HA & BC

# CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

#63 OCTOBER 1995

## ROBOTICS

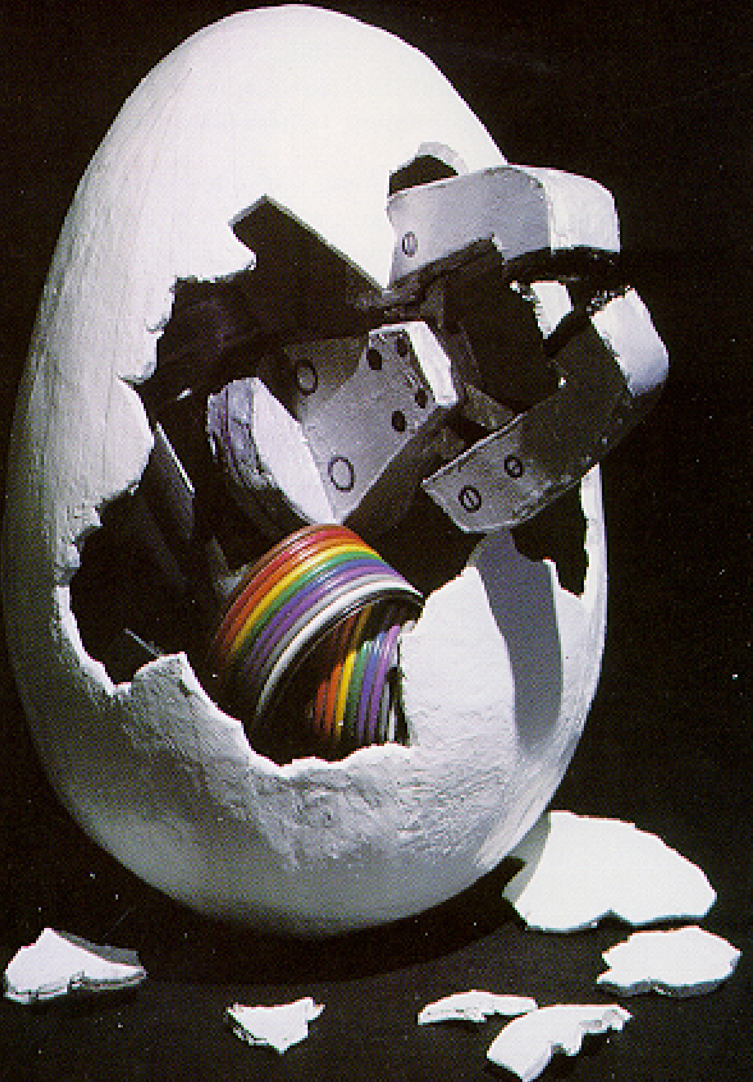
Fire-Fighting Robot

Robotic Proximity Sensors

Detecting CO in the Home

Solid-State Barometer

SPECIAL BONUS SECTION: HOME AUTOMATION & BUILDING CONTROL



$3.95 U.S.
$4.95 Canada

# EDITOR'S INK

## Birth of the Firebot

While there are quite a few robot contests around (perhaps the most famous being the Micromouse competition held all over the world), we have a yearly contest that takes place just down the road from our editorial offices and is growing in popularity. The goal of the Fire-Fighting Home-Robot Contest is for a robot to seek out and extinguish a flame located in a "home." Judging is done on how quickly the robot accomplishes its task, whether the robot is tethered or autonomous, and whether it can deal with "furniture" obstacles.

Our first article this month looks at one of the entries in this years contest. It's not the fastest beast, but it's one of only four from a field of dozens to even finish. The designers have come up with some unique solutions to common problems, so be sure to check it out.

When it comes to building a robot, you don't necessarily need to start from scratch. Having an off-the-shelf base on which to build the eyes and brains of the robot can save a lot of time. In our next article, the author retrofits a kid's toy with enough smarts to make Fido jealous.

Although a human adds the brains to a race car, the engine still needs some internal smarts to squeeze the most performance from each piston stroke. In the second of a three-part series, our next feature article describes the ignition electronics and shielding in the Engine Control System.

Finally, we check out an often-overlooked mode of the popular M68HC11 processor: bootstrap mode. It can be used to enable the chip to test itself with no external memory.

In Home Automation & Building Control this month, we start by looking at what it takes to detect deadly carbon monoxide in the home. Next, we continue our detective work with the X-10 Spy. Finally you can see just what is being transmitted onto the power lines. In the last two features, we cover even more sensors: one for measuring atmospheric pressure, and another for measuring temperature (but this one sends the results back to a standard serial port).

In our columns, Ed continues his look at Virtual-86 mode on the '386SX, Jeff adds position sensing to his motor control module, and Tom explores the latest in digital television support chips.

[illegible]

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

TECHNICAL EDITOR
Janice Marinelli

EDITORIAL ASSISTANT
Beth Andrix

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
Rick Lehrbaum
Russ Reiss

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
Lisa Ferry

PRODUCTION STAFF
John Gorsky
James Soussounis

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

PUBLISHER
Daniel Rodrigues

PUBLISHER'S ASSISTANT
Sue Hodge

CIRCULATION MANAGER
Rose Mansella

CIRCULATION ASSISTANT
Barbara Maleski

CIRCULATION CONSULTANT
Gregory Spitzfaden

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Dan Gorsky

CIRCUIT CELLAR INK®, THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) is published monthly by Circuit Cellar Incorporated. 4 Park Street. Suite 20, Vernon, CT 06066 (860) 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 Circuit Cellar INK Subscriptions, P.O. Box 696, Holmes, PA 19043-9613 or call (800) 269-6301.
POSTMASTER. Please send address changes to Circuit Cellar INK, Circulation Dept., P O. Box 698, Holmes, PA 19043.9613

Cover photography by Barbara Swenson
PRINTED IN THE UNITED STATES

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

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

| NORTHEAST & MID-ATLANTIC | SOUTHEAST | MIDWEST | WEST COAST |
|---|---|---|---|
| Barbara Best | Christa Collins | Nanette Traetow | Barbara Jones & Shelley Rainey |
| (908) 741-7744 | (305) 966-3939 | (708) 357-0010 | (714) 540-3554 |
| Fax: (908) 741-6823 | Fax: (305) 985-8457 | Fax: (708) 357-0452 | Fax: (714) 540-710 |

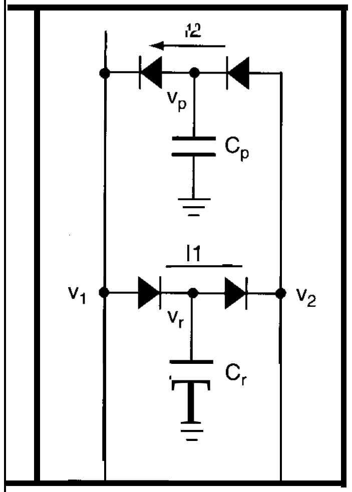
# INSIDE ISSUE 63

# READER'S INK

## CIRCUIT ANALYSIS SOLVED

In INK 61's ConnecTime, Rich Vitucci presented a problem he was having with a capacitor tilt sensor. Although James Meyer and Pellervo Kaskinen suggested solutions, Rob Mock has already solved this problem.

Rob's solution is for a square-wave excitation and neglects a few things, so it is a first approximation.



The quad-diode capacitance sensing circuit is pictured in the first figure. As you can see, $V_1$ and $V_2$ are the DC values for the AC signals $v_1$ and $v_2$. The reduced signals adjacent to $v_1$ and $v_2$ do not represent actual signals but are used here to depict the high and low values of $v_p$ and $v_r$, which are different from $v_1$ and $v_2$ by the voltage drop across a diode $(V_d)$.

Because the circuit is bounded by capacitors and the noninverting input to an op-amp on one side, and a capacitor and ground on the other side, we assume that no net current traverses the circuit. However, current circulates inside the subcircuit itself (i.e., $I_1 = I_2$).

The anticipated voltage signals for $v_1$ and $v_2$ are shown in the second figure. The resulting charge passed on each positive voltage transition would be

$$(((V_{peak} - V_d) + (V_1 - V_2)) + (V_{peak} - V_d))C_r$$

from $v_{rl}$ to $v_{rh}$ coinciding with $I_1$, and

$$(((V_{peak} - V_d) + (V_1 - V_2)) + (V_{peak} - V_d))C_p$$

from $v_{pl}$ to $v_{ph}$ coinciding with $I_2$.

The corresponding charge for each negative transition is

$$(((V_{peak} - V_d) + (V_1 - V_2)) + (V_{peak} - V_d))C_r$$

from $v_{rh}$ to $v_{rl}$ coinciding with $I_1$, and

$$(((V_{peak} - V_d) + (V_1 - V_2)) + (V_{peak} - V_d))C_p$$

from $v_{ph}$ to $v_{pl}$ coinciding with $I_2$.

The effective currents from this pulsing charge are

$$I_1 = ((2(V_{peak} - V_d) + 2(V_1 - V_2)) + 2(V_{peak} - V_d))C_r f$$
$$I_2 = ((2(V_{peak} - V_d) - 2(V_1 - V_2)) + 2(V_{peak} - V_d))C_p f$$

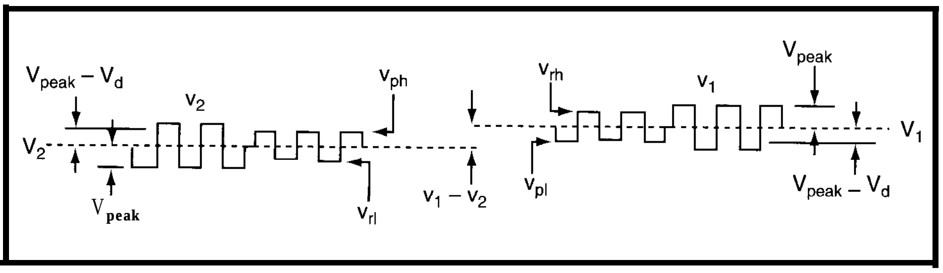
where $f$ is the signal frequency.

Equating the currents and solving for the potential difference between $V_1$ and $V_2$,

$$(4(V_{peak} - V_d) + 2(V_1 - V_2))C_r f = (4(V_{peak} - V_d) - 2(V_1 - V_2))C_p f$$
$$(2(V_{peak} - V_d) + (V_1 - V_2))C_r = (2(V_{peak} - V_d) - (V_1 - V_2))C_p$$
$$(V_1 - V_2)C_p + (V_1 - V_2)C_r = 2(V_{peak} - V_d)C_p - 2(V_{peak} - V_d)C_r$$
$$(V_1 - V_2) = 2(V_{peak} - V_d)(C_p - C_r)/(C_p + C_r)$$
$$(V_1 - V_2) = (V_{pp} - 2V_d)(C_p - C_r)/(C_p + C_r)$$

These equations neglect differences in individual diode drops, diode-stored currents and leakage, load currents, and pulse duty cycle and rise times. In properly designed circuits, these effects should be negligible.

Edward **LaBudde**
Westlake, CA



## PIC PENCHANT?

What is it with your writers' infatuations with PIC processors? I keep reading about using a PIC, and find myself wondering why the author didn't use something else and save money. Previously one author used two PICs when I would have used one 68705, at the same or lower cost. "A PIC-based Motor Speed Controller" by Chuck McManis *(INK 60)* takes the cake. He uses a PIC to replace a 70¢ servo amp!

Had he checked things out, he would have found that the NE544 servo amp from Signetics does everything his PIC does plus it has exponential or linear rates, adjustable deadband, and a 500-mA drive built in. It is also capable of driving external transistors for more current drive. Many of the current R/C servos on the market use this part.

So why PIC and program when you can buy a part off the shelf that does the job?

Robert **LaMoreaux**
Ann Arbor, MI

*Actually, if you do a survey of the processors used in projects here, you'll find a pretty even mixture. For example, in this issue, we have projects using the 80386, 68HC705, 68HC11, 68HC16, and two PICs.*

*One of the attractive features of the PIC is its availability. Jeff started using the 'HC05 in several of his projects until he found he couldn't get any parts from anyone. He's since used PICs for a number of projects because you can even get them from Digi-Key.*

*As for using a PIC (or any other processor) in place of a dedicated part, keep in mind that the processor may also be used to perform many other functions as well, so it's not necessarily an apples-to-apples comparison. You have to assess the project as a whole to determine when to use a dedicated part and when to use a processor.*

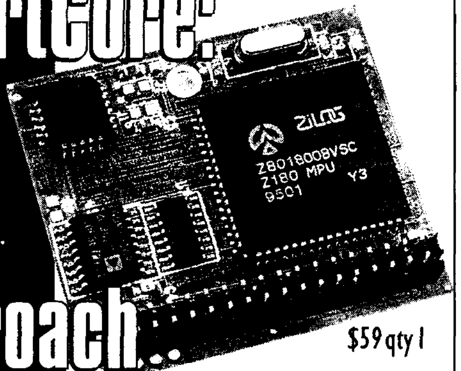
*—Editor*

## Contacting Circuit Cellar

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

Mail: Letters to the Editor may be sent to: Editor, Circuit Cellar INK, 4 Park St., Vernon, CT 06066.
Phone: Direct all subscription inquiries to (800) 269-6301. Contact our editorial offices at (860) 875-2199.
Fax: All faxes may be sent to (860) 872-2204.
BBS: All of our editors and regular authors frequent the Circuit Cellar BBS and are available to answer questions. Call *(860)* 871-1988 with your modem (300–14.4k bps, 8N1).
Internet: Electronic mail may also be sent to our editors and regular authors via the Internet. To determine a particular person's Internet address, use their name as it appears in the masthead or by-line, insert a period between their first and last names, and append "@circellar.com" to the end. For example, to send Internet E-mail to Jeff Bachiochi. address it to jeff.bachiochi@circellar.com. For more information, send E-mail to info@circellar.com.

◆SmartCore™ ◆the ◆New ◆Approach...



$59 qty 1

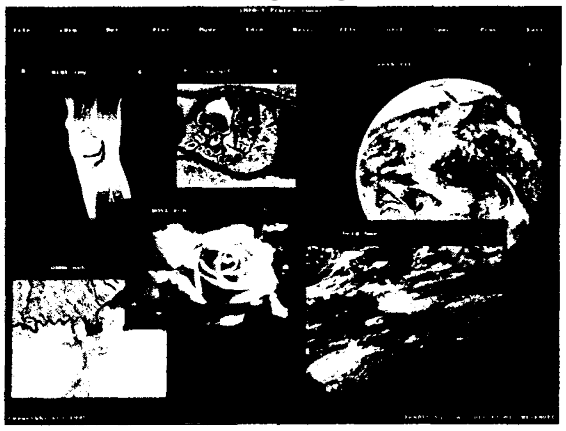
***$99 Evaluation Kit.***
See how the SmartCore approach can save you time and money. You'll receive a fully featured SmartCore, trial version of Dynamic C software, evaluation board, accessories, and documentation. Call today for immediate delivery.

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

Z WORLD ENGINEERING
1724 Picasso Ave.
Davis, CA 95616
916.757.3737
916.753.5141 FAX

#104

A Serious Imaging Solution



IMPACT Professional is a complete image analysis system that includes a broad range of cross-*discipline* tools grouped *into eight* separate *processing environments* These *modules include*

- Image Processing System
- 2-D Analysis System for Edge. Shape and Surface Characterization
- Object Counting. Analysis and Sorting System
- Neural Network Graphic Recognition and Classification System
- Image Comparator and Anomaly Detection System
- 3-D Mapping System including Stereo- and Random-dot Projections
- Fractal Analysis System for Point. Line. Edge and Surface Processing
- Color Mapping System which handles Greyscale. Pseudo and 24-bit RGB

SYSTEM REQUIREMENTS: PC/AT or compatible 386,486 or Pentium, with at least 16 MB of RAM and a harddisk DOS 3 1 or higher Uses a flat memory model with its own extender and Virtual Memory Manager capable of addressing 3 gigabytes of memory A super VGA video card

TARDIS Systems
PO Box 1251
Los Alamos, NM 87544 U.S.A.

**FREE DEMO**

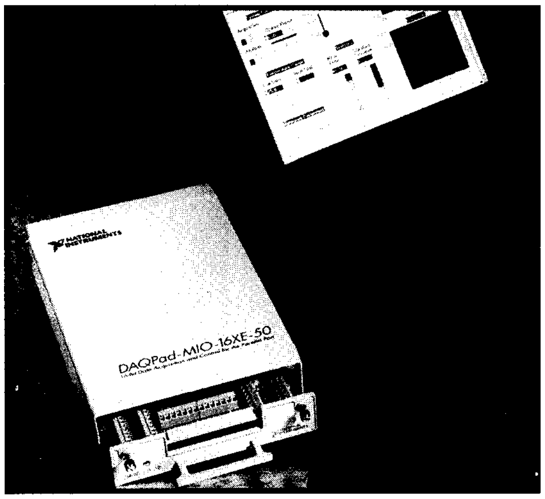
Phone: (505) 662-9401
Fax: (505) 662-6780
Technical Support: (505) 662-5623

#105

# NEW PRODUCT NEWS

Edited by Harv Weiner

## PORTABLE DATA ACQUISITION SYSTEM



A high-resolution, multifunction, portable data-acquisition box that communicates through the parallel port with PC, XT, AT, and compatible computers has been announced by National Instruments. The **DAQPad-MIO-16XE-50** can be used with notebook computers or any PC with a parallel port in a variety of portable applications.

The DAQPad-MIO-16XE-50 incorporates the company's custom system-timing controller ASIC for counting and timing-related functions. The unit features a 16-bit ADC with a sampling rate of 20,000 samples per second, 16 single-ended or 8 differential inputs, and enhanced timing and triggering capabilities. Also incorporated are a programmable gain up to 100, two 12-bit DACs with voltage outputs, one constant-current source for powering resistance temperature detectors (RTDs), eight lines of TTL-compatible digital I/O, and two 24-bit up/down counter/timers for timing I/O.

The unit is compatible with the Enhanced Parallel Port (EPP) standard defined by IEEE 1284 and works with fast EPP ports as well as the original Centronics or standard parallel ports (SPP). A second parallel port connector is also included for a transparent pass-through connection to an SPP device such as a printer.

The DAQPAd-MIO-16XE-50 includes a detachable terminal block with screw terminals for convenient, compact signal connection. The block is built into the enclosure to eliminate the need for additional cables or external termination accessories. An AC power adapter is included but the unit can be powered by an optional rechargeable battery pack or any 9–42-VDC source.

Driver software for DOS and Windows is included, and the unit is designed to integrate seamlessly with the company's LabVIEW and LabWindows/CVI application software.

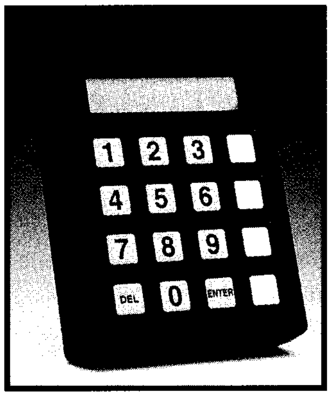
The DAQPad-MIO-16XE-50 sells for $1795.

National Instruments
6504 Bridge Point Pkwy. • Austin, TX 78730-5039 • (512) 794-0100 • Fax: (512) 794-8411
E-mail: infoQnatinst.com • WWW: **http://www.natinst.com/** **#500**

## LARGE-KEY INDUSTRIAL TERMINAL

The **QTERM-R30 Industrial ASCII Terminal** is a low-cost, countertop or wall-mounted operator-interface terminal that can operate in multidrop systems (RS-485 or RS-422).

Features of this terminal include either full-duplex (RS-232, RS-485, or RS-422) or multidrop operation (up to 32 terminals per host port), a large-character 2 x 16 backlit supertwist LCD display, 16 extra-large tactile steel-dome keys, and user-programmable LEDs.



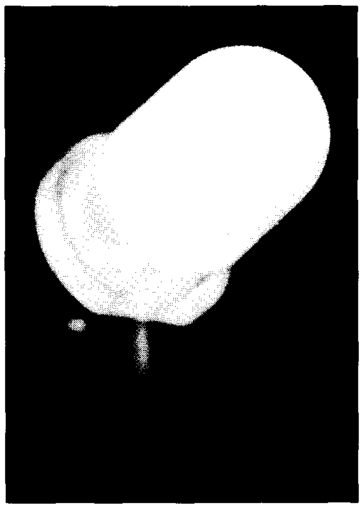
Every key can be programmed to send a character or string, or to control the terminal (e.g., display a menu) when pressed and/or released. User-programmable LEDs provide easy and flexible status indication. A tilted stand is available for countertop use. Menu systems are configured quickly and easily with 48 programmable macro strings and an autoexecutable string. Up to 20 KB of user data can be stored in the unit. The QTERM-R30 uses less than 100 mA of current and accepts 5 or 7.530 VDC.

The QTERM-R30 sells for $345 in small quantities. Custom OEM labeling of one row of function keys is included free. Full custom keypad graphics are available for an additional price.

**QSI** Corp.
2212 South West Temple, Ste. 50
Salt Lake City, UT 84115 • (801) 466-8770 • Fax: (801) 466-8792
E-mail: infoQqsicorp.com
WWW: **http://www.xmission.com/~qsi/** **#501**

# NEW PRODUCT NEWS

## FULL-COLOR LED



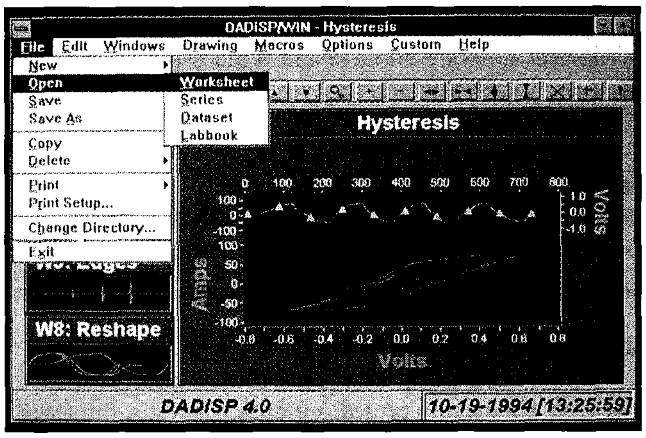
The first LEDs that glow in full color (i.e., any color desired for the application or the condition monitored) have been introduced by Lumex. The **RGB LEDs** are suitable as multistatus indicators and/or a solid-state source of white light. Applications include instrument backlighting, indicators (e.g., temperature, pressure, or velocity), diagnostic and analytical equipment (e.g., color meters, blood and tissue analyzers], and other situations where there is only room for one LED but a range of colors is necessary. For example, engine temperature could be indicated by blue when cold, yellow in normal operating range, and red for an overheated condition.

The T-5 size RGB LED contains four chips (one red, one green, and two blue), encapsulated in a single epoxy package. The chips are individually addressable via six leads: two center commons, and one for each of the four chips. Emitted color is changed by varying power to each chip. Two viewing types are available: a diffused wide-angle lens and a narrow-angle, high-intensity clear lens. Power requirement is 5-30 mA with a forward voltage of 1.7-3.2 V.

Lumex **Opto/Components,** Inc.
292 E. **Hellen** Rd. • Palatine, IL 60067
(706) 359-2790 • Fax: (706) 359-8904

**#502**

## GRAPHICAL DATA ANALYSIS SOFTWARE



DSP Development has announced a new release of its graphical data analysis software. **DADiSP,** an interactive graphics worksheet, enables the collection, analysis, and display of scientific and technical data in an easily understood format. With the DADiSP package, a user can acquire, input, and generate sample data, displaying the results in multiple windows for immediate graphic comparison.

The new version offers native GUI standards, DDE and other interprogram communications, and SPL. Expanded program features include the use of variables, improved printing capabilities, and on-line documentation.

Native MS Windows and OSF Motif GUI standards use common dialog boxes for file section, printer setup, feature customization, and cross-application operations. Dynamic Data Exchange (DDE) is an interprogram communications protocol that allows disparate applications to be combined into a single, custom system. This MS Windows version is offered through command lines and pull-down menus.

Series Programming Language (SPL) is a complete programming language which is modeled after C and provides programming facilities that include user-defined functions, looping and iteration, conditional statements, array references, and variables. In addition to programming constructs, SPL includes a unique capability called ***Hot Variables.*** A Hot Variable links a formula to a variable and may contain integers, complex or real numbers, strings, data series, and matrices.

DADiSP's on-line documentation takes advantage of tools offered by the graphics interface. A manual can also be printed locally. In addition, examples from the manual can be pasted and evaluated directly in an active worksheet.

DADiSP is priced at $1895 for PCs and $2995 for workstations. Software applications and a student version are available.

DSP Development Corp.
One Kendall Sq.
Cambridge, MA 02139
(617) 577-1133
Fax: (617) 577-8211

**#503**

# NEW PRODUCT NEWS

## TRUE REAL-TIME DATA ACQUISITION SYSTEM

PMD Scientific announces DAS-6101, a multichannel, 22-bit, PC based data-acquisition system with two programmable phase-lock loops (PLL). These PLLs maintain accurate universal time and generate precise, real-time, synchronous sampling rates. The first PLL divides the clock frequency by a program-defined 24-bit number to synchronize with any external clock reference (e.g., National Institute of Standards in Boulder, CO). Using this reference, the system can maintain real-time with ±5-ms accuracy. An optional GPS receiver provides ±1-ms accuracy.

The second PLL generates precise clock frequencies for the A/D converter chips. These frequencies are always synchronized to the internal time reference, so that the average error of the sampling rate also is zero.

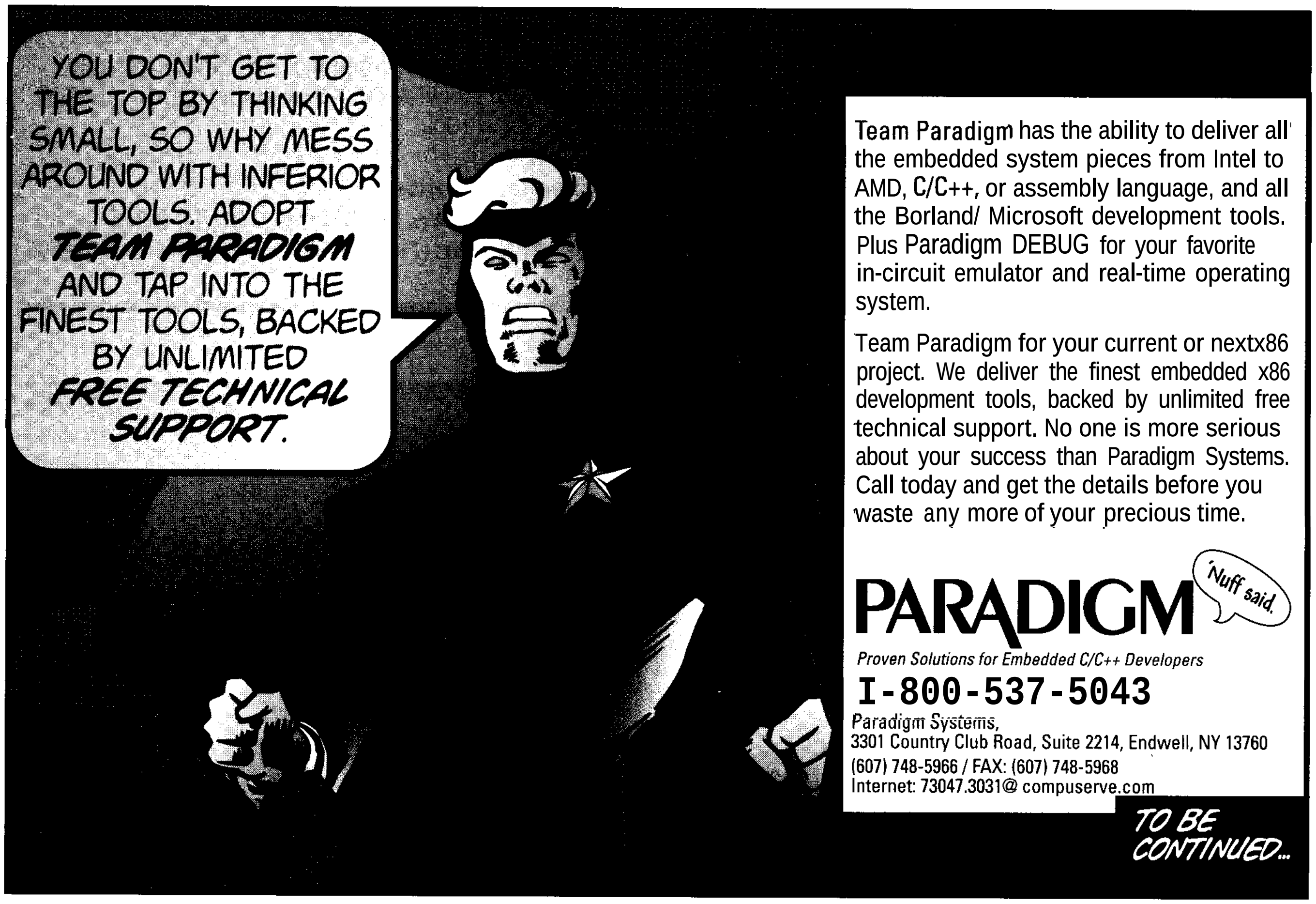
The system is built around the Analog Devices AD7716 d-channel, 22-bit sigma-delta converter chips featuring ±0.003% integral nonlinearity, sampling rates up to 2000 samples per second, and on-chip programmable antialiasing digital filters. Each of the two AT-bus, two-slot-wide boards (master and optional slave) contains up to two plug-in A/D converter boards. The fully configured system has 16 channels. Each differential analog input is equipped with an AD620 instrumentation amplifier. Binary switches offer individual selection of gains of 1 or 16 at each input.

The system generates a data-ready interrupt after each conversion cycle and a tick interrupt for precise time keeping every 100 ms. The slave board works synchronously with the master board. The master board provides four CMOS-level digital outputs and two digital inputs-one of which generates an additional user programmable interrupt.

DAS-6101 sells for from $995 to $2595, depending on the number of channels.

PMD Scientific, Inc.
139A West Dudleytown Rd.
Bloomfield, CT 06002
(860) 769-6088
Fax: (860) 769-6091

#504


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

Team Paradigm has the ability to deliver all the embedded system pieces from Intel to AMD, C/C++, or assembly language, and all the Borland/ Microsoft development tools. Plus Paradigm DEBUG for your favorite in-circuit emulator and real-time operating system.

Team Paradigm for your current or nextx86 project. We deliver the finest embedded x86 development tools, backed by unlimited free technical support. No one is more serious about your success than Paradigm Systems. Call today and get the details before you waste any more of your precious time.

PARADIGM

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

**I-800-537-5043**

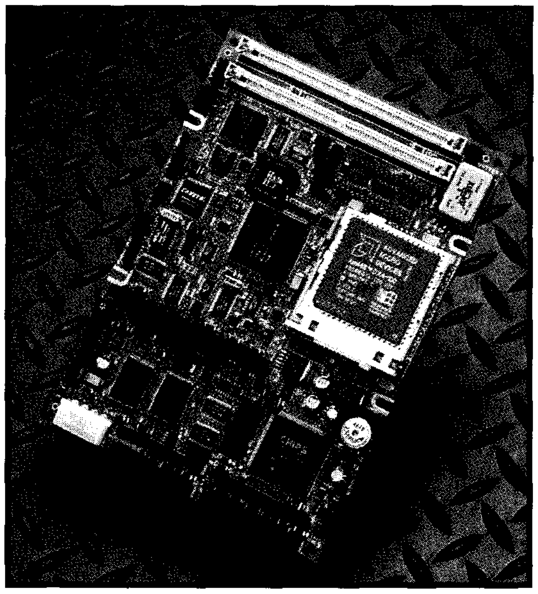
Paradigm Systems,
3301 Country Club Road, Suite 2214, Endwell, NY 13760
(607) 748-5966 / FAX: (607) 748-5968
Internet: 73047.3031@ compuserve.com

#106

# NEW PRODUCT NEWS

## '486-FAMILY SINGLE-BOARD COMPUTER

Innovative Technologies is shipping a family of single-board computers. The it/486 supports the entire range of true '486 family processors, including SX2, DX2, and DX4, with bus speeds up to 40 MHz and internal clocks up to 120 MHz.



An onboard voltage regulator lets you use 5-V-tolerant CPUs running at 3.3-4.0 V, which results in lower power consumption and reduced heat generation. A thermostat circuit under the CPU forces the system into a slower clock mode if the processor begins to overheat, permitting fanless operation in even the tightest of spaces. Write-back internal caches are supported along with the more traditional write-through scheme. Up to 64 MB of DRAM can be installed using two single- or double-banked SIMMs.

The graphics subsystem supports virtually every graphic flat panel available today, along with analog CRTs and simultaneous CRT and flat-panel modes. Based on Chips & Technologies' 65540, the subsystem is Local Bus attached and comes standard with 512 KB of video RAM. An optional interface has also been provided for connection of a PC-video daughtercard for integration of full-motion video.

The onboard disk controllers include a fast IDE interface, which is capable of working with the latest CD-ROM drives as well as traditional hard disks; a floppy controller, which supports two drives of any standard format up to 2.88 MB; and a bootable ROM disk interface, which supports devices up to 1 MB in size.

An onboard Ethernet controller is also available. This subsystem provides a direct 10BaseT twisted-pair interface or a 10Base2 thin-coax interface with addition of an AUI daughtercard.

In addition to the controllers and features already described, the it/486 provides two 16550-type serial ports, an EPP enhanced parallel port, real-time clock, keyboard and PS/2 mouse interfaces, watchdog timer, speaker, and full PC/104-compliant ISA bus. The board measures 8" x 5.75" and runs from a +5-V only source when BIAS voltage generators are not needed. The it/486 supports active power management.

The it/486 is now shipping with small quantities available from stock. A DX2-66 system is priced at $700 in quantities of 100.

Innovative Technologies
10301 Northwest Frwy., Ste. 514 • Houston, TX 77092
(713) 683-0107 • Fax: (713) 683-8478 **#505**

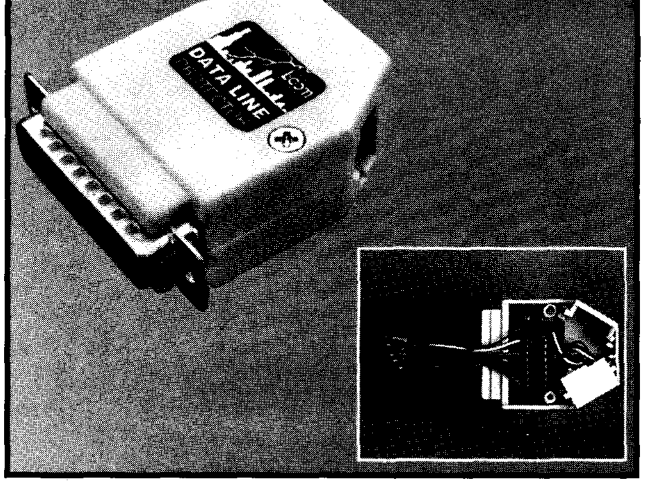
## SURGE PROTECTOR

L-com is offering the **TDHS Modular Adapter Kits with** all RJ45 or RJ12 lines fully protected against the possibility of damage due to line surges and high-speed transients. Each adapter features the traditional 6 or 8 "crimp and poke" for custom final assembly.

A new compact design offers six or eight avalanche diodes (rated at 500 W and 27 V clamping] for each of the lines. Four basic types are offered: male or female DB25 (RS-232) with either an RJ12 or RJ45 modular end.

The TDHS series is priced at $27.95 and $29.95 with additional discounts for quantity purchases.

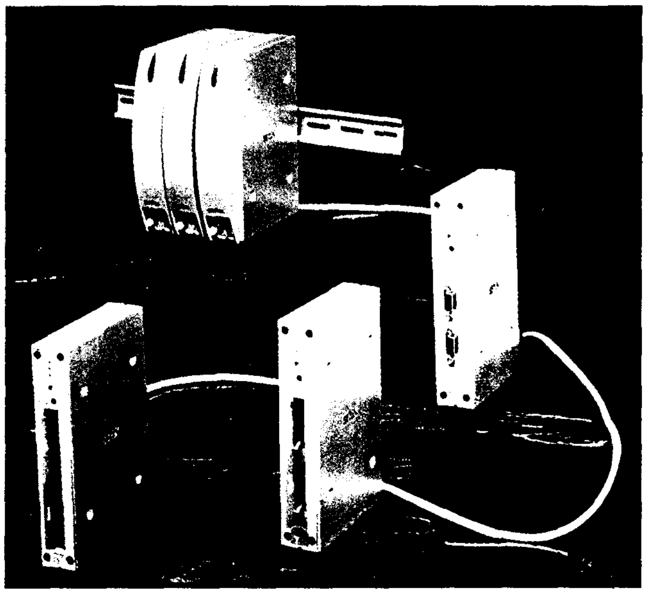
L-com
1755 Osgood St.
North Andover, MA 01845
(508) 682-6936
Fax: (508) 689-9484 **#506**



# NEW PRODUCT NEWS

## PROGRAMMABLE CONTROLLER

Semix has developed a programmable controller that is a distributed and/or a centralized network of motion control and general inputs and outputs. The **NetMotion** 300 consists of a master module and five types of slave modules for serial communication, servo motor control, stepper motor control, and digital and analog inputs and outputs.



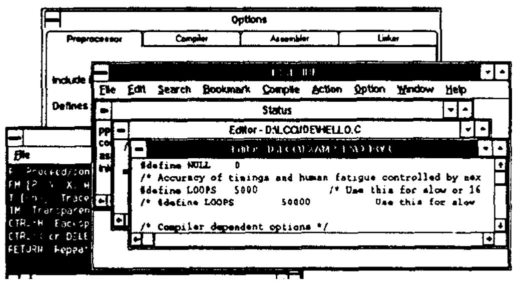
A complex network of control can be devised with the controller. Each master module can be wired simply with up to 4 slave modules connected using a parallel bus and up to 16 slave modules connected on a twisted-pair line. The master module coordinates and runs the user's programs. Various slave modules can be added to control the required components of the user's system.

Up to 16 subnetworks (master and slave modules) can be joined into one cluster and up to 16 clusters can be joined into one large network. With these controllers, one small task or a whole factory could be controlled from a PC or standalone without a PC.

Windows-based software allows setting up an entire network. The master controller can be programmed for SECI/II and GEM for semiconductor applications.

Semix, Inc.
4160 Technology Dr. • Fremont, CA 94538
(510) 659-8800 • Fax: (510) 659-8444 **#507**

## 68HC11 ANSI C Development Environment


Options
Preprocessor
Compiler
Assembler
Linker
Include
Defines
File Edit Search Bookmark Compile Action Option Window Help
Status
Editor - D:\LCC\IDE\HELLO.C
File

```
#define NULL     0
/* Accuracy of timings and human fatigue controlled by max
#define LOOPS   5000              /* Use this for slow or 16
/* #define LOOPS        50000            Use this for slow

/* Compiler dependent options */
```

v 3.0
$100
Visa/MC
check
MO
accepted

**ICC11 V3** includes a Windows IDE with integrated terminal emulator, plus the field-proven ANSI C Compiler with peephole optimizer, assembler, linker, libraries, library source and a comprehensive manual. Advanced features include full floating point support, interspersed C and assembly listing and support for many HC 1 l-specific features, Usable for all HC1 1 variants and system configurations. We also produce **REXIS**, a subsumption architecture based multitasking executive for robotic control systems.

**ICC11 V3:** $100. **V3-lite** (command line tools without IDE, for DOS and DOS emulators): $50. **REXIS**: **$50 ($200** for source). $5 S&H U.S./Canada, $10 S&H elsewhere.

IMAGECRAFT

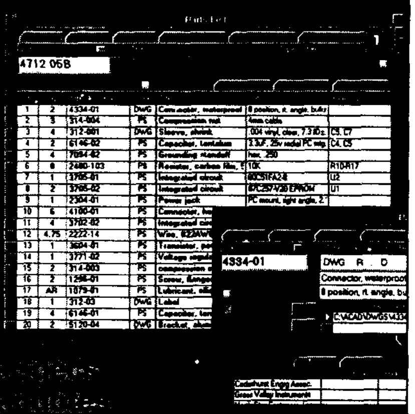
P.O.Box 64226 Sunnyvale, CA 94088-4226
Phone / FAX: (408) 749-0702 imagecft@netcom.com

#107

## This Parts List Software is for Engineers / Designers

*Not MRP: P&V actually assists development!*

Windows™ program helps you stay organized



4712 05B

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 2 | 4334-01 | DWG | Connector, waterproof | 8 position, rt. angle, bulkr | |
| 2 | 3 | 314-004 | PS | Compression nut | 4mm cable | |
| 3 | 4 | 312-001 | DWG | Sleeve, shrink | .004 vinyl, clear, 7.3 IDs | C3, C7 |
| 4 | 2 | 6146-02 | PS | Capacitor, tantalum | 3.3uF, 25v radial PC mtg | C4, C5 |
| 5 | 4 | 7084-02 | PS | Grounding standoff | hex, 250 | |
| 6 | 8 | 2480-103 | PS | Resistor, carbon film, 5 | 10K | R10-R17 |
| 7 | 1 | 3705-01 | PS | Integrated circuit | 80C51FA2-8 | U2 |
| 8 | 2 | 3705-02 | PS | Integrated circuit | 87C257-V20 EPROM | U1 |
| 9 | 1 | 2304-01 | PS | Power jack | PC mount, right angle, 2 | |
| 10 | 6 | 4100-01 | PS | Connector, he | | |
| 11 | 4 | 3702-02 | PS | Integrated cir | | |
| 12 | 4.75 | 2222-14 | PS | Wire, 022AW | | |
| 13 | 1 | 3604-01 | PS | Transistor, po | | |
| 14 | 1 | 3771-02 | PS | Voltage regula | | |
| 15 | 2 | 314-003 | PS | compression n | | |
| 16 | 2 | 1298-01 | PS | Screw, flange | | |
| 17 | AR | 1079-01 | PS | Lubricant, sili | | |
| 18 | 1 | 312-03 | DWG | Label | | |
| 19 | 4 | 6146-01 | PS | Capacitor, tan | | |
| 20 | 2 | 5120-04 | DWG | Bracket, alum | | |

4334-01 DWG R D
Connector, waterproof
8 position, rt. angle, bu
C:\ACAD\DWGS\4334
Cadeshurst Engrg Assoc.
Grass Valley Instruments

Keep track of
-Drawings & [illegible]
-Suppliers & quotes
-Product maintenance
-Engineering stock

**Parts&Vendors**

*...for independent and departmental projects* [illegible]

$99 + shpg **Call 800 280** [illegible]

Trilogy DESIGN
[illegible]
Fax [illegible]
P.O. Box 2270, Grass Valley, CA 95945

Requires 486, 8meg min. RAM, Windows™

FEATURES

# A Robot Firefighter

FEATURE ARTICLE

**Matthew Linder & Kent Harris-Warren**

A firefighting robot must navigate a floor plan, find the fire source, and extinguish it. Sound simple? Well, it's not so very. Out of a field of 30–40, this robot was one of only four to complete the course.

On April 23, 1995, the Second Annual Home Firefighting Robot Contest was held at Trinity College in Hartford, CT. To win, a robot no bigger than 12" x 12" × 12" had to search an 8' x 8' scale model of a house floor plan, find a lit candle, and extinguish it in the shortest amount of time. The candle simulates a house fire that has presumably started while the occupants are away.

Figure 1 depicts the floor plan, which was given ahead of time since a such a robot would know the floor plan of its territory. Bonuses were given for robots which:

- are untethered (i.e., not dependent on an external computer or power supply)
- start on a 3.5-kHz tone (i.e., a smoke detector alert)
- return to their starting spot after putting the candle out
- operate in furniture mode (3.5" diameter cylinders are placed vertically in the floor plan to simulate furniture the robot must navigate around)

## CHASSIS

The chassis for our robot is a 16-sided 12" high cylinder (see photo 1). This shape effectively uses space and reduces abrupt corners, which tend to catch on things. The robot is also able to perform its functions from any angle.

The robot is primarily made of 0.05" aluminum sheets. Although all the other openings were cut with a hand-nibbling tool, a local shop cut the

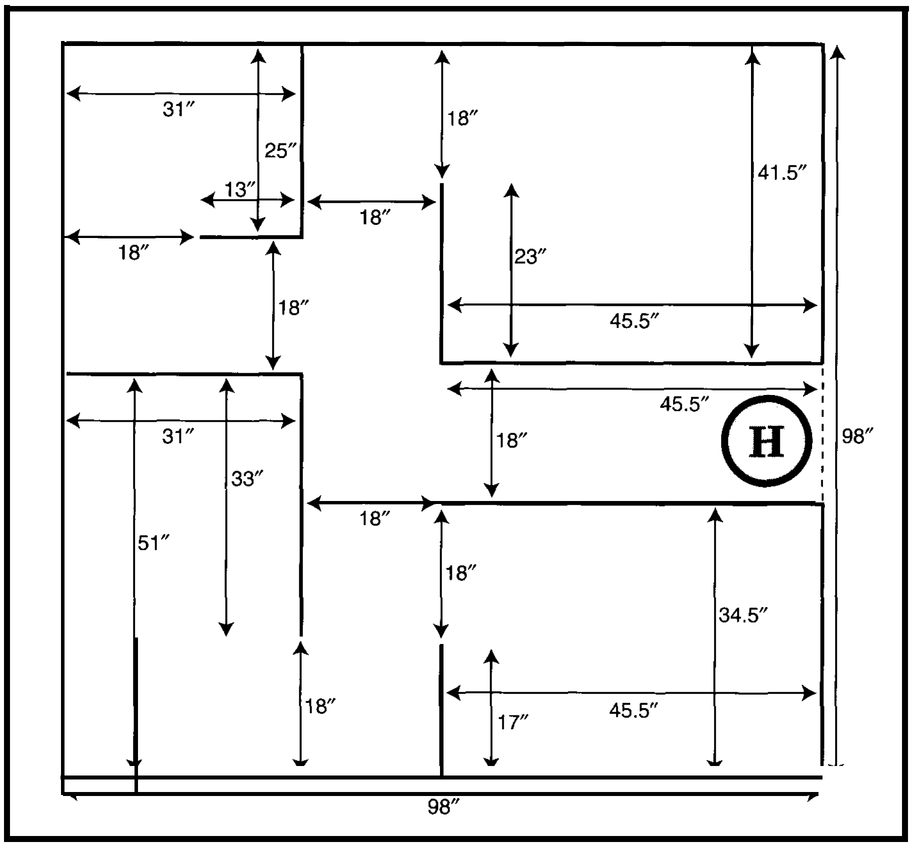


Figure I--The floor *plan represents a typical house with several rooms interconnected by ha//ways. The robot's 12″ diameter is about the size of the home-plate circle. Since dimensions were not exact, the robot navigated via sensor feedback.*

basic shape for the three pieces. Not counting mistakes, it cost about $20.

There are two other kinds of aluminum on the unit. The first is ¾″ x ⅛″ bars. The 16 bars, mounted vertically, hold everything together. Sensors are mounted to these.

The other kind, aluminum angles (¾″ x 1/16″), were harder to make. A V-shaped notch was cut from 16 places on one side of the angle stock, which allowed the piece to be bent into a 16-sided circle. The blank flat sheets were then mounted on top of these circles to hold them together. These sheets separated the robot's top and bottom sections.

The bottom section contained the drive system, batteries, power supplies, and distance sensors. The top section was primarily for the computer, but the flame sensors and extinguisher were also in the top half. This two-level approach enables the robot to be used as a base for other projects by simply removing the flame sensors and extinguisher.

## POWER

Power for the robot comes from two 6.5Ah, sealed, lead-acid batteries connected in series, which gives an unregulated ±12 V, or 24 V across them both. The unregulated voltage feeds the power op-amps driving the motors, distance sensors, and other noncritical devices.

However, since the computer requires regulated voltage, Vicor DC-DC converter modules provide +5 and ±12 V. On the +12-V modules, a simple PI filter reduces the ripple to less than 15 mV p-p. For the main +5-V supply, a Ripple Attenuator Module reduces ripple to less than 5 mV p-p.

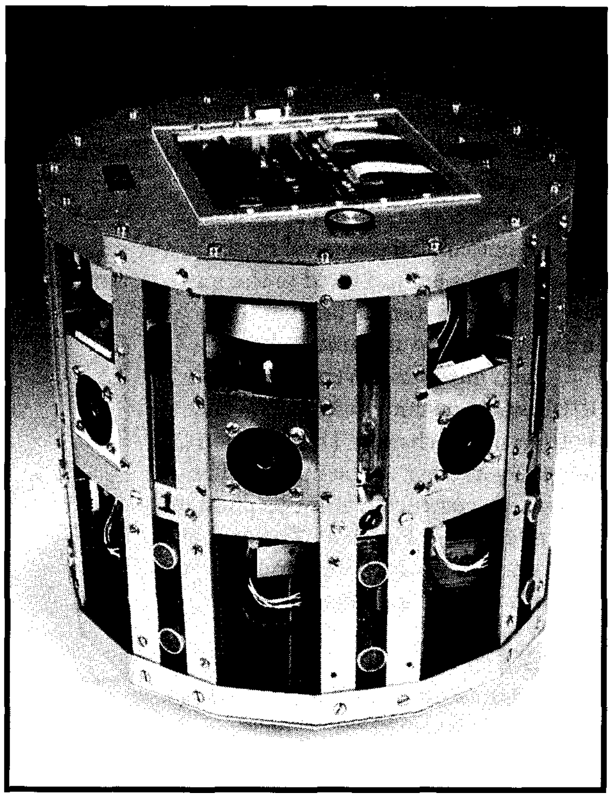


Photo I-With *maximum dimensions of 12″ x 12″ x 12″, a V-sided aluminum frame offers the most real estate. The machine has two levels with fhe bottom ha/f housing the drive mechanism and the fop ha/f containing the processing and sensory systems. This design lets the fop half be refitted to another application.*

All the converters have a wide input-voltage range and are connected across both batteries to balance the load. Battery power is fed to the robot through a set of 4PDT relays. These relays connect to two external plugs—one for a battery charger and the other for an external power supply, which saves the batteries during development.

## DRIVE SYSTEM

The drive system consists of a pair of 5" wheels balanced in front and back by a set of ball casters. The main drive wheels are purposely ⅛″ further down than the casters. Although this positioning causes the robot to rock back and forth slightly starting and stopping, the wheels obtain maximum traction, moving the robot fairly well even on thick carpet. It also prevents the casters from riding up bumps in the floor and lifting the drive wheels off the floor.

