** branch to loop
          ZERO  .L1  A2          ; zero out MPY input
          ZERO  .L2  B2          ; zero out MPY input
LOOP:      ADD   .L1  A6,A7,A7    ; sum0 += (ai * bi)
          ADD   .L2  B6,B7,B7    ; sum1 += (ai+1 * bi+1)
          MPY   .M1X A2,B2,A6    ; ** ai * bi
          MPYH  .M2X A2,B2,B6    ; ** ai+1 * bi+1
[A1]      ADD   .S1  -1,A1,A1    ; ***** decrement loop counter
[A1]      B     .S2  LOOP        ; ***** branch to loop
          LDW   .D1  *A4++,A2    ; ***** ld ai & ai+1 fm memory
          LDW   .D2  *B4++,B2    ; ***** ld bi & bi+1 fm memory
; Branch occurs here
          ADD   .L1X A7,B7,A4    ; sum = sum0 + sum1

```

execute packets for dispatch to each functional unit.

Each single-stage functional unit then requires a particular number of cycles depending on the operation (i.e., one for simple ALU ops, two for multiplies, five for loads, six for branches).

Rather than interlocks, the 'C6201 relies on delay slots to handle the varying cycle count in the execution stage. The compiler tries to find useful instructions to fill the void, but it's tough to do. Sometimes, the only choice is to kill time with NOPs. To this end, TI includes a NOP #N (n = 1 to 9) instruction to prevent needless duplication.

Along similar lines, concerns about VLIW code density have been made. After all, if the machine only delivers "a few" instructions per cycle on real programs, then there are "8 minus a few" NOPs in baggage. Memory may be cheap, but it's not cheap enough to throw half or more away.

TI's solution is to make the least significant bit of each 32-bit instruction a parallel bit. If it's 1, the next instruction in the fetch packet is added to the current execute packet. If 0, the next instruction goes into the following execute packet.

All in all, the 'C6201 goes a long way in placating the NOP naysayers.

## MOTOR MOUNTS

Though the 352-pin ball grid array (BGA) package may make you fear the worst, the 'C6201 glue logic pretty much insulates the system designer from the on-chip complexity. Actually, about half the pins are devoted to the dual power supply, comprising 2.5 V for internal operation and 3.3 V for I/O.

To date, final production device power hasn't been characterized. But given the clock rate and wide data-paths, a half dozen or so watts won't be a surprise.

The chip has the increasingly standard triad of power-reduction modes that stop the CPU, I/O, or both. They trade off less stand-by power for fewer wake-up options.

Making the 'C6201 an easy drop-in starts with a clock generator featuring a programmable 1: 1, 2: 1, or 4: 1 PLL. So, the clock source is limited to 50 MHz, cutting design and FCC hassles.

Like many DSPs with on-chip program RAM, a built-in DMA controller handles bootloading from external memory, which can be slow and/or narrow (e.g., x8 EPROM). It also takes care of user-defined data-transfer chores.

The CPU and DMAC both get access to external memory through the EMIF (External Memory Interface). It features 23 address lines and 32 data lines coupled with individual byte enable lines (BE\*O-3) and three chip selects (CE\*0:2).

All three chip-select spaces support 32-bit data width and asynchronous (i.e., EPROM, SRAM) memory. As well, CE\* 1 can be configured for 16- or 8-bit width, while CE\*0 and CE\*2 can operate in high-speed burst SRAM and synchronous DRAM modes.

Finally, a dedicated port is provided for access by a host CPU. Having asserted Host Request (HREQ) and received Host Acknowledge [HACK], it can access the on-chip memory using 16-bit address and data buses with read and write strobes.

Note the protocol asks for a degree of cooperation from each party. The host isn't granted access until all pending on-chip data-memory accesses cease. But once the host has control, it can keep it indefinitely, locking out the on-chip CPU and DMAC.

A few additions are planned to the first version of the 'C6201, including dual serial ports and timers (shown in dotted lines in Figure 1). Also, some existing functions (e.g., the SDRAM

interface and memory-map options] will be improved.

