CIRCUIT CELLAR INK®

# THE COMPUTER APPLICATIONS JOURNAL

## HOME AUTOMATION

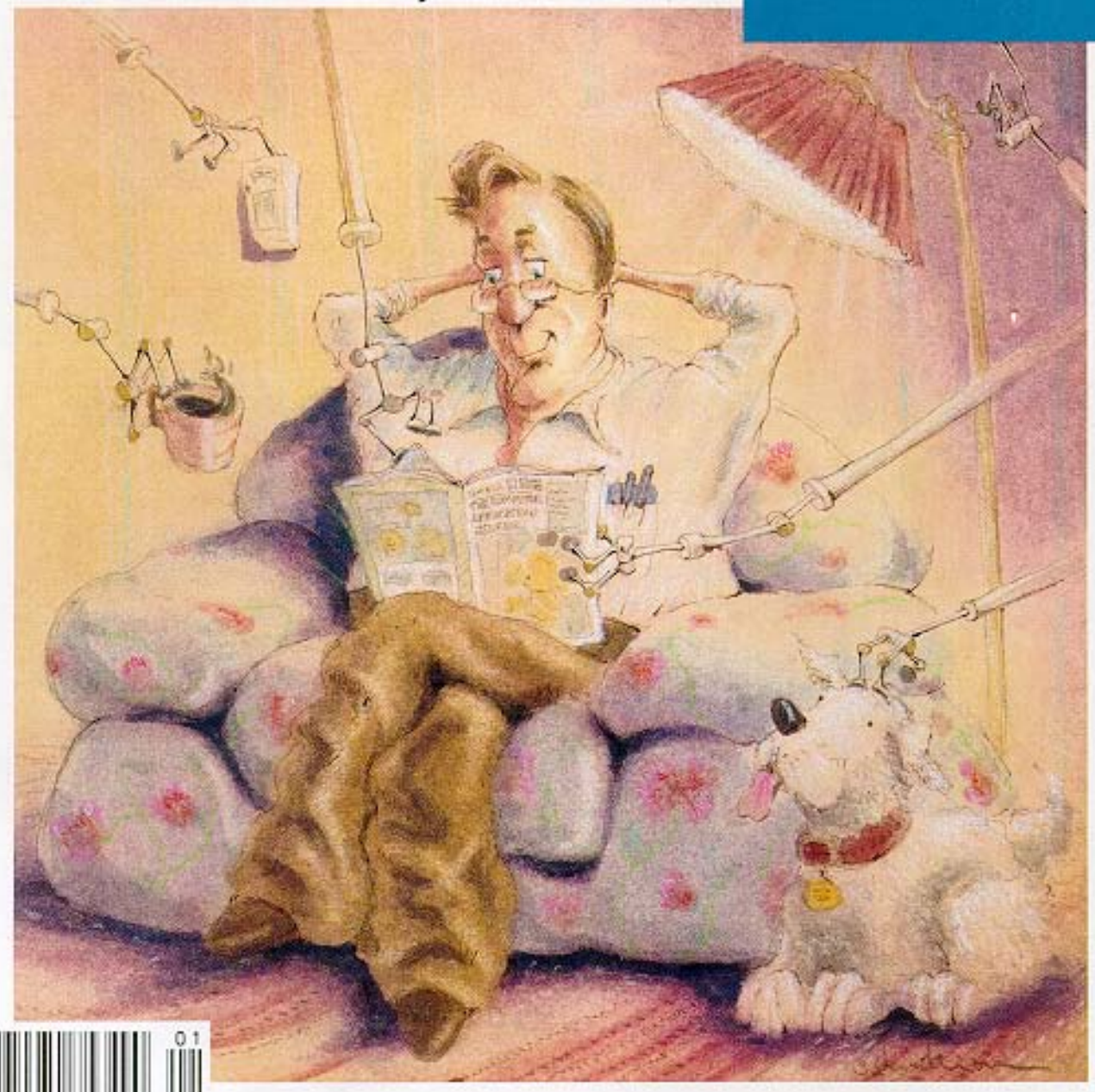
CEBus Interface Chips (at last!)

Smart Serial X-10 Control

High-speed PCB Design

Interfacing to Large LCD Panels

January 1994 — Issue #42



0 74470 75349 0 01

$3.95 U.S.
$4.95 Canada

# EDITOR'S INK

## Catch the Bug

Have you been bitten yet? I mean, you've been reading all about home automation in our pages for years now, but have you actually hooked something up and impressed your friends? With me, as I'm sure it was with others, it all started innocently enough, but once the bug had taken hold, there was no shaking it.

It started some years ago right around the time of year that has just passed: Christmas. We had a pile of electric candles in the front windows of our house. Every day at dusk, someone had to run around plugging them all in. Each night at bedtime, someone had to shut them all off (being careful not to wake those sleeping under the windows). I sprinkled some X-10 lamp modules around the house, added some really dumb intelligence, and we didn't have to touch the candles once all season. I was hooked.

You needn't give your house the intelligence of the HAL 9000 right off the bat. Pick up a few inexpensive devices like X-I 0 modules and do some experimenting. I'm sure it won't be long before you won't know how you got along without it. (I still walk into the bathroom at a friend's house and stand there in the dark wondering why the lights haven't come on yet.)

As any long-time Circuit Cellar reader knows, we have been providing continuous coverage of the latest happenings on the CEBus front. In the past, we've given you a peek at the paper specification. In our last home automation issue, we showed you a CAL compiler. It's finally time to get your hands on some hardware. In our first feature article, we present some new chips available now designed to make adding a CEBus power line interface to your project much easier.

You can't very effectively automate your house's HVAC without a basic understanding of how temperature control works. In our second feature, we take a quick look at just how simple it can be.

Of course, we can't have a home automation issue without something dealing with X-10. In our next feature, we present a chip similar to the PLIX chip that Jeff covered a few months back, but replaces the parallel interface with a serial port.

Finally, today's high-speed processors are bringing to the surface problems once faced by ECL designers: that of printed circuit board design techniques that more closely resemble those of analog designers than those of bit jockeys. Check out some of the concerns you must keep in mind when designing PCBs for high-speed circuits.

In our columns, Ed gets back to the hardware side of his embedded '386SX by starting the process of adding a large LCD panel. Jeff gets into the home control spirit by looking at a new thermostat chip from Dallas Semiconductor. Tom takes a break from sniffing out new silicon to build a show demo. John looks at the benefits of a real-time clock in embedded applications. Finally, Russ picks out patents that explore the human/machine interface.


CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

TECHNICAL EDITOR
Michael Swartzendruber

ASSOCIATE EDITOR
Rob Rojas

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
John Dybowski & Russ Reiss

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
Lisa Ferry

GRAPHIC ARTIST
Joseph Quinlan

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

Cover Illustration by Bob Schuchman
PRINTED IN THE UNITED STATES

PUBLISHER
Daniel Rodrigue:

PUBLISHER'S ASSISTANT
Susan McGil

CIRCULATION COORDINATOR
Rose Mansella

CIRCULATION ASSISTANT
Barbara Malesk

CIRCULATION CONSULTANT
Gregory Spitzfader

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Dan Gorsky

CIRCUIT CELLAR INK, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) is published monthly by Circuit Cellar Incorporated. 4 Park Street Suite 20, Vernon, CT 06066 (203) 875-2751 Second class postage paid at Vernon, CT and additional offices. One-year (12 issues) subscription rate U.S.A. and possessions $21.95, Canada/Mexico $31.95, all other countries $49.95. All subscription orders payable in U.S. funds only, via international postal money order or check drawn on U.S. bank. Direct subscription orders and subscription related questions to The Computer Applications Journal Subscriptions, P. O. Box 7694, Riverton, NJ 08077 or call (609) 786-0409.
POSTMASTER: Please send address changes to The Computer Applications Journal, Circulation Dept., P.O. Box 7694, Riverton, NJ 08077.





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

# INSIDE ISSUE 42

# READER'S INK

## Chordic Comments

Scot Colburn's article "The Covert Chordic Keyboard" (December '93) is excellent! I really love alternative I/O device articles.

Chording is going to become a major trend in computer input. To make sure that credit is given where credit is due, I must point out that over 25 years have passed since the Chord Keyboard was first built and used.

The original Chord Keyboard was invented by Doug Engelbart (who also invented the mouse; in fact, he invented them together as complements to each other] while at SRI.

I quote from Engelbart, D., and English, W., "A Research Center for Augmenting Human Intellect," ***Proceedings of*** *FJCC,33*(1):395-410, AFIPS Press, Montvale, NY, Fall 1968:

"The five-key handset has 31 chords or unique keystroke combinations, in five 'cases.'

"The first four cases contain lower- and upper-case letters and punctuation, digits, and special numbers. (The chords for the letters correspond to the binary numbers from I to 26.)

"The fifth case is 'control case.' A particular chord (the same chord in each case) will always transfer subsequent input-chord interpretations to control case.

"In control case, one can 'backspace' through recent input, specify underlining for subsequent input, transfer to another case, visit another case for one character or one word, etc.

"One-handed typing with the handset is slower than two-handed typing with the standard keyboard. However, when the user works with one hand on the handset and one on the mouse, the coordinated interspersion of control characters and short literal strings from one hand with mouse-control actions from the other yields considerable advantage in speed and smoothness of operation."

He goes on to say that it takes about five hours of practice to be proficient enough to make it worthwile, after that practice makes perfect.

Hope this is interesting background. For 1968, Engelbart was ***way*** ahead of his time. I'm glad to see Scot's article; hopefully I'll be "chording" my next letter!

Tim Deagan
Austin, Tex.
revtim@well.sf.ca.us

P.S. The Internet connection is fabulous! Circuit Cellar is a real lifeline!

## Loading can be Easy

Hi. Welcome to internet. I just finished reading the "Firmware Furnace" article on using Turbo C for embedded programming. It really isn't as hard as Ed Nisley purports.

I have been using Turbo C and others for embedded projects on 8018*x* systems for about 10 years now. What I did was write a simple program that "loads" the .EXE file at an address that I specify, in much the same way as does DOS. Differing versions of C compilers and assemblers cause me no more problems than they cause DOS.

This loading process involves adding an offset address (beginning of ROM) to every item that is in the relocation table of the EXE file. The resulting relocated program is then converted to Intel hex and is ready for the EPROM burner. This, combined with a very simple startup routine that zeros RAM, sets up the stack and segment registers, and jumps to the C program, and I am up and running. This technique works with the "small" and "medium" models (i.e., 64K of data+stack, and as much code as you want).

The "loader" that I wrote is called EXEHEX and it is available free from "wuarchive.wustl.edu" by way of anonymous ftp. It is in the /msdos/filutl directory. I have made some bug fixes and improvements since that version was archived. I'll also send you the latest version to post on the Circuit Cellar BBS.

Chuck Harris
Laurel, Md.
chuck@eng.umd.edu

## Maxim Musings

I enjoyed Ed Nisley's "Firmware Furnace" in the August '93 (#37) issue of the ***Computer Applications*** *Journal*. As the Business Manager at Maxim for Microprocessor Supervisors, his article provided the best hands-on instruction I've seen for using the MAX691.

There are several points I think will help out your readers on this subject:

1) Mr. Nisley talked briefly on UL approval when using lithium backup batteries. Many of Maxim's microprocessor supervisory ICs have received UL registration, including the MAX691A. This means a user can hook up a lithium battery directly to the IC without the need for extra diodes or current-limiting resistors and still get UL approval. To obtain a list of the microproces-

# READER'S INK

sor supervisors which have UL registration and the UL file number (for independent verification), one may call our applications department at (408) 737-7600, ext. 4000.

2) If EPROMs or EEPROMs are used and battery backup isn't needed, Maxim has recently introduced the MAX792 microprocessor supervisor. It has all the functions of the MAX691A, including chip enable gating, but excluding backup switchover. Thus the MAX792 costs less than the MAX691A.

Maxim currently has over 50 microprocessor supervisory and voltage monitoring ICs with more being introduced every quarter. Our applications department can help users pick the best one for their design.

Thank you for the in-depth article and you can count me as a new subscriber.

Eric Munro
Maxim Integrated Products
Sunnyvale, Calif.

## Contacting Circuit Cellar

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

Mail: Letters to the Editor may be sent to: Editor, The Computer Applications Journal, 4 Park St., Vernon, CT 06066.

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

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

BBS: All of our editors and regular authors frequent the Circuit Cellar BBS and are available to answer questions. Call (203) 871-1988 with your modem (300–14.4k bps, 8N1).

Internet: Electronic mail may also be sent to our editors and regular authors via the Internet. To determine a particular person's Internet address, use their name as it appears in the masthead or by-line, insert a period between their first and last names, and append "@circellar.com" to the end. For example, to send Internet Email to Jeff Bachiochi, address it to jeff.bachiochi@circellar.com. For more information, send Email to info@circellar.com.

# NEW PRODUCT NEWS

Edited by Harv Weiner

## 386SX/486SLC SINGLE-BOARD COMPUTER

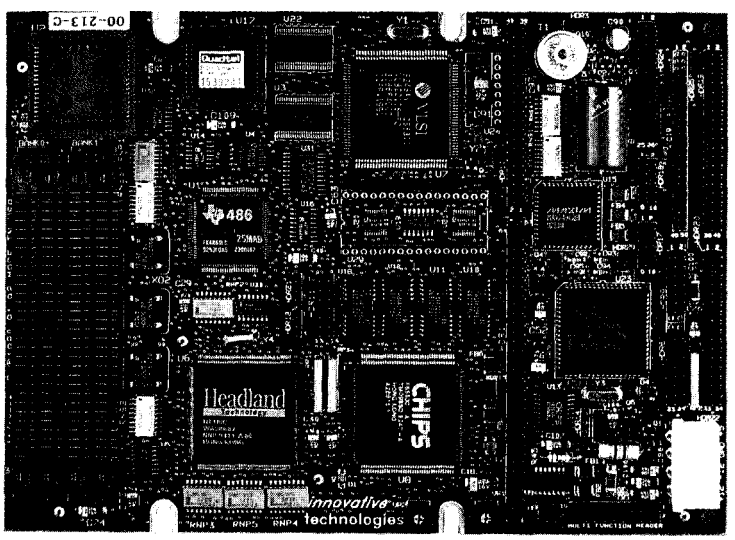
Innovative Technologies has announced an all-in-one, fully AT-compatible, single-board computer system. The **it/SLC** integrates all peripherals normally found in complete AT-compatible systems onto a single circuit board measuring just 5.75 by 8 inches.

Based on either a 386SX or 486SLC microprocessor running at speeds of up to 33 MHz, the it/SLC is completely compatible with the IBM AT standard at the hardware level. Equivalent software compatibility is ensured through incorporation of an industry-standard Phoenix/Quadtel system BIOS. Cache flushing on the 486SLC has been implemented in hardware to maximize performance. The it/SLC may be populated with up to 16M bytes of memory, and has an on-board socket for a 387SX-compatible math coprocessor.

A key feature of the it/SLC is its universal SVGA graphics controller. Based on Chips & Technologies' 65530, this graphics subsystem provides support for monochrome, STN color and TFT LCDs, EL displays, plasma panels, and noninterlaced analog CRTs with resolutions of up to 1,024 by 768 pixels. An on-board negative bias generator provides the voltage levels required by most monochrome LCDs, while the built-in power/signal sequencing improves LCD viewing characteristics and helps to extend panel life.

The on-board floppy disk controller supports all standard formats, including the newer 2.88-MB drives. An industry-standard IDE interface is implemented with connectors to allow direct cabling to all physical drive sizes. Alternatively, the on-board ROMdisk, which supports EPROMs, SRAM modules, and flash memory devices with capacities of up to 5 12 KB, can be designated as the boot device for totally diskless operation.

Other standard features of the it/SLC include two EIA-232D serial ports, a bidirectional parallel port, keyboard and PS/2 mouse ports, full ISA bus interface, and on-board speaker. The it/SLC can be operated from a +5V-only supply: the 386SX version with 4M bytes of memory typically dissipates less than five watts; the 486SLC version draws about six watts.




The 486SLC-25 version of the it/SLC sells for $696 in quantity. Pricing for other CPUs and quantities can be obtained from the factory.

Innovative Technologies
10301 Northwest Fwy., Ste. 514
P.O. Box 90086
Houston, TX 77092
(713) 683-0107
Fax: (714) 683-8478

**#500**

## HOME AUTOMATION VIDEO TAPE LIBRARY

Home Systems Inc. announces a series of seven training video tapes for the home automation industry. Each two-hour tape is accompanied by a 120-page reference book covering information related to the same topic as the tape. The first volume is titled **Power Line Wiring For Lights and Appliances** which provides a basic overview of home automation and teaches installation procedures for X-10 and other powerline transmission technologies.

The list of other announced titles includes Power Line Wiring for Attached Products, Coax & Low-voltage Wiring for Communication, Distributed Entertainment Systems, Environment & Energy Management Systems, Home Security Systems, and Automation System Controllers.

The tape library is created by an experienced television producer and is designed to be useful to contractors, educational institutions, and do-it-yourselfers. Each edition of the library has a suggested retail price of $54.90.

Home Systems, Inc. . P.O. Box 6236 . Edmond, OK 73083 . (405) **840-4751** . Fax: (405) 842-3419 **#501**

# NEW PRODUCT NEWS

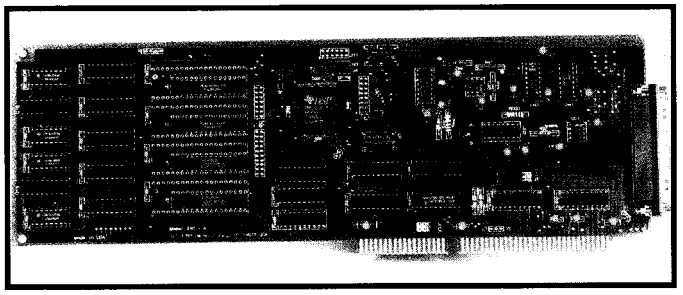
## DSP AND DATA ACQUISITION BOARD

A low-cost, PC-based, add-in board used for digital signal processing and data acquisition has been announced by Dalanco Spry. The Model 310A can be used for DSP education, telecommunications, audio, instrumentation, and control. The board features the Texas Instruments TMS320C31 floating-point DSP operating at 33 MHz and offers up to 33 MFLOPS performance.

The Model 310A provides data acquisition for four differential channels at 14-bit resolution with programmable gain and a maximum sampling rate of 150 kHz. One 12-bit, 300 kHz analog output is provided. The board features a PC-compatible interface with I/O-mapped dual-ported memory. It may be populated with zero- or one-wait state SRAM ranging in size from 32K to 512K words.

The Model 310A features high throughput and can be easily accommodated in multiple-board systems. PC-to-Model 310A data transfers may be as high as 3M bytes per second. Thus, in data acquisition and output applications, the maximum continuous throughput to and from disk is limited only by the capabilities of the host PC's disk system.

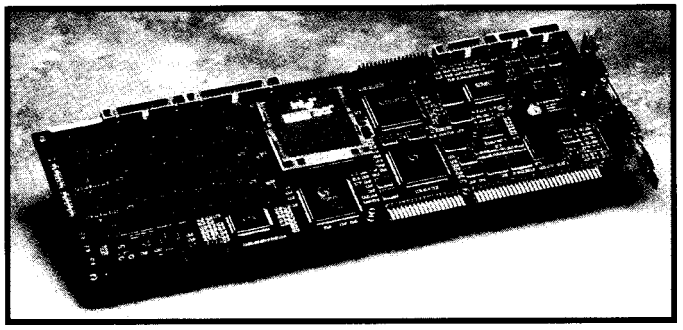
The Model 310A is bundled with the following software: assembler, debugger, signal and spectrum display, record and playback to/from disk. Also included is MODA, a program that manages multichannel data acquisition, simultaneous record and playback for stimulus/response applications, and provides advanced pretriggering options. The Model 310A is priced from $699 including software.



Dalanco Spry
89 Westland Ave.
Rochester, NY 14618
(716) 473-3610
Fax: (716) 271-8380

#502

## SINGLE-BOARD '486 COMPUTER



A fully AT compatible, passive backplane, single-board computer with a high-performance 32-bit local bus and linear addressing mode video interface is available from HM Systems. The HMS-486 board offers the performance and full functionality of a complete PC/AT system.

CRT monitors or flat panel displays (LCD, TFT, EL), floppy and hard disks, parallel and serial devices, mouse, and keyboard can all be directly connected to the HMS-486. The product is available in 486SX (25 or 33 MHz), 486DX (33 or 50 MHz), and 486DX2 (50 or 66 MHz) configurations with up to 64 megabytes of 36-bit-wide DRAM. A ZIF socket on the HMS-486 allows for future upgrades to the next generation of Pentium microprocessors. Options provide for up to 256K of secondary cache, slimline 2.88-MB floppy drive support, 16550 UART-based serial ports, 2.5-inch hard disk support, and a high-performance SCSI-2 daughter card.

The HMS-486 is especially suited for designs incorporating embedded PC/AT computer systems due to its single-slot ISA-bus requirements and low-power CMOS design. Designed with the Chips and Technologies 65535 video controller, the high-performance video system supports virtually any LCD or CRT monitor available. The HMS-486 board is designed to enable any standard ISA-bus passive backplane system to be upgraded to the latest 32-bit local bus video solution by simply replacing the existing CPU and video boards with the HMS-486.

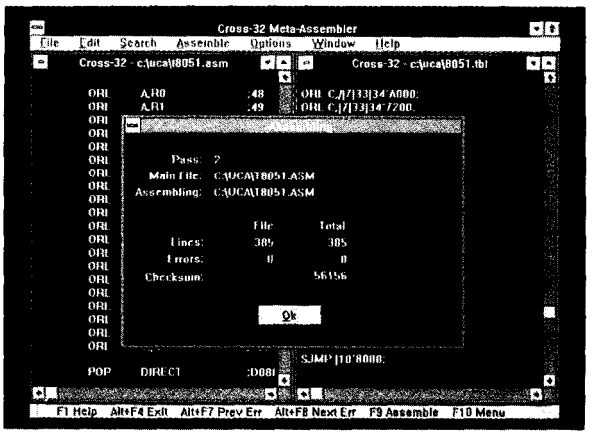
The HMS-486 ranges in quantity price from $395 for a 25-MHz 486SX noncached version to $1145 for a 66-MHz 486DX2 version including 256K secondary cache and 32-bit local bus video graphics with 512KB of memory.

HM Systems, Inc.
2192 Dupont Dr., Ste. 214
Irvine, CA 92715
(714) 955-2043 • Fax: (714) 955-1849

#503

# NEW PRODUCT NEWS

## UNIVERSAL CROSS-ASSEMBLER




Universal Cross-Assemblers is shipping Cross-32 **Meta-**Assembler for Windows, version 1 and Cross-32 **Meta-**Assembler for MS-DOS version 3. These table-driven macro cross-assemblers allow the user to compile assembly language programs for over 40 different microprocessors, microcontrollers, and digital signal processors. The tables use the manufacturer's original assembly language mnemonics, and full instructions are included so the user can create new tables for other processors.

The assembler supports logical and arithmetic operators, and integer constants identical in form and precedence to the ANSI C programming language, as well as several common assembler conventions. For ease of programming, both products provide a multiple document integrated development environment with on-line contextual help. Should an assembly error occur, the system will automatically display an error message and highlight the offending text.

Cross-32 reads the assembly language source file and a corresponding assembler instruction table and writes a list file and an absolute hexadecimal output files in the Intel, Motorola, or binary formats. It is, therefore, compatible with most EPROM programmers, EPROM emulators, and in-circuit emulators.

The Cross-32 is a case-insensitive, two-pass assembler with third pass if a phase error occurs. A binary checksum is displayed on the screen and the program features a program counter with a range from 0 to 4,294,967,295. The command line version assembles 5000 lines per minute of 68HC05 source code on a 20-MHz 386SX.

The Cross-32 Universal Cross-Assembler sells for **US$199.00** or **CDN$249.00**.

Universal Cross-Assemblers
9 Westminster Drive . Quispamsis, NB Canada . **E2E 2V4** • (506) 849-8952 • Fax: (506) 847-0681
Internet: **70730.3576@compuserve.com** **#505**

## ENVIRONMENTAL CONTROL SYSTEM FOR MACINTOSH

Remote Measurement Systems Inc. has announced the release of EnviroMac, an environmental monitoring and control system for the Macintosh. The EnviroMac package can be used with any Macintosh model making data acquisition and control easy and cost effective.

EnviroMac is ideally suited for applications such as energy management, industrial and university research, environmental monitoring and data collection, and product testing and small-scale process control in manufacturing.

The EnviroMac package turns a Macintosh computer into a "green machine," providing it with the capability to monitor such factors as temperature, air quality, and energy use. With advanced control capabilities, EnviroMac continuously evaluates external conditions and automatically issues commands to control electrical devices. EnviroMac can provide precise measurements and reliable control and can save money by reducing energy consumption.

The EnviroMac hardware serves as the interface between the Macintosh and sensors or instruments. The interface includes 16 analog inputs and 4 digital inputs for connection of sensors. A controller for the X-10 system of power line control modules plus 6 digital outputs enables the Macintosh to control up to 38 separate external devices.

Software offers the point-and-click ease of a standard Macintosh application allowing users to collect, display, record, and graph data obtained from sensors. Background operation is possible under System 7 or MultiFinder, so the Macintosh may be used for other tasks while monitoring continues.

The EnviroMac package sells for $899 including all the pieces necessary to begin collecting environmental measurements with Macintosh-hardware, software, temperature and light-level sensors, an X- 10 control module, cables for connecting to the Mac, and complete documentation.

Remote Measurement Systems, Inc.
2633 **Eastlake** Ave. East, Ste. 200
Seattle, WA **98102-3231**
(206) 328-2255 • Fax: (206) 328-I 787

**#506**

# NEW PRODUCT NEWS

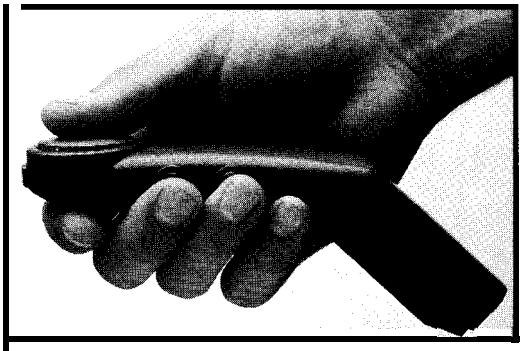
## WIRELESS COMPUTER CONTROL DEVICE

A remote cursor device designed to replace the venerable mouse has been announced by ArcanaTech. Called Imp, this sophisticated remote control allows cursor positioning as well as execution of user-programmable keyboard functions from as far as **15** feet away through infrared communication and a specially developed film that is sensitive to the touch. Common mouse functions like pointing, clicking, and dragging are all performed using Imp's unique control disc technology. Keyboard functions are assigned to Imp's auxiliary buttons through an easy-to-use software control panel.

Since Imp is hand-held, it does not require a dedicated flat surface or cord connecting it to the computer like a mouse or trackball. This gives the user freedom and flexibility.

Imp consists of an ergonomically designed, lightweight, battery-powered, wireless transmitter and a compact receiver which connects to a host computer's RS-232 serial port. The transmitter contains the control disc, which is used to control cursor motion, clicking, double-clicking, and dragging. The unit also has four user-programmable buttons. Cursor speed and direction are governed by light pressure on the control disc; there are no moving parts to collect dust or fail. The receiver is powered by the host computer and contains indicator lamps that reflect communication activity and transmitter battery status.

The suggested retail price of Imp is $199 and includes the remote transmitter, receiver and cable with DB-9 connector, DB-9-to-DB-25 adapter, software, 4 AAA batteries, and a user guide.



ArcanaTech
120 South Whitfield St.
Pittsburgh, PA 15206
(412) **441-6611**
Fax: (412) 361-5103

**#507**

# NEW PRODUCT NEWS

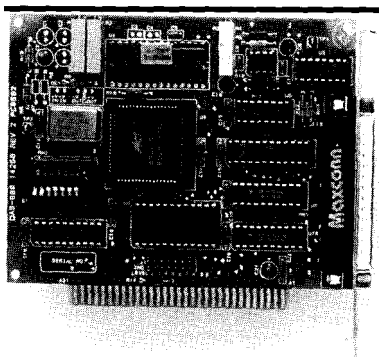
## HIGH-SPEED DATA ACQUISITION BOARDS

The **DAS-800** Series analog and digital I/O boards from Keithley MetraByte combine acquisition speeds of up to 40k samples per second with performance features typically available on more expensive boards.

The DAS-800 Series includes three boards. The **DAS-800** has eight single-ended analog inputs with a fixed input range of ±5 V. The DAS-801 and DAS-802 have eight analog inputs which can be individually switch-selected for single-ended or differential operation. The DAS-80 1 offers nine low-level unipolar and bipolar software-programmable analog input ranges. The DAS-802 has nine unipolar and bipolar software-programmable ranges for high-level inputs. Each board includes three digital inputs and four digital outputs.

A four-location first-in first-out (FIFO) memory helps maintain an acquisition rate of up to 40k samples per second by overcoming variations in the personal computer's interrupt response time. Data can be stored in the FIFO while the board waits for the computer to respond to the interrupt and retrieve the data. This ensures accurate, jitter-free sample timing. Automatic channel incrementing and generation of an interrupt after a conversion ensures that samples are taken and transferred to the computer as fast as possible. Users have four software options for operating the DAS-800 Series boards.

The DAS-800 board sells for $349. The DAS-801 and DAS-802 sell for $449 each. The ASO-800 Advanced Software Option sells for $99.




Keithley Instruments
Data Acquisition Division
440 **Myles** Standish Blvd.
**Taunton,** MA 02780
(508) 880-3000
Fax: (508) 880-0179

**#508**

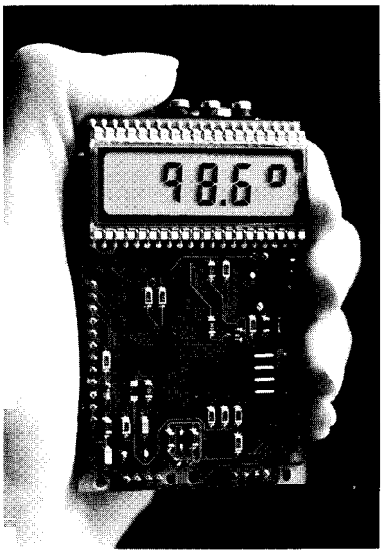
# NEW PRODUCT NEWS

## COMPACT DATA LOGGER

The **Tattletale 5F-LCD** provides a new level of portable, unattended data gathering ability for a wide variety of applications. By combining a four-character LCD display with their Tattletale 5F, Onset has produced a compact 2.1" by 3.1" package that not only logs data and provides control signals, but also offers an instant readout of activity. The Tattletale 5F-LCD offers eight channels of analog input, each with 12-bit resolution, and 480K of data storage. Application programs developed using either an IBM PC or a Macintosh may be stored in the 5F-LCD's ROM without using special programmers.

The 5F-LCD operates from a standard 9-V battery. Users may quickly develop sophisticated data acquisition and control applications with Onset's TTBASIC or tokenized TxBASIC language dialects with added commands for data acquisition tasks. For example, the SLEEP command provides precise timing as well as minimum-power operation. The BURST command reads and stores data from multiple channels. The STORE command writes to the data file and takes care of memory addressing and management. These languages have been proven in over eight years of use with the full Tattletale line.

The Tattletale 5F-LCD Data Logger sells for $495 in small quantities. A Starter Kit, which includes everything necessary to develop complete application programs, sells for $95. A durable plastic case ($35) and a three-button keypad ($25) are available options that can be used for portable data gathering applications.




Onset Computer Corp.
536 MacArthur Blvd., P.O. Box 3450
Pocasset, MA 02559
(508) 563-9000
Fax: (508) 563-9477

**#509**

# FEATURES



# Put a CEBus Power Line Interface in Your Next Design

You've been reading about CEBus for years now, and it's never been much more than words on paper. Warm up your soldering iron and get out your software tools; it's time to put your hands on some hardware.

**FEATURE ARTICLE**


**Christopher Yasko**


The CEBus specification was formally released as interim standard IS-60 by the Electronic Industries Association (EIA) in October, 1992. Since that time, several manufacturers have changed their "wait and see" attitude to an aggressive product development schedule. The first CEBus applications in home automation, utility metering, access control, and telemetry have already reached the marketplace. Yes, CEBus is for real!

The CEBus standard is a description for an open architecture protocol using various communications media. (For more details, see "CEBus Update: How is the Health of EIA's Baby," June/July 1990 and "CEBus Update: More Physical Details Available," June/July 199 1.) The consumer can choose from a wide variety of physical media for their CEBus system including power line, infrared, radio frequency, coaxial cable, and twisted pair.

I'm going to limit my discussion here to CEBus communications over the most ubiquitous medium: the power line. Two ASICs are available from Intellon that can be used as building blocks for CEBus power line products. CELinx PL is a Spread Spectrum Carrier transceiver implementing 100-kHz–to–400-kHz signaling that can carry 10,000 symbols per second. CEThinx is a CEBus Network Controller designed to mate with the power line transceiver and handles the time-critical CEBus Data Link Layer.

## CEBus PROTOCOL MODEL

The CEBus protocol is structured after the ISO/OSI seven-layer network

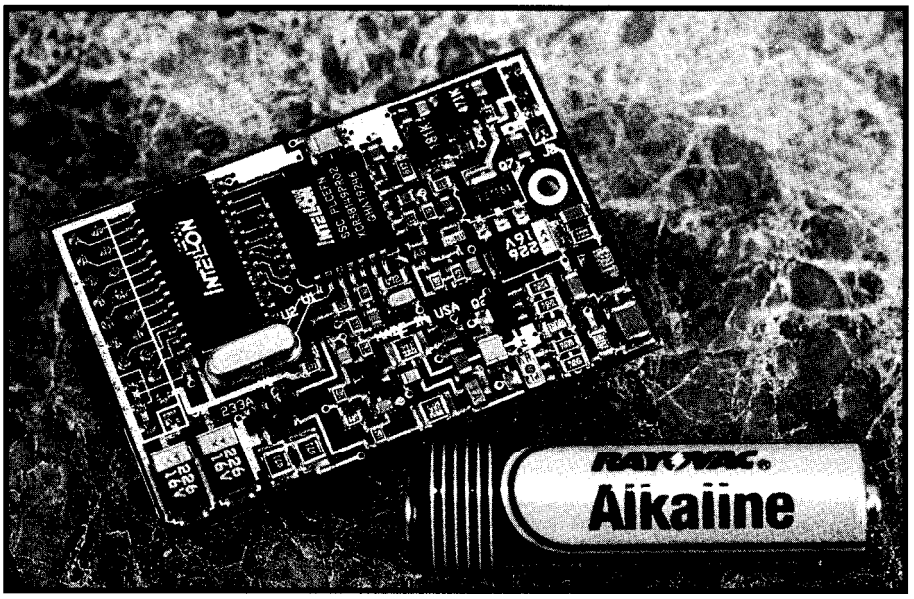



Photo 1-A complete *CEBus* power *line interface (minus the* coupling *transformer)* ***takes up*** *very* little space inside ***a*** *product*

model (a few of the layers are combined]. The ***Physical Layer*** performs the spread spectrum symbol encoding and decoding, receiver correlation, tracking, and error detection. The *Data* ***Link Layer*** (DLL) implements address assignments, channel access arbitration, collision avoidance, and packet acknowledgment services. The ***Network Layer*** provides services for multiple media routing, packet segmentation, and flow control. The ***Application Layer*** consists of the high-level language command syntax, called the ***Common Application Language*** (CAL), from which a product's control messages are created.

