CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

BUILDING AUTOMATION

SPECIAL SECTION:
Embedded Signal Conditioning

New Circuit Cellar HCS

Linear Optoisolator Design

Two-Way X-10 Interface

February/March, 1992 — Issue 25

$3.95 U.S.
$4.95 Canada

EDITOR'S INK

Change is Not in the Cards

One of my favorite lines from a movie was uttered by Dr. McCoy in *Star Trek: The Motion Picture*. "I know engineers. They love to change things."

By now, I'm sure you've noticed some changes in our look. Our last major redesign was over two years ago, and really didn't involve more than the cover and some minor details inside. As we enter our fifth year of publication, a fresh look is in order to spice up our presentation a bit.

Rest assured our content hasn't changed (and won't change anytime soon). We engineers may like to change things, but only if the change is for the better. We're having too much fun the way things are.

A BUILDING AUTOMATION EXTRAVAGANZA

What better way to kick off our fifth year than with an issue jam-packed full with home automation articles. Always a popular topic with our readers, this issue is also our largest so far.

The cornerstone of the issue is a tale of home automation gone awry, written by perhaps the most famous home automatist, Steve Ciarcia, where he also introduces the new Circuit Cellar Home Control System II. Other articles detailing system building blocks are also presented by Ed Nisley and me.

You might start to wonder, if we're spending so much time covering technologies such as CEBus, LONs, and SMART HOUSE, why are we suddenly designing our own home control system? You have to be careful not to get the two types of systems confused.

Systems such as CEBus and SMART HOUSE are actually home networks. They provide communication media and protocols, but don't really deal with the issue of control. Small CEBus installations may be able to be done with no central controller. However, we feel in most installations, some central controller will still be necessary.

The HCS II addresses primarily the control side of a home automation installation. We use X- 10 for power line and RS-485 for twisted pair, but only because those are network technologies that are available at reasonable cost today. When CEBus devices start showing up on store shelves, we'll be able to connect a CEBus interface to the supervisory controller and add CEBus to the list of supported networks (and indeed replace X-10 and RS-485).

So don't think we're a bunch of rebels bucking the trends you see unfolding in the news, on TV, and even within these pages. We're simply using today's technology so we can be ready when tomorrow's finally shows up.

Ken

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

MANAGING EDITOR
Ken Davidson

ASSOCIATE EDITOR
Lisa Nadile

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

CONTRIBUTING EDITORS
Tom Cantrell 8 Chris Ciarcia

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
Lisa Ferry

PRODUCTION MANAGER
Mark Vereb

STAFF RESEARCHERS:
Northeast
John Dybowski
Midwest
Jon Elson &Tim McDonough
West Coast
Frank Kuechmann

Cover Illustration by Robert Tinney

PUBLISHER
Daniel Rodrigues

PUBLISHER'S ASSISTANT
Susan McGill

CIRCULATION COORDINATOR
Rose Mansella

CIRCULATION ASSISTANT
Barbara Maleski

CIRCULATION CONSULTANT
Gregory Spitzfaden

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Dan Gorsky

CIRCUIT CELLAR INK (ISSN 0896-8985) is published bimonthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (203) 875-2751. Second-class postage paid at Vernon, CT and additional offices. One-year (6 issues) subscription rate U.S.A. and possessions $17.95, Canada/Mexico $21.95, all other countries $32.95. All subscription orders payable in U.S. funds only, via international postal money order or check drawn on U.S. bank. Direct subscription orders to Circuit Cellar INK, Subscriptions, P.O. Box 3050-C, Southeastern, PA 19398 or cdl (215) 630-1914.

POSTMASTER: Please send address changes to Circuit Cellar INK, Circulation Dept., P.O. Box 3050-C, Southeastern, PA19399.

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

INSIDE ISSUE 25

SPECIAL SECTION Embedded Signal Conditioning

READER'S INK

Don't Blame the Tool

In response to Thomas M. Nathe's letter, "What About The Schematic Reader?" in ***Circuit Cellar INK,*** issue #24:

1) Signal Names: OrCAD allows naming of signal paths. Consistency is up to the layout engineer.

2) Power and Grounds: OrCAD ***gives*** the user both the choice of where to place Power/Ground pins and whether or not they are visible. Power and Ground pins may be placed on the part. Use of the Line-Draw and Label functions allows one to make a Power/Ground List if desired.

3) Block Diagrams: OrCAD allows the building of Hierarchical Schematics. Thus, page one may be a block diagram of the device in its simplest form. Page two would be a breakdown of one of the blocks on page one (this could be a block diagram, a schematic or both). Labels can identify signal paths. The entire schematic may be handled this way.

4) Programmable Gate Arrays: Use of the Line-Draw and Label functions allows one to make a list. Or a printout of the PGA functionality could be included.

5) Signal Flow: OrCAD supports intersheet signals with Module-Ports. These consist of user-defined text inside a box. One or both ends of the box may be pointed to indicate signal flow.

6) Part Locations: This is entirely in the hands of the layout engineer.

7) Parts Lists: OrCAD supports Parts Lists, of course. Also, each part has eight separate fields. These may be used for such items as Supplier Information, Part Substitution Data, Part Tolerance, and so on. Each of these fields may be made visible or invisible.

I have been using OrCAD for all my schematic work for the past four years. I have used all the features mentioned here. They are neither difficult to learn nor difficult to use. Despite its imperfections (yes, OrCAD does have some), "ease of use" and "maximum user choice" have kept me a loyal fan. Thus, I have some trouble with your rating OrCAD a "2 out of 10." I do agree that the seven points of yours do need to be addressed by layout personnel. I too have been forced to "thrash" through unreadable schematics.

If your layout people are using a recent version of OrCAD, they have the power to produce readable, informative schematics. It sounds to me like they are blaming the tool for their inability/unwillingness to use it fully. Reminds me of the story of the screechingly awful violin player, who was convinced he could make beautiful music if only he had a Stradivarius.

Allen K. Milakovic, Glen Rock, PA

'Round and 'Round He Goes

Boy, I'm really getting dizzy. I've gotten on and off the "C merry-go-round" a few times now.

I wanted to learn to use a C compiler for the 8051 microcontroller. I heard many times that C is the way to go, so I enrolled in a night-school course at a local college to learn C programming for the PC, just to get a taste of it. As it turned out, I found that I had some difficulty grasping the cryptic mannerisms of C.

I might add that I am usually a determined self-starter-type person who normally doesn't need to be led by the hand. The talk about C is that it is sort of tough to learn, so I thought taking the course would be a good primer. Instead, it left a bad taste in my mouth. It also left me with the belief that C is ideally suited for database-type programs and should be left to programmers in the business world.

So I jump off the ride.

Keeping in mind that "C is the way to go," I thought to myself that perhaps C for embedded controllers might be easier to learn since I have a reasonable understanding of the hardware and assembly programming.

I jump back on the ride.

Then I read an article where a comparison of a few C compilers was made. Wow! The price! At $600 and up, that's prohibitive enough to keep me programming in assembler. It's only a hobby.

I jump back off the ride.

Then I get all pumped up when I see Ed's "Firmware Furnace" in issue #23 titled "Restarting C." "That sounds like my disposition," I say to myself. (Well, maybe even not quite restarting it.) I read, "If you don't learn C, you won't be a hit at the party." Ed also mentions an affordable C cross-compiler. I really should give this a shot.

I jump back on the ride.

As I continued reading the article, I wondered what the C is really doing other than calling assembly routines or performing simple tasks that would easily be accomplished by assembly code. I try to convince myself to learn it anyway.

Well, I just received issue #24 and T.B. Kester's letter to the editor ("Taking Ed to Task") had me jump off the "C merry-go-round."

Wait a minute. "A" is for Assembler, "C" is for...I don't want to talk about it. Hey! I missed "B." "B" is for BASIC like in BASIC-52. I think I'll give that some attention. What the heck! It's only overhead...right C?

Hugh Duff, Toronto, Canada

Design Contest Winner Fan

Your recent issue *(Circuit Cellar INK,* issue #24) was a pleasure, again. However, there was something that really caught my eye: the second prize winner in the open category of the Design Contest.

It was not the device design that fascinated me so much as the attractive package it was in. Although noted by the judges, I still think it was not emphasized enough. If this was a one-off job, the enclosure was really terrific!

How about some articles on that subject? Or at least a note as to how Sanjaya did do the packaging of his project? This is really a nontrivial matter for us nonartistic types. For example, quite often I have to come up with a prototype or a proof-of-concept device. It works right, but with the looks of it... . And nice packaging shows a design off as much as its functionality.

Robert Walker, West Orange, NJ

Our Design Contest winners always generate a great deal of interest from our readers, in phone calls, letters, and BBS messages. A point we tried to make in the introduction, but apparently not succinctly enough, is most of the design contest winners, and many of the entrants that didn't win, will be writing complete articles about their projects. The articles will be appearing in the pages of upcoming issues of The Computer Applications Journal *and in future volumes of "The Circuit Cellar Project File." Sanjaya's EPROM emulator is slated for the next volume of the project file, as is Alan Rauscher's XYZ drilling table (about which we've had many calls).*

I also want to stress that, in the interest of our entrants'privacy, we won't give out their addresses or phone numbers unless they've given us permission to do so. If you're looking for more information about a project and can't wait for it to be published, write a letter to the designer and send it to us here at the magazine. We'll be sure to forward it.-Editor

Correction

In issue #24 [December '9 1, January '92), "Firmware Furnace," the EPROM data lines (00-07) should be connected to the other side of the 74LS373 (U2) in Figure 3.

We Want to Hear from You

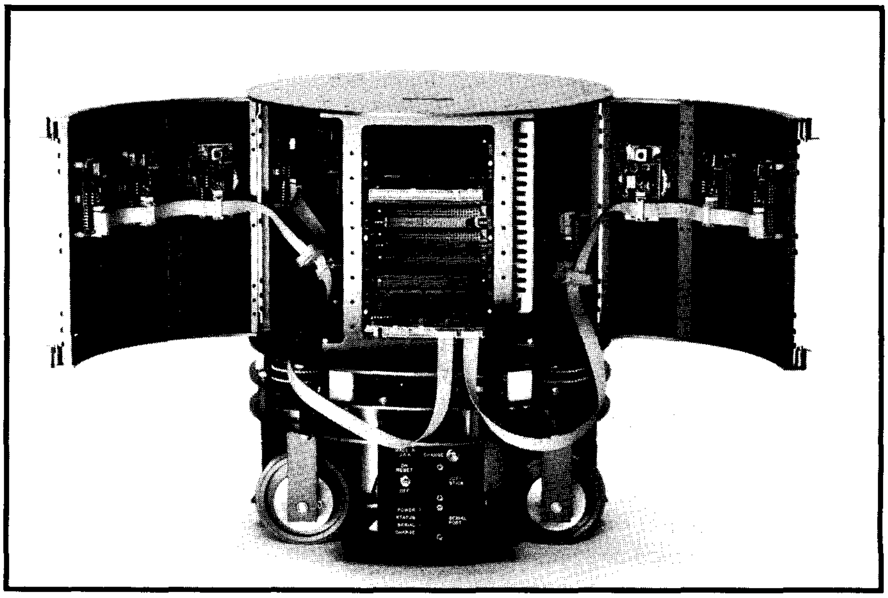
Send letters of praise, condemnation, or suggestion to the editors of The *Computer Applications Journal.*

The Computer Applications Journal
letters to the Editor
4 Park Street
Vernon, CT 06066

NEW PRODUCT NEWS

Edited by Harv Weiner

MOBILE ROBOT



Home automation reaches new levels with a self-contained interactive mobile robot base from Real World Interface Inc. The B 12 Series is a modular system, available with off-the-shelf components to speed research projects. The B12 Mobile **Robot** contains motors, power amplifiers and an on-board control computer and software in a rugged aluminum frame. The base measures 6.75" high by 12" diameter, weighs only 25 pounds, and can carry a payload up to 40 pounds. Its small size alleviates the inconvenience and potential hazards of larger robots, while its low center of gravity allows it to climb steep inclines and travel at speeds of up to 6 feet per second.

The B12 Robot significantly reduces prototyping time and cost of mobile robotic systems because of its size and ability to conform to a variety of application requirements. The programming environment of the B12's control software uses simple mnemonic language that translates through a standard RS-232 serial port into precise real-time positioning. Synchro drive mechanics allow each servo loop to be operated in Velocity, Position, Power, and Limp modes. Each servo mode can be changed at will, even when the B12 is in motion.

A total access Development Enclosure with card cage, power supply system, and 12-slot backplane is available, and it accommodates various compatible computer boards as well as sonar drives and transducers. The standard enclosure measures 8" high by 12" diameter and features 48 potential sensor sites on four interchangeable door frames. A 68000 computer card is available for both on-board control and sensor processing.

Real World Interface, Inc.
Main Street • P.O. Box 270
Dublin, NH 03444 • (603) 563-8871 **#500**

DATA ACQUISITION SYSTEM USES PARALLEL PORT

DAS Inc. announces a low-cost data acquisition system that uses the parallel port of a standard personal computer parallel port. **Easy Data** eliminates the need and fuss of screwdrivers, jumpers, and interrupts typically required for data acquisition hardware by providing an external unit that plugs directly into the parallel port. The hardware and software supplied with the system is user friendly, requiring only a few minutes of setup and familiarization time before acquiring data.

Performance specifications for the Easy Data unit rival those of systems costing two or three times as much. The unit can acquire up to 16 signals simultaneously with a nominal single input throughput of 4 kHz. Input polarity is software selectable for unipolar or bipolar operation, with a measurable voltage from -10 volts to +10 volts. Software-selectable gains are in steps of 1, 10, 100, or 1000.

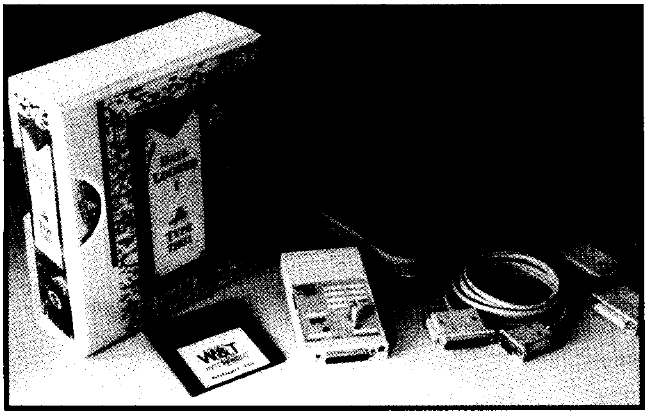
As an added feature, the Easy Data unit has a single analog output capable of producing pulsed, square, ramp, or triangular waves, giving it many of the features found only in expensive computer-controlled signal generators.

The Easy Data unit sells for $269 and includes the graphics oriented software and a 30-day money back guarantee.

DAS, Inc.
1034 Industrial Park Road
Orem, UT 84057
(801) 224-8080
Fax: (801) 224-8087

#501

NEW PRODUCT NEWS



VERSATILE LOW-COST DATA LOGGER

Obtain a low-cost approach to real-time data capture with a novel Data Logger system from W&T Products Corp. Traditional data loggers usually involve the wiring of numerous terminals and sensors, but the W&T Data Logger are installed only where needed and without wires.

The calculator-sized unit operates approximately 200 days on a disposable 6-volt lithium battery, and features memory sizes from 32K to 5 12K bytes. It records up to eight analog values, and captures changes in the switching state of up to eight internal switches (external switches can be connected) and an external voltage. Every measurement is automatically given a serial number, and a marker can be assigned. Internal data compression ensures only changes in measured values take memory space, so the RAM usage is economically managed.

The Data Logger includes two sliding switches, one rotary switch (with 16 positions] and two push buttons that can be used for any application. An internal, magnetically activated reed switch, with external magnet is also included. The Data Logger accepts any voltage between 6 and 48 VDC or AC (with optional voltage input) and stores every key press and every change in switch status along with the relevant time.

The Data Logger sells for $299 for the 32K-byte model and $419 for the 128K-byte model. The price of the 512K-byte model is available on request. Each unit is supplied with a universal software package, serial connecting cables for an IBM PC, and complete documentation. Also available are a variety of plug-ready accessories, including a footswitch, a passive infrared detector, vibration sensors, humidity sensors, and temperature sensors.

W&T Products Corp.
2209 NE 54th Street . Ft. Lauderdale, FL 33308
(800) 628-2086 . Fax: (305) 351-9099

#502

CEBUS POWER LINE INTERFACE PRODUCTS

Intellon is pleased to announce the availability of three products incorporating its revolutionary Spread Spectrum Carrier technology for use with the Electronic Industries Association's CEBus: the SSC PLCE power line modem IC, the SSC PLCEMO power line modem board, and the SSC EVS-PL evaluation system.

The SSC PLCE is Intellon's power line modem integrated circuit. The SSC PLCE may be used to bring low-cost power line communications to a broad range of products. It is available in sample packs of six ICs as part number SSC PLCEFNPAK for $300. Production quantities are priced at less than $5 (25,000).

The SSC PLCEMO is a board-level product featuring the SSC PLCE, transmit amplifier, and coupling circuit for 110/220-VAC, 50/60-Hz power lines. The SSC PLCEMO is configured for operation in the 100-kHz to 400-kHz range to allow development of networks utilizing the CEBus protocol. The SSC PLCEMO is available in sample packs of two modem boards for $500 as part number SSC PLCEMOPAK. The SSC PLCEMO is priced at $105 each in quantities of 10.

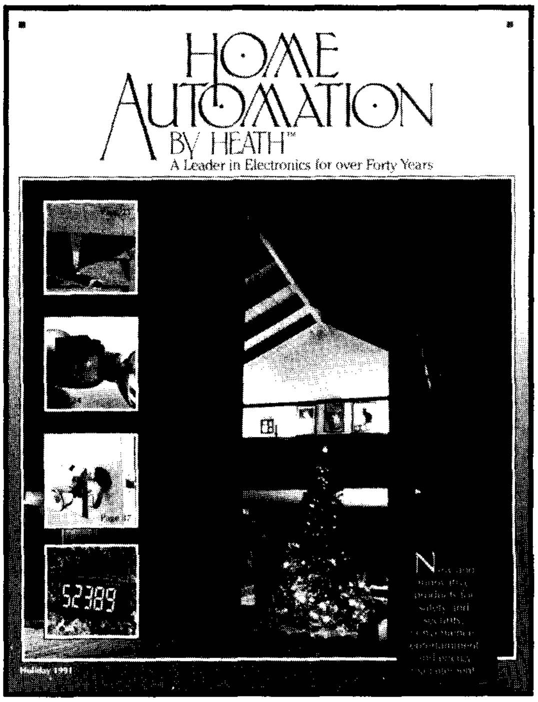
The SSC EVS-PL is a complete system for evaluating the performance of CEBus power line networks. The system includes three complete CEBus nodes and control software for an IBM PC/AT or compatible. The system is available for $3495. Additional nodes for the evaluation system are available for $995.

Intellon Corporation
5150 West Highway 40
Ocala, FL 32675
(904) 237-7416
Fax: (904) 237-7616

#503

NEW PRODUCT NEWS

HOME AUTOMATION CATALOG



Exciting, innovative electronic products for the home, yard, and automobile are featured in Heath Company's latest Home Automation by Heath Catalog.

Home Automation by Heath contains a wide variety of consumer products designed for safety, security, convenience, entertainment, and energy management. These products are designed to create a home environment that is a safe, pleasant, convenient, easy to manage, and energy efficient.

The newest issue of Home Automation by Heath is a 40-page, full-color catalog aimed at the do-it-yourself home enthusiast as well as the electronics innovator. The Holiday issue introduces many exciting new products, such as drape controllers, air cleaners, gas detectors, and wireless light switches. Use the electronic drape controller to control drapes by a handheld remote, programming them to open or close at predetermined times throughout the day. The catalog features three new air cleaners that electronically deep-clean the air. The gas detector sounds a powerful alarm in case of a poisonous or explosive gas leak. And for the homeowner, the wireless add-on switch is perfect for stairways, long hallways, or large rooms needing a light switch at both ends.

Other products offered in the catalog include whole house automation and security systems, motion-sensing indoor and outdoor lighting controls, timers, security cameras, wireless video broadcasters, energy-saving thermostats, and much more.

Heath Company
Dept. 350-058 • Benton Harbor, MI 49022
(800) 444-3284

#504

THE ULTIMATE SPACE SAVER

Computer users who lack desktop space will appreciate the new computer announced by LinkSys Corp. The **KeyboardPC** is a complete 386SX computer built into a standard 101-style keyboard. The system operates either as a stand-alone unit or as a local area network workstation, and it can be configured with an 80386SX 16-/20-/25-MHz or 80286 12-/16-MHz CPU.

The system is appreciably smaller than any standard computer and offers virtually the same power, memory, and expansion capabilities. The standard system includes an on-board CPU, 1 MB DRAM (expandable to 16 MB) with autosensing memory expansion, and an AMI keyboard and main system BIOS. It also uses a 1024 x 768 super VGA interface, an IDE disk controller, a built-in 16-bit LAN card slot, and a PS/2 mouse.

Standard internal ports include one game, two serial, and three parallel, with external ports for bar code reader or scanner and floppy/tape drives.

Some of the optional features available are an 80287 or 80387 math coprocessor, 32K cache buffer, internal floppy and (20, 40, and 60 MB) hard drives, external floppy or tape drives, 2400-bps Hayes-compatible modem, and a LAN card. The choice of having the system shipped with standard disk drives or without the ability to add drives is one unique feature.

The KeyboardPC weighs just 9.3 pounds fully disked, and its overall dimensions are 19.5"L x 9.5"W x 2.75"H (rear), 1.25% (front). The unit is shipped in an attractive suitcase-style cardboard container.

Suggested list price for the system starts at **$1095** and optional features are priced separately. The system comes with a 3-year parts and labor warranty.

LinkSys Corp.
P.O. Box 18558
Irvine, CA 92713
(714) 261-1288
Fax: (714) 261-8868

#505

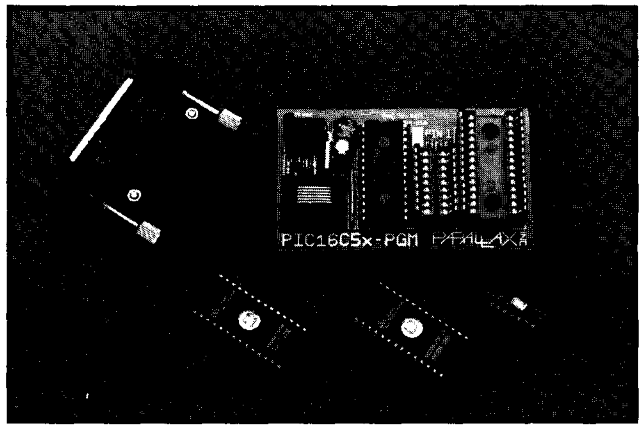
NEW PRODUCT NEWS

LOW-COST MICROCONTROLLERS

A series of versatile, low-cost microcontrollers is available from Parallax. The PIC16C5x series is a family of eight-bit CMOS devices that combine EPROM technology and a fast CPU with bit and byte addressing for all I/O pins and registers. The PIC family of controllers features 18- and 28-pin packages, 12 or 20 tristate I/O lines, 512–2048-word EPROM, a 3-microamp sleep mode, and DC-to-8-MHz clock.

A comprehensive set of PC-based tools are also available and include an assembler, a programmer, and an emulator.

The price of the PIC microcontrollers ranges from $3 to $5 in one-time-programmable packages, and from $17 to $26 in erasable packages. The PIC programmer sells for $199, the emulator for $299, and the programmer and emulator for $449. The assembler is included with the hardware.



Parallax, Inc.
6200 Desimone Lane, #69A . Citrus Heights, CA 95621
(916) 721-8217 . Fax: (916) 726-1905

#506

MICROCONTROLLER FEATURES ON-BOARD DSP

Zilog announces the industry's first 8-bit microcontroller with a 16-bit digital signal processor (DSP) on a single chip, the Z86C94. The addition of DSP to the company's Z8 microcontroller permits closed-loop servo functions to be executed digitally, reducing cost and eliminating noise and reliability problems associated with analog servos.

The device also features extremely fast 8-bit analog-to-digital and digital-to-analog converters, a serial peripheral interface, and a single-channel pulse-width modulator. These features make it ideally suited for 2.5-inch and 1.8-inch disk drives as well as tape drives, voice/data processing, and sophisticated automotive and consumer electronics.

The on-chip DSP operates as a slave processor to the Z8 and is used in applications requiring extensive math calculations. When executed from DSP program RAM, it is capable of 16 x 16-bit multiplication and accumulation in one clock cycle, or less than 100 ns with a 24-MHz system clock. The 8-channel, 8-bit A/D converter is a half-flash converter with a maximum conversion time of 1.7 µs. The 8-bit D/A converter has a settling time of 3 µs and a 4-bit digitally controlled gain stage.

The Z86C93 Z8 core, used in the Z86C94, contains three 16-bit counter/timers, two with a 6-bit prescaler and one with a 2-bit prescaler to improve timing flexibility. One of the counter/timers has five capture and three compare registers. The Z86C93 math unit is retained on the Z86C94 and allows hardwired 16-bit multiplication and division at lower speeds for less time critical math functions.

The Z8 also includes 24 I/O lines, up to 64K bytes of addressable external program space, a 256-byte register file, and 236 general-purpose registers. The on-chip oscillator accepts a crystal or external clock drive. A UART is included for asynchronous serial communications and an industry-standard serial peripheral interface is included for synchronous serial communications. The Z86C94 has three power-down modes for power-sensitive applications, such as small-form-factor disk drives in notebook computers. It also supports both multiplexed and demultiplexed address/data buses.

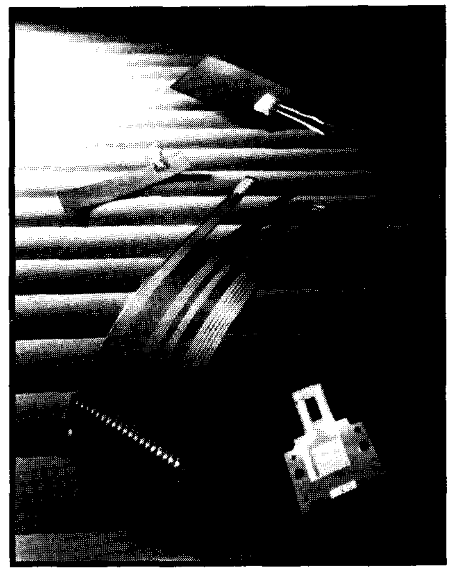
The Z86C94 sells for $15 in 1000-piece quantities.

Zilog, Inc.
210 E.HaciendaAve.
Campbell, CA 95008-6600
(408) 370-8000
Fax: (408) 370-8056

#507

NEW PRODUCT NEWS

FLEXIBLE TEMPERATURE SENSORS



Elmwood Sensors Inc. introduces a line of flexible temperature sensors that make measuring the temperature of curved, cylindrical, and conical surfaces easy. Flexible Temperature Sensors conform to a variety of surfaces and provide complete surface-to-surface contact. Their ability to sense the temperature of an entire surface area, rather than a single line or point contact, ensures a fast, reliable response to surface temperature changes.

The sensor achieves flexibility through the use of either an etched foil or wire-wound sensing element. The element is encapsulated between layers of flexible, moisture resistant, dielectric materials, such as Kapton *or* silicone rubber. The sensors can also be manufactured on transparent film dielectrics, such as polyester.

Pressure-sensitive adhesive allows for press-in-place installation and improves performance by eliminating the need for thermal grease or epoxy.