## HIGH-OCTANE SOFTWARE

If the 'C6201 is the motor, then your software is the fuel. You need the good stuff to avoid NOP knock. Remember the basic premise of VLIW is that you and the compiler not only get to-but must-generate optimal code.

For an idea of what's involved, look at Listing 1a. It shows a classic vector 16-bit multiply/32-bit accumulate loop (dot product) written in C, similar to the inner loops of many DSP applications. Listing 1b shows the same loop translated to serial assembly language.

Except for the functional unit designations (e.g., . L1, . D1, . M1, etc.) and conditional execution feature ([A 1 ] makes the branch conditional), the code is similar to what you find on a conventional RISC with delay slots. And, just as on that RISC, the next step is to schedule around resource (functional unit, register, and bus) constraints and to fill delay slots as shown in Listing 1c.

The parallel bars in the first column indicate the instruction can execute in parallel with the previous one (i.e., the opcode p bit described earlier). The first two instructions, using different units, parallelize easily.

Notice how a second unit (. D2) is allocated to allow the two LDH instructions to proceed. Instructions are also moved around to fill delay slots.

Execution time is cut in half, which is good, but so far, not much better than

what you'd find on a run-of-the-mill CPU. One simple optimization: use LDW (32 bit) instead of LDH (16 bit) and work on two elements of each array at a time.

The process known as loop unrolling (see Listing 1d) isn't so much about cutting overhead as exploiting more parallelism. Here, the inner loop cycle count remains unchanged. But, there are only half as many iterations, so performance doubles again.

It's definitely interesting, but still not spectacular. After all, IPC is still less than 2, a small fraction of the 'C62018-issue capability.

But now, the fun begins. The premise of VLIW proponents is that, with full program visibility and explicit knowledge of and control over machine resources, much more aggressive optimization is possible.

Research has centered on advanced techniques like memory disambiguation (to get around the dependency-inducing alias problem) and trace scheduling (to move code across basic blocks). Hennessy and Patterson [1] and others [2] describe these techniques in gory detail.

One key optimization-software pipelining-is especially useful for tight vector loops. The concept is, like a hardware pipeline, rather simple in principle if not in practice. The goal is simply to start a new iteration of the loop as soon as possible.

Evaluating resource and dependency constraints determines the minimum iteration interval (i.e., the minimum number of cycles between iterations). So, the code breaks up into a prologue (i.e., prime the pipeline) and epilogue (i.e., drain it) surrounding a fully parallel inner loop.

Turns out, the very fastest schedule can bloat code size a lot. But, subsequent optimizations (e.g., extraneous load removal and prologue and epilogue reduction) cut size significantly with only a minor reduction (-10%) in speed.

Pipeline Phase	Pipeline Stage	Symbol	During this Phase:
Program Fetch	Program Address Generate	PG	Address of the fetch packet is determined
	Program Address Send	PS	Address of the fetch packet is sent to memory
	Program Wait	PW	Program memory read is performed
	Program Data Receive	PR	Fetch packet is expected at CPU boundary
Program Decode	Execute Packet Dispatch	DP	Next execute packet is sent to functional units
	Decode	DC	Instructions are decoded in functional units
Execute	Execute 1	E1	Instruction conditions are evaluated, operands read Load/store addresses are computed/modified Branches affect fetch packet in PG stage Single-cycle results are written to register file
	Execute 2	E2	Load address is sent to memory Store/STP address and data are sent to memory Single-cycle instructions can set SAT bit Multiply results are written to register file
	Execute 3	E3	Load memory reads continue Multicycle instruction can set SAT bit
	Execute 4	E4	Load data arrives at CPU boundary
	Execute 5	E5	Load data is placed in register

Table 2-The 'C6201 pipeline consists of a single front-end that fetches and cracks long (up to 256 bit) instructions into pieces for execution by each functional unit. Delay slots, rather than interlocks, accommodate slow (>1clock) operations, including multiplies, loads, and branches.

To make a long story short, Listing le shows a code-efficient software-pipelined version of the dot-product example. You may need more than a few moments to decipher it, but the key point is that the entire loop has been parallelized into a single cycle 256-bit instruction with all cylinders firing for a nearly 8x speedup!