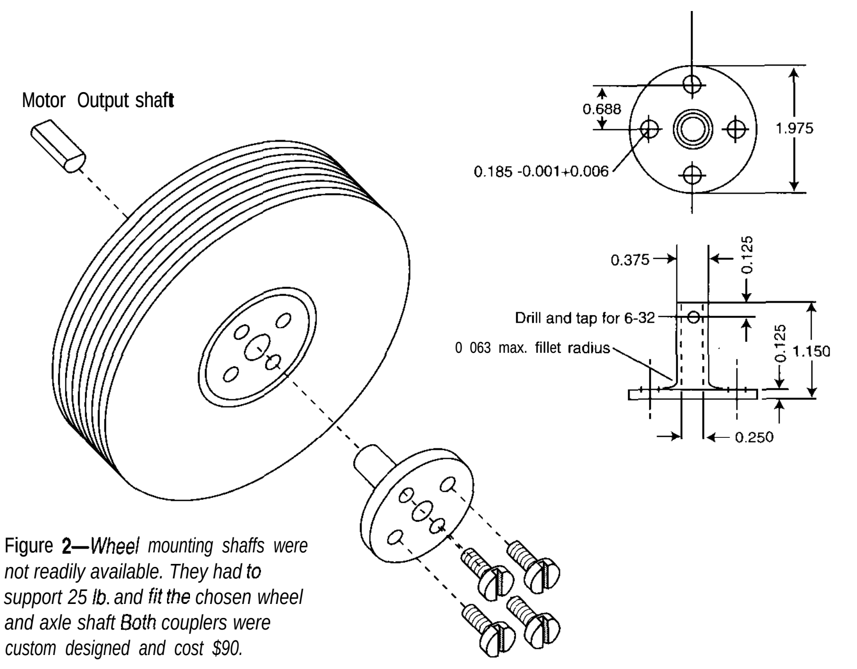


Figure 2—*Wheel mounting shaffs were not readily available. They had to support 25 lb. and fit the chosen wheel and axle shaft Both couplers were custom designed and cost $90.*

The main motors are regular 12-V DC brush motors with a gear head. These motors are *extremely* strong, and I have never been able to stop one. They are rated at 55 lb.-in. peak, with a gear-head ratio of 167.1: 1, and an almost ideal speed of 22 RPM. Both the motor and wheel are available from H&R.

Connecting the wheel to the motor shaft was difficult. Everything we tried didn't work. Figure 2 pictures the coupler we had manufactured.

Each motor has an optical encoder on the motor output shaft with three channels: A, B, and Index. They are the HP-type available from Newark Electronics and consist of an exposed disc and an optical and electrical module. The modules are mounted to the motor case with a simple bracket made from some scrap aluminum.

Each motor is driven by an LH-0101 power op-amp mounted on the bottom of the chassis. This location keeps them close to the motors. The circuit is constructed point-to-point at the op-amp (see Figure 3). The gain of each op-amp is adjustable, and they are set for a ±10-V input signal.

## SENSORS

Polaroid ultrasonic sensors are one of the troublesome areas. The main problem is the minimum range of 1.33'. Although you can get closer using the Blanking Inhibit (BINH) signal, due to the single transducer system, 6" is the best you can do. The sensors also don't allow the frequency or number of pulses they transmit to be changed. They are almost the only thing available.

After a lot of experimentation with the Polaroid ultrasonic developers kit, we built our own ultrasonic modules. Although it took a lot of time and effort, it was worth it. Our modules' minimum was 1" with approximately 0.1" resolution. Each module connects to a central interface card located in the computer.

Being able to detect the flame reliably is probably the most important thing a firefighting robot needs to do. And, not surprisingly, every robot that did well had good sensors.

There are a couple of approaches to detecting flame. Heat sensors are not sensitive enough since most of the heat rises. The simplest is a photocell, which can be fooled by the ambient light. Since the walls of the floor plan are white, the contrast is low.

You can also use infrared. However, there are many stray sources of infrared from video camera autofocusing, remotes, and overhead lighting.

We decided to use ultraviolet. Hamamatsu's R2868 UV-Tron tube ($38) is designed specifically as a flame detector and has a very narrow spectral response. It is also *very* sensitive and can detect a lighter at over 5 m. Although these devices require 325 V, Hamamatsu offers a driving circuit.

The driving circuit outputs pulses that vary in frequency depending on the amount of UV light. Figure 4 offers a simple interface circuit which counts the pulses. Each sensor connects to a central interface card in the computer.

Although in a actual fire the last thing you want to do is fan the flames, this approach is the simplest. Water or dry chemicals are messy, take space, and adds weight. Also, if you run out, you are essentially out of the contest. For a scaled-up version, the fans can be replaced with pumps or compressors

More than one fan establishes a "killing zone." The robot can put out the flame without getting too close or needing to be too accurate.

## COMPUTER

The computer is a 25-MHz 386 PC from Octagon Systems (model 4010).

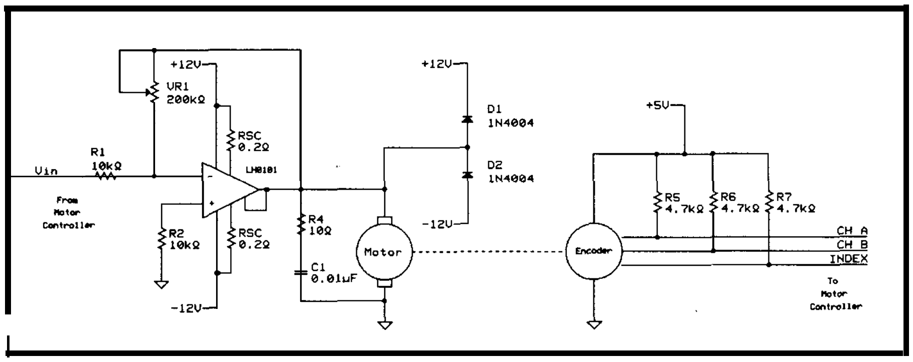


Figure 3—*The motor driver and encoder system make up a typical feedback driver system. Although some motors come with built-in encoders, we mounted ours externally. You can save yourself a /ot of grief by knowing exactly bow and where you're mounting optical encoder wheels before you buy a motor without them.*

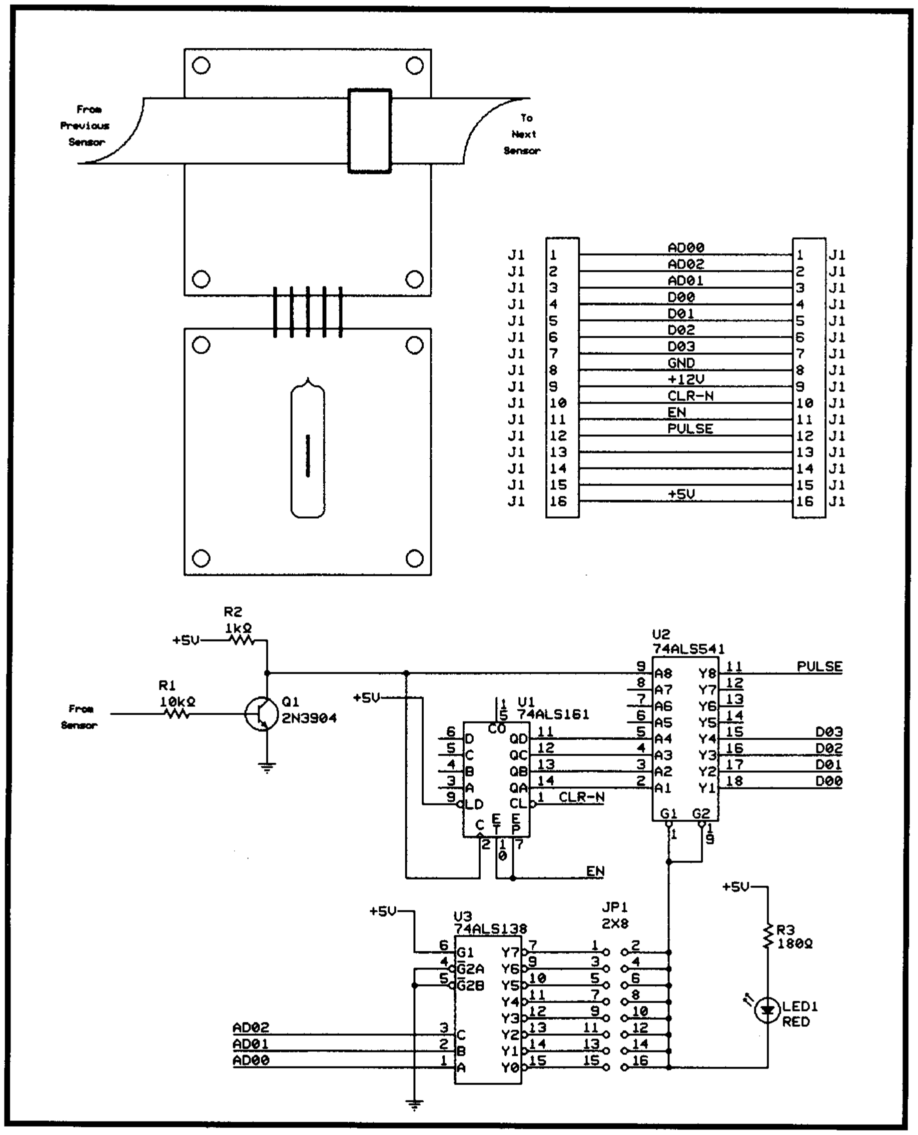


Figure 4—*Eight UV sensors are placed in 45° intervals around the robot. Each sensor has its own decoder interface to allow for digital switching between sensors. Digital switching offers speed and design simplicity while UV sensors offer great sensitivity.*

These computers are relatively small, measuring only 5″ × 5″, and include ISA edge connectors (testing interface boards is simple—just plug them into your PC). The 4010 has a 80386CX processor, two serial ports, DOS on solid-state disk, 1-MB solid-state disk, onboard flash programmer, floppy and hard-disk controller, and a watchdog timer.

The system consists of a passive 4-slot backplane. The computer uses the first slot and comes complete with ROM-DOS. PC-compatible software can be downloaded into RAM and directly executed. The embedded PC feature of the machine was designed to keep software development inexpensive and reliable.

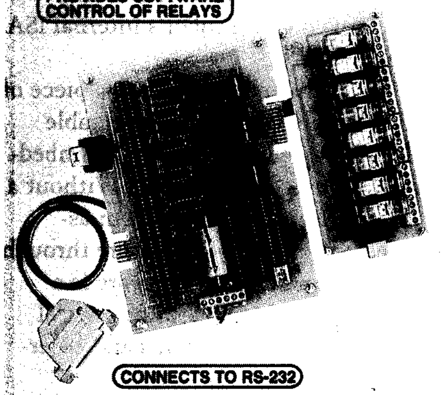
Other reliable, off-the-shelf hardware can be used with the microcontroller's ISA interface. Hardware such as Octagon's 5328 motion controller and 5971 PC prototyping interface let us wire wrap our own custom hardware for the UV and sonar interface controllers. Using a standard bus system minimizes external wiring and increases reliability.

The embedded computer also includes a PC/104 expansion bus. With all the PC/104 hardware options cropping up, it offers more potential.

All software was developed on a desktop PC. We tested the actual interface cards by plugging them into the PC expansion slots—another reason why we chose the ISA approach over

**RELAY INTERFACE**

**PROVIDES SOFTWARE CONTROL OF RELAYS**

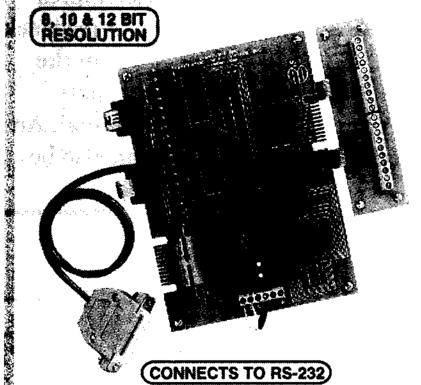


**CONNECTS TO RS-232**

**AR-16 RELAY INTERFACE (16 channel)............$ 89.95**
Two 8 channel (TTL level) outputs are provided for connection to relay cards or other devices (expandable to 128 relays using EX-16 expansion cards). A variety of relays cards and relays are stocked. Call for more info.
**AR-2 RELAY INTERFACE (2 relays, 10 amp)....$ 44.95**
**RD-8 REED RELAY CARD (8 relays, 10 VA).......$ 49.95**
**RH-8 RELAY CARD (10 amp SPDT, 277 VAC)...$ 69.95**

**ANALOG TO DIGITAL**

**8, 10 & 12 BIT RESOLUTION**



**CONNECTS TO RS-232**

**ADC-16 A/D CONVERTER* (16 channel/8 bit)..$ 99.95**
**ADC-8G A/D CONVERTER* (8 channel/10 bit).$124.90**
Input voltage, amperage, pressure, energy usage, joysticks and a wide variety of other types of analog signals. RS-422/RS-485 available (lengths to 4,000'). Call for info on other A/D configurations and 12 bit converters (terminal block and cable sold separately).
**ADC-8E TEMPERATURE INTERFACE* (8 ch)..$ 139.95**
Includes term. block & 8 temp. sensors (-40° to 146° F).
**STA-8 DIGITAL INTERFACE* (8 channel).........$ 99.95**
Input on/off status of relays, switches, HVAC equipment, security devices, smoke detectors, and other devices.
**STA-8D TOUCH TONE INTERFACE*................$ 134.90**
Allows callers to select control functions from any phone.
**PS-4 PORT SELECTOR (4 channels RS-422)....$ 79.95**
Converts an RS-232 port into 4 selectable RS-422 ports.
**CO-485 (RS-232 to RS-422/RS-485 converter)......$ 44.95**

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

- **FULL TECHNICAL SUPPORT**...provided over the telephone by our staff. Technical reference & disk including test software & programming examples in Basic, C and assembly are provided with each order.
- **HIGH RELIABILITY**...engineered for continuous 24 hour industrial applications with 10 years of proven performance in the energy management field.
- **CONNECTS TO RS-232, RS-422 or RS-485**...use with IBM and compatibles, Mac and most computers. All standard baud rates and protocols (50 to 19,200 baud).

**Use our 800 number to order FREE INFORMATION PACKET.** Technical information (614) 464-4470.

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

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

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

#108

PC/104. Once interface cards such as the motion controller and sonar controller checked out in the PC, they were moved to the robot's internal ISA expansion bus.

From that point on, once a piece of code was developed, the executable was downloaded to the robot's embedded computer and executed. Without a keyboard attached, our system assumes all communications are through COM1. Downloading software is a simple matter of connecting a serial cable between the systems and entering a few commands.

The only other software needed is a communications program such as Procomm. Once the program is loaded, the program name is entered, and it is up and running.

Working with the 4010 was a little like running a desktop PC on the end of a RS-232 cable. It was impressive to see the same character feedback of the PC test programs occur in the same format and same exact spots from the 4010 to a dumb terminal. An autoexecutable batch file can also be created in the 4010 so any program can automatically boot on a powerup or reset condition.

## UV INTERFACE

The UV interface board fits in the second slot. This board interfaces to the UV sensors, has a decoder for the 3.5-kHz tone, and controls the fans.

Interface to the UV sensors is controlled by a Motorola M68HC705K. It gives the sample time of the sensors, reads them into memory, and passes them on to the computer.

The tone decoder is just a dynamic microphone amplified by a 741 op-amp then fed to a NE567 tone decoder.

Fans are controlled using eight transistors and a relay.

## SONAR INTERFACE

The ultrasonic controller is in the third slot and connects to eight remote modules. The sonar-remote modules consist of an amplified transmitter can and a preamplified receiver can with a programmable gain stage to reduce false-return signatures at close range.

False returns are received signals that come directly from the transmitter. They are not bounced off a solid object that lies ahead. False returns occur if the receiver amplification is set too high during the transmit cycle.

To counteract this phenomenon, the gain of the receivers amplifier is dynamically stepped each time a sonar signal is transmitted. It is digitally stepped so that the gain is very low during the initial stages of the transmit cycle, and increasingly stepped up as the transmitted sonar signal reaches further into space.

This approach requires all clocks used for frame reference, transmission, and digital gain stepping be in sync.

The sonar controller had several tasks. It is interfaced to the PC bus (all communications are through the system controller), controls eight remote modules, and generates transmit signals while receiving the return signals.

In Figure 5, you can see that everything is derived from a 5-MHz clock oscillator. We chose this speed because it nicely divides down to ultrasonic

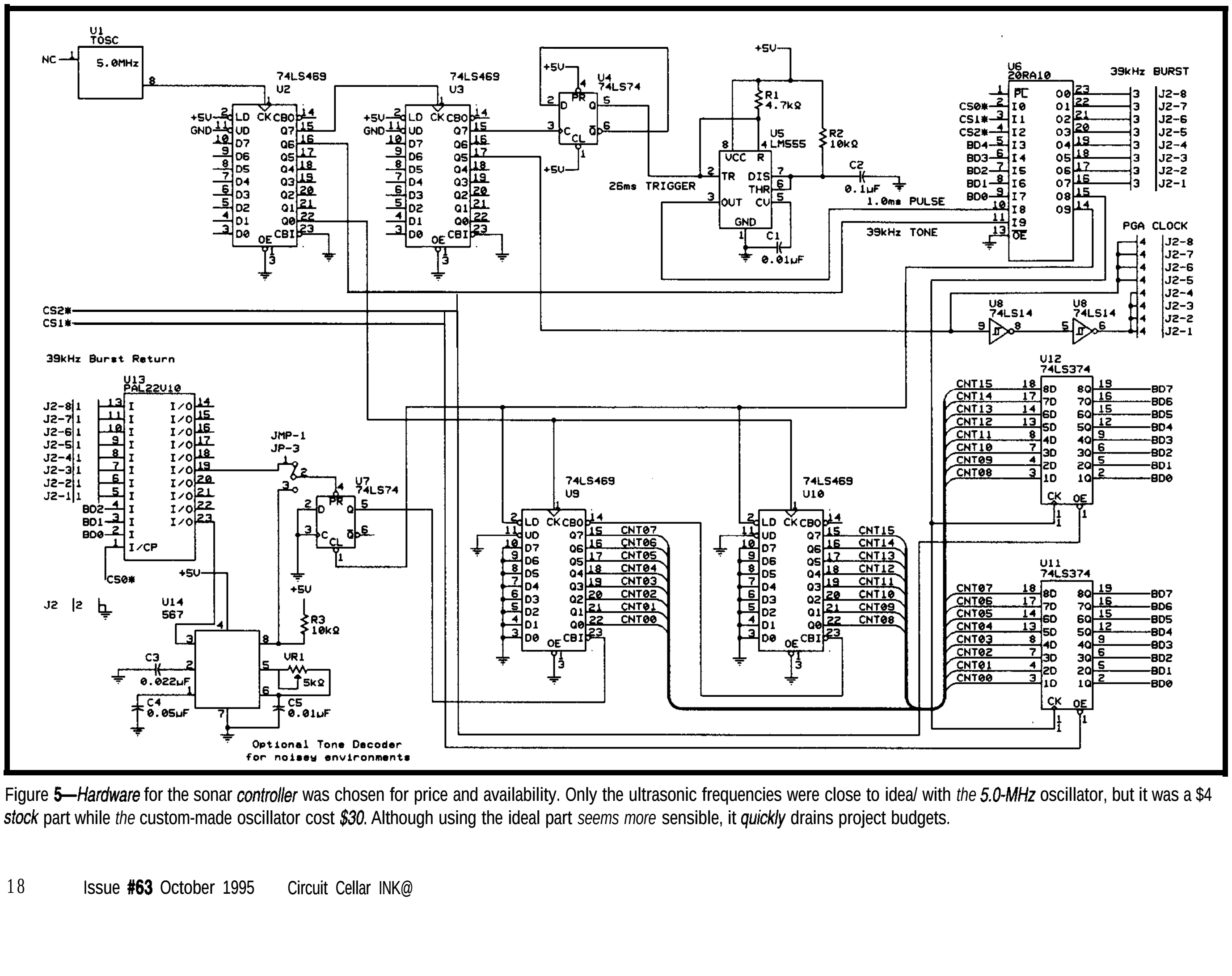


Figure 5—*Hardware* for the sonar *controller* was chosen for price and availability. Only the ultrasonic frequencies were close to *ideal* with *the 5.0-MHz* oscillator, but it was a $4 *stock* part while *the* custom-made oscillator cost *$30.* Although using the ideal part *seems more* sensible, it *quickly* drains project budgets.

frequencies using readily available binary counters.

U2 and U3 are 10-bit counters cascaded to tap all other reference clocks. These clocks include the 39-kHz sonar signal, 2.5MHz echo counter clock (to counters U9 and U10), the programmable-gain-amplifier step clock, and a 26-ms frame sync and trigger clock (to U5). This cascade generates the 1 .O-ms sonar-gating pulse.

The controller functions by sending a pulse of 39-kHz tone to the remote module defined by the system controller. The decoding to determine which remote receives the tone is handled by PAL U6, which is nothing more than a decoder.

The 1 .O-ms pulse from U5 gates the 39-kHz tone for a fraction of the 26-ms frame time. The frame time is the period between pulses sent to a remote sonar module.

Therefore, all sonar signals to and from the sonar controller are referenced by frame time. The beginning of frame time also starts the PGA counter onboard the remote sonar units. This action is necessary because the PGAs are digitally stepped.

Frame time also starts the sonar signal-return counters U9 and U10 by clearing them. These counters start at sonar signal transmission and stop when an echo is received. In other words, they record the sonar signal's time-of-flight so the target's distance can be determined.

Once a return sonar signal is received by the remote, it is sent to U13, another remote channel decoder which functions similarly to U6. The return signal then goes through a tone decoder U14 and a flip-flop U7 to stop the sonar signal return counters.

The final count is latched to octal latches U11 and U12 at the end of frame time. Since the ISA bus was used, the distance information is retrieved as two S-bit bytes.

## MOTION CONTROLLER

The fourth slot is the motor controller board, which we purchased from Octagon (model 5328). It is a two-channel board that controls the position and direction of the two drive motors onboard the machine.

Listing I--To turn *the machine left or right, all we had to do was slighly modify the test functions supplied by Octagon. The 5328 motion controller card comes with a complete set of test functions and example operation procedures. These came in handy when interfacing to our controller software.*

```
void turn_left(int i,  int j) /* i = speed 1 - 10 */
{                             /* j = angle of turn 0 - 360 */
  unsigned long a = 66;       /* motor acceleration rate */
  unsigned long v = i:        /* motor velocity rate     */
  unsigned long p = j;        /* distance in angular measurement */

  v = i * speed-rate;
  p = j  * angular-rate;
  send_traj(1, a, v, p);      /* send trajectory parameters, ch 1 */

  p = -p;                     /* left wheel moves backward */
  send_traj(2, a, v, p);      /* send trajectory parameters, ch 2 */

  start_motionO;              /* start motion */
  wait_traj_complete();
}
```

The motion controller card accepts position signals from a quadrature encoder and contains onboard processing power for a PID filtering scheme. This model uses an analog feedback signal for error correction, although other models provide a PWM feedback signal. All of this model's parameters could be changed on-the-fly, which makes it ideal for robotics applications.

In fact, it turned out to be a very important feature. Once the robot is off and running, if sonar suddenly indicates a wall has been passed on the right side, a command is sent on-the-fly to the motion controller to stop at a specified deceleration rate.

The trapezoidal velocity profile generator let the machine accelerate, level off, and decelerate at practical levels for a machine weighing nearly 25 lb. Of course, this is programmable, so you can choose various velocity profiles for certain conditions.

While programming the motion controller commands, we put in a STOP_QUICK( ) function for panic conditions. We tried it out while in a full-speed run. The error correction was so good the machine stopped in a straight line as if it had ABS.

Listing *2-The robot uses eight UV sensors displaced at 45° intervals. The robot decides whether to turn right or left depending on which sensors receive the most UV energy. If the highest energy registration is direct/y behind the robot, however, unlike humans, the machine always turns right.*

```
void rotate_to_flame(int i)

  int beam:

  beam = find_flame_direction();
  if (beam > 180){
    beam = (360  beam):
    turn_left(i, beam):
  }
  if (beam < 0){
    beam = abs(beam);
    turn_left(i, beam):

  if ((beam > 0) && (beam <= 180))
    turn_right(i, beam);
}
```

Listing 3—*Motion is also a matter of sensing the environment to determine when to go, stop, and turn. In this example, the robot knows if has to go from room 1 to room 2. It also knows all the turns, but if doesn't know how far to go.*

```
void room1_room2(void)

  set_sonar_channel(3);
  turn_left(8, 180);
  move_forward(8, 100);        /* move fwd w/o interruption */
  while  (sonar_return(10) < 12){

  stop_smooth();               /* halt motion */
  set_sonar_channel(1);
  turn_right(8, 90);
  move_forward(8, 100);        /* move fwd w/o interruption */
  while (sonar_return(10) > 11){

  stop_smooth();               /* halt motion */
  check_wall(3);
```

Listing 4-F*l* a me_present *fells the robot whether there is a flame in a room without entering if.* Kill_fire *fells the robot to enter the room and search for the flame. To kill a fire, the robot must* rotate_to_flame *(see Listing 2).*

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
i/include "motordrv.h"
#include "sonar.h"
#include "uv.h"
#include "fan.h"
i/include "tone.h"
i/include "control.h"
#include "motion.h"

int fire_out = 0;               /* will be 1 if fire was put out */

main0

  while  (1){                   /* constant loop */
    fire-out = 0;               /* fire not out */

    set_uv_channel(3);          /* initialize all setups */
    setup_motor_drive();
    sound_alarm(3);             /* system is ready */

    while  (tone_return() > 127){     /* start on 3-kHz tone */
    }                           /* wait, no tone detected yet */

    home_hall();   /* go to center hall from starting pad */
    hall_room1();  /* go to room #1 from center of main hall */
    if  (flame_present(3) > 0){
      kill_fire(32, 22);
      fire-out =

    if (fire-out  = 1) {        /* go to room #2 from room #1 */
      room1_room2 );
      if (flame_p esent(3) > 0){
        kill-fire 21, 5);
        fire-out = 1;
      }
    }
```

*(continued)*

# $149 4A C Compiler?

You heard right! A quality K&R C compiler designed for the **8051** microcontroller family, just $149, **including** the Intel compatible assembler and linker. A source level simulator is also available for just $169. A great companion to our fine Single Board Computers, like those below. **CALL NOW!**

## 552SBC

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

Priced at just $229 in 100 piece quantities. Call about our 552SBC C Development Kit, just $499 qty 1.

## Hyperactive '51!

Our popular 8031SBC can now be shipped with Dallas Semi's hyperactive **DS80C320**, an 8051 on steroids. Averaging 3x faster than the standard 51, your project can really scream! **Call or ftp for pricing and brochures today!**

Other versions of the 803 1SBC have processors with on-chip capture registers, EEPROM, IIC. A/D and more. Call or ftp for a list!

**803 1 SBC** as low as **$49**

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

HTE
**HiTech Equipment Corp.**
**9400 Activity Road**
**San Diego, CA 92126**
**[Fax: (619) 530-1458]**
Since 1983
**(619) 566-1892**

VISA MasterCard

Internet e-mail: info@hte.com
Internet ftp: ftp.hte.com

#110

## SOFTWARE

The software was originally written in C++. However, due to the computer's 5 12-KB RAM limitation, we changed to C.

We used a top-down design approach, starting with obvious function names (e.g., turn-1eft) to navigate the robot from sensor input. The code then worked its way down to the drivers controlling the environmental sensors and extinguishing fans.

Five software drivers control the machine's sensors and drive mechanisms. On top of the drivers is CO NT RO L . C, which directs specific functions such as turn_left i ,j, where i is the angular distance and j is the speed (see Listing 1).

Some functions in C 0 NT RO L . C use more than one driver. This flexibility lets the machine contain functions like rotate_to_flame(i)(see Listing 2) that sense the outside environment before executing a move. Eight geometrically displaced ultraviolet sensors locate the most concentrated beam of UV before directing the machine's drive motors towards it.

MOT ION . C, also sitting above the drivers, takes the machine from any location in the floor plan to another, using sonar to stop, turn, and avoid walls. This approach offers tremendous flexibility in the actual measurements of all the rooms, as it did not depend on precise measurements to navigate from one room to the next.

As you can see in Listing 3, the robot knows it must move from room 1 to room 2, but it doesn't know the exact distance, nor does it care. It simply aims in the right direction and uses sonar to determine when to stop or make its next move.

ROBOFIRE.Cisthemainfileand the final layer of code (see Listing 4). This file combines the action between the sensors and drive mechanisms. The program works the robot's sensors and drive mechanisms so it can navigate its environment, search for an open flame, activate fans to extinguish the flame, and detect when it has effectively done so.

Should the environment change, you can modify the code by adding or altering the functions of MOT ION . C or

Listing 4—*continued*

```
    if (fire-out != 1){       /* go from room #2 to room #3 */
      room2_room3();
       if  (flame_present(3) > 0){
         kill_fire(25, 30);
         fire-out = 1;

    if (fire-out != 1){       /* go from room #3 to room #4 */
      room3_room4();
       if  (flame_present(3) > 0){
         kill_fire(36, 16);
         fire-out = 1;

  }                  /* constant loop and return to main */
}                    /* close main */
```

ROBOFI RE. C. Note that these files are relatively small and simple. There's no need to extensively review the software before making changes.

## ON YOUR MARK, GET SET, GO!

On the day before the contest, people could test their robot in the actual floor plan. The gymnasium's overhead lighting triggered many false alarms since it contained a lot of infrared and ultra violet light.

There was a wide range of entries, all with a slightly different twist. Robots using water, powder, or anything else were in one floor plan, and the rest used the other. There were 30–40 contestants, each getting three runs. Since only two runs counted, reliability was not a factor. Slower robots completing all three runs could be beat by a faster one that only completed two.

Of all the entries, only four successfully completed two runs. Our robot was the most consistent, but not the fastest. It was the only one to sense the walls and not just use an encoder for position.

We're working on making our robot lighter and faster. We'll be back next year! ❑

*Matt Linder has been interested in electronics and robotics for as long as he can remember. He is employed as an electronic engineer for Applied Measurement Systems and may be reached at m.linder6@genie.com.*

*Kent Harris-Warren has been designing embedded controller systems for over eight years. Currently, he is an engineer for Applied Measurement Systems of New London, CT. He may be reached at kenthw@aol.com.*

## SOFTWARE

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

## SOURCES

H&R Company
18 Canal St., P.O. Box 122
Bristol, PA 19007-0122
(800) 848-8001
Fax: (215) 788-9577

Newark Electronics
4801 N. Ravenswood Ave.
Chicago, IL 60640-4496
(312) 784-5100
Fax: (3 12) 907-5217

Octagon Systems
6510 W. 91 Ave.
Westminster, CO 80030
(303) 430-1500
Fax: (303) 426-8126

## I R S

401 Very Useful
402 Moderately Useful
403 Not Useful

# Turning Toys into Tools

FEATURE ARTICLE

Chuck **McManis**

Chuck transforms Radio Shack's Wild Cougar RCV into a functional robot, complete with sensors and the ability to navigate. He focuses on getting the robot to respond properly to sensor feedback.

In making a robot, it is sometimes easier to start with what's around. For instance, Radio Shack's remote-control vehicle, Wild Cougar, has several attributes which make it amenable to conversion. Its track drive is easy to steer. Its rechargeable NiCd battery packs can be swapped easily. Its all-plastic construction is easily machined with hand tools.

Within this article, I'll walk you through the process of converting the Wild Cougar into a robot. I'll cover the components I've used, both off-the-shelf and those I've made, and their integration into the system. I'll finish up with some experiments that can be run using the completed system.

## THE PLATFORM

Wild Cougar's external molding is in the shape of a pickup truck. Figure 1 captures the vehicle shape with the molding removed.

This vehicle is particularly amenable to conversion because its dual-track design enables the base to be steered by driving the tracks at different speeds or in different directions for extremely tight turning.

As you can see, the rear wheel is the drive wheel and has a diameter of 2.75" while the idler wheel is 1.6875" in diameter. This particular track design is not very efficient, though, since the full track drags across the ground during a turn. Modern track vehicles often use a more beveled design to minimize surface area while turning.

This deficiency is overcome by a pair of powerful Mabuchi motors, capable of driving the platform at over 20 MPH! Stalled, they can be a problem as they then draw 5 A. To prevent

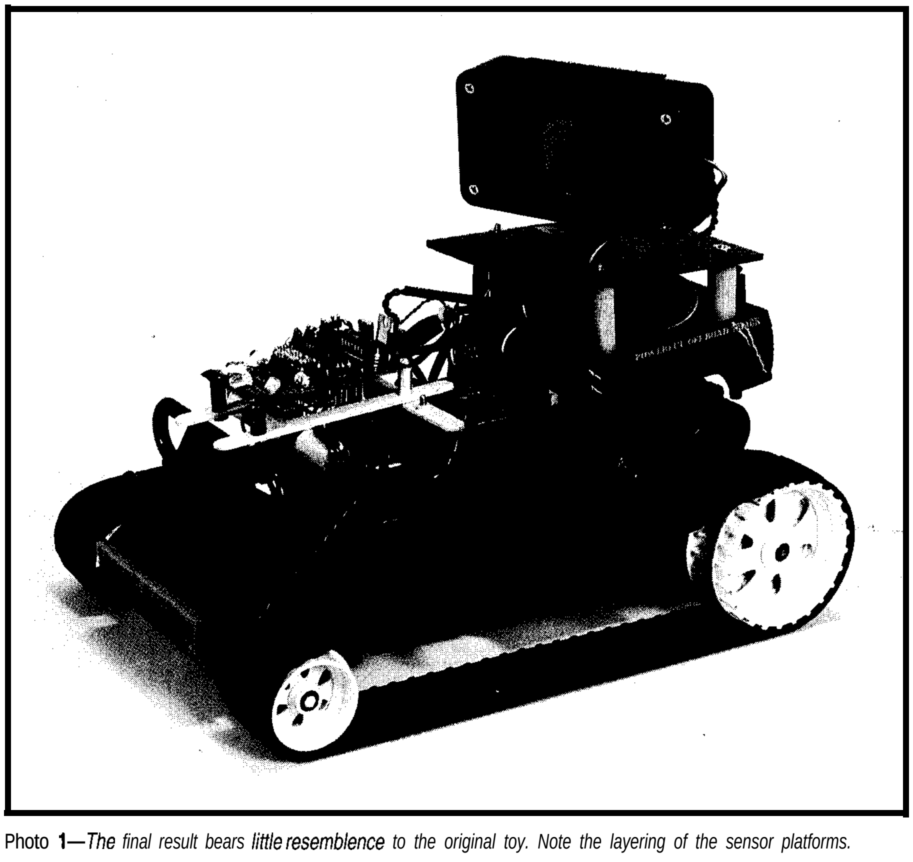


Photo 1—*The final result bears little resemblence to the original toy. Note the layering of the sensor platforms.*

this, I built a custom H-bridge (described later).

Since the wheel rims overhang the axles, I am able to mount magnets on the wheels and Hall-effect switches on the axles. This gives me active feedback on the motion of the vehicle.

I added sensors to the stripped-down chassis to detect obstacles and targets. I retained the rear portion of the plastic for mounting sensor packages.

Finally, I added a CPU to control the robot.

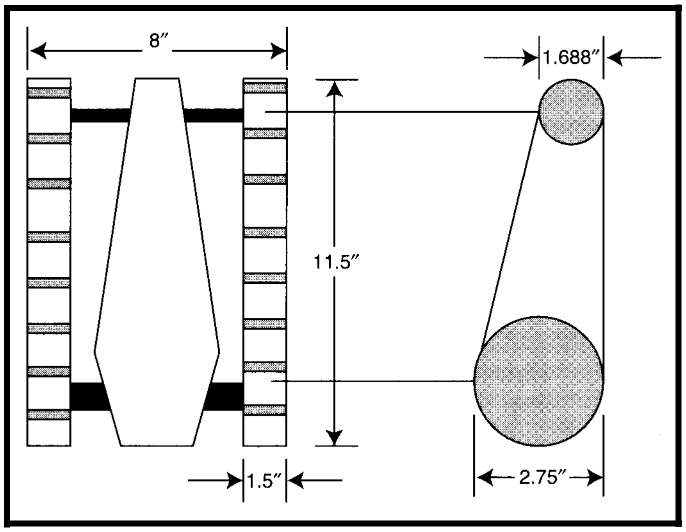


Figure 1—*The dual-tracked layout makes the kinematics and conversion easier. The overall size of the base supports several additional electronics packages.*

## CENTRAL INTELLIGENCE

The base-level intelligence behind the motion and sensors of the Cougar is the Miniboard 2.0. This single-board computer, designed by Fred Martin and Randy Sargent at the MIT media lab, uses the E2 version of Motorola's very capable 68HC 11 microcontroller.

This version of the 68HC 11 series has two general-purpose 8-bit digital ports. One port is dedicated to four full H-bridge high-current drivers implemented by two SGS-Thomson L293D chips. The other is used for general-purpose digital I/O.

The chip also has an 8-bit timer interface section, which can be used for timing events, generating the PWM necessary for operating R/C servos, or as general-purpose I/O.

Two additional ports offer 8 channels of single-ended analog inputs and 8 bits which form the basis of two serial communication interfaces (one asynchronous with an RS-232C interface and the other synchronous). As with the timer channels, these bits can be used as general-purpose I/O. I use the 4 bits of the synchronous interface to talk to an ultrasonic transducer system.

The Miniboard uses the on-chip 2-KB EEPROM of the 'E2 to store programs. These programs are compiled on a host computer and then loaded into the EEPROM using a downloading program. This nonvolatile memory is augmented by 256 bytes of RAM. While small, this amount of memory is sufficient for a lot of useful routines. Other versions of the chip, especially the MC68HC711E9, offer more EPROM for particularly demanding applications.

The development environment consists of cross-assemblers and C cross-compilers. Motorola and Coactive Aesthetics have free C compilers available for this chip. Neither of these options is useful for people who aren't comfortable diving in and debugging problems with the compiler since stressing them tends to break things. A variety of commercial compilers are available from the complete Archimedes system to the budget-conscious Micro-C from Dunfield Development. ImageCraft plans to release a compiler soon. A prerelease version is available free from ftp://ftp.netcom.com/pub/im/imagecft/icc1 1/free/.

## I/O UTILIZATION

Nearly all of the I/O on the Miniboard is used by the Cougar's motion and sensor packages.

The 8-bit I/O port is divided into two portions. Bits 4-7 drive the custom H-bridge. The timer port, port A, is multipurpose. Bits 1 and 2 are used to field interrupts from the Hall-effect sensors on the front wheels. Bit 0 is used as a software serial port that receives input from the IR sensor array and bit 5 controls the servo in the ultrasonics system.

A/D converter channel 0 monitors the current battery state. Channels 1,

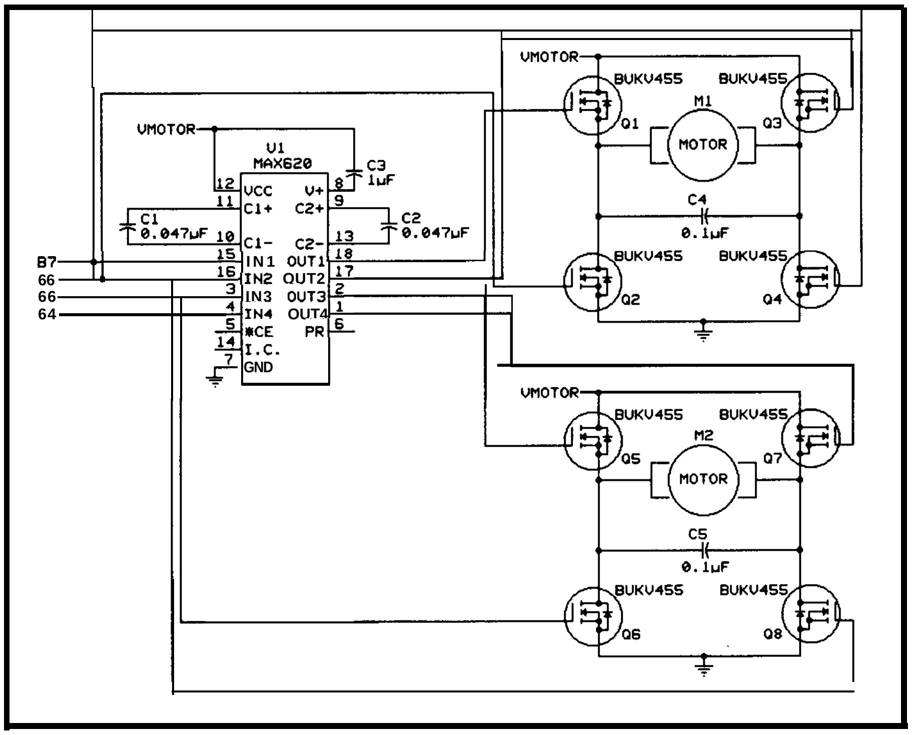
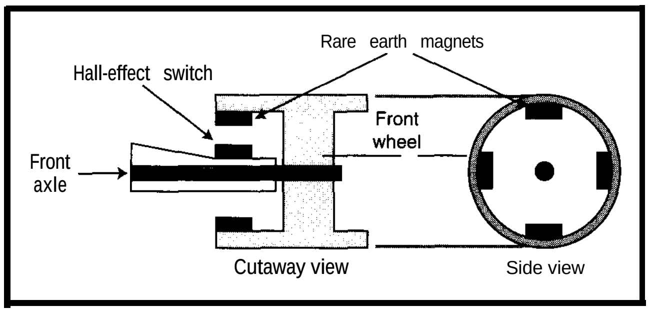


Figure 2—*A MOSFET-based H-bridge is essential for driving the motors on higher end WC-tracked platforms. This design is easily capable of handling the 10 A of current the motors draw on startup and stall.*

Figure 3—*Adapting* the wheel geometry *for speed feedback requires mounting sensors on the axle and wheels.*



2, and 3 are connected to photocells to receive light input while channel 4 is connected to the audio amplifier.

Since the motor drivers are not driving motors, I've used one to switch power to the IR sensor platform. Doing this, I was able to minimize power consumption when the platform was not in motion.

## MOTOR DRIVE

Motor control, especially motor-speed control, is one of the most challenging aspects of this system. At 20 MPH (i.e., 30 feet per second), the Cougar can cross a good-size laboratory in slightly more than 0.5 s! To haul in such speed to a level that the other system parts can manage requires active control of the motor's speed by the processor.

Thus, there are two challenges to overcome. I need to drive motors capable of using 5 A when stalled, and I need to control the speed so that it's possible to implement typical robotic behaviors without constantly smashing into things.

## THE H-BRIDGE MOTOR DRIVER

As Figure 2 illustrates, a custom-designed, dual MOSFET H-bridge controls the motors. A Maxim MAX620 quad MOSFET driver sits at the core of the design. Maxim charge-pump technology generates a drive voltage 10 V higher than $V_{motor}$, which ensures that the MOSFETs on the high side of the bridge are switched fully on.

Input is driven by 4 bits of the general-purpose 8-bit digital I/O port. These 4 bits are configured as active-high inputs with bits 7 and 5 defined to drive forward and bits 6 and 4 to drive backward. While this scheme is easy to drive, there's a danger of turning both bits on at the same time, which causes the MOSFETs to create a short to ground and burn out. To prevent this situation, MOSFETs are traditionally driven with logic gates that turn one output on and the other output low. My second-generation H-bridge design includes this enhancement.

You can replace the MAX620 with the MAX621, thereby eliminating the external capacitors. Although I used Philips BUKV455s which have a current capacity of 38 A ($R_{ds}$ of 0.011 Ω), the MOSFETs can be any logic-level FETs. You could also optically isolate the H-bridge from the controller to prevent the motors' EMF noise crossing into the digital supply of the controller.

Anatomyofa
Great Frame Grabber

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

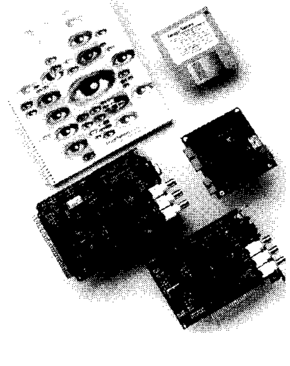
Precision
**Image**
Industry leading sampling jitter less than ±3nS. Video noise less than 1 LSB.

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

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

Call 800-366-9131

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



*ImageNation*™
*"Vision Requires Imagination"*

1-800-366-9131 • Phone: 503-641-7408 • Fax: 503-643-2458
E-mail: info@ImageNation.com
Homepage: http://www.ImageNation.com/~image
P.O. Box 276, Beaverton, OR 97075-0276

#114

## SPEED FEEDBACK

To effectively control the speed of the motors, I needed to know the vehicle speed. If I were designing this system from scratch, I would include tachometers on both motors. Since I was unable to retrofit tachometers, I chose to approach the problem by monitoring wheel turns. (Note: since the wheels come after the gear box, this solution is not ideal because the wheels provide fewer pulses per revolution than a tachometer.)

Finally, because the sensors are not on the drive wheels, they can't detect the drive wheels slipping inside the treads. In practice, slippage has proven not to be a problem.

I configured the Cougar for feedback using Hall-effect switches and rare earth magnets (Radio Shack 64-1895) mounted on the front rims of the wheels (see Figure 3). The switches are bistable, which means they switch on when the north pole of a magnet passes them and off with a south pole. On each revolution of the wheel, four edges are clocked. Because the wheel has a diameter of 1.6875″, these sensors return one clock edge for every 1.33" of forward travel. (You can calculate this by taking the circumference and dividing it by four.)

The outputs from the Hall-effect switches are wired to the timer port (port A) on the Miniboard with the left switch going to capture-input 1, and the right switch, capture-input 2. By wiring them this way, it is possible to write an interrupt routine that counts pulses or to use the timer values to calculate the rotational speed of the wheel associated with that input.

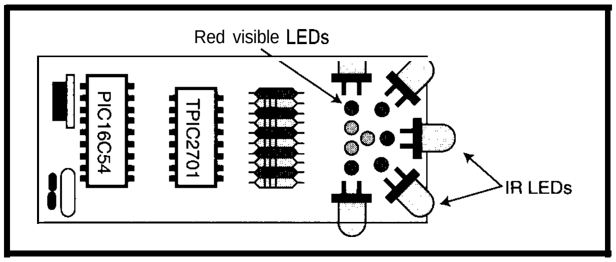


Figure 4—*Infrared* sensors *provide* a medium-range obstacle-detection *facility.* The mounting boardprovides *the electrical* connection *and holds the IR* emitter *LEDs* in position.

Listing 1—*A* flexible *speed-control algorithm is essential to keep the Cougar from running info obstacles.*

```
while (distance != requested-distance) do
  if (emergency_stop) then
    return stop-status
  if (r-encoder > l_encoder) then
    adjust_speed(+1)
  if (l_encoder > r-encoder) then
    adjust_speed(-1)
end

function adjust_speed(amount)
do
  on (amount) do
    case +1:
      speed5 := speed5 << 1
      if (speed5 == 0xE000) then do
        speed5 := (speed5 >> 1) OR 0x8000
        speed6 := (speed6 << 1)
        if (speed6 == 0xF000) then
          stalled := TRUE
      end
      break
    case -1:
      speed6 := speed6 << 1
      if (speed6 == 0xE000) then do
        speed6 := (speed6 >> 1) OR 0x8000
        speed5 := (speed5 << 1)
        if (speed5 == 0xF000) then
          stalled := TRUE
      end
      break
  end
  return
end
```

## MOTOR DRIVER SOFTWARE

The motor driver software is encapsulated in an interrupt-service routine that is run 1000 times a second, or once every millisecond. The software takes the requested motor speed as it is stored in memory, the current speed as indicated by the Hall-effect sensors, and the relative speed of the two tracks. It then integrates the speeds into a signal which is sent to the H-bridge.

The Miniboard drives the motors with an interrupt routine that is part of the system-clock interrupt. The original Miniboard libraries set up the internal 0C4 timer of the 68HC11 to interrupt the processor at a rate of 1 kHz. At each millisecond, the original library implements the PWM algorithm for the motors that had been turned on by calls to the mot. o r () function.

I used a similar technique for the Cougar, except I controlled the speed by writing to a global variable rather than making a function call. This technique saves space in the generated code.

## SPEED CONTROL

The algorithm for controlling the speed of the Cougar uses closed-loop control techniques (see Listing 1). Before discussing the software, let's look at its requirements.

The code must drive the Cougar at a speed less than warp 9 without consuming the entire processor. Next, it is necessary to detect when the Cougar should be moving forward but isn't because one of its motors stalled. Finally, it would be nice to know the distance traveled so that controlled navigational moves can be made.

Getting feedback is straightforward. The Cougar moves along and periodically interrupts the CPU with a tick from the wheel encoders. This tick is captured and stored. The previous value of the input capture register is subtracted from the new value to get the number of ticks that have occurred. Since the clock of the processor is running at 2 MHz, it generates one tick every 500 ns. To get 100 RPM on the Cougar, there needs to be 20,000 ticks in one revolution

Speed feedback is collected by the capture-input 1 and two pins (input capture 0 is used in the IR sensor platform). The interrupt service routine also detects stalls on either motor or from an obstacle using the IR sensors. Either of these events turns off the motors, and their status is reported to the main control program.

When running, the interrupt routines maintain a count of edges (seen from the wheel encoders). This information is stored in global variables r-encoder and l-encoder. The motor-drive function monitors the values in these variables and adjusts the values in speed5 and speed6 to keep the Cougar on track. The variables are initialized to OxFF00, which gives a 50% duty-cycle square wave with a frequency of 60 Hz when shifted out to the motor bits. The integration function is shown in Listing 1.

## IR SENSORS

For short-range (O-24") detection of obstacles, the Cougar uses reflected, modulated IR light. The controller provides modulated IR illumination in five directions and IR receivers to detect IR reflected from obstacles in its path. To screen false positives, the controller also filters out detected reflections. When an obstacle is detected, it signals the Miniboard using a simple serial protocol on the capture-input 0 pin.

The IR source is a set of five IR LEDs driven by the PIC 16C54. This controller connects. to a TI driver chip containing seven MOSFETs. A MOSFET driver chip is required because the LEDs need 60 mA when illuminated, which exceeds the PIC's maximum current-sink capacity.

The controller lights each sensor in turn. Figure 4 depicts the circuit which was prototyped on a micro-Engineering Labs' PICProto18 board.

Listing 2—*The Sharp IR sensors can be converted into ranging obstacle detectors by skewing the input frequency to the emitters. Two zones, 4' and 2', are created using this technique.*