The sensing elements have a temperature range of 328°F to +455°F and are available in platinum, nickel, or nickel-iron with various temperature coefficients of resistance. The three-wire lead configuration negates lead wire resistance. A price was not available at press time.

Elmwood Sensors, Inc.
500 Narragansett Park Drive • Pawtucket, RI 02861-4325
(401) 727-1300 • Fax: (401) 728-5390

#508

FEATURES

The State of Home Systems

Where are we headed with home automation? Find out what one of the industry's leading marketing experts thinks is coming down the pike.

Tricia Parks

Home automation is an effect rather than a product: it is a major benefit expected from the proliferation of advanced systems and products in the home capable of sharing control and data messages via home networks. Ideally, and over time, our homes will adapt to the way we live in them while we live in them. Of course, this adaption requires expert system software that can't be developed in any sophisticated manner until networks exist for testing and evolution.

CEBus, detailed in ***Circuit Cellar INK*** issues 10, 15, and 21, is close to release. What remains is a bit more debate over whether adequate levels of testing have occurred for the power line spread spectrum solution. Echelon's LonWorks is being used by several companies for product development. SMART HOUSE, a full system for new construction, will roll out in 1992. The message behind these statements, regardless of one's preference, is home networks are a reality. The '90s will contain the proliferation of networked products in the home.

Indeed, the market is not waiting, although it is Parks Associates' view that no broad-based market can occur for central controllers until home owners have both more awareness and more comfort with these systems. However, there are companies who do make such offerings currently. Unity Systems of California has emerged as the leader in central home automation systems with its Home Manager, its acquisition of Hypertek's Home Brain technology, and, to be blunt, its survival in a tough early market.

Home Automation Inc. is growing. It markets its system primarily as one for security that offers extra benefits beyond burglar alert.

X- 10 (USA) Inc. controllers continue to proliferate and offer simple benefits, such as automatic lighting and timed responses from appliances including coffee makers and stereos.

Consumer Trends		Consumer Actions		Consumer Purchases
Out-of-work middle (streamlining of corporate America) High growth in telecommunications (corporate/government push) Desire to spend more time with family out of traffic	→	Work at home	→	*Office equipment *Telecommunications control equipment -Multiple lines
Increase in *dual-income families •single-parent families	→	*Convenience •Time savers •Money savers	→	*Integrated user interfaces *Premium on easy-to-use or automated systems *Diagnostics and PM programs
Fear/anxiety about "outside world" -crime •AIDS *pollution	→	Leisure time at home ("armored cocoon")	→	Security systems -Entertainment systems (home theater, whole house sound)
Rational purchasing Personal style	→	'Customization" of desired systems (variable, persona options)	→	*Intelligent subsystems across all categories *Consultative sales approac
Concern for "earth" Rising energy costs Air quality concerns	→	-Increase house energy efficiency *Purify air	→	Zoned HVAC systems Efficient units Better controls Air quality features

Of equal note is the proliferation of intelligent subsystems from manufacturers in established categories. These companies are taking traditionally separate components and integrating them. The most obvious area of growth is home theater. The integration of audio and video into supersystems offering superior experiential benefits as well as distributed sourcing is occurring rapidly. While home theater remains too esoteric for most consumers' budgets, it is certainly not too esoteric for their tastes. Every major entertainment manufacturer is entering this arena. With volume, equipment prices will drop, enabling more families to afford home theater.

In energy, both Carrier and Lennox are offering zoning systems. Carrier's HomeZone provides for up to four zones of HVAC off one heat pump, internal diagnostics, remote dial-out to a central station, and even electric utility pricing signals. Harmony, Lennox's new offering, also offers four zones and internal diagnostics. Each of these companies is taking back the control function traditionally left to control manufacturers to allow smoother integration of functions and enhance consumer benefits.

In security, several companies are expanding their systems to link to automatic lighting and remote (away from home) status, and even add control capabilities. Some are crossing categories into entirely new areas. ITI, the leading manufacturer of wireless security systems, has announced a new product: The Meter Minder. The Meter Minder works with ITI's security system and allows electric utilities to read meters automatically and do load control as well as conduct traditional security monitoring.

Telecommunications equipment and services will explode by the end of this decade. With restraint relief visible for the Regional Bell Operating Companies, ISDN will finally emerge. So will new consumer services and equipment that allow consumers to comfortably use these services. Parks Associates believes it will take our giant telephone companies another 12 to 18 months to begin effective market rollout of new offerings, but they will do it and they will do it well for the most part.

Just as important to the process of home system penetration during the '90s are the reasons why consumers want these systems. Faith Popcorn, coiner of the term "cocooning" in the '80s, postulates that the '90s will be the era of the "armored cocoon." Our consumer research echoes her belief.

Consumers want to spend more time at home than they did in the '70s and '80s. Reasons for this trend include the consequence of baby boomers approaching or reaching 40 years old; the result of the "baby boomlet" (couples in their thirties having children); the reaction to scary external conditions ranging from pollution to crime to AIDS; and the shift in values towards those more family oriented.

Dual income families and single working parents are now a staple of America. These families are tired. They want hassle-free equipment (automatic) they can easily understand and use. These families are also busy. Retreating to home on weekends is deemed a luxury and a desire. Entertaining at home is becoming popular again, which is good for all sorts of products including home theater and fancy A/V systems. Home theater is also viewed by many consumers as an excellent approach to shared family time at home.

Most American families desire privacy from outsiders while at home. Consumers will subscribe to automatic number identification as it becomes available. Also, managing a variety of life roles is important to consumers. Fifteen percent of American homes have two telephone lines. Many homeowners would love to be able to distinguish their telephone calls by type. For example, knowing if an incoming call is business, social, or family in nature before you answer would be nice. ISDN will emerge in the '90s [finally) and will bring with it this capability for busy families.

In sum, the integration of functions to create more and better systems will occur rapidly in the '90s. This development will be possible because home networks will be available. The pricing of technological parts will continue to decrease, and consumers will want the benefits of these types of systems to make their lives easier and to make living at home more comfortable, more convenient, and safer. ❑

Tricia Parks is president of Parks Associates in Dallas, Tex. Tricia provides information to industries that serve residential environments with technologically advanced products, systems, and services.

Take a Tour of the Bright Home

FEATURE ARTICLE

Ken Davidson

Take a handful of home automation companies, add one model home, and you have the recipe for the first in-house CEBus showcase open to the general public.

If you recall the end of my last CEBus update article ("CEBus Update: More Physical Details Available," ***Circuit Cellar*** INK #21), I briefly introduced the Bright Home and made a comment that perhaps a trip to Indianapolis could be arranged. Well, enough of the right people were listening, and I soon found myself Indiana-bound. I took lots of pictures while visiting the Bright Home, so allow me to show you around.

The Bright Home was the first in-home demonstration of CEBus technology open to the public. The house was actually a model home in a new development on the outskirts of Indianapolis. The builder for the development used the house to show prospective buyers what they could expect if they signed on the dotted line to begin their own construction. The home offered much more than your typical model home, however.

Sponsored in part by Indianapolis Power and Light Company and PSI Energy, the home was a showcase for the latest in energy-saving devices and building techniques. The house was built with every conceivable joint and crack sealed with a plastic vapor barrier or expanding foam. Even the holes used to run wires and pipes through the wall studs were sealed with foam. Walls were insulated to better than R-19, and the attic was insulated to better than R-50 [the typical Indianapolis attic is insulated to R-30).

An electric heat pump was installed to do all the heating and cooling of the house, and officials expect to keep the house comfortable year-round for an average of $19 per month (electricity is cheaper in the Indianapolis area than in much of the country, so this difference must also be taken into account). Fluorescent and halogen lighting were used throughout the house to further save electricity.

My interest in the Bright Home wasn't its novel building techniques, though. Part of the overall energy management scheme included whole-house automation using CEBus-linked devices. During the construction phase, a pair of coax cables and four-pair twisted wire were run to each room in the house, all converging at a central location in the basement. Extra power outlets were also installed around the perimeter of each room and in other strategic locations throughout the house (more on these outlets later).

Visitors to most model homes simply wander through the house, perhaps with a salesman in tow pointing out specific features that make his home the best in the market. With all the extra features found in the Bright Home, most people either would be overwhelmed by all the new gadgets or, more likely, wouldn't notice them because they were so well integrated into the house. Visitors to the Bright Home are greeted by a fully scripted performance, complete with actors and staged events.

I really should say "visitors were greeted" because public viewing ended in September, 1991. At that time, the family who actually purchased the house took possession of it. Because your chance to see the action in person has passed, I'll stage the performance to the best of my recollection and take you through the house myself.

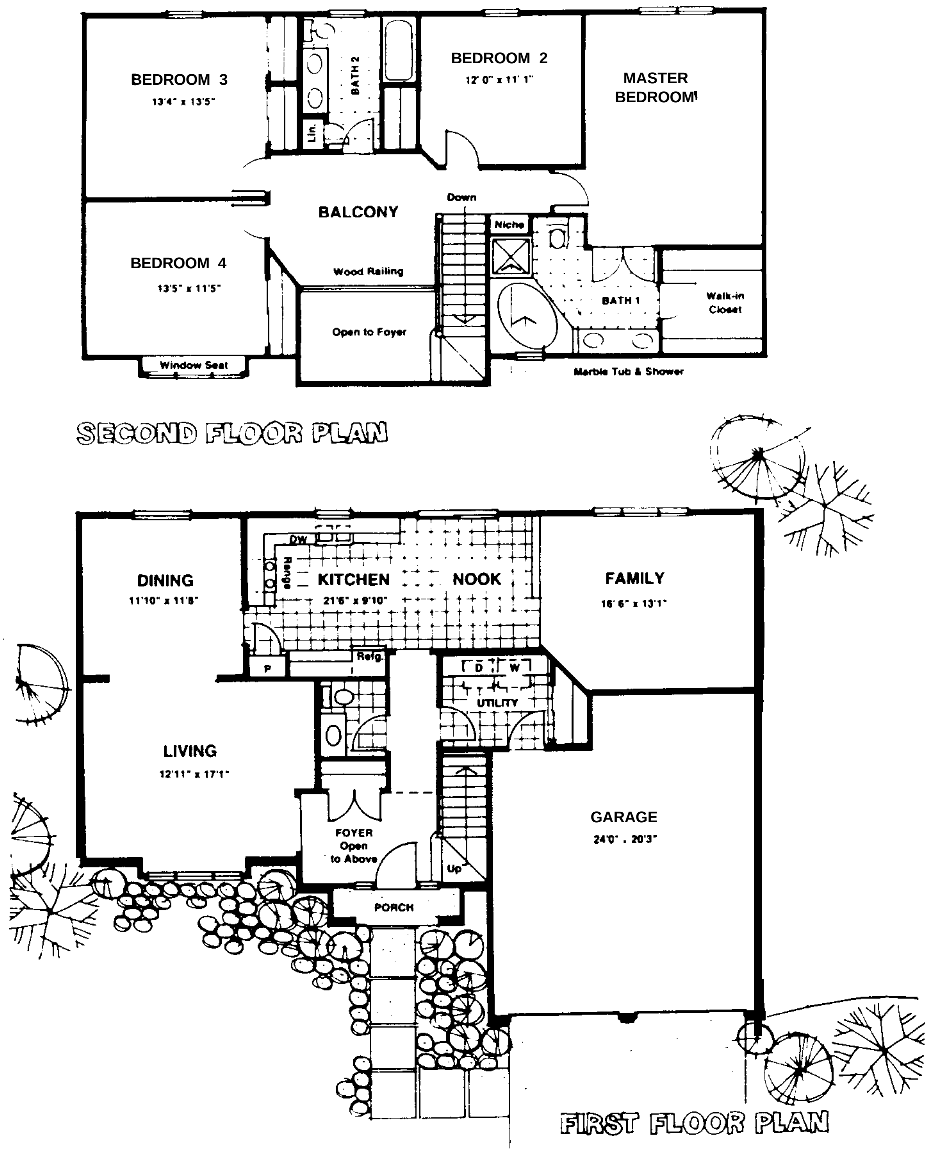
ACT 1

The drama unfolds as we approach the front door. Ringing the doorbell, we are greeted by a voice from a speaker next to the door, which explains that "Mrs. Jones" is on the phone at the moment, but the house will let her know we are waiting and she'll be right out. Momentarily, the door opens and an apologetic Mrs. Jones greets us to "her" home.

▶ *The Bright Home is a 2400-square-foot, two-story, traditional home in suburban Indianapolis designed to showcase energy-efficient construction techniques, high-efficiency appliances, and prototype CEBus technology.*

We are led into the living room while she explains the use of high-efficiency halogen and fluorescent lighting throughout the house. She also introduces us to the idea that the home has been fitted with a new technology called CEBus, which will revolutionize the way we live in the future. All of the devices in the house communicate using one of the CEBus media, including power line, twisted pair, coax, and infrared.

(RF wasn't used by any of the devices in the house. All of the CEBus devices were working prototypes supplied by numerous manufacturers. Most were based on an older version of the CEBus spec, but that's not surprising, given how new the spec is and how long it takes to get a prototype working.)



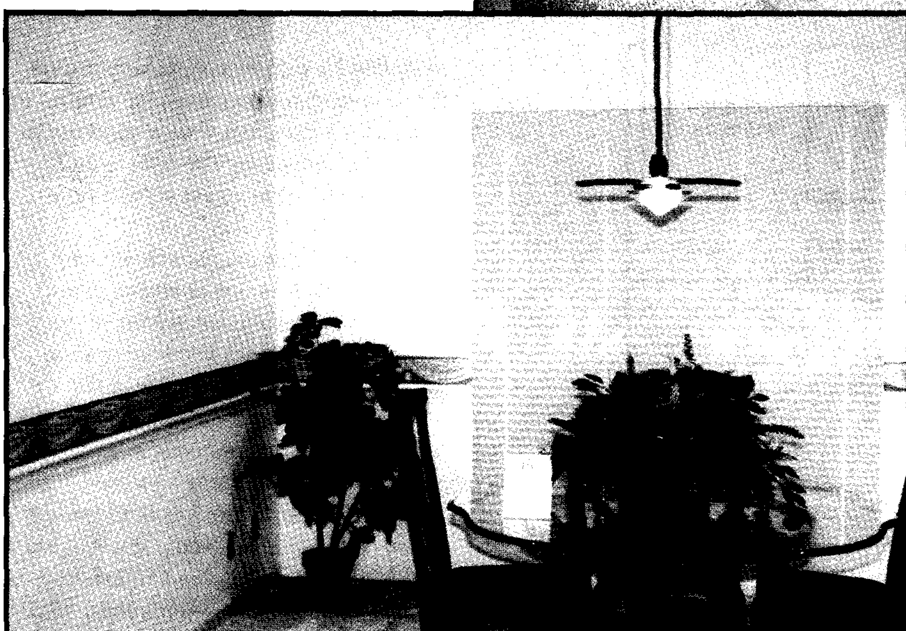
▲ *Use the floor plan of the Bright Home to follow the path of your guided tour:*

Next, we're led into the dining room where Mrs. Jones shows how easy it is to automatically bring the lights up to a preset level, or to dim them to provide a more romantic ambiance for those evenings when the kids are away at Grandma's, all with the press of a button or two.

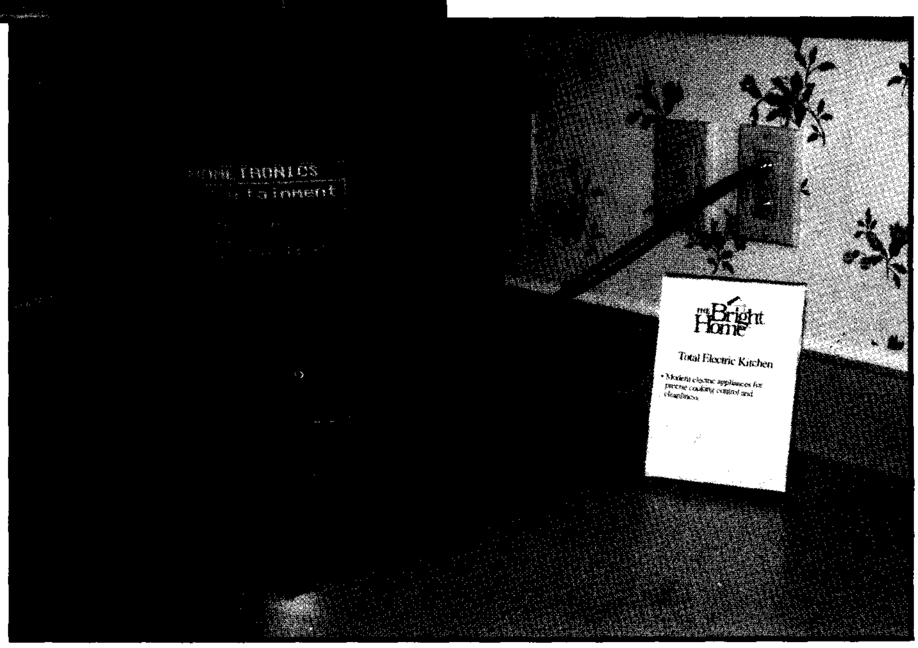
Walking into the kitchen, we are greeted by more high-efficiency lights and appliances, but also a television sitting on the counter with some extra outlets on the wall behind it. Mrs. Jones picks up a handheld remote, points it at a receiver on the wall above the door, and a menu pops up on the television screen. She proceeds into a monologue about how wonderful the future will be where we can bring up recipes on the screen from our in-home database, or how we can check out the local restaurants, complete with pictures of the interiors, then make reservations for the evening.

(I found the use of futuristic on-line services within the context of a demonstration of current or near-future devices to be out of place. The already confused consumer may look at the far-future services and attribute

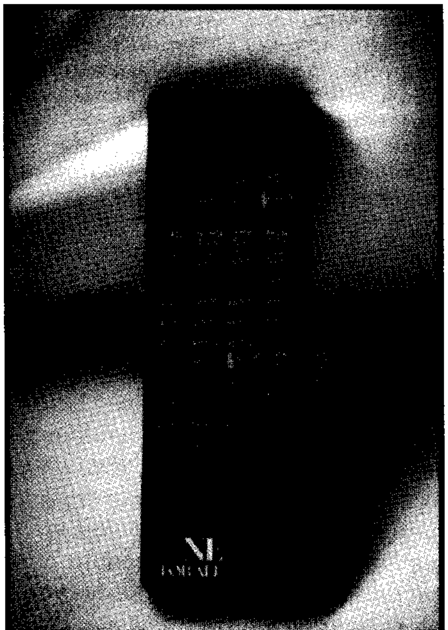
▶ *The living room sports high-efficiency fluorescent lighting. Also note the motion detector on the wall in the upper left corner.*



◀ *Directly* **off he living mom,** *the* **dining** *room is* **lit by contemporary halogen** *fixtures that* **can be dimmed** *to suit* the desired mood. As in **the** living room, **note** the **motion detector** on the **wall** above **the corner plant.**



▶ *In the corner of the kitchen between the stove to the left and the sink to the right is the main interface to the home's CEBus devices: a television. Extra jacks installed in the wall make connection to the house's network as easy as plugging in a cable. The hand-held remote is used to call up menus on the TV screen which control devices throughout the house.*



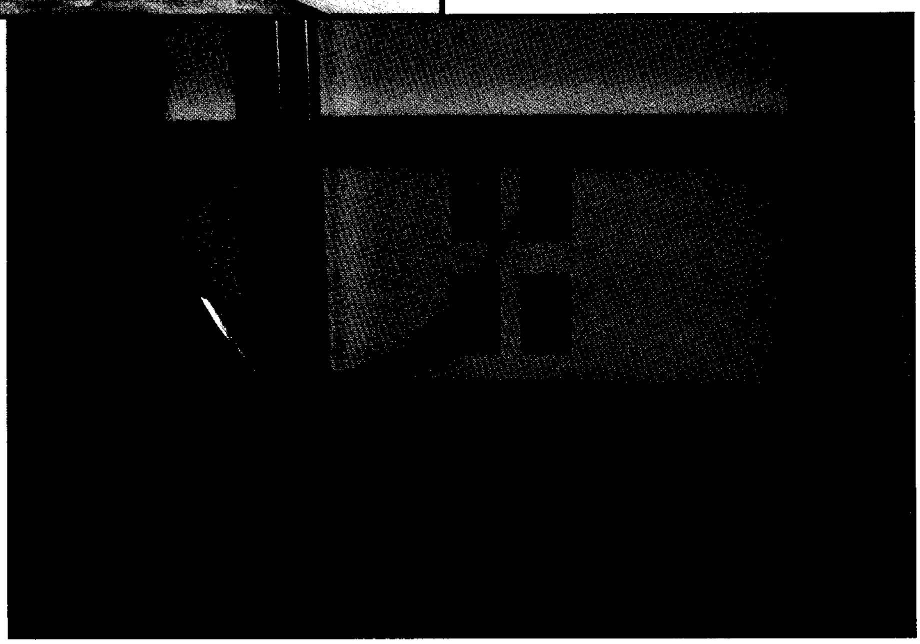
◀ *Universal Electronics'* **One-For-All** *hand-held IR remote can be bought off store shelves throughout the country today. Universal added CEBus codes and a custom label to a stock remote control and used if to provide user input to the Bright Home. One-For-All remotes bought today can be easily updated in the future by dealers to handle CEBus control codes once more CEBus devices start to hit the market.*

everything in the house to also being fake and in the distant future.)

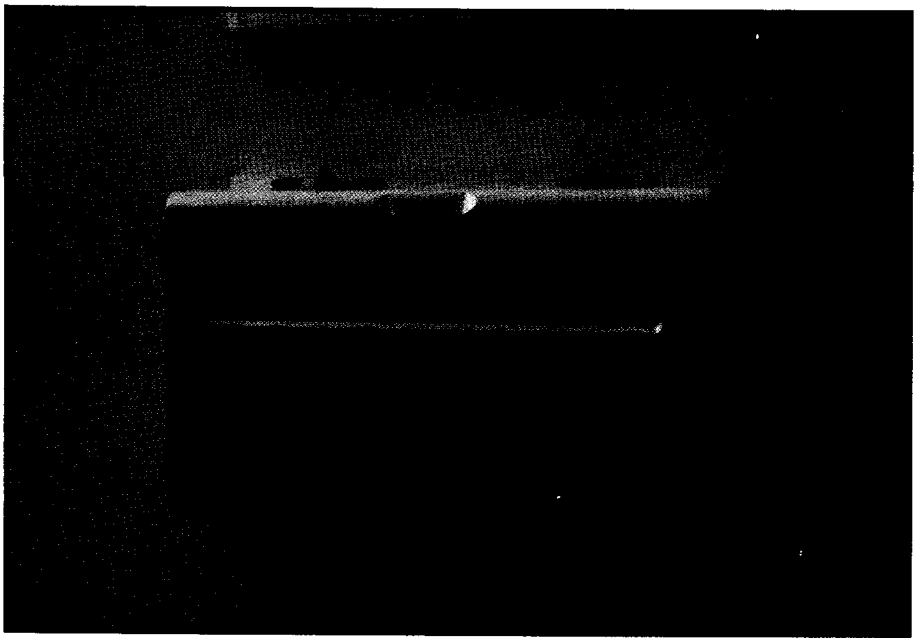
With background music piped into the kitchen from overhead speakers, Mrs. Jones pretends the phone is ringing and answers it. Much to the amazement of everyone in the room, the music cuts out as soon as the receiver is lifted.

Walking through the breakfast nook into the family ro still in hand, Mrs. Jones introduces s to the home entertainment system. A complete stereo system sits off to one side of the room while a large television sits in a another corner. Pressing a single button on her remote, she activates a series of automated events, which include the drapes behind us closing; the stereo, TV, and VCR coming on; and the movie in the VCR starting to play. Again, all communication between devices is via CEBus media. The drapes are controlled over the power line and are accommodated by extra outlets installed near the tops of the windows throughout the house. TV/VCR communication is over coax, while communication between the owner and the system is through the IR remote.

◀ *The family room contains another television which operates identically to the one in the kitchen. Drapes can be closed, the temperature can be adjusted, and the VCR can be started all by using a single remote whose signal is relayed through the TV.*



▶ *Behind the scenes in the family room, we can see more built-in jacks that facilitate easy interfacing of CEBus devices to any of the supported media.*



◀ The Johnson Controls CEBus thermostat can be set by walking up to if and pressing a few buttons just like any other electronic thermostat. The difference is this one can be set from anywhere in the house by communicating through he CEBus twisted-pair medium. Current temperature and set point can be called up on any television screen and adjustments can be made on the spot.

Mrs. Jones next shows how devices can also be controlled interactively by pressing a button on the remote to open the drapes again. A message on the bottom of the TV screen reflects the change in status. She also brings up a display on the TV showing the current temperature and thermostat set point. Changing the set point is as easy as pressing a few more buttons. Communication with the thermostat on the wall in the front hall is via twisted pair.

Mrs. Jones then picks up the phone on the end table and calls upstairs to "Mr. Jones" to see if he is ready for some company. With the go-ahead, the group proceeds upstairs for more.

ACT 2

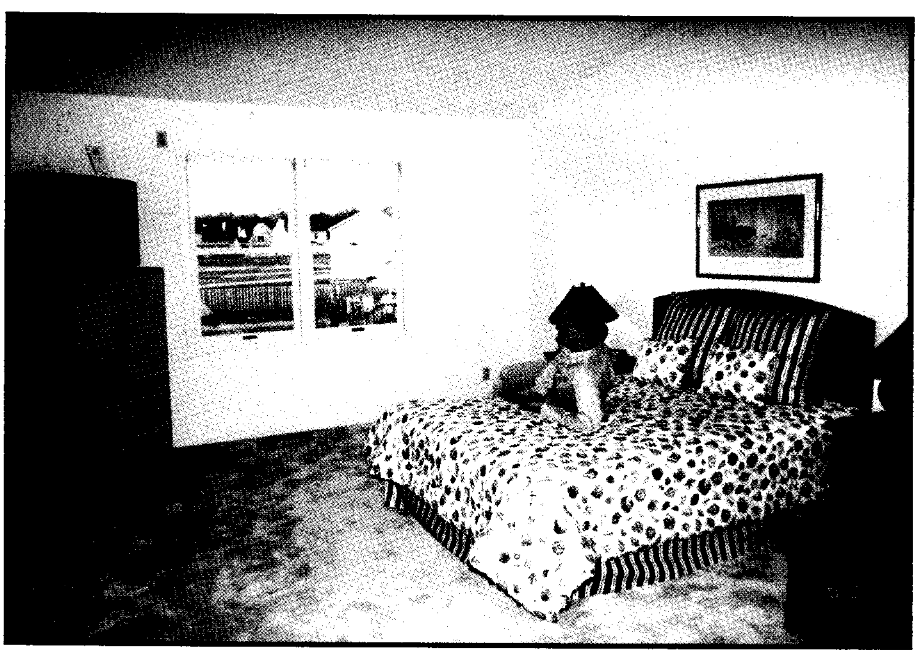
We are greeted at the top of the stairs by Mr. Jones, who also starts espousing the virtues of the home's efficiency and the wonders of CEBus. We are led into the master bedroom where, armed with a similar IR remote, he turns on the TV in the comer of the room and brings up the same movie we'd been watching downstairs. He then switches to a camera mounted in the garage to show us we can select from multiple video sources, including both cable and in-home generated. With another "here's something neat that you're not likely to see real soon," Mr. Jones proceeds to describe the idea of buying movies from home and having them sent via high-speed telephone connections (presumably ISDN or beyond), so we get personalized service without leaving the comforts of home.