## COMPILER COMPARO

That's impressive. Compared to the unoptimized serial assembly, the final version speeds up the overall routine by a factor of .25, only slightly derated (due to epilogue and prologue) from the inner-loop speedup of 32x.

Yes, the chip may seem expensive (\$96 at 25k). But, what if it can handle IO-15 modems in software compared to one for a \$10 DSP?

There's certainly nothing wrong with hand coding and tuning your application's critical loops. Indeed, doing anything less invariably leaves many MIPS on the table. However, there's also no doubt all the head scratching gets old quick.

The million-dollar question: will the TI tools, including the optimizing assembler that schedules delay slots and allocates registers, the C compiler that features global (i.e., entire program) scope, and the software-pipelining optimizations (\$2495 for the C and ASM combo for Windows 95/NT), not to mention the JTAG-based debugging scheme, live up to the promise?

My guess is the combination of smarter tools, libraries of hand optimized code, and continuing march of silicon (notably large and wide on-chip memory), combined with the demand for more media savvy applications and the blessing of a heavy hitter like TI, may mean VLIW's time has finally come. □

*Tom Cantrell has been working on chip, board, and systems design and marketing in Silicon Valley for more than ten years. He may be reached by E-mail at tom.cantrell@circellar.com, by telephone at (510) 657-0264, or by fax at (510) 657-5441.*

## REFERENCES

- [1] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, 2nd Ed., Morgan Kaufmann Publishers, 1996.
- [2] J. Ellis, *Bulldog: A Compiler for VLIW Architectures*, The MIT Press, 1986.

## SOURCE

'C6x series, Programmers Guide SPRU198

Texas Instruments, Inc.  
Semiconductor Gr. SC-97001A  
Literature Response Ctr.  
P.O. Box 172228  
Denver, CO 80217  
(800) 477-8924, x4500  
Fax: (303) 294-3738

## IRS

428 Very Useful  
429 Moderately Useful  
430 Not Useful

## Real-Time Multitasking Tools for Embedded Systems and DOS

### RTKernel

Professional, high-performance real-time multitasking system for DOS and 16-bit Embedded Systems. For Borland UC++, Microsoft UC++, and Borland Pascal. Libraries: \$550 Source Code: add \$500

### RTTarget-32

Cross Development System for 32-bit Embedded Systems. Supports Intel 386 and higher, as little as 16KRAM/ROM. For Borland UC++, Microsoft C/C++, and Watcom C/C++. Libraries: \$1700 Source Code: add \$1000

### RTKernel-32

Professional, high-performance real-time multitasking system for 32-bit Embedded Systems. Supports Intel 386 and higher. For Borland UC++, Microsoft UC++, and Watcom UC++. Libraries: \$1950 Source Code: add \$1600

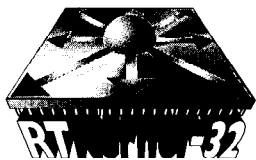
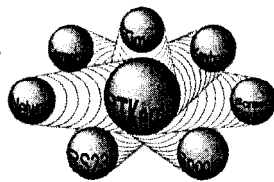
In North America, please contact:  
On Time  
88 Christian Avenue, Setauket, NY 11733, USA  
Phone (516) 689-6654  
Teletax (516) 689-1172  
email info@on-time.com



Other Countries:  
On Time Informatik GmbH  
Hofweg 49, 22085 Hamburg, GERMANY  
Phone +49-40-2279405  
Fax +49-40-2279263  
email 102212.3101@compuserve.com

NO ROYALTIES! FREE DEMO DISK!  
<http://www.on-time.com>

On Time  
REAL-TIME AND SYSTEM SOFTWARE



## Parts List Manager

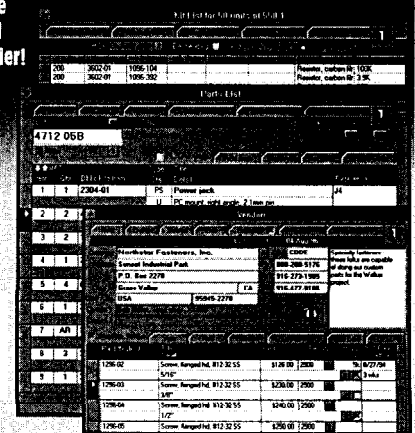
For Engineers, Product Designers, Prototypers

Windows-based software helps you stay organized ...And makes the job easier!