```
function adjust_detectors(desired_state, adjustment)
do
  for num := 1 to 3 do
    if (detector[num] == desired-state)
      detector[num] := detector[num] + adjustment
  end
  return
end

function long_range(led_number)
do
  for iteration := 1 to 192
    LED[led_number] := ON
    delay(14.52 uS)
    LED[led_number] := OFF
    delay(14.52 uS)
    adjust_detectors(TRUE, 1)
  end
  return
end

function short_range(led_number)
do
  for iteration := 1 to 192
    LED[led_number] := ON
    delay(13 uS)
    LED[led_number] := OFF
    delay(13 uS)
    adjust_detectors(TRUE, 1)
  end
  return
end

while (TRUE) do
  current_LED := 0

  for cycles := 1 to 5 do
    if (detected == FALSE)
      long_range(current_LED)
    else
      short_range(current_LED) (* Generate silence for 0.005 s *)
    for iteration := 0 to 192 do
      LED[current_LED] := OFF
      adjust_detectors(FALSE, -1)
    end          (* Check detectors for a collision *
    if (abs(detector_1) > THRESH) then
      detected := detected + 1
    if (abs(detector_2) > THRESH) then
      detected := detected + 2
    if (abs(detector_3) > THRESH) then
      detected := detected + 4
  end            (* 0.1 s for each LED *)

  if (detected != 0) then do
    collision := collision | bit_maps[current_LED, detected]
    detected := 0
  end
                 (* return next LED, handles bounce back at ends *)
  current_LED := next_LED(current_LED)

  if (((current_LED == 1) OR (current_LED == 5)) AND
                 (collision != 0)) then do
    send_update(collision)
    collision := 0
  end
end
```

On the IR board, the red visible LEDs immediately behind the IR LEDs are driven in parallel to the IR LEDs in a pull-down configuration. This setup offers visual inspection of the board operation.

The cluster of three red LEDs in a triangle behind this set reflect the state of the IR detectors as they return to the PIC. Since these are updated in real time, it is possible to see that the Cougar has detected an obstacle by observing the associated detector LED being illuminated. The PIC is clocked by a 2.4576-MHz crystal to give accurate timing of the LED outputs.

## IR SENSOR SOFTWARE

The firmware in the IR sensor board generates a 38-kHz square wave to feed to the LEDs, monitors the detection results from the Sharp IS1U60 IR detectors, and then sends those results to the Miniboard whenever an obstacle is detected. Given that the system is prone to detecting ambient IR, the LEDs are further modulated using a 100-Hz input to distinguish between random and generated IR.

The innermost loop of the program generates a 38-kHz square wave on one of five I/O lines connected to IR LEDs. There are three copies of this loop: one that generates 38 kHz, one that generates 34 kHz, and one that is quiet. This combination lets the 100-Hz AM modulation be generated on the output (see Figure 5).

During both output and quiet stages, the detectors are scanned for the detect state, which alternates between true, when 38 kHz is being produced, and false, when there's no signal. The detectors are repeatedly sampled to eliminate false triggering with glitches. The detection values are then filtered through a threshold.

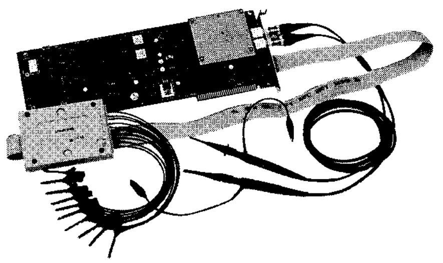
When a reflection is valid, I combine the information about which detectors saw the reflection and which LED was turned on to calculate where the obstacle is located. This calculation returns a bit representing roughly 25.7" of space in front of the Cougar. Listing 2 offers the program in pseudocode.

All detected reflections are shown in the three inner LEDs on the IR scan-

**PC-Based Instruments**

**200 MSa/s DIGITAL OSCILLOSCOPE**

**HUGE BUFFER**
**FAST SAMPLING**
**SCOPE AND LOGIC ANALYZER**
**C LIBRARY W/SOURCE AVAILABLE**
**POWERFUL FRONT PANEL SOFTWARE**



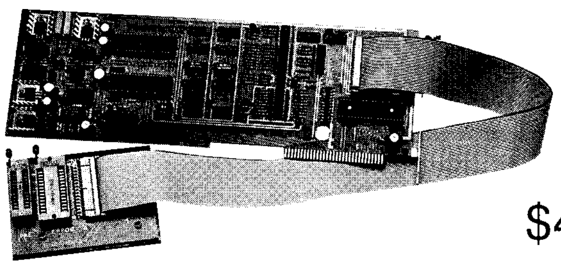
**DSO Channels**
2 Ch. up to 100 MSa/s
or
1 Ch. at 200 MSa/s
4K or 64K Samples/Ch
Cross Trigger with LA
125 MHz Bandwidth

**Logic Analyzer Channels**
8 Ch. up to 100 MHz
4K or 64K Samples/Ch
Cross Trigger with DSO

**$1799 - DSO-28204 (4K)**
**$2285 - DSO-28264 (64K)**

**Universal Device Programmer**

PAL
GAL
EPROM
EEPROM
-LASH
MICRO
PIC
etc..

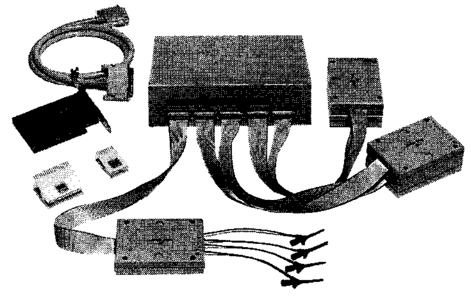


$475

**Free** software updates on BBS
Powerful menu driven software

**400 MHz Logic Analyzer**

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



$799 - LA121 00 (100 **MHz, 24** Ch)
**$1299 - LA32200 (200 MHz, 32 Ch)**
**$1899 - LA32400 (400** MHz, 32 **Ch)**
$2750 - LA64400 (400 MHz, 64 **Ch)**

Call (201) 808-8990

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

#115

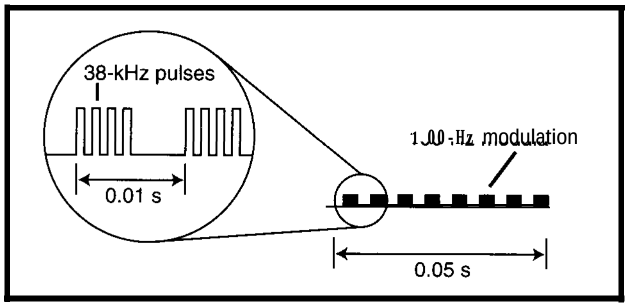


Figure 5—*Modulating the output of the IR emitters enhances their ability to discriminate against background noise (IR light). The Cougar uses a 100-Hz modulation of the 38-kHz carrier.*

ner board. Collision information is sent to the Miniboard in the form of an 8-bit byte with the most significant bit being the short-range bit (see Figure 5).

Bit 0 represents the 25.7" to the furthest left of the Cougar. If the Cougar was heading at 0", this would be 62.5–90°. Bit 1 would be 38.6–64.3° and so on through the half circle until you get to bit 6 which represents 270-295.7". This relationship is shown more clearly in Figure 6.

The scanner board takes advantage of an additional property of the IR detectors-their sensitivity to out-of-band signals. Cougar's IS 1U60s are most sensitive to IR light modulated at 38 kHz. Inside the detector, there is a bandpass filter whose 3-db points are at ±4 kHz, which means the detector is approximately half as sensitive to 34- and 42-kHz modulated IR than it is to 38-kHz IR.

By adding an instruction in each half of the IR sending loop, the frequency can be decreased by 4 kHz. I use this to create two on loops-one designed to produce 38 kHz and one to produce 34 kHz. The IR detectors can detect an obstacle at 48" using the 38-kHz loop, but at only 18–20″ using the 34-kHz loop.

This refinement gives me four effective zones to detect obstacles. In zone A (<24″), Cougar operates only at slow speed. In zone B (24–48″), medium speed is OK. Zone C includes obstacles up to 35' (420″), while zone D holds anything beyond 35'.

## ULTRASONICS

I used a Polaroid TI01 ultrasonic ranging module for long-range sensors. This module detects 1.535' with ±0.5″ accuracy.

The sonar module is mounted on a Hobbico Car/Aircraft servo and has 180" of movement. When centered, the servo points the transducer straight ahead. This setup lets the transducer point 90° left or right or any angle in between with better than 2" accuracy.

The Polaroid module has solder holes for eight lines. Since I was unable to get the module's multiple-echo mode to work, I concentrated on using single-echo mode. In this mode, only four lines to the module are significant: $V_{CC}$, Gnd, INIT, and ECHO. This approach freed up more I/O lines. The pins of the unused SPI port on the 68HC11 can be used as general-purpose I/O when they're not being used in the SPI function.

Because the ultrasonic module is expensive ($40-80 depending on the source), it seemed impractical to have several modules looking in several directions as I did with the infrared sensors. This limitation of looking in only one direction can be overcome in two ways. I could attach the module to the Cougar and turn the Cougar in the direction I wanted to look or I could attach the sensor to a servo platform.

Attaching the module to the Cougar greatly limited functionality. Given the feedback on the tracks, I would not be able to precisely point the Cougar in any direction nor could I adjust my heading by a small number of degrees. Secondly, using the motors to point the transducer consumes a lot of battery reserve. By fixing the position of the transducer, I wouldn't be able to look in a direction the Cougar wasn't pointed. I couldn't use the sonar for target acquisition or tracking progress around obstacles. Clearly, I needed to turn the sensor independent of the robot's direction.

Creating a servo platform turned out to be quite simple. Figure 7 presents the basic design. I cut a mounting platform from plexiglass to the same dimension as the Cougar's truck bed. Dowels, glued into the four corners of the bed, provide support for the four platform corners. Additionally, the dowels are cut so that the platform is level with the ground when it is attached to the Cougar.

In this platform, a rectangular hole about the size of the servo is cut, and the servo is attached via its mounting bolts to the platform. A circular attachment that came with the servo was glued to the bottom of the project case and the whole thing attached to the servo. Wires from the transducer exit the side of the project box and travel through a hole drilled into the mounting plate attached to the Miniboard and the main Cougar chassis.

## ULTRASONIC SENSOR SOFTWARE

There are two components to the ultrasonic sensor software: servo control and distance measurement.

Radio-control model servos easily interface to modern microcontrollers.

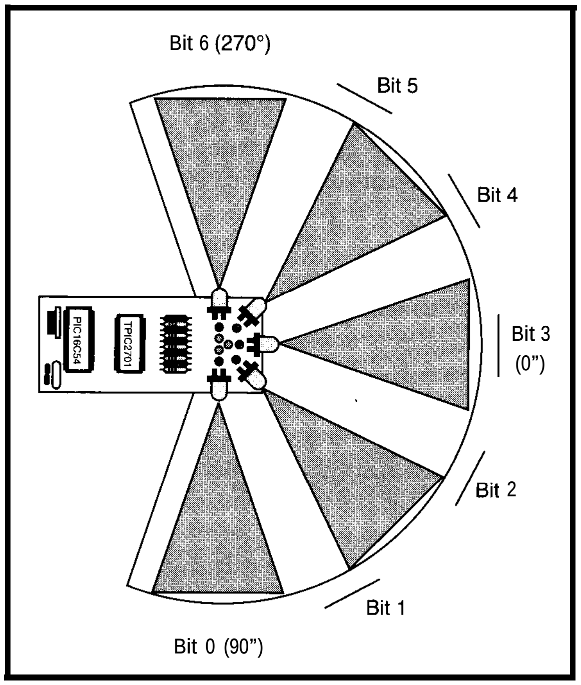


Figure 6-*The Sharp IR defector's field of view is wider than the emitters' output. By combining defection information with that of the emitter, seven areas can be identified.*

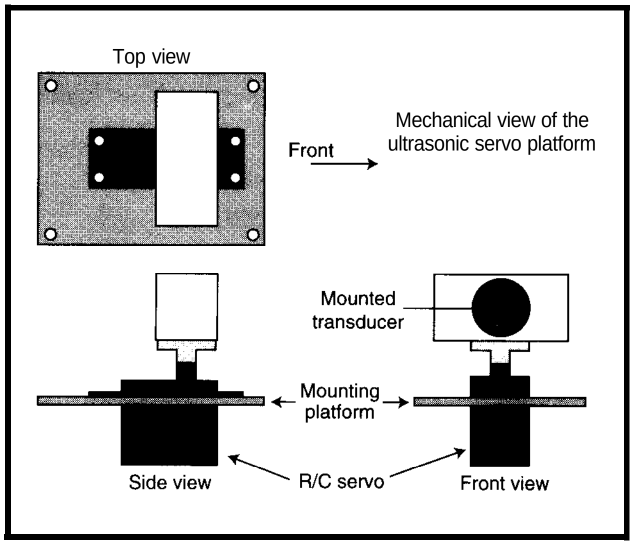


Figure 7—The Ultrasonic sensor platform provides a stable, steerable, and elevated base from which to "look" using the ultrasonic sensor. Nearly 180° of rotation at a distance of 35' is visible.

Servos use a pulse signal that is varied in width and repeated 50 times per second. When the pulse is narrow (about 1 ms wide), the servo is at one extreme of its travel. When the pulse is wide (about 2 ms), the servo is at the other extreme. The servo manufacturer tells you the servo's neutral pulse width. In the case of the Hobbico servos, this value was 1520 µs (Figure 8).

To control the servo, the interrupt service routine runs every 20 ms and sets an output bit on the output-compare port. The routine calculates when the output-compare bit should toggle to 0 and writes that into the output-compare register. The timer section automatically toggles the output when the appropriate time is reached.

The interface to control the servos takes a number between 0 and 100, which represents a percentage of travel. At 50%, the servo is centered with the sonar transducer pointing straight ahead. The code lets you tune these values. While I-2 ms represents the nominal range you can send to the servo, in practice the servo's range is not quite so broad. This difference is caused by mechanical and electronic variations in servo design. Thus, you choose the numbers for each servo where the width is at one end or the other of its travel.

Another area of servo design I ignore is servo slew rate (i.e., the rate for the servo to change to its new position). You have to wait for the servo to settle into a steady position before using the transducer, but you're not attempting to approach that position at anything less than full speed.

In order to accommodate slewing, the interrupt service routine has to compare the desired position with the current position. It then decides whether the next position should be the new position or a fraction of the distance.

It also needs to provide a notification mechanism so that the main code can tell when a requested position has been reached. Slewing reduces servo current consumption, since you aren't driving it as hard, and smoothes the motion, providing a more stable sensor platform.

Unlike the infrared system which is active all the time, the sonar system only activates when the distance to a wall or target is needed. Consequently, it is used by the higher levels of the control program, which are involved with achieving the desired goal.

For this reason, the ranging function ping is quite simple (see Listing 3). It pulls the INIT line of the interface high, and then counts up until the echo line goes high. The accuracy of this technique is affected by the system being interrupted, so the Cougar doesn't move while using sonar.

The count returned by ping can be calibrated into inches by multiplying it by a constant. I've found, however, that simply knowing the relative magnitude of a count is sufficient for navigating decisions.

When combined with infrared sensors, the ultrasonic sensor gives the robot an idea of what is around and what things need to be approached or avoided depending on the application.

Typically, the long-range infrared sensors detect an obstacle, point the sonar in that direction, and take a ping

Listing 3—Turning an echo into a distance measurement requires calculating the time between the emitted pulse and its echo.

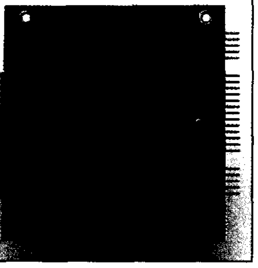
```
function ping
do
  init := TRUE
  count := 0
  while (echo != TRUE) do
    count := count + 1
    if (count > MAXCOUNT) then
      return MAXCOUNT
  end
  return count
end
```

Listing 4—Photocells can defect the presence of ambient light continuously or modulated light periodically.

```
function photo0
do
  if (adc_ch1 < PHOTO_THRESHOLD) then
    result := TO-LEFT
  else if (adc_ch2 < PHOTO_THRESHOLD) then
    result := STRAIGHT-AHEAD
  else if (adc_ch3 < PHOTO_THRESHOLD) then
    result := TO-RIGHT
  return result
end
```

**Sets** the Pace
**in PC/104**
**CPU and DAS**
**Technologies**

33 MHz
486 SLC
**$496**

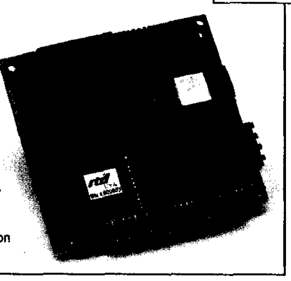


**Features:**
486SLC
33 MHz
387SX
2MB DRAM
SSD
EEPROM
WDT
IDE
FDC
2 Serial
EPP
PS/2 mouse
Keyboard
Quick boot
Virtual devices

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

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

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



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

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

**Make your selection from:**

9 **cpuModules™**

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

7 **utilityModules™**

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

18 **dataModules®**

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

&Real **Time** Devices USA
200 Innovation Boulevard • P.O. Box 906
State College, PA 16804-0906 USA
Tel: 1 (814) 234-8087 • Fax: 1 (814) 234-5218
FaxBack®: 1 (814) 235-1260 -BBS: 1 (814) 234-9427

RTD Europa
Budapest, Hungary
Fax: (36) 1 212.0260

RTD Scandinavia
Helsinki, Finland
Fax: (358) 0 346-4539

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

#117

to determine distance. This is particularly useful when the obstacle is a poor IR reflector and is actually closer than the IR scanner indicates.

## VISIBLE LIGHT SENSORS

In addition to the infrared scanner subsystem, several photocells are connected to the analog-to-digital converters on the Miniboard. This is easily the simplest sensor package on the Cougar. These sensors scan the sides and front to track the blinking "watering-hole" beacon. They can be used in photovore applications as well. Listing 4 presents the basic photo function.

Theconstant **PHOTO-THRESHOLD** is set empirically for the light environment at the time. I experimented with setting it dynamically by lowering it until the beacon was detected. However, this generally takes too much code and the benefit is marginal. In this example, I use A/D converter channels 1-3.

## BATTERY-LEVEL SENSOR

**A/D** converter channel 0 measures the state of the battery. This channel connects directly to the motor power supply through a precision resistor divider (1 kΩ/ 2 kΩ @ 1%). This is a standard feature of the Miniboard, so no additional hardware was needed.

The state of the battery is kept in a global variable `bat_level`. This variable is used in the speed control routines to figure out how much power will be delivered when the motors are turned on full.

## AUDIO SENSORS

The final sensor package on the Cougar is a simple microphone connected to an audio amplifier. The microphone detect whistles, which can be used to send signals to the Cougar.

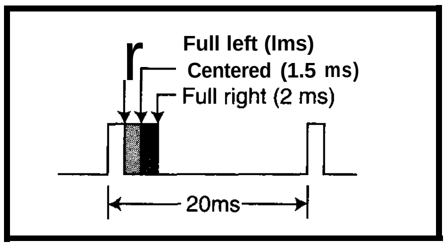


**Figure 8**—*Controlling* **the servo that** *positions the ultrasonic transducer* requires a pulse of variable *width. A 1500-µs* pulse *causes the* transducer to *look straight ahead. Pulses between 1000 µs and 2000 µs set fhe* transducer to one of over 150 positions.

This feature became useful during debugging. I could whistle to tell it to stop, execute a diagnostic, or track (it uses a tone on a target to track when it cannot see a target using IR).

Ideally, the Cougar should have two ears, so it can identify tone and direction. Using a phase-locked loop set, you could identify one tone from two inputs. The output of the left microphone could feed the source signal of the phase-locked loop and the output of the right microphone could feed the input of the second microphone. The phase difference general indicates the direction of the tone.

Other options for the audio subsystem include an FSK modem chip, a DTMF decoder chip, or simply some LM567 tone decoders for single-bit output.

## COUGAR PLATFORM EXPERIMENTS

The complete package is shown in Figure 9. Once you've completed your robot, run it through a few experiments to see if it can do autonomous

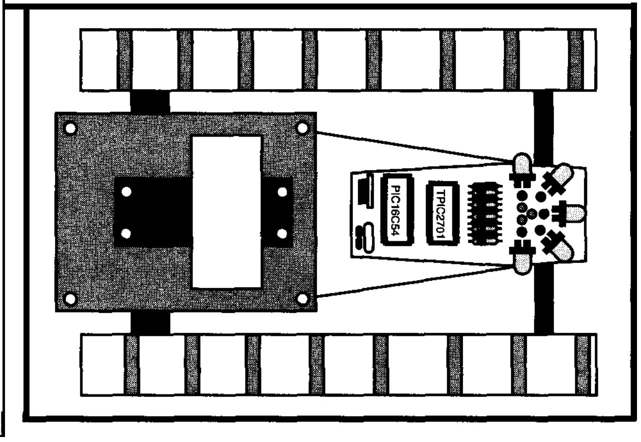


**Figure 9**—*Sensor platforms* are designed to not interfere with *each other. At a minimum, the ultrasonic transducer measures about 1'. This sensor is placed to the rear of the platform to maximize the* useful information *it returns.*

goal navigation (i.e., navigating to a fixed goal while avoiding obstacles) and pursuit-based navigation (i.e., the goal is dynamic rather than static).

First, try to find a beacon. To do this, your robot must wander around, identify and close on the beacon, and recognize when its goal is achieved.

In a maze, the robot must get from point A to point B. Given a lack of precise movement, a maze tests dynamic goal planning. The robot is forced to reassess its plan based on current sensor input.

In the "Track the Dino" test, the Cougar follows a beacon attached to another robot. This experiment differs from the initial experiment in that the beacon is dynamically relocating instead of staying fixed. Note that the Dino also recognizes when it is being followed because it can see the Cougar's IR scanner. When it detects its pursuer, it attempts to evade.

If your robot can master all of these tests, congratulations. You're well on your way to mastering fundamental robot control. ▣

*Chuck McManis is an engineer (BSEE) who has been writing software for the last 20 years. He's currently responsible for networking and security for the Hot Java group at Sun Microsystems in Mountain View, CA. He may be reached at cmcmanis@sun. com.*

## SOFTWARE

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

## SOURCES

**Ultrasonic transducer and module**
Micromint, Inc.
4 Park St.
Vernon, CT 06066
(860) 871-6170
Fax: (860) 872-2204

**Hall-effect switches (HESW-2)**
All Electronics Corp.
P.O. Box 567
Van Nuys, CA 91408
(818) 904-0524

**MAX620, +5-V RS-232 drivers**
Maxim Integrated Products, Inc.
120 San Gabriel Dr.
Sunnyvale, CA 94086
(408) 737-7600

**PICProto 18 and Dual PICProto**
microEngineering Labs
P.O. Box 7532
Colorado Springs, CO 80933
(719) 520-5323

**CS-51 R/C servo**
Hobby Services
1610 Interstate Dr.
Champaign, IL 61821
(217) 398-0007

**68HC11 cross-compiler for PC**
ImageCraft
P.O. Box 94226
Sunnyvale, CA 94086-9991

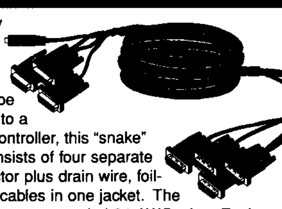
## I R S

404 Very Useful
405 Moderately Useful
406 Not Useful

ALL ELECTRONICS CORP.

**QUALITY PARTS • DISCOUNT PRICES • FAST SERVICE • HUGE SELECTION**

**INCREDIBLE SAVINGS!**
**30' CABLE**



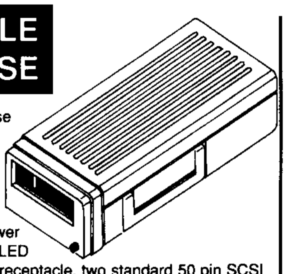
Originally made to connect Digital Audio Tape machine to a remote controller, this "snake" cable consists of four separate 9 conductor plus drain wire, foil-shielded cables in one jacket. The conductors are stranded 24 AWG wire. Each end of each smaller cable is terminated with DB-9P connectors. The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

**$5.50 each**

**CAT# CBL-3**

**2 for $10.00**

**PORTABLE SCSI CASE**



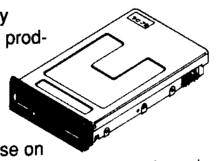
Nearly complete case for SCSI device. Includes 40 watt power supply (5 Vdc @ 3 amps, 12 Vdc @ 2 amps) with cooling fan, power switch, fuse holder, LED socket, power input receptacle, two standard 50 pin SCSI interface jacks, molex-type power connectors, interior cable with 50 pin socket connector and interior mounting rails. Front face plate is an aftermarket piece that doesn't quite go with the box but serves the purpose pretty well. Absent from the case is the switch in back which specifies the device's position in the chain. This can be easily bypassed with a few 0.1" shorting jumpers (our CAT# SJ-1). Also not included is a standard IEC power cord (our CAT# LCAC-60).
Off-white case is 14.25" X 7.38" X 3.26" and has a folding handle for easy carrying.

**$19.95 each**

**CAT# SCSI-1**

**2 for $36.00**

**720K 3.5" INTERNAL FLOPPY DRIVE**



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

**$14.00 each**

**CAT# MPF-11SF**

**10 for $120.00**

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

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

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

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

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

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

#118

FEATURE ARTICLE

Ed Lansinger

# Developing an Engine Control System

## Part 2: Ignition Subsystems

Getting chips to work in the electrically noisy environment of a race car is a critical problem in a microprocessor-controlled ignition system. Ed shows us how to bring 2–8-V spikes under control.

ancient Greeks thought lightning bolts were spears thrown by Zeus. Unlucky mortals, like us, were frequently his target. Now that electronic technology has improved, we can throw a little lightning of our own. This article is the second in a series describing a microprocessor-controlled automotive ignition system.

I developed the system as an integral part of the Engine Control Module (ECM) for the Formula SAE race car at Rensselaer Polytechnic Institute. In the process, I learned how to keep my fragile electronic circuits out of harm's way while creating my own little firestorms.

### RF NOISE AND ITS SUPPRESSION

In a gasoline engine, a mixture of air and fuel is drawn into a cylinder and compressed (see ***INK 62*** for a description of the fuel injection and induction processes and for an overall system block diagram). After compression, this mixture is ignited by a spark. The spark must deliver enough heat energy to light the fuel-air mixture, and must be timed accurately to gain maximum engine power.

The spark itself is caused when current is forced across an air gap. Air is a pretty good insulator, but when a high enough voltage potential exists between the electrodes of a spark plug, it ionizes and conducts electricity. As current flows across the ionized air gap, it heats it. In fact, for a brief instant, the temperature in the gap approaches 60,000 K [1]. If air and fuel are present and in the right proportions, it starts the burning process.

Unfortunately, when the spark occurs, the high rate of current change through the spark plug wires induces significant currents in any nearby conductor. In some cars, you can hear this noise when you tune in an AM radio station and the engine is running. This RF noise can wreak havoc on electronic circuits.

During development of the ECM, I shared a lab with a group that worked with an old research engine. I noticed that I couldn't talk to my ECM when they were using their engine. Every time I tried to run a program, the processor halted in an indeterminate state with its memory corrupted.

A little oscilloscope work uncovered the problem. When monitoring the power bus, I noticed voltage spikes on what should have been a rock-solid +5-V line. Increasing the timing resolution, I found to my horror that the +5-V line was actually leaping as high as +8 V and as low as +2 V at a frequency of about 100 MHz. The spikes occurred more frequently when the research engine ran faster. Further detective work revealed that the ignition system on their engine was causing the problem.

I figured the research engine indicated not only what a competitor's engine could do to my system, but also what my system could do to itself. I had to protect the ECM from the noise or it simply would not run. Since RF noise came in on the wires that ran through the sides of box, putting the ECM in a metal box did not work. The complete solution involved a number of design decisions that reduced both RF production and susceptibility.

For starters, I used resistor-type spark plugs. With these plugs, current goes through a resistor that reduces RF production. Although the arc of the spark itself also produces RF noise, it cannot be heard by the outside world since it is sealed inside the metal walls of the cylinder.

I also chose spark plug wires designed specifically for RF suppression. I used the same resistor core wires commonly used in cars to reduce ignition noise. Some of the high-performance, low resistance wires gave me trouble. Ironically, the standard cheap wires worked just fine.

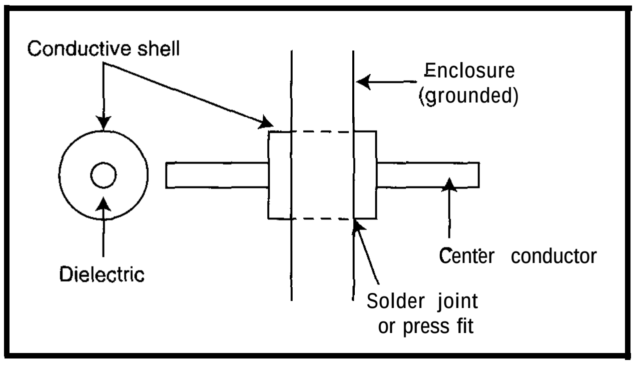


Figure 1—*A* feed-through capacitor *filters* RF noise right at *the wall* of the *enclosure.*

I sealed the CPU board [Motorola 68HC16} and interface electronics inside an aluminum box, which was then grounded to the battery. I kept the high-current drive electronics, which activated the coils and injectors, outside the box on another PCB.

My only remaining problem was getting power and signals through the wall of the box. Surprisingly, putting bypass capacitors on the inner boards did not help because the short (1" or so) length of wire after the hole and before the capacitor was enough of an antenna to rebroadcast the RF inside the box. I needed to ensure the signal was attenuated before it got inside.

I tried fiber optic cables for the digital signals, which worked perfectly. Unfortunately, I also needed to bring in analog signals and power, not to mention that fiber optics was a somewhat costly solution. Finally, I settled on feedthrough bypass capacitors which filtered the noise at the wall of the box.

As shown in Figure 1, a discrete feedthrough capacitor looks something like a resistor with a metal body. The leads of the device are actually a single wire going straight through the center and forming one plate of the capacitor. The outer conductive shell is the second plate and is insulated from the center wire by a dielectric. Insert discrete feedthrough capacitors into holes drilled through a wall of an enclosure (the enclosure must be metal and tied to ground). Any RF noise traveling down the wire sees a low impedance path to ground at the feedthrough and is shorted to the enclosure.

Although discrete feedthroughs are available, I found it more convenient to use filtered DB-25 connectors with feedthroughs built-in, available from Murata Electronics or Spectrum Control. The capacitance depends on the noise frequency you are filtering. I used 2,000 pF with success. (Note that all the lines, including power and logic ground, must pass through the feedthroughs.)

## IGNITION COIL DRIVE ELECTRONICS

After protecting the logic circuits, I got down to the business of firing sparks, which requires the use of an automotive ignition coil.

An ignition coil is a step-up transformer with a turns ratio of about 100: 1. The ignition coil has a resistance in its primary winding in the range of 0.5 to 4 Ω and its inductance is typically 5 mH. The resistance determines the maximum current that can flow through the coil. The combination of resistance and inductance determines how fast current builds.

To fire a spark, you must first charge the coil with current and then stop the current flow quickly. A coil is charged by connecting one side of the primary to +12 V and the other to ground through a transistor that acts as a switch. When the transistor turns on, the current in the coil builds up and creates a magnetic field. When it's time to fire a spark, the transistor is turned off, blocking current flow. The voltage rises on the primary side and, because of the 100:1 turns ratio, induces an extremely high voltage on the secondary side.

The secondary side is connected to the spark plug. When the voltage rises to 10 kV or so, the air in the gap ionizes, allowing current to flow. The magnetic field collapses around the coil and dumps its energy into the spark gap in the form of an electric arc. This arc ignites the fuel-air mixture.

My ignition system features:

- protection of the drive transistor from excessively high voltage
- impedance matched with the coil to reduce time to arc over
- current limiting to avoid overheating the coil

Figure 2 shows the complete drive circuit.

The drive transistor Q1 is a high-breakdown-voltage, high-current Darlington. The high breakdown voltage is

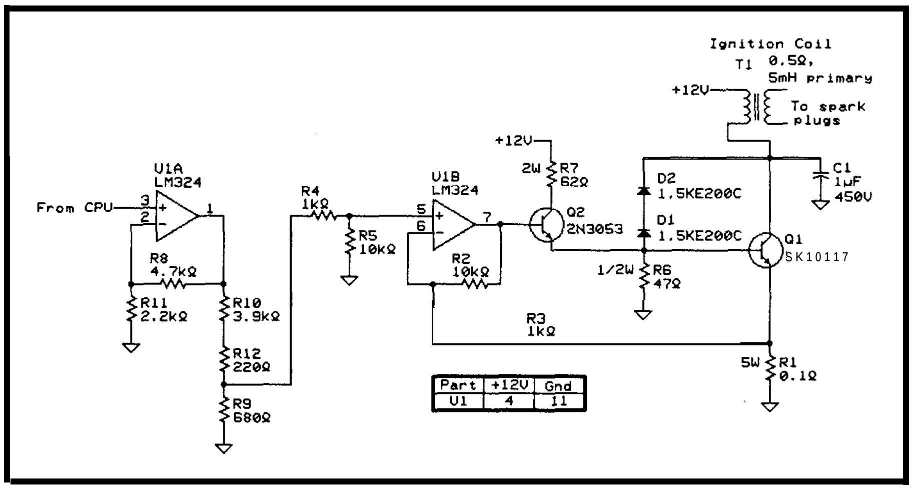


Figure 2—*The ignition* coil drive circuit can *generate* 35-W *sparks. R8–R12* need 1% *tolerance to guarantee the* accuracy *of both the* voltage reference *and the gain.*

necessary because of the 350-V+ primary-side voltages generated by the coil. High current is required because of the need to store a large amount of energy in the coil (stored energy is proportional to the square of the current). The maximum coil current for my system is about 10 A.

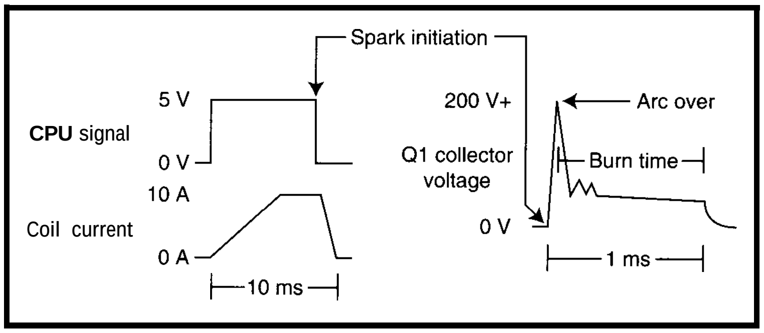


Figure 3—*At* the moment *the* processor signals *the need for a spark, the energy stored in the* coil *discharges* ***very quickly*** *through the spark plug, igniting the air-fuel mixture.*

When the transistor turns off after charging the coil, current does not stop immediately since its flow cannot change instantaneously in an inductor. In other words, that 10 A of current built up in the coil needs to flow somewhere. Ignoring D1, D2, and Cl for the moment, it only has two places to flow: either through the (now off) transistor or across the spark gap, which is connected by induction to the secondary winding.

At the moment of turn-off, the voltage at the Q1 collector rises rapidly. The voltage across the spark gaps also rises rapidly to 100 times the voltage at the collector. Either the spark gap or the transistor is going to lose. We hope the spark gap breaks down first and dissipates the coil's energy. Sparking typically occurs when the secondary side is 10–20 kV and when the corresponding collector of Q1 is 100-200 V.

But, what if you forget to connect the spark plug wires before going out for a drive? Then, instead of a 0.035" air gap across the plug, the current tries to jump the air gap between the two ends of the ignition wires, which could be inches or feet apart. This discharge could require several hundred kilovolts, implying that the Q1 collector would see several thousand volts. With a breakdown voltage of only 450 V, Q1 burns out.

To prevent this from happening, transient-absorbing zener diodes with a total rating of 350 V are connected between the collector and base of Q 1. I used 150- and a 200-V diodes back-to-back to get this clamping voltage.

With these in place, if the voltage at the collector goes above 350 V, the diodes conduct current into the base of Q1 and turn it back on. This feature discharges the coil without subjecting the transistor to destructive breakdown. It would overheat the transistor to run like this for a long time, but normally the problem would be noticed soon enough.

When Q1 turns off, the coil suddenly sees a high-impedance path where it is trying to drive a lot of current. Shortly thereafter, a second lower-impedance path opens across the plug gaps, but not soon enough to avoid a serious ringing condition. The ringing, a result of the impedance mismatch between Q1 and the coil, delays the spark and causes RF noise. Cl was added to eliminate this ringing. Its 1-µF value was determined by trial and error.

## PROTECTING THE COIL

The remainder of the circuit quickly builds up current in the coil and then limits it to a maximum. This action is accomplished by using an op-amp comparator circuit to drive Q 1.

Before flowing to ground, current passes from Q1 through the 0.1-Ω R1. Op-amp Ulb is configured as a differential amplifier. It picks up the voltage at R1 (which is proportional to the current) and compares it to a reference set up by Ula. If the voltage at R1 is lower than the reference, the op-amp drives Q2 to saturation, thus driving Q1 into saturation.

When Q1 is saturated, coil current builds linearly over time at a rate (for this ignition coil) of 1.4 A/ms. As the voltage at R1 nears 1 V, the op-amp output current diminishes, desaturating Q2 and Q 1. The circuit stabilizes when the voltage at R1 is approximately 1 V, which corresponds to a coil current of 10 A. The operation of the entire circuit is summarized by the plots in Figure 3.

The current-limiting circuitry keeps the coil from overheating and the current through Q1 from exceeding that transistor's rating. If the CPU locks up without turning off the coils, the current must be held at a safe level until shutdown occurs. This circuitry also reduces software overhead since the charge time does not need to be monitored as closely.

In the complete system, two coils are used with one drive circuit per coil. Each coil fires two cylinders at the same time. Paired this way, the two cylinders are chosen so that one cylinder is on its exhaust stroke when the other cylinder is on its compression stroke (and vice versa). As such, the extra spark doesn't interfere with anything because it fires in exhaust gas. Spark delivery can therefore be done with a minimum number of coils and without the mechanical distributor of most older engines.

When fabricating the circuit, I needed to ensure that I had sufficiently large traces for all high-current paths. Remember, 20 A total (two ignition coils of 10 A each) needs to flow from the battery through the coils and back to the battery. I saw "ground" on one PCB design rise nearly a volt above battery ground when the coils turned on! Half-inch traces were not enough. It is easier to use 16-ga. wire for these high-current connections.

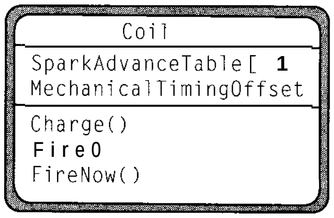


Figure *4-The Co il c/ass* ***controls*** *the ignition coils.*

This circuit has a maximum speed beyond which it will not produce a spark since there simply isn't enough time to charge the coil between suc-

Listing 1—The **software schedules** *the delivery of the spark based on this* **algorithm.**

```
Inputs:
  RPM            current engine speed
  CPSIntTime     TCNT timestamp of last crank position interrupt
  Last90Length length of time taken by last 90 degrees of
                   rotation in TCNT clock ticks

Calibrations:
  SparkAdvance Table []   table recording desired spark advance
                          in degrees at different RPMs
  MechanicalTimingOffset amount to adjust spark timing in
                          degrees due to mechanical error in
                          crank position sensor location

outputs:
  TurnOffAt      turn-off time so spark fires at the right moment

Equations:
  SparkAdvance = SparkAdvanceTable [RPM] + MechanicalTimingOffset
  DelayTime = ((90 degrees - SparkAdvance) / 90 degrees) *
              Last90Length
  TurnOffAt = DelayTime + CPSIntTime
```

cessive sparks. On our engine, 12,000 RPM is about the limit, which fortunately is above the mechanical limit of the engine.

High compression ratios, high manifold pressure, dirty plugs, and wider plug gaps require more energy for reliable ignition. If the air-to-fuel ratio is either too rich or too lean, more energy is also required. The best way to check the ratio is to run the engine and listen for misfire.

## SOFTWARE

The ignition coil drive circuit is controlled through objects derived from the Coil class. Figure 4 shows the data members and messages for the Coil class.

The spark's timing needs to be extremely precise. The fuel-air mixture requires about a millisecond to burn, so the spark must be fired in advance of the piston reaching Top Dead Center (TDC). The advance required for maximum power varies with RPM and other factors and is determined by testing.

I aimed for a timing accuracy of better than ±0.5° at 12,000 RPM, which is ±6.9 ms. The 68HC16 makes precise timing easy because it has built-in timers. The coil is turned on at a particular crank-position interrupt, far enough in advance of the spark that the coil has time to charge. Turning the coil off at the right time involves catching the interrupt 90° before TDC, determining the desired spark advance, calculating the time when the spark should occur, and setting up the built-in timers to do it.

All timing on the 68HC16 is based on TCNT, an internal 16-bit free-running counter. I control the coils using two of the Output Compare (OC) units of the 68HC16. Each OC has a 16-bit register and an associated output pin. The register is loaded with a number that represents a future time. TCNT ticks up and, when it reaches the value programmed in the OC register, the output pin changes state. The output pins directly control the coil drive circuitry.

The software waits until the last possible moment (the interrupt at 90” before TDC) to make its final spark-timing decisions. This reduces the error in predicting engine position when the spark is to be fired and ensures the most up-to-date RPM value.

When the coil is first turned on, its controlling OC is given a default turn-off time that is 65,535 counts in the future. This default ensures that it will turn off at some point even if the software hangs. Listing 1 presents a pseudocode algorithm used by the Coil class to determine how long to

BIG THINGS COME

DOMINO™ Microcontroller

Starting at $79

Micromint's Domino-52 microcontroller is a "supercomputer" in less than 0.75 cubic inches. We've packed the most essential elements into one tiny package. Domino is a plug-and-go module, just attach +5V and a terminal or network. A simple keyed sequence saves an autostarting program in nonvolatile memory.

**SPECIAL FEATURES**
- 80C52 with ROM-resident, full floating-point BASIC
- 32K bytes SRAM and 32K bytes EEPROM
- Two PWM outputs, $I^2C$ bus
- Serial I/O: (up to 19,200 bps) RS-422, RS-485 & RS-232A
- Two interrupts and three timers
- Parallel I/O: 12 bits, 3 shared with ADC and $I^2C$
- Power: +5V @ 15 mA; Size: 1.75"×1.062"×0.4" potted
- A/D converter: 2 channels, 12 bits, 10k samples/sec.
- Connections: via 2×10, 0.1" dual-row header
- -20°C to 75°C operating temperature
- Industrial temperature available

MICROMINT

4 Park Street • Vernon, CT 06066

Call 1-800-635-3355

(860) 871-6170 • Fax (860) 872-2204

#119

wait before firing the spark. This algorithm runs 90" before TDC. The result updates the OC turn-off time.

This scheme breaks down at very low speeds. The engine speed can change dramatically in just 90" of rotation when cranking the engine or while push starting. To get the engine lit, a simplified scheme that always charges 90" before TDC and always fires right at TDC is used. The software switches to the normal mode at about 300 RPM.

The spark-advance curve is developed through testing for maximum torque, which can be measured by acceleration runs or a dynamometer. If the spark advance table for the engine is available, it is a good starting point. If not, you can probably use tables for similar engines.

Too little spark advance can be a problem. If you have a system that fires two cylinders at once, waiting too long after TDC means you could fire a spark during the intake stroke of one of the cylinders, causing a backfire. For any engine, too little advance can increase exhaust gas temperatures to unacceptable levels.

If it causes a knock, too much spark advance can cause serious engine damage. A knock is evident from a characteristic knocking sound and it signals extremely fast combustion of the air-fuel mixture and potentially damaging temperature and pressure. Severe knock can even punch holes through pistons. If you don't know what knock sounds like for your engine, find someone who does before attempting to advance your spark curve!

Another goal for a spark-advance curve is idle stability. At the engine's idle speed-1200 RPM in my case—you can retard the spark a little so it is not producing maximum torque. Then you can tweak the table so that as RPMs fall, spark advance goes up slightly. This produces a little more torque to spin the engine faster and stabilize the idle.

## CONCLUSION

The ignition system proved to be a success, generating enough voltage to jump a 2" air gap, yet remaining unaffected by the tremendous RF noise this caused. Now that the spark advance was in my software and not locked in the OEM black box, I was able to easily recalibrate the engine for more power.

Next month, I'll wrap up by describing the automatic shut-down, crank position, and power-supply circuits. I'll also cover the all-important `Distributor` and `AlarmClock` software objects and the ancillary objects that keep the system running. I'll finish by describing how to test and tune the system as a whole.

So, get ready for a little fire and thunder! ❑

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

## REFERENCES

[1] John Heywood, *Internal Combustion Engine Fundamentals,* McGraw-Hill, New York, 43 1, 1988.

## CONTACTS

**Feedthrough capacitors and DB-type filtered connectors**
Murata Electronics North America, Inc.
2200 Lake Park Dr.
Smyrna, GA 30080
(404) 436-1300
Fax: (404) 436-3030

Spectrum Control, Inc.
6000 Westridge Rd.
Erie, PA 16506
(814) 835-4000

## I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

IN SMALL PACKAGES

BLACKJACK™
TELECONTROLLER

Starting at $139

Micromint's BlackJack-552 is a true "telecontroller" incorporating both microcomputer and FCC-certified Xecom modem in a single package. BlackJack comes preprogrammed with firmware utilities which are optimized for assembly language, C, and BASIC programs. Like Domino, BlackJack is easy to use. Attach power and a terminal, then upload, store, and execute your program.

SPECIAL FEATURES
- 80C552 processor with firmware monitor, 14.7456 MHz
- Serial I/O; (up to 38,400 bps); full-duplex TTL, $I^2C$ bus
- Connection: 1.2" dual-row 48-pin DIP header
- Parallel I/O: 20 bits, 4 bits shared with RTC
- Two interrupts: three timers, watchdog timer, two PWM outputs
- Hardware real-time clock
- A/D converter; 8 channels, 10 bits, 20k samples/sec.
- 2400 bps data modem (data/fax available)
- Power: +5V @ 55 mA; Size: 2.75"×1.375"×0.5" potted
- 32K bytes SRAM, 32K bytes ROM and 128K bytes EEPROM
- Industrial temperature available

MICRO MINT

4 Park Street • Vernon, CT 06066
Call 1-800-635-3355
(860) 871-6170 • Fax (860) 872-2204

#119

# Self-Testing the M68HC11

FEATURE ARTICLE

**Maurizio Ferrari**

The MC68HC11 contains a unique feature that many are not aware of. In bootstrap mode, the designer can load test code into the chip without system EPROM. Maurizio shows us how to use this mode.

One can never say how long a device will last. When Motorola designers started thinking about the successors of the 6801/6803 family, they probably never imagined how successful they'd be. Many years later, Motorola's 68-HC11 is still a favorite in many 8-bit designs, due to its wealth of on-chip peripherals, flexibility, and development tools.

In this article, I dig into some of the special bootstrap mode features. I'll present code which, when downloaded through the serial line to an 'HC1 1 working in this mode, can be used to test RAM, EEPROM, A/D converters, and timers, and store values in EEPROM.

## THE BOOTSTRAP MODE

The 'HC1 1 can be put to work in four basic modes:

- single chip
- expanded multiplexed
- special bootstrap
- special test

As described in Table 1, these modes can be selected by setting appropriate levels on the MODA and MODB pins during reset.

Single chip and expanded multiplexed are normal operating modes. Special bootstrap mode is a variation of single chip and is a user mode. It lets users download programs through the 'HC1 l's SCI (UART) into internal RAM. The programs can then be executed from there. Special test mode is intended mainly for factory testing and is a variation of expanded multiplexed (i.e., with external RAM or ROM) mode.

When the MCU is reset with both MODA and MODB pins at logic 0, it enters bootstrap mode, fetching its reset vector from a small on-chip ROM residing in BF40h–BFFFh. This ROM contains a bootloader routine that takes the following actions:

- initializes the stack
- initializes the SCI (UART)
- sends a break (i.e., pulls down) on the transmit line.

It then waits until a start bit is detected on the receive pin.

Depending on which byte is first received, the MCU can jump immediately to the start of the EEPROM, or it can start downloading 256 bytes of user code that is then written sequentially to RAM from 0000 upwards. Before starting this download procedure, it can also provide security checks (the MCU erases EEPROM, RAM, and the Config register) and switch SCI speed down to 1200 bps from the default of 7812 bps (with an 8-MHz crystal).

A very useful feature is the "Jump to begin of EEPROM" mode. If transmit and receive are pulled up through a resistor when the 'HC 11 is reset, the processor enters this special bootstrap mode variation, forcing an immediate jump to the beginning of the EEPROM. Provided that the EEPROM

| Inputs | | | Control bits in HPRIO (latched at reset) | | | |
|---|---|---|---|---|---|---|
| MODB | MODA | Description | RBOOT | SMOD | MDA | IRV |
| 1 | 0 | Normal single chip | 0 | 0 | 0 | 0 |
| 1 | 1 | Normal expanded | 0 | 0 | 1 | 0 |
| 0 | 0 | Special bootstrap | 1 | 1 | 0 | 1 |
| 0 | 1 | Special test | 0 | 1 | 1 | 1 |

Table l-Assigning *specific values to the MODA and MODB pins at reset forces one of the 'HC1* l's working modes. *Bootstrap* mode *can only be entered this way.*

| Address | Vector Name |
|---|---|
| 00C4–00C6 | SCI |
| 00C7–00C9 | SPI |
| OOCA-OOCC | Pulse Acc. Input Edge |
| 00CD–00CF | Pulse Acc. Overflow |
| 00D0–00D2 | Timer Overflow |
| 00D3–00D5 | Timer Output Compare 5 |
| 00D6–00D8 | Timer Output Compare 4 |
| 00D9–00DB | Timer Output Compare 3 |
| 00DC–00DE | Timer Output Compare 2 |
| 00DF–00E1 | Timer Output Compare 1 |
| 00E2–00E4 | Timer Input Capture 3 |
| 00E5–00E7 | Timer Input Capture 2 |
| 00E8–00EA | Timer Input Capture 1 |
| 00E8–00ED | Real-time Interrupt |
| 00EE–00FD | *IRQ |
| 00F4–00F6 | *XIRQ |
| 00F7–00F9 | SWI |
| 00F1–00F3 | Illegal Opcode |
| OOFA-OOFC | COP Fail |
| 00FD–00FF | Clock Monitor Fail |
| BF40 | Reset (Bootloader Start) |

Table 2-The program counterpoints to *the pseudo-vector's RAM* addresses when an interrupt occurs in *special bootstrap* mode. A *JMP* instruction placed here forces execution *of the real* interrupt *code.*

contains a meaningful program, execution begins.

The 256-byte limit can vary according to the 'HC1 1 type. Those with more than 256 bytes (e.g., the E9) typically stop downloading bootstrap code after a 4-byte time slot (obviously this varies according to selected speed) without any incoming characters on the serial line and jumps to RAM start. User programs can also jump back to the ROM bootloader section, downloading a new program into RAM. This way, several procedures can be executed without resetting the micro every time.

Bootstrap mode provides a way to implement user-specified interrupt routines through RAM mapping (see Table 2). These pseudovectors in RAM are pointed to by interrupt vectors in the bootstrap ROM. Users must fill the RAM's 3-byte pseudovector with a JM P instruction to the user routine.

When an interrupt is triggered in bootstrap mode, it fetches its start address from the ROM vector pointing to the bootstrap RAM address where the actual jump takes place. For example, suppose your SWI routine is at E000h. To use SWI in bootstrap mode, cells 00F7h–00F9h then contain:

```
7e e0 00 JMP  SWI_INTERRUPT
```

SW I fetches OOFA as its program counter and is therefore directed to SW I_ INTERRUPT.

## TESTING VIA BOOTSTRAP

In my experience, microcontrollers are rarely faulty. They generally do the right thing with misbehavior stemming mostly from undocumented software features, otherwise known as bugs.

Nonetheless, a processor must be tested if for nothing else than peace of mind. The 'HC1 l's bootstrap provides an easy and inexpensive way to spot hardware deficiencies should they ever happen. I'll present four types of tests devised for testing RAM, EEPROM, timers, and A/D channels.

Every mode has two ways of being assembled: one being for real bootstrap mode test and the other for debug purposes. With Off setx [where x = the number of the test) label, 0 forces the assembler to generate code at the address 0. Equating Offsetx to c0x (again, this may be c01,c02,. . .) lets you debug your code from an emulator's RAM anywhere in its address space.

The Motorola Macro Assembler lets you write code with a structured high-level-like language syntax. It's not like a full high-level language, but it helps you understand the flow of the test code. A port to a cheaper (freeware], nonstructured assembler is trivial.

I now use my own port of the Motorola freeware stuff to the Acorn Archimedes (a less-familiar ARM RISC-based computer) since I've grown tired of DOS. Taking advantage of the RISC OS environment (the friendliest OS I've ever seen), I've added a windowed environment, an editor throwback (editor windows are opened at the source's offending line), and an ANSI

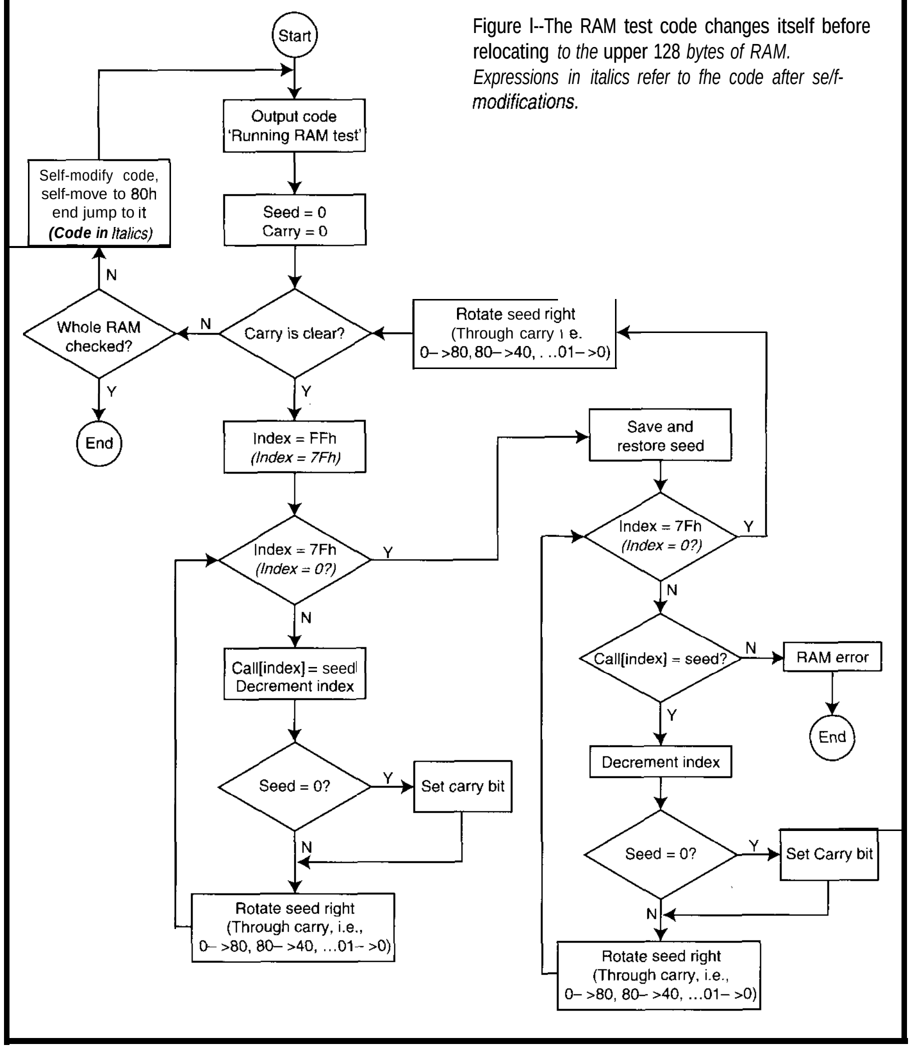

Start
Output code 'Running RAM test'
Seed = 0 Carry = 0
Self-modify code, self-move to 80h end jump to it (Code in Italics)
Whole RAM checked?
N
Y
End
Carry is clear?
Rotate seed right (Through carry i e. 0–>80, 80–>40, ...01–>0)
Index = FFh (Index = 7Fh)
Index = 7Fh (Index = 0?)
Call[index] = seed Decrement index
Seed = 0?
Set carry bit
Rotate seed right (Through carry, i.e., 0–>80, 80–>40, ...01–>0)
Save and restore seed
Call[index] = seed?
RAM error
Decrement index
Set Carry bit

Figure I--The RAM test code changes itself before relocating *to the* upper 128 *bytes of RAM. Expressions in italics refer to the code after self-modifications.*

| Cell Pattern | Carrv Bit |
|---|---|
| 10000000 | 0 |
| 01000000 | 0 |
| 00100000 | 0 |
| 00001000 | 0 |
| 00000100 | 0 |
| 00000010 | 0 |
| 00000001 | 0 |
| 00000000 | 1 |
| 10000000 | 0 |
| 01000000 | 0 |

Table **3**—*The walking-bit RAM test pattern sequence is copied in turn into every RAM cell. Every pattern is written and read back for consistency.*

C preprocessor. If you own an Acorn machine and develop for the 'HC 11, it's available free by anonymous ftp from ftp://mic2.hensa.ac.uk/micros/arch/riscos/b/b044.

## RAM TESTING: THE WALKING BIT

Figure l's RAM test consists of a write/read cycle of every cell in the 'HC1 l's RAM with a rotating-one pattern. Each cell is written with a mask made of one 1 and seven 0s (e.g., 00100000) and read back. Obviously, the written and the read value must match. The pattern is then rotated and the check repeated until all positions in RAM have been written to 1. This is accomplished through a rotate-with-carry operation that yields a series like the one in Table 3.

Bytes 128-256 are tested first. Since the test itself is destructive, the whole program must reside in fewer than 128 bytes, and since the bootstrap mode executes out of address 0, it must initially reside in the lowest half of memory.

The initial seed or pattern is 0, and the stack points initially to 7Fh. RAM cells are written from 256 to 128 with the walking-bit pattern and are subsequently read back to assure a match. The seed is then right-shifted one position and the whole write-read back cycle is repeated again. This pattern continues until the initial seed is shifted right for the eighth time into the carry bit.

At this point, the whole lower RAM has passed the test successfully and we must modify the code to adapt it to test cells O-7Fh. This modification is accomplished with the help of some labels.

Since the code must be assembled and tested before passing it to the device under test, a way of offsetting it to work independently from its ROM address must be devised. Setting Off - set 2 = c 0 2 forces the assembler to create an executable that can be tested while residing in the RAM code space of an EVM or an in-circuit emulator, for example.

Offset2 = 0 lets theprogrambe run in RAM from address 0 for downloading to the device under test. All the labels used in the program are indexed this way when it comes to modifying the code before moving it to the upper addresses.

Note that the branch instructions do not pose a problem since they are relative to the current program counter and hence are position independent. A final BRA * (branch here forever) is added to prevent program runaway at the end of the test.

After the changes, the code moves to upper cells and jumps to its start point, this time at 80h instead of 0.

## EEPROM TEST

There's also a way to program your own data into the 'HC1 l's EEPROM. The test itself is very easy.

Basically, a bulk erase operation (all EEPROM is written to FFh) followed by a check for equality to FFh is performed on every cell. The whole memory is then written to 0, checked, erased (with check), and then written with the final values. In this particular implementation, the final values are read and assembled from an external file (EEFILE.ASM).

Since the whole code must reside within the internal RAM, there's only space left for part of the EEPROM values to be written. Should a particular project need more, a loading routine can be executed which fetches data from the serial link as required.

## TIMER TEST

A timed loop is done to check the efficiency of the 'HC1 l's TCNT. Since the number of cycles is known in advance, a specific value is expected to be read in TCNT at the end of the test.

Notably, each timer/counter is not initialized at the beginning of the test. Instead, it is pushed onto the stack that in turn was initialized at FFh. The

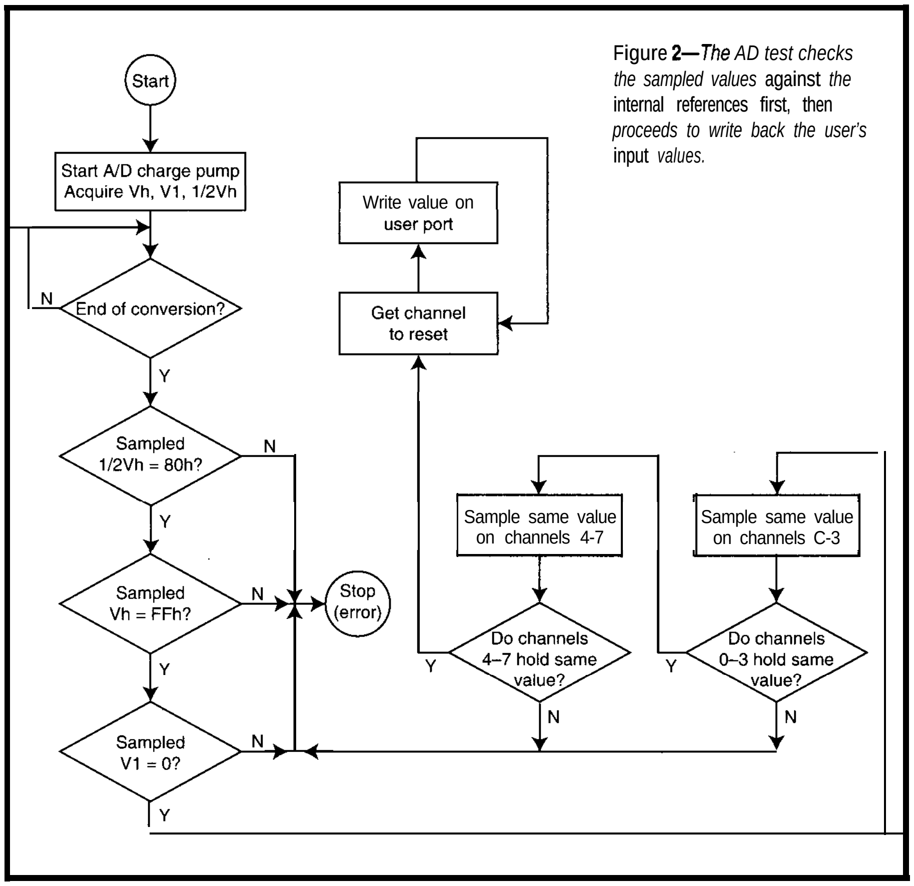


Figure **2**—*The AD test checks the sampled values against the internal references first, then proceeds to write back the user's input values.*

Listing 1—*Returning from a subroutine using the stack to pass parameters saves RAM and keeps code reentrant at a bit of speed expense.*

```
RTSUB MACR              Return from subroutine.
      IFEQ  NARG        If no parameters are passed, then
            TXS         return this way.
            RTS
            MEXIT
      ENDC
      IFC   '\1',''          \0 entry parameters pushed on stack.
            LDAB  #(2+\0)    \1 parameters are returned.
      ENDC
      IFNC '\1',''
            LDAB  #(2+\0-\1)
      ENDC
            LDY   0,X Called Subroutine MUST begin with TSX
            ABX         Caller MUST ensure useful regs are
            TXS         preserved (pshd) before calling the routine.
            JMP   0,Y \1 bytes are returned and MUST be popped
ENDM                    from stack by caller.
```

saved value is subtracted from the latched TCNT value. A difference of 128 cycles is expected between the old and new timer values. The loop is repeated forever, and several starting values of TCNT can be tested this way.

## A/D CONVERTER TEST

The 'HC11 itself provides some features which are intended only for factory test, but can be used to test the on-chip A/D converter circuitry. Of course, this assumes the right A/D channel assignment bits are selected. $V_{refhigh}$, $V_{reflow}$, and $\frac{1}{2}V_{refhigh}$ and are fed to the ADR1, ADR2, and ADR3 channels according to Table 4. The A/D conversion test first checks for any errors between the expected $V_{refhigh}$, $V_{reflow}$, or $\frac{1}{2}V_{refhigh}$.

Assuming the 'HC11's output pins are tied together and to an adjustable voltage source, two more tests, pictured in Figure 2, follow. The first test samples all the channels (4 x 4) and compares the values within each group of four. The values need to be the same to pass. The second test acquires a user-specified input channel and writes the converted value on a user-defined port for a coherence check.

## A LAST WORD ON THE CODE

The accompanying code implements what is described here. It compiles under the Motorola PASM 'HC11 macro assembler, but should be easily ported to other assemblers.

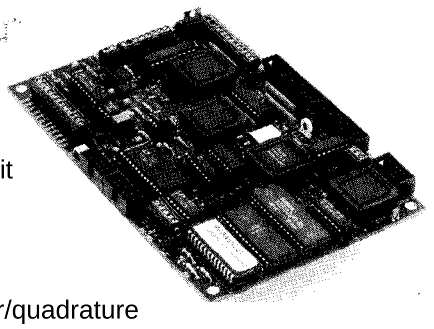
A point worth noting is the RTSUB macro shown in Listing 1. This macro provides a way to return from a subroutine that was called with stacked parameters. The RTSUB macro takes care of positioning the stack according to the number of parameters (in bytes) which are moving to and from the subroutine.

Since the called subroutine begins with a TSX instruction, the X register points to the stack and to the entry parameters that were pushed before calling the subroutine itself. On exit, the B register contains the number of bytes pushed on entry minus the bytes that must be returned as exit parameters on the stack itself.

This difference is added to the X register to reposition the stack pointer to the place we want it to be on exit. The Y register inherits the return address (LDY 0,X when X still points to the top of the stack). Executing an indexed jump to 0,Y returns the caller. It is the caller's responsibility to pull any return parameters from the stack.

With techniques like this one, it is also easy to allocate more local variables onto the stack, thus avoiding a waste of global memory, saving some headaches-ever asked yourself what would happen if this global variable

**Will Your Embedded Controller Do All This?**



- Measure 8 A/D lines with 12-bit accuracy.
- Control relays, motors, switches, etc., using 32 digital I/O lines.
- Motion control using a counter/quadrature encoder input.
- Support RS-485 networking through one of two buffered serial ports. Also supports RS-232.
- Provide operator interface separate from other digital I/O lines.
- Program easily using a PC.
- Save programs and data in 1 MB memory.
- Operate on only 5 volts. Use only 5 mW in idle mode.
- Provide built-in BASIC that supports all on-card hardware features and has floating point math.

The **RPC-320** will and Much More!

From $365 in 1's. Call for more information and our NEW catalog.

REMOTE" PROCESSING CORPORATION  303-690-1588

6510 W. 91st Avenue, Westminster, CO 60030, FAX: 303-690-1875

| CD | CC | CB | CA | Channel Signal | Result in ADRx if MULT=1 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | PEO | ADR1 |
| 0 | 0 | 0 | 1 | PE1 | ADR2 |
| 0 | 0 | 1 | 0 | PE2 | ADR3 |
| 0 | 0 | 1 | 1 | PE3 | ADR4 |
| 0 | 1 | 0 | 0 | PE4 | ADR1 |
| 1 | 1 | 0 | 1 | PE5 | ADR2 |
| 1 | 1 | 1 | 0 | PE6 | ADR3 |
| 1 | 1 | 1 | 1 | PE7 | ADR4 |
| 0 | 0 | 0 | 0 | Reserved | ADR1 |
| 0 | 0 | 0 | 1 | Reserved | ADR2 |
| 0 | 0 | 1 | 0 | Reserved | ADR3 |
| 0 | 0 | 1 | 1 | Reserved | ADR4 |
| 1 | 1 | 0 | 0 | $V_{REF\ HIGH}$ | ADR1 |
| 1 | 1 | 0 | 1 | $V_{REF\ LOW}$ | ADR2 |
| 1 | 1 | 1 | 0 | ½$V_{REF}$ HIGH | ADR3 |
| 1 | 1 | 1 | 1 | Reserved | ADR4 |

Table 4-The 68HC1 I's ADC can be tested using a channel assignment to feed reference values to the ADC channels, thus needing no external hardware.

was overwritten unexpectedly?-and keeping the routine reentrant. Unless brute speed is critical, I find this technique very useful.

## YOURS TO CALL

I have outlined a powerful 'HC 11 feature that can be effectively used to expand its usefulness. Several other tests can be carried out this way, depending on the specific need, and the bootstrap mode may be put to work in several other ways.

I find it particularly useful for writing and reading values to and from the EEPROM. Combined with the ability to jump to the start of EEPROM, the 'HC 11 can be reprogrammed in the field. For example, suppose you have a BASIC interpreter in ROM and you store your BASIC tokens in EEPROM. You can reprogram it on-the-fly to execute an entirely different set of instructions.

Check it out. Its special features are only limited by your needs and imagination!

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

## SOFTWARE

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

## REFERENCES

Motorola. *MC68HC11 **EEPROM Programming from a Personal Computer.*** Application Note 1010.

Motorola. *8-bit **MCU.*** Applications Manual, 1990.

Motorola. *M68HC11 **Reference Manual.*** 1991.

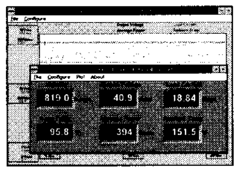
Peatman, John B. ***Design with Microcontrollers.*** Singapore: McGraw-Hill, 1988.

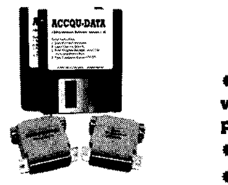
## I R S

410 Very Useful
411 Moderately Useful
412 Not Useful

**Tiny Data Acquisition System**
**Connects to any PC Printer Port**

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





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

**Sensors and Signal Conditioners for:**
**Temperature, Pressure, Humidity, Flow, and more**

| | | |
|---|---|---|
| AD612 | 6 channel, 12 bit, Data Acquisition module w/3 digital I/O lines.Includes software for displaying, plotting, and logging to disk. | $99.95 |
| TSC5-T | Thermocouple Signal Conditioner for use with AD612. Accepts 5 type T thermocouple inputs on channels 1-5 and displays ambient temperature on channel 0 | $79.95 |
| MSC01 | Multi-input Signal Conditioning Module for use with AD612 . Includes 2 type T Thermocouple inputs, 1 Differential input, 1 Tachometer input (for frequency output sensors), 2 voltage inputs, and 3 digital I/O lines. Specify Tachometer range: MSC01-L 0 to 100 Hz; MSC01-M 0 to 500 Hz; MSC01-H 0 to 1000 Hz | $89.95 |
| DIFF6 | 6 Differential input module for use with AD612. | $69.95 |
| PT-1085 | Terminal board. Converts D-sub connector to screw terminal block. | $19.95 |
| TP4-T | 4" Thermocouple probe (Type T) | $29 ea. (5 for $115) |
| PS- xxx | Pressure sensor for use with MSC01 or DIFF6. Specify pressure range: PS- 001 0 to 1 PSI Differential/Gage; PS- 030 0 to 30 PSI Differential/Gage; PS- 100 0 to 100 PSI Differential/Gage | $58.95 |

**ACCQU-DATA**
297 Route 72 West Suite C-175 Manahawkin, NJ 08050-2890
FAX (609) 597-1606 24 Hr FAX-BACK / INFO : (609) 597-I 221
BBS : (906) 774-7499 email:ACCQUDATA@aol.com
**TO ORDER : (800) 341-5590**
**OUTSIDE US : (970) 872-8355**

#122

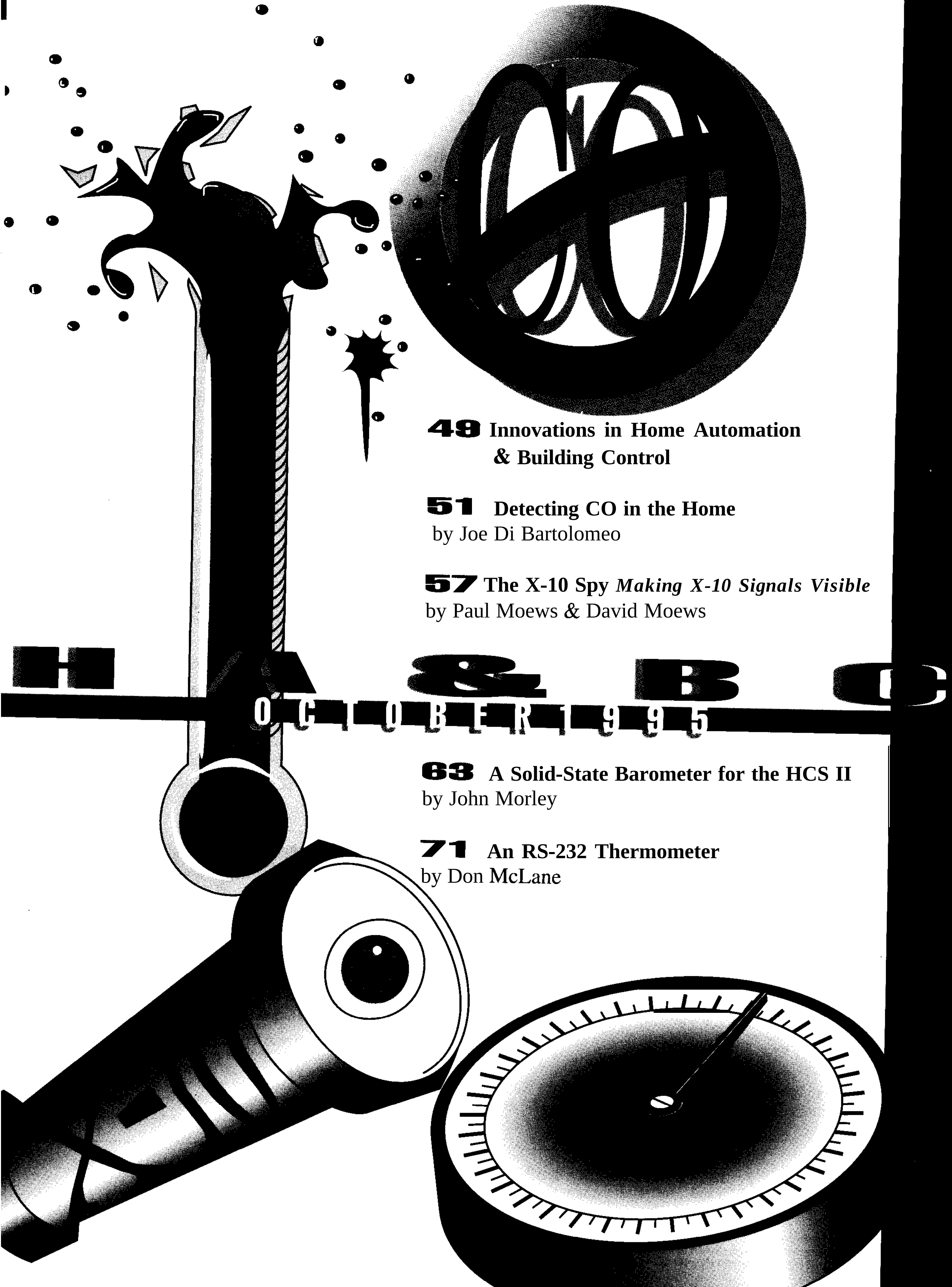
**The *only* 8051/52 BASIC compiler that is 100 % BASIC 52 Compatible *and* has full floating point, integer, byte & bit variables.**

- *Memory mapped variables*
- *In-line assembly language option*
- *Compile time switch to select 8051/8031 or 8052/8032 CPUs*
- *Compatible with any RAM or ROM memory mapping*
- *Runs up to 50 times faster than the MCS BASIC-52 interpreter.*
- *Includes Binary Technology's SXA51 cross-assembler & hex file manip. util.*
- *Extensive documentation*
- *Tutorial included*
- *Runs on IBM-PC/XT or compatibile*
- *Compatible with all 8051 variants*
- ***BXC51 $ 295.***

508-369-9556
FAX 508-369-9549

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

#123



# H A & B C

OCTOBER 1995

# InnoVAtions IN HOME AUTOMATION & BUILDING CONTROL

edited by
Harv Weiner

## HOME AUTOMATION SYSTEM

NetMedia introduces **TABS** (Totally Automated Building System), an integrated home and office automation system that combines device control (security, lighting, appliance, irrigation, and audio/video), voice messaging, paging, scheduling, energy management, and climate control into a comprehensive system. TABS interacts with its environment and the homeowner through a wall-mounted panel called the **TABS Communicator.**

TABS Communicators are placed throughout the home. Each Communicator monitors and maintains its own local environment. A keypad on the TABS Communicator and an infrared remote control offer convenient choices for entering system commands. The homeowner can define actions that perform sequences of events, such as "lights down, stereo on, and spa bubbling," with a single keypress.

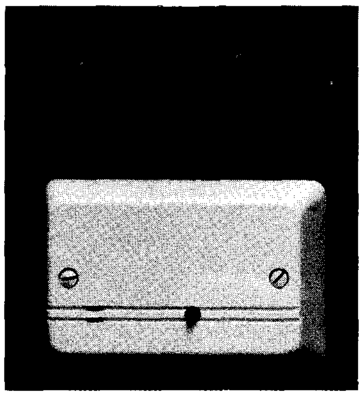
Voice and menus provide feedback and prompt for selections. For more complex functions, a TABS channel which visually represents all controlled systems is available on any TV in the home. A telephone interface offers world-wide access to the system. After verifying your password, TABS leads you through the options.

TABS Communicators are connected to a central computer system called the **TABS PC.** This is an IBM-compatible PC with specialized hardware designed by NetMedia. It includes the home control and communications hardware and software necessary to interact with multiple TABS Communicators. The PC makes sense of all the information collected from the Communicators and decides what should be done. The PC is dedicated to controlling the home and is not available for other purposes.

Wiring a home for TABS requires simple telephone-style wire. A complete system can be installed in typical homes under construction for **$5,000–7,500**.

**NetMedia, Inc.**
**10940 N. Stallard Pl. • Tucson, AZ 85737**
**(520) 544-4567 • Fax: (520) 544-0800**
**E-mail: netmedia@rtd.com**
**#508**

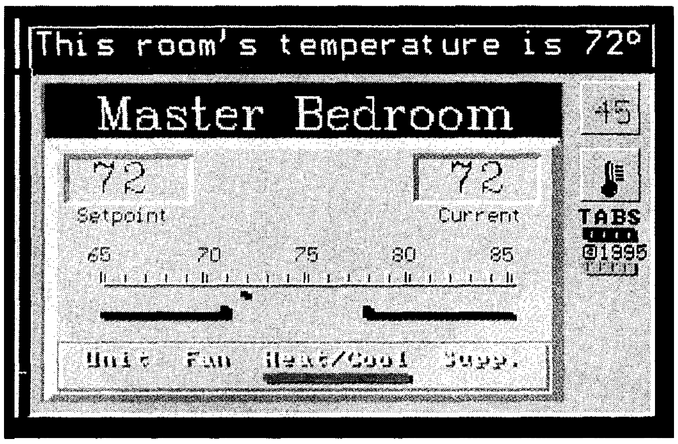
## HOME-SAFETY DEVICE



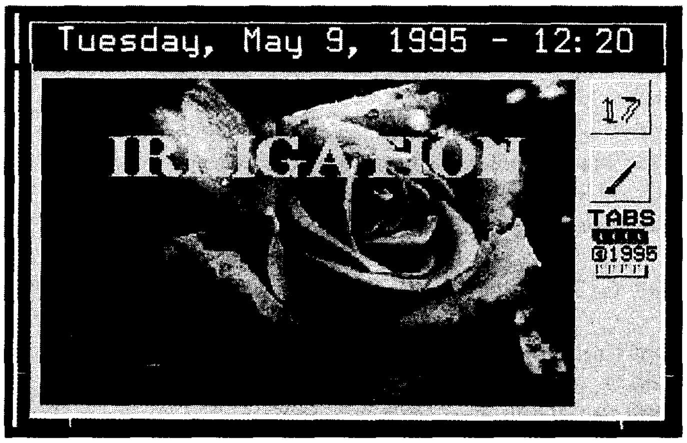
IRIS Electronics offers a low-cost solution to the problem of failing to turn off an appliance when leaving home. **StoveMinder** is an AC current sensor that connects to an existing home-alarm system and prevents it from being armed if a stove (or other appliance) is left on.

StoveMinder consists of a **PowerSensor** and display panel. The PowerSensor is tie-wrapped to the appliance power cord or AC cable feed and detects the electrical field around the AC cable when the appliance is drawing power. It ignores features such as the stove clock or light. StoveMinder is then wired to the alarm system like any alarm sensor. The PowerSensor can also be used for any other electrical device which, if left unattended, would pose a potential fire danger to the household.

**IRIS Electronics Corp.**
**931 Pinewood Cres.**
**Ottawa, ON**
**Canada K2B 5Y3**
**(613) 829-2514**
**Fax: (613) 829-5202**
**#509**

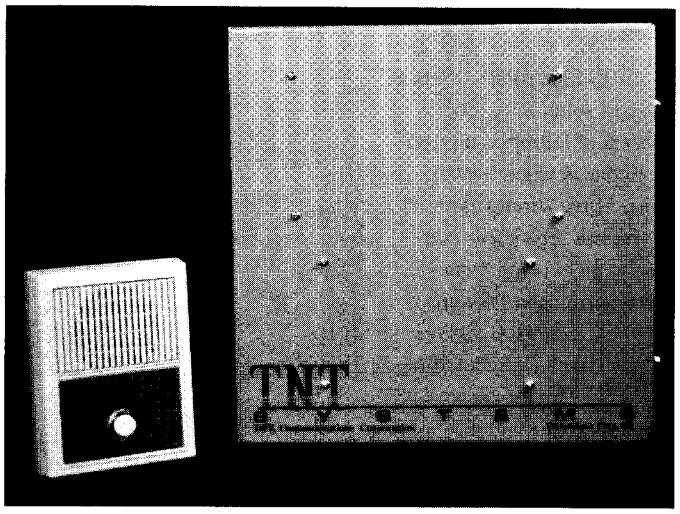


This room's temperature is 72°



Tuesday, May 9, 1995 - 12:20

# Innovations IN HOME AUTOMATION & BUILDING CONTROL



## PHONE-TO-PHONE INTERCOM

The **Touch-N-Talk 8000** from DFE Communications provides users with a phone-to-phone intercom on a standard single-line home or business phone system. By dialing a two-digit code, the user may call any other phone on the system. When that phone is answered, a private intercom path is established between the two phones.

TNT-8000 has three all-call zones which can be customized to meet individual needs (e.g., zone 1 can ring all zones, while zone 2 rings the upstairs and zone 3, the downstairs). For larger applications, TNT-8000 can be expanded by eight stations, giving the user a total of 16 selectable stations/phones.

Two-way communication with an entrance from any of the connected phones can be achieved with the optional TNT-8210 Door Answer Module. This module also rings the telephones when the doorbell is pressed by a visitor at the entrance. By adding the **TNT-**8410 Control Module, the homeowner can have up to four relays control a wide variety of home-automation functions such as gates, door strikes, or various X-10 controllers.

TNT-8000 retails for $589, TNT-8210 Door Answer Module for $109, and TNT-8410 Four-Zone Relay Module for $249.

**DFE Communications Corp.**
**1705 W. Main St. • Oklahoma City, OK 73106**
**(405) 232-2809 • Fax: (405) 232-2837** **#510**

## PROGRAMMABLE SCHEDULER/ CONTROLLER

TimeCommander-Plus, a computer-programmable scheduler and controller, integrates scheduling and advanced control of X- 10, infrared, and hard-wired input and output devices. Event Manager software simplifies programming and offers many advanced control features that can be customized for residential or commercial applications.

The unit integrates control of 256 X- 10 addresses and 16 optoisolated digital inputs from motion detectors, security sensors, thermostats, and so on. Eight analog inputs accommodate hard-wired temperature and humidity sensors and eight relay outputs (expandable to 152 inputs or 72 outputs) are also available. The unit can be programmed so ASCII text data triggers scheduled events. It can also send ASCII text data to trigger other computer programs or control an external modem.

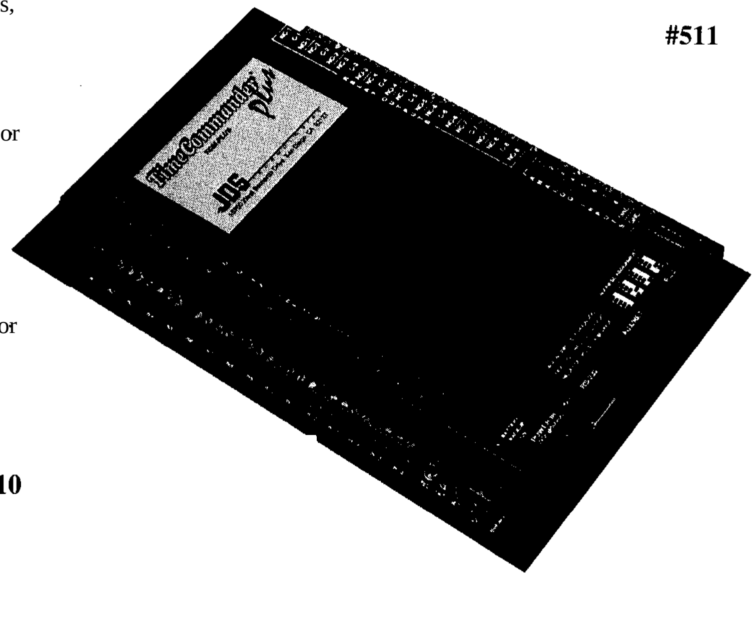
The IR-Xpander Infrared Interface controls audio and video components directly from any X- 1 O&compatible controller and/or automatically by preprogrammed schedule. Custom macros can turn on power, select source (AM/FM, TV, VCR), switch channels, set volume, close drapes, and dim lights.

The menu-driven Event Manager software (DOS or Windows) offers a host of advanced features such as macros, timers, counters, flags, and conditional programming. Sound files ( . WA V) can be programmed to respond to any input condition. An interactive on-screen display monitors and logs all X- 10 activity and offers direct control via mouse or keyboard. A modem can be connected for remote programming, control, and monitoring as well as advanced communication functions.

TimeCommander-Plus is available for $595 and the IR-Xpander option for $150.

**JDS Technologies**
**16750 West Bernardo Dr.**
**San Diego, CA 92127**
**(619) 487-8787**
**Fax: (619) 451-2799**

**#511**



# Detecting CO in the Home

JOE Di BARTOLOMEO

A lot of attention in home automation focuses on the quality of indoor air. With new safety regulations, this concern is going to grow. In this article, Joe offers a thorough review of what CO is and how it is detected before moving on to show how to integrate CO sensors in your own home-control system.

In home automation and control, a great deal of effort is devoted to "conditioning" indoor air. Among other things indoor air is heated, cooled, humidified, and dehumidified. An equally important, but often overlooked aspect of indoor air is its physical qualities. Through their home automation and control systems, people need to ask whether anything in indoor air can affect the health and/or safety of the occupants.

Health issues center on pollutants and toxins while safety centers on the combustion of hydrocarbon fuels. Any home with a combustion source should have two detectors, one for detecting fuel leaks (methane, propane, etc.) and the other for exhaust leaks.

Tin dioxide ($SnO_2$) semiconductor gas sensors are commonly used to detect both fuel and exhaust leaks. The most deadly exhaust gas, carbon monoxide (CO), is easily detected with tin dioxide sensors.

Tin dioxide sensors offer several attractive features. They are small, inexpensive, readily available, rugged, have a lifetime of five years, and are simple to use. The sensor undergoes a physical transformation when exposed to CO (i.e., its resistance changes). Rather than sensing CO, a CO detector quantifies the CO sensor response. If the CO level is too high, the detector sounds an alarm.

For many years, people were unaware of the risk of CO, and few homes were equipped with CO detectors. However, recent well-publicized cases of CO poisoning have greatly increased public awareness of the danger that CO poses.

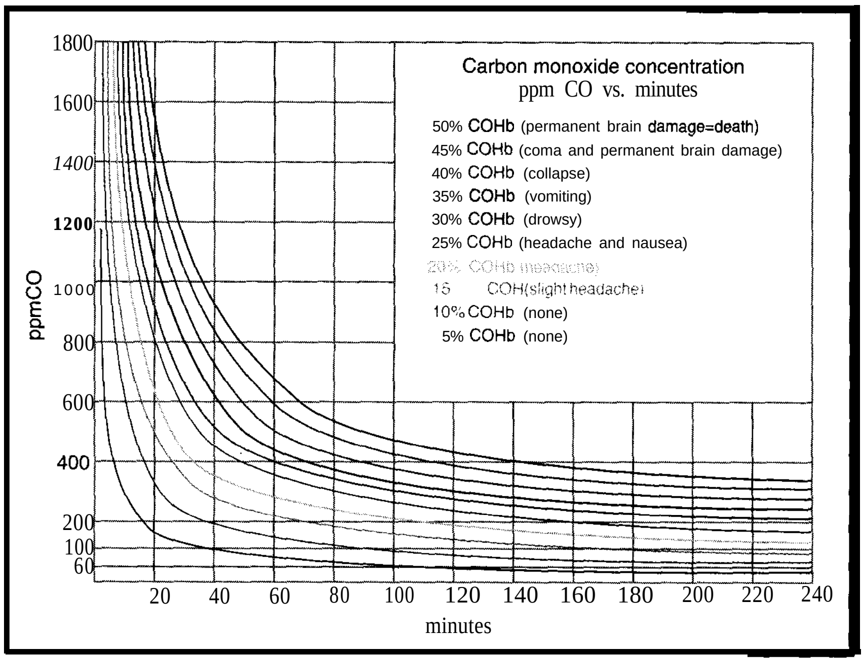


**Figure 1:** *Due to the nature of CO, both concentration (ppm) and exposure time (minutes) are important. CO effects on children and adults engaging in physical activity are more severe than those indicated here.*

This awareness has been accelerated by government legislation and the marketing efforts of CO detector manufacturers. In Chicago, every home must have a CO detector. This has led to a large increase in the purchase of CO detectors.

As with most consumer products, CO detector sales are price sensitive. Since no politician wants to face voters after passing legislation forcing them to purchase expensive CO detectors, there has been added political pressure to keep consumer costs down. Manufacturers now produce CO detectors in the $50 range.

Most of the consumer CO detectors use tin dioxide as the sensing element. For the average consumer, these CO detectors are fine, but for home automation and control, these sensors are too basic.

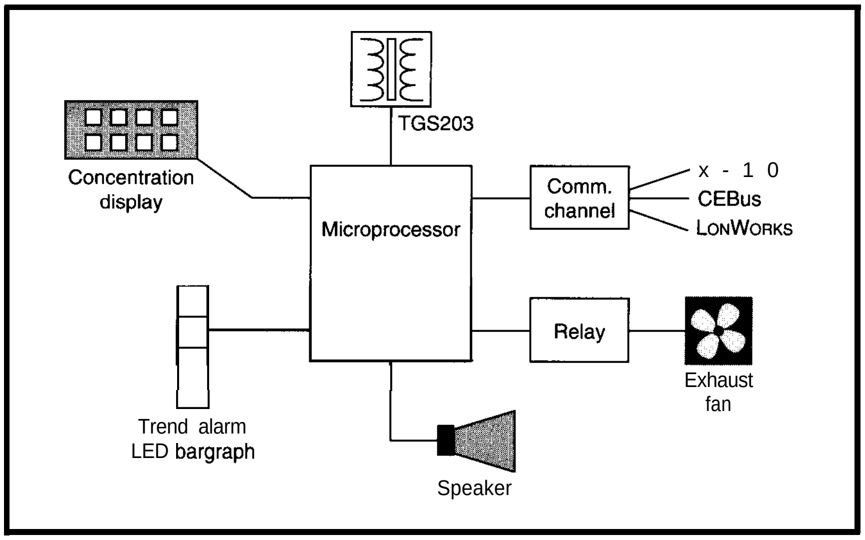


**Figure 2:** *A CO detector is ideal for use in home automation and control.*

## WHAT IS CO?

Carbon monoxide is an odorless, colorless gas that is extremely toxic. It is the leading cause of death by poison in North America. From 1979 to 1988, over 50,000 people died of carbon monoxide poisoning in North America, of these deaths approximately half were accidental. It's commonly referred to as the silent killer and is difficult to diagnose since the symptoms are similar to those of the common cold or flu.

Carbon monoxide is a byproduct of the incomplete combustion of hydrocarbons. During combustion, the separation of the hydrocarbon fuel releases energy. The separated carbon and hydrogen atoms bond with the oxygen in the air to form water vapor ($H_2O$) and carbon dioxide (CO,). If combustion is complete, no CO is produced.

Incomplete combustion occurs when there is an excess or deficiency of the combustion components. Normally, incomplete combustion is due to the air-to-fuel ratio being too low (i.e., there's not enough combustion air). In this case, the dislodged carbons cannot find enough oxygen atoms and carbon monoxide forms.

In the average home, several combustion sources powered by hydrocarbon fuels can potentially generate CO. The most common sources are furnaces, stoves (heating and cooking), space heaters, clothes dryers, and fireplaces. Properly installed and maintained, these appliances pose no threat.

When CO is inhaled, it inhibits the delivery of oxygen throughout the body, and the victim is asphyxiated. CO combines with hemoglobin (the oxygen carriers in red blood cells) to form carboxyhemoglobin (COHb). CO is particularly dangerous because hemoglobin's affinity to CO is much greater than its affinity to oxygen.

When the COHb level is about 15%, the victim experiences slight headaches and dizziness. At 25%, the victim has severe headaches and nausea. Between 30% and 40%. the victim vomits and may collapse. Exposure beyond 40% COHb causes permanent brain damage, coma, and eventually death (see Figure 1).

a) CO Conc.

| (ppm) | (Min.) |
|---|---|
| | |

b) CO Conc.

| | |
|---|---|
| ±5 | |
| +3/–5 | 480 |

*The UL specifications for CO detectors include both a response time (a) and a false-alarm resistance (b) criterion. Both are based on concentration versus exposure time.*

## THE TECHNOLOGY

There are several commercial manufacturers of CO detectors. Most adhere to the Underwriters Laboratories standard UL 2034, which states, "A carbon monoxide detector shall operate at or below the plotted limits for the 10% COHb curve."

As you can see in Figure 1, the time required to reach 10% COHb depends on the concentration of CO. Clearly, if one is exposed to 1000 ppm, a 10% COHb level is reached more quickly than if the individual is only exposed to 100 ppm.

An infinite combination of concentration and exposure time leads to 10% COHb. UL deals with this by ensuring that a CO detector responds in less than 90 minutes when exposed to 100 ppm of CO, in less than 35 minutes with 200 ppm, and in less than 15 minutes with 400 ppm since these exposure rates all cause a person's COHb level to be 10% (see Table 1). However, the detector should not respond when exposed to 100 ppm for 5 minutes or 15 ppm for 480 minutes. A CO detector built to the UL standard is sufficient for the average home.

Any CO detector intended for home automation and control should pass the UL standard, but needs additional features such as:

- display of the CO concentration
- an output that is compatible with CEBus, X-10, and LONWORKS
- a form-C relay for control of a venting fan
- a trend alarm

Figure 2 offers a basic setup for a CO detector in a home automation and control system.

# Detecting CO in the Home

JOE Di BARTOLOMEO

A lot of attention in home automation focuses on the quality of indoor air. With new safety regulations, this concern is going to grow. In this article, Joe offers a thorough review of what CO is and how it is detected before moving on to show how to integrate CO sensors in your own home-control system.

In home automation and control, a great deal of effort is devoted to "conditioning" indoor air. Among other things indoor air is heated, cooled, humidified, and dehumidified. An equally important, but often overlooked aspect of indoor air is its physical qualities. Through their home automation and control systems, people need to ask whether anything in indoor air can affect the health and/or safety of the occupants.

Health issues center on pollutants and toxins while safety centers on the combustion of hydrocarbon fuels. Any home with a combustion source should have two detectors, one for detecting fuel leaks (methane, propane, etc.) and the other for exhaust leaks.

Tin dioxide ($SnO_2$) semiconductor gas sensors are commonly used to detect both fuel and exhaust leaks. The most deadly exhaust gas, carbon monoxide (CO), is easily detected with tin dioxide sensors.

Tin dioxide sensors offer several attractive features. They are small, inexpensive, readily available, rugged, have a lifetime of five years, and are simple to use. The sensor undergoes a physical transformation when exposed to CO (i.e., its resistance changes). Rather than sensing CO, a CO detector quantifies the CO sensor response. If the CO level is too high, the detector sounds an alarm.

For many years, people were unaware of the risk of CO, and few homes were equipped with CO detectors. However, recent well-publicized cases of CO poisoning have greatly increased public awareness of the danger that CO poses.

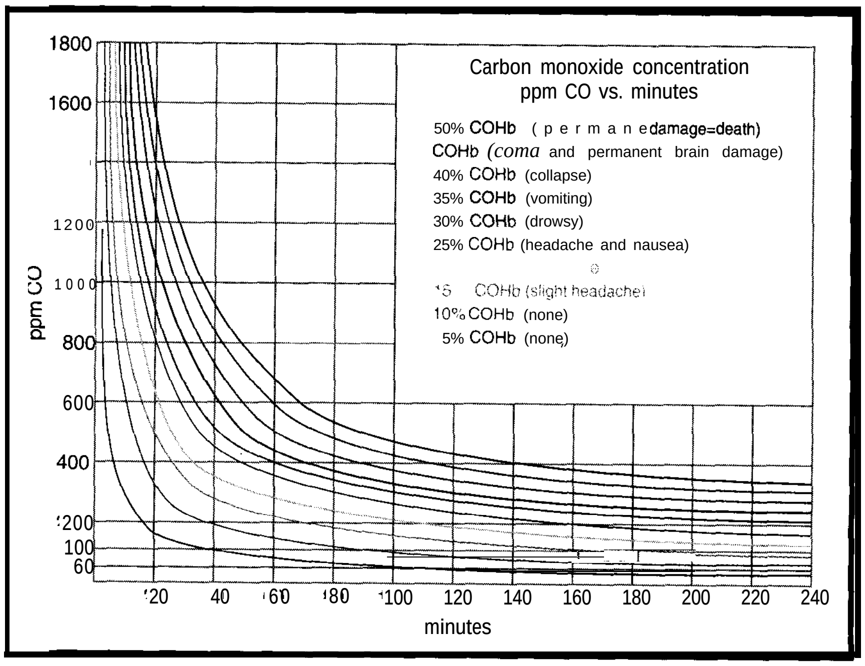


**Figure 1:** *Due to the nature of CO, both concentration (ppm) and exposure time (minutes) are important. CO effects on children and adults engaging in physical activity are more severe than those indicated here.*

of the tin dioxide sensor is highly nonlinear. The common approach is to normalize the sensor's response.

To do this, an arbitrary concentration is'chosen, usually 100–1000 ppm at 20°C and 60% RH. The resistance of the sensor at this concentration is designated as $R_o$. Using the resistance at 1000 ppm (R, is supplied by the manufacturer), you can determine the concentration by measuring the resistance of the sensor and scaling it to $R_o$.

Notably, even though the 8 13 responds better to CO than methane, it is rarely used as a CO detector since its operating temperature is about 400°C. As Figure 3 shows, the selectivity to CO increases as its temperature decreases. The curves illustrate how unselective these sensors can be and that care must be taken to select the proper sensor.

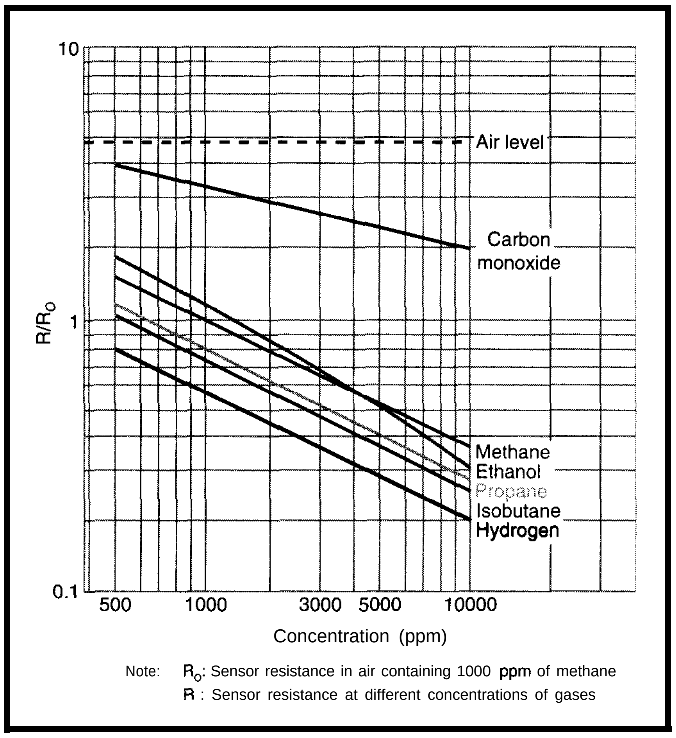


**Figure 4**: *The ratio of resistance to concentration is unique to each detector These statistics hold true for the Figaro 813. Note the concentration is plotted on a log scale.*

## FIGARO TGS203

The TGS203 manufactured by Figaro Engineering is commonly used for detecting CO. The sensing element is an n-type $SnO_2$ semiconductor, which has a resistance change when exposed to CO. The resistance decreases as the CO concentration increases.

As with all semiconductor sensors, the operating temperature determines sensitivity. The TGS203 is unique in that its operating temperature is below 100°C—a sharp contrast to other semiconductor gas sensors that operate in the 400°C range. At this low temperature, the TGS203 response to CO is slowed. However, there's a significant gain in selectivity from interference gases.

By running the sensor below 100°C, water vapor and airborne contaminants deposit on the surface of the sensor and cause interference. To eliminate interference, the TGS203 operates in a two-temperature cycle. The sensor runs at the high temperature for 60 s to boil off the water vapor and contaminants and then it is run at the low temperature for 90 s to stabilize it. Once the sensor stabilizes, a CO reading is taken.

As Figure 5 shows, the TGS203 is a four-pin device. The sensor has two heating elements and the sensing element is represented by a rectangular box in the center. In actual fact, the sensor is woven into the heater elements. Electrically, the sensor's terminals are between pins 1 and 3 or between pins 2 and 4. The user selects which pair of pins to use.

Suggested operating temperatures for the TGS203 are 300°C for the high temperature and 80°C for the low temperature. The sensor's temperature is controlled by the current passed through the heaters (the greater the current, the higher the sensor temperature).

To achieve the suggested operating temperatures, the high current should be 369 mA and the low current 133 mA. Figure 6 and Table 2 present a basic circuit used to measure CO using a TGS203. Q1, Q2, and Q3 are MOSFET switches and CC 1 and CC2 are constant-current sources.

The measurement cycle begins by running the heater at the high temperature to clean the surface. This is done by closing Q1 and Q2 switches and opening switch Q3. The cleaning cycle lasts 60 s. Next, the sensor is run for 90 s at the low temperature to stabilize it. This is done by closing switches Q1 and Q3 and opening Q2. In the final step, the sensor's resistance is measured. This is done by opening all three MOSFET switches and measuring the voltage V,. The sensor resistance can easily be found from:

$$R_s = \frac{V_C - V_L}{V_L} \times R_I$$

Knowing $R_s$ and $R_o$, the CO concentration can be determined. Figure 7 shows the sensitivity characteristics of the TGS203. Note when $V_L$ is read, no heater current is applied, which causes the sensor temperature to begin dropping. To keep water vapor and other contaminants from depositing on the sensor surface, this read time should be as short as possible.

## TESTING

To test your CO detector, you need a test setup. A proper test setup requires a test chamber capable of temperature and humidity control. The sample air flowing into the chamber must be controlled by flow controllers. Tanks of clean air and CO are required.

Of course, this is a very expensive proposition. Instead, a simple test setup can be made using a plastic container and its lid, a rubber stopper, syringe, and some silicone. Get an ordinary plastic container and sealing lid (e.g., Rubbermaid). Drill two small holes in the lid. Pass wires through one hole and place a stopper in the other. Seal around the holes with silicone to maintain an air seal. (The stopper needs to be made of self-healing rubber.)

Now, by inserting the syringe through the rubber stopper, inject a known amount of CO into the container. If you have a 100-l container, 1 cc of pure CO gives a concentration of 100 ppm. Using this method, virtu-

**Figure 5**: *The TGS203 CO sensor includes a sensing element, rectangular box, and a heater element.*

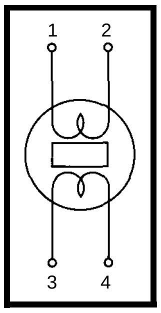


**Table 2:** *The TGS203 CO sensor requires two different temperatures (and hence, two current levels) for its various modes* **of** *operation.*

| Operation | Time | Heater Current | Sensor Temp. |
|---|---|---|---|
| Clean | 60 s | 369 mA | 300°C |
| Stabilizing | 90 s | 133 mA | 80°C |
| Read | >1 s | 0 mA | >80°C |

Trend alarms are very useful. If 100 ppm for 90 minutes is toxic, what about 90 ppm for 120 minutes or 50 ppm for 240 minutes? Any time there is an elevated CO level, say more than 40 ppm, a trend alarm actuates. Although elevated CO levels are not lethal, they indicate a problem.

## THE CO SENSOR

For more than 30 years, it's been known that the surface conductance of semiconductor oxides is influenced by the composition of gasses in ambient air. When the oxide surface is contacted by a gas it is sensitive to, its conductance changes. In effect, the semiconductor oxides are gas-sensitive resistors. This behavior is exploited to produce gas sensors from many different semiconductor oxides, tin dioxide being the most common.

The response of a tin dioxide semiconductor sensor is completely dependent on the reaction at the sensor-to-air interface at the sensor's surface. The dominate reaction at the surface of the semiconductor oxide sensors involves the change in concentration of surface oxygen species.

When an n-type semiconductor, such as tin dioxide, is exposed to an ambient that contains oxygen (e.g., air), the oxygen traps electrons. At the surface, a charge builds up and resistance changes.

The charge continues to build until saturation is reached and resistance stabilizes. The surface is now extremely sensitive to any change in oxygen concentration. When the sensor comes in contact with a reducing gas, such as CO, the concentration of surface oxygen decreases, which in turn decreases the sensor's resistance. The surface reaction for CO is:

$$2C0 + 0, \rightarrow 2CO_2 + e^-$$

Any gas accepting or donating an electron causes a change in the concentration of oxygen at the sensor surface. This means any reducing or oxidizing gas causes the sensor to respond. In fact, tin dioxide sensors are capable of sensing many gases.

This sensitivity can lead to a lack of selectivity. Fortunately, sensors can be "tuned" to a particular gas. Although several methods enhance selectivity, the only one in the user's control is sensor operating temperature.

The semiconductor sensor's response to any particular gas is greatly dependent on its operating temperature. Figure 3 shows the typical behavior of an $SnO_2$ sensor. When operated at 500°C, the sensor is more sensitive to CH, than to CO. However, at 375°C, it is more sensitive to CO than CH,.

Normally, sensors operate between 300°C and 600°C. For CO, the sensor actually operates below 100°C, where the semiconductor oxides respond well to CO but not to other gases. To further reduce interference, the sensors are fitted with an activated carbon filter.

The sensors are usually four-pin devices with two pins for the sensor and two for the heater. A constant voltage or current is applied to the heater, which maintains the sensor at a constant temperature.

The response of the sensor is measured through its resistance change. Figure 4 shows the response curve of the Figaro 8 13 tin dioxide methane gas sensor. As you can see, the response

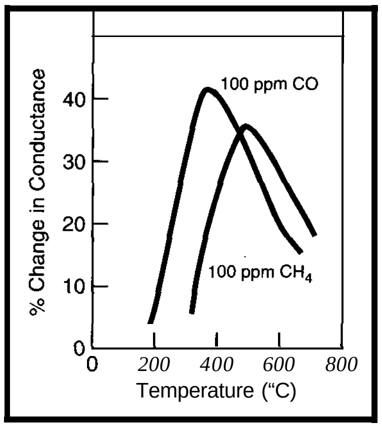


**Figure 8:** **CO sensor** *operating temperature can have a large effect on sensor response. Note that at lower operating temperatures, the sensor's response to CO improves.*

HEATING AND AIR CONDITIONING CONTROLS!
Largest selection of Heating and Air Conditioning products in the USA

*Zone Control Systems
'Inflatable Dampers
*Motorized Dampers

*Free Design Service
Send us Floor plans!!!

*Remote Control Thermostats*
(X-10, **Serial** ASCII Command Control **RS-485)**

**Enerzone Systems Corp.**
4103 Pecan Orchard Parker Texas 75002
(214)424-9808 * (214)424-8055 fax

#201

An Open Invitation...
Don't Miss Out!!!
Our next
EMBEDDED PC
Bonus Section
will appear in our
December issue!
Advertising Space
deadline: Oct. 11
Materials
deadline: Oct. 18
Call (860) 875-2199
or your ad rep.

ally any concentration can be obtained in the container. Although this method is not exact, it is sufficient for most applications.

With tin dioxide sensors, there are a few points to keep in mind. First, as briefly mentioned, temperature and humidity affect them. If the sensors have been on the shelf for a period of time, they must be burned in before use.

The cost of tin dioxide sensors is quite reasonable (approximately $12 for one). Their lifetime is about five years. Figaro makes using the TGS-203 very simple in that they sell a hybrid chip, the FIC 540 1, that controls the TGS203. Figaro's application package is quite good and helps you get the sensors up and running quickly.

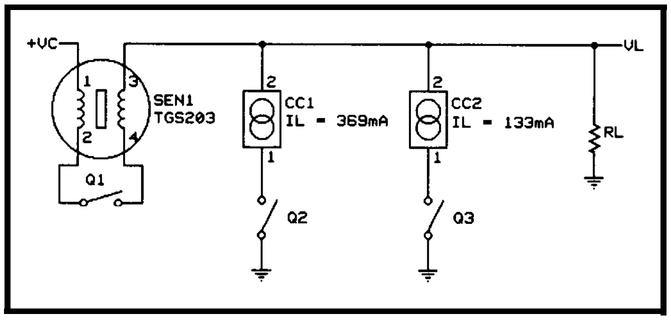


**Figure 6:** *Cleaning and stabilizing times can be varied to suit the application. However, the reading time should be kept as short as possible.*

## FURTHER APPLICATIONS

Tin dioxide semiconductor gas sensors are versatile. They can be used to detect many gasses. In addition to detecting methane (natural gas), they can detect ozone (generated by home air cleaners), CFC (from air conditioners and refrigerators), hydrocarbons (from paints and varnishes), and even smoke. In industrial situations, tin dioxide sensor can detect sulfurs and ammonia. And, in case you or one of your guests drank too much, they can even detect breath alcohol.

Nearly every municipality has outdoor air-quality standards. Monitoring stations throughout a municipality gather data for local pollution indexes. No such effort is made for home air, even though we spend 75% of our time indoors (90% in northern climes).

With growing public awareness of the danger of CO and other airborne compounds, legislation will be enacted. As the home automation industry is discovering in the communications debate between X-10, LonWorks, or CEBus, a standard is required for the industry to grow. This is also true for indoor air detection. A standard is needed so manufacturers can start producing detectors for the home automation market.

In fact, if a standards committee for detection of airborne toxins in the home has not been started, I'll start a committee. Do I have any volunteers?

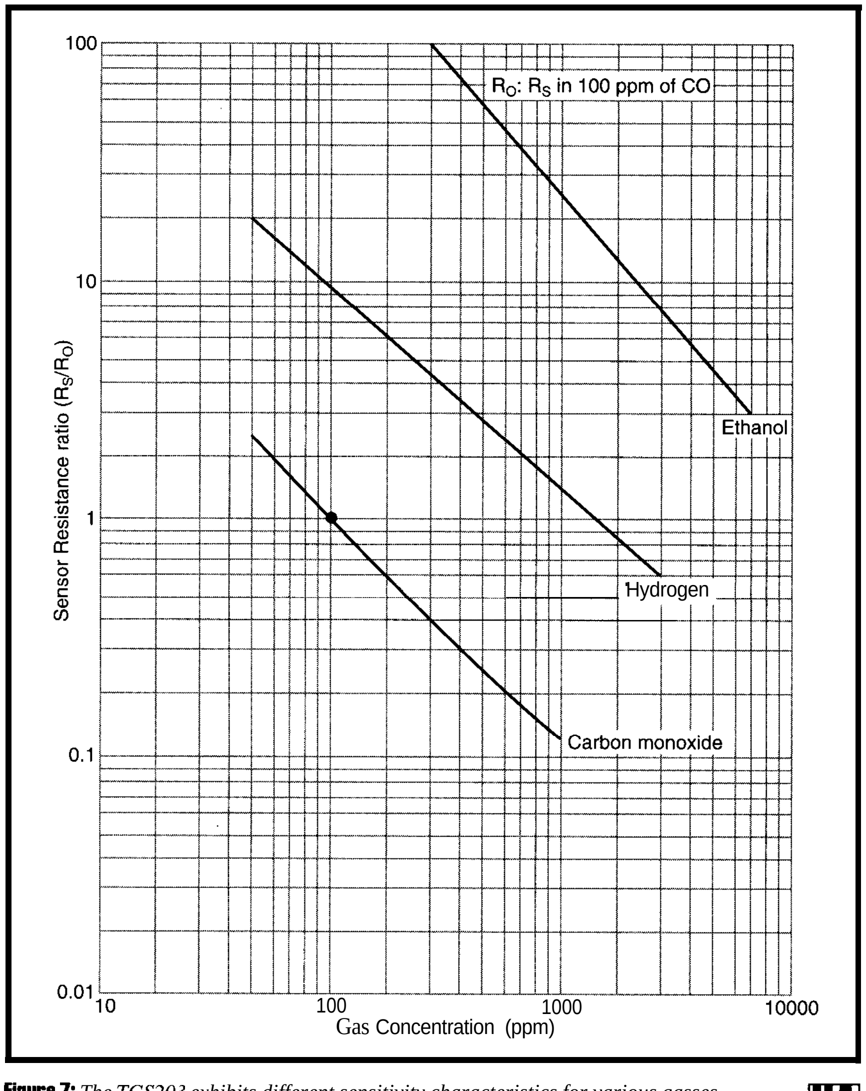


**Figure 7:** *The TGS203 exhibits different sensitivity characteristics for various gasses. Note that $R_o$ is taken at 100 ppm.*

*Joe Di Bartolomeo, P.Eng., is the chief engineer at Unisearch Associates, the world leaders in Tunable Diode Laser Spectroscopy (TDLS) for ambient air monitoring. He has worked extensively with low-noise analog electronics, embedded microcontrollers, artificial intelligence, and tin dioxide sensors. He may be reached at (905) 669-3547, ext. 234.*

**CONTACT**

Figaro Engineering, Inc.
P.O. Box 357
1000 Skokie Blvd., Ste. 575
Wilmette, IL, 60091
(708) 256-3546
Fax: (708) 256-3884

**I R S**

413 Very Useful
414 Moderately Useful
415 Not Useful

# The X-10 Spy

## Making X-10 Signals Visible

PAUL MOEWS & DAVID MOEWS

Although X-10 saves you the bother of running extra wires between your controller and appliances, dealing with the unpredictable power line can be a problem. Here's an X-10 monitor which can trace the timing and execution of commands. X-l 0 debugging will never be the same again.

The X- 10 system is popular for remotely controlling home lights and appliances. It sends modulated high-frequency signals over the power line, removing the need for direct wiring.

Recently, we thought about using X-10 to exchange commands and data between a computer and a remote monitoring device (e.g., a thermometer). X- 10 development becomes much easier if you can tell what's being broadcast and received. So, we decided to build the X- 10 Spy, which monitors the power line via a TW523 and displays X- 10 commands on an LCD.

The simplest example of X-10 equipment is a switch which can be turned on and off from a remote-control box. The switch plugs into an electrical outlet, and a device (e.g., a lamp) plugs into the switch. X- 10 switches have two 16-position rotary dials. One dial sets a house code (A–P), while the other dial sets the unit code (l-16). Our switch is set to house code A and unit code 1.

When we set the controller to house code A and push button 1, the lamp turns on. X-10 Spy then displays Al A2 1, where A is the house code, 1 is the unit code, and 21 represents the on function code (see Table 1).

### X-10 SIGNAL DETAILS

X- 10 signals are serial transmissions designed for a noisy environment. A word length of 9 bits includes a 4-bit word of 16 values and a 5-bit word of 32 values. The 4-bit word contains a house code, which is assigned one of the first 16 letters of the alphabet. The 32 values of a 5-bit word contain the key code and are divided into two groups of 16.

Codes in the first group are unit codes. Each code is assigned an integer from 1 to 16, which represents an X- 10 device such as

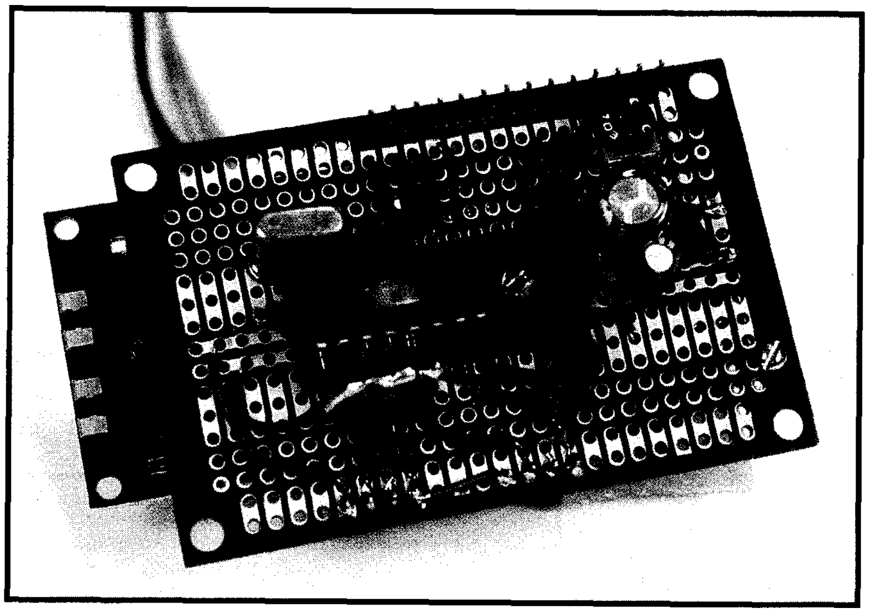


**Photo 1:** *The perfboard is mounted so that its component side faces downward into the enclosure. The LCD module, which has an edge connector soldered to it, is mounted on the reverse side of the perfboard.*

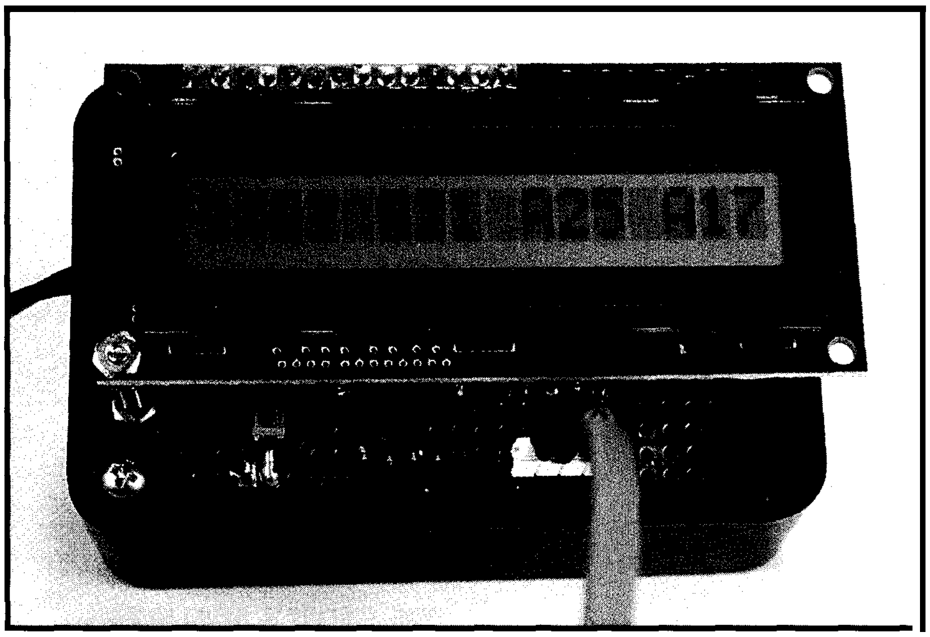


**Photo 2:** *In operation, X-10 commands scroll from right to left on the display. Each X-10 command is shown as one or two alphanumeric groups.*

our switch. Codes in the second group are function codes. Each code is assigned an integer from 17 to 32, representing instructions like on and off. Table 1 shows the correspondence between a function code's number and its effect.

Most asynchronous serial transmissions are timed by external clocks and are preceded by a start bit. Some use parity to find errors, while others use stop bits.

X-10 transmissions are timed by the zero crossings that occur on the power line at a rate of 120 crossings per second. Instead of preceding words with a single start bit, X-10 words start with the 4-bit string 1110. This bit string occurs only as the start signal in an X-10 transmission and is not used for data transmission.

Next, the 9 bits of the X-IO word are transmitted, with each bit is followed by its complement. There is no stop bit. Given this, an X-10 word is 22 bits long: a 4-bit start string with two 9-bit strings for data. Each 22-bit string is normally sent twice without an intervening pause to make a complete X-10 transmission.

To send Al, we send the start string of 1110 and then 01101001 0110100101, the result of interleaving 0110 01100 with its complement. This is then repeated. Similarly, to send A2 1, we send two repetitions of 1110 01101001 0101100110. (When sending these bit strings, the leftmost bit goes first.)

At the lowest level, the transmission of a single 1 bit on the power line consists of a 1 -ms burst of 120 kHz right after the zero crossing. The same burst is also sent 1.77 ms later and again 1.77 ms after that. The extra transmissions correspond to the zero crossings of the other two phases in a three-phase system. A 0 bit is denoted by the lack of any bursts.

## TW523 OPERATION

The TW523 simplifies the construction of a computer interface that transmits and receives X-10 signals. *INK 29* describes an interface which enables a PC to read and send X-10 signals (p. 74).

The TW523 connects directly to the power line and provides an optically isolated input, two optically isolated outputs, and a ground. The active-high input gates a 120-kHz signal onto the power line.

One output is a 60-Hz square wave with transitions coincident with zero crossings of the power line. The second output is used for valid X-10 signal data received from the power line. A 1 bit causes this output to pulse low just after a zero crossing. No output signal occurs for a 0 bit.

The TW523 not only receives and demodulates the X-10 signals, but also checks them for errors. Experiments show that the TW523 contains a 22-bit buffer. New bits demodulated from the power line are shifted in at the right, while old bits are shifted out at the left. If at any time the 22-bit buffer contains a valid X-10 command, this X-10 command is output on the appropriate pin of the TW523 timed to the zero crossings of the power line.

For each 1 bit, the receiver output pin is held low for around 1 ms after the transition of the TW523's zero-crossing output. During a 0 bit, the receiver output pin is high (i.e., undriven) during this time. The TW-523 does not seem to monitor the power line when it is outputting its received command. It resumes monitoring only after three additional half-cycles have passed.

As stated, X-10 commands are normally transmitted twice. If the first copy of a command is correct, the TW523 normally receives and outputs it, ignoring the second copy whether or not it is correct. If the first copy is garbled and the second is correct, the TW523 ignores the first copy and receives and outputs the second.

Since the TW523 only sends out correct commands of its receiver output, error-checking is not necessary. Since it can only receive one out of every two commands, it is not necessary for us to throw away every other command.

## HARDWARE

The TW523 is isolated and the only part connected to the power line, so we don't have to worry about building a shock hazard. Speed is also

**Table 1:** *The numericfunction codes output by the X-10 Spy are the same as those defined by X-IO, except 1 has been added to each number.*

| | | | |
|---|---|---|---|
| 17 | All Units Off | 25 | All Lights On |
| 18 | Hail Request | 26 | Hail Acknowledge |
| 19 | Dim | 27 | Bright |
| 20 | Extended Data | 28 | Status = On |
| 21 | On | 29 | Off |
| 22 | Preset Dim Low | 30 | Preset Dim High |
| 23 | All Lights Off | 31 | Extended Code |
| 24 | Status = Off | 32 | Status Request |

not an issue since X- 10 signals transmit slowly.

A small, slow, and cheap microcontroller such as Motorola's MC68-HC705K1 (it's less than $3 in 1000s) met our needs. As Figure 1 shows, the total circuitry consists of the 'HC705-K1, the TW523, a push-button switch, an LCD module, and some glue.

The 'HC705K1 has a 8-bit port (PA), a 2-bit port (PB), and an interrupt line (*IRQ), which we used as a general-purpose input line. PB 1 connects to the transmitter input of the TW523 and PBO connects to the receiver output. *IRQ connects to the zero-crossing output of the TW523.

Since all TW523 outputs are open-collector, PBO and *IRQ must be pulled up. PAl-PA7 drive the LCD module, and PA0 is pulled up and grounded via a normally open push-button switch. PB I is not used in the X-IO Spy, but can be used in other applications of this hardware (see "Other Applications" section).

We assembled our circuit on a 1%" x 2 % " multipurpose

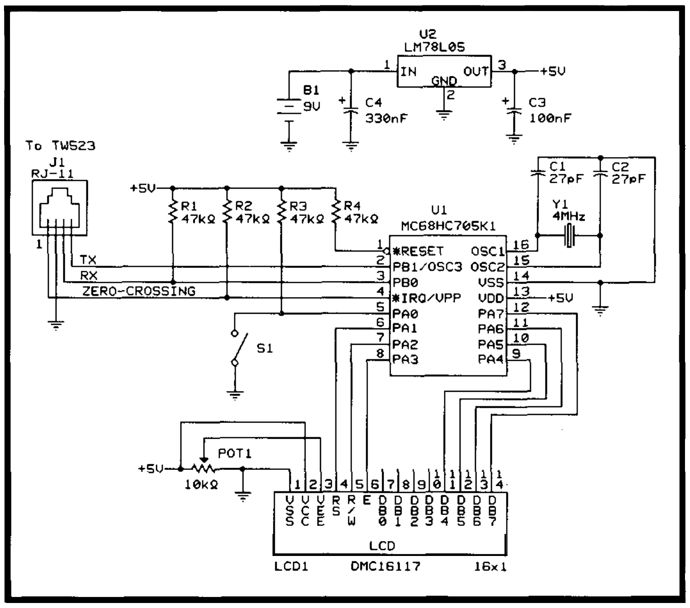


Figure 1: *The* **X-lo** *Spy requires almost no hardware besides the microcontroller, the TW523, and the Optrex LCD module.*

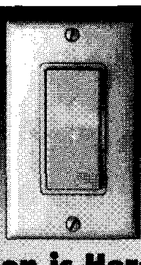
point-to-point wiring. Photo 1 shows the component side of our PC board and the back of our LCD module. The PC board mounts component side down in a project box (3%" x 2%" x 1⅛"). A 14-pin header on the wiring side of the PCB plugs into a 14-position SIP socket which is soldered to the LCD module. It also contains a header for connecting to the TW523 and a push-button switch.

## X-10 SPY OPERATION

To use the monitor, you turn it on and connect the TW523 to a power line. **X-10 Spy** appears on the LCD. Pushing the button causes the device to start monitoring.

**The Next Generation in Lighting Control**

**Introducing . . . PCS**

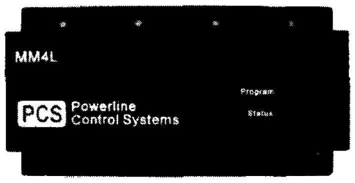


**Features include:**

★ Soft-Start
★ Pre-Set Dim
★ Brighten From Off
★ Dim Level Retained
★ 200 Dim Levels
★ UL Listed
★ 100% Money Back Guarantee
★ Made in the U.S.A.

**The Complete X-10 Compatible Lighting Control Solution is Here!**

Powerline Control Systems has combined the **reliability** of hard-wired systems with the ease of X-IO installation. Our full line of 1000, 1500, and 2000 Watt Lighting Control Modules cover all of your high power lighting needs at a fraction of the cost of other alternatives. All PCS modules offer inductive load control, improved receiving circuitry, **EEprom** memory, advanced programming and numerous other features not found on any other X-IO compatible **modules. These** products are UL listed and available now through all major **Home** Automation distributors.



X-l0 is a trademark of X-10 (USA) Inc..

**For more information about PCS's lighting control products call:**

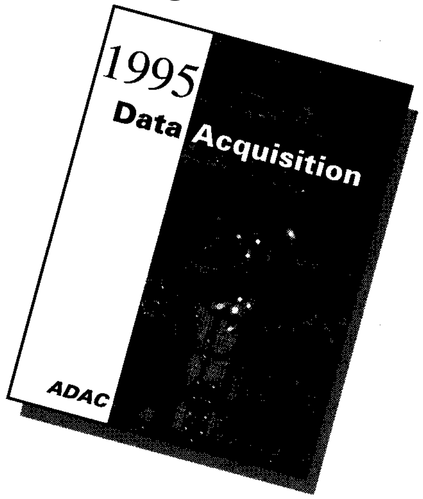
**1-800-727-3445**

**Powerline Control Systems ★ 9031 Rathburn Avenue ★ Northridge, CA 91325**
**(818) 701-9831 ★ FAX: (818) 701-1506 ★ E-MAIL: PCS@infodial.net**

#204

FREE

**Data Acquisition Catalog**



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

**1-800-648-6589**

ADAC

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

#205

**Listing 1:** *The top level of the program reads and decodes signals from the X-IO interface. The display logic is handled by subroutines which are not shown here.*