Walking down the hall, we come to Mr. Jones' home office. Here, we have everything a good home office should have, including a computer, printer, telephone, and fax machine. The bonus is the extra outlets on the wall including four power outlets, a telephone jack, a twisted pair jack, a pair of coax jacks, and some blanks. (In a real CEBus implementation, telephones are incorporated into the twisted pair medium, but in this installation we still have a separate telephone system.) While in the office, we get the standard pitch about how everyone in the future is going to work at home. I'm sure you've heard it before.

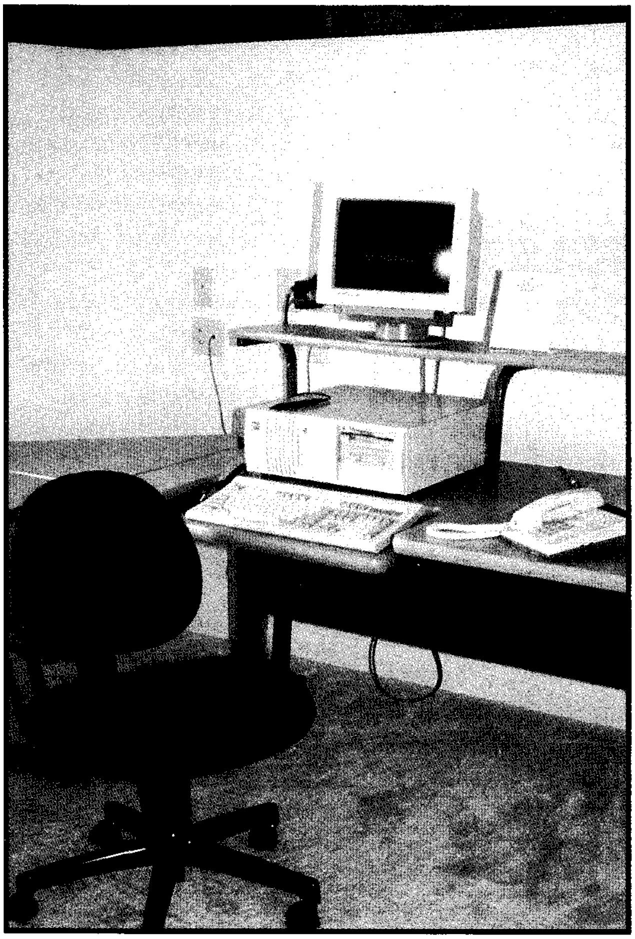
Mr. Jones then tells us a "technician" is down in the basement just finishing up a few last-minute items, and perhaps he'd speak to us. A phone call to the basement confirms that he'd love to see us, so the group meanders down to the basement.

ACT 3

Once in the basement, the technician tells us mostly about the heating and cooling aspects of the

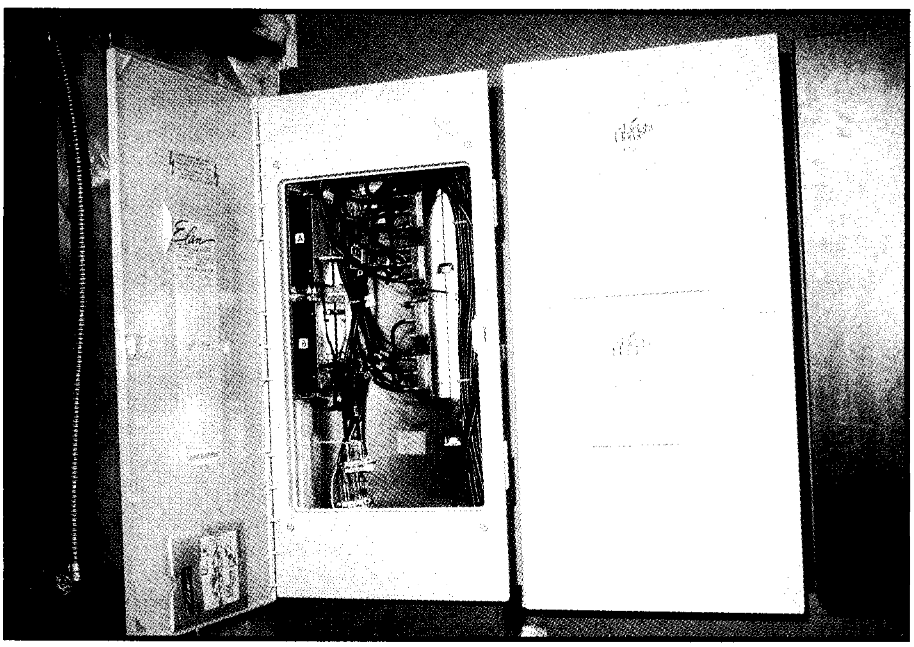


◀ In the second-floor master bedroom, yet another television provides the user with access to most of the house. Programming coming into the house from the local cable television company may be selected for viewing, as may the downstairs VCR or a remote video camera trained on the front door. Note the additional AC outlet lo the upper left of the windows, installed to facilitate CEBus-compatible automated drapes.

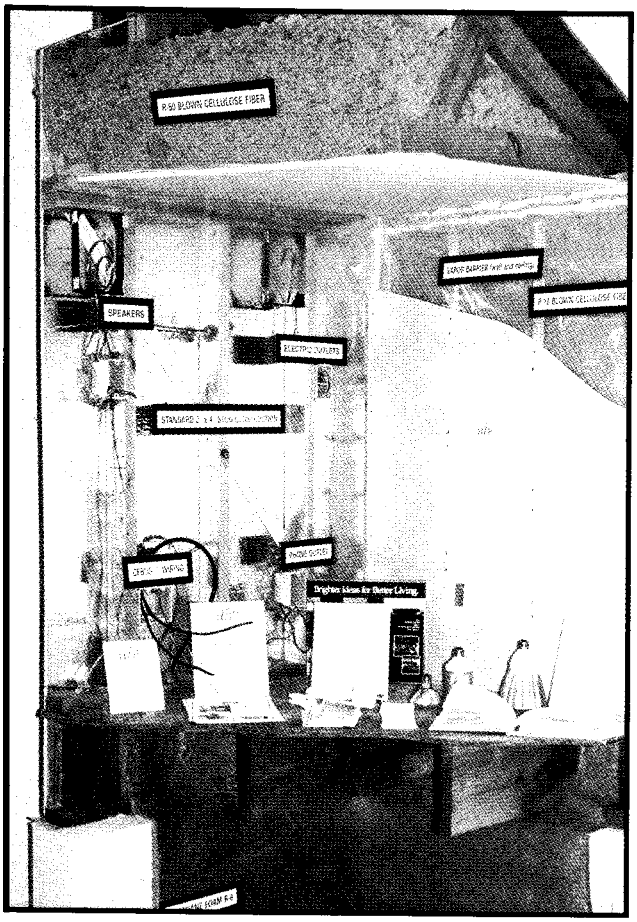


◀ **The home** *office contains* **everything you'd** *expect of the* **domestic** *businessman, including a computer and telephone.* ***As in*** *most of the house,* **additional** *jacks are installed in tie* ***wall to*** *make* ***plugging*** *into the* ***CEBus*** *network a* **snap.**

▼ ***CEBus*** **tired media** *require a* **"Node** *0" to oversee distribution of the signals* ***throughout*** *the house.* ***All*** *the coax in the Bright Home goes to a common pint in the basement where both external and in-home-generafed video are combined and redistributed,.*



#137



◀ **The** *Bright* **Home** *incorporates* a *host of construction techniques to make* **the house** *as* energy *efficient as* **possible.** *It's not* always **possible** *to see what* went *into* building *the* house once *it's complete,* so the *sponsors make it easy to study the techniques used by* providing a *small mock-up that encompasses them. Note, too, the CEBus wired media installed at the time the house was built.*

house, which I've already briefly gone into. He also shows us a single panel that houses all the coax cables from the various rooms throughout the house. Within the panel, the cables go to assorted switch boxes and splitters and look very impressive. He wouldn't open the other two panels or the closet under the stairs for me. "House Rules." [I did manage to get a peek under the stairs later and saw the magic running the parts of the house that had to be faked. While all the CEBus devices in the house were truly operating over CEBus media, the demonstration had to be pulled together quickly, so it was inevitable some hand waving had to be done. I certainly don't take anything away from the designers for doing so.)

THE DRAMATIC ENDING

After we finish poking around in the basement, we are ushered into the garage, which has been made into a makeshift sales office and showcase for the home's construction techniques. Numerous displays are set up to provide a more in-depth look at what we just saw on the tour. As you'd expect, officials are available to answer questions.

The Bright Home was not a place to pick up technical details (and, indeed, it wasn't the home's intention to provide them). I took my initial tour with a builder who was interested in the ideas, but wasn't very technical. He did have questions that the very nontechnical actors couldn't answer, though, so I ended up doing as much narration on the tour as the tour guide. Later in the week, I went through again with members of the CEBus committee who were in town for their regular meeting. Many of them wore expressions of amusement at the nontechnical presentation and over-simplified details, but they were also delighted that the word is starting to get out to the common consumer.

Overall, the Bright Home was an excellent demonstration of what we can expect to begin seeing on store shelves within the next few years. It exposed Joe Consumer to concepts he thought only the rich could afford, and started the all-important consumer education that will be necessary if home automation is to be accepted by the masses.

CURTAIN CALL

The CEBus aspects of the Bright Home were pulled together by members of The HomeTeam Project, administered by Mike Coffey and his crew at Hometronics Inc. Members of the HomeTeam Project contributing equipment to the home include Draper Shade and Screen, Integrated Communication Systems, Johnson Controls, Leviton Manufacturing, Panasonic, Somfy Systems, Sony, Square-D, Thomson/RCA Consumer Electronics, Universal Electronics, and Tenex Computer Express. For more information about the HomeTeam Project, contact Hometronics at 4405 Massachusetts Ave., Indianapolis, IN 46218, (3 17) 54.56239, fax (317) 545-6237.

IRS

401 Very Useful
402 Moderately Useful
403 Not Useful

The Circuit Cellar Home Control System II

FEATURE ARTICLE

Steve Ciarcia

What is our favorite home automator up to these days? When he's not reminiscing about the good old days, he's dreaming up the future.

Boy it's cold out here, I thought. Even with a sweater on under the heavy coat I could feel the cold blast of wind. Being in the middle of a clump of pine trees offered little additional protection. Instead, I felt like I was being flagellated by hundreds of little green whips in unison. The calendar might have said fall, but the ear lobes peeking out from under my hat were screaming winter, winter.

It was really my own fault, of course. I generally try to schedule excavation in a more opportune climate, but this project had to be finished if I want to get it in the next issue of ***Circuit Cellar INK.*** Why did it seem genius and the weather were rarely ever in sync on these projects.

When we decided to design a new more powerful network-based home control system (HCS), the weather didn't seem to be of particular importance. As I started actually installing it, however, the revelation that many new sensors and control outputs could be accommodated prompted the challenge for immediate installation. Of course, the inevitable program exercising all these sensors and controls would be left to the brilliance or madness of the operator. That and cabin fever during the Connecticut winter.

When I placed my foot on the shovel to dig a hole for the new infrared perimeter sensor post I felt significant resistance. Finding more rock than dirt under the shovel would hardly be a new disclosure. Would this go from being a relatively simple post hole digging operation to a major excavation? Would I have to drag out the backhoe for one crummy post? There were only so many places I could put these line-of-sight sensors after all.

I shrugged my shoulders and pulled my jacket collar up to better protect my neck from the wind. Presuming that tough dirt needed tougher technique, I jumped full-weight on the shovel with both feet. The realization that one inch below the surface was solid rock came too late as I balanced precariously on the point of the shovel. Then, falling backwards, I instinctively extended a foot to break the fall. But, as we all learned at an age when our bodies were more flexible, wet pine needles have little friction. With a very audible thud I impacted the ground in a sitting position. Ugh!

Any sane person would have been cussing up a storm under the same conditions but I actually sat there laughing. It was deja vu!

I've been here before!

Sitting in the mud triggered a flashback to my last house when I also found myself sitting in the mud amongst the pines. Much too coincidentally, it too dealt with my security and home control system.

"Merrill, you gotta help me!"

The feeling of panic was coming over me as I beat on Merrill's back door. I needed help, and Merrill was the only person I could trust, and the only person who would understand I wasn't crazy. As I knocked on the door, I glanced over both of my shoulders to make sure no one else was around.

"Merrill, you gotta help me!"

I stood next to the door in a shadow that the moonlight failed to illuminate. It was a cool spring evening. While the stars shown brilliantly in their quiet elegance, I couldn't help but fear that this would be the last quiet moment of the evening if I failed.

"Steve? What are you trying to do? Can't you just ring the bell and wait 30 seconds like everyone else?"

Merrill wasn't really mad, just startled at my wild-eyed look and disheveled appearance. I often visited him but usually announced myself by


"Merrill, you gotta help me!"

some means other than beating down his back door. He waited a few seconds. Then he realized that this wasn't a social call and changed his tone to one of concern.

"What's wrong? You look terrible."

"I locked myself out!"

For any other person in the world, that would not be a catastrophic occurrence. In fact, the words sounded a bit absurd as I said them. I only hoped that Merrill valued our friendship enough to listen to me.

"You locked yourself out? Didn't you once give me a key to hold just in case this ever happened?" Merrill was becoming increasingly curious as to why I should be so distraught. I should have known he had a key.

I nervously glanced at my watch and answered. "That was when one needed a key to get in my house." Such a statement obviously would lead to all kinds of conjectures, but I didn't have time to explain.

"What do you mean, no key? How do you get into your house? Whistle?" Merrill seemed a bit disturbed that I was playing guessing games.

"I don't use a key anymore. I use a digital code. I really don't have time to explain. Please, just put on some dark clothes and help me."

His help-thy-neighbor attitude took five giant steps back when I mentioned the necessity for dark clothes. Glancing at my watch once again to see how much time we had left, I determined that a portion of it had to be allocated for explanation. I stepped into the doorway and moved past Merrill.

"You see, Merrill, I've locked myself out of the house, and I have a souffle in the oven."

Merrill looked at me like I was some kind of nut. He walked over to the kitchen sink and opened the cabinet doors beneath it, revealing a toolbox. "Look, we'll zip over and pull the hinges on one of the doors. It's a cinch."

Before he could pass me any tools, I interrupted him. "Merrill, it's not that easy. You don't understand. Let me explain."

The expression "Please do" was painted all over his face and needed no verbalization. As he sat down in the overstuffed chair, he extended and crossed his legs on the footstool and stroked his gray beard nervously. The little bit of fuzz on the top of his balding head seemed to bristle like a cat. To further the impression that he was ready for a real fish story, he took out a briar pipe from his pocket and nonchalantly started to clean it. Between the sounds of tapping the pipe on the ashtray and blowing through the stem to clear it, he extended his hand toward me and said, "Do begin, please."

The delay was excruciating. It was critical to act soon. The souffle was irrelevant. It was the chain of events that could be accidentally touched off that I was worried about. My only hope was to talk fast.

"Merrill, my house doesn't use a house key anymore because it has a computerized environmental and security control system!"

He puffed on the pipe and interjected. "Fair enough. But what's that got to do with the souffle?"

"This isn't just any home-control and security system. I designed it! An advanced sensor system tied directly into my computer makes it about the most sophisticated home burglar alarm in the world. I got thinking one night that I needed a burglar alarm. Since practically all the lights and appliances were already connected to the control system, I just extended its capability a little. But I got a little carried away on the engineering, and I'm not sure I can get in without setting it off."

Merrill was amused. Every time he and I had spoken lately, it had something to do with computers. He no longer thought I was completely crazy, just a little. There was still that one burning question. "What has that got to do with the souffle?"

"There's a souffle in the oven, and let's see...it should be done in 30 minutes. But the oven timer only buzzes, it doesn't shut the oven off. I know you're only an engineer and not Betty Crocker but even you can guess that it wouldn't be more than another 20 to 30 minutes before it starts to bum."


"But what's that got to do with the soufflé?"

I spoke rapidly. We were eating up precious seconds. "When the smoke from the burning souffle hits the smoke detectors on my alarm system, all hell is going to break loose on this street."

"Wow! What does it do, call the police?"

Most people are familiar with the standard smoke and burglar alarms that automatically dial the fire department. While the end result was the same, my method was quite different. The sophistication of my home-control computer was unmatched by anything that commercial companies had to offer. That, in combination with the mind of your average, everyday mad scientist, can produce startling results.

"Well," I started rather sheepishly. It isn't often one has to explain the limits of his paranoia. "It isn't every day you have a fire in your house. When you do, you want action fast so you can reduce the damage and get people out in time. This system is predicated on everyone acting fast. When a fire or smoke is detected, it first sets off the alarm horns mounted outside next to the garage. I've never tried them, but they're war surplus air-raid sirens.

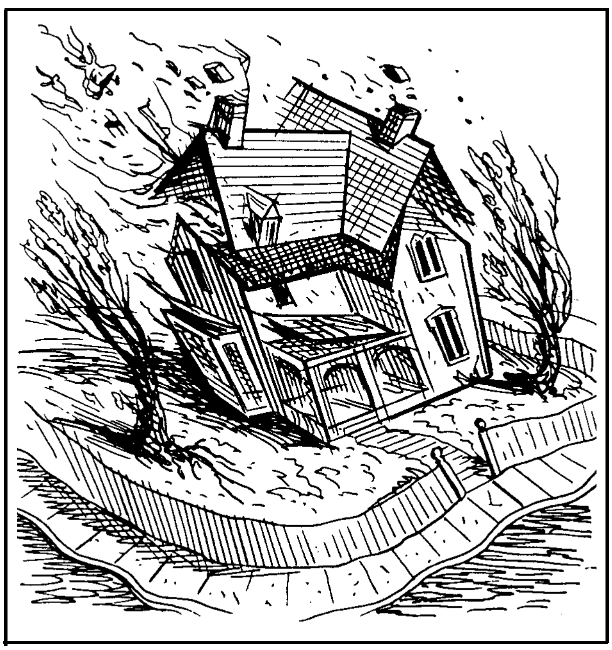
"Mathematically, the sound level ought to be high enough to break about half the windows on the street. Mrs. Picker, who lives directly across from my house, will probably have her whole house moved back about two feet when they go off.

"Secondly, there are four xenon strobe aircraft-landing lights mounted on the corners of the house that will start flashing with about 2 million candlepower each. That was just in case the fire trucks had trouble finding the house.

"Then come the automatic telephone calls out on the three telephone lines. Remember, Merrill, my computer has a voice synthesizer, so I don't need a tape recorder. It definitely doesn't sound like a recording, so it should prompt immediate action. The first call is to the fire department. It also is simultaneously transmitted on CB channel 9. Then a whole bunch more. The end result is more cars and trucks than we can fit on this street."

The pipe in Merrill's mouth drooped lower and lower as I conveyed the consequences of my alarm going off. It was hanging down to his chin when he muttered, "Why don't you add me to the list of calls in case I miss the initial shock wave."

"Don't worry, Merrill! You're the ninth call!" Merrill definitely had a concerned expression on his face. As I expanded upon the next step, it turned to terror.



"Mathematically..."

"Merrill, you gotta help me break into my house and shut the alarm off before the souffle bums."

The pipe fell out of his mouth, and the ashes formed a line down the front of his shirt. He barely noticed them as he exclaimed, "Are you crazy! Break into your own house!"

"Look, Merrill, I designed that system to prove I could do it. Now that I can count the seconds before I know it's going to go off, I recognize it as pure overkill. I'll replace it later with something more sane, like six Doberman pinschers and a minefield. But right now we have to stop it! Will you help me?"

Merrill brushed the ashes off his lap and jumped up. "Do I really need dark clothes?"

"Yes, I'll explain later. And wear a dark sweatshirt with a hood or something to cover your head."

The evening newspaper fell to the floor as it was sucked off the table by the vacuum created by Merrill as he ran to change. I could detect a cold sweat forming as I checked my watch repeatedly. It was only 10 minutes since we had first started talking, but now it was only 20 minutes before the soufflé would be done.

I could picture in my mind the progression of events that would follow. First, the soufflé would blacken and crack. Then, as it shriveled, some of the exterior sections would have dried enough to be combustible. The first whiffs of smoke would go unnoticed, but eventually a billowing cloud would spew forth from the oven. When it reached the smoke detectors, the computer would go into action. Our only hope was to get inside in time to stop the computer. If we failed, we had better make sure we were not standing next to one of those sirens when it blew. Further thoughts were interrupted as Merrill burst into the room fully dressed for action.

"Gee, Steve, why don't you lose a little weight for the next break-in?"

"I'm ready. Let's go."

Merrill looked like a cat burglar. The solid-black sweatshirt had a hood that completely covered his balding head and, while his gray beard still showed, it aided the camouflage. His pants were equally dark and skintight. All reflective surfaces like belt buckles and key chains were carefully omitted. Black track shoes completed the outfit. I only hoped we didn't have to do too much running with the rope.

As we jogged up the street toward my house, Merrill turned and asked, "You sure you know how to get in!"

The details of the computer alarm design flashed through my mind. I knew every wire, every sensor. Yes, I knew what the components of the system were, but the computer had far greater speed than I at analyzing the data received from them. A pressure switch activated in the wrong sequence, a heat sensor detecting human presence, any number of things could activate the alarm. I had let my inventive genius run "open loop." The tiny credit card that now lay on the coffee table in the living room had been my only control over the potential Frankenstein I had created. True, it would foil a burglar or call the fire department, but the ends to which I had gone in devising the system were aimed more at instant incineration of any perpetrator than protection of property.

To fully answer Merrill's question was impossible. I didn't know whether I could beat myself at my own game.

"I don't know, Merrill. I hope so."

We stopped in front of my house. Almost magically a floodlight switched on to illuminate the area before us. Music could be heard from inside. A light in one room switched off, and another turned on. I didn't wait for Merrill to ask because I knew he was curious.

"Most of the AC outlets in the house are remote-controlled. The computer can control almost any light or appliance in the house, except the stove. The computer knows that something or someone is out here from microwave motion sensors planted in front of the house. No one is in the house, but it is simulating habitation by playing music and making it appear as though people are moving from room to room. Just for good measure, it turned the floodlight on to tell you that it knows you're here too."

Merrill started toward the front walk. I grabbed his arm to stop him.

"Forget it. The only way into the house is through some window that doesn't have any sensors attached. They're in the back of the house. Possibly one of the bathroom windows would be the best to try."

"Hey, Steve, before I lay my life on the line to save your souffle, do you mind telling me what happens if we set off the burglar alarm while trying to break in?"

My reputation had preceded me. The fire alarm was only part of the system. The burglar alarm was equally devastating.

"Well, there's a bunch of stuff I'll explain as we go along. It's too complicated to explain in detail. But the end result is that the computer determines the location of the perpetrator and then tries to lock him in the area where he has been detected and calls the police."

"If that's all, you can explain the accidental phone call to the police. They often get false alarms from automatic dialers."

"Wait, you didn't let me finish. Then, it sets off all the sirens and lights, just for good measure. And, oh yeah, there's a very loud noise source inside the house that's triggered, which is supposed to temporarily disable the perpetrator. Then it does all the same telephone calls, explaining there is a break-in instead of a fire."

Merrill looked at me in amazement. The adventurer in him wanted to go full speed ahead and tackle the Mount Everest of electronic obstacle courses, while his quiet engineering instinct suggested that he go home and check his medical insurance first. He shook his head as he said facetiously, "Why didn't you just use tear gas?"

"Oh, I considered it. It's just too hard to get the smell out of the Oriental rugs."

This unexpected response was too much for Merrill. As we stood there in the moonlight, I could see the sweat forming above his brow.

To this point, he had been aiding an eccentric neighbor. Though it had taken a long time and not through any direct explanation, Merrill was ready to admit that this computer alarm had to be stopped. There was no animosity that I had created it, just a realization of the full consequences of its being.

He, too, looked at his watch and sensed the seconds ticking away. No longer was he along for the ride. Now he was a committed participant.

"Let's go."

I knelt down next to a sandy area at the corner of the lot. Merrill looked over my shoulder. Grabbing a short stick to draw in the soft soil, I started to lay out the attack plan. "Here's the house, the property line, and the key obstacles. There's only one way to approach the house from the rear and not be detected. We have to go over the side-yard fence, along through the brush to the pine trees behind the house, then across the back lot. Have you done any pole-vaulting recently?"

"Pole-vaulting? Are you kidding? I just about have enough energy to go from the couch to the refrigerator for another beer. What are you talking about?" His eyes opened wide and projected a common expletive. The general translation was "Hey man, I agreed to break in a house with you, but I ain't pole-vaulting over no fence."

"Bark," I said. "Like this. Arf! Atf! Arf!"

That was the easiest way, but I had to agree with Merrill. The years in the Cellar being a mad inventor rather than a tennis pro had taken their toll. I wasn't about to pole-vault over any fence either.

"We've got to find a way over the fence without actually climbing on it," I said. "There are vibration sensors in the vertical supports that are meant to detect anyone climbing over it. Tripping it won't set the whole alarm off, but it will start a timer where the computer treats perimeter events more seriously. If during that period the computer senses too many motion and vibration inputs, it will treat it as a threat and react accordingly." I didn't elaborate on the latter.

We stood next to the fence. It was constructed of heavy wire mesh attached to metal supports. Trying to vault over such a fence and missing would be like putting your body into a cheese grater. It was only about five feet high though, so there had to be an easy way over it.

Merrill looked at the situation. I could see his engineering knowhow going to work. Pictures of levers, fulcrums, balances, and pulleys were flashing through his mind. Walking over to the tree adjacent to the fence, he started coiling a length of rope in one hand. With one mighty swing, he threw the coil of rope over a 20-foot-high tree limb hanging directly over and parallel to the fence. Now the rope hung down and touched the top of the fence. "Come here, Steve," he said.

I was still a little puzzled, even as he looped the rope around under my arms and tied a knot at my chest. Only when he pulled on the other end and hoisted me off the ground did I realize how he intended to get us over the fence.

"Gee, Steve, why don't you lose a little weight for the next break-in?"

I felt like a side of beef hanging on a rope six feet off the ground. When he started swinging me from side to side, I thought I was going to get seasick. The amplitude of the swing got longer and longer until the arc carried me over the fence to the other side. The realization of what the next part of the sequence would be came a fraction too late for me to protest. As the arc carried me over the fence, Merrill let go of his end of the rope. Logically, I should have expected that this was the only way, but the experience of being swung on the end of that rope hadn't any semblance of logical reasoning on my part. My far-too-late protest started something like a "whoop" and concluded with the tonal equivalent of Tarzan merrily swinging through the jungle and suddenly missing the last vine.

The fall was only six feet, but it felt like a hundred stories. I thought that if this was a sample of things to come, maybe I should take my chances with the alarm. It didn't help matters when I landed sitting down. The ground was quite moist, and my clothing sucked up the water like a sponge. When I put my hand down to reorient my position, I felt the cold spring mud ooze between my fingers. The totality of my situation and the immediate sensations at hand were summed up with the single word, "Yech!"

"I'd swear that Merrill directed my flight toward the toilet on purpose, but I have no proof."

As I turned to check on Merrill, I caught a glimpse of him sailing through the air. Rather than be hoisted, he had secured one end of the rope and tied large knots in the other to aid climbing. Once at the six-foot level, he swung out over the fence as I had and let go. Even though he came down feet first, the momentum was too great for the terrain. It took only a fraction of a second for two skid marks to form behind his heels, and Merrill came crashing down in the same sitting position next to me. His first word was "Yech!"

I glanced at my watch and realized there were only 10 minutes left on the oven timer. I said, "Come on, Merrill, we can't sit here like two idiots. There's not much time left. We have to head for the brush on the right and then crawl toward the pine trees."

"Crawl? Why do we have to crawl?"