CEBus does not specify how these protocol layers manifest themselves. Tradeoffs between hardware and software are left to the designer. Practical considerations will control the choice of microprocessor hardware when building a simple light switch versus designing a complex audio/video entertainment system. Typically, designers implement as much of the protocol layers as possible in embedded software to reduce recurring manufacturing costs.

## PHYSICAL LAYER: CELinx

One component that can be used to implement the CEBus Physical Layer for power-line-based transmission systems is a chip called the CELinx PL. CELinx performs the spread spectrum signal generation for the CSMA preamble-which uses amplitude shift key modulation-and the body of the packet-which uses phase shift keyed modulation. It maintains waveform tables to build the packet symbols. Signal reception and tracking are accomplished by a matched transversal filter that is continuously searching for waveform correlation. The CELinx includes complete CRC generation and detection, plus an end-of-packet indication. This chip is available in a 28-pin PLCC package and costs under $5 in OEM quantities.

The CELinx interface is composed of ten I/O lines to a host microprocessor and includes three data lines, three input control lines, and four output control lines. The three data lines are required for synchronous serial data transfers: data in (DI), data out (DO), and data clock (DCLK). Most firmware engineers will recognize this nomenclature as Motorola SPI, but bit banging can also work.

There are three input control lines that go into the transceiver. Chip enable (CE) allows for shared use of the serial peripheral data bus. The operation of the half-duplex transceiver onboard the chip is controlled by a transmit/receive mode (TX/RX) pin. External interrupt sources are removed when serviced with interrupt clear (CLR). Note the two additional input lines for receive sensitivity (TO and T1) are bonded as CELinx pins, but are typically hardwired.

The four output control lines of the CELinx PL power line transceiver are meant to be interrupt inputs to the host processor. Carrier Detect (CD) indicates the correlation of a spread spectrum chirp and is used in collision detection. When the transceiver data buffer needs attention, Data Available (DA) is asserted. Each CEBus packet ends with a CELinx hardware-generated CRC code and the ***Packet Terminate*** (PTERM) condition. These three output lines are logically "ORed" for convenience to a single pin (INT) and connect to a single external interrupt. A block diagram of this device is shown in Figure 1.

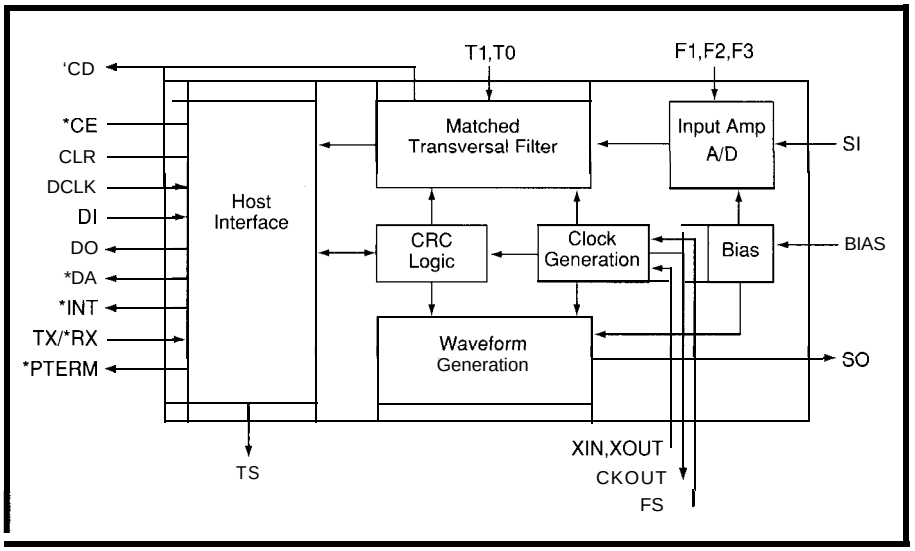



Figure 1—*The* CELinx chip *handles all the physical layer details of a CEBus power line interface.*

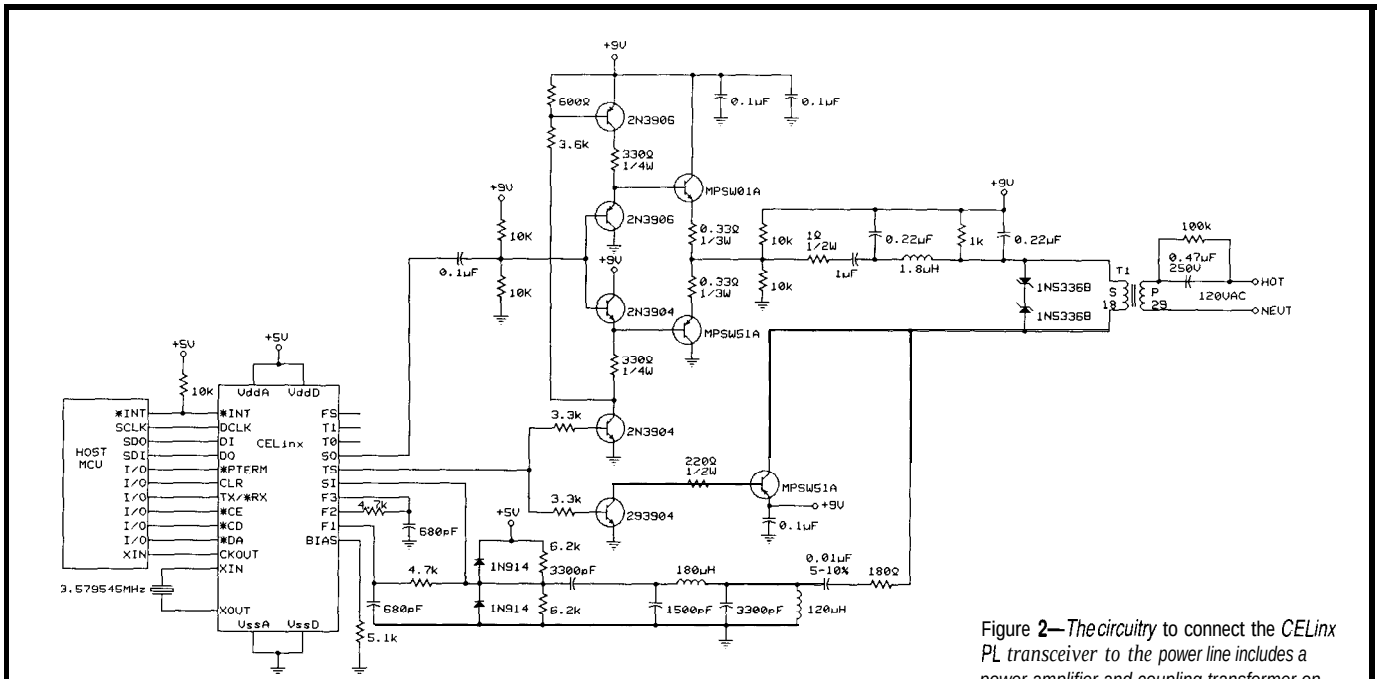



Figure 2—*The circuitry* to connect the *CELinx PL transceiver to the power line includes a power amplifier and coupling transformer on the transmit side, and a bandpass filter and transient/surge* protection *on the receive side.*

The external circuitry required to connect the CELinx PL transceiver to the power line medium is straightforward. A power amplifier is required to drive 4 volts peak-to-peak into the low impedance (10 ohms) of a typical 120-VAC line. Power lines can present inductive or capacitive loads to the amplifier, so an unconditionally stable voltage follower design is recommended. A torroidal signal coupling transformer provides linear transfer of the 100-kHz–to–400-kHz CEBus Spread Spectrum Carrier signal and provides adequate isolation. On the receive signal, a five-pole, passive bandpass filter rejects out-of-band noise with minimal distortion. Voltage transient and surge suppression must be included in the form of zener diodes or MOVs to protect the sensitive low-voltage circuitry from off-line spikes. A sample medium-coupling circuit shown in Figure 2 is taken from Intellon's CETalx power line medium interface card.

## DATA LINK LAYER: CEThinx

The Data Link Layer (DLL) of the CEBus protocol has the real-time processing burden of the IS-60 specification. The DLL is responsible for channel access arbitration, collision resolution, conversion of data bytes to CEBus unit symbols (USTs), address decoding, duplicate packet rejection, and transmission retries on error. Getting all of this to work correctly can get kind of sticky.. .especially since many things can happen in the DLL in just a few dozen microseconds.

Although some may find complicated real-time requirements fun to code, real product deadlines for a completely debugged and tested DLL can be elusive. The medium access rules for the CEBus channel are a function of quiet time, packet priority, queuing state, and random delays. Simuitaneous packet collisions detected by the physical layer transceiver must be intelligently recognized. The *pulse length symbol encoding* scheme required for CEBus must be done in real time as a background task. While a CEBus packet is being received, the DLL must decode it for a match in the destination address. Additionally, error-free packets require an immediate acknowledgment (ACK), where duplicate packets or those with a bad CRC must be thrown in the bit bucket. The real-time processing resources required for the DLL can

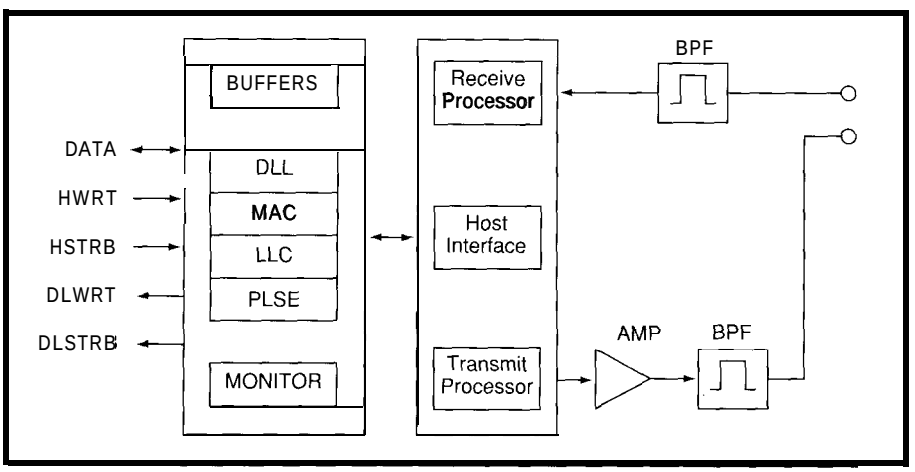



Figure 3—*The CEThinx* Network *Controller IC* is a complete CEBus Data Link *Layer* solution that interfaces to *your* host *processor.*

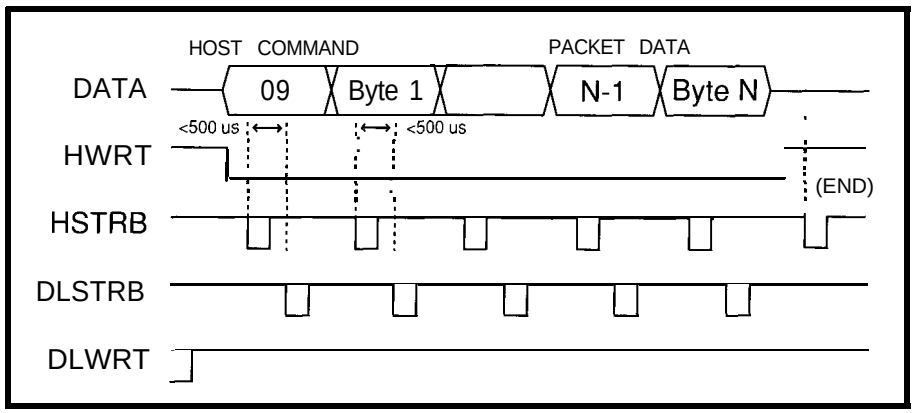



**Figure 4**—*Transmission of data using the CEThinx interface requires a host command followed by the packet data.*

cramp a microprocessor's ability to perform application functions.

The CEThinx Network Controller IC is a complete CEBus Data Link Layer solution packaged as a custom ASIC. It is designed to be paired with the CELinx transceiver IC as a matched set. The CEThinx buffers entire packets, automatically arbitrates channel access, resolves collisions, decodes addresses on the fly, and generates acknowledgment packets as required. It is available in a 28-pin SOIC package and costs less than $5 in OEM quantities.

The CEThinx is a fully integrated package designed to communicate with a host processor. Transmit and receive data packets are independently buffered for asynchronous access. The CEBus address parameters for System (House), MAC (Unit), and Group codes are stored in on-board chip memory. Since all the CEBus Data Link Layer services are provided in CEThinx, the application code is free from the real-time CEBus channel constraints. The chip saves development time, reduces the host processor bandwidth and lessens IRQ response time requirements.

The CEThinx interfaces to a host microprocessor with a total of twelve I/O lines. The chip acts as a slave peripheral to an application host processor. The processor can be as powerful or as small as the non-CEBus part of the product demands. Eight bidirectional data lines, two input control lines, and two output control lines are required from the host processor to interface with CEThinx.

The 8-bit data bus of CEThinx is designed to connect to a dedicated parallel host port or can be memory mapped to an address bus with external glue logic. CEBus data packets are synchronously transferred between the CEThinx and the host application processor. Individual strobe lines are used to write data bytes on the bus and acknowledge a read from the bus. Separate control write lines indicate bus direction. The control write line also signals the start or end of a packet message across the S-bit data bus.

The host processor acts as a master on the data bus. Host Write (HWrt) is an active-low input to CEThinx. When writing a message to CEThinx, HWrt is asserted to indicate that a transfer is pending on the data bus. Host Strobe (HStrb) is a falling-edge-triggered input to the CEThinx indicating a new byte is available from the host. When reading a byte from the CEThinx, HWrt is not asserted and HStrb is used to acknowledge a data byte has been read from the bus.

The CEThinx is designed as a slave peripheral to the application code. Data Link Strobe (DLStrb) should be connected to an active interrupt or latched flag at the application processor. DLStrb is a falling-edge output from CEThinx indicating that a new byte is available on the data bus. Data Link Write (DWrt) is an active-low output from CEThinx. Data Link Write and Data Link Strobe are used together to get the attention of the host processor when needed. A functional block diagram of CEThinx is shown in Figure 3 and timing diagram of data transfers between the host and CEThinx are shown in Figures 4 and 5.

CEThinx has its CEBus address configured by a set of host commands. The CEBus address space is composed of three 16-bit numbers: a System *Address,* a *MAC Address,* and a *Group Address.* Each CEBus device is given a unique combination of MAC and System Address to determine its 32-bit identity on the network. Additionally, multiple CEBus devices can be simultaneously controlled by a shared Group and System Address combination. The CEBus protocol allows both individual devices and logical groups to be controlled transparently.

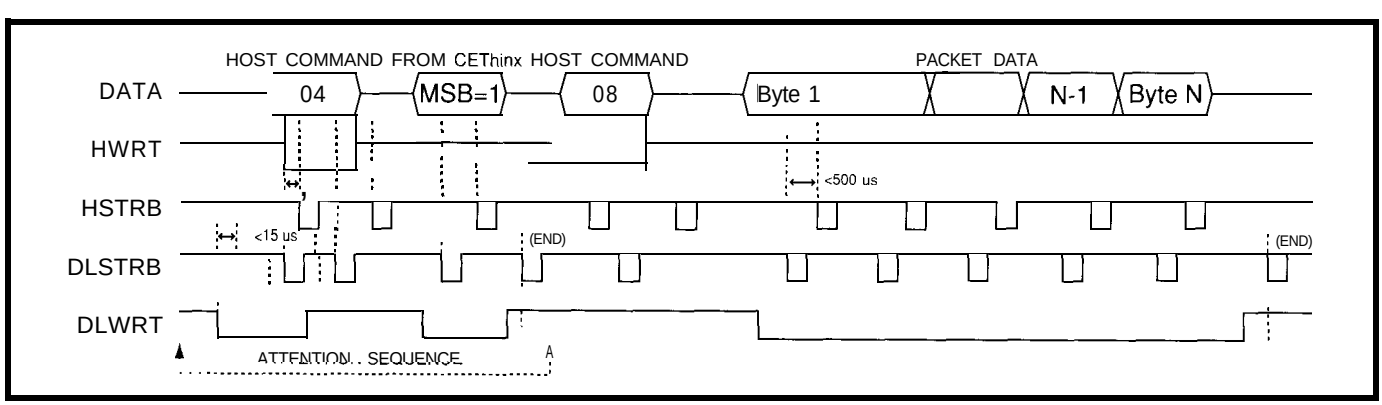



Figure **5**—*Reception of packet data through the CEThinx interface requires an attention sequence followed by a host command and the actual data packet.*

Listing 1—*Example C source code to initialize CEThinx, transmit a CEBus packet, and receive a CEBus response packet under 68HC11 interrupt control.*

```
#define CEBCTRL 0x1000 /* 68HC11 PORT A, I/O and timer          */
#define DATA    0x1003 /* 68HC11 PORT C, 8-bit data             */

#define DLSTR   B0x01  /* PORT A, bit 0 mask, input capture #3 */
#define DLWRT   0x02   /* PORT A, bit 1 mask, input            */
#define HSTRB   0x10   /* PORT A, bit 4 mask, output           */
//define HWRT   0x20   /* PORT A, bit 5 mask, output           */

#include <68HC11E9.H>  /* register/control bit assignments     */

unsigned char

   AddrConfig[7] =
      { 0x40, 0x04, 0xE0, 0x02, 0x81, 0x03, 0x00 },
   TempRequest[15] =
      { 0x09, 0x10, 0x81, 0x03, 0x00, 0x02, 0x81,
      0x03, 0x00, 0x70, 0xE7, 0x40, 0x01, 0x43, 0x43 },
   Rx Buffer[41],
   attention-flags;

void init_CEThinx( void )
{
short i =0;

   DDRA = (HSTRB|HWRT); /* Port A, DDR, host output bits       */
   DDRC = 0xFF;         /* Port C, DDR, set to outputs         */

   CEBCTRL &= ~HWRT;    /* assert Host Write low               */
   DATA = 0x03;         /* Layer Mgmt Write (LR) command       */
   CEBCTRL &= ~HSTRB;   /* toggle Host Strobe                  */
   CEBCTRL |= HSTRB;
   while (CEBCTRL & DLSTRB);     /* wait for Data Link Strobe  */

   for (i=0; i<7; ++i)           /* write config message       */
      {
      DATA = AddrConfig[i];      /* default CEBus address, etc. */
      CEBCTRL &= ~HSTRB;         /* toggle Host Strobe         */
      CEBCTRL |= HSTRB;
      while (CEBCTRL & DLSTRB); /* wait for Data Link Strobe   */


   CEBCTRL |= HWRT:       * end of message                     */
   CEBCTRL &= -HSTRB;
   CEBCTRL |= HSTRB;

   TCTL2 = 0x02;          * capture #3 falling edge            */
   TMSK1 |= 0x01;         *  enable  capture #3  interrupt     */
   DDRC = 0x00;           * Port C DDR, set to inputs          */


void transmit_CEBus_packet( void )
{
   TMSK1 &= 0xFE;         * disable capture #3 interrupt       */
   DDRC = 0xFF;           * Port C DDR, set to output          */

   CEBCTRL &= ~HWRT;      /* assert Host Write low             */
   DATA = 0x09;           /* Packet Transmit (PT) command      */
   CEBCTRL &= -HSTRB;     /* toggle Host Strobe                */
   CEBCTRL |= HSTRB;
   while (CEBCTRL & DLSTRB): /* wait for Data Link Strobe      */

   for ( i=0; i<15; ++i )    /* write CEBus packet info        */
      {
      DATA = TempRequest[i];/* CAL Context, Object, Method,IV */
      CEBCTRL &= ~HSTRB;     /* toggle Host Strobe             */
```

*(continued)*

I should mention that CEThinx also has a special Monitor Mode of operation. The host processor can set configuration flags in CEThinx to observe and report all packets on the medium regardless of CEBus address. The Monitor Mode also stamps each packet with a 16-bit free-running time code, giving two-microsecond resolution for real-time analysis of CEBus traffic. CEBus protocol analyzers are easily implemented using this feature.

## SOFTWARE INTERFACE TO CEThinx

Excerpts of code from an HVAC application are shown in Listing 1. It is included here to show you how to use the CEThinx chip. This intelligent HVAC controller uses a Motorola 68HC11 to control a 12-digit keypad, an A/D temperature sensor, and a multifunction LCD display. The CEBus messages will be handled as a peripheral port to the host 68HC11 processor. In the user's application code, the HVAC controller would regularly poll remote temperature and occupancy sensors throughout different rooms in the home and determine if any action should be taken.

The example code focuses on how to initialize CEThinx, transmit a CEBus request packet, and receive the response packet. The initial configuration of the CEThinx is accomplished by the host sending a Reset Command followed by a Layer Management (LM) Write command. The LM Write sets the CEBus device address for System, MAC, and Group codes. The LM Write also sets the number of packet retries allowed before a failure is reported on the medium. Usually, the CEBus packet will get across the channel on the first try, but if the transmission is unsuccessful the CEThinx is smart enough to try again. The network settings of our HVAC controller are: System Address = 0003, MAC Address = 8 102, the Group Address = EO 04.

Transmitting a packet uses very little code because CEThinx handles all of the CEBus channel requirements. The host processor signals the CE-Thinx with a Packet Transmit Command followed by the actual data bytes that make up the CEBus packet. The

Listing 1-continued

```
        CEBCTRL |= HSTRB;
        while (CEBCTRL & DLSTRB);/* wait for Data Link Strobe    */


    CEBCTRL |= HWRT;      /* end of message                      */
    CEBCTRL &= -HSTRB;
    CEBCTRL |= HSTRB:

    TMSK1 |= 0x01;        /* enable capture #3 interrupt         */
    DDRC = 0x00;          /* Port C DDR. set to inputs           */
}

int receive_CEBus_packet( void )
{
short i=0;

    if (!(attention_flags & 0x80))/* test IRQ receive flag       */
        return -1;        /* if not set, return error            */

    TMSK1 &= 0xFE;        /* disable capture #3 interrupt        */
    DATA = 0x08;          /* Packet Receive (PR) command         */
    CEBCTRL &= -HSTRB;    /* toggle Host Strobe */
    CEBCTRL |= HSTRB;
    while (CEBCTRL & DLSTRB);        /* wait for Data Link Strobe*/

    while (!(CEBCTRL & DLWRT))       /* wait for DL Write to end */

        while ( CEBCTRL & DLSTRB );  /* wait for DL Strobe       */
        RxBuffer[i] = DATA:          /* buffer packet data info */
        CEBCTRL &= -HSTRB;           /* toggle Host Strobe       */
        CEBCTRL |= HSTRB;
        if (++i > 40 ) break:        /* inc index, ?out of range */


    TMSK1 |= 0x01;                   /* enable capture #3 int.   */
    return 0;


INTERRUPT attention_CEThinx(void)  /* Capture vector: 0xFFEA-EB*/

    TFLG1 = 0x01;         /* clear capture flag, #IC3F           */
    if (CEBCTRL & DLWRT) /* ?DL Write asserted low               */
        return:           /* exit if set, not atten. seq.        */

    TMSK1 &= 0xFE;        /* disable capture #3 interrupt        */
    DDRC = 0xFF;          /* Port C DDR, set to output           */
    CEBCTRL &= ~HWRT;     /* assert Host Write low               */
    DATA = 0x04;          /* Interface Read (IR) command         */
    CEBCTRL &= ~HSTRB;    /* toggle Host Strobe                  */
    CEBCTRL |= HSTRB;
    while (CEBCTRL & DLSTRB);/* wait for Data Link Strobe        */

    CEBCTRL |= HWRT:      /* turn off Host Wr., set high         */
    DDRC = 0x00:          /* Port C DDR, set to inputs           */
    CEBCTRL &= -HSTRB:    /* toggle Host Strobe                  */
    CEBCTRL |= HSTRB;

    while( CEBCTRL &( DLWRT | DLSTRB ));
                          /* wait for DL Write and DL Strobe*/

    attention-flags = DATA;    /* store CEThinx flags            */
    CEBCTRL &= -HSTRB:         /* toggle Host Strobe             */
    CEBCTRL |= HSTRB;
    TMSK1 |= 0x01:             /* enable capture #3 interrupt    */
}
```

CEThinx handles all Unit Symbol conversions, channel access rules, collision avoidance backoffs, and retries if the transmission is not successful the first time. The CEThinx signals the completion of transmission with an Attention Command back to the host processor. The host gets the transmission status by reading TX_Done and TX-Status flags,and,if desired, can then transmit the next CEBus packet.

Receiving a packet with the CEThinx is done behind the scenes from the application code's point of view. CEThinx buffers all packets from the CEBus network. The CE-Thinx checks for a good 16-bit CRC indication at the end of the packet and automatically decodes the destination address in the header of each packet on the fly. If required, the CEThinx generates the appropriate immediate acknowledgment packet (IACK) within the required 200-µs response window. If there is a match in the destination address and the CRC is good, CEThinx notifies the host microprocessor that a packet has been received.

The slave CEThinx notifies the host with an Attention Sequence. In our HVAC controller application, we make use of the 68HC11 input capture as an external interrupt resource for the Attention Sequence. The Data Link Write line from CEThinx connected to the host processor is asserted only when the CEThinx has something to say. By using the Port A Input Capture, the host background routine handling the keypad and LCD functions is only interrupted when required. This eliminates software polling of the CEThinx as a peripheral and this saves precious application processor time.

When an Attention Command is received, the host processor checks the CEThinx flags with an Interface Read Command. One flag indicates if a packet has been received and is ready to be transferred. If true, the host initiates the Packet Receive Command which tells CEThinx to put the received CEBus packet data on the bus. The data is strobed on to the bus with DLStrb from CEThinx, and is read by a HStrb strobe from the host processor.

A unique condition where DLWrt is high and DLStrb is asserted low notifies the host of the end of packet data.

The Network Layer portion of this CEBus application is moot. Since all temperature and occupancy sensors we are using connect directly to the power line, routing to other media does not apply. Although reserved bytes exist in every CEBus packet to determine routing, this application will use fixed codes.

The Application Layer handles the construction and interpretation of the CAL packets required for the HVAC controller. The transmitted packets to request remote information are prebuilt and stored in ROM. Parsing the CAL response packets to determine if temperature values are in range can be made as simple as a lookup table. The Application Layer Software would also include the required routines to support the keypad and LCD display.

## TIME TO PACKET IN

CEBus is an open standard for home automation released by the EIA. The engineering community has embraced the concept of control networks based on intelligent nodes and more products based on this concept are being announced every day. By conforming to the CEBus standard, both complementary and competitive consumer products have a common language that is capable of being used to communicate information with each other. Information for the automation of lighting, audio, security, HVAC, and utility metering can be shared throughout the home.

The power line medium is in every home, which creates a market for both new and retrofit CEBus designs. Hardware ASICs are available today to accelerate development of CEBus power line products. The CELinx Power Line Transceiver delivers the spread spectrum symbols across the most harsh physical environment. The CEThinx Network Controller IC is a packaged DLL relieving the host from handling all the CEBus timing, packet buffering, and error checking. Together, these new ICs are reaching the marketplace in consumer products that will save energy and create a home automation fantasyland.

Christopher Yasko received a *B.S.E.E. with honors in 1987 from Worcester Polytechnic Institue. He is an Application Engineer at Intellon Corporation where he has been working on CEBus protocol software and CEBus product development systems.*

## SOURCE

The CEThinx Network Controller IC is the first in a family of intelligent embedded controller products from Intellon Corporation. The CEBus Data Link Layer is a common element to all CEBus product architectures. The CEThinx Network Controller IC directly connects to either the Power Line or 915-MHz RF Spread Spectrum Carrier CEBus transceivers. The product application does not require knowledge of the specific media it is operating on.

Other versions of the CEThinx family will be available early in 1994 to perform simple sensor and actuator functions. Devices that require a simple CEBus Application Layer and I/O functions will be bundled in a single embedded controller IC. These CEThinx devices will be capable of controlling relays, triacs, setting digital and analog levels, and sensing various digital and analog sources.

Intellon Corp.
5100 W. Silver Springs Blvd.
Ocala, FL 34482
(904) 237-7416
Fax: (904) 237-7616

## IRS

401 Very Useful
402 Moderately Useful
403 Not Useful

# Home Temperature Control Basics

FEATURE ARTICLE

**Anthony Segredo**

For some, warming up the living room means warming the whole first floor of the house. For the energy conscious, temperature control is heating just the rooms that need it. Read on for the fundamentals.

Have you ever wished you could adjust the temperature in each room of a building as easily as you can adjust the lighting level? Wouldn't you like a house in which you didn't have to physically adjust dampers whenever you switched from heating to cooling? Does a sunny room always have to be warmer than its shady neighbor? Have you looked for design information on automatic temperature control and only found data on setback thermostats? If the answer is "yes" to any of these questions, then you've come to the right place. I've wondered about these things too and I've also been frustrated by a lack of answers. This article was written to help you create those systems and products. After all, you wouldn't be an engineer if you didn't prefer to "roll your own" rather than take what the market offers.

## WARMING UP TO THE TOPIC

So let's do it! We need to take a look at how heat flows into a physical system and through its subsystems. Then we'll see what can be done to create and channel those flows. Finally, when you understand what is being controlled and how it is manipulated, we can look at control strategies.

The quantity of heat in a body, Q, is proportional to its volume, *V*, and temperature, T, by a constant known as the *specific heat capacity*. The following illustrates this principle:

$$Q=CVT \qquad (1)$$

Heat flows essentially by three means: conduction, convection, and radiation. Conduction is the transport of heat by physical contact of two bodies at different temperatures. The conduction of heat through a contact area, *A*, of thickness *L*, between two bodies at different temperatures is a linear process characterized by a constant called conductivity, *K*.

$$\frac{dQ}{dT} = K\left(\frac{A}{L}\right)[T_{higher} - T_{lower}] \qquad (2)$$

The inverse of K(A/L) is called the *resistance* (*R*), and is normally quoted in the building trades. This is the "R-value" that is seen on insulation and exterior windows. Combining equations (1) and (2) results in:

$$\frac{dT_{inside}}{dt} = \left(\frac{KA}{CVL}\right)[T_{outside} - T_{inside}] \qquad (3)$$

The solution to this form of linear differential equation is well known; in fact, it is the same as the equation describing the rate of change of current in an induction coil, and is represented by an exponential approach to equilibrium characterized by a time constant given by:

$$\frac{CL^2}{K}$$

Therefore, the time scale of temperature changes due to conduction depends only on the material and thickness. Table 1 gives time constants in units of seconds per square centimeter for some common materials. Note that the time scales range from seconds to minutes.

## THAT'S COOL

Now that we understand how heat flows, let's look at how we might make it flow the way we want. The traditional method of building temperature control is the so-called *bang-bang* method: A furnace or air conditioner is turned on at a preset temperature, runs full blast until a second preset temperature is reached, then shuts off.

Often, the interior environment of a building will be significantly hotter or colder than the exterior environment due to time lags and internal heat generation. Significant energy savings can be achieved along with

needed ventilation by drawing in outside air.

Solar heat loads can be blocked by the use of drapes and shutters. Vertical hanging Venetian blinds, with their single control track and modern appeal, seem an ideal choice for automation.

Switching flows in forced air ducts can be accomplished by motorized dampers. These can be used to balance the heat flows between rooms or floors in a building. Since buildings tend to have many more rooms than floors, cost considerations make main floor duct dampers seem more practical than individual room controls.

Now that we understand how heat flows and the devices that can control it, let's consider how to put them together to write a control program.

Since you know that temperature in a building is a first-order system, a PID control system could be used to hold temperature to a preset value. The linearly variable heating/cooling output could be accomplished by controlling mixing valves in a hydronic heating or chilled water cooling system. A variable-speed blower motor would be requred to achieve this level of control in a forced air system, either by modifying the air handler's existing motor, or disabling it and adding an external blower. But, except for some exotic materials processing systems, this sort of control, with temperatures held to a fraction of a degree, is not needed. Most people are comfortable across a range of several degrees, permitting the use of the more efficient bang-bang system.

It would seem that a more promising application of home temperature control would be in the realm of zone or room control. Consider as a concrete example any two-story, three-bedroom home with the bedrooms on the upper floor. The house uses a forced air heating system with an integrated central air conditioner. There is no reason to heat the kitchen when the oven is on just because the family room is cold. Similarly, at night the lower story doesn't need cooling, while the bedrooms do. An ideal system would have individual thermostats in every room. A minimal system would have upstairs and downstairs thermostats. Our problem now is how to adjust air flows between zones by altering the duct restrictions (i.e., by moving dampers). If only one duct is open, pressures in the air handler might become unacceptably high, resulting in noise and air leaks. In a system with a basement, an extra duct in the basement that could be opened for pressure relief is one option. Otherwise, either the ducts must be sized for a lower pressure drop, or the blower speed must be controllable.

From Table 1, you can see that the time constraints on the control program are rather loose. In fact, a DOS TSR or a Windows background task running the following pseudocode would be fast enough for our purposes.

```
FOR All Zones
   IF Zone wants heat
      Open Damper
   ELSE
      Close Damper
   ENDIF
ENDFOR
IF Any Damper is Open
   Energize Furnace
ELSE
   Deenergize Furnace
ENDIF
```

Scanning every few seconds to update damper positions should be sufficient. There is no need for optimized assembly code. Any high-level language, even a BASIC interpreter, can run fast enough to keep up with this kind of real time. The Energize Furnace procedure can include a test of outside air temperature and humidity in order to save energy in a fan-only mode that opens an intake damper from outside.

| Material | Time Constant (s/cm²) |
|---|---|
| Aluminum | 1.2 |
| Iron | 8.5 |
| Porcelain | 250 |
| Wood (fir) | 869 |
| Marble | 2280 |
| Granite | 115 |
| Quartz | 21.6 |
| Glass | 167 |
| Cement | 804 |

Table I-Conductive *time constants* for various materials show *aluminum to have the quickest rate of conduction while marble has the slowest.*

## CHILL, DUDE!

There it is. It was surprisingly simple, wasn't it? Simple physics yielding a simple equation resulting in simple requirements for a simple program. In fact, the program is so easy it could be implemented in hardware with TTL gates. Why use software? So we can go beyond the obvious requirement of controlling temperature between fixed ranges at fixed times.

Back in the beginning, I questioned whether you had to be tied to manually adjusting thermostats, dampers, and generally inflexible control systems. The hallmark of software is flexibility. It is easy to add data logging to the simple algorithm. By taking timed samples of temperature in each zone, a dynamic picture of interactions between zones can be seen. This can be used to anticipate temperature changes and hold the level more constant than a simple thermostat system. Time changes no longer need to be programmed as abrupt sleep/wake cycles, but can be smoothly blended over the course of about half an hour. Leave/return cycles aren't needed at all! Room occupancy can be monitored and temperatures adjusted up or down by the same software that shuts off the lights when a room is empty.

I hope this article will inspire you to do some research in this field. Share with us your observations of cooling/heating rates in actual buildings. More data is needed to integrate heating/cooling into a comprehensive home automation system. Significant energy savings combined with increased personal comfort and convenience will ensure market success in the '90s! ❑

*Anthony Segredo holds B.S. and M. S. degrees in Physics. He has 12 years of experience in Logistics and Manufacturing Process Planning as well as 15 years in the Embedded Control Software arena, 5 years of which he has been a consultant.*

# Add a Serial X-10 Interface to Your PC

FEATURE ARTICLE

**Rick Zarr**

If you've ever looked at most "serial port X-10" interfaces out there, you know how software intensive they are. Add one smart single-chip micro, and you'll have plenty of time left for your application.