```
// The reset vector of the HC705K1 is set to point to START,
// so that execution begins at START when the X-10 Spy is
// turned on. We come here to pause the X-10 display.

PAUSE: Wait for PA0 to go high (wait for switch release)
   Call LDCMD and LDDATA to display an asterisk at the left
     hand end of the display
   Wait for the switch to be pressed and released
   Jump to EINTST

// Execution normally starts here

START: Set PA0 and PB0 to be input: set the others for output
   Set DELAY to 100
   For I from 1 to 5, call LDCMD (LCD initialization byte I)
   Call BUFMSG("X-10 Spy")
   Call REFSH
   Set DELAY to 1
   Wait for the switch to be pressed and released

// Here. we look at the X-10 input after startup, a pause, or
// a period of time with no detectable zero-crossings

EINTST: Set the 'display invalid' flag (a bit in RAM)

// Come here at the start of each new X-10 command

EINT0: Clear our X-10 command buffer

// Come here for each new bit of an X-10 command. Wait for a
// power-line zero crossing before reading new X-10 cmd bit

EINT: Save *IRQ state in a flag (also in RAM)
   If *IRQ becomes unequal to this flag, go to EINT3
   If the switch is pressed (PA0 goes low), go to PAUSE
   If -20 ms elapses without either event, call BUFMSG (no
     crossovers), call REFSH, and go to EINTST

// A zero crossing has just occurred. Wait to read the X-10
// input, since the TW523 only guarantees validity of the
// input signal from 200 µs to 1 ms after the crossover

EINT3: Delay for -500 µs

   Shift PB0 (the X-10 input bit) into the right end of our
     X-10 command buffer.

// If this is the first X-10 bit after a pause, startup, or
// a period with no crossovers, we need to clear the display
   If the display invalid flag is set, reset it, call
     BUFMSGO. and then call REFSH

   Check if bits 1110 are found 18 bits to the left of the
     rightmost end of our buffer.  If not, go back to EINT for
     another bit

// At this point, an X-10 command has just been read in
   Shift to extract the 4-bit house code and 5-bit key code
     from the rightmost 18 bits of our command buffer
   Call MOVE with a blank character
   Call MOVE with character representing the house code
   Call MOVE with first character representing the key code
   Call MOVE with second character representing the key code
     (if nonblank)
    Call REFSH
    Go to EINT0 for another command
```

The display then clears and X- 10 signals appear on the right and scroll to the left. Photo 2 shows the assembled monitor in operation. Each X- 10 command is displayed-house code (A-P) followed by key code (1–32). Key codes that are unit codes display as themselves, while key codes that are function codes display Table 1's numbers.

If the zero crossing output from the TW523 is not active (i.e., the TW523 is disconnected from the power line), "No crossovers" appears. If the zero-crossing output becomes active again, the display clears and monitoring resumes.

The monitor can be paused by pressing the button. An asterisk appears at the left of the display, and no X-10 signals display. Repushing the button removes the asterisk, clears the display, and resumes monitoring.

## CODING CONSIDERATIONS

Although the 'HC705K1 contains only 32 bytes of RAM and 504 bytes of EPROM and is at the low end of the 'HC05 family, it provides enough performance for the X- 10 Spy. Pseudocode for the top level of our program decodes the X-10 signals from the power line and is shown in Listing 1.

The code waits for a zero crossing on the power line, shifts a bit of X-10 data from the power line into a 22-bit buffer immediately after each zero crossing, and displays the valid commands on the LCD. However, this is complicated by initialization details, pauses, and the possibility of losing zero crossings from the power line.

Two l-bit flags stored in RAM affect the control flow of the program. The first flag, the display-invalid flag, indicates when the display needs to be cleared before displaying X-10 signal data on it.

The second flag is used when we are at EINT and wish to wait for a zero crossing of the power line. Since *IRQ is connected to the zero-crossing output of the TW523—a 60-Hz square wave synchronous with the power-line signal-we can do this by waiting for a change in the level of *IRQ. The flag then stores the old level of *IRQ so we can tell when it has changed. HABC

The LCD interface is managed by various subroutines. A 16-byte buffer mirrors the characters displayed on the LCD. MOVE puts a character into the rightmost end of this buffer, scrolling the existing characters to the left. BUFMSG blanks the buffer and places a message at its left-hand end. REFSH actually sends the buffer to the display by calling the low-level routines: LDCMD and LDDATA.

These routines take a byte in the accumulator and send it to the Optrex module as a command or as data, respectively. They differ only in the level they assign to PA1 (Optrex RS) during writing. Since we interface to the LCD over a four-bit bus (PA4–PA7), these routines must send a byte out as two nybbles. PA3 (Optrex E) is used as an active-high strobe for each nybble. It is held high for approximately 50 µs times the value held in DELAY.

During initialization, some commands take about 4 ms to execute, so DELAY is set to 100. After this, the commands only take 40 µs to execute, so DELAY is set to 1. As a consequence, REFSH takes around 3 ms to complete after initialization. Since we call REFSH at most once per X- 10 bit and there

JMD *JaMar Distributing*

ORDERS
800-477-4181
1292 Montclair Drive
Pasadena, MD 21122

All of the products
All of the tech support
All the best prices
All at one place!

*64 PAGE CATALOG SENT WITH ORDER*
HELP 410-437-4181
FAX 410-437-3757

**X-10 COMPATIBLE THERMOSTAT**

This is the one everyone's been waiting for. By utilizing the on and off commands from one house code you have 32 commands available giving you complete remote and automatic control of your HVAC.

ZCTS INCLUDES TW523 INTERFACE $199

$149.99 INCLUDES 2 BEEPERS PRIVATE PAGE EXTRA BEEPERS $39.99
Local wireless paging system has a range of up to 2 miles
Great for restaurants, small businesses, camping or calling the kids for dinner

**RAINDRIP IRRIGATION**

| | | |
|---|---|---|
| Drip watering landscape kii | 28.99 | These kiis are expandable |
| Container drip watering kit | 14.99 | Complete catalogs and prices |
| Drip soaker vegetable kii | 28.99 | are sent with kiis or send $5 |
| 24-volt 3/4" automatic anti-syphon valve | 24.24 | for catalog and planning guide |
| UM506 to control valve w/X-10 | 10.97 | redeemable on first order |

BOOKS, LITERATURE AND SOFTWARE
"A PRACTICAL GUIDE TO HOME AUTOMATION" $18.99 "THE HOME AUTOMATION NETWORK" $1.00
'TELEPHONE WIRING SYSTEMS" $2.50
"HOW TO TROUBLESHOOT PREMISE WIRING SYSTEMS FASTER AND MORE EFFICIENTLY- $3.95
"DECORA HOME CONTROL TECHNICAL MANUAL $8.99 "3D LANDSCAPING" (WINDOWS) $79.95
"GET WIRED!" PC BASED BOOK ON HOME WIRING $29.9 "3D DECK" (WINDOWS) $79.95

KX-T30810 PANASONIC HYBRID TELEPHONE SYSTEMS $586
KX-T9500 PANASONIC 900MHz CORDLESS TELEPHONE $199
X-10 MODULES MIX MD MATCH $139 DZ (LM465,WS467,AM486)

BECAUSE OF THE WEIGHT THE IRRIGATION CATALOGS ARE ONLY SENT TO EITHER PEOPLE PURCHASING A STARTER KIT OR THOSE SENDING IN $5 FOR POSTAGE AND HANDLING
TO BE PLACED ON OUR MAILING LIST JUST PLACE ONE ORDER
DEALERS WRITE OR FAX ON COMPANY LETTERHEAD

#206

Call 24hrs for FREE
64 page Color Catalog

Largest Selection of HOME AUTOMATION products in the World!

HOME AUTOMATION SYSTEMS, INC.
Pre-Set X10 Dimming
Wholesale Pricing With Retail Service
FREE

*Over 500 unique products!*

Hundreds of hard-to-find automation, X10, and wireless control products. Computer control of your home, security systems, surveillance cameras, infra red audio/video control, HVAC, pet care automation, wiring supplies, and much, much more!

HOME AUTOMATION SYSTEMS, INC.

*800-SMART-HOME*

#207

is 1/120 s or about 8.3 ms between each pair of X- 10 bits, we are never in danger of missing a bit because we're busy refreshing the display.

## OTHER APPLICATIONS

There are other applications for the TW523 and 'HC705K1 combination. For instance, you could make other remote sensors (switches, humidity detectors, etc.), remote actuators (motor control), remotely controlled displays and message boards, and so on.

We made a thermometer with an X- 10 interface using the TW523, the 'HC705K1, a temperature sensor, and a frequency-to-voltage converter. The device listens for a suitable unit code followed by an "on" key code to be sent via X- 10. It interprets this as a command to send the temperature and returns 16 bits of temperature data via preset dim X- 10 commands.

When sending one of these commands, the thermometer always uses the key code "preset dim low" and uses the house code to contain 4 bits of information per command, so it takes four commands to send 16 bits. Since to our knowledge no X- 10 device uses the preset dim command, there is no danger of interfering with an existing X- 10 system. *[Editor's note: Powerline Control Systems (818-701-9831) does make a number of lighting control modules that respond to preset dim.]*

The TW523 data sheet, however, defines a protocol for the preset dim command. This command potentially contains 5 bits of information. The most-significant bit determines whether the preset dim command uses a "preset dim low" key code (when the bit is 0) or a "preset dim high' key code (when the bit is 1). The least-significant bits are sent in the house code.

This thermometer can be polled using a PC via a parallel X-10 interface or, with appropriate software, the X- 10 Spy can also be used. In this application, we programmed a push-button press to send the temperature-sending command to the thermometer. The device then waits for the thermometer's response, converts the resulting temperature to Fahrenheit, and displays the resultant figure.

We leave it up to you to come up with more applications for the 'HC705K1 and the TW523.

**SOFTWARE**

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

*Paul Moews is a research associate at the University of Connecticut, where he maintains crystallographic software and data-collection equipment. He may be reached at moews@xraysgi.ims.uconn.edu.*

*David Moews recently received a Ph.D. in mathematics and currently maintains and develops software at Parametric Technology Corporation. He may be reached at dmoews@xraysgi.ims.uconn.edu.*

**IRS**

416 Very Useful
417 Moderately Useful
418 Not Useful

# A Solid-State Barometer for the HCS II

JOHN MORLEY

Want to predict weather with more than your bones? Then, listen up. John lets us in on his design, shows us how to calibrate it, and then hooks it up to the HCS II. **Presto!** You have the option of including atmospheric pressure readings in your home control system.

No doubt, you've heard, "the weather is in for a change. I can feel it in my bones," and have discounted it as an old wives tale. However, there is a medical basis for the statement.

Blood gases (primarily oxygen and nitrogen) dissolved in the muscles, ligaments, and connecting tissues of our bodies exert minute forces on the surrounding tissue in response to changes in atmospheric pressure. The forces exerted by these bubbles of gas are most apparent when the atmospheric pressure changes significantly over a relatively short period of time.

Since these forces are incredibly minute, they are most often felt by people who are older or have suffered a soft tissue injury. This condition parallels on a smaller scale what happens to scuba divers when they experience the painful and sometimes fatal condition known as the bends.

Let's start by looking at what atmospheric pressure is. From this basis, I'll show you how to use it to your own advantage when making your own weather observations?

## THE BASICS

Atmospheric pressure is the result of the gaseous atmosphere surrounding our planet. Imagine a 1" square column of air extending from the ground to the farthest reaches of the atmosphere. Atmospheric pressure is the force of the air molecules' weight in the hypothetical column, which is exerted on the lower atmosphere.

The nominal force that these molecules of air exert on the lower atmosphere is equal to 14.7 psi. The ambient atmospheric pressure, however, is in constant flux due to

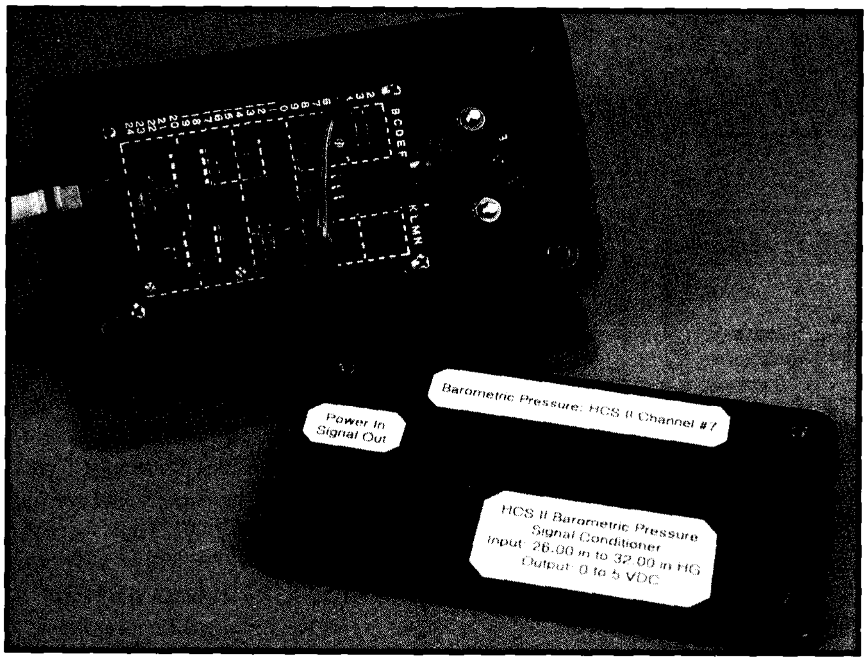

Barometric Pressure: HCS II Channel #7
Power In
Signal Out
HCS II Barometric Pressure
Signal Conditioner
Input: 26.00 in to 32.00 in HG
Output: 0 to 5 VDC

**Photo** *1: The prototype circuit was constructed using perfboard and mounted in a plastic enclosure. The SenSym pressure sensor is mounted off the circuit board to the right of the board.*

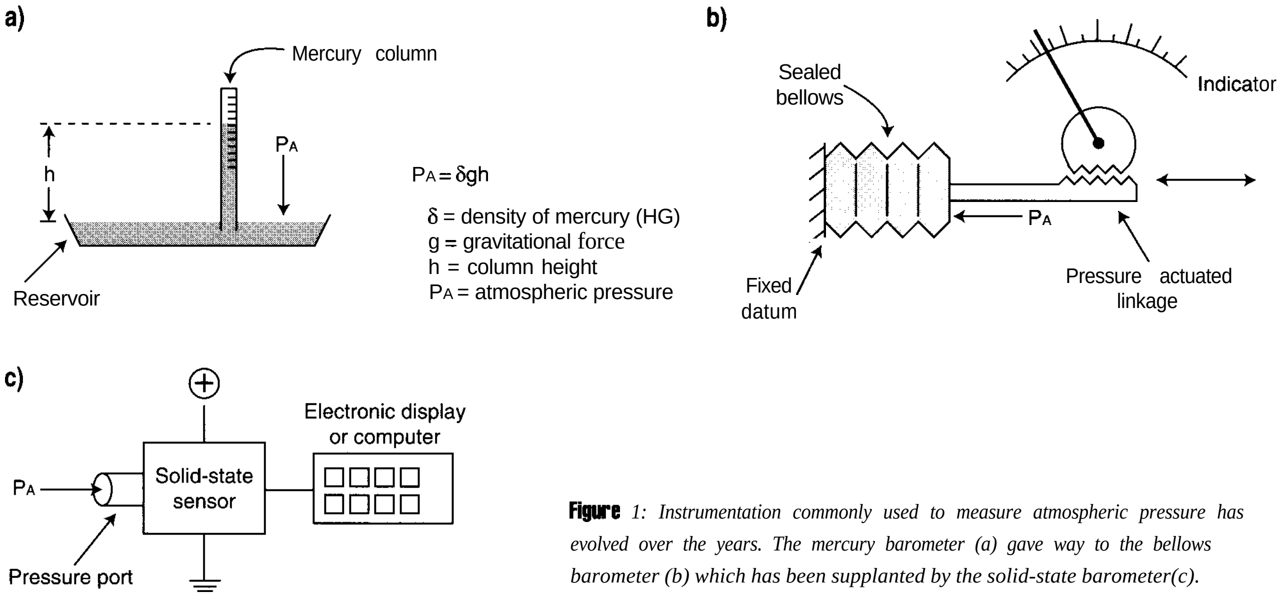


**Figure** *1: Instrumentation commonly used to measure atmospheric pressure has evolved over the years. The mercury barometer (a) gave way to the bellows barometer (b) which has been supplanted by the solid-state barometer(c).*

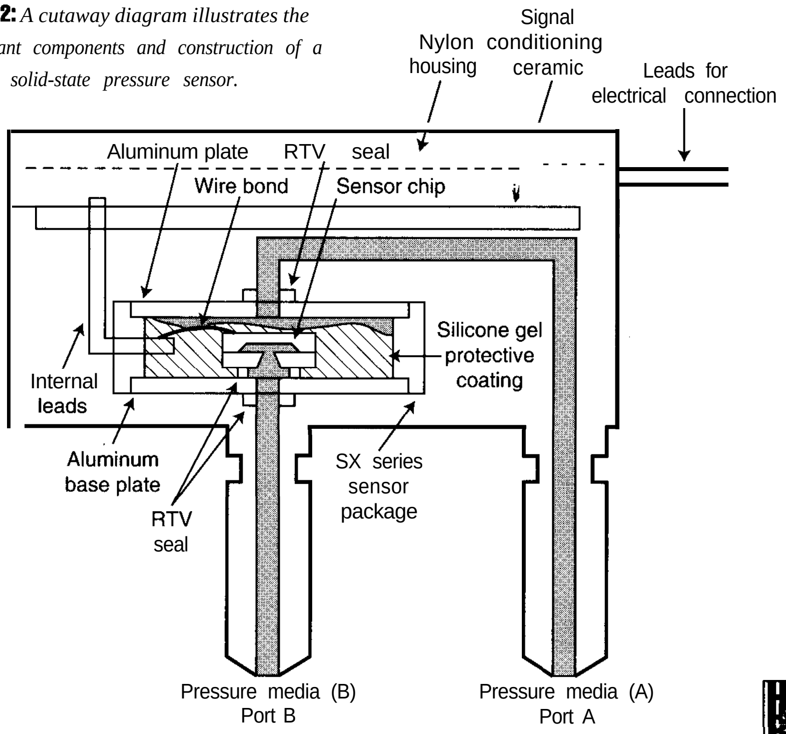
constantly moving high- and low-pressure systems that circulate around the world. These pressure systems are the primary forces driving the formation of weather patterns all over the globe.

Because the type of weather we experience is closely linked to the atmospheric pressure, a great deal of information about the current and future weather in your local area can be gleaned from atmospheric pressure measurements. In general, high atmospheric pressure indicates fair weather, while low atmospheric pressure brings unsettled weather. Fluctuating atmospheric pressure indicates that a change in the local weather is imminent.

The atmospheric pressure's effect on our weather has been studied for hundreds of years. The correlation is often difficult and inconsistent. Instruments measuring atmospheric pressure are called barometers. Hence, barometric pressure is synonymous with atmospheric pressure.

Now, you don't need an old football injury or a stiff back to predict changes in the weather. Gather a few readily available parts and warm up your soldering iron. I'm about to show you everything it takes to add a barometric pressure sensor to your HCS II home-control system.

**Figure 2:** *A cutaway diagram illustrates the important components and construction of a typical solid-state pressure sensor.*


Signal conditioning ceramic
Nylon housing
Leads for electrical connection
Aluminum plate
RTV seal
Wire bond
Sensor chip
Silicone gel protective coating
Internal leads
Aluminum base plate
RTV seal
SX series sensor package
Pressure media (B) Port B
Pressure media (A) Port A

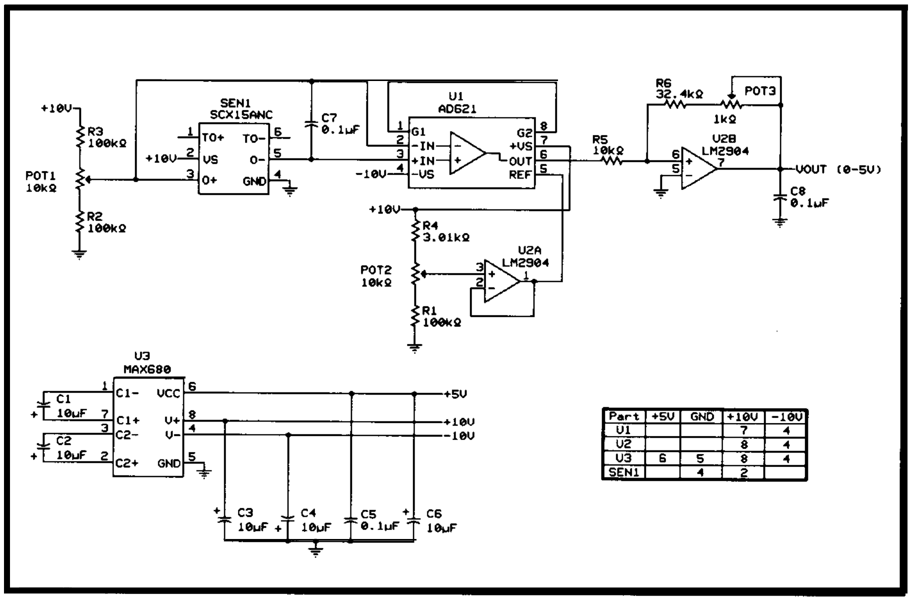
## MEASURING BAROMETRIC PRESSURE

"Barometer" is derived from the measurement units used to express atmospheric pressure. The SI unit of pressure is Pascal (Pa). One Pa is equal to one Newton per square meter (1 N/m$^2$). The bar is a related unit defined as 10$^5$ Pa. This unit inspired the term barometer (bar-o-meter).

The nominal atmospheric pressure (excluding the effect of weather systems) of 14.7 psi equals 1 .013 x 10$^5$ Pa, and is often referred to as one *atmosphere.* The most common unit, however, is inches of mercury (″ Hg). The nominal atmospheric pressure of 14.7 psi equals 29.92" Hg.

## MERCURY BAROMETER

One of the earliest means to measure the atmospheric pressure was an instrument known as the *mercury barometer.* This barometer consists of



| Part | +5V | GND | +10V | -10V |
|---|---|---|---|---|
| U1 | | | 7 | 4 |
| U2 | | | 8 | 4 |
| U3 | 6 | 5 | 8 | 4 |
| SEN1 | | 4 | 2 | |

**Figure 3:** *The voltage output of the SenSym SCX15ANC pressure sensor is amplified and scaled to generate a O-5-V output equivalent to 26.0–32.0"Hg barometric pressure.*

a long glass tube sealed at one end and filled with mercury. The glass tube is inverted and suspended with its open end immersed in a container of mercury. The height of the mercury column in the glass tube is proportional to the atmospheric pressure.

As atmospheric pressure rises, it exerts more force on the surface of the mercury in the reservoir, forcing the column of mercury in the glass tube to rise. As atmospheric pressure decreases, it exerts less force on the surface so the mercury falls.

Atmospheric pressure can be found according to the formula $P_a = \delta gh$, where $\delta$ is the density of the mercury, $g$ is the local gravitational force, and $h$ is the height of the mercury column. As Figure 1 a shows, the glass tube containing the mercury is often graduated with the desired unit of measure.

This type of barometer is most often found in the laboratory environment.

## BELLOWS BAROMETER

For home and office use, a bellows barometer is less accurate, but much more convenient. This barometer contains a set of bellows or aneroid wafers, which consist of thin metal canisters internally sealed with a known reference pressure. One end of the bellows assembly is fixed to the frame of the instrument, while the other end moves. The free end is connected via a set of gears and levers to a pointer on the meter face of the measuring instrument.

As atmospheric pressure rises above the reference pressure sealed in the bellows, the higher external pressure forces the bellows to contract, causing the attached pointer to indicate a higher relative pressure. As atmospheric pressure falls, the higher internal pressure forces the bellows to expand, causing the attached pointer to indicate a lower relative pressure. Figure 1 b pictures the meter face graduated with units of measure.

Due to this instrument's mechanical nature, a direct calculation of atmospheric pressure is extremely difficult. For this reason, the bellows barometer is usually calibrated using a mercury barometer laboratory standard.

## SOLID-STATE BAROMETER

With the advent of integrated circuit wafer technology, the solid-state barometer has become a reality. This barometer converts atmospheric pressure into an electrical impulse. In this type of sensor, a strain gauge attached to a diaphragm measures the deflection of the diaphragm in response to changing pressure. One side of the diaphragm in the solid-state barometric-pressure sensor is exposed to the atmo-

For Borland C/C++, Microsoft C/C++, Borland Pascal

**RTKernel**

**Real-Time Multitasking for DOS**

**RTKernel** is a professional, high-performance, real-time multitasking system for **MS-DOS** and **Embedded Systems. It** can use DOS device drivers and BIOS, and **runs** other DOS applications as a task - even **Windows!**

RTKernel is **loaded with features:** an unlimited number of **tasks,** excellent performance, a full set of inter-task communication functions (semaphores, mailboxes, synchronous message-passing), real and protected mode support, drivers for up to 38 COM ports and Novell's IPX services, and lots more...

It's **ROMable** and very **compact** (about **16K** code, 6K data), making it ideally suited for Embedded Systems.

RTKernel is well-documented and easy to use. All hardware drivers always come with source code; kernel source **code available** at extra charge. No **run-time royalties.**

**Join thousands of satisfied customers!**

Use **RTKernel** for:
- process control
- data acquisition
- real-time simulations
- background **processing**

**Libraries:** $495
**Source Code: add $445**

*In* North America, **please contact:**
On Time Marketing
88 Christian Avenue Setauket, New York 11733 USA
Phone (516) 689-6654 Fax (516) 689-1172
BBS (516) 689-6285 · FaxFacts (516) 689-6315 CIS 73313,3177

On Time
MARKETING
Professional Programming Tools

**From other countries, please contact:**
On Time Marketing
Karolinenstrasse 32 · 20357 Hamburg · GERMANY
Phone +49-40-437472 · Fax +49-40-435196
CompuServe 100140,633

**Free demo disk!** Request Info Kit!
**Internet:** http://www.on-time.com
ftp.on-time.com
info@on-time.com

#208

| Characteristic | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|
| Operating Pressure Range | | — | 15 | psi |
| Sensitivity | — | 6.0 | — | mV/psi |
| Full-scale Span | 89.10 | 90.0 | 90.90 | mV |
| Zero Pressure Offset | -300 | 0 | +300 | µV |
| Combined Linearity and Hysteresis | — | ±0.1 | ±0.5 | %FSO |
| Temperature Effect on Span (0-70°C) | — | ±0.2 | ±1.0 | %FSO |
| Temperature Effect on Offset (0-70°C) | — | ±100 | ±500 | µV |
| Repeatability | — | ±0.2 | f0.5 | %FSO |
| Input Impedance | — | 4.0 | — | kΩ |
| Output Impedance | — | 4.0 | — | kΩ |
| Common-mode Voltage | 5.8 | 6.0 | 6.2 | VDC |
| Response Time | — | 100 | — | µs |
| Long-term Stability of Offset and Span | — | ±0.1 | — | %FSO |

**Table** *1: Important technical specifications define the operation of the Sen Sym SCX15 solid-state pressure sensor.*

sphere, while the other side is held at a fixed reference pressure (sealed vessel).

As atmospheric pressure rises above the reference pressure sealed in the sensor, the higher external pressure contracts the diaphragm, causing the strain gauge to indicate a higher relative pressure. As atmospheric pressure falls, the higher internal pressure expands the diaphragm, causing the strain gauge to indicate a lower relative pressure. The electrical impulses generated by the solid-state barometric pressure sensor are then displayed using analog electrical measurement circuits or digital computers.

With the solid-state barometer, pressure can be expressed in a variety of units and can be easily stored for later use (see Figure lc).

Let’s take a look at the sensor that makes this barometer possible.

## SOLID-STATE PRESSURE TRANSDUCERS

Solid-state pressure sensors are fabricated using silicon-processing techniques (photolithography) familiar to the semiconductor industry. Typically, four strain-sensitive resistors (piezoresistive strain-gauge sensors) are ion implanted on a silicon wafer in a Wheatstone-bridge configuration.

After the wafer is fabricated, a diaphragm is created by chemically etching the silicon from the back side of the wafer. The diaphragm thickness controls the pressure range or sensitivity of the sensor. When pressure is applied to the diaphragm, the resistors are strained, causing an imbalance in the Wheatstone bridge proportional to the applied pressure. Figure 2 gives a detailed view of the construction of the typical solid-state sensor.

The differential output voltage of the sensor has often been, calibrated by laser trimming the bridge resistors and temperature compensated to negate thermal effects on the sensor. Specifically, the various sensors I investigated share the following properties:

- the output voltage of the sensor is proportional to the pressure applied at the input port of the sensor
- the voltage output of the bridge is ratiometric with the applied excitation supply voltage. The output voltage at a given pressure is determined using the following equation:

$$V_{out} = \frac{\frac{P}{P_{max}}}{90\,mV} \times \frac{V_{supply}}{12.0\,V} \qquad (1)$$

where $V_{out}$ is the sensor output voltage, $P$ is the applied pressure, $P_{max}$ is the sensor maximum pressure, and 90 mV is the sensor output span with a 12-V excitation. The ratio ($V_{supply}$ / 12.0 V) scales the output span for sensor excitation voltages other than 12.0 V.

- the sensor output was temperature compensated. The effect of temperature is only 0.2% of the full-scale output.

The performance specifications for a typical solid-state pressure sensor are shown in Table 1.

Finally, there are three common types of pressure measurements. The mechanical construction of the solid-state pressure determines what type of pressure measure is possible.

***Absolute pressure*** offers pressure measurements referenced to a fixed-reference pressure (generally a hard vacuum). In this case, a vacuum exists in the cavity under the sensor diaphragm. Common applications for absolute pressure sensors include barometers and aircraft altimeters.

Gauge ***pressure*** provides pressure referenced to the barometric pressure. In this case, the reference cavity of the sensor is vented to the atmosphere. This type of sensor is capable of measuring positive and negative pressures in relation to the barometric pressure. Common applications for gauge-pressure sensors include blood pressure and engine vacuum sensors.

***Differential pressure*** gives pressure measurements referenced to second-reference pressure. In this case, a second pressure port is added to the sensor to connect the reference pressure to the cavity behind the sensor diaphragm. Common applications include air-flow measurement and filter-flow measurements.

## HCS II BAROMETER

**The** barometric pressure at sea level on an average day is about 29.92” Hg (14.7 psi) and is called ***standard pressure.*** The barometric pressure can climb to over 3 1 .O” Hg during extremely fair weather and fall below 27.0” Hg during severe storm conditions. I chose a range of 26.0 to 32.0” Hg for the HCS II barometric pressure interface since it is unlikely that you’ll ever see the barometric pressure exceed this range.

To select the correct sensor for this application, I first convert this range to units of psi (sensors are rated in psi). Given that 1 .0″ Hg equals 33.865 mB (millibar) and 1.0 psi = 68.947 mB, you get:

$$\frac{26.00\text{” Hg} \times 33.865\text{ mB} \times 1.0\text{ psi}}{1.0\text{” Hg} \times 68.947\text{ mB}} = 12.7\text{ psi}$$

$$\frac{32.00\text{” Hg} \times 33.865\text{ mB} \times 1.0\text{ psi}}{1.0\text{” Hg} \times 68.947\text{ mB}} = 15.7\text{ psi}$$

I use a 15-psi sensor for my application even though the maximum expected pressure could be 15.7 psi. This small overpressure is acceptable as the sensor actually continues to operate without failure to 30 psi.

Using the calculated pressure values in psi, I determine the sensor output voltage at the extremes of the measurement range:

**Listing 1:** *A BASIC program calibrates the solid-state barometer*