"I'll explain when we get there. Right now, pull your hood up over your head like this. Whatever you do, don't look at the house as you run past the brush into the pines, or the computer will see you."

"What is this, a science-fiction movie or something? What do you mean see us?" Merrill's nervousness was evident by the shrillness of his voice. He should have believed me when I said it was the most sophisticated alarm installed in a home.

"Just that. See that small box on the corner of the porch roof?" I pointed to a small black rectangular enclosure suspended below the corner of the roof line. About every 10 seconds a small red light flashed, giving it the appearance of being activated.

"There's a digital television camera in that box that scans this section of the yard between a height of three and seven feet. When that light flashes, it starts a scan and looks for changes in light patterns from one scan to the next. With our dark clothes, by running just ahead of the scan we should go unnoticed."

The 30 seconds it took while we watched the blinking light until we could anticipate the next scan seemed like an eternity. When the precise moment came, I yelled, "Head for the pines. Go!"

Running with both hands in our pockets to shield our skin from detection made trying to run at full speed rather awkward. It was more like a high-speed waddle than the statuesque gait of a long-distance runner. We had five seconds to make it to the pines before the camera would start to retrace its path and compare the new image to that of the preceding scan. It was barely 120 feet, but it took all our effort to achieve it in time.

As I was about to dive under the first pine for concealment, I remembered something vitally important. I crouched under instead. "Merrill, watch out where you walk. That is where my dogs.. . Oh, I see you just found out. Sorry about that Merrill."

Merrill was apparently just mentally chalking it up on his list of reasons to strangle me when the escapade was over-which it wasn't. Standing out there in no-man's_land was not accomplishing the task. Pointing to his watch, he said, "We have five minutes. What's this about crawling?"

"Don't worry about it, just crawl. Remember, we have to stay below three feet high. Don't stand up or we're dead. Ready? Go!"

Merrill still didn't understand why he was on all fours, crawling toward my house at ten o'clock at night. Life used to be so much simpler.

We were neck-and-neck about halfway across the yard when the computer spotted us. Two bright floodlights came on, illuminating the area where we lay. Merrill, exercising reflex actions learned from years in the Marines, instinctively dove into a prone position, as though he anticipated an imminent artillery barrage. At the same time the lights came on, the tumultuous roar of many vicious snarling dogs filled the yard.

Frozen in his position, Merrill yelled, "What have you got, a pack of hungry timber wolves in the basement? What do you need an alarm for?"

"Don't talk. Just bark!"

"Bark?" Merrill looked at me and shook his head.

"Bark," I said. "Like this, Arf! Arf! Arf!"

Soon we were both barking and woofing up a storm. My two Scotties would've been proud of us. We kept it up for about 25 seconds, until the lights and the ferocious dogs stopped as miraculously as they had started.

Speaking very softly and not waiting for questions, I said, "Hey, you can stop barking. There's a laser perimeter intrusion detector in this comer of the yard. It sensed our presence below the three-foot level. It turned on the floodlights and the recorded sounds of barking dogs to see what it was or try to scare it off.

"Now, here's what the computer is great for. After all that was triggered, the computer turned on a microphone to listen out here at the same time. When it heard us barking the same as any real dog would do upon hearing the recording, it shut off the alarm sequence. You see, Merrill, the computer thinks we are just a dog that wandered through the yard and not an intruder. A real burglar, smart enough to see the different sensors and trying to crawl as we have been doing, wouldn't know enough to bark back at the computer. Neat, huh? Now we can finish crawling to the house. It won't bother us again."

Merrill rolled his eyes and put his muddy palm to his sweat-laden forehead. As straight-faced as one could be, all things considered, he said, "Steve, you're crazy."

Not wishing to argue, since time was running out, I merely responded, "Genius is never appreciated until it's too late."

"Steve, tell me why I'm going through all this. What do you have in your house that is so valuable that you installed a system designed to counter an invasion?"

"Well, if I really think about it, I guess the computer control system and all the alarm sensors are probably worth the most."

Merrill didn't know how to respond to that information. We computer hackers design things sometimes just for the challenge. Unfortunately, this particular challenge was getting out of hand, and time was very short.

The remaining distance across the lawn was far less wasteful than the first. We encountered no land mines, bear traps, or quicksand. We finally found ourselves resting against the house just below the bathroom window. Reaching the next objective was not as bad as the preceding events. The window lock was easily pried open with a pinch bar. I warned Merrill not to make any noise once he was inside the house. Then I hoisted him up to the window. Grabbing the top of the window frame for support, he lifted himself off my shoulders and knelt on the window ledge. Next, trying to be as graceful as he could in such an awkward position, Merrill swung his body around so that he now

Mm. Picker

sat on the sill, with the trunk of his body hanging outside the window and his legs projecting inside. Once in that position, it was easy to swing into the bathroom and land squarely on the floor.

In a gymnasium, Merrill would have executed it perfectly. A small bathroom was quite another story. One foot came down squarely on the carpeted floor, as it should have. The other foot came down squarely into the open toilet, as it shouldn't have. Remembering that I had warned of excessive noise, he cussed very quietly as he extracted his foot from the toilet.

As he leaned out the window to pull me up, he said "Hey, Steve, I hear some kind of buzzer in the house."

I quickly glanced at my watch and responded, “That’s the oven timer. It’s running a little faster than I thought. Now the souffle is overcooking. Help me up. We haven’t got much time.”

Merrill leaned out the window and grabbed the shoulders of my sweatshirt as I jumped up to the window ledge. My entrance was far less graceful than his. I had no alternative but to go through the window head first. I’d swear that Merrill directed my flight toward the toilet on purpose, but I have no proof. At the last instant, I was able to extend an arm and apply a force opposite to that of my trajectory. The result was a dull, rolling thud on the bathroom floor.

Our totally disheveled appearances lent no levity to the situation. But we were inside the house, and the stove was just 20 feet away. If we could get to the souffle in time to stop it from burning, we would have all the time in the world to shut off the rest of the alarm. “Merrill, don’t say anything louder than a whisper. There are mics planted around the house, and the computer is listening for loud noises.” I extended a forefinger against my lips to dramatize what I had said.

“Steve, I just saw something outside. Outside near the fence in the backyard!” Merrill was looking out the window and after a few moments he excitedly pointed over my shoulder toward a dining room window, visible from the bathroom even though it was on the other side of the house. “There it is again!”

I jerked around in time to detect motion from an unknown object. “What do you think it is!”

Before he had time to answer, a human form stood for a second in front of the window. Extending from an arm was a long, slender object. For a moment, Merrill and I just stood with our mouths open watching the proceedings. The figure turned suddenly. The slender object exhibited a metallic gleam in the moonlight. Then the figure was gone, as quickly as it had appeared.

We looked at each other. Our eyes were wide open as we whispered in unison. “I think that was a gun!”

“I think we have a real prowler, Steve. What are we going to do? He has a gun, too!”

“Don’t ask me! Remember, I’m stuck here, too.”

“Suppose. Now just suppose he was able to get by all the alarms and got into the house. And just further suppose he fills his pillowcase and is about to leave when he decides to go to the bathroom. Voilà, us two looking down the barrel of that gun!”

“Shhh, Merrill. Don’t be an alarmist. Nobody can get through my alarm system.”

“Give me a minute or two to fix it, and I'll show you that the sirens really do go off.”

Simultaneously, as Merrill spoke them, I thought of the exact same words, “But we did!”

The situation presented a problem. Should I leave the oven on and purposely trigger the alarm to bring help and catch the prowler? Or should we still try to finish what we had started and then hope the perpetrator wasn’t smart enough to make it through my alarm?

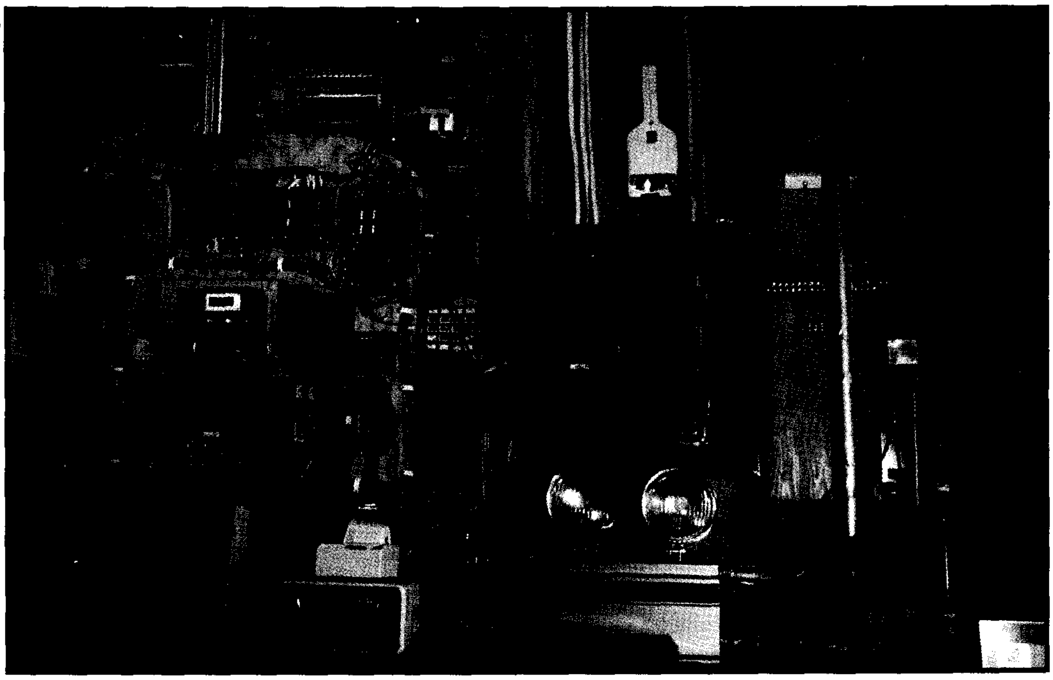
I looked at my watch. The souffle had to have been overcooking about 10 minutes. The stove timer was buzz relentlessly in the background. I sniffed the air. What had previously smelled freshly baked now had the distinct scent of being overdone. It would still take a few minutes before smoke would be produced that the computer could smell. We were in a real dilemma. We were caught between our protector and the prowler.

"Steve, look again!" Merrill pointed toward the dining room window. "It's a woman!"

The figure was in full moonlight in front of the window. The features were easily discernible, and I recognized the person immediately. The metallic glint previously thought to be a gun was the stainless-steel tip of a walking cane. I grabbed Merrill's arm tightly and said, "That's no woman. That's Mrs. Picker from across the street."

"Is that bad?" Merrill had little experience with Mrs. Picker. He could not fully comprehend the grave position we were now in.

"That's worse than any prowler with 10 guns. She probably saw us poking around and thinks we are the prowlers."



The **old** ***HCS (just* above and *to* the tight of *the* monitor) *is*** surrounded **by** add-on circuits that **are** *unnecessary with HCS II.*

"Boy, that must really take guts to confront two prowlers single-handedly." Merrill still didn't understand what I was trying to tell him.

"That feisty old lady might be 80, but I wouldn't put it past her to climb over the fence after us if she discovered the rope. What I'm really worried about is that while she's looking for us, she'll probably set the alarm off. When the law arrives, guess who is wearing the cat-burglar costumes and covered with mud?"

Merrill looked down at his clothes and back up at me. His eyes pleaded with me to act fast. We were in the worst possible combination of circumstances to be caught in. The only solution was to try to turn off the system before Mrs. Picker triggered it.

"Let's go," I said. "We still have to turn the oven off."

Merrill agreed. We had no other choice. Extra time to shut off the system was gone. First, we had to get to the stove. Motioning to Merrill that he should follow in my exact footsteps and mimic my every motion, like a childhood game, we started the ordeal.

"Merrill, see those two holes on either side of the door molding? Those are photosensors. The computer can tell if we pass through the door and in what direction we are going. Fortunately, they are only 18 inches off the floor."

At the doorway of the bathroom, I lifted my right leg very high and extended it out over the other side. Shifting my weight to the now firmly planted foot outside the bathroom, I retracted the other leg by reversing the process. Merrill followed suit. We stood in the back hallway outside the bathroom.

"Every doorway we go through, we will have to follow the same procedure. Got it?" Merrill nodded affirmatively as I continued to whisper. "Now step over this area and these other two. There are pressure switches under the rug that will go off if you step on them. Try not to make too much noise jumping. Remember the mics!"

Ordinarily, all these sensors and switches caused the computer to turn on lights and direct the stereo system to the appropriate rooms as I walked through the house, all in the name of convenience. Now, however, the feeling I had was like being in combat. I was in the middle of a minefield directing those behind to follow in my footprints. While the sensation of stepping on a mine could not be exactly equaled by my computer, the heart attack following the first sound of that air-raid siren could be just as lethal. We silently high-stepped and hopscotched our way through the house until we reached the stove.

As I extended an arm to turn the oven off, I could see the blackened souffle through the oven door's window. It was very disconcerting to see a creation of one's hand and mind shriveled and destroyed. But the realization that we were still at the mercy of another such creation prompted a fast exit. We had not gotten to the stove too soon. Inside it was filled with smoke. While not so dense as to obscure total view, I dared not open the oven door. The smell was of burned baked goods, but it was not dense enough for the computer to get excited about-yet.

Our final objective was the cellar, where the computer was headquartered. It was quicker to go there than try to find and insert the digital card in the reset mechanism in the front hallway. The motion sensors in that area of the house were not as easily overcome as the simpler variety that

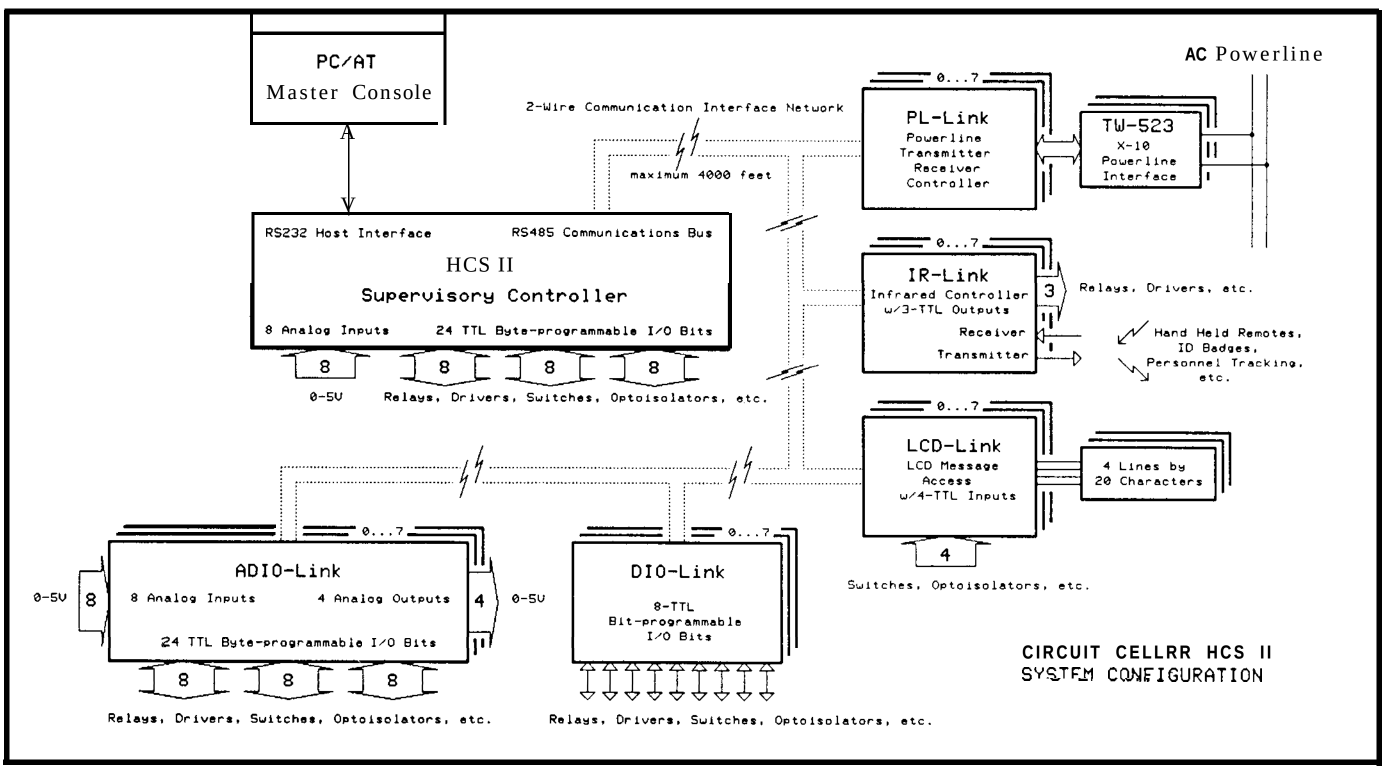


Figure 1--The *HCS II's* reach is *extended through* the use *of an RS-485* network. Five *kinds* of modules are *currently defined with room for more.*

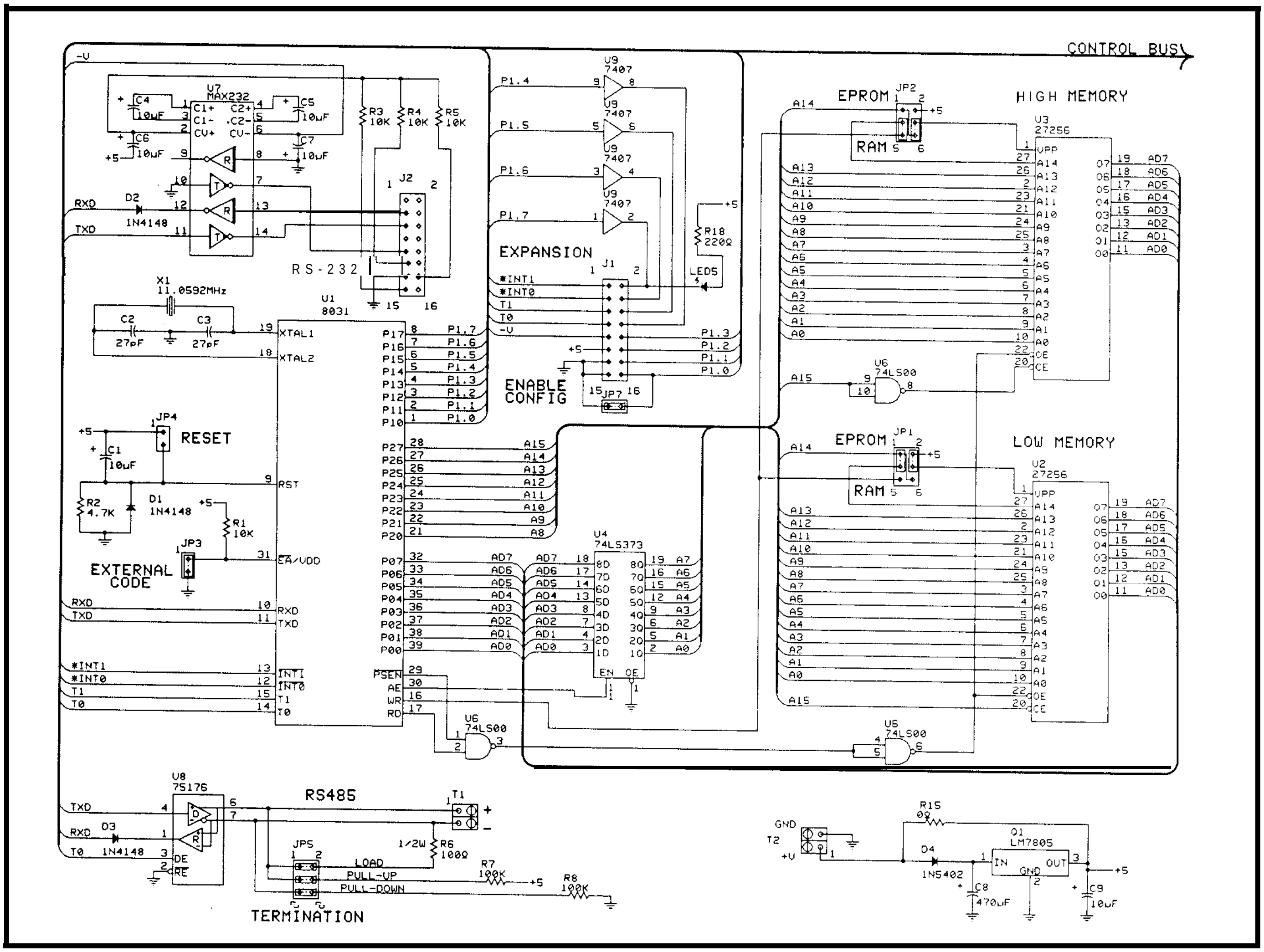


Figure 2— *The basic HCS II expansion module is the COMM-Link. Based on the venerable 8031, each unit has additional circuitry attached to Port 1 or the control bus to customize its operation.*

we had thus far defeated. The cellar door was but five feet and one pressure switch from the stove. We made it to this objective as easily as we had the others. There was no sensor on the door. I opened it slowly so that the squeaking of hinges would not reach an appreciable volume level.

When we opened the door, my two Scotties looked up at us. "No time to play now, guys," I said.

I went bounding down the stairs with Merrill in close pursuit. "When the alarm is armed, the dogs stay in the cellar. So there are just a few sensors down here. We're home free!"

Merrill and I stood in front of the computer control system. This computer did not have the usual panel full of flashing lights. That was old hat. The new stuff all had cathode-ray-tube displays. The monitor attached to the control system displayed a matrix of control parameters and on/off state. Peripheral sensors, not directly used to determine specific alarm conditions and still experimental, scanned the grounds like radar and displayed their activity around an outline of the house on another monitor. A dot flashed on the screen next to the outline. It slowly moved around the periphery of the house.

"That's Mrs. Picker," I pointed out to Merrill. "The computer knows she's out there. It has turned on the lights, but it will ignore her unless she goes over the fence into the backyard. See, she's moving in that direction now. I'll need about three minutes to enter the disarm commands."

Merrill looked around the cellar at all the equipment. Spying a refrigerator, he started to walk toward it. "Hey, Steve, why can't you just pull the plug on the computer?"

"That wouldn't do anything. In case of a power failure, the computer has battery backup and all kinds of redundancy."

I started to type in the first abort code. Merrill, who finally felt relaxed again, stood at the refrigerator and said, "Boy, all this work has really made me thirsty. Do you have any beer in here?"

He opened the refrigerator door. The fact that the refrigerator contained refreshment became immediately irrelevant. Suddenly a small speaker next to the computer started to emit a loud, repetitive sound: "beep...beep... beep...beep."

"Merrill! You triggered the alarm! It's going to go off in 10 seconds!"

My mind raced with the thoughts of things that were about to happen. The computer had sensed an intruder. Everyone but the National Guard would be here in 10 minutes. Large jetliners approaching the nearly airport would be distracted by the brilliant flashing lights and start to circle the house instead. They would find Merrill and me in a state of partial rigor mortis from the loud horns that would now go off inside the house. Finally, and most important, there was Mrs. Picker. If she was standing next to one of those sirens when it started, it would be curtains!

Merrill's eyes bulged with terror. Internally, he screamed, how could this be happening? Vocally, he yelled, "I thought you said that there were very few sensors down here because of the dogs! Why did it go off?"

Simple, yet true. We were done for, but hc still had to know. "Dogs don't open refrigerator doors. That's why."

The 10 seconds had almost elapsed. My final words were, "Hit the deck! Cover your cars!" That was exactly what we did. It was a hard-tiled cement floor, but we dove under one of my workbenches and covered our heads with our arms. Almost immediately, the beeping stopped. After about 15 seconds I peeked out. At 30 seconds we got up and walked over to the computer.

"I don't understand," I said. "It should have gone off. At the end of the beeping, it should have started the sirens and lights and everything. I don't understand."

I walked over to the console and started to list the program. "There must be a program bug or a loose wire in the back here someplace. Otherwise, it would have gone off." I busily typed on the keyboard as I spoke, "Gee, Merrill, that's a lousy demonstration of my talents. I'm a better programmer than that.

"Merrill, wait a few minutes, and let me see if I can fix it. Don't think this was all a waste of your time. I want you to know that this thing really works. Give me a minute or two to fix it, and I'll show you that the sirens really do go off."

Merrill didn't wait. He gave me a fierce glance and took off up the staircase. I yelled, "Where are you going? Don't you believe this will work?"

Merrill yelled down the stairs. "I'll be right back. I'm just going to borrow Mrs. Picker's cane!"††

CONTROL IS CONTROL

The phrase "Home Control System" (HCS) means different things to different people. To the average nontechnical person it might symbolize the Frankensteinian Machine depicted in my introduction. I use the phrase home control in this project for lack of a better term, but control is control, whatever the application.

If we purchase a single-board controller like those sold by many of the advertisers in ***Circuit Cellar INK, we*** tend not to qualify the application. Connecting photosensors to the digital input lines, solenoids to the output drivers, and writing a program for sorting materials on a conveyor belt surely would not be a strange use. However, program the same photosensors to interrogate a punched ID card to control your front door lock and it appears eccentric.

"Home Control," like automated industrial control, is merely the decision-based execution of controllable events that would otherwise have to be done manually. The singular factor that determines the size and architecture of either system is the quantity of controllable devices and the number of monitored events.

A home controller is a dedicated application of what we might view as a downsized version of the industrial variety. As the control requirements expand beyond just lighting control to include environment, security, and entertainment electronics, the control task list can become so complex that the distinction between industrial and home controller architecture becomes less and less.

D	Dump program status (debugging use)			
E	Show and clear error flags (debugging use)			
F	Flush transmitter and receiver ring buffers			
Ln	Set logging mode (bit mapped)			
	L	report current mode		
	L0	disable (default)		
	L1	show received X-10 messages		
	L2	show transmitted X-10 messages		
	L4	show refresh changes		
Nn	Set network/interactive mode			
	N	report current mode		
	N0	set interactive mode		
	N1	network mode (no error messages) (default)		
	N2	network mode with command echo (no err msgs)		
P	Report power failure status: n,n			
	first digit = 1 if power is currently OFF			
	second digit = 1 if power failed since last P command			
Q	Query X-10 module status (see notes)			
	Q	report all modules for all housecodes as Qh00		
		(module 1 is first on each line)		
	QS	report all modules for all housecodes in hex		
		(housecode A, module 16 is first)		
	Qh00	report all modules for housecode h: h=sss..		
		(module 1 is first)		
		0=off	1=on/dim/bright	X=not used
	Qhmm	report module mm for housecode h: hmm=ss		
		OF=off	ON=on/dim/bright	XX=not used
R	Set refresh period (see notes)			
	R	report current period in seconds		
	Rmm	set period in minutes		
	RSss	set period in seconds		
	RCmm	clear refresh buffer, set period in minutes		
	RQ	dump contents of refresh table		
RESET	Perform power-on reset			
S	Send X-10 message (see notes)			
	Shmmff	send function ff to housecode h module mm		
	Sh01AF	send ALL UNITS OFF command to housecode h		
	Sh01AN	send ALL LIGHTS ON command to housecode h		
	Shmmffrr	send function ff to housecode h module mm		
		function message repeated rr times		
	(may be strung together: Shmmff,hmmffrr,hmmff)			

Figure 3—*The command set for the PL-Link allows bidirectional communication with X-10 devices on the power line.*

RATIONAL HOME CONTROL -MY OPINION

Before I get into my idea for solving the home control problems of the '90s, it's probably best if I take a minute to explain my personal philosophy and application of home control. It's one thing for someone to present an opinion, but it's better appreciated if you understand the facts upon which that opinion is rendered.