and manage multi-assembly parts lists for products in development ...And after!

Keep track of:

- Part Specs & Dwg's
- Suppliers & Mfrs
- Boms and Kit lists
- Product Costs
- Engineering Stock



## Parts & Vendors

Version SE: \$99 + s/h EXTended: \$299 + s/h

Call 800-280-5176



Trilogy  
DESIGN

916-273-1985  
fax 916-477-9106

Requires 486, 10 meg min. ram. Win 3.x or Win95

P.O. Box 2270, Grass Valley, CA 95945  
<http://www.trilogydesign.com>

#143

# PRIORITY INTERRUPT

## A Winning Proposition



he editorial direction of Circuit **Cellar** is primarily an extension of my own technical interests. It's a lime line of subjects that started 19 years ago at **BYTE** and continues today. Of course, if you look back at those early projects now, you might come away with the impression that I specialized in presenting some really off-the-wall computing concepts. Back then, these articles were considered state-of-the-art, I assure you.

Today, Circuit Cellar **INK** continues to focus on computer applications, but as you might expect, the technical level of the presentation has grown considerably. The reason is because I base our delivery level on an ever-increasing standard built on accumulated experience and expanding knowledge. We don't rehash the same stuff and periodically count on a new generation of readers to present the same documentation to over and over. When a simple **PIA** is the preferred parallel interface, that's what we write about. When accepted practice becomes a custom-programmed coprocessor instead, that's the way we present it.

This is not an easy balance. Often, you're damned if you do and damned if you don't. Just like job applicants finding potential employers who applaud their cross-technology training but won't hire them because their degree isn't specific enough, we find advertisers who applaud our embedded focus but are tough to sign because their specific product category isn't in the magazine's name. If we published **DSP World**, **Emulator Action News**, or **Software Tools Monthly**, it would be easy.

When we started the Embedded **PC** section to acknowledge that the 80x86 architecture was a viable application alternative, I removed a major obstacle to many who didn't understand our broader focus. You and I know it's just the next step in the accumulated experience base called "embedded control." But to them, it's like waving a flag with an identifiable product category on it. Not only did they become advertisers, but when we sought support for an **Embedded PC** contest, we had to stand aside so as not to get trampled in the rush. That's how we got 17 sponsors and almost \$11,000 in prize money.

Does this mean I plan to change the magazine into an embedded-PC manifest? Hell, no!

The massive support for an embedded-PC contest is the result of having a specific product focus identifiable to specific sponsors. Whenever I've presented a design contest in the past, it has had a general focus aimed at a general group of potential sponsors and with a general objective. There's a message there someplace.

The reality is that a successful general design contest has to have either a specific focus or specific sponsors. I know this sounds ridiculous. Making it specific seems to take it out of the "general" category, doesn't it? While the purists among you might fight my logic, I find that necessity promotes compromise. While a general contest certainly shouldn't have a specific focus, there's no reason a general contest can't have a specific sponsor with a general product line.

At this writing, we are negotiating with a major semiconductor manufacturer to sponsor a spring 1998 Circuit Cellar Design Contest. With their support, it is our intention that the prizes and promotion will be equivalent to **INK's** present **Embedded PC** Design Contest. I can't give you any details until they sign on the dotted line, but my objective is to have a contest that provides a wide option for technical solutions and various levels of application expertise.

Ultimately, it's reader support that still makes it a pleasure to plan and direct Circuit Cellar. I'm sure our parallel destinies --**INK's** and my own-- will take us where we'd never have gone alone. But rest assured, any moves we make will only and always be in response to you. We will stay your course.

P.S.: Speaking of the **Embedded PC** Design Contest, the deadline for submissions has been extended from August 1st until September 1st by popular demand. For any notices or information about the contest, see our Web site at [www.circuitcellar.com](http://www.circuitcellar.com).

[steve.ciarcia@circuitcellar.com](mailto:steve.ciarcia@circuitcellar.com)