If anyone of you has ever tried to use a PC to control a TW523 X-10 interface, you may have discovered that the code for implementing the timing in regards to signal generation is very critical. If you are running a multitasking system that generates a context switch every 10 to 20 ms, such as UNIX or Windows NT, or even a nonpreemptive multitasking system such as Windows 3.1, you will find that your interface will fail to operate reliably (if at all]. X-10 is a very demanding signaling scheme even though the data rate is very slow.

The interface I describe here offloads the PC of all of the signal generation tasks and provides a simple, ASCII-based, RS-232 serial interface to the host for very reliable transmission and reception of X-10 codes. Before we look at the interface, let's first look at some of the issues of X-10 transmission and how they relate to PC-type interfaces.

## TIMING...TIMING...TIMING...

My music teacher used to say, "Timing is a virtue," and that still applies to the problem at hand today. The X-10 standard has very tight timing requirements to encode the bits on the power line. Most X-10 interface schemes run timing loops on the PC that generate the correct timing initiated around the zero-crossing output of the TW523 module. If a TSR happens to take control during a timing loop, or a critical interrupt takes over, your timing just went into the bit bucket.

Reception is even more critical because the X-10 power line communication is a broadcast-oriented scheme. If you miss the data, you don't get a second chance! This means the interface must never miss any incoming X-10 commands and not distort the timing of outgoing transmissions.

A good solution is to place a dedicated sequencer, or processor buffer, between the host and the TW523 X-10 interface module. This will remove all the timing requirements from the host. It also allows multitasking systems to control X-10 modules.

## X-10 COMMAND ANTHOLOGY

For those of you who are sketchy on X-10, let's review the principles of operation. X-10 operation is based on 32 *key codes* and 16 *house codes* that are combined into a single command packet (see Figure 1). Of the 32 key codes, 16 represent *unit addresses* and the remaining *16* represent commands (on, off, etc.). A house code is used to identify which group of units will receive commands. Combining a house code with a unit address results in a total of 256 possible addresses for X-10 units.

The structure of a command is simple. The house code always precedes the key code, which makes nine bits (four for the house code, five for the key code), plus a start sequence of two bits for a total of eleven bits. The unit (or units) are first addressed by

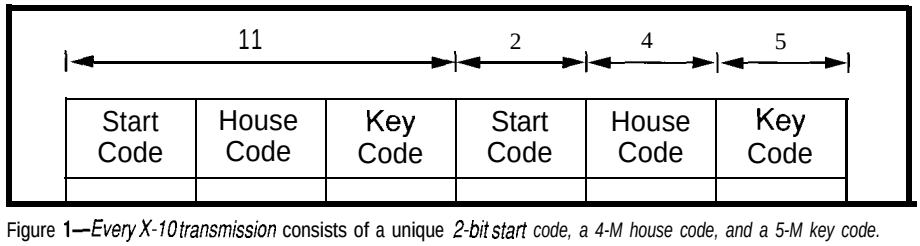



Figure 1—*Every X-10 transmission* consists of a unique *2-bit start code, a 4-M house code, and a 5-M key code. The key code is used either to select a specific module or to issue a command to an already-selected module or modules.*

sending the house code and unit code. This operation tells the units to expect a command. Several units on the same house code can be addressed simultaneously by sending multiple unit addresses before the command. Next, a command or series of commands are sent to the unit(s).

The units remember that they've been selected even after receiving a command, so as long as no new addresses are sent, the same units will receive and carry out subsequent commands. The list of commands is shown in Figure 2.

| House Codes | H1 | H2 | H4 | H8 |
|---|---|---|---|---|
| A | 0 | 1 | 1 | 0 |
| 6 | 1 | 1 | 1 | 0 |
| c | o | o | 1 | 0 |
| D | 1 | 0 | 1 | 0 |
| E | 0 | 0 | 0 | 1 |
| F | 1 | 0 | 0 | 1 |
| G | 0 | 1 | 0 | 1 |
| H | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| J | 1 | 1 | 1 | 1 |
| K | 0 | 0 | 1 | 1 |
| L | 1 | 0 | 1 | 1 |
| M | 0 | 0 | 0 | 0 |
| N | 10 | 0 | 0 | |
| 0 | 0 | 1 | 0 | 0 |
| P | 1 | 1 | 0 | 0 |

| Key Codes | D1 | D2 | D4 | D8 | D16 |
|---|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 | 0 |
| 2 | 1 | 1 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 0 |
| 4 | 1 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 1 | 0 |
| 6 | 1 | 0 | 0 | 1 | 0 |
| 7 | 0 | 1 | 0 | 1 | 0 |
| 8 | 1 | 1 | 0 | 1 | 0 |
| 9 | 0 | 1 | 1 | 1 | 0 |
| 10 | 1 | 1 | 1 | 1 | 0 |
| 11 | 0 | 0 | 11 | | 0 |
| 12 | 1 | 0 | 1 | 1 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 |
| 1410 | | 0 | 0 | | 0 |
| 15 | 01 | | 0 | 0 | 0 |
| 16 | 11 | | 0 | 0 | 0 |

| | | | | | |
|---|---|---|---|---|---|
| All Units Off | 0 | 0 | 0 | 0 | 1 |
| All Lights On | 0 | 0 | 0 | 1 | 1 |
| On | 0 | 0 | 10 | | 1 |
| Off | 0 | 0 | 1 | 1 | 1 |
| Dim | 0 | 1 | 0 | 01 | |
| Bright | 0 | 1 | 0 | 1 | 1 |
| All Lights Off | 0 | 1 | 10 | | 1 |
| Extended Code | 0 | 1 | 1 | 1 | 1 |
| Hail Req. | 1 | 0 | 0 | 01 | |
| Hail Ack. | 1 | 0 | 0 | 1 | 1 |
| Preset Dim | 1 | 0 | 1 | x | 1 |
| Extended Data | 1 | 10 | 0 | 1 | |
| Status = on | 1 | 1 | 0 | 1 | 1 |
| Status =off | 1 | 1 | 1 | 0 | 1 |
| Status Req. | 1 | 1 | 1 | 1 | 1 |

**Figure 2-Once an** *address is sent and not changed, one can send as many commands as they want to the unit.*

Now, the data format and timing gets a bit complicated. The bits are represented by a 120-kHz carrier that is superimposed on the 60-Hz AC power. A bit is represented by the presence or lack of a 1-ms burst of carrier signal synchronized with the zero crossing of the 60-Hz power. The TW523 has a 120-kHz oscillator on board which simplifies the interface.

The interface must toggle the transmit line to enable the carrier signal onto the power line. A start sequence is used to synchronize the data bits and is composed of three 1-ms pulses on successive zero crossings and one idle zero crossing. Next, nine data bits follow. A "one bit" is represented by a "10" pattern or a 1-ms pulse at the zero crossing followed by no signal on the next zero crossing. A "zero bit" is opposite, or 01. The entire sequence is transmitted twice to ensure it gets received.

To simplify the interface, the TW523 module monitors the 60-Hz frequency of the AC power and provides an optically coupled square wave signal that represents the zero crossing. To ensure the signal travels across all three power phases, the pulse is delayed by the phase difference (60") and repeated. This correlates to 2.778 milliseconds between the start of each 1-ms-wide pulse **(1.778** ms between them). This timing is shown in Figure 3.

When receiving, the TW523 looks at the first string of bits it receives from the power line and verifies that the format is correct. It then passes the second copy of the transmission to the receive pin. One drawback of this scheme is you can receive only every third Bright or Dim command (since they are strung together into a continuous stream). The data is synchronized with the zero-crossing signal, and can start on either the rising crossing or the falling crossing.

## GOING TO THE HARDWARE STORE

The circuit in Figure 4 shows the design of the simple serial interface. The microcontroller I used in this design is a National Semiconductor COP8782 OTP (one-time programmable) device with 4K of ROM and 128 bytes of RAM. I chose this processor for its fast instruction cycle and good interrupt support. The instruction cycle using a 10-MHz crystal is 1 us. This fast cycle allows some very fancy timing to be generated.

The host communications are accomplished with a DS14C232 single +5-V supply RS-232 device (with on-

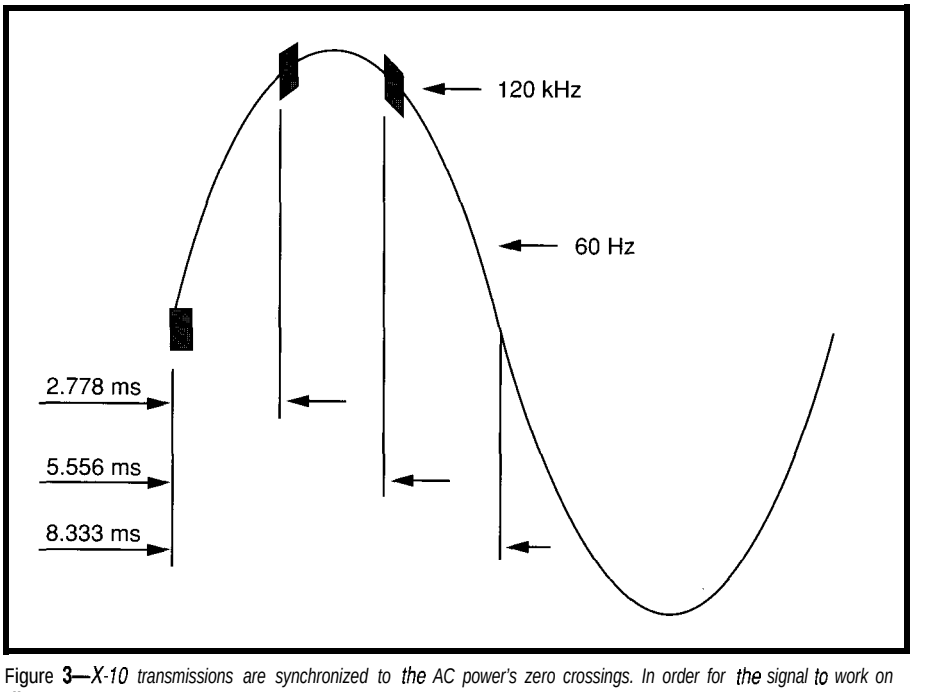



Figure 3—*X-10 transmissions are synchronized to the AC power's zero crossings. In order for the signal to work on* all **three** *AC phases, each bit* **is sent three times, each corresponding to a differentphase's zero crossing.**

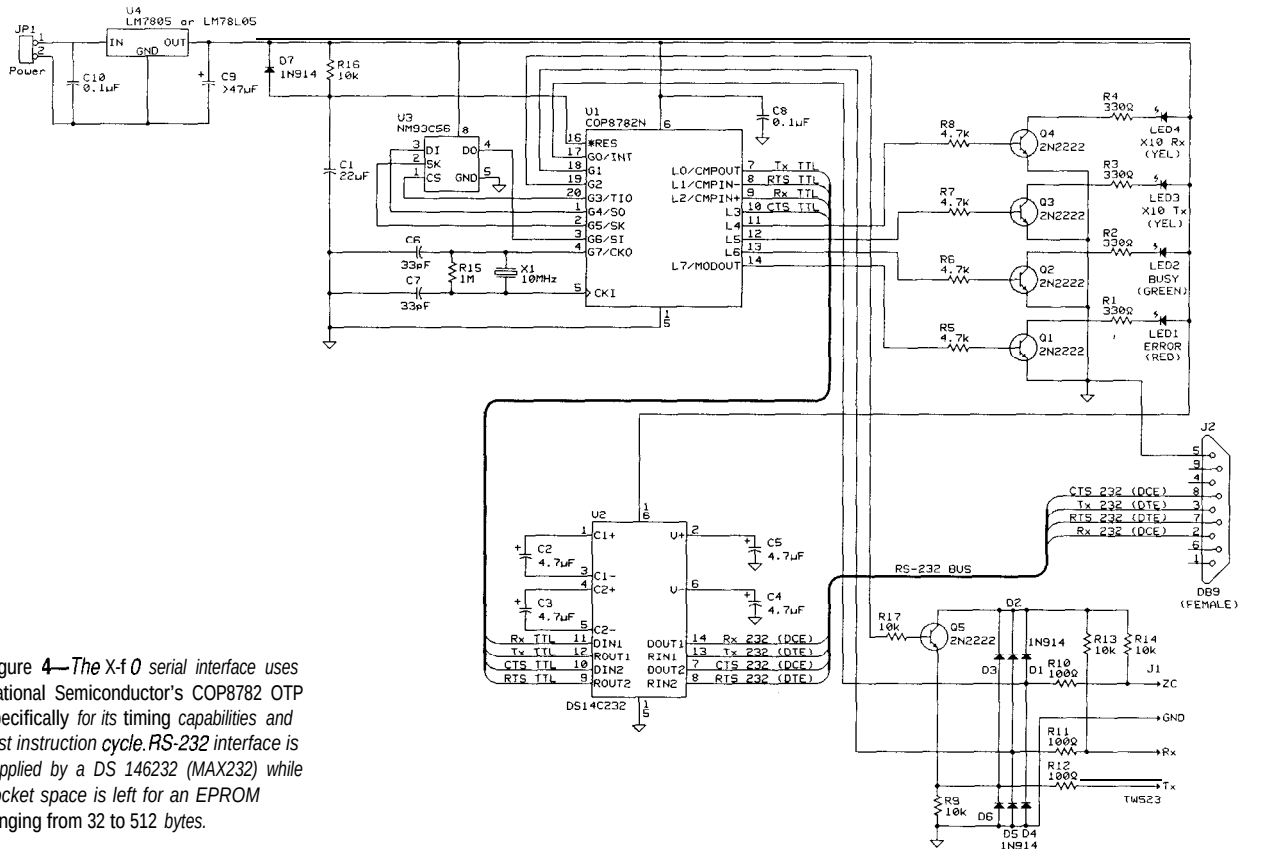



Figure 4—*The X-f 0 serial interface uses* National Semiconductor's COP8782 OTP specifically *for its* timing *capabilities and fast instruction cycle. RS-232 interface is supplied by a DS 146232 (MAX232) while socket space is left for an EPROM* ranging from 32 to 512 *bytes.*

board charge pumps ). This device from NSC has the same pinout as the MAX232.

The TW523 interface is simply some limiting and pull-up resistors, protection diodes, and a transistor to gate the transmitter. Four LEDs and their drive circuits are provided to indicate the X-10 transmission and reception plus the circuit's status.

An optional EEPROM socket is provided on the PC board for expansion of the code to include such features as scene storage, multiple commands, error logs, and any other thing you can think of. The circuit allows devices from a 32-byte EEPROM (NM93C06) all the way up to a 512-byte EEPROM (NM93C66) to work in the same socket. Only the code needs to be modified for the different parts.

The microcontroller provides all the control and timing for both the host serial interface and the X-10 interface. The external circuitry only provides the level translation and drive current.

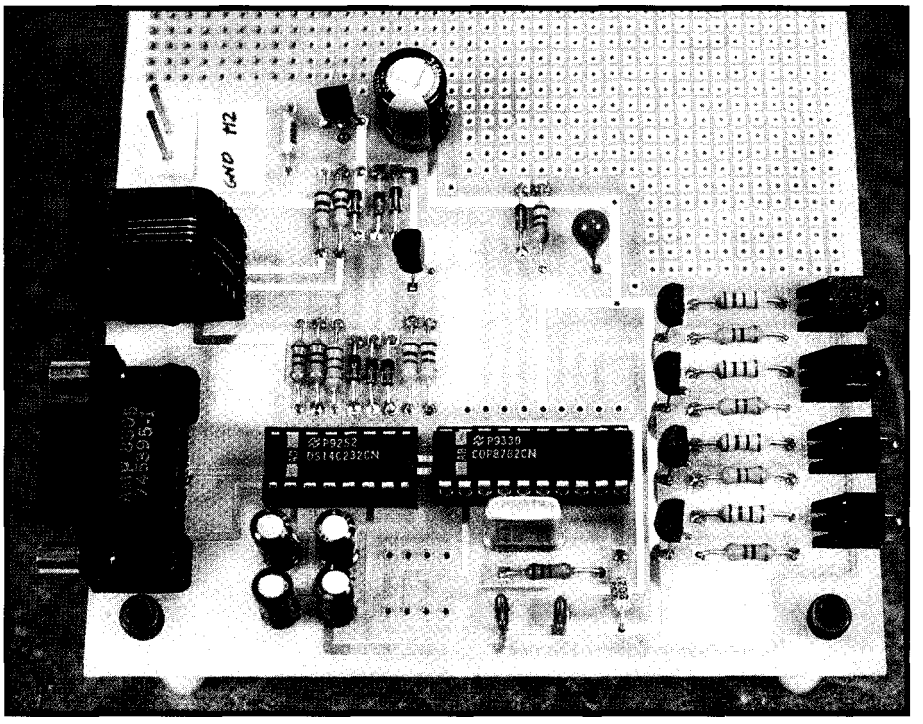


Photo I--The *serial TW523 X-10 inferface uses a COP processor to offload the complex timing requirements from the main processor. The oinly support components needed are for level* shifting and current drive.

## THE MISSING CODE

Obviously, the trick to this design is in the code. It's required to emulate several other pieces of hardware such as UARTs and timers. Figure 5 shows a general block diagram of the major

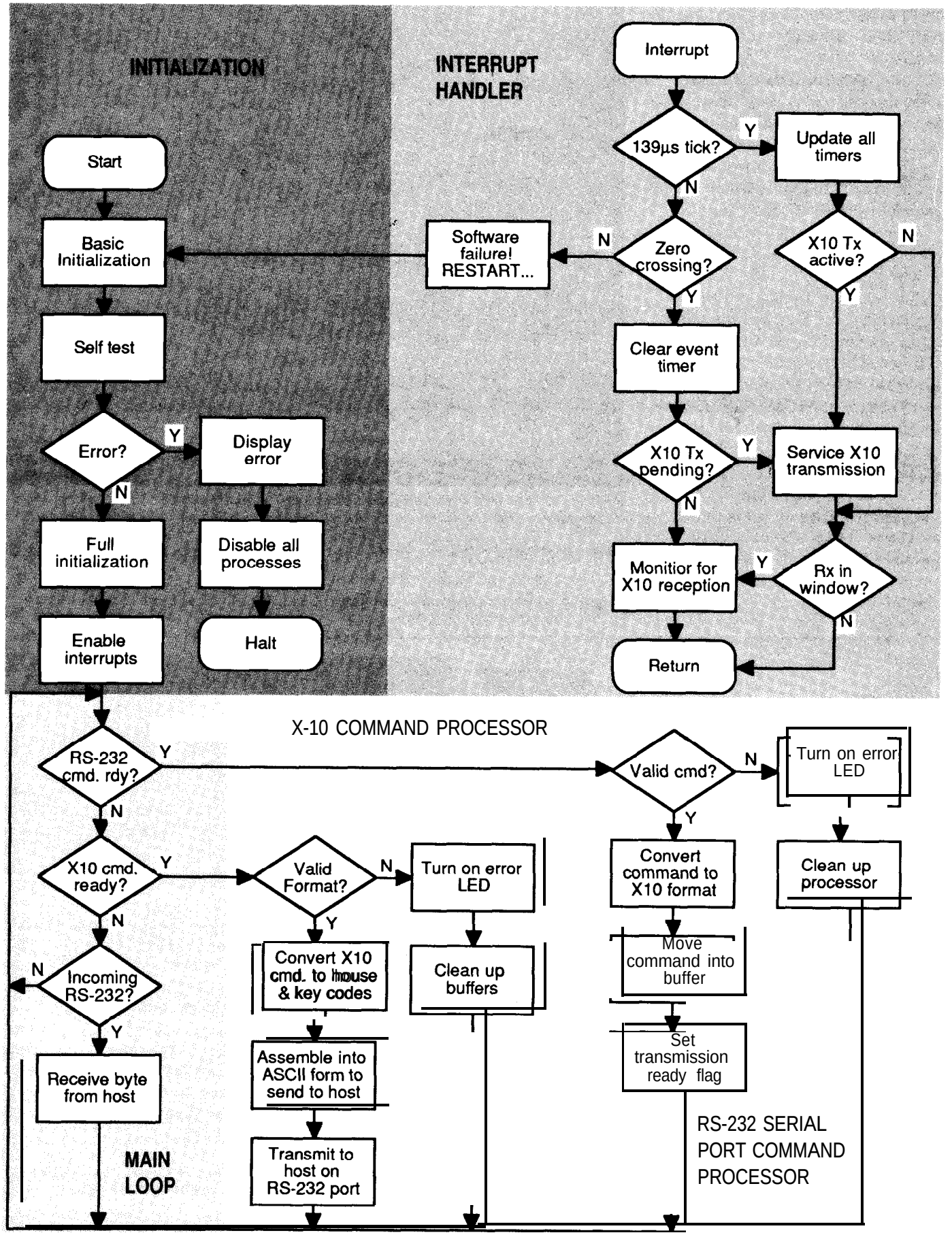



Figure 5—*The support code can be broken info five distinct functions: initialization, main loop, command processor, serial port support, and the interrupt handler.*

code segments. The self test and initialization comes first, then the code enters a general "watch for events" loop. This loop waits for various events to occur such as hardware, timer, and zero crossing interrupts. It also polls the slower RS-232 receive line for activity.

When a command to start an X-10 transmission is received, the loop dispatches the order to the appropriate module. The module then goes about setting up all of the buffers with the X-10 data to be transmitted and returns to the loop to wait for the next zero crossing. When the next zero crossing occurs, the data is shifted out at a rate of one bit per zero crossing. This is done twice to ensure the data gets to its destination (as per the X-10 specification). Once this transmission is complete, control is returned to the main loop. Most of the CPU's time is spent in the loop waiting for something to happen.

When an X-10 reception is starting [checked during the zero crossing), control is given to the X-10 receiver to reconstruct the command. Once this is complete, the code translates the X-10 command into its equivalent ASCII representation and sends it to the host via the UART emulation.

Using the external interrupt pin to sense the zero crossing signal relieves the microcontroller from polling the TW523. With this signal, the microcontroller can generate all of the X- 10 receive and transmit timing which will be synchronized with the zero crossing. A software timer is started when the zero crossing is detected and is used to generate all of the X- 10 timing. Every rising zero crossing, this timer is restarted to make sure it does not drift.

In order to give the code enough resolution for the timing of both the UART and X-10 interfaces, I chose a period of 139 µs for the hardware timer interrupt. This equates to about 120 ticks every 1/60 of a second [one X-10 bit time]. It also equates to six ticks per bit period of the 1200-bps UART interface to the host (real convenient, huh?). Therefore, all the system software timers are in equal intervals of this 139-µs tick. Every tick, the code updates the various timers. It also checks the X-10 transmission and reception status for pending events. This offloads the main loop from timing the various critical events with software loops (ugh!). Whenever the code is instructed to transmit an X-10 command, it just sticks the bits in a buffer, sets a "transmit pending" bit, and the interrupts do the rest.

To make interfacing with this device easier, I've defined a special command structure for the ASCII codes as shown in the sidebar. For example, all raw X- 10 transmissions are started by sending the circuit an ASCII "X." In this case, the next byte is a command byte of either an ASCII "0" or a "1." This will tell the processor what will follow the house code—either a unit address or a command code.

The next byte is the house code. House codes (which are normally A-P) are reduced to the hex equivalent of O-F. For example, house code "J" would equate to 9 hex. The unit or command code byte is last.

Let's do an example. If I want to turn on unit A6, two ways exist to do it. The first (and easiest) is to use the On command or the ASCII letter "N." Using this method, the command looks like "N005." The other way is use raw X-10 commands to first select the unit, then turn it on. You would first send "X005" to address the unit, then send "X101 " to turn it on. This second method will produce the same transmission as the first.

You may be asking yourself, "Why provide two methods when it's easier to do the first?" Well, there may be times when you need to turn on many things all at the same time on the same house code. For instance, you may want to turn on units Al, A3, and Al4 and dim them to a medium level. Using the raw X- 10 command method, you can address all three of them first, and then send the commands. All the units will respond to the command simultaneously rather than one at a time.

Another reason is that you may want to implement your control scheme differently than others. This will allow you flexibility for more

## TW523–TO–RS-232 COMPUTER INTERFACE CODES ANDSTRUCTURES

The computer interface uses ASCII commands to carry out its functions. These commands are outlined below. The basic structure is a letter command such as "N" for On, followed by data to determine the house code, address, and type of action (all on, one unit, etc.).
For example, if you wanted to turn on unit A4, you would send the ASCII string: N003.
The controller would respond with an asterisk (*) when completed, or an "E" if there was an error followed by a code to further explain the error. Here is a typical X- 10 session:

```
Computer: N000      ; Turn on unit Al
Controller: .       ; Done...
Computer: D705      ; Dim unit A6 to level 7
Controller: *       ; Done...
Controller: X004    ; Controller reports X- 10 Received (HC=A, Unit 5)
Controller: X102    ; Controller reports X-10 Recieved (HC=A, On)
Computer: F21       ; Turn off all units, house code B.
```

Look at the last command from the computer. It instructed the controller to send an X-10 All Units Off command to house code B. There was no need to send the address of the unit since the command applies to all addresses on that house code. This is true of all commands that address more than one unit. Reset ("R"), for instance, does not require any other byte to reset the controller. If the controller needs more data, the green "Busy" LED will stay lit until enough bytes have been received to satisfy the command requested. The first byte is the Command byte. The next byte (if required) is the Type byte and is used to further define the action. The next byte is the House Code byte (O-F hex) and describes house codes A-P. The last byte (if required) is the Address byte (O-F hex] that describes the unit address. The exception for the address byte is when sending raw X-10 commands with the "X" command (also receiving). If the byte following the "X" is "0," then the last byte is the address. If the byte following the "X" is a "1," then the last byte is an X-10 command code such as All Lights On. See the above example. Below is an overview of all the commands and their extensions.

### Code Overview

House Codes: O-F (hexadecimal for house codes A through P)

Addresses: O-F (hexadecimal for unit codes 1 through 16)

Commands:

- F : Off
- N : O n
- D : Dim
- S : Status
- X : Send raw X-10 code
- H : Hail Request (checks for other controllers)
- R : Reset X-10 controller

Data for Commands:

| Cmd | Data Byte |
|---|---|
| F : | |
| | 0: Turn off single unit |

complicated control as well as the simplicity you may want for simple commands. Also, it provides a way to receive the raw X-10 commands from other controllers. All that is required of the host is a serial device that can communicate at 1200 bps with 8 data bits, no parity, and 1 stop bit (the fixed mode of the software UART).

Reception is a bit more difficult. Look for the start sequence on both the rising and falling zero crossings as mentioned above. The code for this project looks at both edges for the start of data, then determines what polarity to use for synchronization. Once it determines the edge, then the same code section decodes the X-10 reception, converts it to ASCII, and transmits it to the host via the RS-232 interface. Since the software is receiving raw X- 10 commands, it converts them to the equivalent ASCII for these raw commands as shown in the sidebar and sends the ASCII characters to the host. This makes working with the data a piece of cake. All the computer has to do is look for the incoming "X" and read three more bytes. These bytes will determine what X- 10 event occurred.

## GET CONTROL

You can watch all X-10 events as they happen or control any X- 10 device by using a simple terminal or terminal software on your PC running as described above (1200 bps, 8 data bits, no parity, 1 stop bit). However, if you're feeling really courageous, you might want to run your control stuff in a multitasking environment such as Windows or Windows NT.

It turns out that in the 3.1 release of Windows, Microsoft added very good support for serial communications. The SDK from Microsoft, Visual Basic and C++, and the Borland Windows tools will all support code generation for serial communications under Windows 3.1. This means you can write your own high-end (glitsy!) control program using the Windows interface and run other stuff too without worrying about the X-10 interface.

I've developed a simple X- 10 controller program using Borland's

Turbo Pascal 7.0 that runs under Windows 3.1. The complete listings, resource files, and compiled code are available on the BBS. If you'd rather not get so fancy, you can use this X-10–to–RS-232 interface on ***any*** serial device (modem, terminal, PC, etc.) that meets the baud rate criteria. So have at it. Start controlling your X-10 devices from the serial port and free your system of the nasty X-10 timing requirements. ❑

***Rick Zarr holds a B.S.E.E. from the University of South Florida and is currently employed as a Staff Applications Engineer at National Semiconductor. His interests include home automation, computers, and Karate.***

## SOURCE

The preprogrammed microcontroller is available from Marrick Limited Inc. for $29.95, plus S/H. The double-sided PC board with a prototyping area is also available for $24.95, plus S/H, and a complete development kit is available which includes the preprogrammed microcontroller, PC board, and software for both Windows and DOS, plus complete documentation and software listings for $59.95, plus S/H. Send check or money order to: Marrick Limited, Inc., P.O. Box 950940, Lake Mary, FL 32795. Attn: X-10 Development Kit offer; or call (407) 323-4467 for more information. Florida residents must add 6% sales tax. Add $3.00 S/H for standard ground, and $5.00 for second-day air.

## SOFTWARE

Software for this article is available from the Circuit Cellar BBS and on Software On Disk for this issue. Please see the end of "ConnecTime" in this issue for downloading and ordering information.

## I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

Example: F01F (Turns off unit B16)
1: Turn off all lights this house code
Example: F10 (Turns off all lights in house code A)
2: Turn off all units this house code
Example: F24 (Turns off all units in house code E)
3: Turn off all lights, all house codes
Example: F3 (All lights off, house codes A through P)
4: Turn off all units, all house codes
Example: F4 (All units off, house codes A through P)
5-F: (reserved)

N:
0: Turn on this unit
Example: N0A2 (Turns on unit K3)
1: Turn on all units this house code
Example: N1F (Turns on all units in house code P)
2: Turn on all units, all house house codes
Example: N2 (All units off, house codes A through P)
3-F: (reserved)

D:
O-F: Light level of Dim (0 = off, F = full, 8=medium)
Example: D422 (Dims light to level 4)

s:
0: Send Off status for unit
Example: S000 (Status of unit Al is Off)
1: Send On status for unit
Example: S100 (Status of unit Al is On)
2: Request status of unit
Example: S200 (What is status of unit Al)
3-F: (reserved)

x:
0: Number code: HC = O-F (Unit number 1-15, respectively)
Example: X011 (Send house code B unit 2 code)
1: Cmd Code: HC = 0: All Units Off
1: All Lights On
2: On
3: Off
4: Dim
5: Bright
6: All Lights Off
7: Extended Code
8: Hail Request
9: Hail Acknowledge
A: Preset Dim 0
B: Preset Dim 1
C: Extended Data
D: Status = On
E: Status = Off
F: Status Request
Example: Xl 12 (Send house code B command On code)

H:
0: Hail this house code
Example: HO9 (Hail house code J)
1-F: (reserved)

R:
(none): Reset System (forces internal code reset)
Example: R(Resets controller)

# Designing Printed Circuits for High-speed Logic

FEATURE ARTICLE

**David Prutchi**

With the ever increasing use of high-speed buses, gone are the days of treating logic signals as discrete ones and zeros. Find out just some of the issues involved in laying out a PC board for such high-speed systems.

I've been thinking about buying a new car, and whenever I browse through high-performance car magazines, I can't help dreaming what it would be like to own a Ferrari F40, capable of achieving a speed of 200+ MPH. Just imagine: its magnificent V-12 engine purring while cruising down the road at a speed on the limit of my reflexes.

Waking up to reality though, I know that in my day-to-day life I would never be able to floor the gas pedal of this marvel. Even if I decided to disregard the 55-MPH legal limit, our roads are just not designed to support more than half the maximum speed of a loaded sports car. The awesome power of sports engines can be let loose only in special race tracks, constructed with the right materials and slants.

Although you may not consider adding a Ferrari to your estate at this very moment, you must have thought about using one of those new lightning-fast microprocessors for your next project. However, similar to my sports car analogy, high bus speeds result in interconnection delays within the same order of magnitude as on-chip gate delays, and for this reason typical printed circuit board design-which considers traces as low-frequency conductors rather than high-frequency transmission lines-will ensure that such a project turns into a very impressive and expensive paperweight.

Almost fifteen years ago, while most of us were building microcomputers with 2-MHz Z80s, 8080s, and CDP1802s, engineers designing with ECL technology already faced some of the problems related to the implementation of printed circuit boards, backplanes, and wiring for high-speed logic circuits. Today, however, 66-MHz buses are commonplace, and we should all accept that the Utopian idea that signals behave as ones and zeros must be replaced by a more realistic approach which involves RF transmission-line theory. Through this new approach, printed circuits are designed to convey pulse transmissions with minimal distortion through channels of appropriate bandwidth-no quasi-DC signals anymore !

## A TRANSMISSION-LINE MODEL OF PCB TRACK

PCB design for high-speed logic demands the use of power and ground

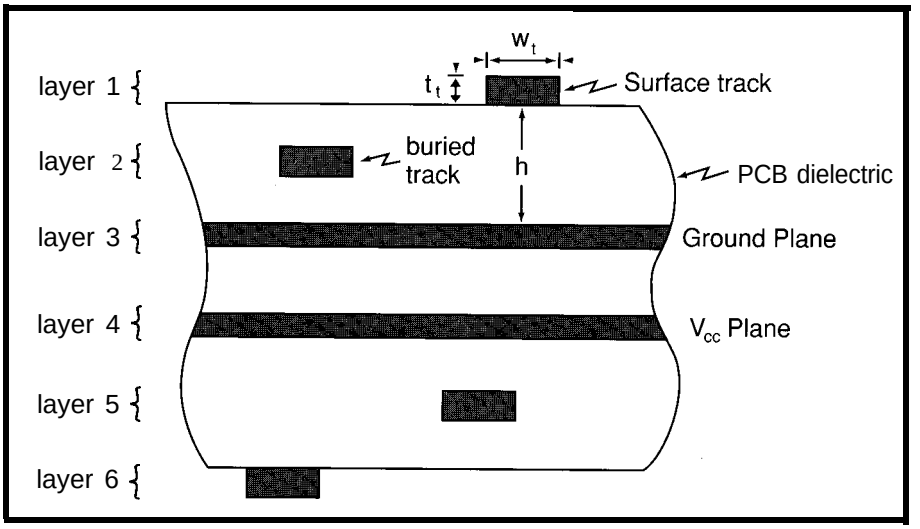



Figure 1--The transmission line impedance of a stripline PCB track is affected by its position relative to the ground or power plane as well as by its geometry and by the dielectric constants of the board and the surrounding medium.

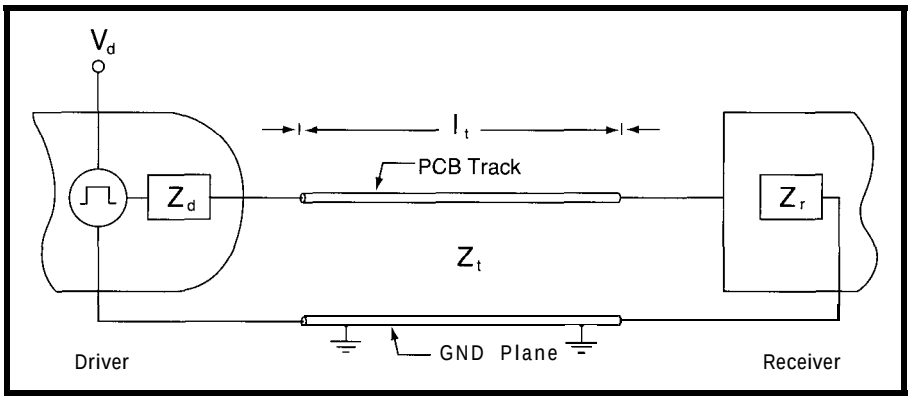



Figure 2-The output of a logic element connected through a PC6 track to the input of another logic element can be modeled as an ideal voltage step generator which drives a transmission line of impedance Z, through an output impedance $Z_d$. The transmission line is then terminated by the receiver's input impedance $Z_r$.

planes, and double-sided PCBs are not recommended. In the former, a surface stripline track, such as in Figure 1, will have an impedance $Z_t$ given by:

$$Z_t = \frac{87}{\sqrt{\mu + 1.41}} \ln\left(\frac{5.98h}{0.8w_t + t_t}\right) \qquad (1)$$

where $\mu$ is the dielectric constant of the PCB dielectric, $h$ is the height of the track above ground or power plane, and $w_t$ and $t_t$ are the width and thickness of the track, respectively. A PCB track buried within the fiberglass/epoxy laminate will have an impedance reduced by about 20% compared with the surface track.