First of all, I am not a control fanatic. I look at my HCS as a convenience first and then as an extension to the security system. I do not look at it as an independent entity that I cannot live without (for too long, anyway] or one that totally compromises security if it fails. And above all, the HCS controls nothing in a way that would supersede local UL-approved appliance controls or would otherwise jeopardize home owner's insurance (I would turn on a circulator fan through its thermostat/contact closure input rather than provide power directly to the fan motor, for example).

Most HCS owners think of security as the predominant application for their system. Given the thousands of watts of lights, magnetic and infrared sensors, and tons of surveillance equipment I have around, you aren't wrong in that assumption here either. However, sensing intrusion and doing something about it are different problems. Most local police don't like dealing with voice synthesizers, taped messages, or false alarms from "user installed" security systems. Furthermore, booby traps, whether they are computerized or not, are illegal.

To live in a conventional society, sometimes you have to do some "conventional things." To take maximum advantage of insurance discounts, I had a commercial hardwired fire/security system installed before the first HCS wire was strung. I pay a certain fee per month for constant monitoring [the few times I have had to greet the police in the driveway have been for "righteous" causes). I merely "borrowed" all the signals from the commercial security system, and connected them (with lots of others) through optoisolators so the HCS could use them too.

I have no interest in pressing buttons as I enter the shower to automatically adjust the water temperature and blow dry myself to a Vivaldi symphony. I don't need a voice synthesized butler to tell me that my suit was automatically cleaned and pressed while I slept last night or to greet people at the front door. I don't need a system that keeps an inventory of the refrigerator and tells me how many calories I've consumed while reminding me of the consequences. And above all, I don't want to interact with anything that makes it appear that I'm living in "its" house.

Instead, I want to be able to walk into rooms without having to use manual light switches. I want a system to monitor otherwise independently controlled activities (like the furnace, wood stove, water pumps, electricity, flood/smoke/fire sensors, etc.] and indicate when these systems are or are not operating within preset parameters (stack overtemp, water in the basement, etc.). I want direct control of the entire system through a convenient remote control (display status, do this or that, set or reset, etc.) or otherwise have "it" know of my specific presence in certain situations.

Finally, while a sense of "being controlled" is something I abhor when I am present in the house, I want the opposite to be true when I am not. I want it readily apparent to anyone in the vicinity of the house that "it" is responsive and sentient and knows every move they make. [There are some funny stories about a few of these visits. To this day the oil delivery guy won't come unless I am home.)

THE TRANSITION FROM OLD TO NEW

The original HCS that I designed seven years ago was a first attempt to bridge the gap between low-cost open-loop home control and closed-loop industrial control. It was adequate up to a point but, in retrospect, it didn't go far enough.

The old HCS combined X-10 capability with sixteen inputs and eight outputs. Having inputs allowed connection of motion detectors, floor switches, water sensors, and so forth. The basic concept was to allow control events to occur as the result of a monitored input and not just limit control to turning things on or off at specific times [e.g., walk into a room with a motion detector and the light would go on for a preprogrammed duration). Walk into the Circuit Cellar and the terminals and equipment will automatically turn on.

Unfortunately, this first HCS didn't go far enough. It could react to an input but it could not make Boolean decisions based on a number of inputs (IF input 1 is set AND input5 is set THEN turn on output8) or deal with analog inputs such as temperature limits. I got around this deficiency by using a separate computer to make these decisions and then have that unit direct the HCS to turn on specific outputs. But after adding two of these outboard decision modules, I was both out of mounting space and HCS inputs. Oh, what I wouldn't have given for another 24 input lines.

The final straw was wiring. To sit in your easy chair and press the X- 10 button for the floor lamp is one thing. If it doesn't go on, you'd press it again, probably look to make sure it was plugged in, try it manually, and then replace either the bulb or the X-10 module.

X- 10 control is accomplished by transmitting a code through the power line. It is a very affordable control medium for some things, but it is susceptible to false triggering or a failure to trigger [when the oil burner is firing in my house, X- 10 communication is about 50/50). Except for sitting next to a light and seeing that it did go on, you cannot presume with 100% assurance that the device you transmitted the code to actually turned on.

Some control connections must be closed loop. The loop is closed by turning on the device, sensing that the device actually turned on, and setting an alarm or taking alternative action if it did not. All this involves directly wiring sensors and controls to the HCS.

For obvious reasons, simple house lighting doesn't warrant this much effort but certain systems do. The pumps and valves in a solar heating system, a backup battery charger, or an emergency sump pump are important enough to control closed loop.

The problem I ran into was that having a single central control required all these wires to come back to one place. That was all right when only the house was considered, but now I have the house and five other buildings (greenhouse, a couple garages, and storage), and extending direct wired HCS control (beyond simple X-10) to them involves more control wires than were initially laid in.

HCS II-AN EXPANDABLE ARCHITECTURE

The solution was to design a new system from the ground up. Because my personal application is still domestic, I refer to the new system as the Circuit Cellar HCS II for no better reason. However, after you review the architecture and see what this entity actually does, you may agree that HCS here stands for Humongous Control Scheme.

First of all, HCS II is not a home controller. It is an expandable network-based intelligent-node industrial-oriented supervisory control system that, in its minimal configuration, performs quite suitably as a home control system (see Figure 1). HCS II incorporates direct digital inputs and outputs, direct analog inputs and outputs, real-time or Boolean decision event triggering, X-10 transmission or reception, infrared remote control transmission and reception, remote displays as well as a master console, and has the capability to perform as a complete badge monitoring and personal tracking system.

The HCS II system architecture consists of a central supervisory controller connected to up to 31 other functional modules (called links) via an RS-485 serial network. The system controller and the links can operate independently and do not need the system controller or other links to function, which allows easy testing or incorporation as intelligent subsystems in other control equipment. Most subsystem links share a common 8031 controller board (generically called a COMM-Link) with the I/O customized for each application. Currently, we have completed the PL-Link, IR-Link, LCD-Link, DIO-Link, and the ADIO-Link. A Circuit Cellar HCS II system need only include those functions that suit the tasks to be performed. I'll explain each module shortly.

A LOT TO TELL

For obvious reasons, designing and explaining all the hardware and software of the Circuit Cellar HCS II has to be a team effort. It is a project that has been completely designed and constructed by the ***Circuit Cellar INK*** engineering staff and all of us have some part to tell.

In this issue I've described how the HCS II came about and the basic controller design used in the networked subsystems. Ed picks it up immediately by explaining how to add the X-10 TW523 module and control software to make the HCS II's smart X-10 PL-Link.

As an essential member of the team, and the only one who knows how this event sequence compiler can ever work, Ken describes the hardware and software specifics of the Supervisory Controller. Next issue he'll finish up with a description of the compiler software.

Also in the next issue, Ed and I will describe the IR-Link section and its uses as an HCS IR remote control input, as a full-featured badge reader, and as the essential ingredient in a complete personnel tracking system.

Finally, Ed finishes out the series with descriptions of the LCD-Link, DIO-Link, and ADIO-Link interfaces and software.

THE HCS II SUPERVISORY CONTROLLER

The supervisory controller is quite literally the brains of the system.

```
All time intervals are in units of 5.16 ms

B       Set badge response delay (default 40 ticks)
CRn     Calibrate remote MC1 46030 bit clock (12.4 kHz,1290 µs)
            n = O-3 sets report detail
CT      Calibrate transmitter oscillator
            connect 38-kHz output to T1 input
D       Dump program status (debugging use)
E       Show and clear error flags (debugging use)
In      Set badge polling interval (default 190 ticks)
Ln      Set logging mode (bit mapped)
            L       report current mode
            LO      disable (default)
            L1      show received IR messages
            L2      show transmitted IR messages
            L4      show generated polls
Nn      Set network/interactive mode
            N       report current mode
            NO      set interactive mode
            N1      network mode (no error messages) (default)
            N2      network mode with command echo (no err msgs)
Pn      Set number of badges to poll (default 0, no polling)
On      Set output bit 1 = high, 0 = low, Default = 1
            Oa is bit P7.6
            Ob is bit P7.5
            Oc is bit P7.4
Q       Query and reset received IDs
            Q       dump all 512 IDs in hex bvtes (about 132 chars)
                        lb 0 is bit 0 of first byte
            Qn      report ID n status in format 003 = 1
            Qn-m    report IDs n through m in hex bytes
                        ID n is bit 0 of first byte
RESET   Perform power-on reset
Sn      Send IR ID n (decimal, O-51 1)
```

Figure 4—*The IR-Link allows* **two-way infrared** *communication in an* *HCS II setup.*

While each link has its own processor, they all rely upon the supervisory controller for timing, control commands, and overall system coordination.

The supervisory controller is a 9.216.MHz HD64180-based single-board computer. It has two serial ports; up to 96K bytes of memory (32K bytes are battery backed); an 8-channel, 8- or 10-bit ADC; a real-time clock/calendar; and 24 bits of parallel I/O (an additional 48 bits of parallel I/O can be added if required). These parallel I/O bits can be further conditioned by externally connecting them to optoisolators, relays, or drivers. You may find that the supervisory controller is all you need depending on your application.

One of the serial ports performs as a dedicated 9600-bps serial RS-485 connection to the various network links. The wire from this port can be up to 4000 feet long (as opposed to 50 feet for RS-232). The second port connects (RS-232) to a PC/AT that functions as the master console. Programs running on the master console allow the user to enter and store control sequences, which will be subsequently compiled and downloaded to the supervisory controller's nonvolatile memory. The master console also acts as an on-line real-time display of all control activity in the system. When these downloading or display functions are not required, the master console can be turned off to reduce system power consumption. See page 46 for more on the Supervisory Controller.

COMM-LINK-INSIDE EACH NETWORK NODE

Besides the Supervisory Controller, the HCS II system consists of a multitude (0 to 31) of specialized-function network nodes that facilitate remote data acquisition, closed-loop control, and display. With the exception of the ADIO-Link, all COMM-Links consist of the same generic 803 1 computer with only the external parallel I/O circuitry and operating software being different (the ADIO has additional I/O address decoding and bus buffering).

Figure 2 outlines the generic computer section common to all network links: Its layout is typical of many previously demonstrated 11.0592-MHz 803 1 -based controllers that we have used in other projects, except this one is configured with minimal components. It accommodates up to 64K bytes of memory configured as either RAM, EPROM, or both. In general U2 will contain an 8K-byte RAM (nonvolatile in some applications) and U3 will have a 32K-byte EPROM.

The single serial port is configured to operate both RS-232 and RS-485 simultaneously. When connected to the Supervisory Controller in normal operation, the connection is via a twisted pair to the RS-485. Additional line balancing and termination resistors are included. According to the driver chip manufacturers, this line can be 4000 feet long. In normal operation only RS-485 is used and the MAX232 (U7) can be removed to reduce power.

The available I/O on the COMM-Link circuit is limited to the Port1 bits of the processor, interrupt lines, TO, and T1. When used as a power-line interface, these lines connect to the TW523 X-10 module; when used as an infrared gateway, these lines connect to infrared LED driver logic and an IR receiver; when used as an LCD display, these lines supply character data to the LCD. In addition, the COMM-Link PC boards contain a small prototyping area to facilitate additional link designs or modifications to the existing circuits.

The RS-232 is reserved for local nonsystem direct testing or use. Each link has a unique command set and, for multiple units of the same type, a unique address. By connecting the link to the serial port of a terminal or any PC running a terminal emulation program, you can directly command the COMM-Link to do any activity that it would normally do for the Supervisory Controller. For example, with the PL-Link alone you could directly control all X- 10 on/off dim/bright functions from a PC. In addition, because the PL-Link is "smart," it offers the user the added capability to "listen" to the power line and record whether any other X- 10 codes have been transmitted (either manually or automatically, from this or any other transmitter), and it will automatically "refresh" desired X- 10 modules at a prescribed refresh rate.

D	Dump program status (debugging use)
E	Show and clear error flags (debugging use)
Ln	Set logging mode (bit mapped)
	L report current mode
	LO disable (default)
	L1 show ANSI decoding sequence
	L2 show LCD command processing
Nn	Set network/interactive mode
	N report current mode
	NO set interactive mode
	N 1 nehvork mode (no error messages) (default)
	N2 network mode with command echo (no err msgs)
Q	Query buttons, only presses since last Q are reported
	Buttons are bit-mapped in hex byte
	Current hardware returns buttons 80, 40, 20. 10 only
RESET	perform power-on reset must be completely spelled out!
S=string	Send string to LCD panel via ANSI decoder
	(string continues to end of line)

Figure 5— The LCD-Link supports a *subset* of *the standard ANSI terminal command sequences in addition to debugging commands.*

The reason for making these links intelligent is two-fold. First, testing and problem diagnosis is greatly enhanced by being able to deal with an individual unit in a convenient way. Secondly, while I outlined my reasons for redesigning the HCS II, that doesn't necessarily make it useful for everyone. I don't expect the Supervisory Controller to be adequate for all process control applications, but these control subsystems might be all you need to build an even more grandiose control scheme.

I expect that some of you will see the COMM-Link perhaps as the missing "link" (no pun intended) in your own personal or professional control applications. Need an X- 10 gateway for a project? Use a PL-Link. Need a badge entry system for the file room or a non-X-lo-related remote control input to your PC or Mac? Use an IR-Link. Need an LCD readout a mile from the computer? Use an LCD-Link. Need to read temperature and wind direction and set control outputs at the top of a 300-foot tower? Use an ADIO-Link and one twisted-pair wire.

Finally, one minor detail about the power supply. One exasperating problem I ran into with the previous HCS was having to power locally all the external sensors. More often than not, the place where I wanted to put something had no convenient AC power outlet and 5 volts was no longer 5 volts if I ran power wires from where it was convenient.

While the COMM-Link circuitry runs on 5 volts, the design incorporates a linear regulator, so the 5 V can be derived from a wide-ranging DC input. The intention is to promote connecting the individual links via a four-wire cable, rather than just a two-wire twisted pair, when local power is unavailable. Two wires supply RS-485 while the other pair supplies + 12 V and ground. (Actually, 9-12 V is fine. Higher voltages requires a larger heat sink on the regulator.) Of course, using the regulator is optional, and the board can be operated (jumper selectable) on +5 V.

Now that you know the basic hardware of the links, the functions and capabilities of each of the network links is, as Ed would say, "a simple matter of software."

PL-LINK-SMART X-10 CONTROL

The first subsystem you might add to configure the traditional home control system would be a PL-Link. This network node performs X- 10 activity. While the HCS II accommodates multiple links of the same type, we presume you would only have one PL-Link in a system in a normal configuration.

The hardware consists of a basic COMM-Link board (Figure 2) with the X- 10 interface section added. A TW523 power-line interface module is plugged into the PL-Link's RJ-11 connector (the TW523 plugs into the AC power line). The software for the PL-Link and the schematic for the X- 10 interface are outlined in detail in Ed Nisley's "Firmware Furnace" column starting on page 74 in this issue. A summary of the PL-Link's commands are shown in Figure 3.

IR-LINK-SMART INFRARED CONTROL AND ID SYSTEM

I don't want to tell everything before next issue's description of the IR-Link because it will take a lengthy explanation. Basically, the IR-Link works via the same RS-485 link as the other network nodes to send and receive data. In this instance, the data consists of wireless infrared transmissions, which are sent and received by the IR-Link board.

The IR-Link board contains a 38-kHz infrared transmitter, a 38-kHz infrared receiver, and three output bits The data format it transmits and receives is called Manchester coding. A single transmitted character consists of nine bits and represents one of 5 12 combinations (multiple characters can be used to represent significantly more combinations if necessary).

The idea behind the IR-Link is to use a trainable handheld IR remote controller, like the one you might already be using with your TV set, and have it contain 10 or 20 (or all 5 12) of these codes. Simply aim the remote at the PL-Link and that code would be received by the HCS II's Supervisory Controller. Within the supervisory controller's event repertoire would be events such as: IF ircode=20 THEN set pump ON, light6 OFF, fan1 ON; or, IF ircode= 1 THEN all lights ON.

Anything that can be done on an existing X-10 remote control can be incorporated within the functions of the PL-Link. Non-X- 10 activities can be combined as well. A beeper could sound, indicating that you have company in the driveway: without getting up you could direct the driveway camera's video to the TV you're watching and turn on a series of greeting lights at the front door, or you could direct the lights to slowly dim while the electronic barking dog volume increases (don't you guys have all this stuff installed already?].

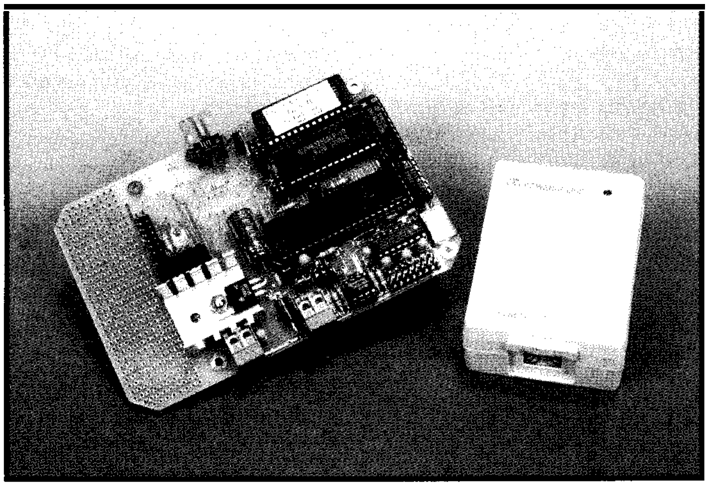
Given 5 12 interpretable commands, your handheld control could do quite a bit.

I'm sure you've heard the phrase, "a funny thing happened on the way to the office." Well, this was no exception. When Ed and I concluded that the most cost-effective way to provide a handheld remote for the PL-Link was to use one of the commercial trainable units rather than build our own hardware interface, we created a dilemma. The only way to train our remote was if we added an IR LED transmitter section to the IR-Link and provided commands to send the codes as well.

Because IR-Link has the capability to both send and receive Manchester codes, it can serve as the basis for a simple "people tracking" system based on either active or passive badges (I'll detail some badge hardware next issue). In this context, active badges send their ID periodically without prompting, and passive badges respond to a poll from the tracking system.

Active badges require no special setup because the IR-Link will record all IDs seen by its sensor. Each ID is cleared after it is reported, so each Query command returns new IDs since the last Query. Passive badges must receive their ID to trigger the transmission of that ID. The IR-Link can send IDs (transmitting high powered IR throughout a room requires additional circuitry on the IR-Link board) at regular intervals and will record the response; a polled ID is no different from any other ID. The IR-Link does not record its own transmissions, so the outgoing and incoming IDs do not conflict. HCS II can also accommodate up to eight IR-Link units on its RS-485 line, so these links can be placed in separate locations to broaden the tracking area.

There are many potential uses for the IR-Link beyond ID system and remote control receiver. Figure 4 contains a list of the direct commands for the IR-Link.



A *COMM-Link with some extra parts and a TW523 becomes a PL-Link.*

LCD-LINK-REMOTE DISPLAY UNIT

The LCD-Link provides HCS II with a remote data display capability. It consists of the same COMM-Link processor with the firmware programmed to receive ASCII strings from the Supervisory Controller and display them. The LCD is 4 x 20 characters and up to eight independently addressable LCD-Links can be connected within an HCS II system. In addition to the LCD display, the LCD-Link has four function buttons that are read as a four-bit value (you can interpret this feature as four function buttons or sixteen functions defined by four buttons, depending on how you write your event sequence commands). Figure 5 shows a list of the LCD-Link commands.

DIO-LINK-DIGITAL INPUT/ OUTPUT INTERFACE

There isn't a whole lot to say about the DIO-Link. Like the LCD and IR units, up to eight can be accommodated within a system. Its hardware is the COMM-Link circuit again with the "DIO" referring to the eight bits of Portl. The eight lines are "bit-programmable": each line can be defined as an input or as an output. However, because these are direct processor bits we suggest that the prototyping area on the link board be used to add custom signal conditioning.

There is also a strobe bit that indicates when new data is available, so you can simulate a printer port for logging or other purposes.

The typical use for this network link is for remote data collection (what are the current logic settings of the relays at the top of the elevator shaft) or closing the control loop without running sensor wires all the way back to the controller.

ADIO-LINK-ANALOG AND DIGITAL INPUT/OUTPUT INTERFACE

The ADIO-Link is even more of a good thing. Rather than just eight digital I/O bits, the ADIO-Link accommodates analog I/O as well. Each ADIO-Link (there can be as may as eight in the system) has 24 TTL-level parallel bits (byte programmable as input or output), 8 channels of 8-bit or optional 10-bit A/D input (O-5 V), and 4 channels of 8-bit D/A output (O-5 V).

IN CONCLUSION

I've tried to explain HCS II in such a way that perhaps its capabilities and your control tasks can overlap into a perfect match. Given the sheer magnitude of descriptions, I hope I haven't overwhelmed you. I think we've succeeded in creating a flexible architecture that addresses today's control requirements and will also accommodate tomorrow's Perhaps it won't even be that long before we demonstrate a CEBus-Link for HCS II. Stay tuned.

tt This story was adapted from the chapter called "Computer On Guard" in "Take My Computer...Please!" by Steve Ciarcia and is reprinted by permission of Scelbi Computer Consulting Inc.

Steve Ciazcia is an electronics engineer and computer consultant with experience in process control, digital design, and product development.

IRS

404 Very Useful
405 Moderately Useful
406 Not Useful

The theme of this issue of ***The Computer Applications Journal*** is building automation, but there is barely enough space to describe the Supervisory Controller and the PL-Link here. The problem with spreading the design and explanation of such a significant topic over a series of articles is the frustration of wanting to use the finished system as soon as the idea is presented.

While the HCS II design description is scheduled to be spread out over six months' worth of magazines (suggesting we still have that much time to design everything], as engineers we knew we had to fully define the entire system before we could accurately present the first article. HCS II is completely designed and PC boards are being fabricated at press time. It seems only reasonable to offer this hardware as it becomes available for those readers who can't wait to build it from scratch when the articles are presented later this year.

The following is available from:

Circuit Cellar Kits
4 Park St.
Vernon, CT 06066
Tel: (203) 8752751
Fax: (203) 872-2204

Item 1. HCS II Supervisory Controller
HCS II 4-layer PC board and all components including HD64180 processor, ADC, real-time clock, battery-backed RAM, IC sockets, multitasking controller firmware in EPROM, HCS event compiler on PC diskette, and user's manual.

Complete kit	order HCSIIK- 1	$199.00
Assembled and tested	order HCSIIA- 1	$269.00

Item 2. PL-Link Smart X-10 Powerline Transmitter/Receiver
PC board and all components (excluding TW523) including 803 1 processor, RAM, IC sockets, PL-Link firmware in EPROM, and user's manual.

Complete kit	order PLINK- 1 K	$99.00
Assembled and tested	order PLINK- 1 A	$159.00
TW523 X-10 power line adapter (not sold separately)		$30.00

Item 3. IR-Link Smart Infrared Transmitter/Receiver
PC board and all components including 8031 processor, RAM, IC sockets, infrared LED, 38-kHz infrared receiver module, IR-Link firmware in EPROM, and **user's manual.**

Complete kit	order IRLINK- 1 K	$119.00
Assembled and tested	order IRLINK- 1 A	$169.00

Item 4. DIO-Link Smart Digital I/O Network Interface
PC board and all components including 8031 processor, RAM, IC sockets, DIO-Link firmware in EPROM, and user's manual.

Complete kit	order DIOLINK- 1K	$99.00
Assembled and tested	order DIOLINK-1A	$159.00

Item 5. LCD-Link Smart 4 x 20 LCD Display Network Interface
PC board and all components including 803 1 processor, RAM, IC sockets, LCD-Link firmware in EPROM, and user's manual.

Complete kit	order LCDLINK- 1 K	$99.00
4-line x 20-character LCD	order LCD25/4	$59.00

Item 6. ADIO-Link Smart Analog/Digital I/O Network Interface
All components including 8031 processor, 8-bit ADC, RAM, 24 bits I/O, IC sockets, ADIO-Link firmware in EPROM, and user's manual.

Complete kit	order ADIOLNK-1K	$199.00
Assembled and tested	order ADIOLNK-1A	$299.00
4-channel, S-bit DAC upgrade		$35 .00

Miscellaneous

12-V modular power supply for link units	Link-Pwr	$4.00
5-V 0.9-A modular power supply for HCS II	PS-11	$19.00

All items are shipped FOB Vernon, Connecticut and shipping is **extra.** All assembled and tested units come with l-year limited warranty. A repair service is available to kit builders at an hourly charge. HCS II, Link designs, and software are available for commercial license.

The Home Control System II Supervisory Controller

FEATURE ARTICLE

Ken Davidson

Current home control systems have a level of functionality not seen in systems of a few years ago. The key building block and brain of the new Circuit Cellar HCS II leaves its predecessor in the dust.

As engineers, I'm sure most of you have no doubt witnessed the birthing of a new industry niche. Electronics in general have become more sophisticated during the last 30 years, and the dawning of the microprocessor certainly spawned a giant evolution in the way control problems are approached.

Software has become more mature since the days of punch cards and FORTRAN. Now entire teams of programmers are needed to complete a project that meets the expectations of today's computer user in terms of features, speed, and user interface.

So too are we witnessing the slow maturation of the home automation industry. When the Circuit Cellar Home Control System was first introduced almost seven years ago, there was nothing quite like it available for the price. It allowed you to turn lights and appliances on and off based not only on time of day, but also on direct inputs. A motion detector could turn on a light in a room, leave it on for 20 minutes, then turn it off, all with simple, menu-driven programming.

The times, they are a' changin', and home control systems are starting to grow up. The features of the original HCS, while simple to use, are far too simple for the growing expectations of today's home automatist. Suppose you want to turn on a light for 20 minutes based on sensed motion, but not while you're trying to sleep. The original HCS couldn't handle the task alone. It required external conditioning of the motion detector signal to block it during the times when motion was to be ignored.

Seeing a need to update the features of the original HCS, I started work on the software for such a system several years ago (as many BBS users may recall). I brought it to a level that was comfortable for me and my living environment, but no further, which meant limited expandability and a hostile user interface. When we decided to get the whole engineering staff involved, the project took off at such a pace I'm still reeling. Add a decent user interface, allow expansion of the system through a local network, add support for more types of devices, the list goes on.

Instead of overwhelming you with all the details, we're going to break the system description into smaller, more digestible parts (as Steve explains in his introductory article). In this article, I cover the "Supervisory Controller," which is the brains of the operation. I'll start with the hardware, then get into the complicated stuff: the control software.

INDEX CARD CONTROL

Figure 1 shows the complete schematic of the Circuit Cellar HCS II Supervisory Controller (SC) board. The SC board measures just 3.5" x 5" (see Photo 1) and includes up to 96K of memory, one RS-232 port, one RS-485 port, 32 bits of TTL I/O, eight channels of either 8- or IO-bit A/D conversion, and a small amount of EEPROM. With the addition of a Dallas Semiconductor SmartWatch, we battery-back 32K of the RAM and add a clock/calendar.

Everyone has a favorite microprocessor. Ed concentrates [often grudgingly) on the 803 1 for its functionality, inexpensive price, and low power consumption. Other people I know swear by Motorola processors to get the job done. I cut my teeth on the 8080 and have since graduated to the HD641 80 as my processor of choice (when I have a choice).