```
BEGIN:
CLS: LOCATE 5, 5
PRINT"********Barometer Calibration Routine********"
PRINT: PRINT
PRINT"     (1) Enter known barometric pressure"
PRINT
PRINT"     (2) Enter known sensor voltage"
PRINT
PRINT"     (3) Quit"
PRINT: PRINT
INPUT"     Enter your selection now (1 or 2): ", ans
IF ans = 1 THEN GOTO PRESSURE:
IF ans = 2 THEN GOTO SENSOR:
IF ans = 3 THEN GOTO DONE:
GOTO  BEGIN:

PRESSURE:
CLS
INPUT"Enter the local barometric pressure (in. Hg): ", BP
MB = BP * 33.865
IN = MB / 68.947
PRINT: PRINT
PRINT"Barometric Pressure (mB): ", MB
PRINT
PRINT"Barometric Pressure (PSI): ", IN
VOUT  = ((IN / 15) * (.09)) * (10 / 12)
VOUT = VOUT * 1000
SIGOUT  = (VOUT / 1000 .0635) * 333
PRINT
PRINT"SCX15ANC Sensor output (mV): ", VOUT
PRINT
PRINT"Signal Conditioner Output (V): ", SIGOUT
PRINT: PRINT
INPUT"Enter CR to continue ", CR
GOTO  begin:

SENSOR:
CLS
INPUT"Enter the SCX15ANC Sensor Voltage (mV): ", MV
PRESSURE  = ((MV / 1000) / ((.09) * (10 / 12))) * 15
MB = PRESSURE * 68.947
IN = MB / 33.865
PRINT: PRINT
PRINT"Barometric Pressure (PSI): ", PRESSURE
PRINT
PRINT"Barometric Pressure (mB): ", MB
PRINT
PRINT"Barometric Pressure (in. Hg): ", IN
PRINT: PRINT
INPUT"Enter CR to continue ", CR
GOTO  begin:

DONE:
```

$$V_{out} = \frac{\frac{12.7\text{ psi}}{15.0\text{ psi}}}{90\text{ mV}} \times \frac{10.0\text{ V}}{12.0\text{ V}}$$

$$= 0.635\text{ V}$$

$$V_{out} = \frac{\frac{15.7\text{ psi}}{15.0\text{ psi}}}{90\text{ mV}} \times \frac{10.0\text{ V}}{12.0\text{ V}}$$

$$0.785\text{ V}$$

These values are offset and scaled so that the output of the final barometric pressure interface is 0 V at 26.0" Hg and 5 V at 32.0" Hg. This voltage range is compatible with the HCS II analog-measurement inputs.

The HCS II solid-state barometer can be placed virtually anywhere and still obtain accurate pressure readings. The barometer does not need to be placed outdoors as the barometric-pressure gradient between the inside and outside of your house is virtually zero. Barometric pressure changes slowly, except during the most severe storms. Air leaks in your home, as small as they may be, ensure that a building's internal pressure is equal to the external pressure.

If this were not true, even the smallest differences in pressure would exert enormous destructive forces (explosive and implosive) on the structure of your home. As a matter of fact, during a tornado when the barometric pressure has been known to plummet precipitously, it is advisable to open windows to prevent this phenomenon.

I'd suggest the completed barometric pressure interface be located in a cool, dry location to minimize the effect of heat and humidity on the sensor circuitry.

## CIRCUIT DESCRIPTION

The HCS II barometer interface circuit is based on the SenSym SCX15ANC solid-state pressure sensor (see Figure 3). While I had a great deal of sensor types to choose from, this unit most closely satisfied my design goals for price, performance, and accuracy. This sensor can directly measure an absolute pressure (referenced to an internal vacuum) of 0–15 psi and is also temperature compensated.

The SCX 1 5ANC sensor is not internally adjustable. However, the internal bridge can be biased externally to correct the voltage output at a given pressure by adjusting the output voltage of the sensor using VR1 The output voltage of the sensor is not ground referenced. While this is typical of the voltage output of a bridge-type circuit, it does present some unique difficulties.

To amplify the sensor output voltage, it is necessary to use an instrumentation amplifier since a single op-amp stage cannot be used. While it is possible to construct an instrumentation amplifier using three discrete op-amps and a handful of discrete components, I chose to implement a prepackaged instrumentation amplifier to reduce the total parts count and enhance the reliability of the final circuit.

U1 is an Analog Devices instrumentation op-amp configured as an inverting amplifier with a gain of 100. It provides a ground-referenced output voltage proportional to the nonground-referenced voltage output of the pressure sensor. At the lowest measurement pressure of 26.0" Hg, the output of U1 is -6.35 V (0.0635 x –100), and at the highest measurement pressure of

32.0” Hg, the output of Ul is -7.85 V (0.0785 x -100).

The instrumentation amplifier also has a $V_{ref}$ input that can directly offset the voltage output of the amplifier. This offset voltage is adjusted using VR2. Op-amp U2a buffers the offset voltage and ensures a low-impedance input to the instrumentation amplifier $V_{ref}$ pin. An offset voltage of about 6.35 V should be applied to the $V_{ref}$ pin so that the voltage output of the amplifier is 0 V at the lowest input pressure of 26.0” Hg.

The output voltage of the instrumentation amplifier should now vary between 0 V (26.0” Hg) and -1.5 V (32.0” Hg). U2b, the final op-amp, is an inverting output gain stage, which amplifies the instrumentation amplifier output by about 3.4. The final output voltage of the circuit is thus 0 V (26.0” Hg) to 5 V (32.0” Hg), controlled by adjusting VR3.

This output voltage range is directly compatible with the 0-5-V analog input range of the HCS II A/D converter. U3 is a Maxim charge-pump voltage converter. This device converts +5-V power-supply input to ±10 V for the sensor excitation and amplifier power supplies.

## CALIBRATION

The calibration procedure for the HCS II barometer interface circuit is relatively straightforward. First, obtain a reliable barometric pressure reading from a nearby source such as a local airport. Since pilots use this reading to adjust their altimeters (altitude measurement), you can be sure the barometric pressure information you receive is both current and accurate.

Using the known barometric pressure, compute the raw sensor output voltage and the overall signal-conditioner output voltage. These voltages are used to completely calibrate the signal conditioner.

A QuickBASIC program, shown in Listing 1, is a handy means to do this. Option 1 allows the user to enter a known barometric pressure and compute the voltages. A digital voltmeter is also required for this procedure.

- adjust the pressure correction pot VR 1 so that the sensor output voltage (across sensor terminals 3 and 5) is equal to the computed sensor output voltage at the known pressure
- adjust the offset adjustment pot VR2 so the voltage at pin 5 of Ul is equal to 6.35 V
- adjust the gain adjustment pot VR3 so that the overall signal conditioner voltage output is equal to the computed output voltage at the known pressure

**Listing 2:** *Sample XPRESS code displays barometric pressure data.*

```
DEFINE Ch7RawData = Variable(7)
DEFINE Ch7Volts   = Variable(8)
DEFINE Barometer  = Variable(9)

BEGIN
 IF Reset THEN
  Timer(60) = ON
END

 IF Timer(60)>=1 THEN

! Read analog input channel 7 and convert to standard
! barometric pressure units.
  Ch7RawData = ADC(7)
  Ch7Volts   = (Ch7RawData * 50) / 256
  Barometer  = (Ch7Volts * 12) + 2600

! Display the barometric pressure on the LCD-Link and in
! the V3.0 HOST message window.
  LCD(0)  = "Barometer: %P2 in". Barometer
  Console = "Barometer: %P2 in", Barometer
  Timer(60) = ON
END
```

The program also lets you enter the measured sensor output voltage directly if a calibration standard is not available. This shortcut method (no pressure correction) is actually quite good, and you can expect no more than ±2% loss of accuracy.

## SAMPLE XPRESS CODE

The example code shown in Listing 2 details the XPRESS programming language statements necessary to display the data collected using the barometric pressure interface. Every 60 s, the code reads the barometric pressure by measuring the voltage connected to the HCS II channel 7 analog input. This voltage is then converted to standard barometric pressure units.

The way this process is accomplished is somewhat unique. Because XPRESS does not support floating-point mathematical operations, all calculations are made using scaled integers. The proper values are represented correctly when they are displayed by carefully controlling the placement of the displayed decimal point. By dealing with scaled integers, we retain the most significant portion of our data otherwise lost to round-off after performing integer math operations.

In this example, raw A/D converter values (O-255) are first converted to volts. The computed voltage ranges from 0 to 50 V as a result of scaling. The computed voltage is then converted to standard barometric pressure units.

To do this correctly, multiply the computed input voltage by 12 and then add 2600. This gives an apparent barometric pressure range of 2600 (lower limit) to 3200 (upper limit), again due to scaling. These values are then displayed with two significant figures to the right of the decimal point, and thus appear as 26.00 to 32.00” Hg.

## LCD DISPLAY INTERFACE

For those of you who don’t have an HCS II system but want a portable means to measure barometric pressure, a simple solution is available. I have taken this circuit and added a single-chip A/D converter with LCD display driver and 3½-digit LCD display. The entire unit is battery powered and fits within a small hand-held plastic enclosure.

The easiest way to make this circuit work is to reconfigure the output offset and gain adjustments to achieve an output voltage of 2.60-3.20 V. This range corresponds to a barometric pressure range of 26.0 to 32.0” Hg. It is a simple matter then to display this output voltage as a barometric pressure simply by controlling the placement of the display decimal point.

For example, at a barometric pressure of 29.92” Hg, the output voltage of the barometric pressure interface circuit should be 2.992 V and would appear as 29.92” Hg.

## AND NOW FOR THE WEATHER

The HCS II barometric pressure interface provides insight into the effect of changing atmospheric pressure on our weather.

While barometric pressure is only one of many variables that contribute to the formation of our weather, it is probably the most significant. Combined with other weather variables such as temperature, wind speed and direction, as well as humidity, it is possible to predict with a high degree of accuracy the current and future weather in your area.

To accurately interpret the barometric pressure data you collect, it is necessary to correlate all of the weather data at your disposal with actual real-world conditions. Over time, as you observe the change of real-world conditions, recognizable patterns in the weather data emerge.

Once you’ve identified the patterns, you should notice the trends which enable you to predict weather conditions. Keep in mind, however, that you’re not dealing with an exact science. As with any extremely complex dynamic system, exceptions can almost always be found.

Good luck, and may the accuracy of your weather predictions exceed the nominal fair.

*John Morley is the senior electrical engineer for a small Boston-area manufacturer of custom electronic test equipment. His primary responsibility is the design of instrumentation used to measure the thermal properties of packaged semiconductors and to predict the reliability of passive electrical interconnects. He may be reached at endeavor@usa1.com.*

## CONTACTS

**Solid-state pressure sensors and pressure sensor handbook**
SenSym, Inc.
1244 Reamwood Ave.
Sunnyvale, CA 94089
(408) 744-1 500

**Instrumentation amplifier**
Analog Devices
One Technology Way
Norwood, MA 02062
(6 17) 329-4700

**Charge-pump voltage converter**
Maxim Integrated Products
120 San Gabriel Dr.
Sunnyvale, CA 94086
(408) 737-7600

## I R S

**419** Very Useful
420 Moderately Useful
421 Not Useful

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

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

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

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

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

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

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

**GETSTARTED TODAYWITHANHCS2 “123-PAK”**

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

**Assembled $651 Kit $461**

**CIRCUIT CELLAR, INC.**

4 Park Street, Vernon, CT 06066
(860) 875-2751 • Fax (860) 872-2204

#209

# An RS-232 Thermometer

DON **McLANE**

We've covered the Dallas Semiconductor **DS1620** digital thermostat chip before, but it's overkill if all you want is to read temperature. The DS1620 is a neat little **3-pin** chip that does the job nicely. Check out some tricks for interfacing it to a standard serial port.

For something so central to our comfort, temperature is rather ho-hum. Most of its mysteries were solved long ago. Nonetheless, Dallas Semiconductor has something new-the DS 1820. This 3-wire temperature sensor offers digital output. You don't need an A/D converter!

The DS 1820 is accurate to approximately 1 °F, and has a range of -67 to 257°F. In addition, it has high and low alarms, each one with a unique serial number.

It's a little tricky to do all that with only three pins, but the protocol is not too bad once you get used to it. All the action happens on one pin which is a bidirectional data bus.

First, a long pulse prepares the DS 1820 for a command. Then you send it bits: a short pulse for a 1 and a slightly longer pulse for a 0. Of course, some commands ask for data to be sent back. To read data, you send out short pulses like you used when you sent it a 1. But, since the data line is in a wired-OR arrangement, it stretches the pulse you sent when it wants to return a 0, and it does nothing to send a 1.

As long as it is above a certain minimum, it doesn't matter how long you wait between bits. The critical timing starts over with each bit, and small timing errors don't accumulate.

I immediately wanted to hook the DS 1820 to my PC since it seemed a perfect sensor for home-automation projects. I first thought I'd use a parallel port interface since I could practically solder it right to a DB-25 connector and start talking to it. But, as the old Russian proverb says, "the devil is in the details."

The DS1820 needs pulses of 15-µs, which is just enough time for a few machine instructions. It's not enough to call a system time function and wait for results. Instead, you have to craft a delay loop for a specific CPU and clock speed or talk directly to the spare timer (if it's spare on your system).

And then there are interrupts. If an interrupt occurs while you're talking to the DS 1820, it messes up the timing. Although it wouldn't happen often, the possibility makes me nervous. You could disable interrupts while you talk to the sensor, but that's a software nightmare. Besides, I only have one parallel port and my printer is hooked to it.

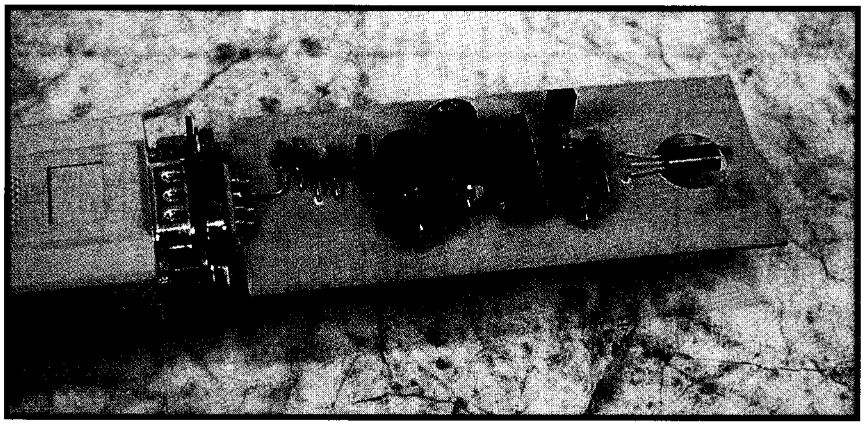


**Photo** *1: Only one connection was needed to the top row of pins on the DE-9 connector. To avoid needing a double-sided PC board, solder D1 right to the connector as shown.*

Serial ports also have problems, primarily because RS-232 uses such awkward voltages. But, if you set a serial port to the right baud rate and send the right character, you can produce a pulse that can be accurately controlled. The critical timing is done in the serial chip. Nothing can interrupt it. And, it's the same from system to system. So, even though it takes a little more hardware, I can interface the DS 1820 to a serial port. Since I have a free serial port, this solution is optimal.

Usually, I try to keep a design straightforward and avoid anything clever. But, since the details fell into place, 1 couldn't resist it. And, you're right-no serial port was intended to be used like this.

I'm doing this project in DOS so you can test and play with the circuit. Ultimately, I'll use a multitasking operating system since it makes more sense. I'll start home-automation tasks in the background and then go on to other things. Such a proposition is awkward in DOS, unreliable in Windows, but rock-solid with OS/2 and Linux. Since this is the direction I'm headed, I've designed the hardware to go with me.

**Commands for the DS1820 Digital Thermometer**

| | |
|---|---|
| 33h | Read ROM-read the unique 48-bit serial number. Follow by reading 7 bytes. |
| 55h | Match ROM-address a single DS1820 when there are multiple devices on the l-wire bus. Follow with 8-byte address. |
| CCh | Skip ROM-skip addressing. If there's only one device on the bus, you don't need to use an address. |
| F0h | Search ROM-identify devices on bus. |
| ECh | Alarm Search-search for a device where the temperature is out of limit. |
| 4Eh | Write Scratchpad-write to the locations that store the low- and high-alarm thresholds. Follow with 2 bytes of data. |
| BEh | Read Scratchpad-read memory. Follow by reading (up to) 9 bytes (see memory map). |
| 48h | Copy Scratchpad-copy thresholds (memory locations 2 and 3) to nonvolatile memory. |
| 44h | Convert Temperature-take a temperature reading. |
| B8h | Recall E2—take the values from the nonvolitile memory and put them into the alarm trigger registers (memory locations 2 and 3). |
| B4h | Read Power Supply-power-supply mode. Read one bit. |

Memory map:

| Offset | Register | Offset | Register |
|---|---|---|---|
| 0 | Temperature LSB | 5 | Reserved |
| 1 | Temperature MSB | 6 | Counts remaining |
| 2 | High alarm threshold | 7 | Counts per degree C |
| 3 | Low alarm threshold | 8 | CRC |
| 4 | Reserved | | |

In addition, there are 2 bytes of nonvolatile memory.

## THEORY OF OPERATION

RS-232 voltage levels are nominally ±12 V. In actuality, they range from ±8 to ±12 V, most often coming closer to ±8 V. So, when I talk about ±12 V here, interpret it loosely.

As you can see in Figure 1, I'm stealing power from a couple of the RS-232 status lines. If I set RTS high, I can get +12 V from it. If DTR is low, I have a -12-V source. Diodes Dl and D4 protect the circuit at the times that these signals are incorrect. Most of the load comes in pulses, and capacitors Cl and C2 smooth out the sudden changes.

The DS 1820 requires a +5-V supply, which I obtain from the +12-V supply with a 78L05 regulator. As any good temperature sensor, it draws minuscule current (to prevent self-heating).

R1 is the pull-up for the wired-OR data line. Most of the time it holds the line at 5 V. The data line can be pulled down by either Ql or the DS1820. A wired-OR is invoked when they both pull down at once.

The computer's data line (TD) is usually at -12 V, thus Q1 is off. Ql is an N-channel enhancement device and doesn't turn on until its gate goes positive by a few volts. When the computer sends data, the signal swings to +12 V, turning Ql on.

Since Ql is a CMOS device, I don't want its gate hanging open when the circuit is unplugged from the computer since a static discharge could zap the transistor. That's why I clamped the gate to the power supply through diodes D2 and D3. The data line then helps power the circuit.

U2 is an RS-232 driver, providing a CMOS version of the more common MC-1488. Since it's CMOS, it doesn't use much power, which is critical. It also pulls rail-to-rail (i.e., its output goes all the way to the power-supply voltages). Since I'm stealing power from signal lines, my output needs to be about the same as my power. I've already lost a diode drop from Dl and D4. I don't want to give up any more voltage.

U2 actually has four drivers in it, although I only use one. While throwing three drivers away seems like a waste, it was the best part I could find for the job. Since CMOS gates only draw current when they're switching, the unused drivers draw insignificant current. I need to ensure the unused inputs are grounded so the gates don't oscillate and draw substantial current. (My first prototype demonstrated this, but don't tell anybody I did something so foolish.)

## CONSTRUCTION

I chose to use a regular DE-9 connector and tit the circuit board

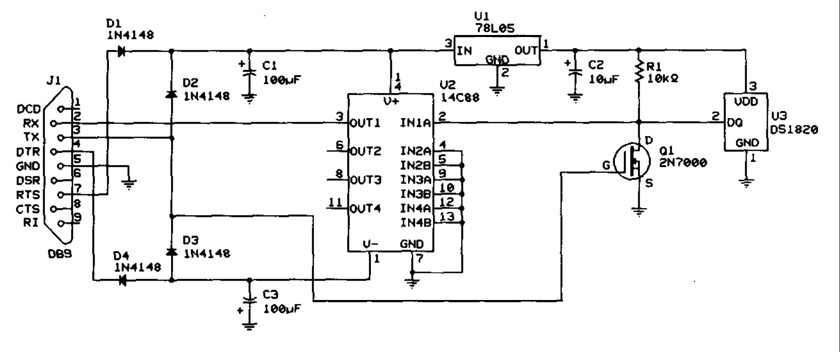


**Figure 1:** *Interfacing to the awkward voltages of the RS-232 interface is difficult. Thanks to low-power components, this circuit is powered directly from the signal voltages.*

**Listing 1:** Notice *the striking simplicity C++ brings this* `main()` **function.** *You just declare an instance of the class (i.e.,* `tempsor`*) and then use it. All the details are "under the hood" of the class.*

```
class DS1820 {
  char save_IER, save_MCR, save_LCR, save-speed;
  void DScommand(const uchar *to_sensor, uchar *from_sensor);
  void DSreset(void);
  DSsend(const uchar *to_sensor, uchar *from_sensor);
  double tempcalc(uchar *buff);
  cksumOK(uchar *buff);
public:
  DS1820(void);          // constructor
  ~DS1820(void);         // destructor
  double Ctemp(void);    // return Centigrade temp
  double Ftemp(void);    // return Fahrenheit temp
};

... ugly code deleted

void main0

  DS1820 tempsor;
  printf("Hit a key to exit\n");
  while (!kbhit())
    printf("temp = %5.1f \r", tempsor.Ftemp());
  getch();               // swallow char
}
```

between the two rows of pins. A right-angle connector might have been stronger mechanically, but I found the symmetry more attractive. I also wanted to use a single-sided PC board, but I needed one connection to the top row of pins. So, one end of D1 is soldered directly to one of the top pins (see Photo 1).

The bottom row of pins on the DE-9 connector is soldered directly to pads on the PC board. To fasten the top side, glue a couple of pins to the PC board (epoxy or the thick cyano-acrylic glue works well).

## SOFTWARE

I've written a program to read and display temperature (see Listing 1). I've included a compiled binary, which executes with **RUN_1820.** It's compiled for COM2. However, if you're using a different COM port, a comment in the source code tells you how to change the source code. Just recompile and run.

The source is written in C++ and compiled with Borland's Turbo C++. Initially, I wrote it in C, but I found myself saying things like "this stuff should be put in a constructor" and "these variables should be private to a couple functions."

Finally I gave in, renamed the source to **.CPP** and made a quick conversion. I'm glad I did since the resulting code is cleaner. While I've always appreciated C++ in large systems, I never expected to like it for low-level stuff like this. But, C++ seems well suited. I guess I'm coming around.

DOS doesn't provide any operating system calls to set the serial port to 115.2 kbps, so I need to program the 8250 serial chip directly. Here's how you talk to the sensor.

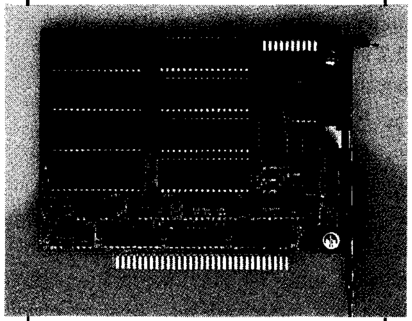
You first send it a reset pulse lasting 480–960 µs. To do this, I set the serial chip to 9600 bps and sent a zero. An RS-232 data line idles in the 1 state when no characters are being sent. A character always begins with a start bit at the 0 state. The RS-232 drivers invert this, so the 1 state is -12 V and the 0 state is +12 V.

We have a start bit and eight 0 bits, which adds up to one long pulse. At 9600 bps, it gives us a pulse of 938 µs. The pulse is echoed to the computer, so I receive the 0 that I sent. Just read the character to clear the register in the serial chip. You don't need to do anything with the received data.

The DS 1820 responds with a presence pulse lasting 60–240 µs.

VMAX®

QUALITY PRODUCTS
RESPONSIBLE SERVICE
RELIABLE DELIVERY

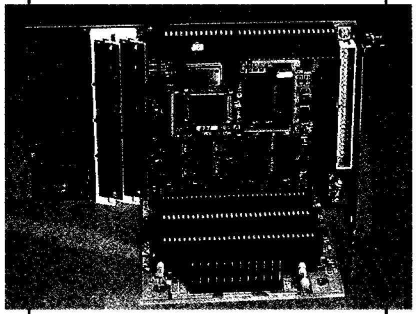


**ENHANCED SOLID STATE DRIVE — $164***

4M Total, Either Drive Bootable
½ Card 2 Disk Emulator
Flash System Software Included
FLASH & SRAM, Customs too

**486 SLAVE PC — $895"**

Add up to 4 Boards to One Host PC
Fast Data Transfer and I/O
PC-I 04 Port, IDE & Floppy Control
Independent Processors on One Bus
No Special Compilers Needed



**TURBO XT w/FLASH DISK — $266"**

To 2 FLASH Drives, 1 M Total
DRAM to 2M
Pgm/Erase FLASH On-Board
CMOS Surface Mount, 4.2"x6.7"
2 Ser/1 Par, Watchdog Timer

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

***TEMPUSTECH, INC.***
***TEL: (800) 634-0701***
***FAX: (941) 643-4981***

Fax for Fast response!

295 Airport Road
Naples, FL 33942

#210

Read the serial chip again to clear the character. It won't be a 0 because it's too short, but it doesn't matter what it is.

Send the DS 1820 commands a bit at a time. To send a 0 bit, I need to send a pulse lasting 60–120 µs. If the serial chip is set at 115.2 kbps, a zero byte (again, a start bit and 8 data bits, all low) gives a pulse of 78 µs.

To send a 1 bit, I send a shorter pulse lasting 1–15 µs. If I send a character with all bits high at 115.2 kbps (i.e., FFh), then I only send a start bit of about 9 µs. The commands are sent a bit at a time, least significant bit first. After each bit, I read the serial chip to clear the receive register (I still don't care about what's read). Typically, I then send the command CCh and then 44h.

You could have more than one DS 1820 connected in parallel. In our case, there's only one and the CCh command tells the chip to skip the addressing protocol. The 44h command tells the chip to begin a temperature conversion.

Within 2 s, the DS 1820 completes a temperature reading. While this is not fast, temperature doesn't change quickly. After reading the temperature, start with a reset pulse and presence pulse sequence as before. Then send a CCh to skip addressing and a BEh to read the temperature.

To read back values, I send short pulses as though I were sending it a 1. However, I need to look at the character when I read the serial chip. If the DS 1820 wants to send back a 0 bit, it stretches out the pulse to at least 15 µs, and if it wants to send a 1 bit, it does nothing. So, if I look at the first bit of the received character, I can get the bit that the sensor is sending. It's tricky, but the timing works out elegantly.

For RS-232, 115.2 kbps is pretty fast. The maximum length of cable you can use is limited to 20'. If you want to push this limit, look for a low-capacitance cable.

## CONCLUSION

Three things have been accomplished:

1) you have a circuit you can use or modify
2) I've demonstrated a method for interfacing a serial port to a one-wire bus. Serial ports are available on most PCs
3) I've included software which provides a starting place to develop your own applications. In particular, the C++ class may be a helpful example.

Clearly, this project is only a beginning. The rest is up to you.

*Don McLane is an engineer with Syscon International, where he designs real-time data acquisition and control systems for industrial applications. He may be reached at dmclane @delphi.com.*

**SOFTWARE**

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

**IRS**

422 Very **Useful**
423 Moderately Useful
424 Not **Useful**

# ENGINEERING PROFESSORS

**LISTEN UP!**

The reason you read *Circuit Cellar INK* is obvious.
Like thousands of others, you have come to realize that *Circuit Cellar INK* presents technically relevant computer-based applications in a comprehensive and instructive manner. In fact, our projects are presented just the way you'd do them if only you had the time.

Since 1994, we have provided thousands of college students with industry-quality development tools and information. This program is **FREE** and because of the continued success we have experienced, we'd like to extend our offer to the 1995-1996 school year. The fall semester is upon us...

If you're an engineering professor or teach qualified technical students, we'd like to give you and all of your students free subscriptions to *Circuit Cellar INK*. Please tell us how we can contact you directly so you don't miss a single issue.

**WE ARE HELPING AMERICA REGAIN THE COMPETITIVE EDGE. WE'RE HOPING YOU WILL TOO!**

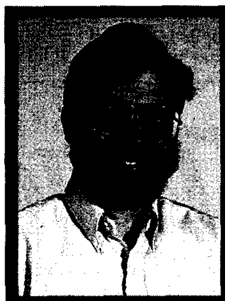
Rose Mansella, Circulation Coordinator
Circuit Cellar INK
4 Park St.
Vernon, CT 06066

Tel: (860) 875-2199
Fax: (860) 872-2204
Internet: rose.mansella@circellar.com

FIRMWARE FURNACE

Ed Nisley

# Journey to the Protected Land: Looking at the Virtual-86 Monitor



This month, Ed builds a simple V86 monitor. After going over what it takes to keep the monitor running smoothly, he shows what to do when it stops.

As far as Karen, our 2½ year old, is concerned, the world is simple. Food appears, we build interesting things, and diapers vanish as if by magic. Her Phillips driver fits every screw in her practice board. Best of all, when she gets sleepy, it's always time for a nap.

From my viewpoint, things are a smidge more complex. Karen grates cheese and measures yeast, while I knead dough and make sure the 'za hits the table by the time Mary gets home. I try to fix the wiper motor on my car while Karen "sorts" all the bolts. And, yes, I've changed enough diapers for a lifetime, thank you very much.

A Virtual-86 task thinks a lot like Karen. It *thinks* it has full control of a 16-bit CPU, a megabyte of memory, and the system's I/O ports. It doesn't know that it can't touch anything without the permission and intervention of a V86 monitor program. Perhaps, one of these days, V86 code will grow up into 32-bit code?

As you saw last month, V86 code resembles plain, old, dull, real-mode programming. The interesting stuff goes on behind the scenes in 32-bit land. This month, we'll build a simple V86 monitor, discover what keeps a V86 task running smoothly, and find out what happens when it stops.

Listing 1—This *version of* ***the FFTS task dispatcher*** installs the ***V86 GPF*** error *handler before entering a* ***V86*** task *and removes if on* ***the way out.*** *Because* ***V86 tasks cannot*** *return* ***directly to the task*** *dispatcher,* ***the V86*** *monitor* ***must call*** *this routine.*

```
   PROC     TaskGtDispatch FAR
   USES     EAX,EBX,ECX,EDX,ESI,DS,ES,FS

   MOV      EDX,SYNC_ADDR       ; mark the start in the old task
   IN       AL,DX
   OR       AL,80h
   OUT      DX,AL

   set kernel's seg regs so we can reach our variables
   MOV      EAX,GDT_DATA
   MOV      DS,AX
   MOV      EAX,GDT_CONST
   MOV      FS,AX

   CALL     UtilToggleDog       ; flip the watchdog bit
   CALL     DbgGetSwitches      ; sample the switches for tracing

   remove special handlers if we are leaving a V86 task
     the task switch is in 32-bit PM code, so we can't test
     EF_VM directly
   MOV      EBX,[DispIndex]     ; get current task index
   TEST     [(DispArray.TaskInfo) + EBX*SIZE DISP_ENTRY], \
            MASK TaskV86
   JZ       @@LeavePM

   CallSys  CGT_V86_REMOVEGATES
@@LeavePM:

;--- select and dispatch the next task
;     ...simple round-robin run through all the tasks in the array
 ;  . . .we dispatch the kernel task (index = 0) every time
@@DispSkip:
   INC      EBX                 ; step to      next task in array
   MOVZX    EAX,[(DispArray.TaskInfo) + EBX*SIZE DISP_ENTRY]
   TEST     EAX,MASK TaskPresent      ; entry in table?
   JNZ      @@DispGo            ; 1 = yes, so do it
   XOR      EBX,EBX             ; 0 = no, hit the end
   JMP      @@DispForce         ; always run kernel!

@@DispGo:
   TEST     EAX,MASK TaskDispatchable; OK to run this task?
   JZ       @@DispSkip          ; 0 = no, skip it

@@DispForce:
   MOV      [DispIndex],EBX     ; save new task index

;--- if new task is in V86 mode, install special handlers
     the task switch is in 32-bit PM code, so we can't use EF_VM
     en passant, aim the string pointer at the appropriate message
   MOV      ESI,OFFSET cMsg_PM; assume PM task
   TEST     [(DispArray.TaskInfo) + EBX*SIZE DISP_ENTRY], \
            MASK TaskV86
   JZ       @@EnterTask

   MOV      ESI,OFFSET cMsg_V86       ; it's a V86 task
   CallSys  CGT_V86_INSTALLGATES

@@EnterTask:

<<< tracing code omitted >>>

 --- do the task switch
   MOV      EBX,[DispIndex]     ; aim at new task pointer in table
   JMP      [FWORD PTR ((DispArray.TaskPtr) + \
             EBX*SIZE DISP_ENTRY)]
```

*(continued)*

Karen recently noticed that the world doesn't stop during her nap. She might be distressed to learn that her monitor takes a nap sometimes, too!

## A MINOR EXCEPTION

In the absence of external interrupts, V86 code runs until it attempts to execute a privileged instruction. The CPU treats this as an error, immediately exits V86 mode, and passes control to a 32-bit, protected-mode error handler. Each possible error corresponds to an entry in the IDT, as shown in *INK 50's* Figure 3. You must provide a handler for every error, lest the CPU detect a missing handler and shut itself down.

Although each error must have a handler, you may use the same handler for several different error conditions. So far, FFTS uses only four error handlers: one each for stack, double-fault, and invalid TSS errors, and another for all other errors. Each handler displays an appropriate error message and halts the system. The FFTS kernel code and all the 32-bit taskettes live in an error-free zone-at least in principle.

V86 mode introduces the notion of a deliberate protection exception that should not terminate the task. The error handler must determine why the error occurred and respond appropriately. If the V86 task begins executing invalid instructions, for example, the monitor should kill it. On the other hand, it may just be asking for help in completing its job.

Trying to read Karen's mind can be rather difficult; figuring out what a V86 task tried to do is easier. Before invoking the error handler, the CPU puts the address of the failing instruction on the stack along with the task's segment registers. The V86 monitor can recover the instruction by treating it as data and decide whether the error was, in some sense, expected.

If you intend to build a complete DOS box, you must simulate the PC's real-mode response to every possible exception caused by every possible DOS program. Our job is simpler by far because we control the V86 tasks and can define what should occur for each error whether deliberate or accidental.

The simple V86 task shown in last month's Listing 3 (*INK* 62) executes an INT 20 instruction on each iteration of its endless loop. I picked that instruction for historic reasons: INT 20, prior to DOS 2.0, was one way for programs to terminate themselves. Subsequent DOS versions provided better methods. INT 20 was labeled "obsolete" in the documentation.

Our V86 task won't actually terminate, though. I've defined INT 20 as a gateway into the FFTS kernel rather than a catastrophic error. Just as INT 21 provides access to (nearly) all the myriad DOS services, INT 20 grants well-controlled access to FFTS kernel services.

There are several "standard" V86-to-PM interfaces available today, most notably DPMI [DOS Protected-Mode Interface) and various DOS extenders. While those interfaces are relevant if you're writing code for a system that uses them, I'd rather keep our attention focused on how the CPU works rather than the intricacies of a particular programming standard. After you know how the hardware behaves, you will be better able to use someone else's code.

For our present purposes, in fact, the V86 monitor must handle only one situation: an INT 0D GPF caused by an INT 20 instruction in a V86 task. When the monitor recognizes that event, it performs a task switch using the familiar FFTS dispatcher call gate. After all the other taskettes have executed, the dispatcher switches back to the monitor, which returns control to the V86 code just after the INT 20 instruction.

That sounds easy enough, right?

## THE END IS COMING!

Because the V86 code runs until it causes a protection exception, you must install the monitor's error handlers before starting the code. It is possible, of course, to mix the V86 monitor and PM error handlers in a single routine. I decided on the separate approach to simplify this explanation. If you don't need V86 mode, just rip out all the offending pieces!

The other taskettes, being entirely 32-bit PM code, won't call INT 20.

Listing l-continued

```
 IN      AL,DX              ; mark the end in the new task
 AND     AL,NOT 80h
 OUT     DX,AL

 INC     [SwitchCtr]        ; count this task switch

--- return to the new task
    the FAR PROC forces this to be a FAR RET...
 RET
 ENDP    TaskGtDispatch
```

They may, however, cause a GPF for any of a distressingly large number of reasons, so the FFTS setup code can't simply aim INT OD at the V86 monitor and be done with it.

Listing 1's FFTS task dispatcher passes control to each task in turn. Before switching to a V86 task, it loads an interrupt gate aimed at the V86 monitor into the INT OD IDT entry. It restores the original interrupt gate after the V86 monitor returns to the dispatcher.

Next month, the V86 monitor sprouts more features and we'll use a slightly different technique. As motivation, consider what happens if an error in the V86 monitor catapults the CPU into the V86 monitor's GPF handler. Talk about shooting yourself in the foot, toe by toe. Protected mode is no match for deliberate stupidity!

The dispatcher tests the TaskV86 bit in the task information array to find out if the task uses V86 mode. Remember that the CPU is in 32-bit

Listing 2—*When* the *FFTS task dispatcher* encounters a *V86 task*, if *calls* this routine *to install the* monitor's GPF handler in the *INT OD IDT entry. A* similar routine restores *the PM error* handler when *the V86* monitor returns fo *the* dispatcher. The *two* routines share a *flag* fhaf indicates which *vector is* in *the IDT.*

```
 PROC    V86GtInstallGates FAR
 USES    EAX,EDX,DS,FS

 MOV     EAX,GDT_DATA
 MOV     DS,EAX
 MOV     EAX,GDT_CONST
 MOV     FS,EAX

 CMP     [HandlersIn],0     ; nonzero = installed
 JE      @@Install          ; zero = not installed, OK
 CALL    ConfSendString,CON_SERIAL, \
         GDT_CONST,OFFSET cMsg_DupeIn
 JMP     @Done

@Install:
 CallSys CGT_MEM_GETINTGATE,ODh,GDT_IDT_ALIAS
 MOV     COWORD PTR OldIntOD],EAX
 MOV     [(DWORD PTR OldIntOD) + 4],EDX

 MOV     AX, CS             ; fetch our own CS
 MOVZX   EAX,AX
 CallSys CGT_MEM_SETINTGATE.ODh,GDT_IDT_ALIAS, \
         EAX,<OFFSET V86IntODHandler>,ACC_INTGATE

 MOV     [HandlersIn],1     ; mark installed
@Done:
 RET
 ENDP    V86GtInstallGates
```

**Listing** *3-This simple V86 monitor* examines *the* instruction *at the V86 task's CS:IP. If if finds an INT 20,* it calls the *FFTS* task dispatcher to *pass control* to the next task. Any other instruction *triggers* an *error message.*

```
  PROC      V86IntODHandler FAR

;--- get pointer to Ring-O stack and set up addressing
  PUSHA                      ; save all bystanders
  MOV       EBP,ESP          ; get ptr to CPU-pushed values

  STR       AX               ; we are running in the V86 TSS
  MOVZX     ESI,AX

  MOV       EAX,GDT_DATA
  MOV       DS,EAX
  MOV       EAX,GDT_CONST
  MOV       FS,EAX

;--- fetch dword starting at faulting opcode
     the opcode appears in AL, conveniently enough
  CallSys CGT_MEM_PEEKREAL,               \
          <[(V86STACK PTR EBP).OldCS]>,          \
          <[(V86STACK PTR EBP).OldEIP]>

;--- Int 20 prompts a task switch away from the V86 task...
  CMP       AX,020CDh        ; CD 20 = Int 20
  JNE       @@BadIntOD       ; anything else is trouble!

  STI                        ; allow interrupts during switch
  CallSys LDT_SWITCH         ; switch to next task
  CLI                        ; turn interrupts off again
```

(continued)

protected mode while running the FFTS dispatcher, thus the EF_VM bit in EFLAGS is always zero throughout the code in Listing 1. I must confess to goofing this the first time around.. . . Testing EF_VM was just so easy!

Listing 2 shows how FFTS installs the INT0D interrupt gate. A similar routine, not listed here, restores the PM gate. A global variable (eeek!) indicates which gate is currently in the IDT, a simple technique that won't withstand more rigorous use. Bear in mind that FFTS includes only a single V86 task at the moment, so it can't be dispatched twice in a row, nor can another task preempt the system while it's executing.

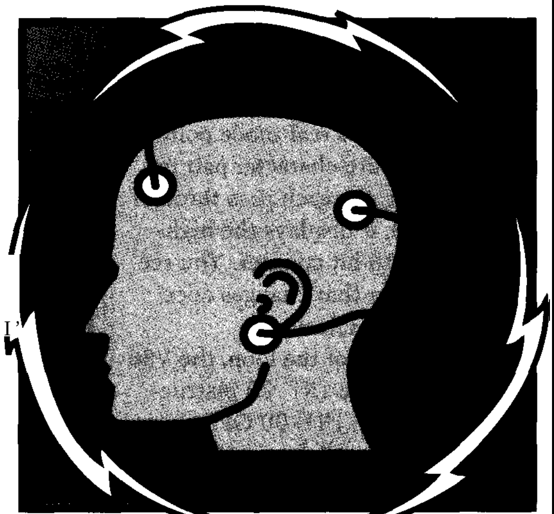
Having prepared for the end, we can now start at the beginning.

## SWITCHING IN, PUSHING OUT

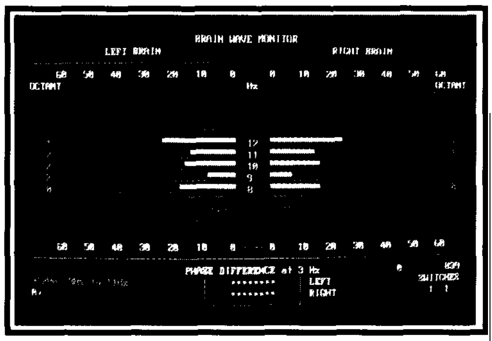
Figure 1 is a cleaned-up version of a sketch I made while writing this code. It charts the flow of control through three tasks and four routines, shows what triggers each transition, and indicates the relevant listings and

# HAL-4

**EEG Biofeedback Brainwave Analyzer**



The HAL-4 kit is a complete battery-operated 4-channel electroencephalograph (EEG) which measures a mere 6" x 7". HAL is sensitive enough to even distinguish different conscious states-between concentrated mental activity and pleasant daydreaming. HAL gathers all relevent alpha, beta, and theta brainwave signals within the range of 4-20 Hz and presents it in a serial digitized format that can be easily recorded or analyzed. HAL's operation is straightforward. It samples four channels of analog brainwave data 64 times per second and transmits this digitized data serially to a PC at 4800 bps. There, using a Fast Fourier Transform to determine frequncy, amplitude, and phase components, the results are graphically displayed in real time for each side of the brain.



HAL-4 KIT...... NEW PACKAGE PRICE - $279 **+SHIPPING**
**Contains HAL-4 PCB and all circuit components, source code on PC diskette, serial connection cable, and four extra sets of disposable electrodes.**

to order the HAL-4 Kit or to receive a catalog,
**CALL:** (860) **875-2751 OR FAX:** (860) 872-2204
**CIRCUIT CELLAR KITS** • 4 PARK STREET
SUITE 12 • VERNON • CT 06066

• The Circuit Cellar Hemispheric Activation Level detector is presented as an engineering example of the design techniques used in acquiring brainwave signals. This Hemispheric Activation Level detector is not a medically approved device, no medical claims are made for this device, and it should not be used for medical diagnostic purposes. Furthermore. safe use requires HAL be battery operated only!

figures. Protected mode may not be simple, but it's a whole lot easier than debugging a real-mode crash!

By definition, every V86 task runs in Ring 3 with the lowest possible privilege: user level. This is the first task we've encountered that doesn't run in Ring 0 with kernel privilege, so several new rules apply. Fortunately, FFTS is a sufficiently attenuated operating system that we won't get lost in the forest.

Recall that a task switch saves the current CPU state in one TSS and loads the new state from another TSS. By definition, the two tasks are completely separate; they need not share any registers, stack areas, data, or code. You can (and FFTS does) relax that isolation on occasion, but the CPU provides enough weather-stripping to seal the cracks if your system demands airtight security.

The CPU examines the EF_VM bit in the TSS EFLAGS field before loading the registers. When that bit is set, as it must be to start running in V86 mode, the CPU treats the CS:EIP fields as a real-mode address rather than a protected-mode segment selector and offset. The CPU begins executing the first instruction in the V86 taskette in 16-bit V86 mode.

The 16-bit code shown in Listing 3 last month creates a real-mode pointer to the last attribute-character pair in the video buffer. On each pass through its endless loop, it displays the high-order byte of a 16-bit counter. You see a single character that changes once for each 256 loop iterations.

At the bottom of the loop, the V86 code encounters the INT 20 instruction, triggering the INT 0D GPF we've spent so much time preparing for.

The INT 0D interrupt gate defines the code segment and offset of the V86 monitor's error handler. The code segment selector specifies kernel privilege: Ring 0. Remember this isn't a task gate and the CPU doesn't perform a task switch into the V86 monitor.

However, a user program should not have direct access to kernel resources. If the monitor used the V86 code's stack, it would be easy enough to "peek" into the stack after the monitor returns control. User-level code

Listing 3-continued

```
    ADD      [(V86STACK PTR EBP).OldEIP],2     ; step over Int 20

    POPA                          ; restore bystanders
    ADD      ESP,4                ; step over Ring-0 error code
    IRET                          ; return to V86 task

;---the Int OD was caused by something we can't cope with: give up!
@@BadIntOD:
    <<< error display code omitted >>>

    CallSys  CGT_TASK_NODISP,ESI        ; make nondispatchable
    CallSys  LDT_SWITCH      ; we will never return!