This PCB track can be modeled as a transmission line [1]. A short pulse applied to one end of this transmission line will appear on the other side, supplying the load impedance $Z_l$ with a distorted version of the pulse, and presenting an effective delay, $\tau$. On a typical surface PCB track, the pulse conduction velocity is approximately 0.15 ns/inch = 0.06 ns/cm, so,

$$\tau = 0.151, \text{ [ns]} \qquad (2)$$

represents the total delay caused by a track of $l$, length (measured in inches).

If the load impedance does not perfectly match the track's impedance, then a part of the arriving signal will be reflected back into the transmission line. In general, pulse reflection occurs whenever transmission lines with different impedances are interconnected, or when a discontinuity occurs in a single transmission line. For a connection between two transmission lines of impedances $Z_0$ and $Z_1$, the reflected voltage $V_r$ is related to the incident voltage $V_i$ through

$$V_r = \frac{Z_1 - Z_0}{Z_1 + Z_0} V_i \qquad (3)$$

The ratio $\psi = \frac{V_r}{V_i}$ is called the *reflection coefficient,* and describes what portion of the pulse incident from $Z_0$ on $Z_1$ will be reflected back into $Z_0$.

$\psi$, $V_i$, and $V_r$ are usually complex quantities because they deal with both the magnitude and phase of the signals that travel along transmission lines.

## PULSE REFLECTION AND TERMINATION TECHNIQUES

In a typical circuit, a driving logic element and a receiving logic element are connected by a PCB track. In the equivalent circuit shown in Figure 2, a pulse with amplitude $V_d$ is injected by the driver logic element, which presents an output impedance $Z_d$, into a PCB track of length $l_t$ and impedance Z,. The pulse carried by the PCB track is then presented to the input impedance $Z_r$ of the receiving element.

If we suppose that a $Z_t = 100\Omega$ track carries the pulse, and after looking at the data sheets for a selected 5-volt family of high-speed logic, we find that the output and input impedances at our frequency of interest are $Z_d = 50\Omega$, $Z_r = 10k\Omega$, respectively, then upon reaching the receiver, a reflected pulse starts traveling back towards the driver with amplitude that can be approximated by

$$V_r = \frac{Z_r - Z_t}{Z_r + Z_t} V_d = \frac{10000 - 100}{10000 + 100} 5\,[V] = 4.90\,V \qquad (4)$$

which assumes a negligible attenuation of the pulse throughout its conduction, and which takes into consideration only the real parts of the variables. This reflected pulse will be rereflected back toward the receiver upon hitting the driver with an amplitude approximated by

$$V_{r'} = \frac{Z_d - Z_t}{Z_d + Z_t} V_r = \frac{50 - 100}{50 + 100} 4.90\,[V] = -1.63\,V \qquad (5)$$

This negative signal will interact with the original incident pulse with a

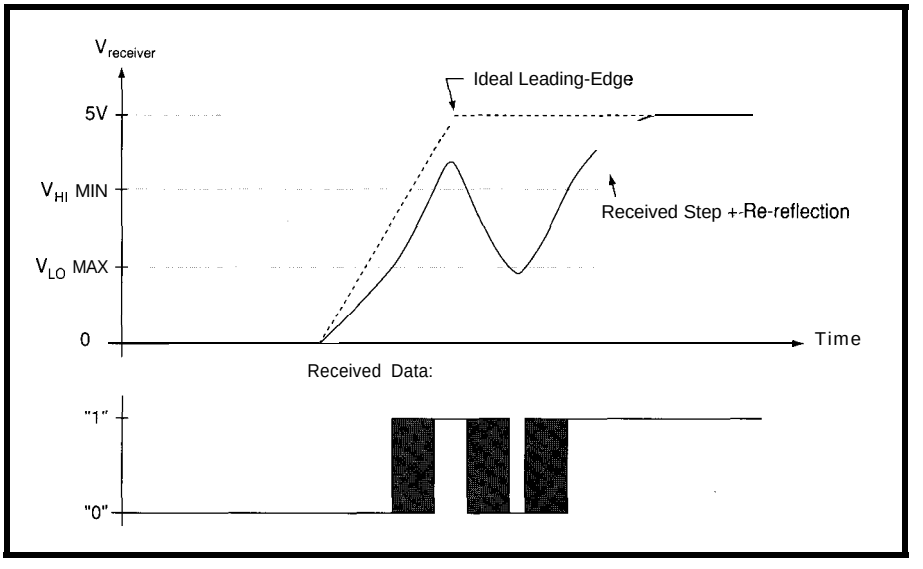



Figure 3--A critical length of PCB track could cause the rereflected pulse to distort the leading edge of the pulse so much that if will cause, in this case, the false defection of a logic-low.

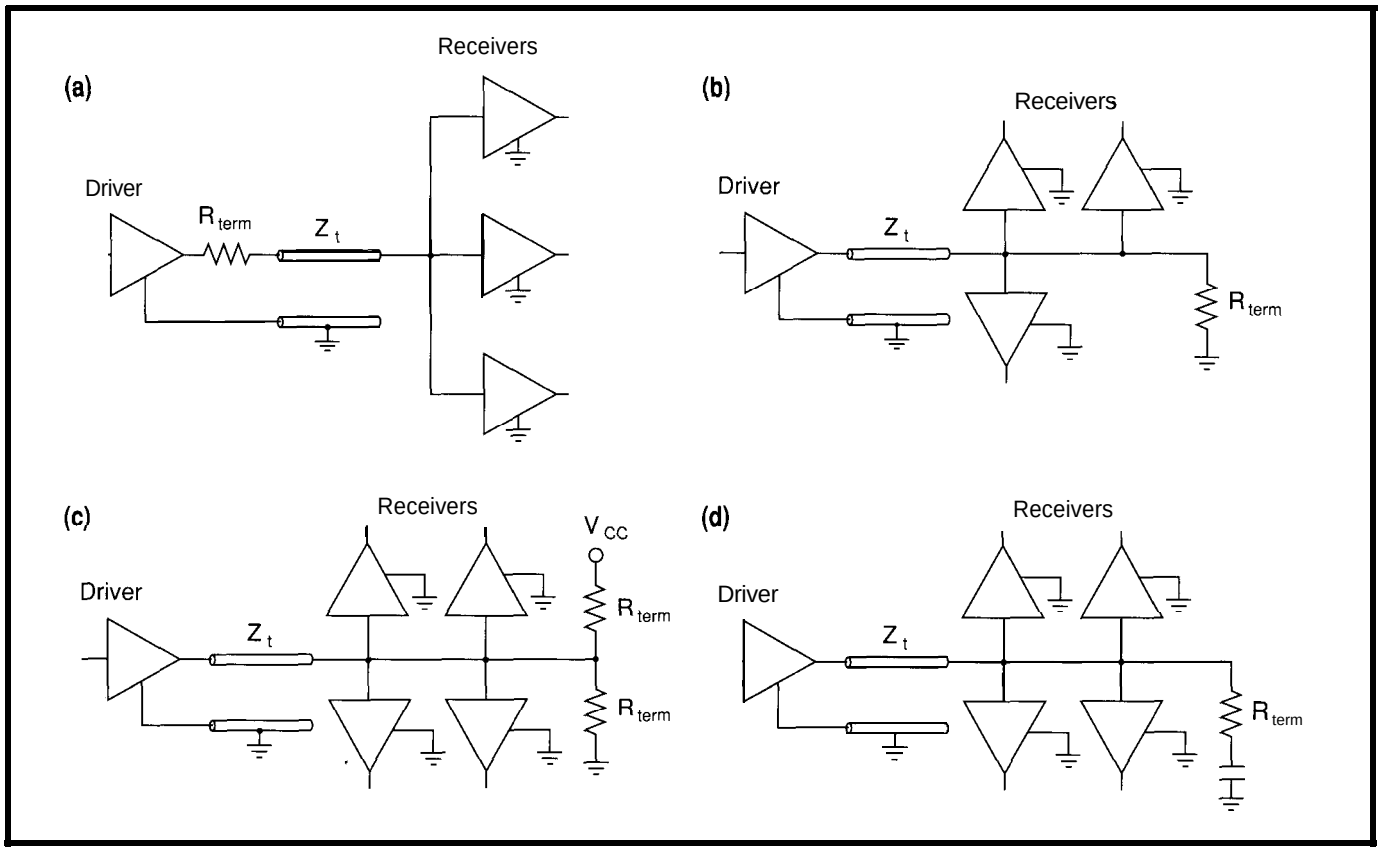



Figure *4-Transmission line termination techniques include a) series termination, b) parallel termination, c) Thevenin termination, and d) AC termination.*

delay equivalent to the time it takes for the pulse to travel back and forth the track (twice that of Equation 2):

$$\tau = 0.31, [ns] \quad (6)$$

Depending on the length of the PCB track, the -1.63-V reflection could distort the leading edge of the pulse so much that it will cause the false detection of a logic-low (Figure 3). A different combination of impedances could have caused the reflected pulse to be positive, possibly causing the false detection of a logic-high, or the false activation of an edge-sensitive device. Moreover, a reflected pulse presented to the receiver will cause yet another reflected pulse which, although with far less amplitude, may still be able to cause erroneous operation of a circuit.

Obviously, the solution to the reflected-pulse problem is to match the impedances in the best possible way. This design procedure is called *transmission line termination,* and can be accomplished in four different ways: series, parallel, Thevenin, and AC, as shown in Figure 4.

Series termination is recommended whenever $Z_d < Z_t$ and the line is driving a reduced number of receivers. This technique, which gives good results in most high-speed TTL circuits, consumes negligible power and requires the addition of only one resistor, the value of which is given by

$$R_{term} = Z_t - Z_d \quad (7)$$

The major drawback of the series termination technique is that it increases signal rise and fall times.

In contrast to series termination, which eliminates pulse reflection at the driver end, all other techniques eliminate reflection at the receiver end of the PCB track. Parallel termination $(R_{term} = Z,)$ as well as Thevenin termination $(R_{term} = 2Z_t)$ techniques consume large amounts of power, however they provide very clean signals. AC termination $(R_{term} = Z,)$, which uses a small capacitor to couple only AC components to ground, is not as power hungry as the previous methods, but adds capacitive load to the driver and increases the time delay due to its inherent RC constant.

## PARALLEL PATH SKEW AND TRACK LENGTH EQUALIZATION

Parallel transmission over data and address buses requires that all signals arrive concurrently at their destination. Often, however, pulses sent down parallel paths don't arrive at the same time because of differences in the length of these paths. As shown in Figure 5, the skew induced in bit sequences sent along parallel paths of different lengths can cause errors in the communication between circuits, specially when transmitter and receiver circuits are placed in different boards interconnected through backplanes or ribbon cables. The obvious solution is to keep parallel paths as short as possible, and ensure equal PCB track lengths for all parallel paths.

Skew also deserves very serious consideration in the design of high-speed microprocessor clock distribution networks. In general, all logic computation during a single clock cycle has to be performed within the very short time left over because of the delays suffered by logic signals during that clock period. Path delays, setup

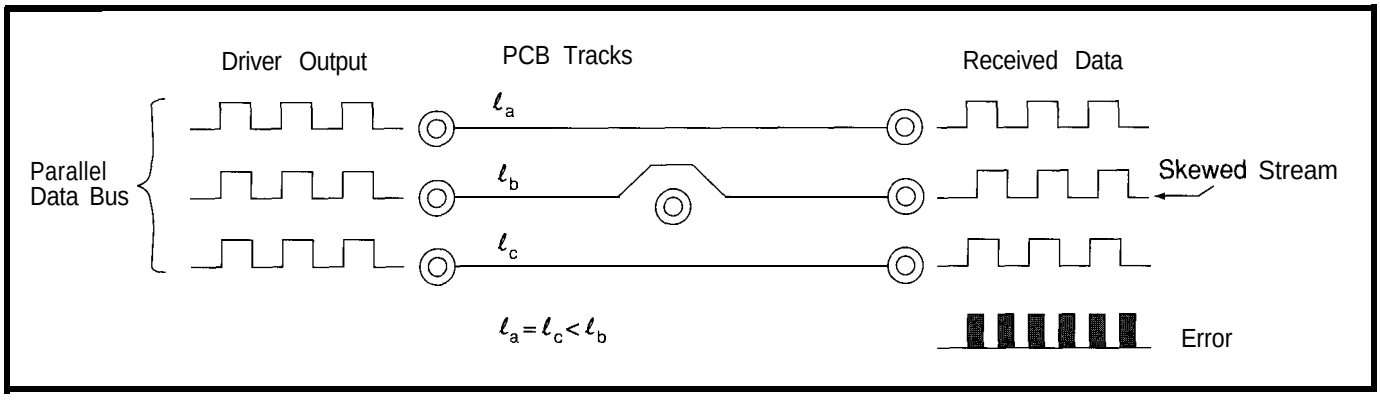



Figure 5—*Unequal* lengths of *PCB track on a parallel bus will cause skew between the pulse streams, leading to reception* errors

and latchup times, logic gate propagation delays, and skew all have to be considered. As clock frequencies increase, the time left over for logical computation decreases, up to the point that skew often causes system failures due to incomplete processing during a clock cycle.

For this reason, PCB tracks that distribute the clock must be tuned so that the delay from the clock driver to each load is the same. Whenever possible, the loading on each track carrying the clock should be the same, and in this case skew is minimized by making all tracks the same length. For unbalanced loads, delay times can be tuned through RC terminations or by carefully adjusting the track lengths.

## CROSSTALK

Crosstalk is the noise induced into a track by the presence of a pulse stream in an adjacent track. The amount of crosstalk is affected by track spacing, routing, signal direction, and grounding. The major problem with crosstalk arises when the voltages induced onto a quiet line are sufficient enough to be detected as a change in logic state by the receivers of that line. In high-speed systems, the capacitive and inductive coupling between lines is considerable, and crosstalk must be reduced through appropriate design.

First of all, proper transmission line termination reduces the amount of radiated energy from a driven track, and spurious emissions that escape nevertheless can be shielded through the use of grounded guards. This design consideration is particularly important for lines driven with high-voltage, high-current, and high-frequency signals. Floating lines connected to high-impedance receivers are notably sensitive to crosstalk, and proper shielding, as well as maintaining a minimum distance from possible radiating tracks, must be ensured. In addition, it is possible to see from transmission line theory that crosstalk between two adjacent tracks is minimized if the two signals flow in the same direction.

## ANALYSIS OF CIRCUIT BOARD PERFORMANCE

Although you may consider such tools as a time-domain reflectometer or an RF network analyzer as belonging strictly in a communications lab, these can aid considerably in the design of circuit boards for high-speed applications. These tools are capable of measuring the actual impedances, time delays, and complex reflection coefficients of a circuit.

These measurements often show that calculations of these parameters result in very crude estimates which

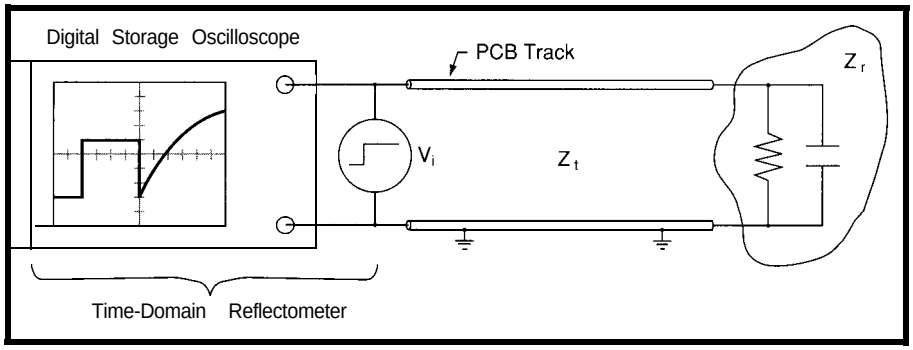



Figure **6-A** time-domain *reflectometer injects a voltage sfep with very short rise time info a transmission line. After a certain delay, a reflected pulse adds* ***up to the step. The timing*** *and waveshape of the reflected pulse contain information regarding the characteristics of fhe transmission line and the termination.*

have to be improved upon for good circuit performance. In most cases, the iterative process of design will require building and evaluating a test board in order to determine if the original design considerations were effective. This test board is usually not populated with the actual active components, but the PCB tracks, passive components, sockets and connectors, as well as the terminated dummy IC packages form a network of transmission lines which can be analyzed with confidence.

A time-domain reflectometer (TDR) [2], which is often used in the troubleshooting of LANs, injects a very sharp pulse into the transmission line under analysis. Then, an oscilloscope or a computer fitted with a high-speed ADC receives the reflected pulses. The time delay and shape of the reflected pulses contain the information required to estimate the impedance of the line and its termination (Figure 6).

In contrast to the TDR, network analyzers (Figure 7) operate in the frequency domain and enable the exact measurement of the complex reflection coefficient as a function of frequency, measurement of crosstalk between lines, and measurement of phase skew between signals [3]. A network analyzer can also be used to identify tracks on which resonance problems could arise and to perform objective crosstalk measurements.

Regardless of the usefulness of these tools, their price puts them out of reach for most electronics experimenters and small engineering firms. But don't be discouraged: building successful high-speed logic circuits on a budget is possible by adopting conservative design policies [4]. As you may realize by now, an analog circuit simulator could be as helpful as a digital circuit simulator in the design of your next high-speed logic circuit [5].

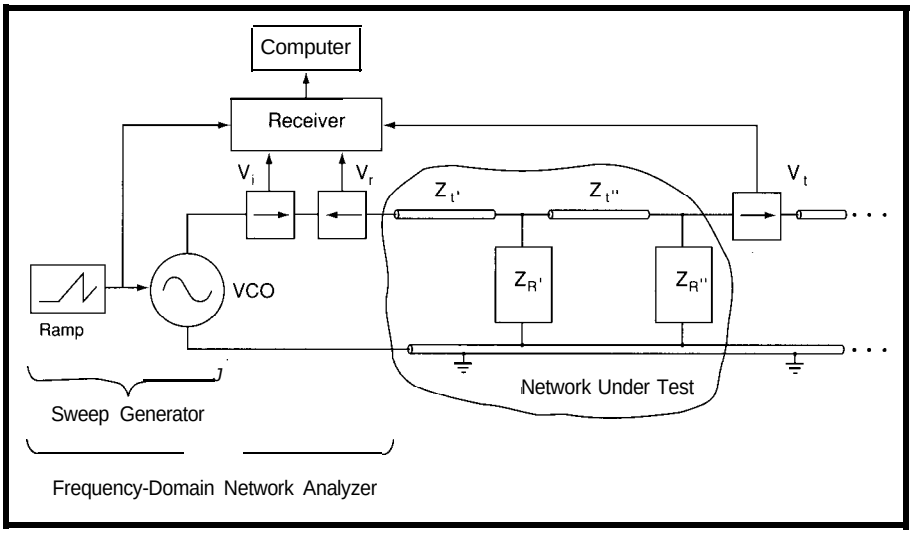



Figure **7**—*The frequency-domain RF network analyzer injects a sweeping sinusoidal signal* into *the input of the transmission* line leading *to* the *subnetwork under test. Directional couplers feed a synchronized RF receiver with samples of the incident, reflected, and transmitted portions of the signal. A computer is used to calculate and display the complex reflection and transmission functions, as well as other relevant parameters.*

## CONCLUSION

Many EEs that work on digital design have long forgotten about analog and RF design. As I have tried to show in this article, however, RF techniques prove to be essential in the design of circuits that can exploit the power of modern high-speed processors. With the advent of 66-MHz Pentium chip buses, 1 OOM-bps transputer links, and ultra-high-speed DSPs, it's time for us to open our dusty RF theory books and consider their teachings under a new light.

***David** Prutchi **has a Ph.D. in Biomedical Engineering from Tel-Aviv University. He is currently a researcher at Washington University where his main R&D interests are array acquisition and parallel processing of biosignals.***

## REFERENCES

1. L.M. Magid, ***Electromagnetic Fields, Energy, and Waves,*** John Wiley & Sons, New York, 1972.
2. D. Strassberg, "Time-Domain Reflectometry: In >50-MHz Digital Design, Measurements are a Must," ***EDN,*** August 19, 1993, 65-72.
3. D. Montgomery, "Borrowing RF Techniques for Digital Design," ***Computer Design,*** May 1982, 207-217.
4. H.W. Johnson and M. Graham, ***High-Speed Digital Design-A Handbook of Black Magic,*** Prentice Hall, Englewood Cliffs, 1993.
5. J. Frost, "Backplane-Design Basics Help Avert System-Design Problems," ***EDN,*** July 8, 1993, 122-130.

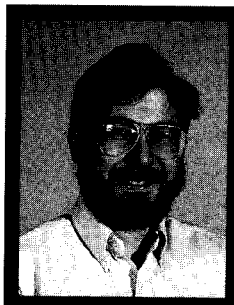
## I R S

410 Very Useful
411 Moderately Useful
412 Not Useful

## DEPARTMENTS



# Lots'a Dots: Big Bit-mapped LCD Panels for the '386SX Project



Everything you've ever wanted to know about LCD panels but were afraid to ask is the subject of this month's and upcoming Firmware Furnace articles. Ed sheds some light on the topic before adding it to his development board.

## FIRMWARE FURNACE


**Ed Nisley**


You've probably been tempted by those ads for cheap surplus LCD panels, but the phrase "external controller required" translated into "one of these days...." Figuring out how to use those panels was just hard enough that you never got around to it, right?

I've been thinking about them for a while, too. What pushed me over the edge is an upcoming series of '386SX projects that will need a big display that *isn't* a CRT or a serial port. It turns out that a handful of logic gates and a dollop of firmware can transform a cheap surplus panel into a useful display. Best of all, you need not wedge a specialized LSI widget between you and the dots.

Overall, this project is about as complex as the whole rest of the Firmware Development Board, but Ken is allowing me five issues to cover it in detail. Be patient...this is good stuff!

This month I'll describe how graphic LCD panels work, present condensed specs for several hot prospects, and go over the interface block diagram. You'll be in a better position to understand why the Graphic LCD Interface works the way it does when you see what all the signals do when they get to the panel.

In February and March I'll present the new FDB hardware and some test code to get a panel running on your system. Because there are many different panels, the circuitry has an unusual number of options. I'll describe some embellishments you may want to add for some specific panels that the standard interface can't quite handle.

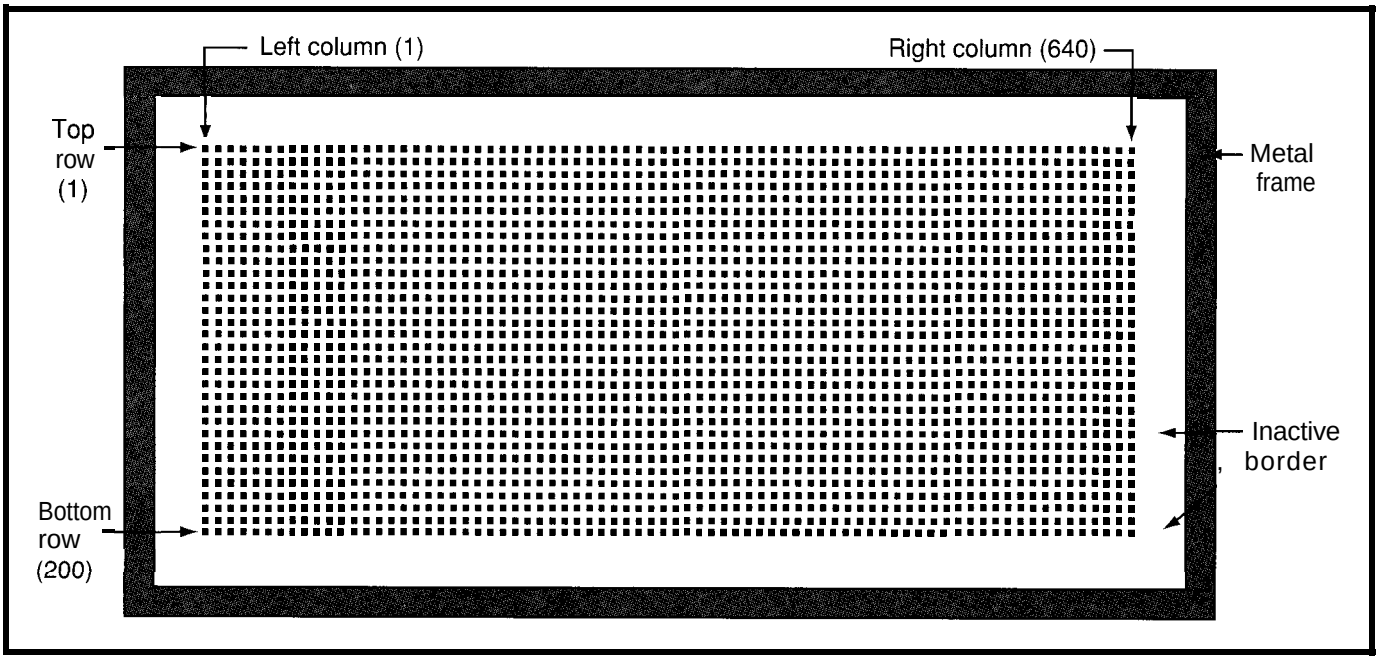



Figure 1-A graphic LCD panel is essentially a *thin double-paned window: what you actually see is the backlight on fop of the circuit board. Transparent electrodes laid out in horizontal rows on one pane and in vertical columns on the other delineate the dots at their intersections. The liquid crystal formula between the panes reacts to an applied electrical field by rotating fhe transmitted light's polarization; polarizers outside the panes cause dots to appear dark or light as the polarization changes. The metal frame clamps the whole affair to the underlying circuit board and ensures good connections to each of the hundreds of column and row drivers.*

The April issue will explore graphics code using Conway's Game of Life to generate the dots. Each type of panel has a different memory layout, so the task is more complex than it seems. Fortunately, graphic output devices are easy to debug because you can see your errors!

Finally, in May I'll make those panels do what I need: plain, simple, fast character output that will come in handy for status displays. The LCD handler will use ANSI cursor control codes so the same output data can drive the LCD panel or a communications program through the serial port. If you have a 640x400 panel, you'll get 50 lines of 80 characters each. That should be enough for a while.

## DOTS IN RANKS AND ROWS

Graphic LCD panels are not hard to understand, but the nomenclature doesn't help. Each manufacturer uses different names for the signals and I've even seen one signal with two names in a single data sheet. I'll use signal names that I find descriptive, but you should plan on a little data sheet spelunking for your panel.

A graphic LCD panel is a rectangular array of dots similar to Figure 1. Generally the grid is arranged with the larger number of dots horizontally, so an array with 640 dots along each of 200 rows is known as a 640x200 panel. Firmware can be used to control the "vertical," so it's reasonably easy to produce a portrait-mode display.

The dots themselves may be rectangular, but many of the more recent panels sport square dots equally spaced in rows and columns. The small gap between each pair of dots is inactive; unlike CRT pixels, each dot is clear and distinct. Figure 2 shows the dot dimensions for the Optrex DMF65 1 640x200 and the Matsushita EDM LG64AA44D 640x400 panels.

The panels we'll use for this project are all binary: each dot is either

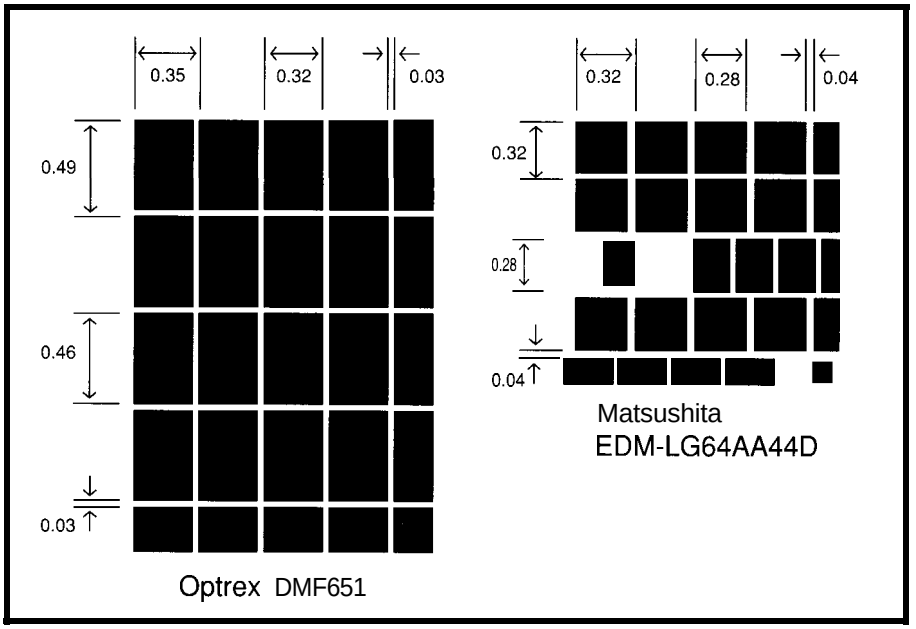



Figure *2-Jhe width of the transparent row and column electrodes determines the shape of the* dots. *Because the electrodes must be isolated from each other, every pixel is surrounded by an inactive border. Recent panels, typically those wifh 400 rows or more, have square dots. This simplifies life for graphic programmers by making circles round and squares square.*

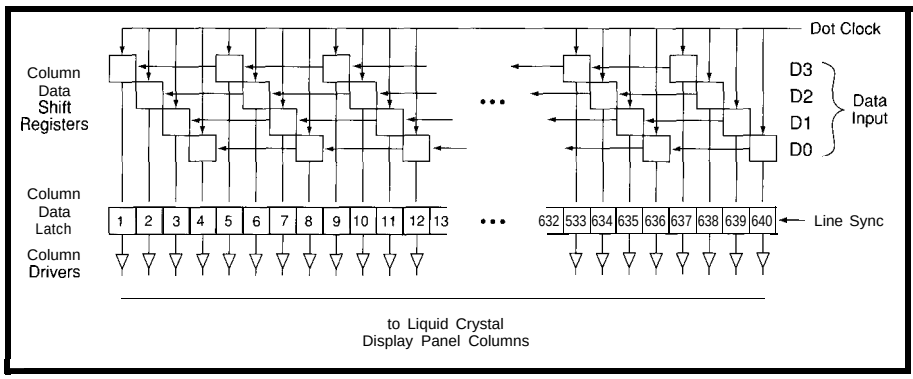



Figure 3—*A shift* register *accumulates 160* groups, each *of which is composed of four bits, to drive all 640 dots in a sing/e row simultaneously. The Dot Clock sets the basic timing for the entire panel as all the other signals are* defined in *terms of ifs transitions. The* Line Sync *pulse transfers data from the* shift *register to a 640-bit parallel* latch *that holds the output stable while the next 640 bits are shifting in.*

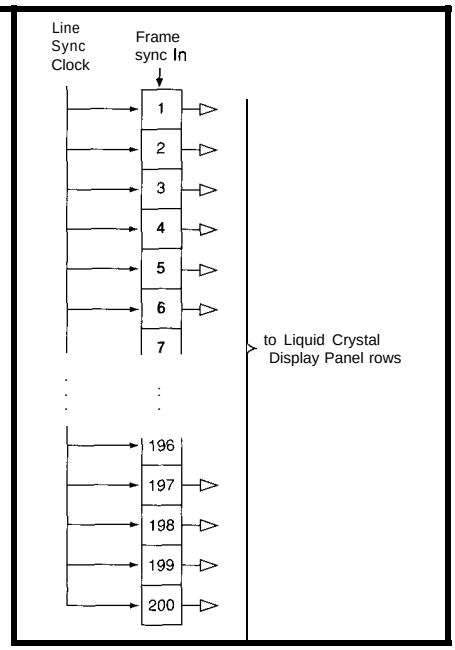



Figure *4-The outputs of a ZOO-bit shiff register are used to drive the LCD pane/ rows. The Frame Sync pulse passes from one bit to the next on each Line Sync pulse, so only one row can be active at any fime. Each row is driven for 1/200 of the total frame fime, or about 80 µs in each 16 ms. The jolt* applied to the liquid crystal *material turns fhe dof on within that 80 µs and if gradually goes off during* the next 60,000 *µs. The trick is to refresh the dof often enough that flicker isn't visible.*

completely on or off. Newer VGA-resolution 640x480 monochrome and color panels sport continuous tones, but driving those is a whole 'nother subject. For now, each dot represents a single bit of information. A DMF651 is capable of displaying 128,000 bits. That is a nice round decimal number, and is not *equal* to 128 kilobits = 128x1024 = 131,072 bits.

A binary "1" bit may make the dot transparent or opaque, depending on the liquid crystal chemistry. We have complete control over the data, so it's an easy matter to present dark data on a light background or vice versa. Your taste may vary, but either way is just a NOT away.

Contrary to popular opinion, and despite what your eyes tell you, all those bits are not on at the same time. You cannot just write 128,000 bits into the panel and go about your business, because the panel doesn't have a frame memory. You must send the same bits to the panel at least 60 times a second to get a stable, flicker-free image.

A BBS message appeared a while ago from someone who tried to drive a graphic LCD directly from an 803 l's output bits. Once we went over the code's timing, he realized what was wrong: He was refreshing the display at about 3 Hz! Needless to say, that is a little slow. An 8031 just can't supply four bits every 500 µs, so some external hardware is clearly in order.

That, of course, is precisely what an "external controller" does. It manages the control signals, refreshes the panel data, and provides a nice CPU interface. There are two catches: any given controller can handle only a subset of the panels out there, and they all come in awkward surface-mount packages. What fun is that?

Anyhow, once you know the refresh rate, simple arithmetic gives you the bit rate: under 130 ns per dot. Unlike a CRT that only lights up one pixel at a time, the LCD interface accepts several bits at once. The DMF651 has a four-bit interface and clocking in four bits every 520 ns or so will suffice. Transferring the 640 bits in each row takes 160 clock cycles, so the whole frame of 200 lines requires exactly 32,000 Dot Clocks.

Rather than displaying each nybble as it arrives, the LCD panel accumulates them in a 4-bit-wide, 160-element shift register. When the register has all the bits for a single row, a Line Sync pulse transfers them to a 640-bit latch that controls the panel's column drivers. Figure 3 shows the connections for the beginning and end of this circuit.