The basic HD64 180 isn't technically a microcontroller because it has no provision for on-board memory (later versions do, however). It does include a large array of on-board

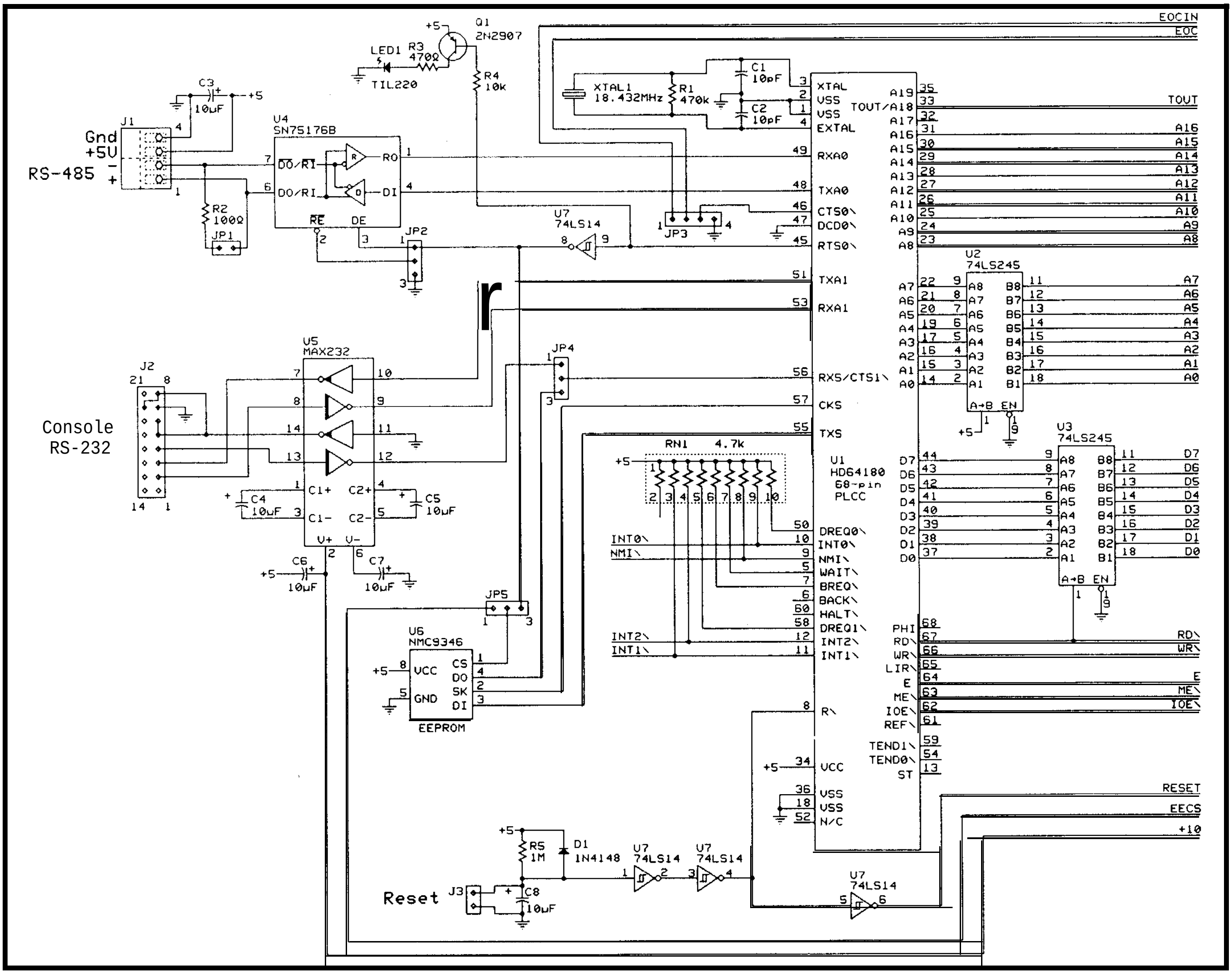


Figure 1 a--The Supervisory Controller uses an ***HD64180*** **as *its* brain. Both RS-232 and RS-485 are supported for communicating** *with the outside world.*

peripherals including two asynchronous serial ports, two timers, two DMA channels, and a memory management unit that allows it to address up to one megabyte of memory. The instruction set is identical to that of the 280, with the addition of a few new instructions (including an d-bit multiply).

Figure 1 a shows the processor, serial expansion, and EEPROM. A MAX232 (U5) provides a full RS-232 port, which allows connection to the host computer. The MAX232 obviates the need for power supplies other than +5 V. A standard RS-485 port is supported using an SN75 176B (U4). With the RS-485 interface, up to 32 devices may be connected to a single twisted-pair wire having a total length of up to 4000 feet. JP2 controls whether the receiver is always enabled, or is only enabled when the transmitter is disabled. When the receiver is always enabled, we will receive anything we transmit. This echo may be good or bad depending on the application, so it is left an option. Per the RS-485 specification, a 100-ohm terminating resistor must be installed when the board is at the end of the wire. The resistor is put in the circuit by installing JP1.

The NMC9346 1024-bit EEPROM is a synchronous serial device that communicates over four lines: data out, data in, serial clock, and chip select. The HD64 180's clock serial port is a near perfect match to the protocol used with the 9346. A bit of tricky programming is required to conform, but in the end it works quite nicely. We don't currently use the EEPROM in the HCS II code, but the 8-pin device didn't take much room, and we saw a lot of potential in future revisions, so we put it on the board as an option.

Figure lb shows the memory, I/O, and decoding. The SC board contains three sockets for 32K static memory devices. Socket U9 must contain a 27256 EPROM and socket U1 1 must contain a 62256 RAM. Socket U10 may be set up for either EPROM or RAM by setting the pair of jumpers on JP6.

Twenty-four bits of parallel TTL I/O are supported on the board using the venerable 8255 PPI (U12). Ports A and B are byte programmable for input or output operation (i.e., all the bits of each port must be set for the same direction]. Port C is nybble programmable (bits O-3 may be set independent of bits 4-7). Note the raw TTL bits are brought out to header J4. If you plan to tie signals to that header, be sure the signals are properly conditioned or you will end up with a dead PPI the first time some static electricity courses through the system.

To accommodate analog inputs, an ADC0808 (U13) 8-channel, 8-bit A/D converter is on the board. A Siemens SDA0810 8-channel, 10-bit ADC may also be plugged into the same socket for somewhat better resolution. Note how the 5-volt reference is being generated. We use an LM336Z-5.0, but it can't be run-off 5 V. Instead, we steal a small bit of current from the high-voltage output of the MAX232. That output typically runs between 7 and 10 volts, depending on the current being drawn by the rest of the chip, and it works quite nicely.

An AmPAL22P10 (U8) does all the grunt work of generating chip selects for everything on the board. Figure 2 shows the PAL's equations. The more popular 22V10 may also be used, but the 22P10 is much cheaper. For those of you who don't have access to a PAL programmer, Figure 3 contains a schematic of the equivalent circuitry. Note how five discrete chips plus lots of wires have been shrunk to fit in a single 18-pin chip.

Active-low chip selects are formed that place the memory devices within the HD64 180's memory space (U9 covers 00000H–07FFFH, U10 covers 08000H–0FFFFH, and U1 1 covers 18000H–1FFFFH). The HD64180 supports separate memory and I/O spaces, so the rest of the devices on the board are mapped to the I/O space. The PAL generates an active-low chip select for the PPI at I/O address 8000H. The ADC requires separate active-high signals to start the conversion and to read the results. The ADC is mapped to I/O address 9000H. Also decoded are three I/O selects that go active in the ranges of AOOOH-BFFFH, C000H–DFFFH, and EOOOH-FFFFH. These selects are brought to the expansion headers to make the decoding on peripheral boards easier.

I think the only feature I haven't mentioned yet is the sleek crimson human-visual sanity-check feedback indicator: an LED tied to the transmit enable line of the RS-485 transceiver. Nothing like a quick blink a few times a second to give you that warm fuzzy feeling of knowing the system is still alive.

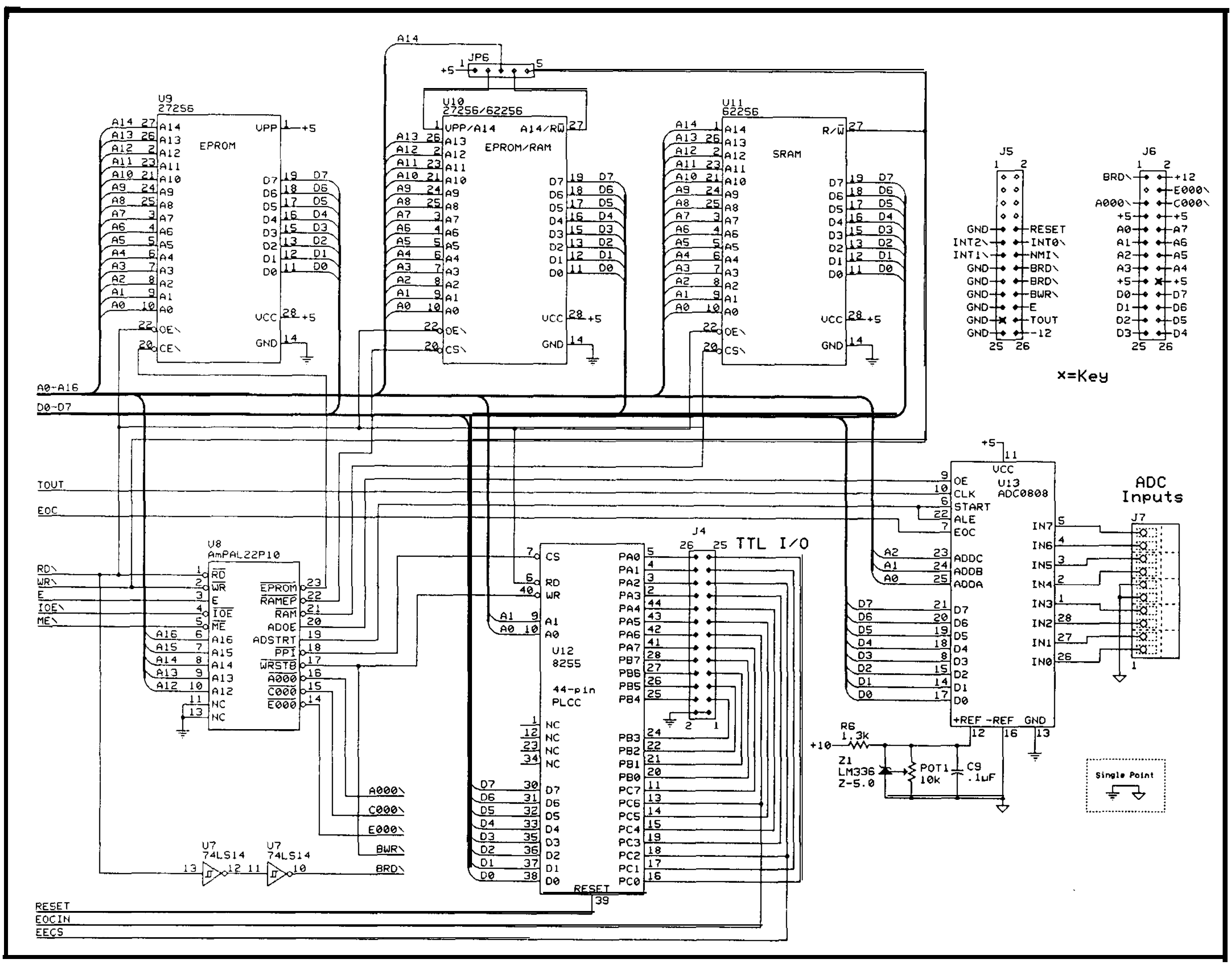


Figure 1b—*The second half of* **the Supervisory** *Controller* contains up *to three* static memory *devices totalling 96K, eight channels of A/D conversion, 24 bits of TTL I/O, and a custom PAL that provides all the decoding.*

```
"I/O address and function decoder for the SC processor board"
"Copyright (c) 1991 by Circuit Cellar, Inc."
"Version 1 .0"
DEVICE SC (P22P10)
PIN
  /RD    = 1 (INPUT combinatorial)
  /WR    = 2 (INPUT combinatorial)
  E      = 3 (INPUT combinatorial)
  /IOE   = 4 (INPUT combinatorial)
  /ME    = 5 (INPUT combinatorial)
  A16    = 6 (INPUT combinatorial)
  A15    = 7 (INPUT combinatorial)
  A14    = 8 (INPUT combinatorial)
  A13    = 9 (INPUT combinatorial)
  A12    = 10 (INPUT combinatorial)
  /E000 = 14 (OUTPUT active_low combinatorial)
  /C000 = 15 (OUTPUT active_low combinatorial)
  /A000  = 16 (OUTPUT active_low combinatorial)
  /WRSTB = 17 (OUTPUT active_low combinatorial)
  /PPI   = 18 (OUTPUT active_low combinatorial)
  ADSTRT = 19 (OUTPUT active-high combinatorial)
  ADOE = 20 (OUTPUT active_high combinatorial)
  /RAM   = 21 (OUTPUT active_low combinatorial)
  /RAMEP = 22 (OUTPUT active_low combinatorial)
  /EPROM = 23 (OUTPUT active_low combinatorial)

"Logic Equation Section"
BEGIN
  ENABLE(EPROM);
  ENABLE(RAMEP);
  ENABLE(RAM);
  ENABLE(PPI);
  ENABLE(AOOO);
  ENABLE(COOO);
  ENABLE(EOOO);
  ENABLE(ADOE);
  ENABLE(ADSTRT);
  ENABLE(WRSTB);
  EPROM = ME . /A16 . /A15;
  RAMEP = ME * /A16 . A15;
  RAM = ME * A16 * A15;
  PPI = IOE * A15 * /A14 * /A13 * /A12;
  A000 = IOE . A15 * /A14 . A13 * /A12;
  C000 = IOE * A15' A14 * /A13 * /A12;
  EOOO = IOE * A15 * A14 * A13 * /A12;
  ADOE = RD * IOE * A15 * /A14 * /A13 * A12;
  ADSTRT = WR * IOE * A15 * /A14 * /A13 * A12 * E;
  WRSTB = WR . E;
END.

TEST-VECTORS
  IN /RD,/WR,E,/IOE,/ME,A16,A15,A14,A13,A12;
  OUT /E000,/C000,/A000,/WRSTB,/PPI,ADSTRT,ADOE,/RAM,/RAMEP,/EPROM;
BEGIN
  1101000000HHHHHLLHHL;
  1101001000HHHHHLLHLH;
  1101011000HHHHHLLLHH;
  1100101000HHHHLLLHHH;
  1100101010HHLHHLLHHH;
  1100101100HLHHHLLHHH;
  1100101110LHHHHLLHHH;
  0100101001HHHHHHLHHHH;
  1010101001HHHHLHHLHHH;
END.
```

Figure 2—Since PALs are programmable devices, the operation of the board can be modified to some extent simply be reprogramming We part.

YES, MASTER

User interaction with the SC is through its RS-232 port (which I'll call the host port). The SC has enough to do [more on that later) without having to worry about how the status screen is formatted, so we use an IBM PC compatible as a host system that handles such details. Short binary messages are exchanged between the

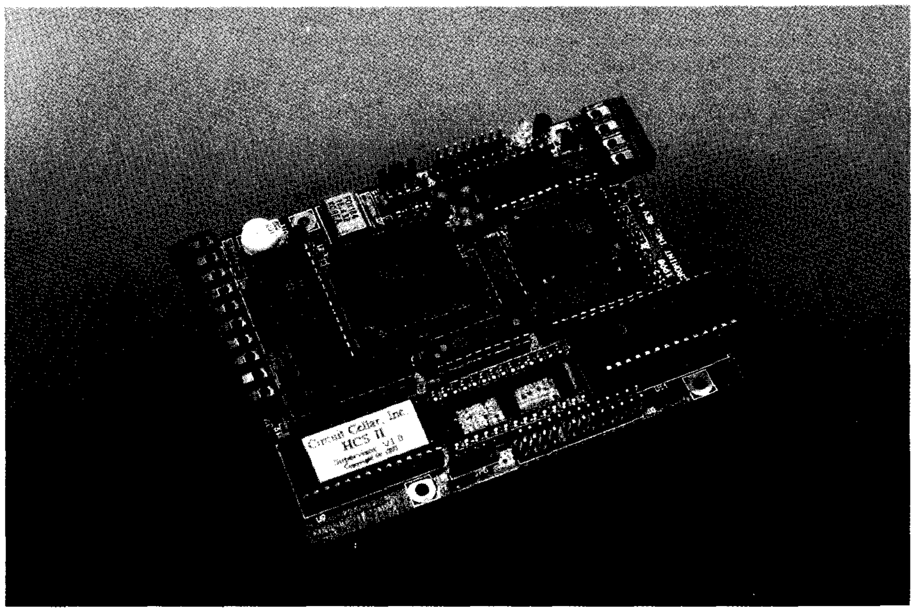


Photo 1—*The HCS II Supervisory Controller* **packs a** *lot of functionality onto a* 3.5" x 5" *board.*

SC and the host, which converts them to something pretty for the user to look at. The host also is responsible for creating the control program for the SC to execute. However, I'll cover the host in more detail in the next issue because I want to focus this article on the SC itself.

OUT IN THE BOONIES

The SC supports a large number of digital inputs and eight analog input channels on the board itself. But what happens when you want to place a sensor at the other end of the house, or you need a user terminal on another floor? RS-485 allows us to make a nice, low-speed local-area network to solve the dilemma without requiring you to outfit your house with miles of copper. A single twisted-pair wire may be run throughout the house, and up to 32 devices may be connected to it.

The network isn't intended to handle large amounts of information, so we keep the speed at a manageable

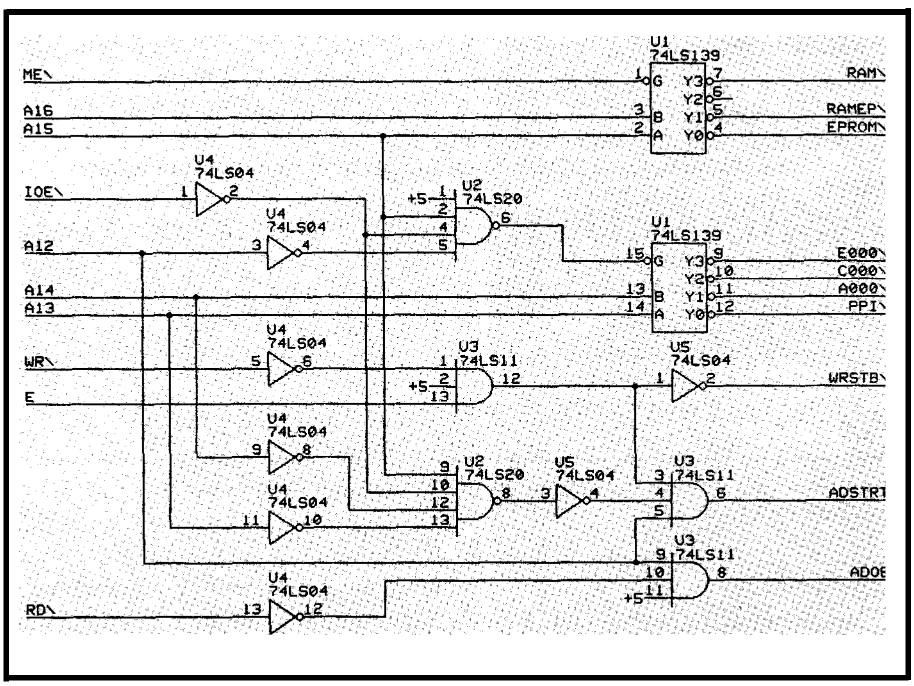


Figure 3-*Using a PAL to do the decoding reduces the chip count significantly. The equivalent circuitry needed to replace the PAL uses five drips and lots of wires.*

```
a)
        !_X10_SL01ON

b)
        $_X10_C02=OF
```

Figure 4—*(a) Network messages sent by the Supervisory Controller always start with an exclamation point This message is destined for the PL-Link and asks it to send an On command to module L10. (b) The response is formatted almost identically. The PL-Link is responding to a status query from the SC stating that module C02 is off*

9600 bps. When you're passing a limited number of 10- or 20-byte packets back and forth, the speed at which the bits travel becomes less important. Still, there can be a perceptible delay between a sensor being triggered and the resultant action occurring when communications take place over the net, so time-critical sensors should be connected directly to the SC whenever possible.

The SC is the only device on the network allowed to initiate communications. Only one device may have its driver enabled at any given time, and it's very difficult if not impossible to detect collisions, so strict discipline must be observed. No network module may speak unless spoken to (just like your kids at home, right?).

Network messages use strictly printable ASCII characters to make monitoring easier and to allow testing of network devices using a simple serial terminal. Figure 4a shows the basic format of a packet sent by the SC. It begins with an exclamation point, followed by a space, the name of the target device, another space, and the packet data. All packets are terminated with a carriage return. Response packets (shown in Figure 4b) are nearly the same in format, except the leading character is a dollar sign. The address contained in the response is identical to that in the SC packet so the SC can verify the correct module is responding.

I won't get into the details of how the individual network modules operate. Steve and Ed describe the first one elsewhere in this issue and will be discussing others in the next two issues.

A SIMPLE MATTER OF SOFTWARE

As I mentioned, the original HCS used a simple menu-driven "program-

Listing 1—*The Supervisory Controller is programmed using a series of 'event equations.'*

```
Program EventList1

BEGIN
   IF (Time >= 19:30:DY AND Time <= 23:30:DY) OR Timer(0)   ON THEN
      IF Input(0) = ON AND Module(L11) = OFF THEN
         Dim(L11.10):
         Dim(L9.14)
      END;
      IF Input(0) = ON AND Module(L11) = ON THEN
         StopTimer(0)
      END:
      IF Input(0) = OFF AND Modu e(L11) = ON THEN
         StartTimer(0)
      END;
      IF Timer(0) = ON AND Timer 0)>= 1200 THEN
         ModOff(L9);
         ModOff(L11);
         ModOff(L10);
         StopTimer(0)
      END
   END;

   IF Time = 16:30:DY THEN
      ModOn(L16)
   END

END.
```

ming" scheme that made for very simple, but very inflexible, operation. When I set out to design something new, I wanted a complete programming language that could handle complicated chores at my disposal.

I ended up basing the HCS II language on what I call the "event equation." The entire control program is made up of nothing more than a series of event equations. The equation consists of an "if" clause and a "then" clause. Within the "if" clause, the time of day, day of week, status of inputs or outputs, status of timers, state of variables, and so forth may be tested, resulting in a true or false answer. When the clause evaluates true, the list of events defined in the "then" clause is executed. If the "if" clause evaluates false, the event list is skipped and the next equation is evaluated. Listing 1 contains a short example of what a control program looks like. Expressions in the "if" clause of the equations may be nested to any number of levels. Any number of events may be defined within the "then" clause.

Figure 5 gives the low-level details about how the event equations are stored in memory. The basic "if" clause is made up of a pair of states with an operator in the middle. Each state may consist of a more complicated expression within parentheses. Multiple operators and states may follow the initial pair. A "stop" character indicates the end of the "if" clause. The byte following the "stop" is used to indicate whether or not the expression has evaluated true on previous passes through the equation table. If the byte is zero, the expression hasn't been true before and if it's true now, the event list may be executed. If the byte is nonzero, then this expression has already been evaluated true and the event list has already been executed. If the expression is true this time, leave the byte nonzero and skip the events. If the expression is false, set the byte to zero and, again, skip the event list.

Following the flag byte is the event list, terminated by a "last" character. Figure 5 also shows the allowable events.

Equation Form

state, op, state, [...], slop, 0
event, event, [..], last

States

Time of Day	80	min,hr,dow	=
dow = 1-7	81		>
dow = 0 for all	a2		<
	a3		≥
	a4		≤
Timer	90	#,tmrlo,tmrhi	=
# =o-15	91		>
tmr = O-65535	92		<
	93		≥
	94		≤
	98	#	on
	99		off
Analog Input	A0	#,setplo,setphi	=
# = o-7	A1		>
setp = O-65535	A2		<
	A3		≥
	A4		≤
Digital Input	BO	#	on
# = O-255	B1		off
Boolean Variable	C0	#	true
# = O-15	C1		false
X-10 Module	DO	house,mod	on
house = "A"-"P"	D1		off
mod = 1-16			

Ops

28	"("	
29	")"	
2A	"*"	and
2B	"+"	or
7E	"~"	not

Special Delimiters

7F	stop
2F	last
00	end of table

Actions

<80		X-10 command
80	#	start timer
81	#	stop timer
90	#	output on
91	#	output off
A0	#	variable true
AI	#	variable false

Figure 5—The Supervisory Controller has a complete *"home control machine language" defined for it. This simple set of states, operators, and actions can be combined to perform just about any task.*

By now, the amount the SC must keep track of should be apparent. It must continually run through the list of event equations, keep track of the time of day, update internal timers, watch inputs, send status information to the host port, watch for user input on the host port, and keep track of network activity, all of which must be done asynchronously and independent

Listing 2-The EVAL routine is he core *of the SC, and is responsible for evaluating tie eventequations.*

```
; Evaluate the event expression pointed to by IX. This and
;  'evarg'  are recursive routines that call each other while
; evaluating an expression.

eval:
        push    bc
        push    de
        call    evarg           ; Evaluate first argument
        ld      b,a             ; Save it
nextop:
        ld      a,(ix)          ; Get next operation
        inc     ix
        cp      stop            ; Done?
        jr      z,evdone
        cp      cparen          ; ")"
        jr      z,evdone
        ld      d,a             ; No, save  op
        call    evarg           ; Evaluate next argument
        ld      c,a
        ld      a,d             ; Perform operation
        cp      andf
        jr      z,andop
        cp      orf
        jr      z,orop
        ld      b,0             ; Error in operator
        jr      evdone
andop:
        ld      a,b
        and     c
        ld      b,a
        jr      nextop
orop
        ld      a,b
        or      c
        ld      b,a
        jr      nextop
evdone
        ld      a,b
        or      a
        pop     de
        pop     bc
        ret
```

Listing *3—EVARG is called by* EVAL *to evaluate individual arguments.* EVARG *also* recursively *calls EVAL when the argument it's passed itself contains an equation.*

```
; Evaluate the argument pointed to by IX   Recursively calls and
; called by 'eval'.

evarg:
        push    bc
        push    de
        push    hl
        ld      b,0             ; NOT off
eva0:
        ld      a,(ix)
        inc     ix
        cp      notf            ; NOT?
        jr      nz,eval
        ld      b,0ffh
        jr      eva0
```

(continued)

Listing 3—*continued*

```
eval:
        cp      oparen          ; "("
        jp      z,lazy

        cp      inpon
        jp      z,inon
        cp      inpoff
        jp      z,inoff

;... other argument types including time of day.
        X-10 module state, etc. are here

        cp      vartru
        jp      z,vron
        cp      varfls
        jp      z,vroff

        xor     a               ; Illegal arg. so eval false
        jp      argdone

lazy:
        call    eval            ; Let EVAL do the work
        jp      argdone

inon:
        ld      hl.inputs       ; Point to input storage
        ld      e,(ix)          ; Offset to input in question
        ld      d,0
        add     hl,de
        ld      a,(hl)          ; Get state
        or      a
        jr      z,falsel        ; Off if zero
        jr      true1
inoff:
        ld      hl,inputs
        ld      e,(ix)
        ld      d,0
        add     hl.de
        ld      a,(hl)
        or      a
        jr      nz.falsel
        jr      true1           : Off if zero

vron:
        ld      hl.vars         : Point to variable storage
        ld      e,(ix)          ; Offset to variable in question
        ld      d,0
        add     hl,de
        ld      a,(hl)          ; Get state
        or      a
        jp      z.falsel        ; False if zero
        jp      true1
vroff:
        ld      hl,vars
        ld      e,(ix)
        ld      d,0
        add     hl,de
        ld      a,(hl)
        or
        jp      nz,falsel
        jp      true1


ialsel:
        ld      a,00h
        jr      f1
truel:
```

[continued]

of anything else on the system. Such a setup screams for assembly language and multitasking, and is just what I used.