    ENDP     V86IntODHandler
```

could then extract interesting kernel-level local variables, addresses, and suchlike. The 80x86 CPU, CISC to the max, won't let this occur.

The TSS structure that is shown in Listing 4 of *INK 54* includes three stack pointer fields: StackPtr0-StackPtr2. Whenever the CPU passes through a gate specifying a higher protection level, it automatically loads SS:ESP from the corresponding TSS field. In our case, StackPtr0 holds the location of the Ring-O stack for the V86 monitor.

StackPtr1 and StackPtr2 define stacks for code running in rings 1 and 2, which we don't use in FFTS.

More complex systems may have several layers of code at various protection levels, but FFTS is simple enough that Ring 0 and Ring 3 suffice. The Intel doc suggests running such things as user-installed device drivers in the intermediate levels to protect the kernel from their misadventures.

Conspicuous by its absence is a StackPtr3 field. Kernel code with unrestricted access to all hardware, instructions, and features cannot run at user privilege. Therefore, the CPU need not change stacks when passing from user code in Ring 3 to other, presumably user, code at the same protection level. Think about it....

Listing 4—*When the CPU switches to the Ring-0 stack defined in the V86 task's TSS, if automatically pushes the CPU segment registers and error code. The V86 monitor uses a PUSHA instruction to save the remaining registers. This structure defines the ensuing stack layout.*

```
            STRUC  V86STACK
OldEDI      DD     ?              ; last of PUSHA regs
OldESI      DD     ?
OldEBP      DD     ?
OldESP2     DD     ?              ; points to ErrCode!
OldEBX      DD     ?
OldEDX      DD     ?
OldECX      DD     ?
OldEAX      DD     ?              ; first of PUSHA regs
ErrCode     DD     ?
OldEIP      DD     ?
OldCS       DD     ?
OldEFLAGS   DD     ?              ; should have VM & RF set
OldESP      DD     ?
OldSS       DD     ?
OldES       DD     ?
OldDS       DD     ?
OldFS       DD     ?
OldGS       DD     ?
            ENDS   V86STACK
```

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

After loading SS:ESP from the TSS, the CPU pushes an assortment of registers on the new stack, including the previous address-bit contents of SS:ESP, then pushes an always-zero error code. At this point, the V86 monitor in Listing 3 gets control. A PUSHA instruction stashes the remaining CPU registers on the stack. Setting EBP equal to the current ESP aims it at the start of the stacked values. The V86 STACK structure that is shown in this month's Listing 4 maps the stack layout.

There is an interesting subtlety here. In V86 mode, the segment registers hold address bits that [most likely) do not correspond to valid GDT or LDT descriptors. After saving the segment registers on the Ring-O stack, the CPU automatically clears them so that only CS and SS are nonzero when the V86 monitor gets control.

This solves a problem that FFTS doesn't encounter as it uses separate error handlers for V86 mode and PM tasks. A combined handler must save and restore the segment registers, but restoring a V86-mode segment address in protected mode causes an error. By zeroing the registers and restoring them from the Ring-O stack as part of the I RET sequence, the CPU enables code to work correctly with either type of register on the stack.

We are now safely in the V86 monitor, where we must find out why we got here.

## PEEKING INTO CODE

The CPU examines each instruction and triggers an exception if it isn't valid in V86 mode. Therefore, the I NT 2 0 instruction causes a protection exception *before* the CPU executes it. The V86 code's 16-bit CS:EIP address on the stack points to the I NT 20, not the next instruction, as you might expect from your real-mode experience.

Examining the instruction is easy enough.The CGT_MEM_PEEKREAL kernel function returns the double word at a real-mode seg:offset address. If that function returns CD 20, byte-reversed in Ax, the monitor knows this "error" was deliberate. Any other bytes indicate something completely unexpected went wrong.

The error analysis code can obviously become more complex. A DOS box would detect I NT 2 1 instructions, extract function parameters from the

***C COMPILERS***
***CROSS ASSEMBLERS***
***DEBUGGERS***

| | |
|---|---|
| 68HC08 | 8051/52 |
| 6809 | 8080/8085 |
| 68HC11 | 80861186 |
| 68HC16 | 80961196 |

Low **Cost!!** PC based cross development packages which include EVERYTHING you need to develop C and assembly language software for your choice of CPU.
- MICRO-C compiler, optimizer, and related utilities.
- Cross Assembler and related utilities.
- Hand coded (efficient ASM) standard library (source included).
- Resident monitor/debugger ( source included)*
- Includes Integrated Development Environment and command line tools (incl: editor, telecomm and many other utilities).

* 68HC08 and 68HC16 kits do not include monitor/debugger.

***Each Kit: $99.95' US + s&h (please specify CPU)***
***Super DK (supports all 8 CPU families) $300.00 US***

***Call or write for our free catalogue of development tools.***
*FREE Demos available on BBS (or send $5 for diskette)*
***We accept Cheque/MO/Purchase Orders/VISA***

DDS ***Dunfield Development Systems***
**P.O.** Box ***31044*** **Nepean,** Ont. **K2B 8S8** CANADA
Tel: 613-256-5820 Fax: 256-5821 BBS: 256-6289
**email:76162.3355@compuserve.com**

DESIGN CIARCIA WORKS

oes your Big-Company marketing department come up wit more ideas than the engineering department can cope with? Are you a small company that can't afford a full-time engineering staff for once-in-a-while designs?

Steve Ciarcia and the Ciarcia Design Works staff may have the solution. We have a team of accomplished programmers and engineers ready to design products or solve tricky engineering problems.

Whether you need an on-line solution for a unique problem, a product for a startup venture, or just experienced Insulting, the Ciarcia Design Works is ready to work with yo Just fax me your problem and we'll be in touch.

REMEMBER... A
CIARCIA DESIGN WORKS
FAX: (860)871-8986

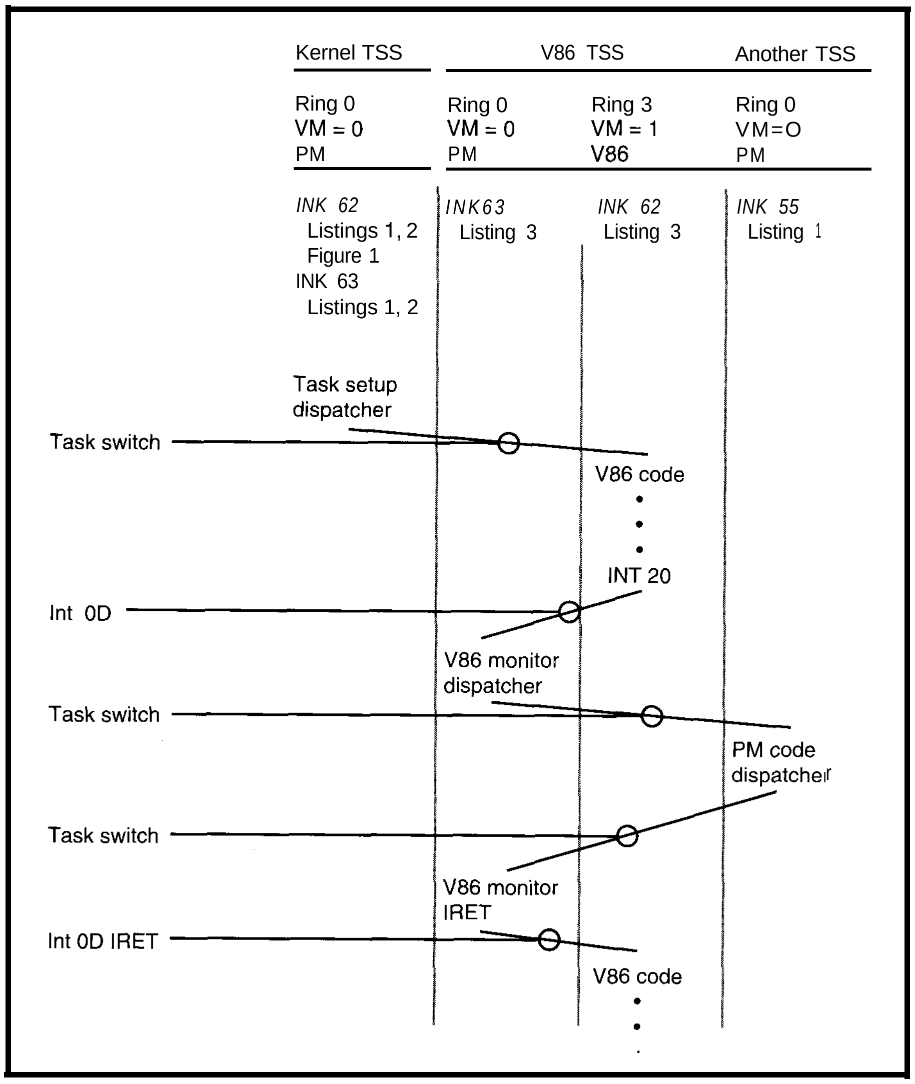


Figure l--This *roadmap traces the CPU through the task switches, interrupt gates, and IRETs used in this month's code. The V86 task and V86 monitor share the same TSS, with the VM bit in EFLAGS* defining their modes. *When the V86 task causes a protection exception, the CPU stores EFLAGS on the stack, clears the VM bit, and begins executing the V86 monitor in 32-bit protected* mode, *The INT 20 instruction in the V86 code causes a General Protection Exception vectored through INT 0D.*

stacked registers, and branch directly into its DOS emulator routines. We'll do some of that later on, but for now we need only detect a single instruction without decoding any register contents.

When the monitor finds an INT 20 instruction, it enables interrupts and invokes the FFTS dispatcher using the call gate in LDT[0]. Remember that we're in 32-bit protected mode now, the GDT and LDT are entirely usable, and everything works the way it has in previous columns. The dispatcher reinstalls the PM error handler's gate before switching to the next 32-bit task, as shown in Listing 1.

After again installing the V86 monitor's interrupt gate in the IDT, the dispatcher returns control to this task. The monitor adds two to the V86 task's stacked EIP register, pointing it at the instruction immediately following the INT 20. A POPA recovers the registers saved by PUSHA, adding 4 to ESP discards the stacked error code, and finally, IRET returns to the V86 task.

This being a CISC machine, the IRET instruction is just as complex as the initial entry. The CPU pops EIP, CS, and EFLAGS from the Ring-O stack. The VM bit in EFLAGS tells the CPU to restore the remaining registers from the stack, load the segment registers without the usual PM descriptor checks, and resume the V86 task at the new CS:EIP. The transition from 32-bit PM to V86 mode occurs ***en** passant* with no special attention.

The 16-bit code remains blissfully unaware that it gave control to a protected-mode task switcher. As far as it's concerned, the INT 20 didn't do anything other than delay for a while. That's the key to V86 mode and V86 monitors: everything is simply a matter of software!

The V86 code makes another pass around its endless loop, twiddles the parallel port bit, writes CH into the video display, and smacks into the INT 20. That starts the whole cycle over again.

In the unlikely event that the monitor discovers something other than an INT 20 at the task's CS:EIP, it calls the CGT_TASK_NODISP kernel routine that marks the V86 task as nondispatchable and then returns to the dispatcher. The next time around the task list, the dispatcher skips over the V86 task. As far as Listing 3 is concerned, the dispatcher never returns from that final call.

That certainly can't happen here, of course.. . .

## RELEASE NOTES

The V86 monitor I described here appeared on the BBS last month simply because you can't have a V86 task without a monitor. Everything should make more sense when you look at the figures and explanations in both columns.. .at least, I hope so!

Next month, the V86 task sprouts a hardware-interrupt handler. The new V86 monitor must handle both software and hardware interrupts, which turns out to be trickier than it first seems.

***Ed Nisley (KE4ZNU), as Nisley Micro Engineering, makes small computers do amazing things. He's also a member of Circuit Cellar INK's engineering staff. You may reach him at** ed.nisley@circellar.com **or 74065.** 1363@compuserve.com.*

## IRS

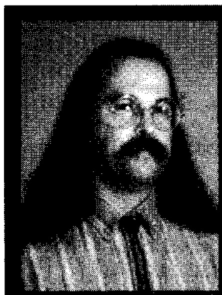
425 Very Useful
426 Moderately Useful
427 Not Useful

# Creating the Smart-MD

## Part 2: Quadrature Decoding for the Motor Driver

FROM THE BENCH

Jeff Bachiochi



In *INK* 62, Jeff used National's H-bridge motor driver with the I²C and PWM functions of a PIC16C73. This month, he adds the decoder circuitry necessary to keep track of the motor's rotation, an essential part of achieving accurate positioning.

"Are we there yet?" comes the familiar cry from the back seat. Out comes the road map. I'm not lost, but the kids need to answer their own question.

Knowing where you are and where you're going helps you plan the best way to get there. Locating each map symbol along the way passes the time and teaches the kids something too.

Luckily, our autos are equipped with odometers. We can follow directions like "Go this way for 2.3 miles then turn left. It's ½ mile farther on your left" without having been there before. Using just the speedometer, you could drive at 50 MPH for 2 minutes and 45 seconds, turn left, and drive another 40 seconds and still arrive at the same location.

However, there are usually a number of unknowns which alter the drive time. For instance, you have to take into account acceleration and deceleration. Traffic might restrict your progress or a steep hill could shift your momentum. Using the odometer makes much more sense. Since the odometer counts increments of distance, it is accurate at any speed, acceleration, or deceleration.

Likewise, when a motor moves a load, its position can be determined by measuring its incremental movement. This measurement can be done at any point from the load right on back to the motor.

Automobiles use tire rotation and circumference to determine distance. An overhead crane might use a linear-displacement sensor which measures the actual position. Many applications can be simplified with an incremental decoder which counts fractions of a revolution when mounted to the shaft of a drive train. Numerous motor manufacturers offer encoders preinstalled on their products.

## OPTICAL INCREMENTAL ENCODERS

Measuring shaft rotation is the simplest form of position sensing. All you have to do is place a small plate with a slot in it onto any shaft in your drive train with a light source on one side of the plate and a light sensor on the other.

When the shaft turns, the plate rotates and, once a revolution, the slot lets light through to the sensor (see Figure 1). This arrangement does have an inherent problem, however. You cannot determine the direction of rotation from the received pulses of light. By adding a second sensor, offset slightly from the first, you can determine direction by watching which sensor sees the light first (Figure 2).

As the number of equally spaced slots on the disc increases, smaller fractions of a revolution can be counted, which increases the resolution of the shaft's position. These slotted plates are called *code wheels.*

Standard code wheels may contain fewer than 100 or more than 1000 slots. The maximum number of slots is based on the minimum width of the shadow cast on the light receiver by each spoke (or nonslot).

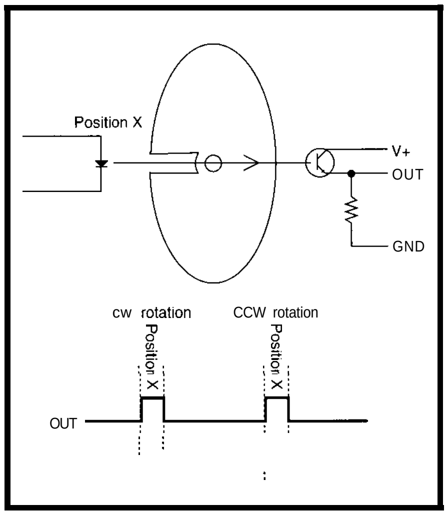


Figure 1—Counting *shaft rotations is easy using an optocoupled pair and a slotted disc mounted to the shaft.*

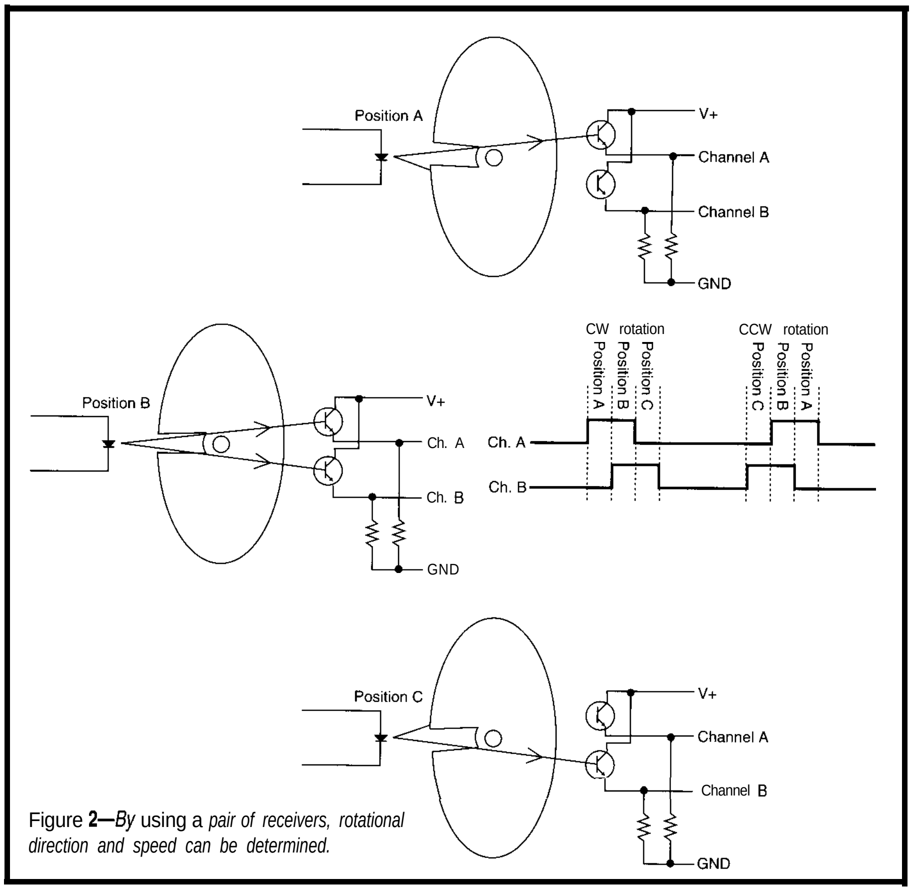


Figure 2—By using a pair of receivers, rotational direction and speed can be determined.

A single LED may be used as a light source on one side of the code wheel for multiple light receivers on the other. Signal-processing circuitry can be used to create a clean dual-channel output. It is important to note the relationship of the receivers to one another and the sizing and spacing of the slots (or spokes) of the code wheel.

Specialized ICs accept the quadrature output from optical incremental encoders and can update an up/down counter based on the input. These are treated as I/O ports and a micro can then read the total counts as needed. Decoder/counter ICs, like the HCTL-2020 from Hewlett Packard, cost about $14 each (I needed two). The HCTL-2020 is an S-bit parallel interface (plus control lines) while the Smart-MD is only giving up a total of 4 bits for the decoder interface.

I believe this decoding is an interesting application for a second PIC processor. By tapping off the Smart-MD processor's crystal oscillator, a second crystal isn't needed. The least expensive PIC, a 16C54, should be adequate (although another PIC with change-of-state and external interrupts would make the job easier).

I must be a glutton for punishment. Could I have used a 40-pin 16C74 for the Smart-MD in the first place and eliminated the second CPU? Yes, but that would have increased both the size and cost of a Smart-MD, which didn't require incremental decoding. Second, I don't believe the '74 could handle the two high-speed priorities of $I^2C$ and dual decoding without a potential loss. Finally, choosing the '74 wouldn't have given me a chance to further demonstrate distributed control by experimenting with interprocessor communication.

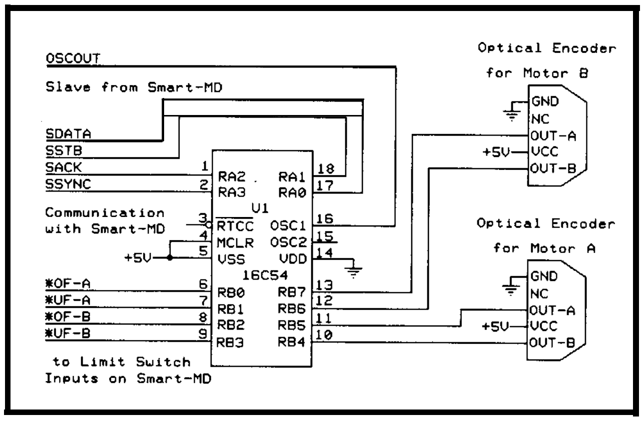


Figure 3-A second PIC processor is used as a slave to last month's Smart-MD project. This slave monitors two optical encoders and counts to $2^{24}$.

## PIC DECODER CODE

Two routines are necessary in the PIC decoder: the decoding or counting of the two optical incremental encoders and communications between the two PIC devices.

If you remember from last month (INK 62), I had four I/O bits left on the Smart-MD. These bits can be used to pass data between the two PIC devices (see Figure 3). I defined one line for synchronization (SSYNC), which goes low only for the first bit in the 26-bit data stream. If the decoder PIC sees this line low at any other time, it assumes it's lost and jumps back to the sync routine.

Two lines are used for handshaking between the two PICs. A write uses a "Here's new data-I got it—I heard you have it-Yup!" sequence. The read sequence proceeds similarly. Data is transferred in both directions over the fourth line. The 26-bit protocol I use is in Table 1.

| Bits | Data Information |
|---|---|
| 1 | 0=read 1 =write |
| 2 | O=A_counter 1 =B_counter |
| 3-10 | most-significant byte (MSB to LSB) |
| 11-18 | next-significant byte (MSB to LSB) |
| 19-26 | least-significant byte (MSB to LSB) |

Table 1-A 26-bit inferprocessor protocol is used for exchanging encoder counts.

After the first two bits are sent, the decoder PIC knows whether it gets or sends 3 bytes and where they are going to or coming from. Although the counters for each encoder can be set up for any value, they are initialized to 800000h, which represents an artificial center. This gives the encoder over 8 million counts either up or down without over- or underflowing the counters.

If you want to move 15,529 steps and automatically stop, you may connect the four least-significant bytes on port B to the Smart-MD's brake inputs. If either counter over- or underflows, one of port B's bits is programmed to go low (one bit for over- or underflow for each of the two encoders). If the counter is set appropriately, it counts up or down and rollover or -under, which applies the brakes to the Smart-MD.

The output bit is reset by simply writing to the counter.

## PRIORITY ONE

Without interrupts, polling determines change of state. The encoder outputs are the highest priority, so must be checked on a regular basis.

A change-of-state routine monitors the encoder outputs and branches when it is necessary to accurately track the counts. A call is made to check the change of state every time the communication routine is waiting for the falling or rising edge of SSTB (the Smart-MD's handshaking line) and TMRO has counted to 64 µs.

In addition, a change of state isn't acknowledged as legal unless the last state has remained constant for at least three time periods. (Three is an arbitrary number, creating a sort of low-pass filter. If your steps per second are slow, raising this number makes the system less susceptible to noise.] This sampling speed (64 us x 3) ensures a change of state is not missed (unless the rate of change exceeds 5,000 steps per second). These constants can be altered to suit your application.

If you are using a 100-slot code wheel mounted directly to the motor's shaft, you receive 100 x 4 (quadrature] or 400 changes per revolution. If your motor does 10 RPM at full speed, that's 4,000 samples per second.

Notably, you may want to alter the number of constant periods to match your encoder's maximum step rate. This improves noise margins and lets interPIC communications proceed with fewer pauses. In addition, a five-time increase in sample time is gained by using the 16C54's 20-MHz crystal.

As you can see in Figure 2, the quadrature encoder outputs are 90° out of phase with each other. With a forward rotation, encoder output A leads output B and vice versa. This technique lets the encoder be used for bidirectional and quarter-step sensing.

Refer to Figure 4, and you can see how this state machine works. First, the encoder outputs are sampled on port B. The O_SAMP (old sample) is compared to the N_SAMP (new sample). If the two bits of encoder B haven't changed, B_C N T is incremented.

If they have changed, B_C N T is checked to see if it has been stable for three samples. If not, the routine goes on to check encoder A bits. If it has been steady for three samples, the last state of encoder B bits are tested to determine the last state of the quadrature encoder. The new state's encoder

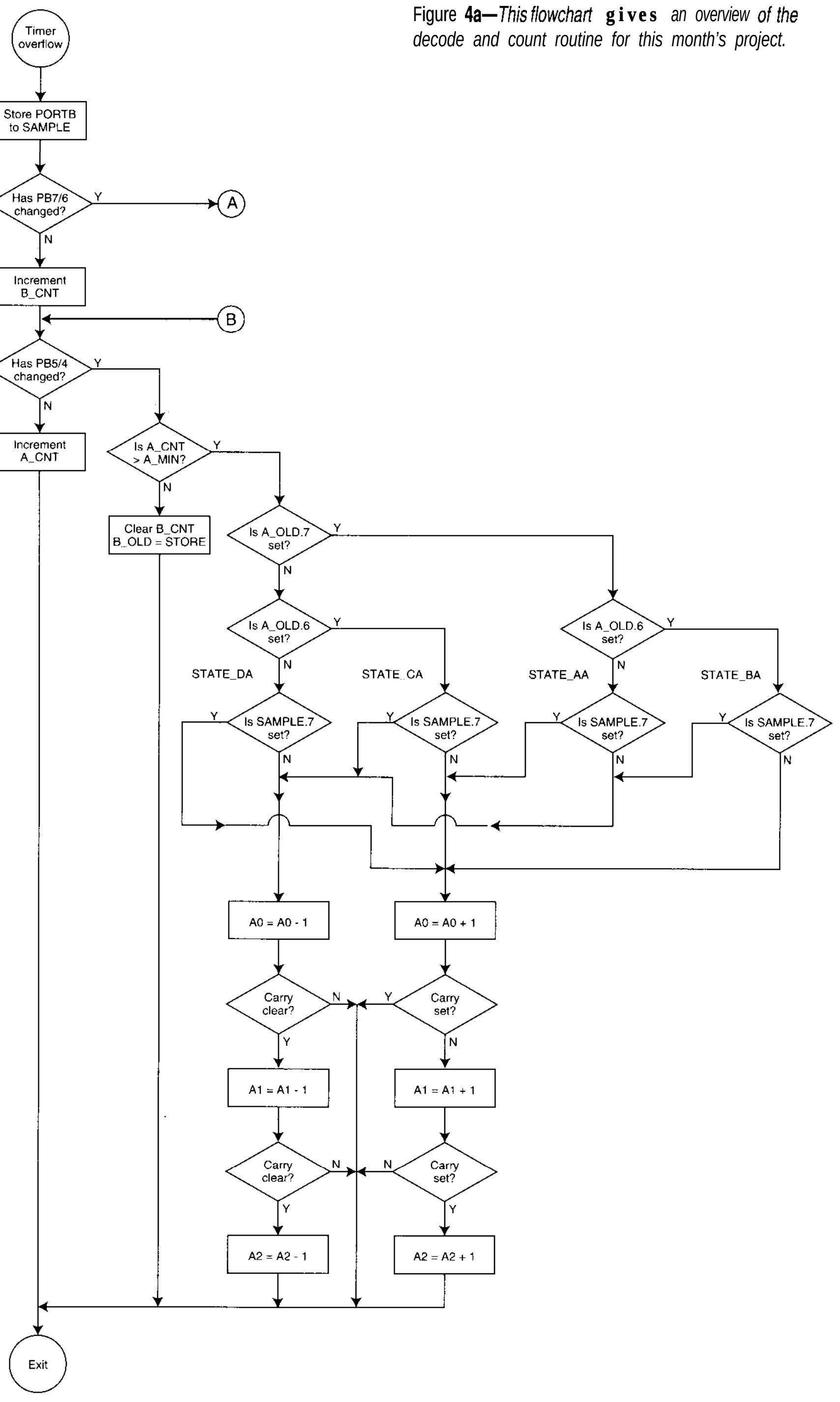


Figure 4a—*This flowchart gives an overview of the decode and count routine for this month's project.*

bits steer the branch to either increment the S-byte (24-bit) counter or decrement it.

Encoder A is given the same examination before returning to its secondary task of communication.

## ENOUGH IS ENOUGH!

What is enough resolution? Of course, the answer depends on the application. If you're moving an overhead crane, 1" might be enough resolution. If you're moving a pick-and-place head, 0.001" is not unreasonable.

Many DC motors have speed-reduction gears to reduce the RPM to a usable level (it might be done to reduce speed or increase torque). Reducing the RPM by a factor of 10: 1 proportionally increases the torque (disregarding transmission loss). Small high-speed motors can actually move large loads through gear reduction.

You have to analyze the power train in your system to establish the most reasonable location for your position-decoding hardware. Assume the final transmission pinion gear, which moves the crane on a rack, has a linear pitch of 1". Hall-effect sensors, placed as pictured in Figure 5, can use the gear's (or the rack's) teeth for an unconditioned output similar to that of optical sensors. The motor ahead of the transmission may rotate at 1000 RPM to move the crane that 1", so there really isn't much use in counting revolutions of the motor's shaft.

On the other hand, if your pick-and-place head is moved by ¼-20 threaded rod, then it only has to rotate 20 times per inch. To get 0.001" resolution (or 1 mil), measure the final shaft rotation 1/50 of a turn.

A DC motor with reduction gears of 5:1 and a code wheel of 100 on the motor yields a quadrature count of:

$$\frac{4 \times 100 \times 5}{50 \times 0.001} = 40\text{cntsper}0.001"$$

Significantly, this is more than enough resolution. In fact, at 1000 RPM (that's about 16 RPS), the quadrature decoder produces 6,400 count per second (i.e., 16 x 4 x 100). Moving the decoder wheel from the motor shaft to the threaded rod reduces the counts per second to 1,280 by the 5:1 gear reduction. This shift also reduces 40 counts per 0.001" to 8 counts per 0.001", which is still plenty of room.

One of the surplus motors that I bought has an HP HEDS9100 encoder built onto the end of the motor shaft [see Photo 1). Hewlett Packard also sells units which can be easily added onto any round shaft. The nice part about these units is that they use the bearings of the motor (or shaft) for support of the code wheel. Unfortunately, they cost about $25 (extra code wheels are about $15 .)

Although Digi-Key does not carry HP, it has some rotary encoders which are suitable for many projects. Built like a potentiometer, these devices are normally used for digital pots or switches. They work well when coupled to a slow-turning shaft. Most give quadrature outputs well below 100 pulses per revolution.

## INTERPROCESSOR COMMUNICATION

Before coding the Smart-MD to access the PIC encoder, I used a Domino (a shrunken and encapsulated version of the RTC52) to test the four-wire interface. Rigging up the test cable took more time than writing the test program.

The test program lets me twiddle four I/O bits replicating the 26-bit communication which normally takes

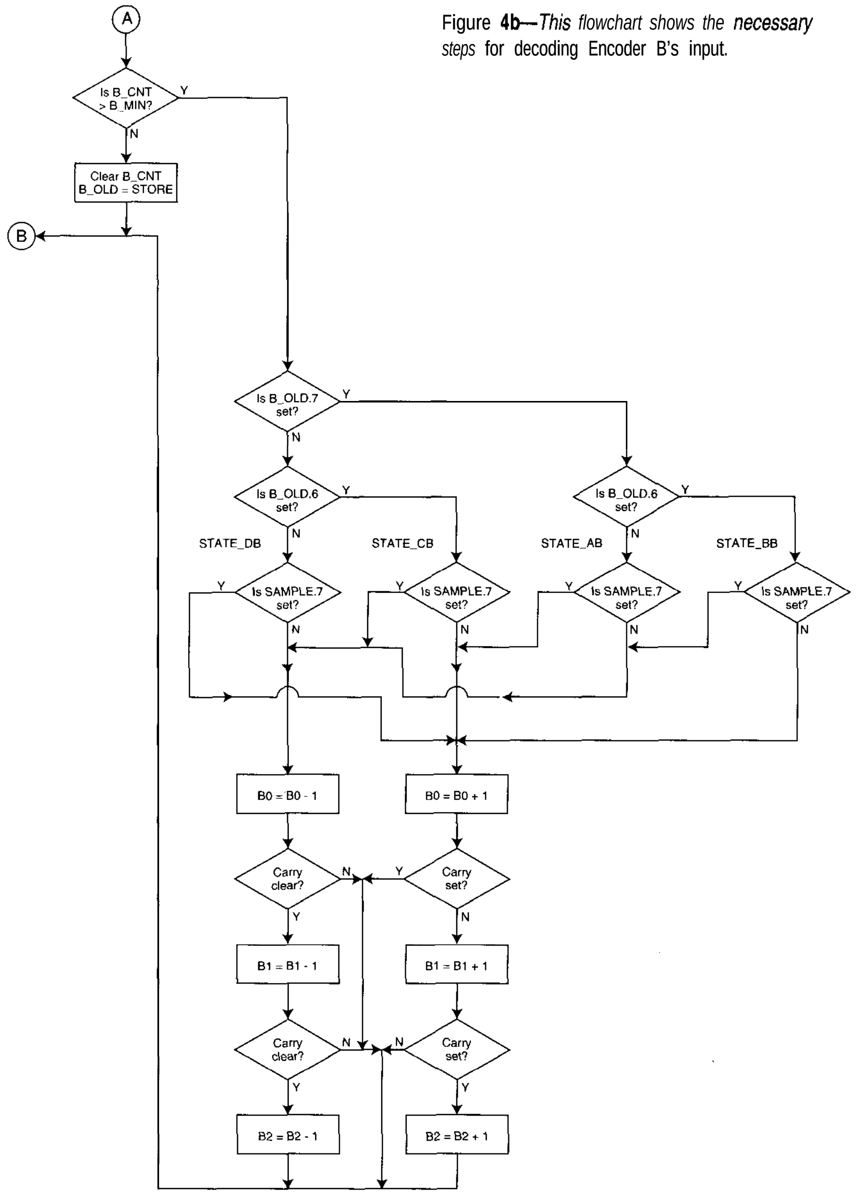


Figure 4b—*This flowchart shows the necessary steps* for decoding Encoder B's input.

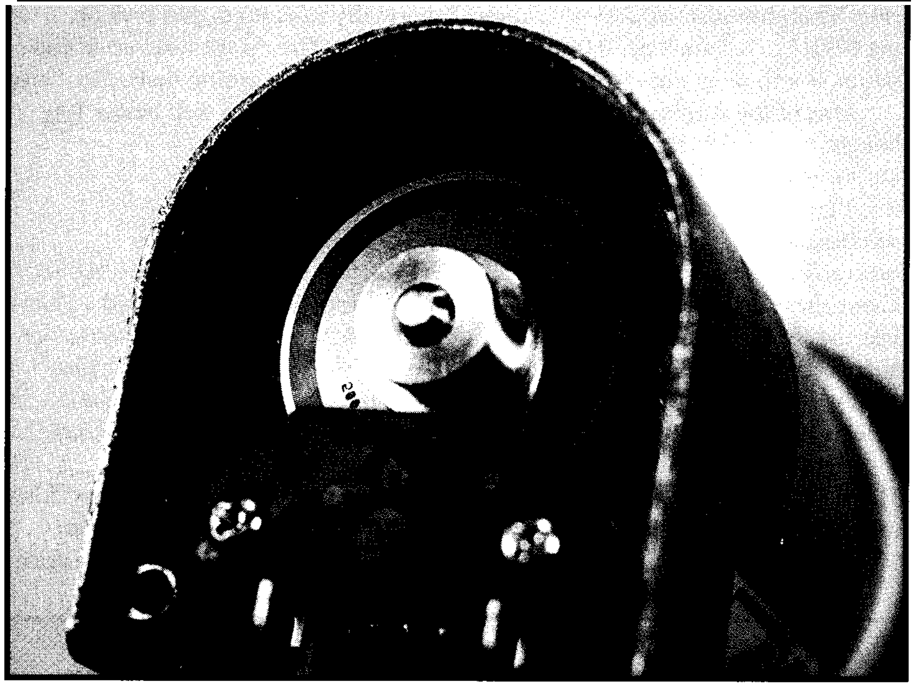


Photo I-An HP *optical* encoder was *preinstalled on this* surplus *DC motor.*

place between PIC processors. Once I was convinced it was reacting correctly, I added the M (most-significant byte), N (next-most-significant byte], and L (least-significant byte) commands to the Smart-MD.

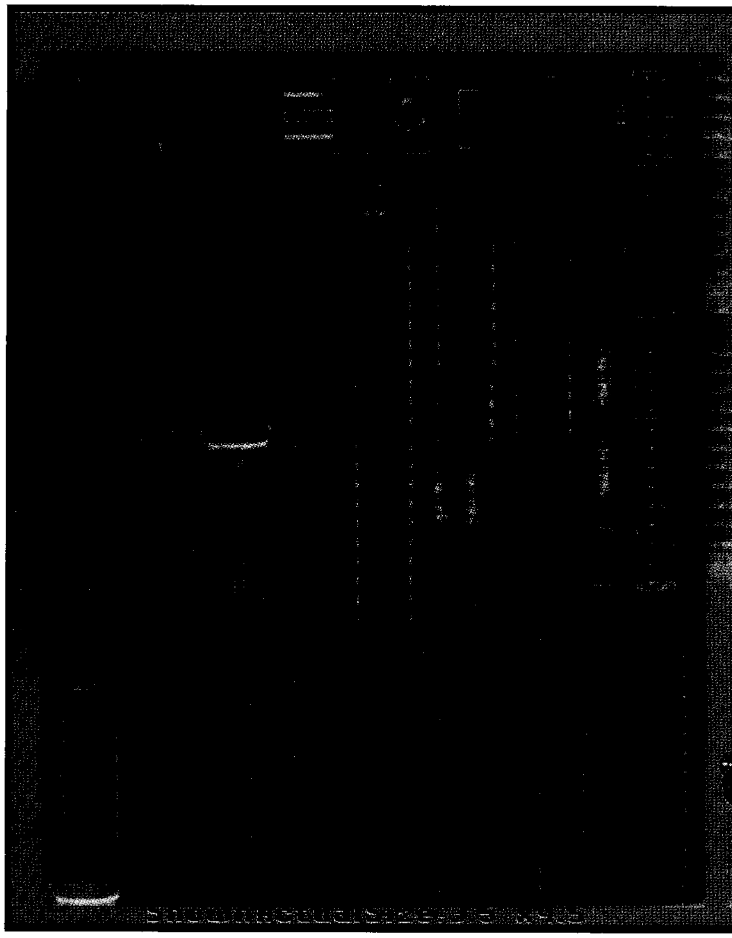
These three commands work together to form a 24-bit count. Even though it takes three commands to set a count, it doesn't require the I²C master to know how to send (and receive) variable-length data. A simple registered byte write and read protocol can be used.

When writing a count, M and N pass parameters but nothing is sent to the PIC decoder until the L of the count is sent. Receipt of L triggers an interPIC write. When requesting the M of a count, an interPIC read occurs to retrieve the latest 24-bit count.

Subsequent requests for N and L merely pass over the already received bytes. Like the existing commands, upper-case commands rule motor A while lower-case commands govern motor B.

## TRADEOFFS

The $5 PIC16C54 replaces two $14 HCTL-2020 quadrature decoder/counters, creating significant savings. I admit, however, that the HCTL-2020s count quadrature pulses more than 200 times faster than a PIC using the same clock frequency.



Big Features SmallSpace

[illegible]

Phone: 31[illegible]

#124

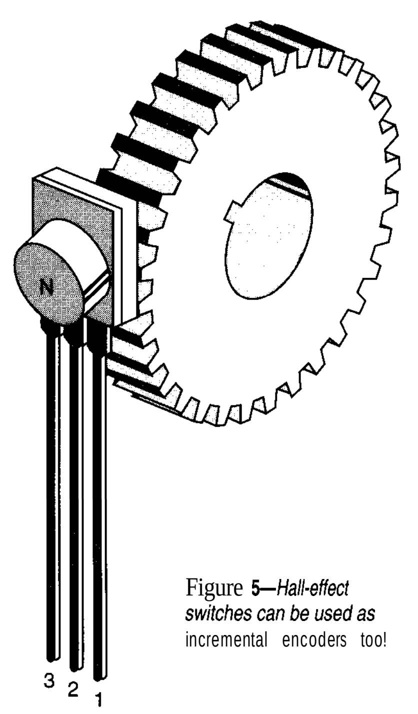


Figure 5—*Hall-effect switches can be used as* incremental encoders too!

On the other hand, the Smart-MD needs a more expensive PIC16C74 (versus '73) to have enough pins to access to HCTL-2020s.

So, what do you need-higher performance or a less expensive design! ❑

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

## SOURCES

**PIC16C54**
Microchip Technology, Inc.
2355 W. Chandler Blvd.
Chandler, AZ 85224-6 199
(602) 786-7200
Fax: (602) 899-9210

**HEDS9100** optical rotary encoders
Hewlett-Packard
OptoElectronics Division
(800) 452-4844

**9-mm** square sealed encoders
Bourns, Inc.
1200 Columbia Ave.
Riverside, CA 92507
(909) 781-5500
Fax: (909) 781-5273

Rotary encoders
CTS Corp.
905 West Blvd. N
Elkhart, IN 465 14
(219) 293-7511
Fax: (219) 293-7511

Clarostat Sensors and Controls
1600 W. Piano Pkwy.
Plano, TX 75075
(603) 742-l 120
Fax: (214) 423-4661

Bourns, CTS, and Clarostat encoders are also available at:
Digi-Key Corp.
P.O. Box 677
Thief River Falls, MN 56701
(800) 344-4539
Fax: (218) 681-3380

## I R S

428 Very Useful
429 Moderately Useful
430 Not Useful

WE'RE N[illegible]

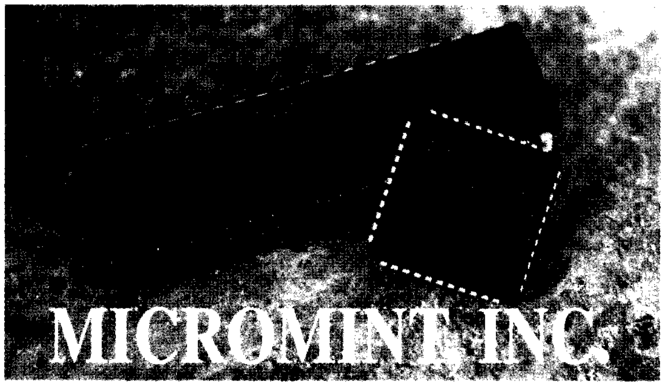
*80C52-BASIC CHIPS IN VOLUME*

Micromint's 80C52-BASIC chip is an upgraded replacement for the venerable Intel 8052AH-BASIC chip. Ours is designed for industrial use and operates over the entire industrial temperature range (-40°C to +85°C). Available in 40-pin DIP or PLCC.

*80C52-BASIC* chip **$19.00**

*OEM 100 qty.* **$12.00**

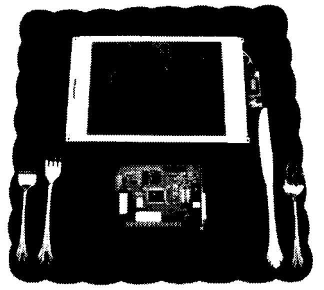
*BASIC-52 prog.* manual **$15.00**



Call *(860) 871-6170* or *l-800-635-3355* to order
MICROMINT, INC. 4 PARK STREET, VERNON, CT 06066

#119

**Flat Panels Served Here**



House Specialty - Monochrome LCD Kit $249

*Le Earth LCD Restaurante Menu*

| | |
|---|---|
| *Appetizers* (Controllers for PC ISA Bus) | |
| Earth LCD/M Monochrome LCD Controller | $149 |
| EarthVision/ISA Color LCD Controller | $299 |
| *Entrees* (Include LCD, ISA Controller, Cable & Backlite Inverter) | |
| Monochrome 9.4" Kit | $249 |
| Color Single Scan 8.2" | $450 |
| Color Dual Scan 9.4" | $749 |
| Color 9.4 TFT | $995 |
| *Ala Carte* (LCD Only) | |
| Mono 9.4" LCD | $95 |
| Color 9.4" Single Scan | $125 |
| Color 9.4 Dual Scan | $399 |

EARTH chnologies "The At Panel Solutions Company"

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

#125

# I Want My DTV

**SILICON UPDATE**

**Tom Cantrell**

## Philips Chips Bring Digital TV One Step Closer

Tom reviews digital video before moving on to Philips' new SAA7111 video input processor. Tom zooms in on how this chip simplifies and improves processing. Digital TV may soon be off the ground.

Much (too much?) has been said about the imminent convergence of television and computer. Along with games and blizzards of E-mail, TV on PCs promises to soak up what little productivity office workers have left. Heck, wouldn't you rather watch OJ or Oprah than work on that boring report?

Already, satellite services are beaming digital TV earthward, and the cable and telecommunication folks are champing at the bit to get into each other's businesses. Of course, there's the requisite army of lobbyists searching for the right pocket to line.

However, with 200 million or so plain old analog TVs in place, the switch to digital won't happen overnight. During the transition period, technology will have to finesse a way around the fact that the connectors on the backs of TVs and PCs don't even plug into-much less play with-each other.

Fortunately [or not, depending on your outlook), when it comes to silicon, if there's a will (or better yet, a PO), there's a way.

## TV 101

Before getting started, it's worth saying a few words about how digital video works. Since I know just enough to be dangerous, I suggest you dig through the closet until you find *INK* 6, 7, and 8 and read "ImageWise/PC— The Digitizing Continues" by our very own Ed Nisley.

A masterful design, the Image-Wise/PC (Photo 1) predates all of today's PC video hoopla by eons in IC terms. Nevertheless, even today, the design fundamentals discussed in the article remain the same. Only the ICs have changed. Figure 1 shows the ImageWise/PC block diagram which itemizes the basic elements of all digital-video designs.

The first step is digitizing incoming video, which sounds easy, except that major headaches are conveniently hidden in innocent-sounding analog-processing and timing-generation boxes.

Most of you are probably aware of the basics of composite video (also known as CVBS). A 1-V peak-to-peak signal encodes a variety of information besides video, including the horizontal- and vertical-sync pulses that signal the CRT beam to reset from the end to the beginning of the line and field,

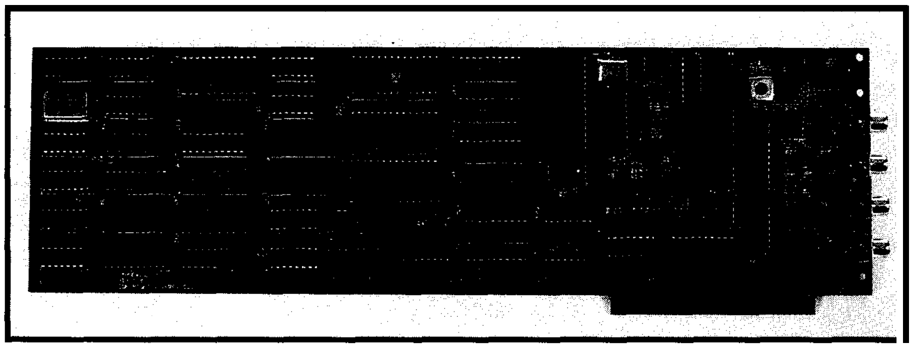


Photo 1 —*Despite* clever design and limited specs (256 x 256 x *8-bit grey scale), in the '80s a digital video design like the Imagewise/P C required a board full of chips.*

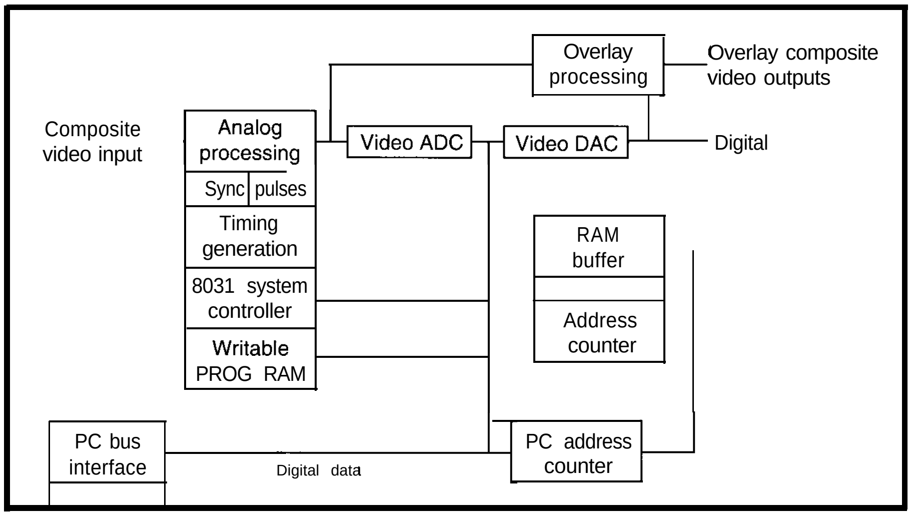


Figure I--The *Imagewise/PC* block diagram highlights *fhe major* functional blocks *(analog* processing, timing generation, *frame buffer, and AD and D/A converters)* of *all digital video designs,*

respectively. Two fields (even and odd) are displayed one after the other (i.e., interlaced) to make up a single frame. The video-signal voltage determines the brightness (or luminance) while the color (or chroma) information is handled by superimposing an AC signal on the intensity.

So right away, we're in a heap of trouble. We've got to separate the signal into luminance and chroma components, a task the luminance-only (8-bit gray scale) Imagewise/PC was able to punt around by simply throwing away the color with an aptly named chroma trap. Once the signal is split, luminance and chroma can be sent to high-speed A/D converters.

Ah, but when to sample? It basically boils down to figuring out where the heck we are (which field, line, dot) at any given time by extracting and interpreting the sync pulses.

What would be just a difficult design challenge becomes horrendous when you consider the myriad of real-world gotchas. For instance, Ed's article well conveys the exquisite pain of discovering that different video gear suppliers play the video specs fast and loose, rendering simple-minded sync-counting schemes useless.

Then there's the minor problem that the immaculate-looking signal waveform is rarely seen outside a lab. Everything from your PC to the migrating geese pulling over for a rest stop on the antenna is a potential interference source. Users won't listen to your excuses since they expect your gadget to pull in shows from the next state. Meanwhile, VCRs, saddled with Rube Goldberg-ish transport mechanisms, get a little lackadaisical about sync delivery.

Just to up the fun quotient, remember there are a number of international standards: NTSC (USA, Canada, and Japan), SECAM (France, Eastern Europe, and Russia), and PAL (Western Europe, Asia, and most everywhere else). Except for a few basics (1 V p-p, 2:1 interlace, 4:3 aspect ratio), everything else including the all-important sync timings (i.e., lines per frame and fields per second) and color-carrier frequencies is geocentric. Oh, don't forget S-VHS, which uses separate connections for luminance and chroma.

Assuming you finally decipher the feeble and mysterious signal, it can be sent along to the A/D converter. Don't think you're going to get away with the same wimpy instrumentation-type A/D converter you might use for a temperature or pressure sensor either.

A few quick-and-dirty calculations reveal that the required conversion bandwidth in megaHertz is roughly approximated by two times the pixels per line divided by 100.

In other words, to match a PC's 640-pixel line requires pumping the data at 12+ MHz-ouch! Don't forget the sample rate should be at least doubled to minimize aliasing anoma-

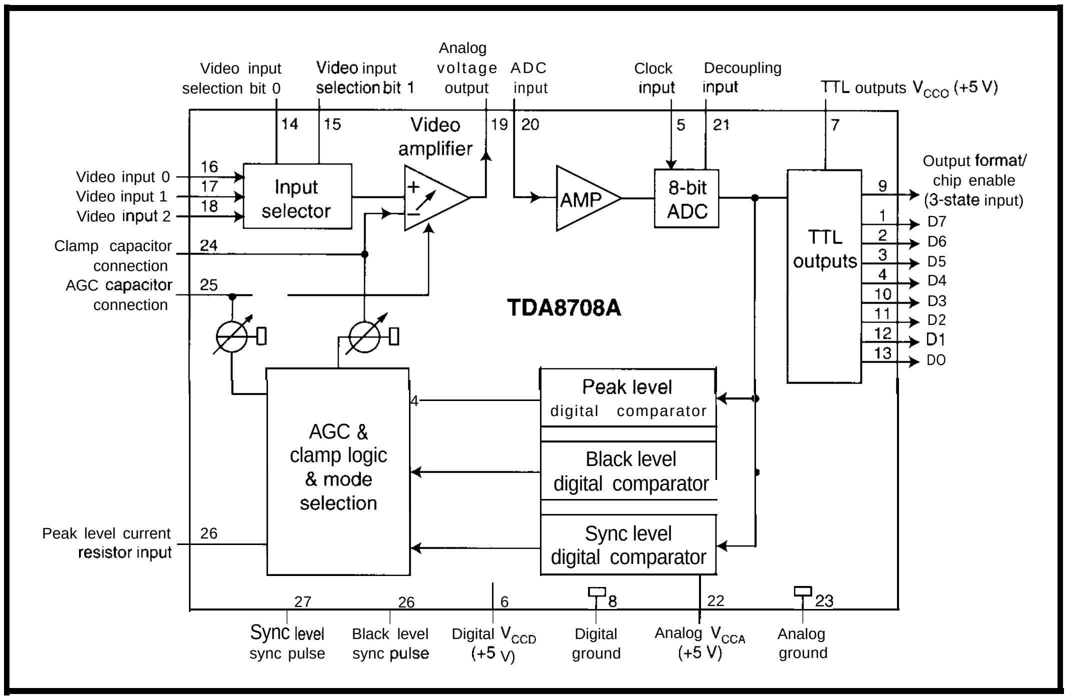


Figure **2**—*The critical spec for a* video *A/D converter is* sampling *rate. The popular TDA8708* gets *up to* 32 MHz.

lies. Needless to say, so-called video or flash A/D converters have earned their reputation as rather expensive+ finicky, high-power beasts.

The high sample rate also harshens the fact that once the video, has been grabbed, it must be stuffed somewhere quickly. Thus, the frame buffer memory is no slouch either. Ed had to use SRAM, but fortunately for our wallets, today's DRAMs, VRAMs, and specialized line memories can just keep up.

We can breathe a sigh of relief once the video is safely stashed away as 1s and 0s. Now, a good old micro or DSP can crunch it any which way but loose. Scaling, clipping, filtering, special effects, and so on make up these functions sometimes called *feature processing.*

However, the digital joy is short-lived if you plan to send the result back into the video realm. Mushing everything back together calls for a similarly speedy D/A converter and the need to deal with quirky timings and myriad standards all over again.

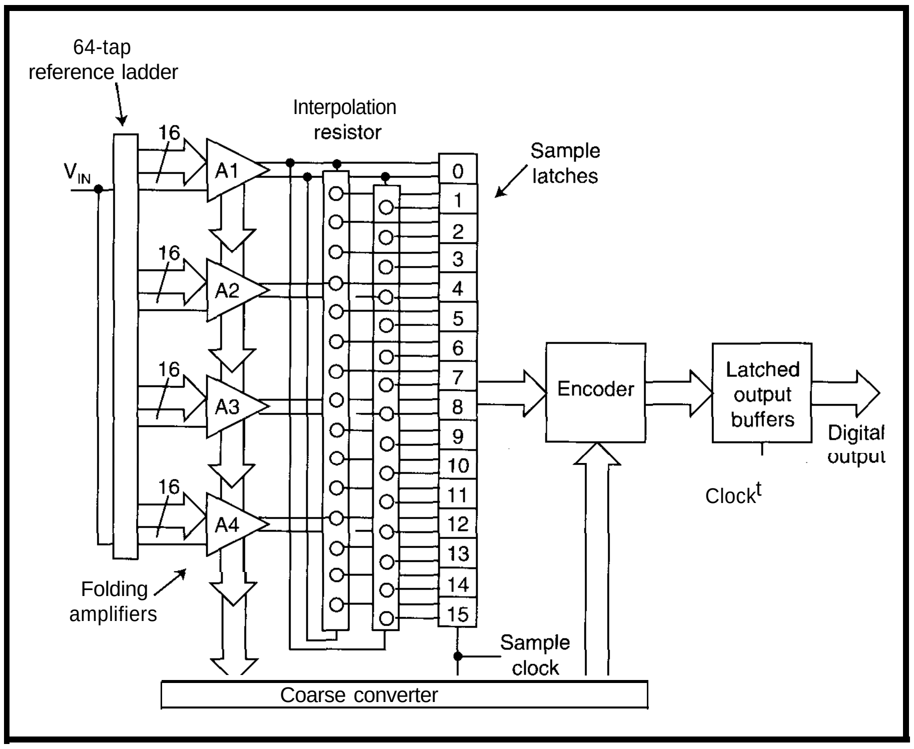


Figure 3—*Philips* video *A/D* converters use a *combination* of folding *amplifiers and interpolation resistors to significant/y reduce cost and power consumption.*

## SILICON TO THE RESCUE

Given the technology of the time, it's amazing Ed was able to fit the design on a PC plug-in board. He may claim his "favorite programming language is solder," but in fact the Image-Wise/PC depends on some truly head-scratching 803 1 assembly language. Otherwise you'd have to handle various functions (e.g., timing control) with even more chips.

Silicon marches on, so why not up the ante with demands for much higher resolution, color support, fewer chips, simpler design, and lower power? For those of us with mere human intelligence, it would also be nice not to have to write any guru-class firmware.

Given its background [the TV know-how of a consumer giant and the merchant market IC savvy acquired via purchase of Silicon Valley old-timer Signetics a few years back), it's not

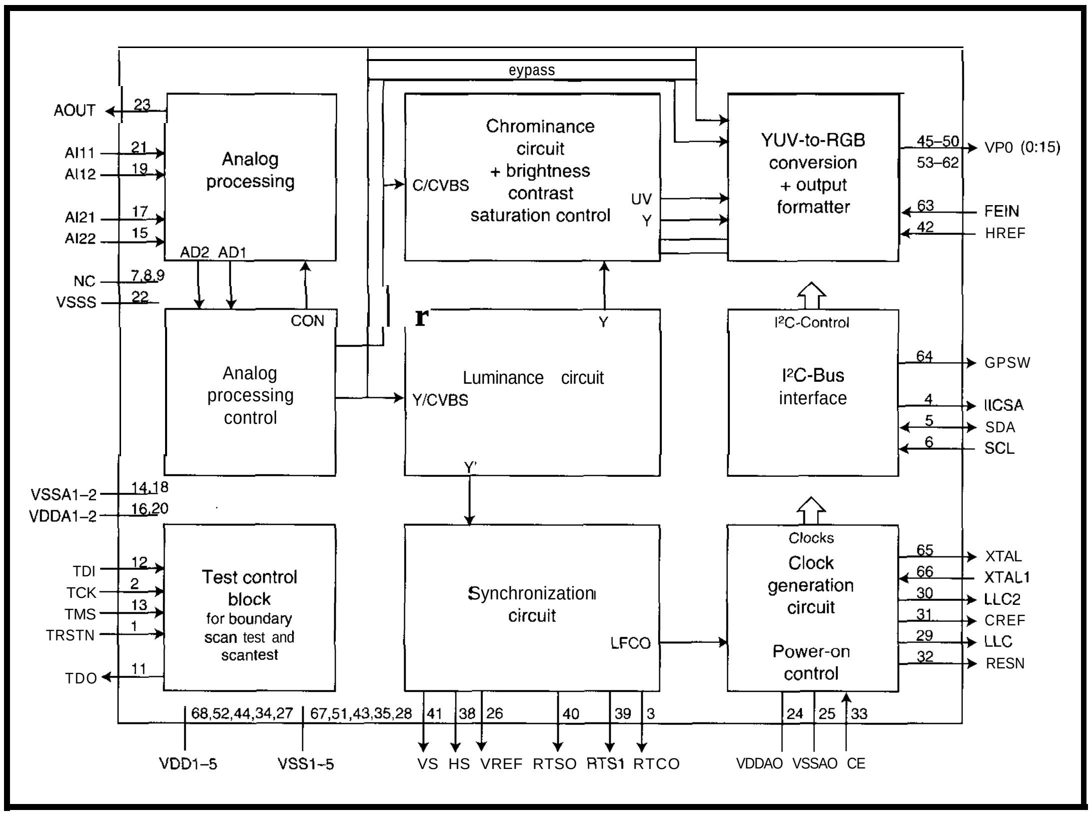


Figure *4-The VIP (SAA71 11) integrates the entire front-end including analog processing, multistandard decode, two 8-bit ND converters, and a YUV/RGB output formatter on a single chip.*

$$\begin{aligned} Y &= 0.299 \times R + 0.587 \times G + 0.114 \times B \\ U = C_b = (B - Y) &= -0.169 \times R - 0.331 \times G + 0.500 \times B \\ V = C_r = (R - Y) &= -0.500 \times R - 0.419 \times G + 0.081 \times B \end{aligned}$$

Figure C-These *equations* convert *YUV,* which separates luminance *(Y)* and chroma *(UV), info RGB,* which combines them. Note *that all* fhe values *are* normalized (i.e., *Y* and *RGB* vary *from 0 to 1* while *U* and V range from -0.5 *to +0.5).*

surprising that Philips Semiconductors offers some helpful ICs. In fact, look inside most multimedia designs (including the Mac), and you're likely to find some Philips chips.

Over the years, they've offered key pieces of the puzzle, such as the widely used TDA8708 video A/D converter (see Figure 2). The device includes three selectable video inputs, a variety of analog-processing circuits (clamp, gain, sync-level detection) and a quick 32-MHz S-bit A/D converter.

The '8708's popularity is largely a byproduct of its relatively low price (under $10 in 100s) and power consumption (less than 0.5 W typically). Both specs are the final result of some clever design techniques.

For instance, to complete conversion as quickly as possible, a traditional S-bit flash A/D converter uses a 256-tap precision resistor ladder feeding 256 comparators followed by 256 result latches, which are subsequently encoded into an S-bit output. The Philips design, depicted in Figure 3, exploits some unique folding and interpolation techniques to reduce the respective counts to 64, 64, and 16. The bottom line is more bits for fewer bucks (and watts).

Philips has supplemented the all-important A/D converter with other bits and pieces such as clock generators, sync extractors, and the requisite D/A chips. Subsequently, recognizing the IC biz reality that last year's chip set is tomorrow's chip, they've integrated functions together step-by-step. In fact, as of today, they've managed to shrink most of the tricky stuff into a mere two chips.

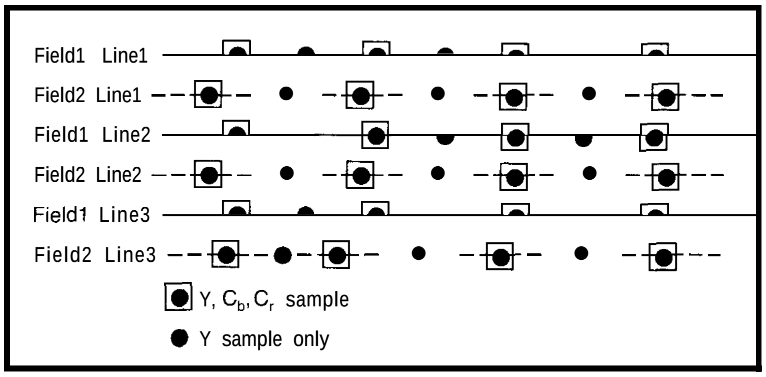


**Figure 6**—*Exploiting the fact that fhe human eye is more sensitive fo intensify than co/or, YUV sampling schemes (in this case 4-2-2) devote more bits to the former than the latter.*

## VERY IMPORTANT PROCESSOR

Actually, the SAA7111's moniker stands for Video Input Processor. As its name implies, the VIP, shown in Figure 4, handles the entire front-end (i.e., most everything between the In jack and the frame buffer) in a single chip.

Connected to a single 24.567.MHz crystal, the VIP's very sophisticated digital PLL circuits handle all the messy clock generation and sync extraction details. Indeed, the chip is practically smart enough to figure out and configure itself for the proper variant of NTSC and PAL automatically! Magically, out comes H-sync, V-sync, and a line-locked clock (LLC), plus a bunch of useful control and timing signals.

Four analog inputs can be configured for a variety of input combinations such as four composite, two composite, and one S-VHS (also known as YC), and so on. Since we're handling color, two '870%like S-bit A/D converters (one for luminance, one for chroma) are provided.

The separated and digitized luminance and chroma are fed to the respective processing blocks that perform a myriad of enhancements including programmable gain, peak control, filtering, brightness, contrast, saturation, and so on. The cleaned-up digital output is almost ready to cram into RAM via the 16-bit VPO (Video Port Out) bus.

At this point, discussion of some details about color, which wasn't an issue for the ImageWise/PC (lucky for Ed) is unavoidable. The major point is that there are two popular ways to describe (or code) a pixel.

RGB splits the description into red, green, and blue components, each contributing to the overall intensity. YUV refers to a model where Y (the intensity) is accompanied by U and V, which are the so-called color-difference signals (Y -blue and Y – red, respectively).

It's quite possible to translate between the two conventions, as you can see in Figure 5. So, you might ask what's the diff? The answer falls into that interesting category of tricks and optical illusions that exploit particular characteristics of the human eye.

Way back when you may have learned how the human eye relies on rods and cones that act like photodiodes *(INK 60), you* may even remember that the rods detect intensity and the cones detect color. The final piece of the puzzle falls into place when you recall that there are more rods than cones and the rods feature finer spatial resolution to boot.

Thus, the main advantage of YUV is that splitting the signal along human lines offers a unique compression opportunity (i.e., fewer bits need to be devoted to color). Actual practice includes schemes such as 4-l-l and 4-2-2, where each number corresponds to a Y, U, and V sample as is shown in Figure 6.

RGB, by mixing intensity in with the color information, sacrifices compression to simplicity of connection to a CRT's guns or a camera's CCD. Oh yeah-don't forget RGB is where it's at for PCs too.

Getting back to the VPO bus, the VIP handily offers four output formats: 4-1-1, 4-2-2, CCIR 656 4-2-2 (MPEG compatible), and RGB as shown in Table 1. The wider formats run at half

| BUS | 411 | | | | 422 | | CCIR 656 | | | | RGB |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Signal | 12 bit | | | | 16 bit | | 8 bit | | | | 16 bit |
| VP015 | $Y_{07}$ | $Y_{17}$ | $Y_{27}$ | $Y_{37}$ | $Y_{07}$ | $Y_{17}$ | $U_{07}$ | $Y_{07}$ | $V_{07}$ | $Y_{17}$ | R4 |
| VP014 | $Y_{06}$ | $Y_{16}$ | $Y_{26}$ | $Y_{36}$ | $Y_{06}$ | $Y_{16}$ | $U_{06}$ | $Y_{06}$ | $V_{06}$ | $Y_{16}$ | R3 |
| VP013 | $Y_{05}$ | $Y_{15}$ | $Y_{25}$ | $Y_{35}$ | $Y_{05}$ | $Y_{15}$ | $U_{05}$ | $Y_{05}$ | $V_{05}$ | $Y_{15}$ | R2 |
| VP012 | $Y_{04}$ | $Y_{14}$ | $Y_{24}$ | $Y_{34}$ | $Y_{04}$ | $Y_{14}$ | $U_{04}$ | $Y_{04}$ | $V_{04}$ | $Y_{14}$ | R1 |
| VP01 1 | $Y_{03}$ | $Y_{13}$ | $Y_{23}$ | $Y_{33}$ | $Y_{03}$ | $Y_{13}$ | $U_{03}$ | $Y_{03}$ | $V_{03}$ | $Y_{13}$ | RO |
| VP010 | $Y_{02}$ | $Y_{12}$ | $Y_{22}$ | $Y_{32}$ | $Y_{02}$ | $Y_{12}$ | $U_{02}$ | $Y_{02}$ | $V_{02}$ | $Y_{12}$ | G5 |
| VP09 | $Y_{01}$ | $Y_{11}$ | $Y_{21}$ | $Y_{31}$ | $Y_{01}$ | $Y_{11}$ | $U_{01}$ | $Y_{01}$ | $V_{01}$ | $Y_{11}$ | G4 |
| VPO8 | $Y_{00}$ | $Y_{10}$ | $Y_{20}$ | $Y_{30}$ | $Y_{00}$ | $Y_{10}$ | $U_{00}$ | $Y_{00}$ | $V_{00}$ | $Y_{10}$ | G3 |
| VP07 | $U_{07}$ | $U_{05}$ | $U_{03}$ | $U_{01}$ | $U_{07}$ | $U_{05}$ | X | X | X | X | G2 |
| VPO6 | $U_{06}$ | $U_{04}$ | $U_{02}$ | $U_{00}$ | $U_{06}$ | $U_{04}$ | X | X | X | X | G1 |
| VP05 | V07 | $V_{05}$ | $V_{03}$ | $V_{01}$ | $U_{05}$ | $V_{05}$ | X | X | X | X | GO |
| VP04 | V06 | $V_{04}$ | $V_{02}$ | $V_{00}$ | $U_{04}$ | $V_{04}$ | X | X | X | X | B4 |
| VP03 | x | x | x | x | $U_{03}$ | $V_{03}$ | x | x | x | x | B3 |
| VP02 | x | x | x | x | $U_{02}$ | $V_{02}$ | x | x | x | x | B2 |
| VP01 | x | x | x | x | $U_{01}$ | $V_{01}$ | x | x | x | x | B1 |
| VP00 | x | x | x | x | $U_{00}$ | $V_{00}$ | x | x | x | x | B0 |
| | | | | | | | | | | | |
| Pixel Order | | | Y 0 | 1 2 | 3 0 | 1 | 0 | | 1 | | |
| Pixel Order UV | 0 | | | | 0 | | 0 | | | | |
| Data rates | LLC2 | | | | LLC2 | | LLC | | | | LLC2 |

Table 1—*The VIP output* port offers *four different* formats: *YUV* 4-l-1, 4-2-2, CC/R 656 4-2-2 *(MPEG* compatible), and *RGB.*

the line-locked clock frequency (i.e., 13.5 MHz, which is CCIR compatible).

Software to drive the VIP is simply a matter of setting up a few dozen configuration and control registers. Access is via the well-known I²C bus, which is pretty much the LAN-In-A-Box of choice in the consumer electronics world.

The VIP also includes a so-called JTAG (IEEE 1149 standard) clocked-serial interface that is slowly but surely appearing on more and more ICs. It's particularly designed to support boundary scan, which basically sets and / or reads the levels on the device's pins. Needless to say, this can be very handy for debugging during the design phase and production testing as well.

## SLAM DENC

Just as the VIP sweeps the entire front-end into a single chip, the SAA7184 DENC (Digital Video Encoder) cleans up the back-end [i.e., the frame buffer to Out jack). Designed to work in concert, the VIP and DENC share many common signals (most notably the various clocks and sync pulses), thereby providing a simple connect-the-dots design. As shown in Figure 7, the DENC offers 24 digital data lines split across the MP (MPEG), VP (Video), and DP [Data) 8-bit ports.

The MP port only accepts the CCIR656 8-bit data stream as output by the VIP. The VP port can accept that format as well by switching between the two channels via SEL_ED.

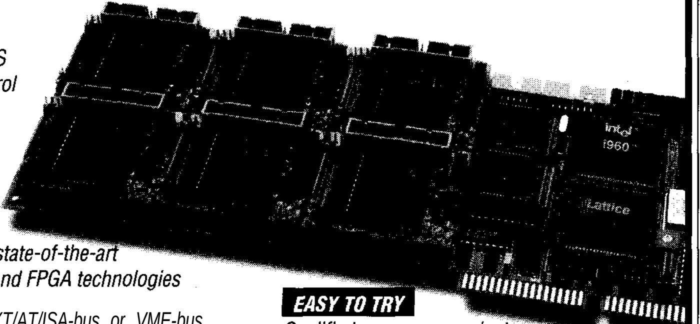
The DP port can be configured as a parallel micro interface, supplementing the expected I²C port option, depending on the state of the SEL_MPU pin. If SEL_MPU is high, the control lines take on a 68k flavor (*CS, R/* W, AO, DTACK), and DP functions as an 8-bit data port.

Alternatively (i.e., the I²C interface is selected), DP is configured to work with VP to implement a 16-bit 4-2-2 input with Y on VP and UV on DP. Once again, this is compatible with the corresponding output mode of the VIP.

The selected input is fed to an encoder that massages it into either NTSC or PAL format. Care is taken to generate a clean signal with a variety of techniques including various gain

**MOTION** CONTROL

FOR ROBOTICS & **MACHINE** AUTOMATION



- *Up to 8 AXES motion control and 148 I/O control on a single board*
- *Controller boards incorporate state-of-the-art RISC, DSP, and FPGA technologies*
- *Use in PC/XT/AT/ISA-bus or VME-bus. Can also operate* STAND-ALONE *using their own on-board "CPU t math co-processor t memory t I/O" capabilities*
- *ON-BOARD* features include MULTI-*TASKING (the ability to simultaneously perform many independent operations), COMPLEX-CONTOURING, and the ability to be programmed directly in USER-UNITS such as inches, meters, etc.*

**EASY TO TRY** *Qualified users may evaluate our products free of charge*

**EASY TO BUY** *Controllers are available for immediate shipment to any part of the world*

**EASY TO USE** *Controllers are supplied with POWERFUL SOFTWARE (free), and clear; concise manuals for easy set-up, programming and operating; supported with immediate technical application assistance*

PMC
Precision MicroControl Corp.

*For more information or to order call:*

***(619) 930-0101***

2075-N Corte *del Nogal* • *Carlsbad, CA 92009-1415* • *FAX (619) 930-0222*

#127

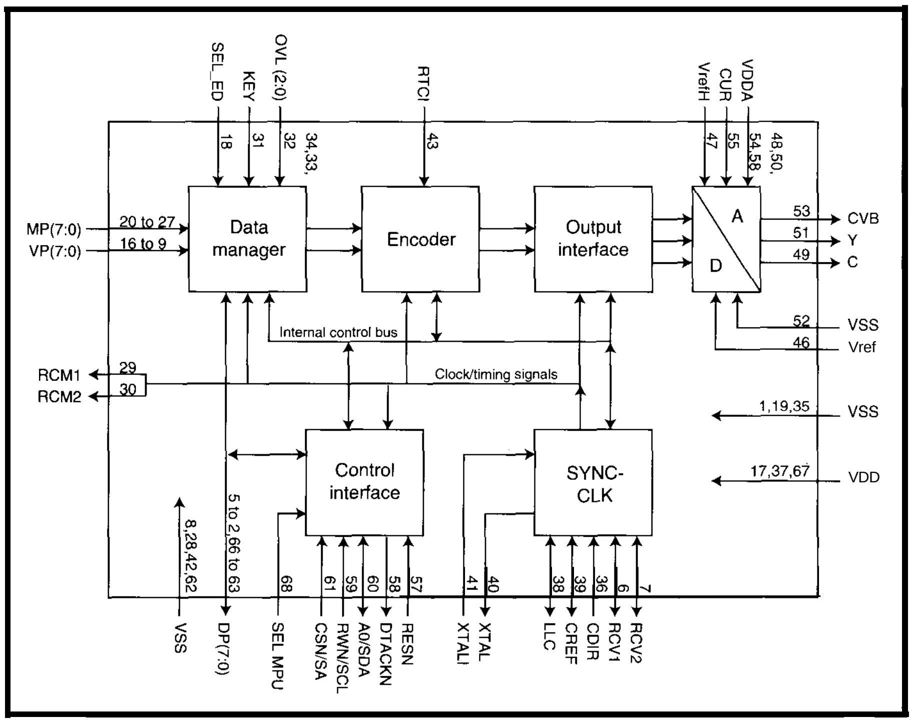


Figure 7—*The DENC (SAA7184) handles the back end of the design. Accepting V/P format YUV 4-2-2 data, the DENC encodes if in NTSC or PAL format* (including *closed* captioning *and optional antitaping protection) and outputs both* composite video *(CVBS)* and *S-VHS (YC)* via *10-bit, 27-MHz D/A* converters.

controls and filters. Notably, the input is upsampled (from 13.5 to 27 MHz) and interpolated to achieve effective IO-bit resolution.

The encoder also handles closed captioning and extended data service, both of which rely on cramming a couple of bytes of info into the nonvisible portions of each field.

In passing, it's interesting to note the DENC can also twiddle the signal to implement the so-called *Macrovision antitaping scheme.* Ignoring the politics, finances, and emotions surrounding such a move, the scheme is technically interesting. Basically, the concept is summed up in the question "What can we do to this signal that won't be noticeable on a CRT, but will really goof up a VCR?" The answer includes things like pseudo-syncs, tomfoolery with AGC pulses, colorburst phase inversion, and other weird stuff. Finding out more requires signing up for a license with Philips.

The last step before the reconstructed signal hits the screen is via two 10-bit D/A converters: one each for luminance and chroma. With a little more filtering, the Y and C outputs are also summed to generate a CVBS [composite) output as well.

Smart chips like VIP and DENC surely take a lot of the pain out of difficult designs. Yes, they diminish the satisfaction of doing the impossible but, at least for me, complicated high-speed designs are invariably accompanied by more agony than ecstasy. At $21.81 (in 100s) for the VIP and 514.75for the DENC, I couldn't come up with a lower cost solution, much less one with a more reasonable-bucks-per-gray-hair quotient. ❑

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

## CONTACT

Philips Semiconductors
811 East Arques Ave.
Sunnyvale, CA 940883409
(800) 234-7381
Fax: (708) 296-8556

## I R S

431 Very Useful
432 Moderately Useful
433 Not Useful

# CONNECTIME conducted by Ken Davidson

The Circuit Cellar BBS
**300/1200/2400/9600/14.4k** bps
24 hours/7 days a week
(860) 871-I 988-Four incoming lines
Internet E-mail: **sysop@circellar.com**

The *big news* around central Connecticut these *days is our new* area code. Connecticut has *had a single* area code *for* decades, and nobody is happy about having to change now. While the old code will continue *to work for the next year or so, it's not too soon to change your dialing directory. Change the BBS area code to 860 and you can forget it (at least for a few more years).*

*We start this month with a thread on touch sensors that gradually evolves into an elevator discussion. It does eventually come full circle, so jump right in.*

*In* the other *thread, we* look at some *methods for calculating integer square roots, with a rather clever outcome.*

## Touch sensor

**Msg#:10134**
From: Tom Eng To: All Users

I am investigating design alternatives for a touch sensor designed to detect a contact between a finger and a sensing pad [about ¼″ x ¼″). The sensor will be integrated into a custom mixed-digital/analog IC with an embedded 8-bit microcomputer core. The chip will be used in a battery-powered portable electronic equipment operating indoors or outdoors in the temperature range of 10–40°C (50–104°F).

The output of the sensor, preferably digital, will be read by the microcomputer. The sensor output will be asserted for as long as the finger contacts the touch-sensor pad. The duty cycle (percentage of time a contact is made) is less than 5%.

Here's my wish list:

1. Small space. The ideal solution is to integrate the design in a single chip with the microcomputer. Extra ROM space (a few hundred bytes) and CPU cycles are available for touch sensor implementation if necessary. The total area available for additional external surface mount parts, if any, is about ¼″ x ¼″.

2. Low power. Draw current in the microampere range when no finger contact is made, and no more than 2 mA when a contact is detected.

3. Low voltage. Must operate at 3 V ±10%. Under 1.5 V is even better.

4. Low-cost sensing pad. No exotic material and no complex manufacturing process. Maintenance free when used in a clean office environment. Last about 100,000 contacts of no longer than a few seconds each in the product life time of about three years.

5. Antistatic control. Survive static electric charges picked up by a human body in a typical office environment.

6. Robust design. Tolerate reasonable IC process variations on analog components. Your suggestions will be greatly appreciated.

**Msg#:10383**
From: Russ Reiss To: Tom Eng

It seems as though you need a simple on/off closure type of input. What am I missing? Why not simply use one of the many microminiature push buttons around? Or how about a membrane switch? Some are available with domes in them to provide for just about any "feel" you wish. Either of these offers zero current when off (except for possibly a pull-up resistor on the micro's input port bit) and low current (even as low as a few microamps) when pressed. Technology is simple, reliable, and very inexpensive. "Interface" is trivial.

If this doesn't meet your needs, perhaps you can explain why. Then we could dream up something that does meet your needs. I take it there is no conventional keyboard on this device? What other I/O mechanisms are present?

**Msg#:13458**
From: Tom Eng To: Russ Reiss

Electrically, your suggestion is absolutely correct. The touch sensor is electrically equivalent to a mechanical switch. However, it all comes down to ergonomics. Based on user feedback on early prototype units, the switch has to be activated by very light pressure. The desirable level of pressure is about the pressure of a light touch, which is too light to activate any micro switches I've tested. The contact surface also has to conform to a unique contour not available in off-the-shelf switches. It may be possible to design and build such a switch, but it probably won't be inexpensive. Therefore, I am pursuing the touch sensor option. Since I already have to invest in designing a custom mixed-analog/digital chip, adding the necessary electronics on the same chip to replace a special mechanical switch is actually an inexpensive option.

# CONNECTIME

**Msg#:13879**

From: Russ Reiss To: Tom Eng

OK, but realize that some of the microminiature push-button switches-such as the Panasonic units available from Digi-Key—have quite light force requirements. Plus, membrane switches can be *very* sensitive. The most sensitive ones, however, can false more readily, and I guess operation at wide altitude ranges can be a problem.

As for the unique contour surface requirement, perhaps you should say some more since that would impact on what kind of touch sensor could be used, too! There is a variable-resistance membrane-type sensor that has been discussed in earlier threads on here (if you check back, you might find it). I had a sample around, but cannot find it now that I could use it. That might work, and a variety of interface possibilities exist, from simple digital/threshold types to an A/D input which you could fine-tune for user sensitivity.

A capacitive-type sensor could also work well. No actual pressure and, thus, inherent near-infinite life. Years ago someone made a chip that supported a matrix of such "buttons" which were simply etchings on the circuit board. Contours and such would not be a problem, I'd think. But I haven't seen or heard about that approach in a while. It's what was used in elevator controls for a 'long* while!

**Msg#:18216**

From: Tom Eng To: Russ Reiss

I've tested a large collection of Panasonic switches from Digi-Key and haven't found a suitable one. The special contour is a curved surface that wraps around the edge of the case. Etching a conductive pad on the case for cap sensing is a possible approach. I also vaguely remember seeing the cap-sensing IC you mentioned in the catalog of a small semiconductor company many years ago, but I can't recall who and when. I am investigating similar techniques involving capacitance sensing and a conductive pad. Any additional information will be helpful.

**Msg#:19123**

From: George Novacek To: Russ Reiss

I have spent many years designing consumer electronics. Finding an inexpensive, reliable *electronic* push button has been an unreacheable dream. It tore my heart to pay more for a simple switch than for many an ICs. We tried capacitive pickup, light sensing, heat sensing-you name it. Every method had a weakness. In the end, when faced with reliable, low-power operation, we always ended up with a mechanical switch.

If you need low mechanical force to switch, there are conductive foams you can use to connect pads on a PCB. There are also numerous manufacturers with all kinds of membrane as well as classic switches.

**Msg#:14995**

From: Lee Stoller To: Tom Eng

That bit about elevator controls reminded me of some elevator buttons in use years ago that responded to the heat of a finger. Maybe you could find some of that material for your application.

Another possibility for a light-touch switch would be for the finger to interrupt a light beam between an LED and a phototransistor. Might be too expensive, though, and require stuff that sticks up above the panel surface.

**Msg#:16067**

From: Richard Kanarek To: Lee Stoller

I think that "heat of the finger" stuff is an old wive's tale. As far as I know, all common touch switches operate on one of two principles:

1) Capacitance (your body's capacitance to ground stop's an otherwise running oscillator) or

2) AC hum pickup (your body acts like an antenna to pick up 60-Hz hum which is then detected). Considering that an elevator cab is a big metal box, choice 1 would seem ideal.

If anyone has definite info, do let me know!

**Msg#:17053**

From: Tom Eng To: Richard Kanarek

I tend to agree with your assessment. Detecting heat on the finger tip is a very tricky thing to do. I have experimented with the AC hum detection method. It actually worked reasonably well. My only concern is when the equipment is used in a "quiet" outdoor environment where the body cannot pick up enough electrical energy to activate the switch. I haven't verified the quiet environment performance because I cannot find such an electrically quiet place within a reasonable driving distance. I suspect that a pristine environment untainted by human presence and modern technology is so rare that the AC hum technique will work just fine.

I have begun an investigation into various capacitance-sensing techniques. Any information on cap sensing will be appreciated.

**Msg#:17790**

From: Lee Stoller To: Richard Kanarek

I cannot provide any definite information on the heat sensor story. All I know is that I was told not to use an elevator during a building fire, "because some elevators use heat-sensitive switches that will cause them to be called to the floor that the fire is on." This came at a police training class, and may or may not be accurate. There are good reasons not to use an elevator during a fire, even if the story is false.

# CONNECTIME

**Msg#:18634**
From: Richard Kanarek To: Lee **Stoller**

As you quite accurately point out, there are many good reasons not to use an elevator during a fire. For example, just because your building is on fire is no reason to miss an opportunity for exercise! <g>

The original post, however, dealt with touch-activated switches. Regardless of what an elevator does during a fire, I think it's safe to say that it does not do it because its touch-activated floor switches are actually heat activated.

As far as the elevator deliberately dispatching itself to a floor of a building that is on fire, this would hardly seem a very cleaver thing to do (of course I have no training in this area). Elevator cabs are a bit expensive, so ensuring that they will receive maximum damage possible during a fire wouldn't seem wise. Also, an elevator might be a handy thing to have during a fire-for transporting fire fighters, heavy equipment, and so forth. Setting the cab on fire would therefore seem counterproductive. And, of course, opening the doors to the elevator shaft on a floor that's on fire would seem to do little to help limit the spread of fire and smoke.

**Msg#:18796**
From: John Breitenbach To: Richard Kanarek

A number of years ago, Otis did, in fact, manufacture a touch button that was heat activated. These buttons were responsible for burning up a cab full of people during a fire. The car was called to the floor with the fire, and then the doors were held open because the switch was still activated. As far as I know, they don't manufacture those buttons anymore, or they built a thermal disconnect into them.

Elevator control systems actually have two special modes during a fire, but they need to be activated by firemen on site. The first is called "Phase I Fire Recall." When a keyswitch is activated by the firefighters, the elevator car stops without opening the doors and returns to the fire recall floor (typically the lobby floor) where it parks with the doors open. Then the firefighters can put the car into "Phase II Fire Service." This allows them to take the car to whatever floor they want. All the push buttons in the hallways are disconnected.

Once they get to a floor, local codes dictate such things as whether constant pressure or one touch of the door open and close buttons cycles the doors. Each state and city seems to have slight wrinkles on these requirement.

During either Phase I or Phase II, any reopening devices on the doors that can be affected by heat or smoke, such as electric eyes and light curtains, must be disconnected. Unfortunately, the elevator doesn't go into fire service until someone throws the switch. This means that if the car is not in fire service and stops at a smoke-filled floor, it is possible that a light curtain or electric eye on the door will keep the car at that floor. Most newer control systems have a feature called "nudging" which will force the doors to close, at reduced speed and torque, after a timeout.

Oh, by the way, I work for an elevator company (not Otis) and design electronic gizmos for elevators like light curtains, motor controls, and other goodies. I am currently designing a new elevator control system and just spent the last two weeks going over the fire service code to understand how it should work. It was interesting to see this thread pop up.

**Msg#:19735**
From: Ken Davidson To: John Breitenbach

> Unfortunately, the elevator doesn't go into fire
> service until someone throws the switch.

I'm not sure that's entirely true. We just had an elevator installed in our offices here about a year ago. Each floor has a smoke detector outside the elevator door. When a detector is tripped, the elevator automatically enters Phase I. I don't know if things are different in other areas of the country or for elevators with more than three floors.

**Msg#:20132**
From: Ralph Willing To: Ken Davidson

This is pretty much standard throughout the country. It's required by the ANSI standard (a17.1, I think) on elevators, which in turn is adopted by most building codes. That change came after the fire previously referred to trapped some folks in a car at a fire floor.

**Msg#:21212**
From: John Breitenbach To: Ken Davidson

You're right, my mistake. Smoke detectors can set off Phase I recall; they are also used to send the car to an alternate recall floor if the detector on the main floor has been tripped. As soon as I hit the upload key I knew it didn't sound right.. . .

**Msg#:23195**
From: Richard Kanarek To: John Breitenbach

I've always had an interest in things elevator related, but no opportunity to exercise that interest. I shall keep your presence in mind!

So, what principle is used for those buttons that light when touched? Are they thermal, capacitive, or something else?

# CONNECTIME

**Msg#:25367**
From: John Breitenbach To: Richard Kanarek

Touch buttons for elevators have been made using both capacitive and heat-sensing elements in the past. The newer ones are piezo. The bulk of the buttons out there in the field, though, are still mechanical. People like the way they feel; I know I tend to prefer the feel of the mechanical ones.

Unfortunately, there are no standards in our industry for voltages used by the control system, as least not for fixtures. This presents a real problem when you try to manufacture an electronic push button that will work with any control system. On the switch side, it needs to be compatible with supply voltages of anywhere from 24 to 600 V, AC or DC. The same is also true of the lighting in the push button: it needs to work with all those voltages as well.

Most manufacturers will make their buttons in a number of different flavors, which means that you've got to order and stock fixtures for specific buildings based on the voltage of the controller in that building. We've stuck with making only mechanical push buttons-their contacts arc suitable for any voltage. We've also created a small power supply circuit which accepts from 12 to 300 V, AC or DC, and provides power to drive an ultrabright LED behind the button. This gives us a "universal" push button that you can use in any building.

As more and more control systems switch over from relay logic to microprocessor based, we hope to drive the controller voltages down. Sometimes it's frustrating to make a neat, tiny circuit only to be told that you've got to put a clunky 10-A cube relay on your board so it will work with a 30-year-old control system. Ah well.

**Msg#:30688**
From: Tom Eng To: John Breitenbach

Do elevator manufacturers design and build their own touch switches? Are there any companies specialized in touch-switch technology for OEMs? Do you have any details on the principle of operation for the various sensing techniques you mentioned?

**CADPAC**

PCS AND SCHEMATIC TOOLS
2 for the price of 1
PCB II & SUPERSKETCH FOR NEW LOW LOW PRICE

** features comparable to packages costing thousands!
** must be tried to be believed
**"easiest product to use for designing PCBs"
** customers call it "the 8th wonder of the world!"

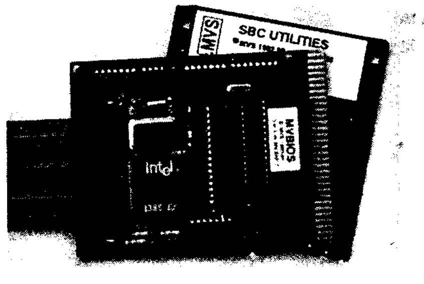
**!!REDUCED!!**

R4 SYSTEMS INC.
1100 GORHAM ST. SUITE 11B-332
NEWMARKET ONTARIO
CANADA L3Y 7V1

FREE DEMO
WRITE OR CALL
TODAY

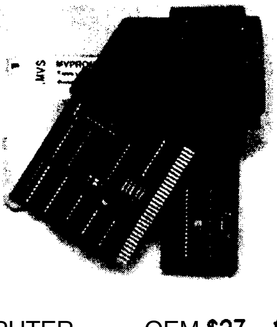
905 898-0665 FAX 905 898-0683
Email r4system@astral.magic.ca

**386 SBC $83** OEM (1K) PRICE
INCLUDES:
- 5 SER (8250 USART)
- 3 PAR (32 BITS MAX)
- 32K RAM, EXP 64M
- STANDARD PC BUS
- LCD, KBD PORT
- BATT. BACK. RTC
- IRQ0-15 (6259 X2)
- 0237 DMA 6253 TMR
- BUILT-IN LED DISP.
- UP TO 6 MEG ROM
- CMOS NVRAM



USE TURBO C, BASIC, MASM
RUNS DOS AND WINDOWS
EVAL KIT $295

**$95** SINGLE PIECE PRICE
**UNIVERSAL PROGRAMMER**



- DOES 6 MEG EPROMS
- CMOS, EE, FLASH, NVRAM
- EASIER TO USE THAN MOST
- POWERFUL SCRIPT ABILITY
- MICROCONT. ADAPTERS
- PLCC. MINI-DIP ADAPTERS
- SUPER FAST ALGORITHMS

OTHER PRODUCTS:

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

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

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

#128

# CONNECTIME

**Msg#:31237**
**From: John Breitenbach To: Tom Eng**

I would guess that some of the manufacturers do build their own. Otis, as a division of United Technologies, has incredible in-house manufacturing capabilities. As far as switches for OEMs go, C&K makes a piezo-based switch. There is also a company called Wilson-Hurd which sandwiches piezo material behind nice stainless or aluminum panels. You end up with an indestructible front panel which is environmentally sealed and really attractive, with no moving parts.

If you want to roll your own, there was a good article in *Circuit Cellar INK* a while back on Kynar piezo film made by AMP. You can get a design kit for the Kynar film which includes all the theory you want and some neat samples of the film.

## Integer square root

**Msg#:24937**
**From: David Smith To: All Users**

Does anyone know of a fast integer-square-root routine? I've seen a slick routine somewhere, but now I can't find it. I know of the iterative loop to calculate square roots, but there is a faster method. Thanks.

**Msg#:27013**
**From: Ed Nisley To: David Smith**

This is gonna make you sick, but you can do it by binary searching. If your integers are all less than 32K (thus, fitting neatly into 15 bits), you know the square root is less than 182 decimal (just over 7 bits). If your CPU has an 8x8 multiply giving a 16-bit result, you can find the answer in, at most, eight trials.

Might be worth running a few benchmarks....

**Msg#:32807**
**From: David Smith To: Ed Nisley**

Thanks for the idea. I never thought to use a hash table. I wish my brain cells worked like a look-up table so I could remember where I saw that fast algorithm. Couldn't one use shifts and compares to get a good first seed for the iterative algorithm? I'll have to look into it and compare the speed to the hash table. Also, code space is limited.

**Msg#:29468**
**From: Calvin Krusen To: David Smith**

When you say "integer" square root, are you referring to the number which you desire the square of or the resultant square root? If you know the minimum and maximum values, you may be able to get away with a look-up table. Look-up tables are *very quick*, but require *lots' of memory.

For integer results from integers in the range of 0–65535, 64 KB of storage would be required (sqrt(65535) < 256). For floating-point results, you would need 256 KB (assuming using the 4-byte float variable), or limit your range to O-16383 for a 64-KB array of floats. Once you've got all the values, put them in order in an array.

The following is a C example for the integer results:

```
const unsigned char sqrt_table[65536]=
   {
   0,1,1,2,2,2,...,4, /* sqrts for 0-15 */
   4,4,...,6,         /* sqrts for 16-31 */

   255,...,255        /* for 65520-65535 */
   };
```

To "call" the "function," simply refer to the element you want. For example:

```
unsigned int result, number=18;
  /* the 18th element (value 4) is */
  /* be stored in result */
 result = sqrt_table[number];
```

**Msg#:29650**
**From: Herbert Nowell To: Calvin Krusen**

Why would you need 64 KB for an integer-result look-up table? Off the top of my head, you could make it much shorter by just having it look up by the lowest value that gives a given answer.

For example:

| In | Out |
|---|---|
| 1 | 1 |
| 3 | 2 |
| 7 | 3 |
| 12 | 4 |

and so on (rounding may be wrong on the above table).

Then just search the table for the last entry equal to or less than your value to find its integer square root.

**Msg#:33455**
**From: Jeff Pipkins To: David Smith**

Here's a method that is very simple to code and uses only a small amount of code space and RAM space. It's not very time efficient in its raw form, but there are ways to improve that considerably.

The integer square root of an integer is equal to the number of odd numbers that can be subtracted from it.

As an example, let's find the integer square root of 28: 28–1=27; 27–3=24; 24–5=19; 19–7=12; 12–9=3; 3–11=stop, number is negative.

Since five subtractions were successfully completed, the integer square root of 28 is 5.

**Msg#:33581**
From: Russ Reiss To: Jeff Pipkins

I'm glad you remembered that! It's been haunting me since the fellow asked! I once knew it, but for the life of me could not remember what it was.

**Msg#:34630**
From: lee **Stoller** To: Jeff Pipkins

Who would have thought of that one? I'd love to see a proof of it..

*We invite you to call the Circuit Cellar BBS and exchange messages and files with other Circuit Cellar readers. It is available 24 hours a day and may be reached at (860) 871-1988. Set your modem for 8 data bits, 1 stop bit, no parity, and 300, 1200, 2400, 9600, or 14.4k bps. For information on obtaining article software through the Internet, send E-mail to info@circellar.com.*

## ARTICLE SOFTWARE

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

To order Software on Disk, send check or money order to: Circuit Cellar INK, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your Visa or Master-Card and call (860) 875-2199. Be sure to specify the issue number of each disk you order. Please add $3 for shipping outside the U.S.

434 Very Useful 435 Moderately Useful 436 Not Useful

## A NOTICE TO OUR READERS:

*Circuit Cellar INK*® will occasionally provide a listing of subscribers to vendors with offers of substantial interest to our readers. If you would prefer not to be part of this listing, send the mailing label from the front cover of *Circuit Cellar INK*® along with your request to:

**Circuit Cellar INK**
**Subscriber Service Department**
**P.O. Box 7694**
**Riverton, NJ 08077**

# STEVE'S OWN INK

## Now it's Lollipops

A few times in past editorials, I've written about the continuing influence of embedded controllers. That, combined with the fact that I publish a magazine promoting embedded control might make you think I believe in its universal application as well. Believe it or not, *even* I have my limits.

When I'm not trying to diet away the unfavorable effects of being a closet gourmet cook, I like finding new restaurants. Recently, I stopped into a Chinese place to check out the cuisine. In the lobby they had a lollipop machine. Now, ordinarily I don't get too impassioned over lollipops, but this machine was electronic, not mechanical. Instead of the typical inverted fishbowl with coin slot and mechanical slide lever that we all instantly recognize, this new device looked like the result of assigning a programmer and an electrical engineer to design a lollipop machine. The contraption had a striking contemporary appearance: Clear acrylic plastic was glued together to form a connected yet logically segmented architecture. The lollipops were in a rotating acrylic carousel (made of pink plastic) in one of the cubes while the coin-receiving mechanism and change holder were in another. Additional cubed compartments contained various electronic boards and batteries.

With it presenting a perfect example of excessive technique applied to a simple task, I studied the mechanism further. Inserting the correct money into the coin mechanism looked like it should cause a "dispense lollipop" signal which activates a solenoid to drop the money into the change holder. From there, the signal went into a wiring maze of three circuit boards. One board contained the processor and program control logic. A second board seemed to be primarily power supply. There was no AC power source and it seemed strange to me that the entire device was powered by two 9-V batteries. The third board was a stepper motor controller! Yep, the carousel used a stepper motor to rotate it around to the dispensing slot and drop the lollipop. Cute.

Not that I needed a lollipop with my Chinese food, but I felt I had to witness this extraordinary exhibition of embedded overkill in action. As I reached to insert a quarter, I noted for the first time a little sign above the slot: Out Of Order!

Replacing a reliable $100 mechanical device that has persevered for so long with a $2000 electronic unit that can't even spit out a simple lollipop seems silly. Of course, the high-tech solution (from the same people who bring you electronic lollipop machines) is to add a modem so the machine can call for maintenance. Argh!

I know I often advocate embedded control as a challenging solution for many problems. However, when it comes to actually applying computers to replace tried-and-true mechanical controls, it should be done such that the price/performance and cost/advantage ratios are to your benefit. With that in mind, "Keep it simple, Stupid!"

Steve

P.S.: By popular demand, we've decided to offer subscribers free HCS II controller boards again. See "Play it again, Steve" *on* page 97.