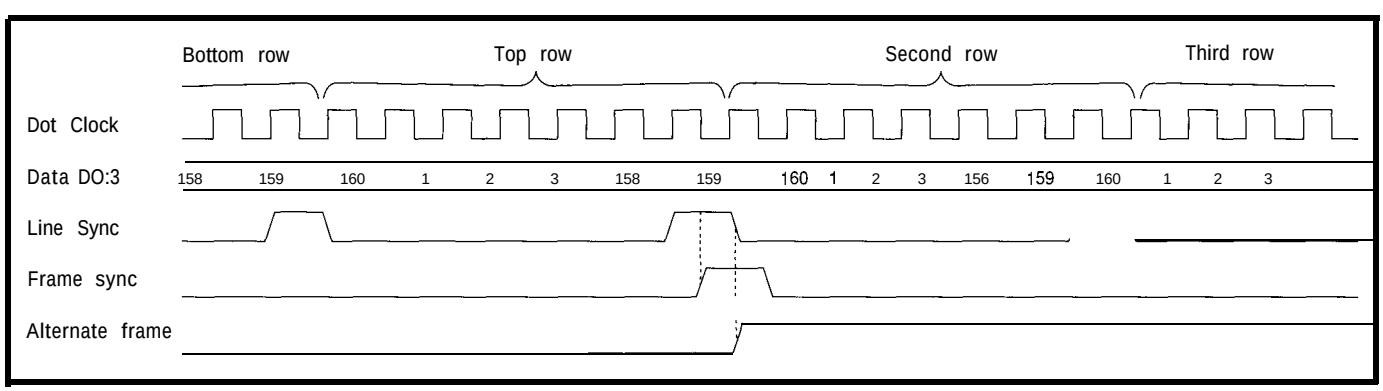



Figure 5—*Each* falling edge of the Dot Clock signal transfers *data* into the column data shift registers. If the Line *Sync signal is* high when Dot Clock goes low, the shift *register contents transfer to fhe column driver latches. The Line Sync signal a/so clocks the Frame Sync pulse through the row driver shift register as shown in Figure* 3. Note *that Frame Sync* is active *at fhe* end *of the first row, rather than at the beginning as you might expect from your experience with CRTs.*

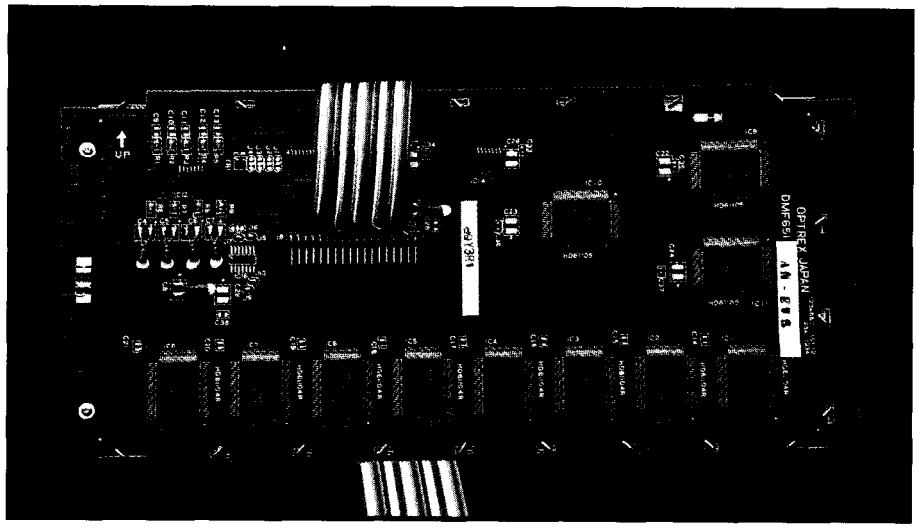


Photo I--The *Optrex DMF621 640×200 LCD panel uses eight Hitachi 61104 column driver chips and three H61105 row drivers.*

Because the dots in each row are displayed until the next row is ready, each dot has a duty cycle of 1/200 rather than 1/32000. As you can guess, it's hard enough to make a dot appear on when it's active only 0.5% of the time, let alone 0.003%. Hats off to the engineers who make LCD panels work at all, let alone as well as they do.

Row selection isn't done with a binary counter and decoder the way you might expect. The panels include another shift register, illustrated in Figure 4, behind the row drivers. This register is clocked by Line Sync pulses, and the Frame Sync pulse is just a data bit that's passed through all the flip-flops in the chain. Unlike the 640-bit

| | Optrex DMF651 | Matsushita LG64AA44D | Sharp LM64105T | Toshiba TLY-365-121 | Hitachi LM215XB | Epson EG7004 |
|---|---|---|---|---|---|---|
| Dots | 640x200 | 640x400 | 640x400 | 640x200 | 480x128 | 640x400 |
| Bits | 4 | 8 | 4 | 4 | 4 | 4 |
| Col Drivers | 640 | 640 | 640x2 | 640x2 | 240x2 | 640x2 |
| Row Drivers | 200 | 200x2 | 200 | 100x2 | 64x2 | 100x2 |
| Clk/row | 160 | 160 | 320 | 320 | 240 | 160 |
| Dot Clock | 480 ns | 480 ns | 240 ns | 480 ns | 960 ns | 240 ns |
| | | | | | | |
| Pin | | | | | | |
| 1 | Bezel Gnd | t5 v | Frame Sync | Bezel Gnd | D1 UL | t5 v |
| 2 | Line Sync | Bezel Ground | Line Sync | n/c | D2 LL | Ground |
| 3 | Dot Clock | Dot Clock | Dot Clock | Frame Sync | Frame Sync | Adj -13 V |
| 4 | Alt Frame | Enable | n/c | Line Sync | Alt Frame | Line Sync |
| 5 | -Contrast | Frame Sync | n/c | Dot Clock | Line Sync | Alt Frame |
| 6 | t5 v | Line Sync | t5 v | Ground | Dot Clock | Enable |
| 7 | Ground | Ground | Ground | D0 | D3 UR | Row Shift |
| 8 | n/c | n/c | Adj-21 V | D1 | D4 LR | Frame Sync |
| 9 | -23 V | DO upper | DO | D2 | t5 v | Dot Clock |
| 10 | D0 | D1 | D1 | D3 | Ground | Col Enable |
| 11 | D1 | D2 | D2 | Ground | -10 v | DO |
| 12 | D2 | D3 | D3 | +5 v | -Contrast | D1 |
| 13 | D3 | -22 v | | -Contrast | | D2 |
| 14 | n/c | -Contrast | | Adj -22.5 | | D3 |
| 15 | n/c | Ground | | Ground | | |
| 16 | n/c | DO lower | | | | |
| 17 | n/c | D1 | | | | |
| 18 | n/c | D2 | | | | |
| 19 | | D3 | | | | |
| 20 | | Ground | | | | |

Figure 6—*Each of these graphic LCD panels has unique electrical and connector specs. This table summarizes the important characteristics and pin functions for some of the panels in my stash. An "n/c" means a pin is not used, while a blank cell means the connector doesn't have that pin. A separate cable carries power to the backlight.*

shift register used for a row, there is no output latch because only one row is active at any time.

A Frame Sync pulse accompanying a Line Sync pulse marks the first display line. Because Line Sync pulses occur at the *end* of each line, the Frame Sync pulse occurs *after* the first row of data. Figure 5 shows the relationship between all the various pulses. It goes without saying that not all panels are alike, but that's the general idea.

The DMF65 1 requires an additional signal that alternates from frame to frame. To produce this signal, which I call Alternate Frame, you just toggle a flip-flop on each Frame Sync pulse. More recent panels generate the signal internally. I suspect this has more to do with the size of the driver IC packages than anything else: it's easy to add a flip-flop if you have a spare output pin, but hard to justify a whole IC for just one bit.

Using shift registers for the row and column logic allowed the engineers to split the circuitry into identical units, which is exactly the right tactic for LSI chip design. For example, the DMF65 1 shown in Photo 1 has eight Hitachi 61104 column driver chips and three H61105 row drivers. Each 61104 has 20 elements of the 4-bit column shift register and drives 80 columns. Each 61105 holds 80 row selection bits, so half of the last chip is unused.

The actual process of converting row and column selection bits into a turned-on dot is, mercifully, hidden by those LSI drivers. There are several different schemes that all boil down to applying a high voltage to the dots at the intersection of each active column with the currently selected row while not applying quite so much juice to all the other dots. Under the influence of that jolt, the liquid crystal compound twists the plane of polarization of the transmitted light and the dot becomes either opaque or transparent, depending on how the panel is set up.

Because of the high voltage needed to drive the chemistry, graphic LCD panels are not friendly 5-volt-only devices. In addition to the usual +5-V logic supply and ground, you must

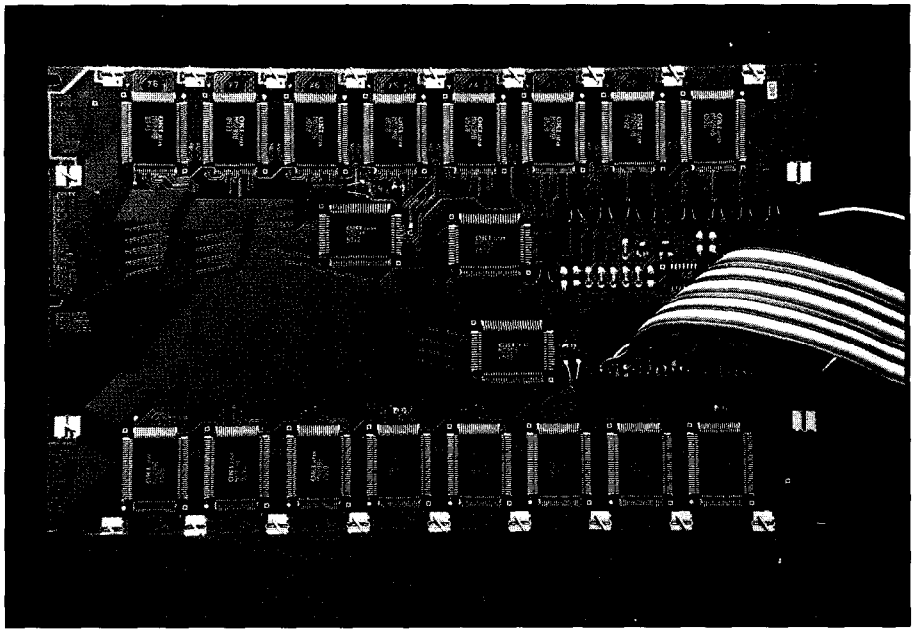


Photo *2-Compared to fhe 640x200 unit in Photo 1, Matsushita's EDM LG64AA44D 640x400 LCD panel uses an additional set of eight column drivers.*

provide an LCD drive voltage of -9 to -30 volts at about 25 mA. Because the liquid crystal's optical response is strongly temperature sensitive, the supply must be adjustable to control the display contrast. Some displays use a fixed LCD drive voltage with a separate contrast adjustment.

These LCD panels were intended for use in laptop computers or similar widgets, so they use CMOS logic to reduce power consumption. Their input logic level voltages make no concession to TTL drivers: $V_{IH}$ is typically $0.8V_{CC}$, which translates into 4.0 volts. One panel expects $V_{IH}$ levels exceeding $0.9V_{CC}$! You *must* drive the interface signals with HCT or HCTLS gates because ordinary LSTTL has a $V_{OH}$ specification of 2.4 V.

And, no, the panel's interface signals are not protected against high negative voltages. Despite the fact that the negative supply lines may be sandwiched between logic signals on the connector, you must not short those adjacent pins, not even once, not even when your scope probe slips. Word: buy two LCD panels and save on shipping....

Different panels differ on the order in which you must apply and remove the supply voltages and logic signals during startup and shutdown. The penalty for your failure to comply with these requirements can be death due to SCR latchup while the CMOS logic incinerates itself. If you're designing a specific panel into your project, you can meet its needs, but I don't know of

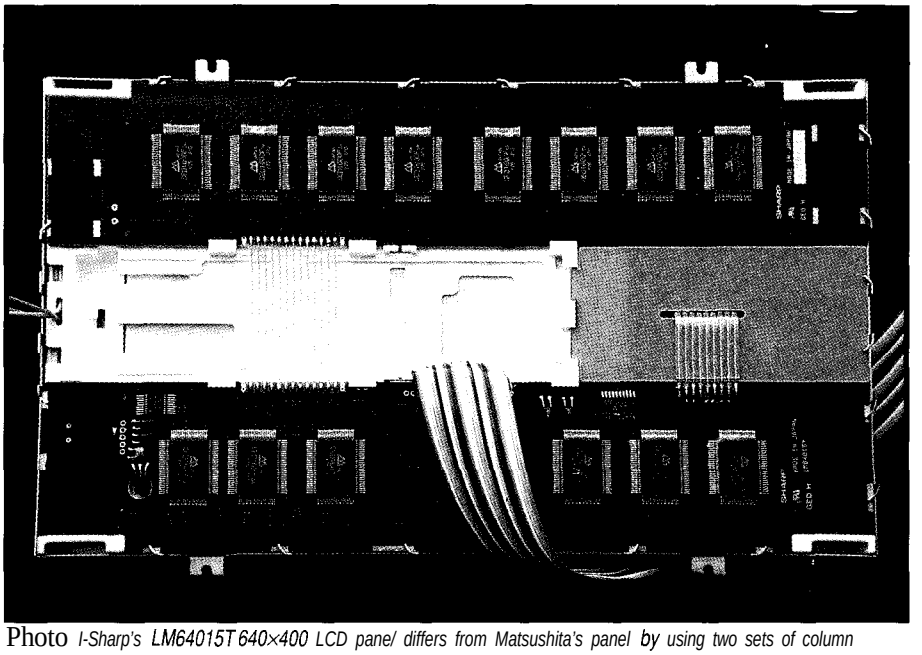


Photo *I-Sharp's LM64015T 640×400 LCD pane/ differs from Matsushita's panel by using two sets of column drivers connected in series rather than being driven in parallel by separate data inputs.*

a good general solution that will handle a variety of panels. The FDB circuit has a relay to disconnect the LCD drive voltages when the '386SX is reset, but I know that doesn't meet all the specs. So far, though, so good.

Not only is there no agreement on signal names, voltages, or power sequencing, but each LCD panel sports a unique connector as well. Figure 6 summarizes what I know about some of the panels I've tested, along with my pin name translations. The Graphic LCD Interface uses a 2x13 ribbon cable header that doesn't match any of the panels, but I'm pretty good at soldering wires to panel connectors. Don't waste your time trying to find the mating connectors!

## SEEING THE BIG PICTURE

The circuitry I'll present next month is, perhaps, an example of retro-design: the "right" way to do it nowadays is with the exact LSI controller for the panel you're using. But this column shows you how things work, so I don't feel too bad about using a handful of TTL chips and some firmware to illustrate the key points.

With that in mind, Figure 7 is the overall Graphic LCD Interface block diagram. The PC sees the interface as a 32K-byte block of RAM and an output port. The LCD panel sees it as four data bits and the control signals. I'll start with the LCD side because it determines how the PC side works.

Displaying one 640x200 frame requires 32,000 clock cycles. Even

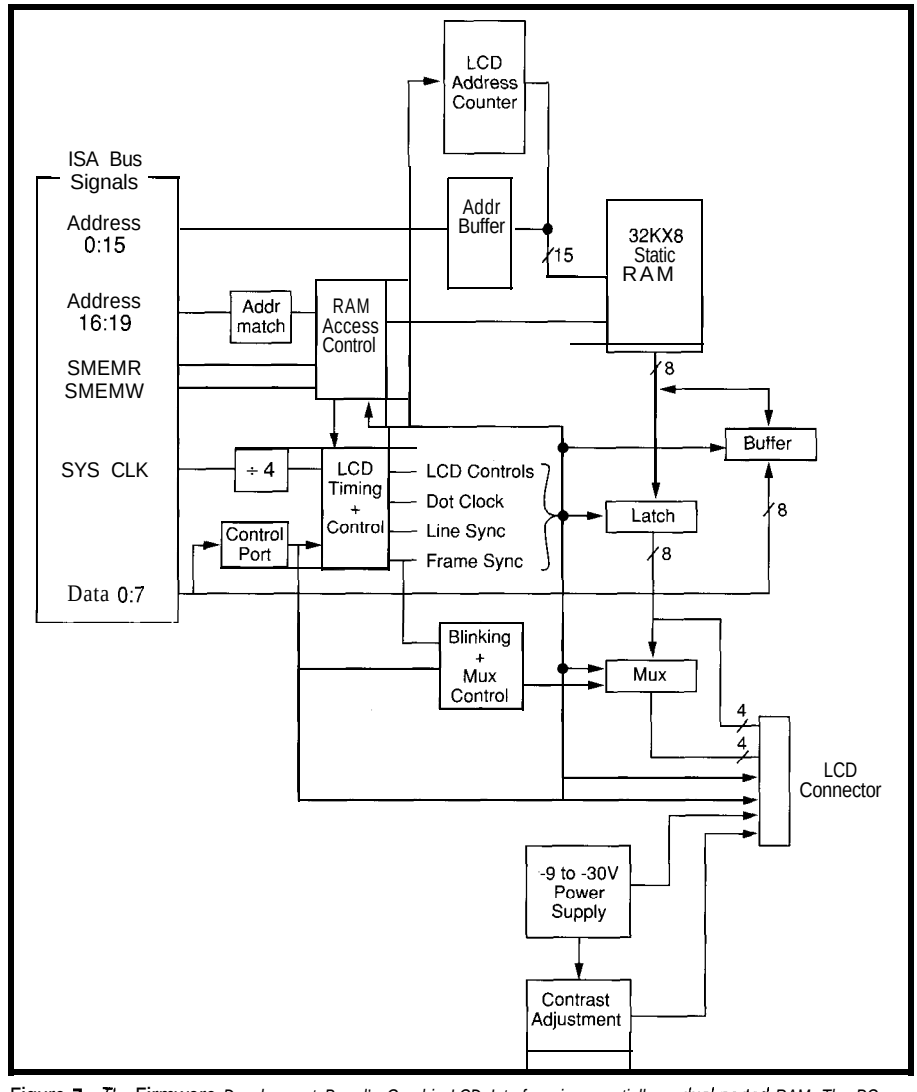



Figure 7—*The* Firmware *Development Board's Graphic LCD Interface is essentially a dual-ported RAM. The PC can read and write data while the LCD pane/ is displaying dots fetched from the same location. The two sides of the* interface use *alternate* clock *phases so* the *accesses aren't really* simultaneous. *The output multiplexer* and *blink* logic include additional circuitry (and jumpers!) *to* support a variety of LCD *panels.*

though each cycle transfers only four bits, a 32K-byte (32,768 in decimal) static RAM is an obvious choice. If we store the dots in the low-order nybble of each byte, a simple 15-bit address counter will access them in the right order.

The period of the LCD Dot Clock must be about 520 ns to refresh the display at 60 Hz. A key part of this design fell into place when I realized a 480-ns Dot Clock would provide 65-Hz refresh. It turns out 480 ns is an ISA-bus magic number since it can be easily derived by dividing the 120-ns SysClk signal by four.

Most LCD panels specify a refresh rate from 60 to 80 Hz with minimum Dot Clock periods in the 160- to 330-ns range. As long as the refresh rate is high enough to prevent flicker and the Dot Clock period is above the specified minimum, you can run the panel at any speed you like. We already know 3 Hz falls well outside those specs, so we should try to go faster than that.

However, if the RAM runs at 480 ns, the PC has no time to write or read the data. The Graphic LCD Interface uses both 240-ns clock phases to allow simultaneous access by both the PC and the LCD panel. When Dot Clock is high, the PC can access the RAM through the address and data buffers. When it's low, the LCD has full access for its address counters and latch.

Because the LCD panel needs stable data throughout its cycle, Dot Clock's rising edge captures the RAM's output in a '374 latch. Figure 5

shows that the data is transferred into the panel on the next Dot Clock falling edge, so the signal has nearly 240 ns of setup and hold times. The latch holds the data while the RAM is busy with PC accesses.

Although the DMF65 1 needs only four data bits, it seems a shame to waste half the RAM. A multiplexer after the '374 data latch selects either the high or low nybble under control of a signal from the Blinking and MUX Control logic. That signal is one of the outputs of an 8-bit counter driven by Frame Sync, so the multiplexer switches every 1/16 to 4 seconds.

A little firmware can thus implement nearly any blinking scheme you'd like because the two nybbles are entirely independent. If you fill the high nybble with zeros, the on dots in the low nybble blink. Fill it with ones to get a blinking background while the data dots remain on. Duplicate the low nybble in the high nybble, complement the bits, and you get a blinking reverse image. Versatile enough?

The overall blink rate is under firmware control because different panels have different response times. The blink rate of 1/16 of a second is faster than any panel I have available, while a 4-second blink is glacial enough for a nearly frozen panel. You can also vary the blink rate to attract attention; the only restriction is that all the dots blink at the same rate...unless you get tricky and rewrite the RAM on the fly.

All of this is easy because dots on the LCD are just bits in PC memory. The RAM appears in the PC's address space at D000:0000, just after the battery-backed RAM at C800:0000 through C800:7FFF. That circuit appeared in Issue 37 and I discussed some of the problems involved in ISA bus memory in Issues 36.

The RAM Access Control Circuitry synchronizes the Dot Clock with the bus signals during each memory access, so the RAM is always in the right state by the time the ISA bus cycle finishes. Basically, the bus is slow enough that we can pull a fast one on it. A 120-ns RAM is quick enough for this application, so we don't need exotic hardware, either.

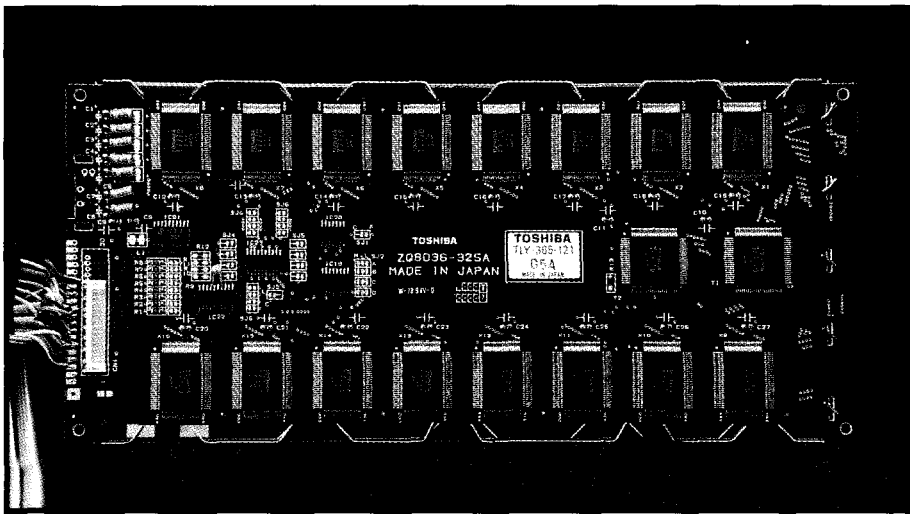



Photo 4-The Toshiba TLY-365-121640×200 panel uses two sets of column driver chips and three row drivers. It is actually a 1280x100 pane/ with fhe two halves of each row stacked atop each other.

In desktop PCs, the 64K between D000:0000 and D000:FFFF is often used for an Expanded Memory page frame. Address space is a precious commodity below the l-megabyte line, so EMS boards (or EMM386 programs) use that 64K as a window into the megabytes of expanded RAM. You can also move the LCD buffer to A000:0000 or B000:0000 if your system doesn't have a video card at one of those locations.

## VARIATIONS ON A THEME

If a 640x200 panel doesn't have enough dots for you, the next step up is 640x400. Because VGA resolution is 640x480 and everyone simply *must* keep up with that standard, you'll find 400-line displays at reasonable prices. In fact, I suspect you could go into OEM production with just the surplus panels from last-generation portables.

Doubling the number of lines doubles the number of dots on the panel. The refresh rate must remain about the same to avoid flicker, so we must send twice as many bits in the same amount of time. There are two ways to do this: double the number of bits per Dot Clock cycle, or double the Dot Clock frequency. You will find panels using either method.

Photo 2 shows the back of a Matsushita EDM LG64AA44D 640x400 panel. Comparing this with Photo 1 shows an additional set of eight column drivers. Each set handles four bits, so this panel accepts eight bits in each Dot Clock cycle.

There are still only three row driver chips with each of the 200 outputs connected to two rows. Figure 8 sketches the layout: the display is split in half, with one set of column drivers for each section. Two rows are active at once, but the column drivers present different data to each row.

Because the LG64AA44D requires eight bits on each cycle, the Graphic LCD Interface's data multiplexer isn't used. The Blinking and MUX Control logic emits a constant zero to route the low-order nybble from the '374 latch to the LCD connector. The high-order nybble is wired directly from the latch to the connector, so the LCD sees all eight bits simultaneously.

Bits zero through three are displayed on the upper half of the panel, while bits four through seven appear on the lower half. In effect, the LG64AA44D is just two 640x200 panels on the same piece of glass. The firmware must account for the fact that the bits in one byte will appear in two widely separated locations.

The nice part about this display is that the additional dots are basically free: you already have a byte-wide RAM, so all it takes is a little firmware to plunk the dots in the right spots. In fact, if you write the code correctly, you can use either 200 or 400 line panels. Just don't try to blink the big display!

In contrast, the Sharp LM64015T 640x400 panel shown in Photo 3 uses the other technique: it accepts four bits on each double-speed 240-ns Dot Clock cycle. The two sets of column drivers are connected in series rather than being driven in parallel by separate data inputs. It's essentially the same hardware as the LG64AA44D, but it must run faster to keep up with the data.

Figure 9 sketches the connections for this panel. In effect, it is a 1280× 200 panel whacked in half, with the two pieces stacked atop each other. Each row is 320 double-speed Dot Clocks long, but the second half of each row appears on the bottom of the screen.

The Blinking and MUX Control logic switches the multiplexer between the nybbles on each half of the Dot Clock cycle. Because the panel accepts one nybble per half-cycle, the data bits are actually contiguous on the screen. Bits four through seven are presented first, followed by bits zero through three. It's easy to write graphic routines for this panel!

The Graphic LCD Interface RAM circuitry runs at the same speed as before, fetching and latching a new

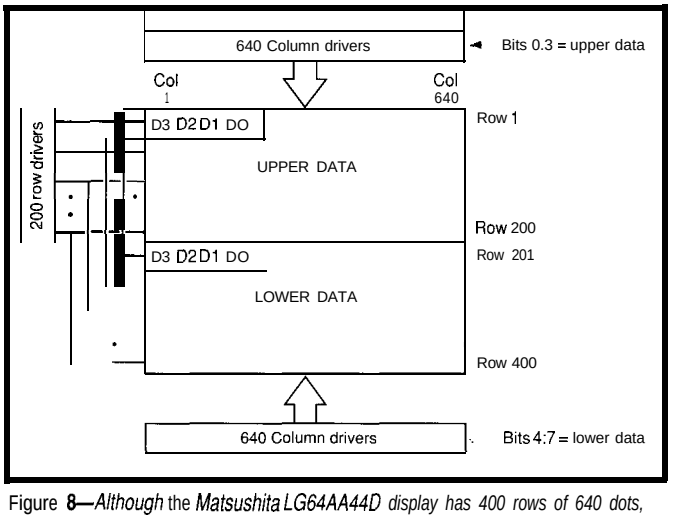



Figure 8—*Although* the *Matsushita LG64AA44D display has 400 rows of 640 dots, there are only 200 row drivers. Each row driver activates two rows, one in each ha/f of the display. One set of eight column drivers displays dots in the upper* half, while *an identical sef drives the lower ha/f. Because each set of column drivers handles four bits, the pane/ accepts eight bits on each 480-ns Dot Clock cycle.*

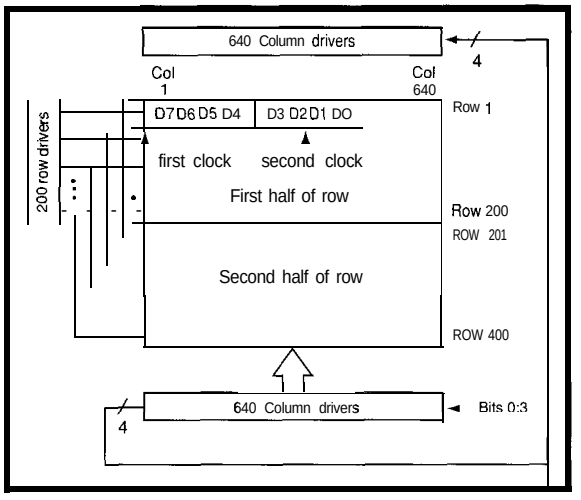



Figure 9—*The Sharp LM64015T has 400 rows of 640 dots, but it uses two sets of column shift registers chained together to hold 1280 bits. Each of the 200 row drivers activates two physical dot rows, but, unlike the Matsushita panel shown in Figure 8, they show two halves of a single logical row. Each double row requires 320 cycles of a 240-ns Dot Clock, transferring 1280 bits four at a time. The Toshiba TLY-365-121 mentioned in the text has a similar layout with 100 row drivers, 200 rows, and a 480-ns Dot Clock.*

byte every 480 ns. The trick is the multiplexer, which presents each nybble for 240 ns. The panel runs from a double-speed Dot Clock, but that doesn't affect the rest of the interface.

The LM64015T uses a blindingly bright, cold-cathode fluorescent tube backlight which needs about 1 kV to fire up. Although unlit electroluminescent panels are dim, this one is completely useless without the light. I don't know a surplus source for the special CCFT inverters, and a new inverter costs about as much as the surplus panel. It's a nice panel if you can use it....

However, 400-line panels are just a recent branch on the LCD evolutionary tree. Photo 4 shows the back of a Toshiba TLY-365-121 640x200 panel with two sets of column driver chips and three row drivers. It clocks four bits every 480 ns, but each line is 320 clocks long. This is a 1280x100 panel with the two halves of each row stacked atop each other.

The panel connections are similar to Figure 9 with 100 row drivers and 200 display rows. You can think of this as a double-speed version of a 960-ns 640x100 panel, but 1 doubt if one of those was ever made because the aspect ratio is so weird. In any event, the Graphic LCD Interface can handle this panel with no problem.

I've also looked at the Hitachi LM215, a 480x128 display sporting *four* sets of column drivers clocked at 960 ns. Each of the four data bits paints one quadrant, which makes drawing anything an intricate bit twiddling exercise. The Graphic LCD Interface can handle this one with an additional flip-flop to cut the 480-ns Dot Clock rate down to size. Be sure to store duplicate data in successive even/odd bytes so the output latch is stable for the entire 960-ns cycle.

The Epson EG7004 640x200 and EG2401 256x64 panels need an extra set of clock signals to route sync pulses through the shift registers. Think of these as a challenge; there are much easier panels to work with, even as surplus items.

## RELEASE NOTES

One thing is certain: more, bigger, and better graphic LCD panels will show up for your perusal as the laptop computer market churns along. As 1 was writing this column, I couldn't resist a Sharp 640x480 panel for $35. If it's easier to use than its ancestors, I'll modify the Graphic LCD interface to provide those extra bits.

Quite uncharacteristically, there are no BBS files this month. However, there's been a lot of interest in this topic so dial in with questions or comments.. .and stay synched for the hardware next month!

## SERIOUS BUSINESS

My Jensen tool kit is an essential carry-on whenever I visit my parents: there's always *something* that needs fixing. On my latest trip, I mortised a replacement doorknob, updated an electrical outlet that was older than I am, and tried to rehabilitate an ancient adding machine. There are, however, some things that I just can't fix. On the last day of September, Mom and 1 held Dad's hands when he died at home.

Dad would have been 84 on New Year's Day, but there is yet no cure for pancreatic cancer. He refused exotic treatments, saying "Why take a long and rocky road to reach the same destination?" Shortly after getting the diagnosis, they visited us just so Dad could be sure everything was in order at our new house. He spent his last months finishing projects and visiting-one last time-many of his friends and relations.

Folks, each of you knows someone you ought to call *right now*. Later may be too late to swap stories, apologize for stupidities, or just say "I love you." Trust me on this. If it is at all possible, visit them now, give them lots of hugs, leave nothing unsaid or undone.

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

## SOURCES

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

Timeline carries many different LCD panels; check their ad in this issue. (310) 784-5488, fax: (310) 784-7590.

Marlin P. Jones has a wide variety of graphic LCD panels as well as character units with built-in controllers. They sometimes have electroluminescent backlight inverters, but they tend to sell out quickly. (407) 848-8236, fax: (407) 844-8764.

All Electronics has the occasional LCD panel and many of the electronic parts you'll need for projects like this. (818) 904-0524, fax: (818) 781-2653.

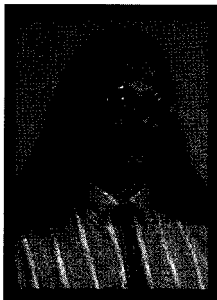
## I R S

413 Very Useful
414 Moderately Useful
415 Not Useful

# Temperature Sensor Eludes Analog Interfacing

## FROM THE BENCH

**Jeff Bachiochi**



Are you in need of a thermostat that is analog-free? Follow Jeff's lead as he introduces you to Dallas's DS1620 Thermometer and Thermostat. The name of the game is solid state with the help of an EEPROM.

As I write this, I am enjoying the best time of year to go hiking in New England. The peak of the foliage season yields a terrain of patchwork colors. If you are fortunate enough to catch this view from atop a mountain or even a high hill, you know how inspiring it can be. Add to this picture a crisp blue sky scattered with wisps of cottony clouds and an autumn sunset as it slowly unfolds into a psychedelic overload for your eyes.

As the last bits of summer fall from the trees, we rake the pleasant memories of warmer weather into small piles. Oil delivery trucks have come out of summer's hibernation, which can only mean the heating season has started. Thermostats are being turned up all across the forty-first parallel.

Conventional thermostats are simple devices. A mercury-filled tilt switch is rocked back and forth by the expansion and contraction of a bimetallic coil. When the coil contracts because of cooling temperatures, it tilts the mercury toward awaiting contacts, completing a circuit which requests heat from the furnace. Simple and dependable, but certainly has no place in today's Intelligent House.

I bought one of those digital thermostats years ago. I threw it away years ago. I didn't like it. I needed the instruction book every time I wanted to change the setpoints. The backup battery lasted five minutes less than the duration of every power outage. And twice a year, at Daylight Saving Time/Standard Time changes, it had to be reprogrammed. This all took place before I added on to our cozy little cape. Now I have three zones of heating. I'm considering breaking these three into a few more individually controlled zones for the selective comfort of each family member. Along with each added zone comes the necessity for an additional thermostat.

Up to this point if you wanted to measure temperature electronically, it required biasing and gain circuitry designed to match the sensor's analog output to an A/D converter. The digitized data could then be used to calculate the actual temperature. This is all changing, and Dallas Semiconductor is providing the instrument of change: the DS 1620.

### TEMPERA-DIGITATION

Dallas has packed a solid-state temperature sensor with an EEPROM and some comparators into a single package to form a stand-alone, solid-state thermostat. A 3-wire interface allows both high and low setpoints to be programmed into the internal EEPROM. Temperatures above the high setpoint turn on the $T_{HIGH}$ output, while temperatures below the low

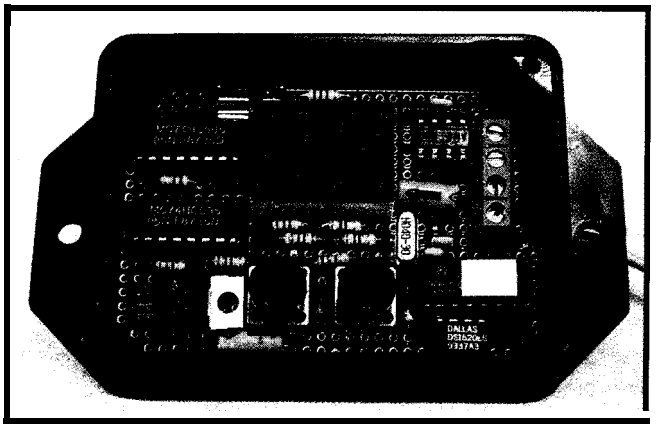


Photo 1--Temperature measurement and control is possible using Dallas Semiconductor's DS1620 thermometer/thermostat chip. A two-digit display and other features may be added by including a small, dedicated microcontroller.