Figure 6 contains a somewhat oversimplified block diagram showing how the tasks are broken up. For all intents and purposes, you can view the system as if all the tasks arc running simultaneously. Most communication between the tasks is accomplished through the USC of shared memory. A cardinal rule in any multitasking system is no more than one task is allowed to write to a memory location at any given time. In most cases, you'll see in the block diagram no more than one arrow going into a memory area, though multiple out arrows are kosher. In cases where there is more than one in arrow, the arrows either go to different regions within the area or special safeguards are installed in the tasks to make them cooperate with one another.

For a good introduction to multitasking with the HD64180, I refer you to Jack Ganssle's "Writing a Real-Time Operating System" in issues #7 and #8 (Jan/Feb '89 and Apr/May '89) of *Circuit Cellar INK.*

EVALUATE THE SITUATION

While multitasking and network management are fine, the heart of the system is a little (relatively speaking] piece of code that evaluates the equations. Evaluating an expression that may contain any number of levels of parentheses can be a nightmare if straight-line code is used. Imagine having to keep track of what level you're on in addition to all the intermediate results. The microprocessor has a device built into it that makes such management almost trivial: the stack.

Listing 2 contains EVAL, which evaluates two or more arguments separated by operators. The first argument *is* evaluated and the result is saved. The second argument is evaluated, and the operator is applied to the pair. If more operators and arguments are present, they are evaluated in sequence. The final result is simply returned in the A register.

Listing 3—*continued*

```
        ld      a,0ffh
f1:
        inc     ix              ; Skip over 1-byte argument
        jr      argdone
;
false2:
        ld      a,00h
        jr      f2
true2:
        ld      a,0ffh
f2:
        inc     ix              ; Skip over 2-byte argument
        inc     ix
        jr      argdone
;
false3:
        ld      a,00h
        jr      f3
true3:
        ld      a,0ffh
f3:
        inc     ix              ; Skip over 3-byte argument
        inc     ix
        inc     ix

argdone:
        xor     b               ; Flip result if NOT flag set
        pop     hl
        pop     de
        pop     bc
        ret
```

Most of the work is obviously done in **EVARG**, shown in Listing 3. Upon entry, it checks for a NOT operator and, if present, sets the flag that flips the final state before returning. Next, it checks for an open parenthesis. If it finds one, then it knows the argument contains a subexpression that must be evaluated. **EVAL** is perfectly suited to the task, so is called to evaluate the subexpression. But didn't **EVAL** just call **EVARG?** And won't **EVAL** simply turn around and call **EVARG** again to help evaluate the expression **EVARG** just asked it to evaluate?

The trick is in the stacking. As long as all the registers used by **EVAL** and **EVARG** are pushed on the stack upon entry and restored upon exit, the routines may recursively call each other to get the job done. Intermediate results are automatically stored. Most computer science students see this technique in the ever popular sorting and factorial exercises. I'm happy to say recursion has its uses in everyday situations as well.

Now if both routines simply continue to call each other, how is any

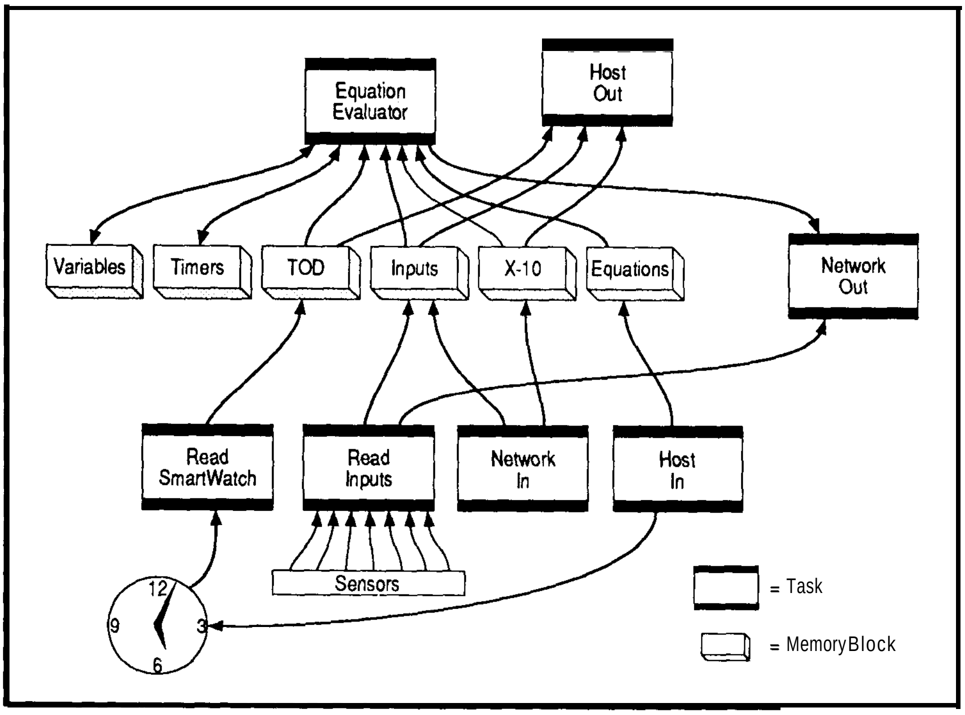


Figure 6 - *A basic multitasking kernel is used to break the operation of the Supervisory Controller into several independent tasks.*

useful work going to get done? Notice I said above that EVA L is called only if EVARG encounters an open parenthesis. If it doesn't encounter one, then it can evaluate the argument for true or false and EVA L will finally have a real answer. The majority of code in EVARG simply checks to see what state is being tested, does the test, and returns the result. I abbreviated the code a bit in Listing 3 because much of it is redundant.

EXPANDING YOUR HORIZONS

So what's next? As I already mentioned, Ed is presenting the software for a series of network modules that fill out the system. Small installations would work quite well with just the SC presented here and the PL-Link presented elsewhere in this issue. Additional modules can be added as the need arises. With the internal SC code broken into logical, independent sections, I can easily add support for new devices by adding a new task to deal with that device. Straight-line code would quickly become spaghetti code if I tried to expand it beyond the initial design specification.

In the next issue of ***Circuit Cellar INK,*** I'll describe a compiler that takes the control program you write with a word processor and converts it to my "HCS machine language" to be sent to the HCS by the host software. ❑

SOURCE

See page *44* for information regarding the availability of HCS II components.

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

Ken Davidson is the managing editor and a member of the Computer Applications Journal's engineering staff. He holds a B.S. in computer engineering ***and an*** M.S. in ***computer*** science from ***Rensselaer*** Polytechnic ***Institute.***

IRS

407 Very Useful
408 Moderately Useful
409 Not Useful

SPECIAL SECTION Embedded Signal Conditioning

Designing with the LM335 Temperature Sensor

FEATURE ARTICLE

Mark E. Nurczyk, P.E.

Something as simple as a temperature interface can present a host of design problems. Find out how Mark handles such issues as system resolution and automated production calibration.

In a past "Firmware Furnace" column, Ed Nisley used an LM335 to measure the temperature of a water pipe (*Circuit Cellar INK* #17, October/November 1990). As part of his article, Ed specified the resolution available from an unamplified LM335 as being 3.6°F using an 8-bit analog-to-digital converter, or 0.8°F 10-bit converter. His column started me thinking, and this article is the result.

The LM335 has an output change of 10 mV for every kelvin step. Zero volts is at 0 K (1 kelvin degree = 1 Centigrade degree, 0°C = 273.15 K). The transfer function of the LM335 is

$$\text{Vout} = (10\,\text{mV} \times \text{temp. in °C}) + 2.73\,15$$

The outputs of the LM335 at the end points of its operational range are 2.3315 V at –40°C and 3.7315 V at 100°C, a span of 1.4 volts. The analog-to-digital converter operates over the input range of zero to five volts. As a result, the LM335's output only uses 28 % (1.4/5) of the A/D converter's input voltage range.

I wondered how much resolution could be made available if a suitable signal conditioning circuit translated the LM335 output voltage to the input range of the A/D converter. If the output of the LM335 can be translated to the 0-5-volt input range of the ADC, the resolution can be increased to 0.98°F for the 8-bit converter, and 0.25°F for the lo-bit.

In this exercise we will be performing a feasibility study. Such a study can be very perplexing; you will not know where you are going until you have completed your journey. I will not begin with a specification, because the circuit cannot be specified until we are finished and know what is possible.

THE BASICS

A signal conditioning circuit that subtracts 2.33 15 volts DC from the LM335's output will translate the –40°C output to 0 volts DC. The output of the signal conditioner will then be 0 to 1.4 volts. If the conditioning circuit also multiplies the LM335's output voltage by 3.57 (5/1.4), then the translated output at 100°C will be 5 volts. An op-amp configured as a differential amplifier is well-suited to the task.

Figure 1 shows the topology of a differential amplifier. Many texts, such as Walter Jung's *IC Op-Amp Cookbook*, derive the following transfer function for the differential amplifier:

$$\text{Vout} = (\text{VB} - \text{VA}) \times (\text{R2} / \text{R1})$$

when $R1 = R3$ and R2 = R4.

Circuit design seems to be so simple at times. Presumably, all that is needed is to connect a voltage source of 2.33 15 volts (the offset voltage) to VA, connect the LM335's output to VB, and pick resistor values for a gain of 3.57. When the LM335's output changes from 2.33 15 to 3.73 15 volts, you would expect the op-amp's output to change from 0 to 5 volts, correct?

Well, yes and no. It depends mostly on the type of op-amp used. For example, an LMC660 is a true CMOS op-amp. If the load has a high impedance, its outputs will saturate to within a few millivolts of its supply voltage. If an LMC660 is used as the amplifier, the output will span 0 to 5 volts to within less than one ADC bit. On the other hand, an LM324 is a bipolar op-amp. The limits of its output are offset from its power supply pins by the junction voltages of bipolar transistors (typically around 0.6 V). If an LM324 is used as the amplifier, its output won't be able reach the desired limits of 0 and 5 volts (assuming a single-ended 5-V power supply). Also, for proper linear operation, biasing of the input differential amplifier limits

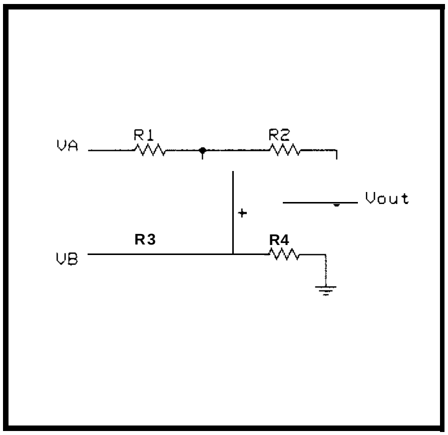


Figure 1—*A differential amplifier is well suited to conditioning the LM335's output.*

the input voltages of both op-amp types.

The input amplifier section of an LM324 is shown in Figure 2a. The current source has a 0.3-volt drop across it, while each transistor has a 0.6-volt drop from its base to its emitter. Therefore, the amplifier will saturate whenever either input is greater than 1.5 volts below V+. The LMC660 has similar input structures in CMOS. If the signal voltage needed exceeds (V+)-1.5, there is no choice but to provide a greater supply voltage to power the op-amps.

The LM324's output structure is shown in Figures 2b and 2c. When sourcing current, the structure of Figure 2b is used. As above, the highest voltage that can be sourced is 1.5 volts below V+. Current sinking is determined by Figure 2c. The output voltage is the sum of the NPN saturation voltage and the base-emitter drop of the PNP, or about 0.8 volts. The various manufactures of the LM324 have souped up the PNP so the output voltage is less than 20 millivolts if the load is 10k or greater at 5 volts.

I'll use the LM324 for this feasibility study. It is one of the most popular bipolar op-amps currently in production. Using a 5-volt power supply, the voltage presented to the A/D converter by an LM324 will vary from 0.02 volts to 3.5 volts. This voltage span will utilize 70% of the A/D converter's resolution, a little over double what we started with, and will provide us with a temperature resolution of 1.4°F for the 8-bit converter or 0.35°F for the 10-bit converter.

OFFSET AND GAIN

The first step is to determine the amplifier's offset and gain voltages. MiniTK, an equation solving program from Universal Technical Systems Inc., is useful for solving the equations involved. The basic equations to solve for offset and gain are

Voutlo = (Vinlo – Voff) x Gain
Vouthi = (Vinhi – Voff) x Gain

To fit our system the following constants are used: *Voutlo* = 0.02, *Vinlo = 2.3315, Vouthi = 3.5,* and *Vinhi = 3.7315.* To give MiniTK a starting point, you have to enter estimated values for *Voff* and *Gain.* Solving the equations results in an offset voltage of 2.323 volts and a gain of 2.486.

The circuit we will use is shown in Figure 3. The offset voltage is set by the voltage divider made up of resistors R1 and R2. The gain is set by selecting the proper values for R3, R4, R5, and R6. Again using the computer as a design tool, I used a pair of programs I wrote called VOLTDEV.BAS and GAINRAT.BAS to determine the proper resistor values, which turned

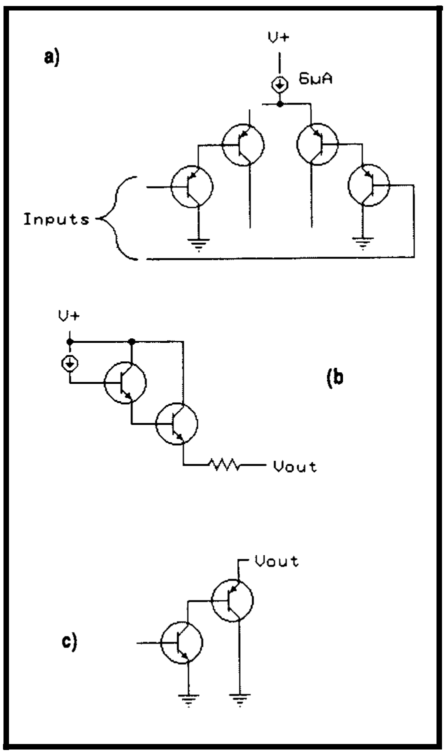


Figure 2—*(a) The input section of the ever-popular* **LA4324** *saturates when either input is greater than 1.5 volts below V+. The output sections are modeled as in (b) when he* **LM324** *is sourcing current; the structure in (c) is used when sinking current*

out to be R1 = 1.82k, R2 = 1.58k, R3 = 105k, R4 = 261k, R5 = 105k, and R6 = 261 k, which are actually scaled from what the programs predicted. You don't want the gain network of the op-amp (R3 and R4) to load the voltage divider made up of R1 and R2. If R3 and R4 significantly load R1 and R2, the gain ratio will change as the output voltage changes. I picked the op-amp feedback network to be two orders of magnitude greater in impedance than the voltage divider network. Usually one order of magnitude will do.

MODELING THE SYSTEM

The student version of PSpice has had some major functions added since its last release. Some of the new functions are shown in Listing 1, a PSpice input file based on Figure 3.

E335, a voltage-controlled voltage source, models the LM335 temperature sensor. The line that defines E335 contains a transfer function. VTEMP is the controlling voltage source, its output, in volts DC, corresponds to temperature in degrees Fahrenheit. E335's transfer function converts the temperature from Fahrenheit to Centigrade and multiplies by the voltage change per degree. Finally it adds an offset to bring the output voltage to the correct value.

The .WCASE line performs a worst-case analysis, which determines circuit parameter values which will cause the poorest circuit performance based on defined component tolerances. First, the worst-case analysis determines the sensitivity of the defined circuit node waveform to each device that has a specified tolerance. At the end of the analysis, each device is set either high or low to give the worst-case waveform. The PSpice output file will contain the nominal as well as the worst-case value.

EOUT, another voltage-controlled voltage source, is defined as a table function. When V(4,0) is zero, the output is 0.02 volts. When V(4,0) is 3.5, the output is 3.5 volts. The output is the linear interpolation of 0.02 to 3.5 when the input is between 0 and 3.5. Also, the output is limited and will never go above 3.5 or below 0.02.

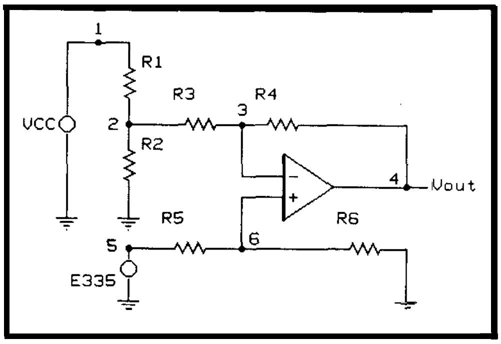


Figure 3—*The complete LM335 interlace is modeled in PSpice to observe its characteristics.*

Listing 1 shows the worst-case section commented out. The circuit is run to check the input and output waveforms, with Figure 4 showing the results. Node voltage *V(5)* shows the LM335 output voltage. I used the PSpice cursor to show the value of *V(5)* at 40°F and 212°F. The modeling of the LM335 produces the proper voltages at each temperature. V(4) shows the amplifier is working as desired, with an output of 0.02 V at –40°F and 3.5 V at 212°F.

Now run the worst-case analysis by commenting out the **. DC** line in the temperature sweep section and removing the leading asterisks from the worst-case section. The device values used will be determined from the tolerance specified in the **. MODE L** statement. The Gaussian distribution (a bell-shaped curve) most closely approximates the tolerance distribution of electronic parts. The tolerance value you use should be one fourth of the part tolerance for Gaussian distribution. The end points of a Gaussian distribution in PSpice are at $\pm 4\sigma$ ($\pm 3\sigma$ contains 99.73% of the population).

Run the worst-case analysis twice, first with the **MAX** keyword in the **. WCASE** statement and second with the **MI N** keyword. **MAX** finds the maximum value of the specified node voltage or component current; **MI N** finds the corresponding minimum value. By including two complete circuit descriptions in one PSpice input file, both sets of results can be sent to a single output file from a single PSpice run. The worst-case results for our circuit are shown in Table 1.

VTEMP was set to the three temperatures shown at the top of each column. The temperatures in the last four rows are derived using the following procedure: The voltage values in the nominal voltage row are turned into bit values with fractional parts by dividing them by 0.0195 (for an 8-bit converter). A linear regression is next performed on the nominal bit values and temperatures to find the best-fitting straight-line equation of the form y=mx+b. The temperature equation turns out to be

$$°F = (1.4288 \times \text{bits}) - 42.6297$$

Next, the bit values at the maximum and minimum limits are determined and truncated to whole numbers. An A/D converter knows nothing about fractional bits-it doesn't round, but truncates. The truncated bit values are changed into temperatures using the temperature equation derived above. This equation can be used by a microcomputer to transform the raw A/D bit values into scaled engineering units.

During a large production run, using 1% tolerance resistors, the temperature error due to the amplifier circuit can approach almost 13°F. Luckily, 99.73% of the amplifiers (the three σ points) will be within 9.75°F (0.75 x 13). Using 0.1% resistors-the

Listing 1—*The LM335 interface* **shown in figure 3 is** *translated to something the computer can understand.*

```
R1   1  2  R1%  1.82K
R2   2  0  R1%  1.58K
R3   2  3  R1%  105K
R4   3  4  R1%  261K
R5   5  6  R1%  105K
R6   6  0  R1%  261K
Xl   6  3  4  LM324
VCC 1  0  DC  5
***********************************
***  Temperature  Sweep  Section  ***
***********************************
E335   5  0   VALUE = {((V(8)-32)*(5/9)*0.01)+2.7315}
VTEMP  8  0   DC  -35
RTEMP  8  0   10K
.DC    VTEMP   -45  220  1
***************************
***  Worst  Case  Section  ***
***************************
*RINC   100   0  1
*VINC   100   0  DC  1
*.DC   VINC 0  1 1
*.WCASE  DC  V(4)  MAX
***************************
*.PROBE
.MODEL R1%   RES   (R=1 DEV/GAUSS .25%)
*  OPAMP  MACROMODEL  SUBCIRCUIT
.SUBCKT LM324     1  2  3
*                 |  |  |
*                 |  |  +-OUTPUT
*                 |  +----INVERTING  INPUT
*                 +-------NON  INVERTING  INPUT
RIN   1   2   10MEG;      INPUT       IMPEDANCE
*  GAIN  AND  PHASE  CONTROL
GM    0   4     12     1
RG        4   0  100000
CP        4   0  1.59P
* OUTPUT  SECTION
EOUT   5   0   TABLE   {V(4,0)} (0,.020 3.5,3.5)
ROUT   5   3   25
.ENDS
.END
```

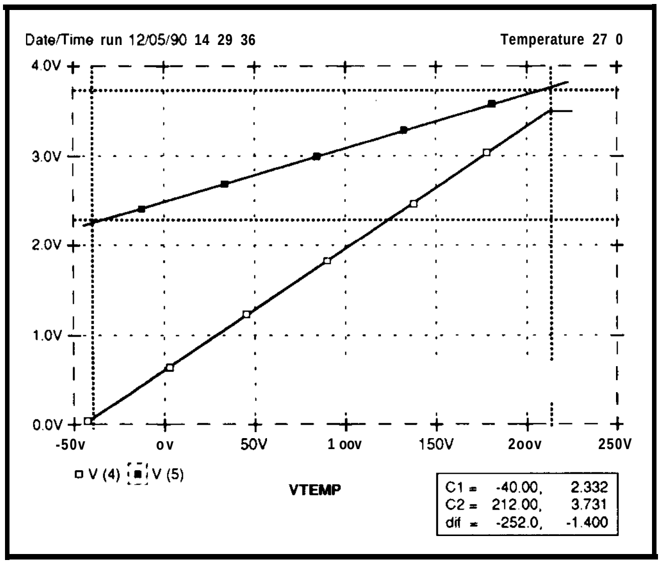


Figure *4-The results of the PSpice simulation defined in Listing 1 show the circuit's characteristics.*

most commonly available commercial parts with the tightest tolerance-will reduce the error by an order of magnitude to 1.3°F. Close matching of the resistor values over the temperature range is essential for proper circuit operation.

The amplifier error must be added to the error of the LM335. The LM335 has a calibration error of 10.8°F (6°C) at 77°F (25°C). The variation in span—the algebraic difference between maximum and minimum temperatures-can be as much as 2.5°C over the operating range of –40°C to 100°C. Consequently, the slope of the temperature equation of the LM335 may vary from 9.9 mV per degree Centigrade to 10.1 mV per degree Centigrade.

The calibration error of the LM335 is an offset error. If the calibration error of the individual LM335 is known, the offset, when appropriately scaled, can be added to the bit value in the temperature equation. The amplifier error may be a combination of offset error and slope error. Applying two precise ambient temperatures to the LM335 allows the microcomputer to determine a slope and offset correction for the amplifier and LM335 sensor combination. Of course, some sort of nonvolatile memory is required to store the calibration data.

PUSHING THE DESIGN

Now that we know how to design a differential amplifier with arbitrary offset and gain, let's "push it" a little. The 8-bit A/D converter that Ed used in his article has eight inputs. If the application permits, all eight inputs can be used for one measurement. Eight amplifiers, using LM324 op-amps, are theoretically equivalent to

	-35°F	69.8°F	206°F
Maximum voltage	0.195V	1.625V	3.495V
Nominal voltage	0.102V	1.541V	3.397V
Minimum voltage	0.020V	1.454V	3.301V
Maximum temperature	-29.8°F	76.0°F	211.7°F
Minimum temperature	-41.2°F	63.1°F	198.8°F
Tolerance error at Max	5.2°F	6.2°F	5.7°F
Tolerance error at Min	-6.2°F	-6.7°F	-7.2°F

Table 1—*The circuit's worst-case analysis results are shown in tabular form*

an A/D converter that spans 1,427 bits—[(3.5 - 0.20) / 0.01951 x 8-giving a resolution of 0.18°F over the -40°F to 212°F operating range of the LM335.

I'll detail the design of one amplifier in about the middle of the temperature range. Each amplifier wi!l span a range of 252/8 or 31.5°F. We will concentrate on the range of 54.5°F to 86°F. For this range, the LM335's end points are 2.8565 volts and 3.0315 volts, respectively.

Using the MiniTK equations described above, we find the gain has to be 19.89 and the offset 2.855 volts. My two custom programs indicate the resistors in Figure 3 should be *R1* = 187 Ω, R2 = 249 Ω, R3 = 23.2k, R4 = 464k, R5 = 23.2k, and *R6* = 464k. I modified Listing 1 to reflect the above values and obtained the results shown in Figure 5. The results are pretty close; minimum output at 54.3°F and maximum output at 85.9°F. Now all we have to do is repeat this design exercise seven more times and we are done. Perhaps, but first check the worst case.

The worst-case results for our enhanced circuit are shown in Table 2. They show that the end points of adjacent temperature ranges don't necessarily meet. To get a continuous function, the end points of each range have to overlap by almost 8°F. The range we looked at above should really be 50.5°F to 90°F, with the consequent gain of 15.86 and offset of 2.833. The preceding range is 15°F to 58.5°F, while the next range is 82°F to 121.5°F. Taking the overlap into account, we can expect a resolution of 0.22°F with an 8-bit A/D converter.

THE CALIBRATION PROBLEM

Now there is a large problem. Each of the eight ranges has to be separately calibrated, at two points each, for gain and offset errors. To calibrate the amplifiers, precision fixed resistors can be used to simulate the LM335. Complicated software has to be written which knows when the calibration is being performed and what range is being calibrated. Perhaps the best way to proceed is to have the embedded controller to which the signal conditioning is attached communicate with a PC during calibration. The PC can worry about applying the proper resistors at the correct time, performing all the mathematical gymnastics on the conversion results, and determining the calibrated slopes, offsets and endpoints for each range. A single- or dual-point calibration of the LM335 can be performed concurrently. The PC can calibrate many, many assemblies quickly.

Calibrating many amplifiers is costly and time consuming. Producing large numbers of assemblies may not be cost effective. Most of the error is due to mismatches between *R1* and R3, and R4 and *R6*.

The first circuit most engineers would pick for this job is a differential amplifier, which is why I have shown circuits based on such a configuration. However, simpler inverting or noninverting amplifiers can perform the same function.

A noninverting amplifier, shown in Figure 6a, produces an output voltage slope with the same sign as that of the input voltage slope. The input voltage is connected to the noninverting input of the op-amp while the offset voltage is connected to the resistor feedback network. To determine the amplifier's output voltage, look at the contribution of each voltage source on the output and add them together.