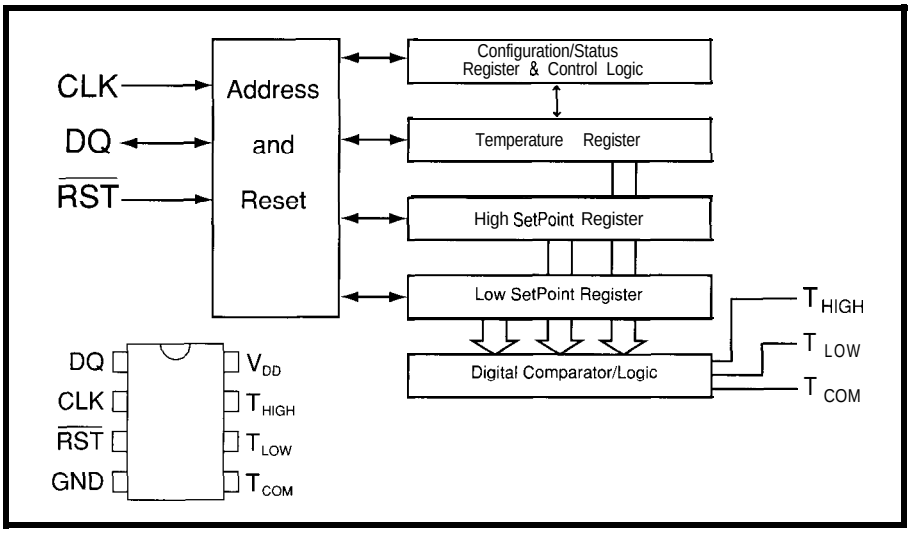



Figure 1—*Three* registers in the *DS1620 determine the mode of operation* and *establish the temperature at which an* action *will take place when compared to an* ambient *temperature.*

setpoint turn on the $T_{LOW}$ output. A third output, $T_{COM}$, turns on when temperatures drop below the low setpoint but remains on until temperatures rise above the high setpoint. Here, the upper and lower setpoints act like an adjustable hysteresis band. The EEPROM retains the setpoints

| Temp | Binary Value | | Hex |
|---|---|---|---|
| +125°C | 00000000 | 11111010 | 00 FA |
| +25°C | 00000000 | 00110010 | 00 32 |
| +0.5°C | 00000000 | 00000001 | 00 01 |
| 0°C | 00000000 | 00000000 | 00 00 |
| -05°C | 00000001 | 11111111 | 01 FF |
| -25°C | 00000001 | 11001110 | 01 CE |
| -55°C | 00000001 | 10010010 | 01 92 |
| | MSB LSB | MSB LSB | |
| | MSByte | LSByte | |

Table 1—*The DS1620* puts out *8-bit conversion values plus a ninth sign bit.*

even in the event of power failure. Figure 1 shows a block diagram and pinout of the device.

Temperature is presented in a two-byte format. The most-significant byte holds the sign bit (actually only the low bit of that byte is used] and the least-significant byte holds the digital value of the actual temperature. The output resolution of this device is in 05°C increments from -55°C to +125°C. If the sign bit is set, then the temperature is negative and the least-significant byte's value is presented as the two's complement. For Fahrenheit measurements, you can use a lookup table or crunch the samples through the ever-popular (⅘C)+ 32 conversion factor. Table 1 shows a sampling of temperature readings along with the data generated by the DS1620.

The DS1620 has three EEPROM registers, a configuration/status register, and two setpoint registers. The two setpoint registers (High and Low) establish the temperature at which some action will take place when compared with the ambient temperature. The setpoint temperature in each of these two-byte registers takes the same form as the two-byte temperature values above.

The configuration/status register determines the mode of operation. Only the two least-significant bits of the byte-wide status register are used to set the mode, however an additional two bits are used to reset flags. Some of the particulars of this register are shown in Table 2. When the status register is read, it returns the current mode of operation and a few extra bits of information.

## STAND-ALONE MODE

The DS 1620 can free run, continuously converting and comparing the temperature while only drawing 1 mA at 5 volts. In one-shot mode, after a conversion is complete, current consumption dips to 1 µA until the next conversion is initiated by dropping the CLK/*CONV line. As long as power is applied and the THF and TLF flags are not reset through the configuration register, the contents of these two flags will continue to indicate if the temperature has ever passed either of the setpoints. This makes an excellent yet simple temperature limit exception logger.

Once the initial setpoints are programmed (through the 3-wire communication bus), the DS 1620 will post vigil as a full-time thermostat and only needs a relay (mechanical or solid state) to control a fan, valve, or furnace.

## 3-WIRE COMMUNICATION MODE

Although the "set it and forget it" mode has many applications, most of us would find this a bit inflexible for use with our home heating system. The 3-wire communication mode allows control over the device through just 3 bits. The bidirectional bit (DQ) provides a data path in both directions

Write Confiauration Byte (x=don't care)
Bit Position 7 6 5 4 3 2 1 0
x THF TLF x x x CPU 1Shot

THF=O (reset High Temperature Flag)
THF=0 (reset Low Temperature Flag)

CPU=1 (using 3-wire comm mode DQ, CLK, *RST)
CPU=0 (when ● RST=O, CLK=O initiates conversion[s])

1Shot=1 (perform single conversion)
1Shot=0 (perform continuous conversions)

Read Confiauration Byte (x=don't care)
Bit Position 7 6 5 4 3 2 1 0
Done THF TLF x x x CPU 1Shot

Done=1 (Conversion complete)
Done=0 (Conversion in progress)

THF=l (Temperature exceeds High Setpoint)
THF=O (Temperature does not exceed High Setpoint)

TLF=1 (Temperature below Low Setpoint)
TLF=O (Temperature is not below Low Setpoint)

CPU reflects last value written
1 Shot reflects last value written

Table 2—*The configuration/status register determines the mode of operation of the DS1620. Bit positions 0 and 1 are* used to set *the mode while bit positions 5 and 6 are used to reset flags.*

and remains tristated whenever the ● RST input is low. Taking 'RST high initiates a transfer in which the DS1620 accepts data on the first eight rising CLK cycles it sees. This first byte of data sent to the DS1620 will determine what happens next. There are nine possible command bytes, as shown in Table 3. You can examine this along with the 3-wire protocol that is illustrated in Figure 2.

Table 3—*The first byte of data sent to the DS1620 determines what happens next. The part supports a total of nine commands.*

| Cmd Byte | Next Action |
|---|---|
| 01H | Next 9 bits received go to TH registers |
| 02H | Next 9 bits received go to TL registers |
| OCH | Next 8 bits received go to configuration register |
| 22H | Stop conversion; registers are transmitted |
| A1H | Next 9 bits come from TH registers and are transmitted |
| A2H | Next 9 bits come from TL registers and are transmitted |
| AAH | Next 9 bits come from temp conv and are transmitted |
| ACH | Next 8 bits come from config register and are transmitted |
| EEH | Start conversion |

## TEMPERATURE CONVERSION

A single temperature conversion requires about one second. If one-shot conversion is being used, the status register must be checked to assure the conversion has completed before requesting the convered temperature. If the DS1620 is in continuous conversion mode, the last temperature converted is sent whenever the temperature is requested.

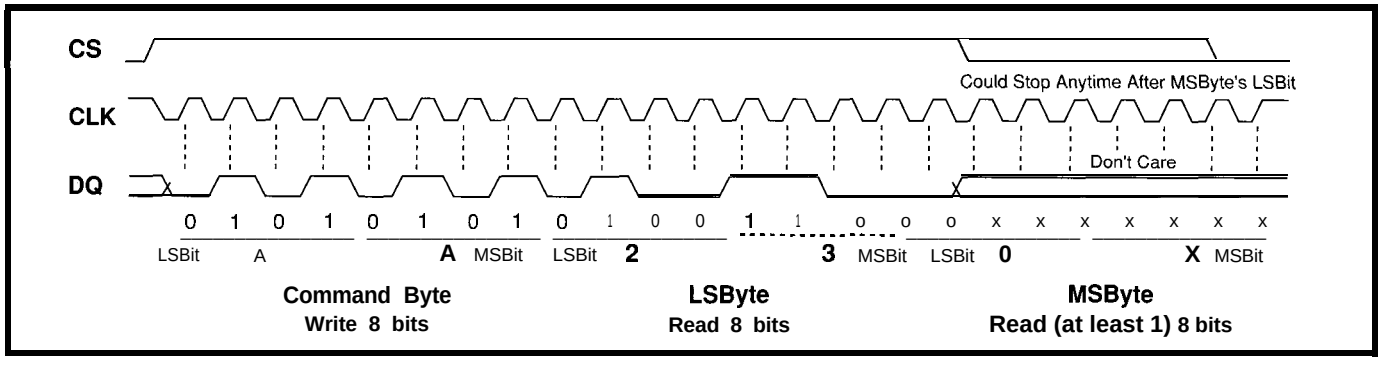



Figure 2-The *timing* diagram *reveals a* command *byte where the next 9 bits come from Temperature Conversion and are transmitted.*

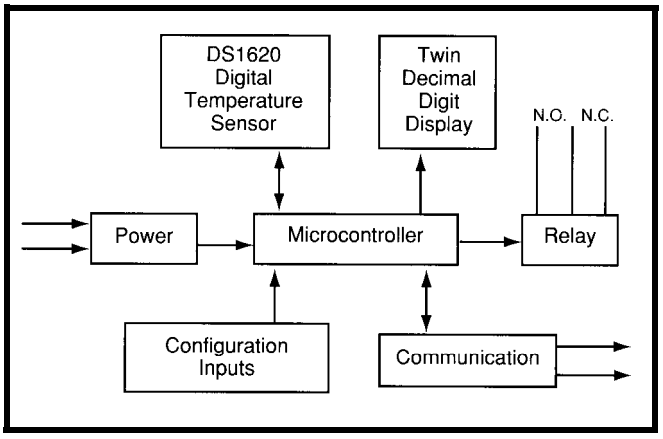



Figure 3—*By adding a dedicated microcontroller, the DS1620 can be the basis for a very* flexible digital thermostat.

Temperature conversion accuracy is ±0.5°C from 0°C to 70°C (32°F to 158°F). The error increases to ±1°C down to -40°C (-40°F) and up to 85°C (185°F), and to ±2°C down to –55°C (-67°F) and up to 125°C (257°F).

## FURNACES UNDER FIRMWARE CONTROL (SORRY, ED)

To satisfy my requirements for a zone thermostat, I want to know what the temperature is and then be able to adjust it up or down from some programmed norm. Since I want the setpoints to be software settable and automatically adjusted by local demand or based on time of day and day of week, this information needs to come from not only the local thermostat control, but also from the central control. The thermostat needs to measure the temperature, display it, and report it when asked. In addition, control inputs will allow local temporary adjustment of the zone's setpoint. For the time being, local adjustments will be canceled at the next automatic setpoint change. However, the software could allow these local changes to permanently alter the active setpoint. This will allow the system to be very flexible. See Figure 3 for the thermostat block diagram.

The local temperature can be displayed in degrees Celsius or Fahrenheit, which means indoor temperatures should not fall below 0°C (32°F) or rise above 37°C (99°F). Using these limits will require only two display digits. Two 74HC595 serial-to-parallel converters will directly drive seven-segment digits to adequate brightness. Gnly three output bits are needed for this: data, clock, and register latch enable. This allows the same CLK and DQ lines for DS-1620 communication to be used with the display.

Digit data comes from a lookup table and is presented serially through the DDATA bit in least-significant-bit-first format, wherein the unit digit is followed by the ten's digit. The unit digit's data is shifted through the first '595 and into the second. When all 16 bits are clocked in, the double word is latched into the output drivers, which illuminate the proper segments to form the digits.

In an attempt to keep costs down, each local zone monitor must be simple, but sophisticated enough to monitor, display, and communicate. I chose RS-485 for the communication medium because it supports multiple drops, is noise immune, and is capable of long distances.

Next month, I will demonstrate how an inexpensive micro can run the local show, yet still be a slave to your home. ❑

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

## CONTACT

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

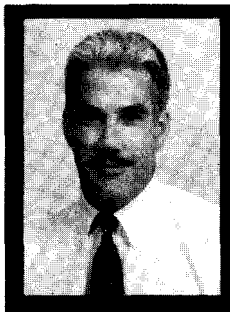
## I R S

*416* Very Useful
417 Moderately Useful
418 Not Useful

# PID-Pong Challenge

# SILICON UPDATE

**Tom Cantrell**



Ever been impressed by the local department store's display of the beachball twirling in the air above a fan? Not one to be outdone, Tom goes a step further with his own high-tech display.

On the one hand, the subject of PID (Proportional, Integral, Derivative) control is potentially very interesting. After all, "control" is what most interesting micro applications are all about.

Nevertheless, I'm usually disappointed by most of the articles I read about the subject. The typical academic treatise-complete with lots of math proofs, step/response curves and Z-plots-leaves a lot to be desired as far as I'm concerned. Notably, borrowing a phrase from those skeptical midwesterners, I want someone to "show me" a real-world PID example that I can get my hands around and fiddle with.

Trade shows are known for elaborate demos, and I've seen a number of working PID setups over the years. Yet, often even these fail to excite.

The best I've actually encountered is the ubiquitous "floating ball" gadget consisting of a steel ball levitated by judicious control of an electromagnet. The last time I saw such a setup, it was under the control of about $20k worth of VME big-iron. I didn't have the heart to tell the proud boothers how Jeff did the same thing on a lowly 80C52 ("Magnetic Levitation: An Example in Closed Loop Control," *Circuit Cellar INK*, issue 18)—and in BASIC, no less.

While nifty, the floating ball setup fails the first rule of trade show demos-it should do something visually interesting. After all, the only action is when the controller-and ball-both crash.

I've seen other questionable efforts, such as the aquarium-like arrangement that purported to illustrate the classic water tank problem that haunts introductory calculus students. You know, tune the inflow depending on the outflow in order to maintain a certain level. Unfortunately, watching water trickle is about as interesting as watching it boil. Worse, my booth visit was cut short by a sudden urge to find a restroom.

Unfortunately, the vast majority of real-world control applications fail the "show me" requirement-often because the speed or magnitude of the control lies outside the realm of our senses. "Ladies and gentlemen, watch how the motor servo keeps the platter spinning at 3000 RPM"-see what I mean?

## BRAINSTORMING

I won't bore you with all the Rube Goldberg-like lashups that filled my dreams night after night as I pondered the problem. I don't know what it is about Silicon Valley that encourages simultaneous flashes of genius mixed with madness. Consider U.C. Berkeley (a.k.a., Berserkly) which is both a proud parent of RISC and home to the Naked Guy!

However, with dawn and a cup of coffee, most of the more elaborate ideas slunk back into my subconscious where they rightly belonged. The fact is, the gizmo was subject to various constraints including cost, ease of assembly, no open flames (darn!), and so forth.

Finally, I came up with the idea for a "PID-Pong" machine consisting of a 3-foot-long piece of 2" diameter, clear plastic tube, a *12-V* muffin fan and a ping-pong ball. The central idea being for the controller to adjust the fan power thereby moving the ping-pong ball between setpoints.

The critical challenge involved sensing the ball position. I considered lining the tube with LEDs and detectors, but dispatched that idea for reasons of poor resolution and lack of ease of assembly (with the former directly impacting the latter, i.e., more LEDs means more chances for me to goof up).

In a Naked Guy-like flash of inspiration, I stripped the problem to its essence and realized a TI01 ultrasonic position sensor (typically used in autofocus cameras and electronic tape measures) provided a potentially great solution. If you're not familiar with these little wonders, they are really just like a sonar except the familiar "ping" of submarine dramas is boosted into the ultrasonic (49.4 kHz) range. By measuring the time between the ping and its echo, distance is easily derived based on the speed of sound.

I had no idea whether or not the configuration would work, fearing acoustic and/or electrical interference might prove troublesome.

I could have tried to prove the feasibility *a priori* but-hacking being the better part of valor-decided the best way to find out was just to put it together (Figure 1 and Photo 1) and see what happens. After all, I had nothing to lose but a few shredded ping-pong balls and, last I heard, they're not an endangered species.

## HIGH TECH WINDBAG

Looking first at the output side of the machine, I examined the fan control issue. Wanting to minimize demands on the as yet undefined PID controller, I chose a PWM (pulse width modulation) scheme in which the controller need only drive a single output line whose duty cycle establishes the fan speed. I figured that 8 bits of resolution (i.e., 256 steps between full on and off) coupled with a rather leisurely frequency (due to the slow response of the fan)-perhaps 10 Hz-would be more than adequate.

So, what I needed was a 12-V fan controller accepting a single TTL-compatible PWM input. Being mainly a 1s-and-0s kind of guy, I swallowed my pride and decided to get some help...

"Hey Steve, could you give me some advice on this?"

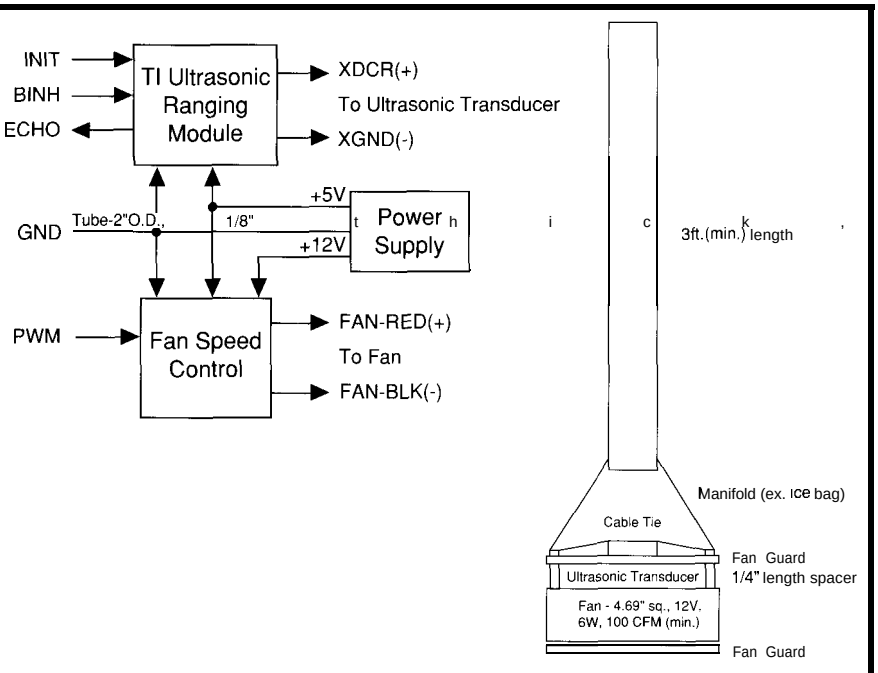
"Sure, it's easy-just use a 2N3055 and a Schottky.. ."

"Er, what's a 2N3055?"

"You know, it's one of those three-pin power trans.. ."

"Uh, what do the pins do?"

"<sigh> Let me give you a hand."




| Part # | Description | Supplier |
|---|---|---|
| TI01 | TI ultrasonic ranging module | Micromint (800) 635-3355 |
| CR069-ND | 12 V@0.5 A (110 CFM) fan | Digi-Key (800) 344-4539 |
| CR176ND | Fan finger guard | " |
| 2N3055 | NPN power transistor TO3 | " |
| HS103-0.5 | TO3 heat sink (option: may be reqd. for higher power fan) | " |
| IN5822 | 40-V Schottky diode | " |
| DM7406N | Open-collector hex inverter | |
| Resistors | 10k, 1k, 100 Ω, ¼ W | " |

Misc. screws/spacers (e.g., 6/32) to mount finger guard to fan.
Cut up ice bag, rubber glove, plastic funnel, etc. may be used for manifold.

Figure I--The key to *the PID-Pong* machine is *the T.I.* ultrasonic ranging *module used at various times over the years by Circuit Cellar projects. Much of the rest of the project can come from your junk box.*

A few days later a neatly wired little black box arrived-voila, instant fan controller (see Figure 2). It's nice to know that if this high-tech stuff doesn't pan out, I've got great potential in that growth career of the '90s: begging. Anyway, thanks to Steve for helping me and all the other "analog challenged."

My final dilemma was some kind of manifold to connect the 4.5" fan to the 2" tube. Since I'm also "mechanically challenged," fabricating some kind of metal or plastic fixture was out of the question. However, having no shortage of MacGyver-like ingenuity, I soon found a quick and easy solution.

Given that my escapades are often the cause of my wife's worst headaches, it's fitting that I sacrificed her ice bag to the cause. While breaking the news to her led to a rather ugly scene, the solution was otherwise wonderful. Cutting off both ends, the "neck" of the bag fit nicely over the tube and was easily secured with a wrap of duct tape. The wider end was a perfect "stretch fit" over the corners of the fan.

So, with a little help from my friends, the output side of the PID-Pong machine was done. Now, it was time to face the feedback side and start dinking with the TIO 1.

## WHERE AM I?

Besides power and ground, the TI01 has five lines to deal with. Two (XDCR+ and XGND-) comprise the

"speaker wire connections" for the separate transducer while three others (INIT, ECHO, and BINH) connect to the controller.

Operation of these latter lines is straightforward (Figure 3). The controller asserts INIT and listens for ECHO; the elapsed time between them represents the distance. An important note that may save you lots of head scratching is that the ECHO output isn't quite TTL compatible and may need a pull-up to +5 V (3.9k is recommended) depending what you connect it to.

An embellishment is the BINH (Blank Inhibit) input to the module. Normally, when this input is grounded, an internal time constant inhibits echo reception to ensure that the transducer settles, otherwise the previous ping may be interpreted as its own echo. The internal blanking time translates into an 18" minimum distance measurement. By explicitly controlling BINH, it's possible to cut the blanking time and thus reduce the minimum distance measurable.

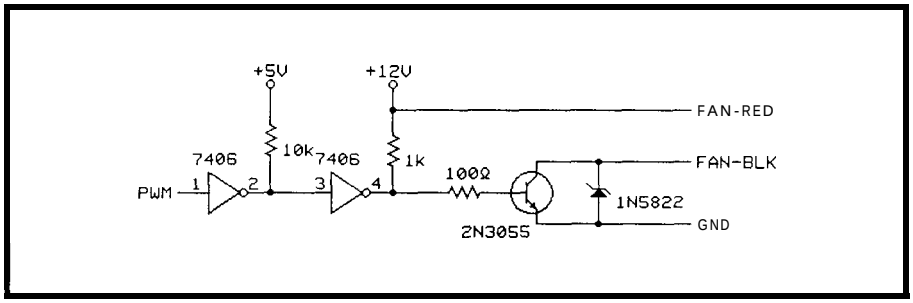



Figure 2-In order to vary the speed of the PID-Pong fan, the motor power is pulse-width modulated. A simple power driver circuit makes the conversion from TTL to fan control.

I faced another potentially dicey fabrication challenge in mounting the transducer to the fan. Eyeballing the situation, I noticed that the transducer has four opposing slots (i.e., 6 & 12, 3 & 9 o'clock in its rim that are about ⅛" wide. Well look at that, there's also about ⅛" spacing beween the mounting arms of the fan guard. Wading through my junk drawer, I soon came up with a ⅛" cable tie. Sure enough, I was able to thread the cable tie through the transducer slots and fan guard. Route the transducer wires along the fan guard and down the side of the fan, where they can be secured by the stretch fit of the ice bag (you don't want those wires playing footsie with the fan). Feeling it wise to keep the TI01 electronics module away from the fan motor, I just let it dangle by the transducer wires about a foot below the fan assembly.

Now, I am almost ready to fire it up, but first I need to make sure the transducer is aligned with the tube. What could be a rather troublesome task is made easier by the fact that the transducer has a gold foil face. So, just peer down the tube and adjust the bag over the corners of the fan until you see your beady orb staring back at you.

Figure 3—The controller asserts INIT and listens for ECHO, the elapsed time befween them representing the distance from the transducer to the ping-pong ball. BINH can be used to reduce the minimum distance measurable by the transducer.

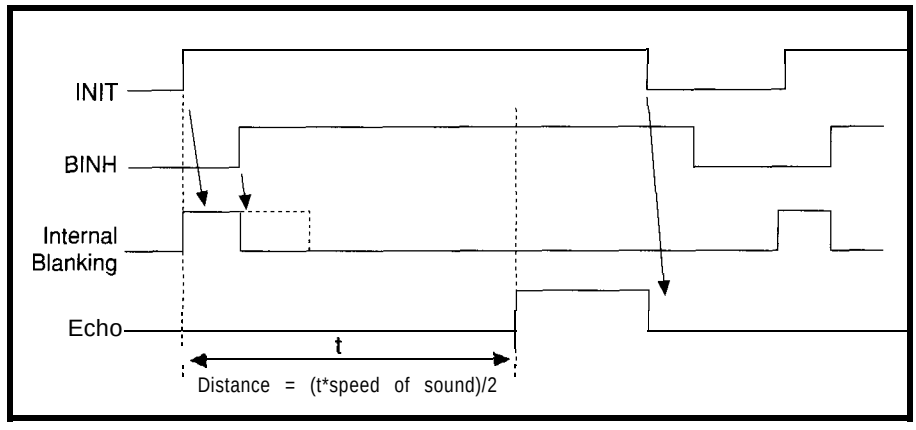



Photo I--The PID-Pong machine is a true tribute to finding new uses for the most mundane household items. The crowning glory is the ice bag used to direct airflow from the fan info the plastic tube. An ultrasonic transducer (not shown here) between the fan and the tube bottom continuously measures the position of the ball in the tube.

As mentioned earlier, I had no idea whether this setup would prove hospitable to the TI01. I was concerned that noise-both acoustic and electrical-might prove to be my undoing. To address the latter, I used a single multiple-output power supply with a fixed output (5 V @ 2 A) for the TI01, and a variable (12 V @ 1 A peak during startup) supply for the fan and its controller.

I hooked the INIT and ECHO lines to a BASIC SBC [remember to ground BINH if you don't drive it and don't forget the pull-up on ECHO) and wrote a simple test program (drive INIT and then enter a loop that increments a counter until ECHO received) to checkout the TI01 operation.

Wisely, I decided to do so with the fan off, having learned the hard way to always test each function in isolation or you'll never figure out what's going on. After starting the test program, I heard the familiar "tick" of the transducer and started moving a diskette (anything hard and smooth serves as a good reflector) up and down over the top of the tube. The good news was I discerned some correlation between position and loop counter. The bad news was there were also a fair number of garbage readings.

I'm glad the fan was left out of the initial test, or I might have been tempted to write off the PID-Pong idea. However, I'd used the TI01 before and felt that, with no fan, there was no reason it shouldn't work. Turns out, although the SBC and TI01 were running from the same power supply, I still needed an explicit ground wire between the two; yet another analog mystery.

Now for the big test: I fired up the fan and continued my TI01 testing. Praise silicon and pass the solder-it worked like a charm.

## HELPFUL HINTS

With both input and output working, the ping-pong balls started flying. Based on my experience, let me give you a few words of advice.

First of all, it turns out that the PID-Pong machine, as simple as it is, is really quite a challenge to control. If you have a variable supply (O-12 V), you can directly connect it to the fan. Then, stick a couple pieces of tape a few inches apart-representing setpoints-on the tube. Remember to keep the lower one above the TI01 blanking zone. I found BINH could be used to cut it from 18" to about 12", but it pays to be conservative, since the TI01 will "bottom out" (i.e., return an immediate echo no matter where the ping-pong ball is) if the blanking interval is too short. For reasons I'll explain shortly, don't put the upper setpoint any closer than 6" or so to the top of the tube. Ready? OK, try to move the ball smoothly between the setpoints using the variable supply.

Surprisingly difficult, eh? Even my kids, with enough hand-eye coordination to bring a Nintendo to its knees, fared little better. If human intelligence is so praiseworthy, how come I feel like such a 4-bitter when it comes to a simple game of PID-Pong?

The worst problem is the severe inertia (in PID-speak, "lag") of the fan, which takes a healthy fraction of a second to spin up / down in response to even the smallest transitions. Despite

my best efforts to consciously compensate ("ball up, power down"), undershoot and overshoot (usually terminal, i.e., flying or shredded ping-pong ball) were unavoidable.

But there's more to this than initially meets the eye. It turns out this is a nonlinear control problem in that the effect of changes in fan power on ball position varies depending on the location in the tube. Near the bottom of the tube, small changes in fan power have a big impact while at the top, it takes a big change in fan power to move the ball a little. That's why I recommend the upper setpoint be placed well below the top of the tube.

Also, don't forget gravity since it makes the PID-pong machine, like many in the real world, asymmetric. Really, it's two different problems (moving the ball up and moving it down) which are theoretically best served by different control strategies.

Finally, no control problem is worth its salt without a few "disturbances"-otherwise you could probably handle it open loop. Yes, you can just choose a fan constant corresponding to each setpoint and "tune" it to work pretty well-for a little while. One problem that, though mundane, can definitely confuse a simple-minded control scheme is that the relationship between fan power and RPM varies depending on temperature Put simply, the fan goes faster when it's "warmed up."

A final-and tricky to handle—embellishment is that the machine proves surprisingly sensitive to atmospheric conditions. Since airflow depends not only on velocity, but also pressure and temperature, something as minor as a slamming door ("Real sorry about the ice bag, dear.") will cause a few-inch glitch in ball position.

## PID-PONG CHALLENGE

Like a stealth fighter, the PID-Pong machine is unflyable by mere humans.

While a fly-by-wire system takes racks of redundant RISCs, I imagine controlling the PID-Pong machine won't take that many MIPS.

PID is the obvious first candidate for a control strategy. Beyond that, more esoteric alternatives (fuzzy, neural network, psychic, ???) surely exist. All you need is four I/O lines and a good imagination.

Let me know if you have any bright ideas. I'll be relying on friends who know a lot more than I do to help me along. When I make progress, you'll hear about it in future issues. In the meantime, keep those ping-pong balls flying.

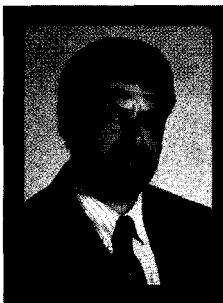
*Tom Cantrell has been an engineer in Silicon Valley for more than ten years working on chip, board, and systems design and marketing. He can be reached at (510) 657-0264 or by fax at (510) 657-5441.*

419 Very Useful
420 Moderately Useful
421 Not Useful

# Embedded Timers

## EMBEDDED TECHNIQUES

John Dybowski



Unlike humans, computers don't have to deal with annoying alarm clocks just to tell them it's time to get up for work. Instead, it's a matter of what RTC you choose for it. John's choice of Signetics' PCF8583 will let your computer sleep easier.

Most embedded systems operate in the real-time domain. Therefore, it's not surprising that they require an accurate timebase as a reference for acquiring, evaluating, and dispensing time-related events. Even if the system's functionality is somewhat self contained, the need for a system heartbeat persists. Often you will find that various peripheral functions can be synthesized under certain circumstances if a system timer is available. Certainly, functions such as keyboard scanning, switch debouncing, counting communications turnaround delays, and other delay functions would be difficult to implement cleanly if a timebase were not available. An activity that is closely related to counting delay intervals is keeping track of real time, which can be useful for a variety of purposes that need the services of a "time-of-day" clock.

In this regard, a multitude of monolithic real-time clocks (RTCs) are available in a variety of shapes and sizes. Since these real-time clocks must be capable of keeping time in the absence of system power, they all have the capability of operating at reduced voltage levels provided from a variety of backup power sources, which include primary and secondary batteries and so-called supercaps. In some cases, the clocks even have encapsulated lithium batteries and watch crystals built right into their DIP packages. Because of their capacity for nonvolatile operation, it's only natural that any excess RTC silicon be dedicated to static RAM cells.

If all you're looking for in a real-time clock is a good timepiece for your embedded system, then you won't have any problem finding exactly the right price/performance combination to meet your needs. Of course, the issue of a system timebase should be carefully considered during the RTC selection process. Depending on your requirements, this concern might cause you to dismiss many of the available RTCs from further consideration immediately. It would be foolish to pick a fine timepiece only to have to add another handful of components to generate the required timer interrupt clock source.

There are other timing-related functions that could be beneficial to embedded systems as well. Often you will find that including some sort of alarm capability can prove to be very useful. In some cases, especially in the case of battery-operated equipment, it's a real advantage to run the instrument in a degraded mode of operation for periods of time while waiting for something noteworthy to happen.

Frequently you will want the system to remain completely dormant for extended periods. In this case you normally use the real-time clock's alarm capability to wake up the controller. In response to this wake-up call, it would take a look around and decide whether it should do some processing or just arrange for an ensuing wake-up call and go back to sleep. This scenario could be centered around operating the controller and its associated peripherals in some sort of power-conserving idle or sleep mode.

Alternatively, the ultimate in power savings could be realized by completely shutting down the system's main power source during times of inactivity. In any case, the means of reestablishing normal operation would be a hardware signal that would most logically be generated by the nonvolatile system timekeeper.

### REAL TIMERS

This last scenario might sound familiar to some of you, and is exactly the problem I faced when I developed my battery-operated data logger ("The

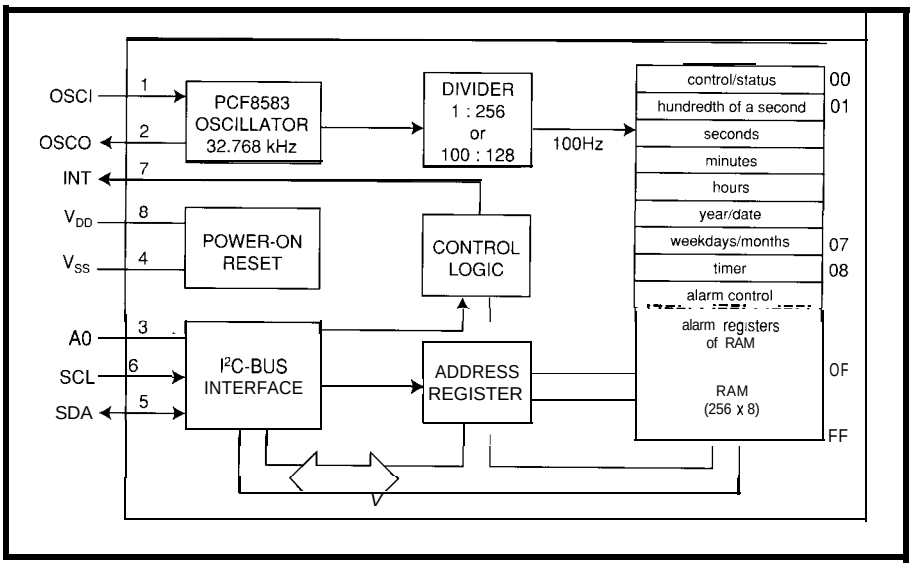



Figure 1—*The PCF8583* real-time clock/calendar *from Signetics can operate at a voltage level as low as one volt. Add to this its powerful functionality and this chip has many uses as a backup power source.*

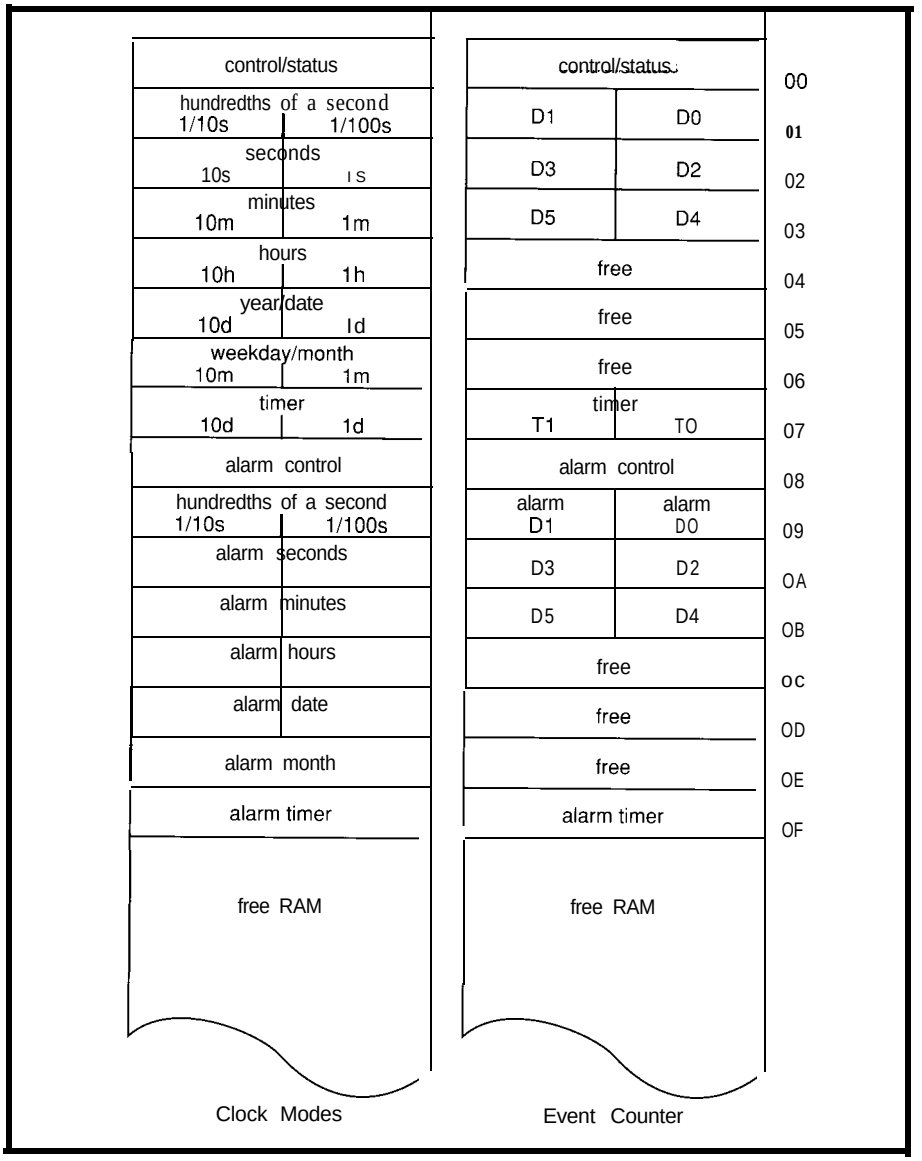



Figure 2—*The PCF8583 allocates* 256 *8-bit storage locations for general and dedicated use. Clock and timer data is stored in packed BCD format.*

Elements of a Data Logger," June/July 1992, issue 27). My main problem was that I required a programmable interval timer that would continue functioning when the main power was shut down and still have the capability to assert its interrupt request line in order to reenergize the system. You may recall that the RTC/timer solution I finally settled on was based on the aging MC 1468 18A real-time clock from Motorola. It worked but, needless to say, it was far from optimal. This thing required a hand-crafted oscillator front end just to cut the 100-µA standby power in half. And as it was, coercing it to function as an interval timer involved some rather unsavory gyrations.

Of course, there are MC 1468 18A-compatible parts available from various sources with built-in batteries and crystals, but with all of them the IRQ alarm/interrupt is inoperative when running in backup mode. Talk about an compatibility! This is a shame, because in all other respects this clock's architecture offers a fair amount of flexibility. There are many other RTCs that are not nearly as useful for generating a periodic timebase. Many offer the selection of one of several taps from the internal divider chain. Others have no timer outputs at all. What all this points out is that, in spite of all the RTCs on the market, once your application starts to move away from the mainstream, your options become much more limited.

If you've ever had the need for an accurate and flexible interval timer that can be configured under processor control, then the problem should be well understood. Frequently, in this respect, there is a need to count delays ranging from subsecond intervals to times spanning days, weeks, or even months. Since, by its very nature, this type of timekeeping implies unattended operation for extended periods, accuracy over the time domain is generally presumed to be a requirement. If the timing element is good, it should also provide for outputs with fast repetition rates that could be used as a reference for a timer interrupt when operating during periods of heavy processing.

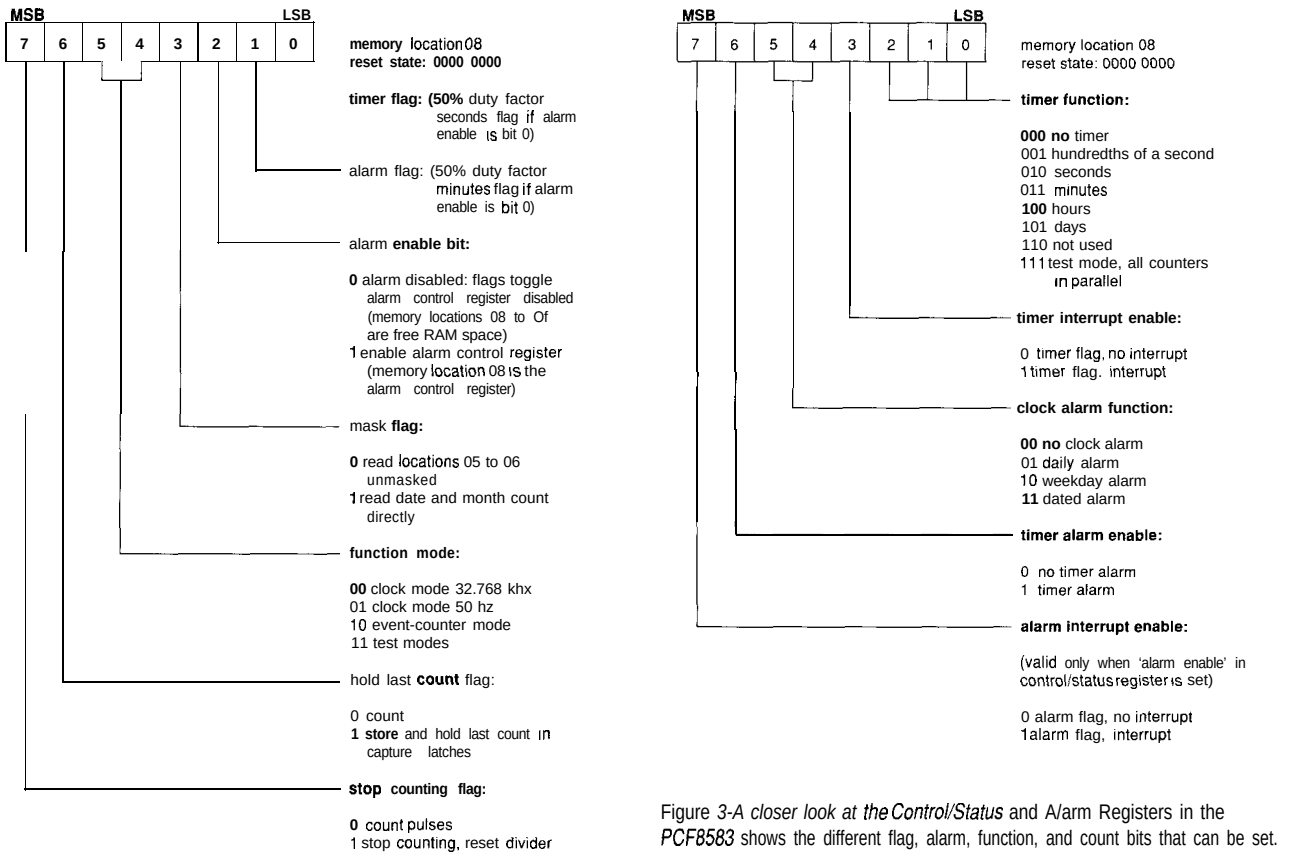



Figure *3-A closer look at the Control/Status* and A/arm Registers in the *PCF8583* shows the different flag, alarm, function, and count bits that can be set.

As an example, consider a battery-operated data collection instrument equipped with power control circuitry and a sophisticated RTC/timer system. That is, a system that has the ability to turn itself on and off based on an interval of time or in response to external stimuli. Such a system would have the capability to lie dormant for an extended period of time. The first sign of life might not occur for days, weeks, or even months after the system had been set up. Typically, the return to life would be evoked in response to a dated alarm generated at the RTC/timer system. Following this, the timing system would perhaps be programmed to reawaken the system more frequently on a periodic basis in the range of minutes, hours, or days. This would give the processor the opportunity to look around in order to determine if any notable events had occurred or if setpoints were being approached.

Once the situation warranted, the timer would be set to a relatively high repetition rate that would allow the system to sample the prevailing conditions much more frequently. Now data points could be either stored or discarded as appropriate to the particular application. Stored transactions would, of course, be time stamped with time marks that originated at the central RTC/timekeeper. On detecting some critical conditions, the system would switch to a continuous mode of operation where the timer would now be set to generate some convenient interrupt timebase, usually on the order of milliseconds. Once the external events returned to a more stable condition, the instrument would return again to a moderate sample rate. By using means such as this, extremely long operating periods are feasible using battery power.

Actually, in a carefully designed system, several levels of power conservation can be realized with the aid of a flexible timing system. Obviously, in the example above the primary power saver is the fact that the system has the capability of shutting itself off during idle times. A secondary level of power savings can be achieved when the sample period is too frequent to allow a complete system shutdown. Using the scenario described above, consider the system's operation when it is running continuously. Here it may be adequate to take a reading perhaps once every 10 or 20 milliseconds.

Rather than waste time spinning a delay loop, program the timer to generate an interrupt at the desired frequency and execute a halt or idle instruction. This effectively stops all bus activity thereby cutting the current consumption significantly. In response to the timer interrupt, you can generally do your business in burst of hundreds of microseconds, which obviously keeps the average power consumption at much lower levels then when running full bore doing nothing. With the right interrupt timebase, you can debounce contacts and switches and count much longer delays, or use it for functions that don't have to be serviced as often.

## NO MORE GAMES

As engineers, we must service the bottom line. This amounts to achiev-

ing the required functionality in the products we design. Often this boils down to making the system building blocks perform functions they weren't originally intended to do. This is what I was alluding to when I described the timing system I used in my data logger. Sometimes, however, you get lucky and find a part that fits the bill without the need for any contortions whatsoever. Functionality without the pain; no more games.

Surprisingly, the answer to my RTC/timing woes comes in a small package that not only fulfills the needs of the hypothetical data collection system I described above, but also contains many additional functions. As an added benefit, this part communicates to the host processor using the I²C bus that only consumes two I/O pins. It also contains 256 bytes of nonvolatile RAM. This part is the PCF8583 real-time clock/calendar from Signetics. As a good example of reasonable engineering, this S-pin DIP features moderately low power consumption and a very complete timing system. With current consumption in the 10–50-µA range, it's obvious the designers didn't kill themselves trying to choke the thing down until it barely worked. While others worked into the night creating RTCs that consumed almost no power at all, these guys concentrated on providing a tremendous amount of functionality into a deceptively small package. All in all, it is a reasonable use of technology.

Anyway, what's the point of getting the thing to run on nanoamps if the battery disintegrates long before you get a chance to tap its capacity? Interestingly, the PCF8583 operates all the way down to 1 volt, which opens up some intriguing backup power possibilities. You could use a single alkaline or NiCd cell to achieve nonvolatility. Figure 1 shows this RTC's basic architecture in block form.

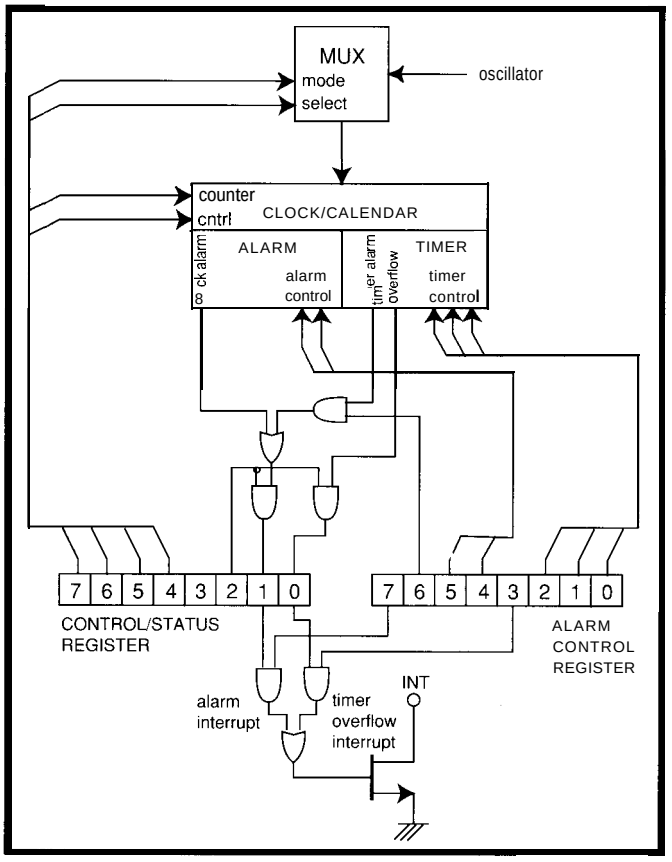



Figure *4-The PCF8583* has a *multitude of ways to generate* an *interrupt that be confusing at first glance.*

The PCF8583 allocates 256 8-bit storage locations for general and dedicated use as shown in Figure 2. In the following discussion, keep in mind that the defined count, alarm, and timer registers all operate on packed BCD values.

Location 0 is the Control/Status Register and is used to set up all of the PCF8583's functions and options. This is where you select the master timebase, whether or not interrupts are enabled, and other various configuration options. Locations **I-6** are the clock/calendar registers and and contain counts for hundredths of seconds all the way through years. The Alarm Control Register is at location 8. All of the alarm and timer functions are programmed through this register The routing of various alarm events to the open-drain interrupt request pin (IRQ) is also established via this register. Location 7 defines the timer register. The timer is an up-counter and it can be configured to count intervals from hundredths of seconds to days.

The Alarm Control Register determines, among other things, the timebase tap that will drive the timer register. Depending on how you further program the Alarm Control Register, an interrupt can be generated on a timer overflow (from 99h to 00h), or when the timer register is equal to the alarm timer (location F). The dated alarm registers are located at a fixed offset of eight relative to their corresponding time/date counterparts, and start at location nine. A clock alarm can be configured as a dated alarm, which requires all of the clock and alarm register to match bit by bit.

Another mode is available that enables a daily alarm and a weekday alarm. Basically, the daily alarm ignores the month and date bits, where the weekday alarm allows you to select the weekdays on which the alarm will be evaluated through the weekday/month register (E). The active weekdays can be selected individually or in any combination. Free RAM begins at location 10H and if you don't use the alarm function, the alarm registers are available for use as RAM storage.

All these options may sound a bit confusing but this RTC is fairly easy to use once you understand it. Unfortunately, it seems the rather dry data sheet yields its secrets reluctantly. Ultimately, I had to resort to the old engineering trick of fooling around with the part until I figured out what was going on. Figure 3 shows the contents of the Control/Status Register and the Alarm Register. The ultimate effect of these two control registers on the IRQ pin is shown schematically in Figure 4.

The PCF8583 actually has two distinctly different modes of operation. In addition to the fantastically complete clock/calendar/timer mode, it can also be programmed to function as an event counter. This mode uses the

oscillator input to count pulses that are externally applied. In this usage, up to six digits of BCD data are stored, giving it the capacity to record up to one million events. If that's not enough, you can enable the event alarm function. Here, the timer location increments once for every one hundred, ten thousand, or one million events depending on the value programmed into bits 0, 1, and 2 of the alarm control register. With all these features, you have the luxury of capture registers that make reading multiple registers on-the-fly without problems.

## WHAT YOU NEED AND WHAT YOU HAVE

When using a complex part such as the PCF8583, most people will undoubtedly find that their individual application requires only a portion of the available feature set. As usual, it's a good thing to concentrate on the task at hand and not get carried away trying to second guess every contingency that may come along. This is especially true if you intend to write general-purpose support drivers.

At the same time, it pays to look ahead a little so as not to produce code that is unnecessarily restrictive. Of course, it helps if you can draw upon work that has already been completed to make the job easier. This is exactly what I did, having already coded some low-level I²C drivers. These drivers perform the read and write function and are grouped into those that read and write bytes when using simple I²C peripherals and those that can handle communications with more sophisticated I²C peripherals that contain addressable registers.

Since most I²C peripherals that have addressable registers also have an address autoincrement capability that allows you to stream data, these support services are named Rec_I2C_String and Xmit_I2C_String, though they don't necessarily have to be used to transfer string data at all. What makes these routines different is that they provide the register addressing capability for the I²C chips that need it. The conventions employed by these so-called string routines include

**Listing 1**—*Since the PFC8583 is a standard PC device, routines that have a/ready been written for PC communication can save on development time.*

```
;GLOBAL ENTRY POINTS

        PUBLIC SET-ALARM
        PUBLIC SET-TIMER
        PUBLIC SET_RTC
        PUBLIC READ_RTC

;EXTERNAL REFERENCES
        EXTERN Xmit_I2C_String (CODE)
        EXTERN Rec_I2C_String (CODE)
        EXTERN Xmit_I2C_Byte (CODE)
        EXTERN Rec_I2C_Byte (CODE)
        EXTERN RTC_AREA (XDATA)

;CONSTANTS
PCF8583 EQU     OAOH

;ASSEMBLE INTO CODE SEGMENT
        SEG     CODE

;SETUP THE RTC AS AN INTERVAL TIMER
;INPUT: ACC CONTAINS COUNTING INTERVAL:
        000B=NO TIMER
        001B=HUNDREDTHS OF SECONDS
        010B=SECONDS
        011B=MINUTES
        100B=HOURS
        101B=DAYS
```

*(continued)*

Listing 1-continued

```
        B CONTAINS BCD RELOAD VALUE (UP-COUNTER OVERFLOWS AT 00)

SE    IMER        PROC
        PUSH      ACC
        PUSH      B
        MOV       A,#00000100B    ;CONTROL/STATUS REGISTER
                    ||      |----          TIMER FLAG
                    ||                     ALARM FLAG
                    ||                     ALARM ENABLE (1=ENABLE)
                    ||                     MASK FLAG (0=UNMASK 5, 6)
                    ||                     MODE (00=32K XTAL AS CLOCK)
                    ||                          "
                    ||                     HOLD LAST COUNT (1=HOLD)
                    ||                     STOP (1=STOP COUNTING)
        MOV       DPTR#R  C AREA
        MOVX      @DPTR,A
        MOV       R0,#1               ;1 BYTE TO SEND
        MOV       B,#0                ;CONTROL REGISTER ADDRESS
        MOV       A,#PCF8583          ;CHIP ADDRESS
        CALL      Xmit_I2C_String     ;SET CONTROL REGISTER

 NOW SETUP 2 BYTE ALARM CONTROL/DATA STRING

        MOV       DPTR,#RTC_AREA
        POP       ACC                 ;RETRIEVE TIMER COUNT
        MOVX      @DPTR,A
        INC   ;   DPTR
        POP       ACC                 ;RETRIEVE INTERVAL
        ORL       A,#00001000B        ;ALARM CONTROL REGISTER
                    ||||||| I----TIMER FUNCTION (CALLER SUPPLIED)
                    |||||||-----       "
                    ||||||------       "
                    |||||-------TIMER INTERRUPT FLAG (1=ENABLE)
                    ||||--------MODE (00=NO CLOCK ALARM)
                    | | |---------       "
                    | |----------TIMER ALARM ENABLE (1=ENABLE)
                    |-----------ALARM  INTERRUPT ENABLE (1=ENABLE)
        MOVX      @DPTR,A
        MOV       DPTR,#RTC_AREA
        MOV       R0,#2             ;2 BYTES TO SEND
        MOV       A,#PCF8583        ;CHIP ADDRESS
        MOV       B,#7              ;TIMER ADDRESS
        CALL      Xmit_I2C_String ;SET TIMER AND ALARM REGISTER
        RET
        ENDPROC
;SET THE RTC
;INPUT: DPTR POINTS TO 6 BYTE BCD TIME/DATE SOURCE BUFFER:
        HUNDREDTHS OF SECONDS
        SECONDS
        MINUTES
        HOURS
        YEAR/DATE
        WEEKDAY/MONTH

SET_RTC PROC
;(NOT SHOWN! COUNTING SHOULD BE STOPPED BEFORE LOADING TIME/DATE
REGISTERS.)
        MOV       R0,#6             ;6 BYTES TO SEND
        MOV       B,#1              ;TIME/DATE ADDRESS
        MOV       A,#PCF8583        ;CHIP ADDRESS
        CALL      Xmit_I2C_String ;SET TIME/DATE REGISTERS
        RET
        ENDPROC

 READ   HE RTC
 INPUT   DPTR POINTS 6 BYTE BCD TO TIME/DATE DESTINATION BUFFER:
        HUNDREDTHS OF SECONDS
```

(continued)

Listing I-continued

```
        SECONDS
        MINUTES
        HOURS
        YEAR/DATE
        WEEKDAY/MONTH
;
READ_RTC        PROC
                MOV     R0,#6           ;6 BYTES TO READ
                MOV     B,#1            ;TIME/DATE ADDRESS
                MOV     A,#PCF8583      ;CHIP ADDRESS
                CALL    Rec_I2C_String ;READ TIME/DATE REGISTERS
                RET
                ENDPROC

;SET THE RTC ALARM
;INPUT: DPTR POINTS TO 6 BYTE BCD TIME/DATE ALARM SOURCE BUFFER:
        SECONDS
        MINUTES
        HOURS
        YEAR/DATE
;       WEEKDAY/MONTH
SET_ALARM       PROC
        PUSH    DPL
        PUSH    DPH
        MOV     A,#00000100B    ;CONTROL/STATUS REGISTER
                    |||||||| I----TIMER FLAG
                    ///)/I/-----ALARM FLAG
                    ||||| l------ALARM ENABLE (1=ENABLE)
                    |||||-------MASK FLAG (0=UNMASK 5, 6)
                    ||||--------MODE (00=32K XTAL AS CLOCK)
                    ||||---------  "
                      l----------HOLD LAST COUNT (1=HOLD)
                     I-----------STOP (1=STOP COUNTING)
        MOV     DPTR,#RTC_AREA
        MOVX    @DPTR,A
        MOV     R0,#1           ;1 BYTE TO SEND
        MOV     B,#0            ;CONTROL REGISTER ADDRESS
        MOV     A,#PCF8583      ;CHIP ADDRESS
        CALL    Xmit_I2C_String ;SET CONTROL REGISTER

;NOW SETUP ALARM CONTROL REGISTER
        MOV     DPTR,#RTC_AREA
        MOV     A,#11110000B    ;ALARM CONTROL REGISTER
                    |||||| I----TIMER FUNCTION (000=NONE)
                    ||||||-----      "
                    ,,,,,       |------ "
                    |||| /-------TIMER INTERRUPT FLAG (1=ENABLE)
                    ||||--------MODE (11=DATED ALARM)
                    ||   |---------  "
                    ||----------TIMER ALARM ENABLE (1=ENABLE)
                    |-----------ALARM INTERRUPT FLAG (1=ENABLE)
        MOVX    @DPTR,A
        MOV     R0,#1           ;1 BYTE TO SEND
        MOV     B,#8            ;ALARM CONTROL ADDRESS
        MOV     A,#PCF8583      ;CHIP ADDRESS
        CALL    Xmit_I2C_String ;SET ALARM CONTROL REGISTER

;SET THE ALARM REGISTERS
        POP     DPH
        POP     DPL
        MOV     R0,#6           ;6 BYTES TO SEND
        MOV     B,#9            ;TIME/DATE ALARM ADDRESS
        MOV     A,#PCF8583      ;CHIP ADDRESS
        CALL    Xmit_I2C_String ;SET ALARM REGISTERS
        RET
        ENDPROC
        END
```

setting up the I²C chip address in the accumulator, the specific register address in the B register, and the number of bytes to transfer in RO. DPTR points to external RAM where the data is to read from or written to. I'll admit that as I use these routines with different I²C peripherals, it occurs to me that they could have been written differently to perform certain operations more effectively and efficiently. In any event, I don't feel overly compelled to fix them; they do the job and at least they're paid for!

The first thing anyone has to do with an RTC is to set and read the time and date. These functions are performed by the routines SET_RTC and READ_RTC, respectively. The SET_RTC routine takes its input through an external RAM buffer denoted by D PT R, where values for hundredths of seconds, seconds, minutes, hours, days, months, and year are placed. Likewise, the READ_RTC routine returns the corresponding values to external RAM locations pointed to by D PT R.

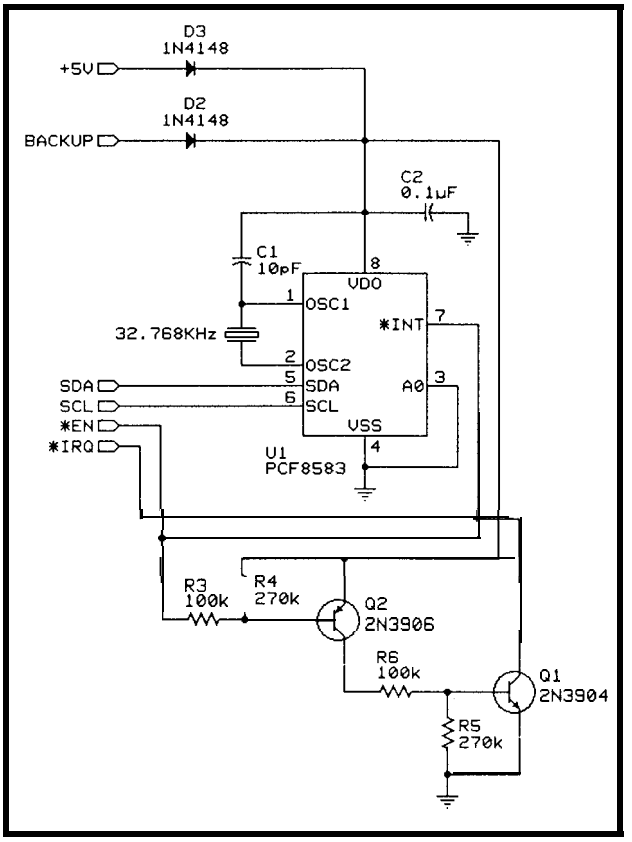



Figure 5-A *handful of extra components is required to prevent the PCF8583's IRQ from driving the processor's input while the* processor is powered down.

Note that the clock doesn't have to be stopped when the RTC is read. The chip is smart enough to transfer all of the clock and calendar registers to capture registers when the read operation is initiated, ensuring all of the counts belong together. This is not the case with a write operation, so you must explicitly stop the clock to prevent problems should a counter overflow occur in the middle of an update.

Setting up for a dated alarm is just like writing to the clock/calendar registers except that the transfer address is 9 instead of 1. The SET-ALARM service routine

handles the settings required by the Control/Status Register and Alarm Control Register. The assumption is made that an alarm output on IRQ is desired, so this function is unmasked as well.

Setting up the PCF8583 as an interval timer is equally easy. Although you could obtain the same end result in a couple of different ways using the PCF8583, the simplest is to configure the timer register to count the desired interval and generate an interrupt on overflow. An interrupt condition is defined as the overflow from 99H to OOH (recall the counters operate in BCD). The clocking event can range from hundredths of a second up to days. You can get very long timing intervals indeed.

The SET_TIMER support routine handles the details of manipulating the Control/Status Register and Alarm Control Register and, as with the dated alarm, the IRQ pin is enabled. Parameters that are passed include the counting increment and the preload value. These are passed through the ACC and B registers, respectively. Say you want an interrupt hit in 20 seconds. Set up the three low-order bits of the accumulator with 010 to tell the PCF8583 to count seconds and B with 80h (100h–20h). Listing 1 shows how these support routines are coded.

The flexibility of the PCF8583's interrupt generation capabilities may lead you to use this feature for more than just generating a microprocessor interrupt timebase. In my case, I used the IRQ pin as both an interrupt source and as a control signal to drive some low-level power control circuitry. In this application the control logic runs off the 3-volt backup power and IRQ is pulled up with a resistor to provide the required logic levels. Obviously, some isolation must be provided on the interrupt leg since it's a bad thing to drive an active level into a powered-down processor. Figure 5 shows how, with a couple of transistors, the isolation can be attained while maintaining the open-drain characteristics of the IRQ signal.

## ALARMING EVENTS

In keeping with what I need in an RTC/timer, I touched on some of the PCF8583's capabilities that are important to me. The PCF8583 has alarming capabilities that go quite a bit beyond what I've been able to cover here. And as for its event counting mode...well, that's another story. ❑

*John Dybowski is an engineer involved in the design and manufacture of hardware and software for industrial data collection and communications equipment. He may be reached at john.dybowski@circellar.com.*

## CONTACT

Signetics Company
811 East Arques Ave.
Sunnyvale, CA 94088-3409
(408) 991-3737

## IRS

422 Very Useful
423 Moderately Useful
424 Not Useful

# PATENT TALK by Russ Reiss

Recently, while searching the patent database for new techniques in human-computer interaction, I became aware of a number of variations to, and improvements on, the design of the ubiquitous computer mouse. As useful as this device already is, there appears to still be room for improvement, particularly for use in specialized applications. One of the biggest drawbacks of the mouse is that it never completely eliminates the need for a keyboard and other input devices. Constantly moving the hand from the mouse to the keyboard and back again is a nuisance. Another problem with the classical mouse design is that it is only two dimensional, while many computer applications involve three (and some even more) dimensions. These and other aspects of mouse design are discussed this month through ten relevant patent abstracts.

The first patent proposes to combine the mouse with other computer input devices in order to provide greater convenience to the user. Abstract 1 presents a triple-function device: a mouse, optical scanner, and digitizing puck rolled into one. With this new device the user can move the cursor around with the mouse, then (without switching devices) scan in some text, or use the very same device to digitize specific points on an engineering drawing. This combination of tools in one device truly seems to provide a great deal of convenience for the user in applications such as CAD.

Abstract 2, on the other hand, recognizes that the mouse is simply an upside-down trackball. The patent covers a physical housing which can be altered so that the same device can function in either mode. This approach would certainly seem to be more cost effective for the user who requires both mouse and trackball, even though both are not available at the same instant.

It seems only reasonable that eventually there will be available new and unique input devices that may compete with the mouse in some applications. The device presented by Microslate Corporation in Abstract 3 uses a stylus—much like a pen-in place of a mouse. This approach would seem ideal for applications with limited (or NO) desk space, such as is often the case when using laptop computers. I know that while my mice often find a place to "sit," they hardly ever have enough room to "run around"! Going what appears to be a step further, the finger-worn graphic interface device referred to in Abstract 4 looks intriguing. While the abstract is scant on details, the promise of acting like both a digitizing tablet and mouse with a more efficient and intuitive feel warrants further investigation.

The next four patents all address the two-dimensional limitation of the conventional mouse. IBM's device in Abstract 5 uses a conventional mouse in all respects, but with the addition of a Z-axis data-generating mechanism beyond that of the simple push buttons normally found. They mention two possible forms of this mechanism: a pressure-sensitive button mounted on the surface of the mouse, and a more novel approach using a hole in the mouse into which a finger may be inserted to varying depths to input Z-axis data.

A similar approach is taken in the patent of Abstract 6. This one also uses an additional "analog button" to input data to the computer. While the discussion concentrates on using this information to control parameters such as line width and color, in reality it could represent any third dimension. They go on to include a small numeric keypad on the mouse design so that the user need not switch back to the conventional keyboard to enter numeric data. They appear to try to cover all bases by also mentioning incorporation of the analog button as part of a light pen.

If the foregoing two abstracts seem somewhat analogous, consider Abstract 7. Here, General DataComm seems to attempt to displace all patents in this area by their very general wording. They describe a device which measures two perpendicular dimensions over a planar surface and has means for having a third analog dimension entered into the

1

| | |
|---|---|
| Patent Number | 4,906,843 |
| Issue Date | 1990 03 06 |
| Inventor(s) | Jones, Terrill H.; Sundby, Dale H.; Wright, Steven A. |
| State/Country | CA |
| Assignee | MARQ Technolgies |
| Title | Combination mouse, optical scanner, and digitizer puck |
| Abstract | A hand-held unit can operate as a mouse to move a cursor on a display, as a hand-held optical scanner for entering into a computer characters or graphic information delineated on a worksheet across which the unit is moved, or as a digitizing puck for tracing and digitizing lines or curves on a worksheet laid over a digitizing pad. The unit is connected via an electrical cable to an interface board plugged into an expansion slot inside a personal computer. |

# PATENT TALK

apparatus while located along the planar surface. Isn't this what the IBM device does too? Their final sentence makes it clear that they mean for this patent to cover all forms involving a mouse, bit pad, or light pen. If their actual claims are as broadly stated as the abstract, I'm sure some patent attorneys will be kept quite busy with this one.

If three-dimensional information is still too limiting for you, consider the patent covered by Abstract 8. This device promises to provide the user with six degrees of freedom in control of graphical objects on the screen. The design begins with a rather conventional mouse ball approach for control of the two translated dimensions. To this is first added a finger-operated conveyor belt, or roller, for achieving control over the third translated axis. Now, for control over the three additional rotational dimensions (pitch, roll, and yaw), three finger-operated controls such as wheels are additionally added to the "super mouse." While the concept would appear to be quite useful for multidimensional control of a computer display, one can only wonder how natural such a contraption would feel to the user. The rotational wheels would need to turn freely with little resistance, yet remain stable in any position while the user is contorting the other controls.

Moving away from the design of the mouse itself, the final two abstracts present a couple of interesting uses of the computer mouse. I am always startled by sweeping


| | |
|---|---|
| Patent Number | 5,063,289 |
| Issue Date | 1991 11 05 |
| Inventor(s) | Jasinski, Joseph E.; Lingle, Charles H.; Pollitt, Richard F.; Shuman, David W |
| State/Country | FL |
| Assignee | Lexmark International. Inc. |
| Title | Combined mouse and trackball |
| Abstract | A combined mouse and trackball having its control ball extend past cover 1. Cover 1 has neck hinges so that it pivots in relations to opposing cover. A cable at the front communicates with the computer receiving the control signals from the ball. A switch reverses the significance of front-to-back signals from the ball depending on whether an actuator integral with cover 1 engages the switch. In the closed position of cover 1 the bottom of the device is flat and the ball is upward for use as a trackball. In the open position of cover 1 the device is formed into a mouse with front and rear surfaces to support the ball for frictional movement on a table. Latch surfaces hold the covers in the two positions until they are manually released. |


| | |
|---|---|
| Patent Number | 5,019,677 |
| Issue Date | 1991 05 28 |
| Inventor(s) | Menen, Balan |
| State/Country | CA |
| Assignee | Microslate Corp. |
| Title | Computer input device |
| Abstract | A computer input device for positioning a cursor on a computer video screen. With this device, information is put into the computer by the penmanship motion of the hand holding a stylus in the fashion of a writing instrument so that only finger motion is required for data input. Device replaces the arm-to-eye coordination of the mouse with hand-to-eye coordination such as that used in ordinary penmanship. Rather than encoders driven by a turning ball or wheels, a resistance array with wiper contacts produces variable resistances which are converted into variable voltages that are a function of position of the stylus. |


| | |
|---|---|
| Patent Number | 4,954,817 |
| Issue Date | 1990 09 04 |
| Inventor(s) | Levine, Neil A |
| State/Country | CA |
| Title | Finger-worn graphic interface device |
| Abstract | A miniaturized finger-worn X-Y graphic interface device. A finger palette, a stylus ring, and an electronic module exhibit benefits of both an X-Y digitizing tablet and a mouse without their inherent difficulties. A unique location and method facilitate use in an absolute or relative mode. When used in conjunction with a conventional keyboard, the computer-human interface becomes faster, more natural, efficient, and intuitive. |

# PATENT TALK

patent claims which appear to achieve coverage for concepts we often take for granted. I think it is necessary for designers to realize that such patents exist and to be on guard whenever they feel their work stands on firm historical grounds. Abstract 9 from Laitram Corporation seems to me to describe a commonplace activity, namely use of a mouse to select textual characters, icons, computer commands/functions, and the like from a menu placed on the screen. This routine activity is common to most CAD systems, specifically schematic capture systems prevalent


| | |
|---|---|
| Patent Number | 5,095,302 |
| Issue Date | 19920310 |
| Inventor(s) | McLean, James G.; Pickover, Clifford A.; Reed, Alvin R |
| State/Country | FL |
| Assignee | International Business Machines Corporation |
| Title | Three-dimensional mouse via finger ring or cavity |
| Abstract | A cursor control/data input device for a computer display system which utilizes a conventional X-Y mouse provided with a third Z-axis data-generating mechanism. The mouse may be used with any nonspecific support surface and would have conventional X-Y data-generating wheels or a rotating ball with appropriate pickup elements to generate the X-Y coordinate data. Third, or Z, coordinate data is produced by a third instrumentality in the mouse body, preferably operable by the operators thumb or index finger. Means comprising a pressure-sensitive button mounted on the surface of the mouse, or alternatively means actuated by the insertion of the operators finger into a hole provided in the mouse's body, generate said Z-coordinate data. Movement of the finger in the hole is measurable by any of a number of different instrumentalities. |


| | |
|---|---|
| Patent Number | 5,063,376 |
| Issue Date | 1991 1105 |
| Inventor(s) | Chang, Ronald G |
| State/Country | CA |
| Title | Numeric mouse one hand controllable computer peripheral pointing device |
| Abstract | A hand-controlled peripheral pointing device having an optional analog switch to permit the user to dynamically and continuously change the value of an attribute, for example, the width of a line or color saturation value, when the device is moved. Activation of the analog button or key permits the user to change an attribute value of graphical data input as desired depending on the depth of depression of the key. In the preferred embodiment, the device is a mouse having a numeric keypad placed thereon. This numeric keypad allows the user to directly enter data without moving the hand back and forth between the mouse and a computer keyboard. The numeric mouse is able to operate in two modes, including a conventional point-and-click mode and an analog mode. An alternate embodiment comprises a light pen having a finger-actuated analog button disposed adjacent the writing tip of the pen. One manner of switching between modes (analog to click and vice versa) is program driven, i.e., depending on what item the mouse or light pen is pointing to, the activation of the key varies an attribute value or functions in the conventional "click" or keystroke manner. |


| | |
|---|---|
| Patent Number | 4,961,138 |
| Issue Date | 1990 10 02 |
| Inventor(s) | Gorniak, Andrew M. |
| State/Country | CT |
| Assignee | General DataComm, Inc |
| Title | System and apparatus for providing three dimensions of input into a host processor |
| Abstract | A system under the hand-held control of a user for providing three dimensions of input to a computer processor while operating on an essentially planar surface is disclosed. The system of the invention generally comprises: an apparatus operated over a substantially planar surface; a means chosen from one of a means for detecting and measuring movement of the apparatus in two perpendicular dimensions along the substantially planar surface and a means for determining the location of the apparatus in two perpendicular dimensions along the substantially planar surface, and for providing first outputs representative of either the two-dimensional location or the movement of the apparatus; means for detecting and measuring an analog third dimension input into the apparatus under the control of the user while the apparatus is located along the planar surface, and for providing a second output representative of the third dimension input; and means for receiving the first outputs and the second output and providing therefrom information suitable for input into the computer processor, wherein the information is representative of the three dimensions of input. The apparatus of the system can take various forms such as a computer mouse, a stylus for a bit pad, or a light pen. |

# PATENT TALK

among electronic engineers. It is also fundamental to nearly all applications that run in GUI-based operating systems. Certainly, most of these systems also allow for "replacement of the keyboard arrays as the computer changes mode of operation, and placement of more frequently used keystrokes on the keyboard arrays for selection by minimized movements of the mouse." System designers should be on guard that such a patent is lurking in the shadows.

Finally, Abstract 10 impressed me as a novel and unique application for a mouse and computer display. The

| | |
|---|---|
| Patent Number | 5,095,303 |
| Issue Date | 1992 03 10 |
| Inventor(s) | Clark, Michael R.; Mustafa, Musa |
| State/Country | CA |
| Assignee | Apple Computer, Inc. |
| Title | Six degree of freedom graphic object controller |
| Abstract | A six degrees of freedom interactive display controller device is disclosed, comprising a hand manipulable housing unit having an opening for the passage of a mouse ball, two motion detectors for detecting the movement of the mouse ball and converting that motion to output signals controlling the translational movement of an object on the display of a computer, a finger-operated conveyor belt or roller for controlling the translational motion of the object with respect to a third translational axis, a first finger-controlled mechanism, such as a wheel and motion encoders, affixed to the housing for controlling the rotational motion (pitch) of the object with respect to a first one of the translational axes, a second finger-controlled mechanism affixed to the housing for controlling the rotational motion (roll) of the object with respect to a second one of the translational axes, and a third finger-controlled mechanism affixed to the housing unit for controlling the rotational motion (yaw) of the object with respect to a third one of the translational axes. All translation and rotation controls are operable to be physically moved in a direction which corresponds to the desired simulated direction of movement of the object on the display. The physical motion of each control is unbounded and the actual physical position of the housing unit is independent of the simulated position of the object on the display. |


| | |
|---|---|
| Patent Number | 5,008,847 |
| Issue Date | 1991 04 16 |
| Inventor(s) | Lapeyre, James M. |
| State/Country | LA |
| Assignee | The Laitram Corporation |
| Title | Cursor-selected keyboard keys displayed on the computer screen for entering alphanumeric characters and instructions, particularly for creating computer-aided design and drafting patterns |
| Abstract | The conventional keyboard is replaced by a virtual keyboard pattern on the computer screen by this invention. Selection of keystrokes is made by a mouse, or the like, positioning a cursor at a desired key for keyswitch selection. This manner of selection of preformed patterns available from the computer, such as alphanumeric characters and computer commands is particularly advantageous in a computer-aided drafting and design system. Thus, notations and lettering need not be formulated by analog movement of the mouse, but can be selected digitally from the computer store by keyswitching. The resulting equipment therefore eliminates the conventional keyboard but not its operational advantages thereby permitting full computer operation with a mouse or the equivalent. Several features of the invention are provided for more convenient and more rapid operation, such as the replacement of the keyboard arrays as the computer changes modes of operation, and the placement of more frequently used keystrokes on the keyboard arrays for selection by minimized movements of the mouse. |


| | |
|---|---|
| Patent Number | 4,995,717 |
| Issue Date | 1991 02 26 |
| Inventor(s) | Damato, Bertil E. |
| State/Country | GBX |
| Assignee | The University Court of the University of Glasgow |
| Title | Device for moving eye campimetry |
| Abstract | The computer-assisted moving eye campimetric device is a computer display of a moveable test grid having a central reference spot. The patients eye is kept focused on the spot by giving the patient the task of keeping the moving spot in a circle by operating a hand-held mouse. With the patient's eye correctly focused on the reference spot, target elements in the field of vision are successively illuminated and the patient reacts by pressing a button on the mouse. Failure to react indicates impaired vision at that point in the patient's visual field. The results are held in the computer memory and plotted out as a map of the patients effective field of vision. |

# PATENT TALK

goal is to keep the patient's eyes focused on a spot while visual stimuli are presented around the periphery. This process permits a mapping of the patient's field of vision. It appears this is achieved by a relatively simple computer display and a conventional mouse. Using a mouse, the patient is required to keep the moving spot centered within a circle, thereby ensuring that the patient is continually focusing on, or near, the spot. The mouse button is used for the patient to signal the computer if and when he perceives one of the other periphery stimuli. This would appear to be a very efficient and cost-effective approach.

As we've seen here, the common computer mouse familiar to us all can take on many new and different forms. And with all these forms available, it will find increasing utility in new application areas. Keep an eye out in the near future for a wide variety of computer mice designed to meet specific needs. ❑

*Russ Reiss holds a Ph.D. in EE/CS and has been active in electronics for over 25 years as industry consultant, designer, college professor, entrepeneur, and company president. Using microprocessors since their inception, he has incorporated them into scores of custom devices and products. He may be reached at russ.reiss@circellar.com or 70054.1663@compuserve.com.*

## SOURCE

Patent abstracts appearing in this column are from the Automated Patent Searching (APS) database from:

MicroPatent
25 Science Park
New Haven, CT 065 11
*(203)* 786-5500 or (800) *648-6787*

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

425 Very Useful
426 Moderately Useful
427 Not Useful

# CONNECTIME conducted by Ken Davidson

The Circuit Cellar BBS
**300/1200/2400/9600/14.4k** bps
24 hours/7 days a week
(203) 871-I 988-Four incoming lines
Internet **Email:@circellar.com**

*A lot has been happening on the Circuit Cellar BBS recently. Our latest improvement has been to add color and fancy boxes to all our menus. If you were bored with plain old black and white, give us a call and give your eyes a treat. For regular callers who are offended at the idea of color, or for those with slow modems, color may be turned off altogether and the system will look the same as if a/ways has. No keystrokes or menu selections have been changed, so you don't have to relearn a whole new system.*

*In this* month's first thread, we look at the virtues of keeping a *lab notebook. It can be a pain, but it can be worthwhile.*

*Next, we talk about controlling hydraulic valves and when too much is too much.*

The third discussion centers around making a small, low-current *power supply. While you might think it's easy, try finding suitable parts.*

*The last topic deals with shifting the pitch of audio signals to speed up recorded speech* while keeping if legible.

## Lab notebooks

**Msg#:23138**
From: BOB PADDOCK To: ALL USERS

Are there any official guidelines for lab notebooks? I've heard that they had to be spiral bound with numbered pages so you could not insert or remove pages without it being noticed. And erasing was not allowed.

One of my colleagues said in a past job his employer, who lived on govermnent money, explicitly did *NOT* want any lab notes taken because the government could then subpoena the notebooks.

Better to have or not have?

I would like to have them so I can remember why we did something the way we did.

Comments?

**Msg#:23165**
From: WALTER BANKS To: BOB PADDOCK

Lab notebooks are sure useful when it comes to building up a chronology of events leading up to a legal action. I had one experience where I met with lawyers from both sides to generally discuss the merits of the case and I pulled out a bound notebook from my briefcase to take notes on the meeting. The lawyer from the other side asked how long I had been using lab notebooks. "About 20 years or so," I replied. He said that all he had to do was convince his client they had no chance of winning.

**Msg#:23507**
From: GARY **CORDELLI** To: BOB PADDOCK

What you are looking for you can get in most good office supply stores as an "engineer's notebook." At the very least they can order you some as they are definitely listed in all the office supply wholesaler's catalogs.

These notebooks are more expensive than loose-leaf or spiral notebooks, but they are indispensable for certain things. When it comes to patent submissions or other cases where chronology must be legally determined, they are the last word. To use them properly, write only in pen; if you make an error, cross out but do not erase the notes; never remove any page (they are numbered and should not be torn out to be photocopied or for any other reason unless you want to invalidate your evidence!); place the current date in the margin whenever you start a new day or a new page. Also, start each day's entries immediately following the previous day's (just draw a line to separate them). Do not skip to the next page and leave blank space. This is generally construed as an attempt to fudge the data later on by filling in blank spaces with stuff that you want to appear you accomplished earlier than you actually did!

As I said, these things are a necessity if you are working on patentable stuff. They are also good if you are just covering your butt in a legal confrontation because such "contemporaneous notes" are basically accepted as fact by a judge unless the opposition has real evidence to contradict you. I guess they figure if you took the time to keep track of all the details in this manner, then you must be right (?).

In any event, an engineering notebook will win all "ties" when it would otherwise be "your word against theirs." Nutty, huh? My advice is to use them, and never let an employer keep you from using a bound engineer's notebook. You should be suspicious of the intentions of anyone who does.

**Msg#:23613**
From: DAN HOPPING To: BOB PADDOCK

I'd like to add a few comments to Gary's I work in medical device design and the FDA gets picky about lab

notebooks. Not just for patent reasons but for reasons of safety as well. I think Gary mentioned it but let me stress that an Engineer's Notebook is not a spiral bound book. It is bound by stitching (as a traditional book would be). Gary is also correct about crossing out instead of erasing (a big no-no). The proper protocol for crossing out something is to put one line through it such that it can still be read (else you might be trying to hide something) then initial and date each occurrence of something being crossed out.

Also sign (full name) and date the bottom of each page in your notebook as you complete it. It is acceptable to start a new page even if you have not finished the prior page if you draw a large Z from just under the last entry down to the bottom of the page and initial and date the Z entry. But always use an engineering notebook. If you are honest and doing legal work then this is your only proof when push comes to shove. If your employer is not being honest and it's not obvious by the work you are doing and have documented this will be your ticket to freedom.

**Msg#:24186**
From: RANDY RASA To: BOB PADDOCK

I don't know; I have a hard time keeping a "real" lab notebook- with all the intricate and silly little rules it's just too much of a pain, and like most unpleasant tasks, it ends up being avoided.

I find it better to keep project notes in a more informal spiral notebook, entered as I see fit, and in whatever format I feel like writing. When working on a software project, I usually keep a "notes" file open in the editor-I find that typing notes as I program is easier than breaking away from the computer to go to the notebook and write something down manually (cuts down on writer's cramp, too). Then, when the project is complete I can print out the "notes" file and make it part of the project records.

So I guess it depends-if you're keeping a notebook to cover your rear in case of legal difficulties, by all means follow the "official" rules.

On the other hand, if your purpose is to help in recording the history of a project (the thinking behind decisions, experiments you've tried, etc.), then you might not want to be so formal about it. Just do whatever feels right.

My humble opinion..

**Msg#:24550**
From: PELLERVO KASKINEN To: BOB PADDOCK

You already got the pertinent poop from others, but I still want to bring up my own conclusions.

I have been using commercially available bound lab notebooks for ages. They come with numbered pages, with one white page and one pale yellow page each having the same number. The yellow page is perforated so you could tear it away. And there is a copy sheet. You are supposed to make two identical documents each time you write something. At the top you put the date and the topic. Three lines are available for all that.

You fill in the data, and it is not necessary to avoid white space if the rest of the procedure is followed: After you have filled the page, you tear off the yellow sheet, date and sign that and deliver it to another person who dates and signs it and safekeeps it. The idea here is that you can continue using your white pages for reference. Then, when any need to prove the authenticity and timing arises, you have the two copies available. If they remain identical and the safekeeper is sworn to truth and nothing else but, then you have your proof.

Now to the personal side of this. I have kept the books, but I have considered it a waste of paper to make the copies. I have not considered myself as an inventor of anything I would try to patent with this kind of timing dispute possible. So I have just numbered the pages like 20, 20a, 21, 21a and so on. But the times that I have had to go back to those calculations, test results, speculations, meeting notes, and so on have been personally very valuable.

I guess the all-important decision is, do you anticipate or even now pursue some litigation-prone activities like inventing something patentable. If so, by all means you should go the extra mile in following all the rules. And if you do not, you probably still should, because someday somebody else may force you into such a situation.

## Controlling hydraulic valves

**Msg#:20320**
From: JIM KELLEHER To: ALL USERS

I need help in designing a circuit that can control a hydraulic proportional flow control valve. The valve in mind is a HydraForce PFR70-33x-F12V valve. It is controlled by a PWM signal. What I have in mind is a stand-alone black box circuit that can be programmed by a microcontroller. An "up" input to the micro will cause the duty cycle on time to increase one percent, while a "down" input will cause the duty cycle to decrease one percent. I need a start-up duty cycle of 0% on time and 100% off time or vice versa.

Is there anything out there (special integrated circuit, already-designed circuit, etc.) that I can use for production purposes. Or do you have another idea, possibly getting rid of the black box and using the microcontroller to create the

PWM signal without interrupts affecting it (possibly using a latch and counter)?

Thank you for your time and any suggestions.

**Msg#:20324**

From: GARY CORDELLI To: JIM KELLEHER

If I understand your problem correctly, you might want to check out some of the off-the-shelf variants on the old 8031 architecture. Some of the chips from Signetics (now Philips Semiconductor) and others have PWM functions built right in to the microcontroller. Signetics has an 87C752 (small 2K EPROM version) with ii-channel 8-bit ADC in and PWM output, and an 80C552 or 87C552 (ROMless and 8K EPROM versions) with 8-channel ADC inputs and two PWM outputs. Also an 87C054(16K EPROM) with 9 PWM outputs. This may be of help in your case.

**Msg#:22536**

From: TIM MCDONOUGH To: GARY CORDELLI

The "genuine" Intel 80C51FA series parts have PWMs as well. They might be easier to get a hold of in small quantities.

**Msg#:20347**

From: MICHAEL SWARTZENDRUBER To: JIM KELLEHER

Or you might also consider the 68HC 11 chip which has a register structure (capture compare) that is very nice for precise control over a PWM signal.

**Msg#:20443**

From: JIM NELSON To: JIM KELLEHER

Actually, you've got to supply some more information, Jim.

1. What is the nature of your up/down signal? Do you want the duty cycle to change by 1% for each single up/down pulse received?

2. What logic state turns the valve physically off?

3. Does your black box have to supply high current? Is the PWM signal amplified before it reaches the valve? Does it have to be!

4. What's going through the valve? Just curious.

5. What microcontroller is controlling the black box, or will this vary by customer (i.e., are you producing this black box for a specific single application or for the valve manufacturer's product line)?

**Msg#:20926**

From: JIM KELLEHER To: JIM NELSON

Well Jim, you have asked a lot of fine questions. I guess I asked the wrong question myself. Here's a little detail and background of what I need.

The place I work for recently hired me to place computer controls on the products they produce. I just graduated from college, where I specialized with a computer option. I can write assembly and hook up TTL I/O, address decoders, and so forth, but I do not have the knowledge to hook up large voltage (12-V) high current (3-A max) loads. Everything in school ran off 5 V, 20 mA max I/O.

The valves I am working with are standard hydraulic valves (according to the literature). They only have two electronic contacts, which I believe are the ends of a coil used to move the valve. The literature says a variable DC voltage O-12 V can be used to open the valve (it is normally closed), where at 0 V it is fully closed, at 12 V it is fully opened, and in between the voltage is proportional to how far it opens.

When used in this manner, the valve is inconsistent in operation depending on how long it's been running. In the morning, 6 V across the terminals would produce a flow of 6 gpm, and if it's been running for 4 hours straight, the 6 V may now produce only 5.5 gpm. So the company sells a dither control unit that outputs a PWM signal around 1 10–130 Hz. Such a signal does not cause the coil to heat up and you get much better control. I don't know how the physics of this valve work, but I am guessing that at 0% duty cycle, the valve is fully off, at 100% duty cycle, it is fully open, and in between who knows what it's doing, but I have a feeling it's going to be buzzing at 110 Hz.

So now back to my dilemma: I would like to set up a test control unit to see how this valve works. I am planning to use a microcontroller or possibly a frequency generator with a changeable duty cycle at 5 V max to create the PWM signal. From there I need the circuitry, possibly a MOSFET switch, that is powered by a 12-V supply and capable of at least 3 A and is also capable of emulating the PWM signal at the 100+ Hz frequency. Like I said, in school every I/O dealt with 5 V and very low current.

For your questions:

1) Right now, I am thinking about using a PWM IC to create the output signal; I think the resolution of these chips (I have in mind the 68HC11K4) should work well.

2) I think a 0% duty cycle.

3) I would like the microcontroller board to have the circuit that handles the 12-V, 3-A output and there could be leads that could be possibly 20+ feet long to the valves.

4) Just hydraulic fluid in lines used to drive hydraulic motors.

5) I am planning on the Motorola 68HC11K4.

**Msg#:20992**

From: PAUL PETERSEN To: JIM KELLEHER

Here are some thoughts about your dilemma [I have 30 years experience as a super-tech in R&D). It sounds like

you're making the test much more complicated than is called for. You want to test the valve, not design a test bed, right? My first thought after reading your last message is why not get a standard pulse generator and a DC amplifier? Years ago I would have gotten my hands on a DataPulse Inc generator which outputs typically 1 V peak driving a DC amplifier. I don't want to get too carried away here, but is this along the right lines?

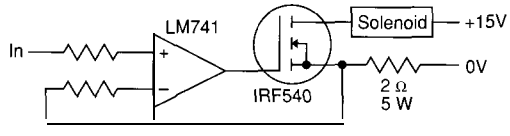
**Msg#:22557**
From: PELLERVO **KASKINEN** To: JIM KELLEHER

I just wanted to add one quick note about the supposed drift in the control valve response characteristics.

The probable reason is the heating of the coil. And its effect is most likely apparent in a much shorter time than the 4 hours you mention. The thermal time constant of the coil is, in my estimate, only on the order of a couple of minutes. Anyway, as long as the magnetic effect is proportional to the coil current and the current varies due to a fixed voltage into the changing coil resistance, you are going to see a change in the valve opening and therefore in the flow.

Now, the flow is by nature very nonlinear, but I do not want to get into that. The valve manufacturer has done all he could for linearizing it. What you can do, though, is to drive the solenoid with controlled CURRENT rather than with controlled VOLTAGE. That would go a long way toward keeping the response from changing.

To implement a constant-current drive, you can do something like the following sketch attempts to show:




The principle is that with 2 V in, the 2-ohm resistor will develop a matching 2 volts and thereby the solenoid also sees 1 ampere of current, regardless of the coil resistance, as long as there is enough supply voltage so that the FET does not go into saturation. Of course, this circuit is DC operated and has a poor efficiency. The FET has to be equipped with a sufficient heat sink to dissipate whatever

heat the worst-case conditions might impose. Let's say the cold state resistance of the solenoid coil is 10 ohms. Then, at 1 ampere we would have 2 volts on the 2-ohm resistor, 10 volts on the coil, and 3 V remaining on the FET. That would mean 3 W of dissipation. On the other hand, with 0.6-A current we would have 1.2 V over the resistor and 6 V over the coil, leaving 7.8 V over the FET and over 4 W of dissipation. You would need to calculate the worst case from the actual numbers.

I hope this gets you started. I also agree with the suggested pulse generator technique. That dithering or pulse width modulation could actually be done with the same IRF540 or similar FET. Just remember to add a free-wheeling diode over the solenoid! Otherwise, you could say goodbye to the FET, even though the PWM principle is supposed to be so great in efficiency. Ultimately, you could combine both actions by feeding the pulse-width-controlled signal into the In terminal of the above circuit. Just like earlier, if you try to modulate the drive to the solenoid, a free-wheeling diode is required.

**Msg#:25398**
From: PETER **HOLZLEITNER** To: JIM KELLEHER

These valves should be driven by a controlled current (DC) to achieve repeatability-the differences you are seeing are most probably due to heating in the coil, and the associated change in coil resistance will change the drive current if you control the voltage. Check out a power op-amp like the LH0101 (Maxim et. al.) or Burr-Brown's OPA 5xx series. The dithering you mention eliminates (or at least greatly reduces) hysteresis effects, especially around zero flow, caused by mechanical friction in the valve. The dithering causes the actuator to oscillate slightly, thus "shaking free" from friction. A 3% dither would be typical.

## Small, low-current power supply

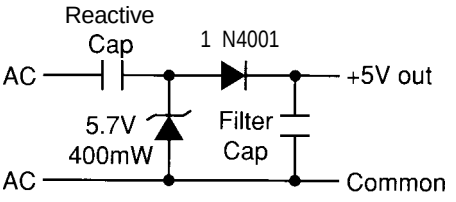
**Msg#:22384**
From: JOHN **MUCHOW** To: ALL USERS

I've built some devices to help me sequence tungsten lights in my studio, but would like to use the line AC power running through them to power them instead of batteries. The units draw microamps when the lights are off and approximately 5 mA when on.

My problem is not building the AC-DC power supply, but finding components that let me build a small 10-mA supply! All the step-down transformers I've found are rated for at least 100 mA and are quite large (about 1.5" x 1.5" x 1.5"). All of the other components I can get in a small form factor. Can I use some other type of transformer or can someone else recommend a manufacturer of very small transformers? I can't use those single-chip power supplies because they don't provide isolation, which I definitely need. Thanks for any help you can give.

**Msg#:22441**
From: DAVE TWEED To: JOHN **MUCHOW**

For very low-current supplies, a common technique is to forego a transformer altogether and use a capacitor instead as a reactive element to drop the voltage to the level you need. The key to the circuit design is that you must allow current to flow in both directions through the capacitor; otherwise it will charge up to the peak voltage of the AC line and stop passing AC current. Here is a typical circuit:




Let's say you want 10 mA out. The capacitive reactance required is 120 V/10 mA = 12k ohms. Using Z = 1/2*pi*f*C, at 60 Hz this works out to be 0.22 µF. Be sure it is rated for at least 200-250 V. (For what it's worth, this is the circuit that I've found in most X-IO modules that I've opened up.)

Obviously, this circuit doesn't provide any isolation, either. You need to ask yourself whether you can provide the necessary isolation (presumably between the AC line and your control input) elsewhere in the circuit. Can you use an optoisolator at the control input, for example? If not, then you are going to need a transformer after all (perhaps a wall wart).

**Msg#:22478**
From: JOHN **MUCHOW** To: DAVE TWEED

I definitely need the isolation, but like your circuit. I'll check if I can add optoisolation after the control input, but I'm pretty sure no current is available from any of the triggering devices to turn on the optoisolator.

A regulated wall transformer supply was my next choice if another low-current solution wasn't available. Thanks for the circuit.

**Msg#:22911**
From: DAVE TWEED To: JOHN **MUCHOW**

There *are* ways to provide isolated excitation to a contact closure. It all depends on how much you want to throw at it (before you throw in the towel).

For example, you can take a small pulse or audio transformer (that has the required isolation between primary and secondary] and put the switch across the secondary winding. Now, when the switch is open, the primary will look like an almost-pure inductance, and when the switch is closed, it will look like an almost-pure (small-value) resistance.

The trick is designing a circuit that can detect this change. You drive the primary with a small AC voltage (or current) and then detect either the magnitude change or phase-angle change in the resulting current (or voltage). Keep in mind the potential time delay in the detection circuit, especially if this is being used for flash lamps. You said tungsten before, so I'm assuming you're looking for relatively low speed on/off control.

Years ago, I saw an AC power relay that could be directly controlled by an isolated contact closure. It had a 120-VAC coil, but it also had some sort of auxiliary winding that came out to two terminals. The relay would pull in only if those terminals were shorted. No more than 12 VAC (isolated] would appear on the terminals when they were open, so low-voltage wiring could be used for the switch.

**Msg#:22967**
From: JOHN **MUCHOW** To: DAVE TWEED

I like it! Even better, I understand it !

I think I'll play with it. I don't want to spend a lot on components, but I'm willing to do a lot of testing/research to save the money/space required for the circuitry. I don't have any idea how to detect those changes [other than a comparator for the voltage change, but that sounds unwise since there's bound to be some electrical noise floating around to mistrigger it), but it gives me a good excuse to dive back into "The Art of Electronics." 1'11 order a couple of small transformers from Digi-Key and start playing. Thank you for your time on this one, I really appreciate it!

## Pitch Correction

**Msg#:18773**
From: BRAD **SKILLMAN** To: ALL USERS

I'm looking for a way to alter the pitch of a digitized audio recording so that when the audio passage is played

back at double (2X) or half (0.5X) the normal speed (altering the pitch), the voice will sound "normal" but just faster or slower. This feature would be useful for trying to understand someone who is speaking very quickly and the recording must be slowed down with the correct pitch in order to understand him or her. I believe that combining this feature with audio compression is probably best suited for a DSP, but I'm not sure what the principles are behind pitch correction. If anyone could help in this area or knows where I could get further information, I would appreciate it if you would leave me some Email. Thanks in advance for your time.

**Msg#:18776**
From: DAVE TWEED To: BRAD **SKILLMAN**

The basic idea of "pitch correction" is really not to modify the pitch in the first place.

You break the data stream into conveniently sized windows (say, 20-50 ms or so), then replicate windows if you're slowing things down or throw windows away if you're speeding things up. The ear determines "pitch" based on the data within a window, while "speed" is controlled by the relative window rate between input and output. Signals that have periods significantly less than the window size will appear to be unmodified (e.g., the pitch of the voice), while signals that have periods longer than the window size (words and syllables) will appear to be faster/slower. If the window size is too small, low-pitched voices will be badly distorted; if it is too large, you'll get a Max Headroom type of stuttering effect.

The main problem is that the replicating/discarding process produces discontinuities in the analog signal at the window boundaries, which are perceived as noise (or "burbling") by the listener. To a certain extent, simple filtering of the output signal (either while it is still in the digital domain, or after conversion to analog) will smooth over most of the bumps. You can also make the windowing process adapt itself to, say, zero crossings of the original signal to minimize the energy content of the discontinuities.

I guess the term "pitch corrector" comes from when the circuit is used with analog tape recorders, on which you can't separate the window rate from the sample rate (effectively). In that case, a shift register (FIFO memory) is used to return the sample rate within the window back to its correct value. Some versions use an analog charge-coupled device (CCD) "bucket brigade" to implement the FIFO, although this is rare now.

There are a number of ways to control when to discard/replicate a window to get the desired output speed relative to the original. Simple integer ratios are easy: replicate every window to get a 2: 1 slowdown, every other window to get 1.5:1, and so forth. More complicated patterns can get you the "in-between" rates, and there's a reasonably general way to generate them if you need them.

Although this algorithm can be implemented purely in hardware, a DSP would make a lot of sense given the other things (compression, decompression, filtering) you also want to do.

**Msg#:18888**
From: JIM WHITE To: BRAD **SKILLMAN**

I remember a product that did exactly what you describe some years ago (10?5?). I believe the intended market was for "books on tape," so they could be more easily understood at faster-than-normal speed. You might check with organizations for the blind to see if it is still around.

**Msg#:22274**
From: BRAD **SKILLMAN** To: DAVE TWEED

You mentioned that a primary problem of replicating/discarding 2530-ms "windows" produces discontinuities in the analog signal at the window boundaries, which are perceived as noise (or "burbling") to the listener. I have run into the same "burbling" problem that you described after doing some research into the Time-Scale Modification of speech and implementing an algorithm based on a Synchronized OverLap Add (SOLA) method utilizing a cross-correlation technique.

Do you think that a simple FIR filter would possibly get rid of this burbling noise? The speech actually doesn't sound half bad and is understandable if you ignore the recurring noise.

My other question is which DSP manufacturer would you recommend based on your experience with this type of audio processing?

**Msg#:22442**
From: DAVE TWEED To: BRAD **SKILLMAN**

No, I expect the SOLA algorithm is already reducing the burbling more than a simple filter could. Also, there's a certain amount of irreducible burbling that you are going to get whenever you do this kind of time-scale modification.

You probably now know more than I ever did on this subject. It sounds like the cross-correlation that you are doing will minimize the distortions at the window boundaries as much as can be expected. The only step beyond this would be to oversample the audio signal so you get a finer time scale on the cross-correlation.

Another way to achieve essentially the same thing would be to interpolate between the samples on either side of a zero crossing to locate the exact time of the real zero crossing and use that as the offset for the overlap add. This

# CONNECTIME

could get really complex, and you have to pay attention to the CPU horsepower required. Scan the literature for sample-rate-changing filters (often called "decimation" filters).

As for DSP selection, the TMS320C2x from TI, the DSP56001 from Motorola, and the ADSP-210x from Analog Devices are all very similar. I use the ADSP-2101 myself because I think it has the most flexible architecture of the three and the best hardware support for serial I/O, but be warned that I'm not very happy with the C compiler they have for it, and I do all of my programming for it in assembly. If you prefer to work with HLLs, I believe there are more (and better) compiler implementations for the other two chips. The other advantage of the Analog Devices family is the very low cost of their ADSP-2105.

*We invite you call the Circuit Cellar BBS and exchange messages and files with other Circuit Cellar readers. It is available 24 hours a day and may be reached at (203) 871-1988. Set your modem for 8 data bits, 1 stop bit, no parity, and 300, 1200, 2400, 9600, or 14.4k bps. For information on obtaining article software through the Internet, send Email to info@circellar.com.*

## ARTICLE SOFTWARE

Software for the articles in this and past issues of *The Computer Applications Journal* may be downloaded from the Circuit Cellar BBS free of charge. For those unable to download files, the software is also available on one 360K IBM PC-format disk for only $12.

To order Software on Disk, send check or money order to: The Computer Applications Journal, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your VISA or Mastercard and call (203) 8752199. Be sure to specify the issue number of each disk you order. Please add $3 for shipping outside the U.S.

## I R S

425 Very Useful 426 Moderately Useful 427 Not Useful

# STEVE'S OWN INK

## Interactive Harmony-An Installer's Market

home control apparently means different things to different people. A sampling of the BBS messages around here suggests a plethora of applications. There are people who want a system that simultaneously closes five sets of drapes while verbally reciting inside/outside temperature differential to prove that heat is actually being conserved. And, when they aren't physically directing control activities, they want their HCS to anticipate their every need. Music, TV, lights, microwave oven, and so forth should all switch on at the right times as if by magic.

From an engineering prospective, any or all of this is possible. The Circuit Cellar Home Control System has, in fact, been structured so that the knowledgeable operator can execute such control. Being able to exercise automatic control doesn't make the exercise practical, however. Energy conservation is laudable, but at about $600 per drapery controller, the return on investment is some time late in the next century. Similarly, while the concept of an AI-based system is intriguing, I can't help but wonder if it could become as bothersome as any human following at your heels.

To me, home control is meant to supplement the convenience and security of living in a house, not eliminate the need for being in it at all!

The problem with home control is not the design-it is with the marketing. Engineers and technical people who understand industrial control have little problem recognizing the value of an HCS. For them it is merely a matter of prioritizing those elements needing automatic monitoring or management and doing it as they can afford it. Because they understand computer supervision and interfacing, they rarely waste time attempting the impractical.

The problem with this highly technical audience is that it is a limited market. Covering the design costs of a continually expanding and well-supported Circuit Cellar HCS requires broad market appeal.

That brings me to the current dilemma. When I talk to most people about home control they picture either Robbie the robot or a talking toaster. It is only after spending considerable time erasing eco-scare misconceptions and TV-deadened brain blocks that they come to understand that an HCS is a new kind of sophisticated appliance designed to facilitate the ultimate level of interactive harmony. Of course, they would unquestionably trust its utilization if I would install it too.

Unfortunately, it is true. To adequately penetrate this new market requires a one-on-one approach. Someone has to evaluate the facility, present a practical combination of security and convenience benefits, and then custom install it. This is what oil and gas burner companies, alarm installers, and commercial home builders do every day.

The difference between them and you is the timing of the information and the formulation of the product. Eventually the large suppliers that deal with these people will introduce systems they can install. I suspect it will still be a few years, however. In the meantime, there is an opportunity for enterprising individuals now who can translate engineering-speak into a new home control appliance.

I reiterate! This is not a matter of if, but rather when, it will happen. Just like central heating, coordinated security and convenience control in homes is inevitable. Your slice of the pie depends on a good sense of timing.

Steve