From the viewpoint of the offset voltage source, the noninverting amplifier actually looks like an inverting amplifier. The output voltage due to the offset voltage contribution is

Voffset x (-Rf / Rin)

To the input voltage source, the noninverting amplifier truly looks like a noninverting amplifier. The output voltage due to the input voltage

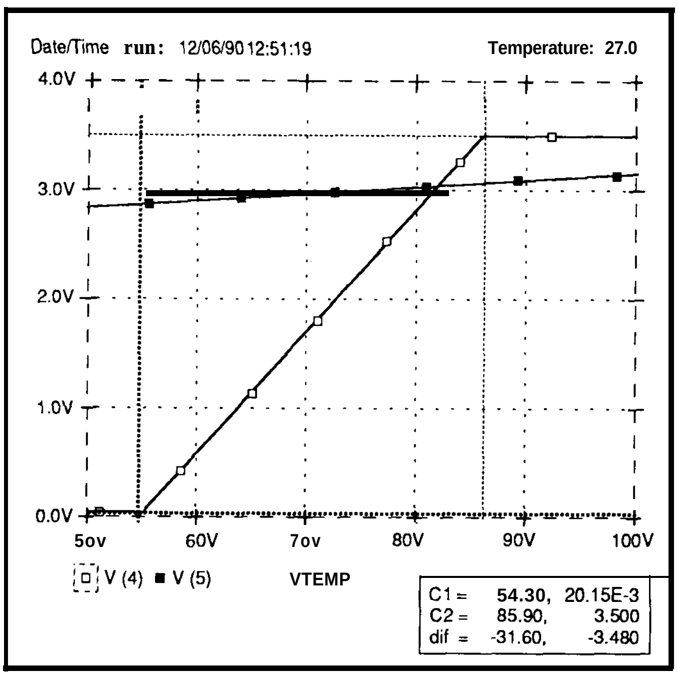


Figure 5—*After enhancing he circuit, PSpice is run again with flew results.*

	59° F	70° F	82° F
Maximum voltage	0.964V	2.164V	3.492V
Nominal voltage	0.530V	1.747V	3.074V
Minimum voltage	0.084V	1.317V	2.669V
Maximum temperature	62.9°F	73.6°F	85.8°F
Minimum temperature	54.9°F	66.0°F	78.2°F
Tolerance error at Max	3.9°F	3.6°F	3.8°F
Tolerance error at Min	-4.1°F	-4.0°F	-3.8°F

Note: °F = (0.177 x bits) + 54.208

Table 2—*The worst-case results for the enhanced circuit show that the end points of adjacent temperature ranges don't necessarily meet.*

contribution is

$$V_{in} \times [1 + (R_f / R_{in})]$$

When added together, the output voltage is

$$V_{out} = \{V_{in} \times [1 + (R_f / R_{in})]\} - [V_{offset} \times (R_f / R_{in})]$$

An inverting amplifier, shown in Figure 6b, generates an output voltage slope with a polarity opposite that of the input voltage's. The offset voltage is connected to the noninverting input of the op-amp while the input voltage is connected to the resistor feedback network. The contributions of each voltage source at the output are added together to come up with the amplifier output voltage.

The amplifier looks like an inverting amplifier to the input voltage source. The output voltage due to the input voltage contribution is

$$V_{in} \times (-R_f / R_{in})$$

The amplifier looks like a noninverting amplifier to the offset voltage source. The output voltage due to the offset voltage contribution is

$$V_{offset} \times [1 + (R_f / R_{in})]$$

When added together the output voltage is

$$V_{out} = \{V_{offset} \times [1 + (R_f / R_{in})]\} - [V_{in} \times (R_f / R_{in})]$$

Gain and offset voltages for both inverting and noninverting amplifiers can be found using the same technique described above.

The gain in these two amplifiers is only dependent on the ratio of two resistors and not their absolute values, so is an ideal application of a potentiometer. Substituting a potentiometer for the network made up of R_f and R_{in} allows a smooth progressive change in amplifier gain. One end of the pot is connected to the voltage source and the other end pot is connected to the amplifier output. The pot wiper is tied

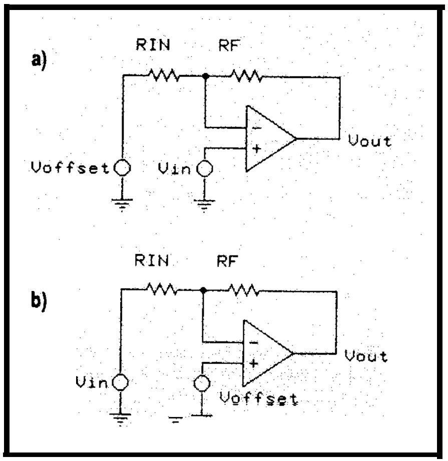


Figure 6—*Standard noninverting* (a) and *inverting* (b) *amplifiers* may be *used in place of he* differential amplifier.

to the point where the Rin/Rf junction is normally connected. Taking it one step further, controlling the potentiometer by the microcomputer would allow virtually any resolution of the input voltage to be synthesized.

Dallas Semiconductor comes to the rescue with their DS1267, a dual digitally controlled potentiometer chip. Each potentiometer is composed of 256 resistive sections. The pot wiper position is controlled by an 8-bit number sent over a d-wire serial port. Any value between 0 and 256 can be sent independently to each potentiometer. The tap point of the wiper on the resistor array is controlled by the value sent to the serial port. Each end of the resistor array and the wiper of both pots are connected to pins on the chip, and both potentiometers have the same resistance value. Values of 10k, 50k, and 100k are available.

A noninverting amplifier circuit using the digital pots is shown in Figure 7. One pot, $R1$, is used to generate an offset voltage. The voltage developed by the potentiometer is calculated using

$$Vcc \times (X0 / 256)$$

$X0$ is the digital value used to set the wiper position. $U1$ is a unity gain buffer amplifier used to present a high impedance to $R1$ and a low impedance to R2. $U1$ keeps the gain ratio from changing as the output voltage changes.

The other pot, R2, is used to set the amplifier gain. The digital value used to set the position of $R2$'s wiper, Xl, corresponds to Rin in Figure 6a, while Rf corresponds to $256-X1$. The ratio Rf/Rin becomes

$$(256 - Xl) / Xl$$

or

$$(256 / Xl) - 1$$

The following system of equations describes the operation of the circuit shown in Figure 7:

Gain	= (256 /Xl) – 1
Voffset	= Vcc x (X0 / 256)
Vin	= {[(5/9) x ("F- 32)] + 273.153 x 0.01
Vout	= [Vin x (1 + Gain)] – (Voffset x Gain)
A/D_bits	= Vout / (Vcc / 256)

This system of equations can be solved to produce a single equation:

°F = [0.703 13 x A/D_bits x (Vcc / 256) x Xl] + (0.70313 x vcc xX0) – (0.002747 x Vcc x X0 x Xl) – 459.67

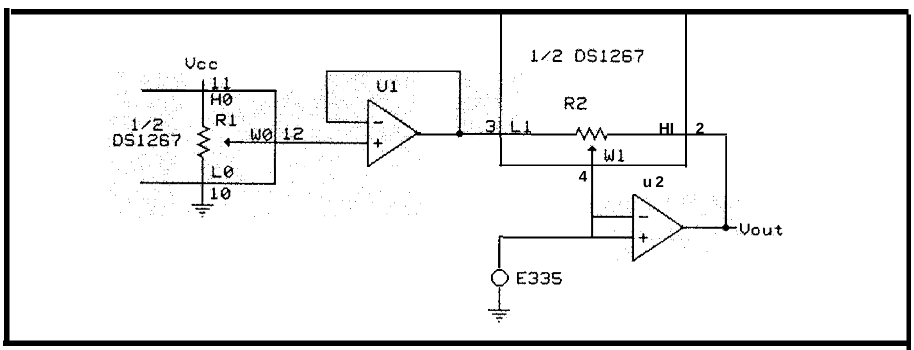


Figure 7- *The DS1267 digitally controlled potentiometer* allows the computer to *adjust* the offset *and gain settings* on *the fly*.

This final equation can be used by a microcomputer to transform the raw A/D bit values into scaled engineering units.

Solving the equation at different gains and temperatures shows the resolution can be as great as 73 bits per degree Fahrenheit and as low as 3.4°F per bit. Be careful at higher gains. Small amounts of noise present in any electronic system may cause wild fluctuations of the amplifier's output. Convolution techniques described in Jack Ganssle's article "Signal Smoothing-Taking the Rough Edges off of Real-World Data" ***(Circuit Cellar INK*** #10, August/September 1989) may be required to produce a stable temperature measurement.

Potentiometers typically have end-to-end tolerances of 20%, as does the DS1267. Because it is the ratio of the resistances on each side of the wiper to one another that determines offset and gain, the end-to-end tolerance is immaterial. The gain and offset errors arc only determined by the potentiometer's linearity. The DS1267 has a specified linearity of half a least-significant bit, or 0.2%.

When applying the circuit of Figure 7, do not start the conversion at high gains where it is easy to saturate the amplifier. Vary the gain and offset slowly; creep up on the values.

SUMMING IT UP

We have looked at three different methods of interfacing an LM335 to an analog-to-digital converter. Which is the best method? All of them! You get to pick the best one for your application.

The digitally controlled potentiometer amplifier is the most versatile. Gain and offset can be changed at any time to optimize performance. You pay for versatility, though. LM324s cost about 30¢ and the DS1267 costs $2.50, both in OEM quantities.

The differential amplifier with fixed resistors is fairly versatile, too. It can be set to any offset and gain desired, but cannot be changed once built. I did not show them in a fixed gain and offset configuration, but inverting and noninverting amplifiers can be used for this function as well.

The cost is very low-about a penny each for the resistors in addition to the op-amp.

Ed's circuit is certainly the least expensive: $0.

Now that you have a better idea of what is possible, it is time to develop the functional requirements for your own temperature input circuit. The cost and performance tradeoffs required by your application define which signal conditioning circuit to use. Be sure to make your functional requirements complete. For example, there is the obvious requirement of reading temperature to some arbitrary resolution. Also, don't forget resistance to destructive transients or misapplication. The list goes on.. ❑

SOURCES

PSpice

MicroSim Corp.
20 Fairbanks
Irvine, CA 92718
(800) 245-3022
(714) 770-3022

MiniTK

Universal Technical Systems, Inc.
1220 Rock St.
Rockford, IL 61101

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

Mark Nurczyk is a Registered Professional Engineer with nineteen years experience in analog and digital design. He works for a large OEM designing microcomputer- and analog-based machine controls..

IRS

410 Very Useful
411 Moderately Useful
412 Not Useful

Isolation Amplifier Design Using the IL300 Linear Optocoupler

SPECIAL SECTION

Bob Krause

> Traditionally, optocouplers are only used with digital signals. A new chip from Siemens makes it possible to use optocoupler isolation in analog circuits as well.

galvanic isolation is desirable and often essential in many measurement systems. Applications such as industrial sensors, medical transducers, and switch mode power supplies require isolation and insulation between the mains primary and secondary. Operator safety and signal quality are also ensured with isolated interconnections. These isolated interconnections commonly use isolation amplifiers.

The IL300 linear optocoupler is a new circuit element from Siemens that enables designers to easily develop high-quality isolation amplifiers. The IL300's 130-dB common mode rejection (CMR), ±50-ppm/°C stability, and ±0.01% linearity provide a quality link from the sensor to the controller input. Its 7500-V withstand test voltage (WTV) insulation, DC response, and high CMR are features which assure patient safety and accuracy for the transducer signals. The linear coupler's 200-kHz bandwidth and gain stability make it an excellent candidate for subscriber and data phone interfaces. With switch mode power supplies currently approaching l-MHz switching frequencies, they often need output monitoring feedback networks with wide bandwidth and flat phase response. The IL300 satisfies these needs with simple support circuits.

CHARACTERISTICS OF THE IL300

The IL300 consists of a high-efficiency AlGaAs IR LED emitter optically coupled to two independent PIN photodiodes. These three semiconductors are assembled in an 8-pin DIP plastic package using a high-voltage double-molded insulation process, and satisfies UL and VDE 0884, 0805/0806 safety requirements.

Figure 1 shows the package footprint and the IL300's electrical schematic. The servo photodiode (3,4) provides a feedback signal which is used to control the LED's (1,2) forward current. The servo photodiode provides a photocurrent, I,,, that is directly proportional to its incident flux. Operating the LED in an optical servo loop linearizes the LED's output flux and eliminates the LED's time and temperature drift. The galvanic isolation between the input and the output is provided by a second PIN photodiode (5,6) located on the output side of the coupler. The output photodiode current, I_{P2}, accurately tracks the photocurrent generated by the servo photodiode.

ISOLATION AMPLIFIER DESIGN TECHNIQUES

The IL300's photodiodes can be operated as photovoltaic or photoconductive current sources. The photovoltaic (zero biased) mode offers the best linearity, lowest noise, and best drift performance. Isolation amplifiers using this circuit configuration are suitable for 12-bit A/D applications. Photoconductive (reverse biased] photodiode operation provides the largest coupled frequency bandwidth. The photoconductive configuration has linearity and drift characteristics comparable to an 8- or 9-bit A/D converter.

Either of the above photodiode modes can be used to create isolation

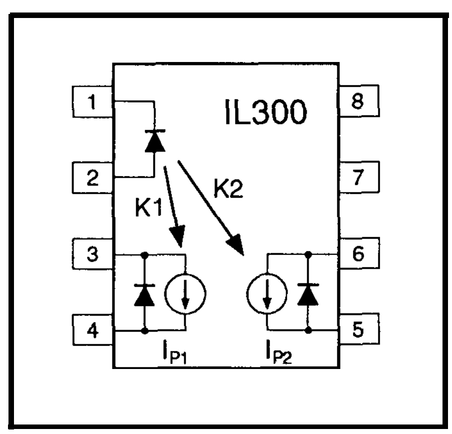


Figure l-The IL300 uses a single LED and a pair of detectors.

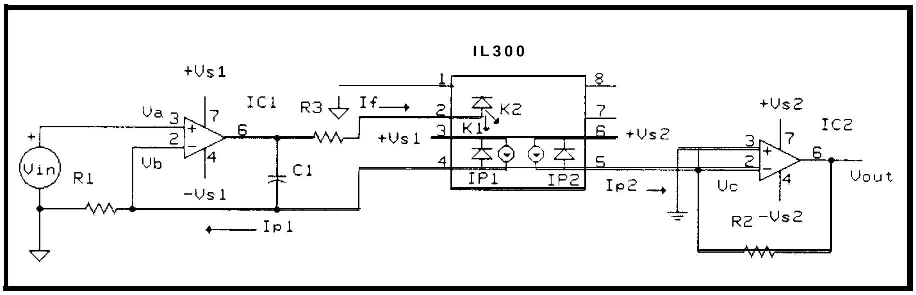


Figure 2—*Positive unipolar inverting photoconductive isolation* amplifier.

amplifiers that will couple either unipolar or bipolar input signals. The optimum circuit topology is determined by the input voltage range, transfer gain, bandwidth.

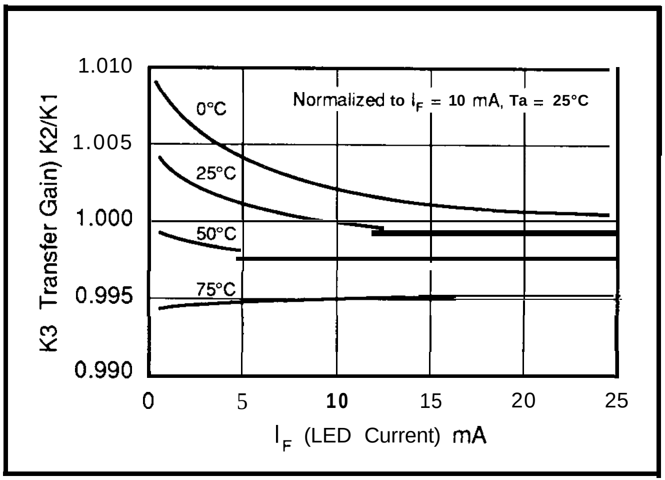


Figure 3—*Normalized servo transfer gain.*

PHOTOCONDUCTIVE ISOLATION AMPLIFIER

The following analysis shows the equations used to design a photoconductive inverting isolation amplifier that will respond to a positive unipolar signal. This amplifier topology is best suited for communicating input signals that have a positive offset above virtual ground. The typical operation of an IL300 isolation amp is more easily understood by reviewing the circuit shown in Figure 2.

The input circuit consists of an operational amplifier, ICI, a feedback resistor, *R1,* and the servo photodiode section of the IL300. The servo photodiode is a current source operating in the photoconductive, or reverse biased, mode. The initial conditions are $V_a = V_b = 0$ V. When a positive voltage is applied to the noninverting input (V_a) of the op-amp, the output of the op-amp will swing toward the positive V_{s1} rail, and forward bias the LED. As the LED current, I_F, starts to flow, an optical flux will be generated which will irradiate the servo photodiode causing it to generate a photocurrent, I_{P1}. This photocurrent will flow through *R1* and develop a positive voltage at the inverting (V_b) input of the op-amp. The amplifier output will start to swing toward the negative supply rail, $-V_{s1}$. When the magnitude of V_b is equal to that of V_a, the LED drive current will cease to increase. This condition servos the circuit into a stable closed-loop condition.

When V_{IN} is modulated, V_b will track V_{IN}. In order for this to happen, the photocurrent through *R1* must also track the change in V_a. Recall that the photocurrent, I_{P1}, results from the change in LED current times the servo gain, *K1*. The following equations can be written to describe this activity:

$$V_a = V_b = V_{IN} = 0 \qquad (1)$$

$$I_{P1} = I_F \times K1 \qquad (2)$$

$$V_b = I_{P1} \times R1 \qquad (3)$$

Combining equations (1),(2), and (3), the relationship of LED drive to input voltage is presented.

$$V_a = I_{P1} \times R1 \qquad (4)$$

$$V_{IN} = I_F \times K1 \times R1 \qquad (5)$$

$$I_F = V_{IN} / (K1 \times R1) \qquad (6)$$

Equation 6 shows that the LED current is related to the input voltage V_{IN}. A changing V_a causes a modulation in the LED flux which will change to a level that generates the necessary servo photocurrent to stabilize the optical feedback loop. The LED flux will be a linear representation of the input voltage, V_a. The servo photodiode's linearity controls the linearity of the isolation amplifier.

The next step in the analysis is to evaluate the output section of the amplifier. This circuit consists of a transresistance amplifier, *IC2,* which is connected to the output PIN photodiode. The photocurrent, I_{P2}, is derived from the same LED that irradiates the servo photodetector. The output, V_{OUT}, is proportional to the output photocurrent, I_{P2}, times the transresistance, R2.

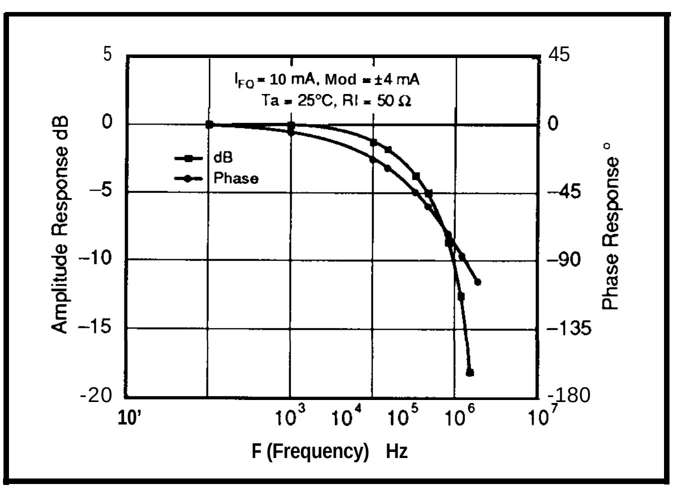


Figure 4—*Frequency and phase response of the photoconductive isolation amplifier.*

$$V_{OUT} = -I_{P2} \times R2 \qquad (7)$$

$$I_{P2} = K2 \times I_F \qquad (8)$$

Combining (7) and (8) and solving for I_F is shown in (9).

$$I_F = \frac{-V_{OUT}}{K2 \times R2} \qquad (9)$$

The input-output gain of the isolation amplifier is determined by combining equations (6) and (9).

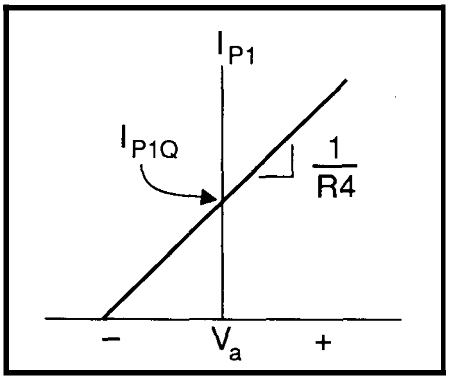


Figure 5— **Transfer** *characteristic* **of a** *prebiased bipolar amplifier.*

This analysis shows that isolation amplifiers using the IL300 are not plagued with the gain and temperature drift problems associated with the standard phototransistor optocoupler. The preceding analysis showed how the servo operation of the IL300 eliminates the influence of LED characteristics on the isolation amplifier transfer gain.

The photoconductive type of amplifier design permits bandwidths which approach the IL300's characteristic frequency response in Figure 4.

the reference voltage (V_{REF1}) across *R4* (2k). V_{REF1} is derived from the voltage divider (R9, *R10*) and the LM3 13 reference diode (*ZD1*) as given in equation (12). The offset control (*R10*) is used to set the quiescent photocurrent, I_{PQ}, to approximately 50 µA. A buffer transistor (*Q1*) is included at *IC1*'s output to allow it to easily supply the typical LED current of 7 mA.

$$I_{PQ} = \frac{R4}{V_{REF1}} \qquad (13)$$

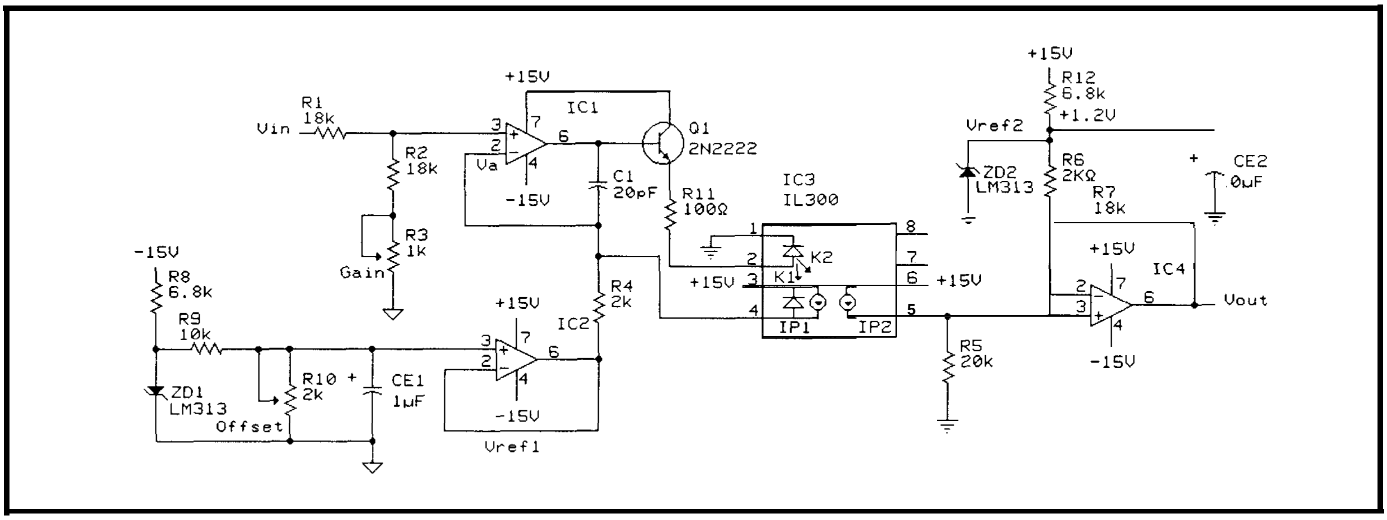


Figure 6—*High stability bipolar photoconductive isolation amplifier.*

$$\frac{V_{IN}}{K1 \times R1} = \frac{-V_{OUT}}{K2 \times R2} \qquad (10)$$

The solution for the input-output gain is given in equation (11).

$$\frac{V_{OUT}}{V_{IN}} = \frac{-K2 \times R2}{K1 \times R1} \qquad (11)$$

Note that the LED current, I_F, is factored out of equation (11). This is possible because the servo and output photodiode currents are generated by the same LED source. This equation can be simplified further by replacing the *K2/K1* ratio with IL300's transfer gain, K3.

$$\frac{V_{OUT}}{V_{IN}} = -K3 \times \frac{R2}{R1} \qquad (12$$

Figure 3 shows the consistency of the normalized K3 as a function of LED current and ambient temperature. The transfer gain drift as a function of temperature is typically *SO ppm%/°C over a 0-75°C range.

BIPOLAR PHOTOCONDUCTIVE ISOLATION AMPLIFIER

Telephone and audio applications require the isolation amplifier to respond to bipolar signals. A prebiased or offset isolation amplifier permits the servo amplifier to respond to bipolar signals. The prebias forces the LED to produce a current in the servo photodiode with a zero voltage input. This quiescent photocurrent, I_{PQ}, is then amplitude modulated by the bipolar input signal. The relationship between the servo photocurrent and the input voltage is shown in Figure 5.

The isolation amplifier, shown in Figure 6, consists of a noninverting servo input amplifier and a noninverting output voltage amplifier. A prebias, I,,, is introduced at the inverting input of *IC1* by

The input voltage range is set by the voltage divider (*R1,* R2, R3) so *IC1*'s input voltage (V_a) does not exceed V_{REF1} (-0.1 V). These resistors provide a 10: 1 divider permitting a ±1-V input range.

The output amplifier, *IC2,* is an offset noninverting voltage amplifier.

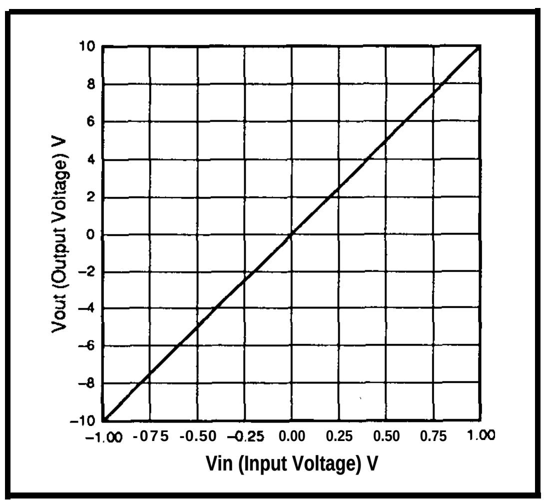


Figure **7**—***Photoconductive*** *amplifier transfer gain.*

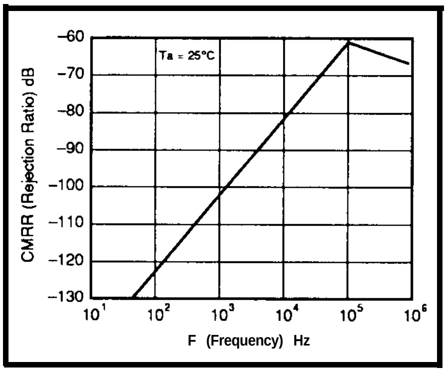


Figure **8**—*Typical* common mode *rejection for the IL300.*

The output offset, V_{REF2}, is provided to compensate for offset introduced in the servo amplifier. The output photocunent to voltage conversion is provided by *R5.* The amplifier gain of 10 is set by the *R6/R7* network. The voltage gain of this isolation amplifier is given in equation (14).

$$\frac{V_{OUT}}{V_{IN}} = \frac{K3 \times R5 \times (R2 + R3) \times (R6 + R7)}{R4 \times R6 \times (R1 + R2 + R3)} \qquad (14)$$

The offset (V_{REF1}) is adjusted to compensate for the output offset (V_{REF2}). The precise offset voltage is predicted by equation (15).

$$V_{REF1} = \frac{V_{REF2} \times R4 \times R7}{K3 \times R5 \times (R6 + R7)} \qquad (15)$$

The output stage offset, V_{REF2}, is supplied by the LM3 13 1.2-V reference. With a unity K3 the V_{REF1} is adjusted to be -0.108 V.

This isolation amplifier has a transfer gain of 10, given K3 = 1 .O and the resistor values selected. The input/output voltage characteristics of the amplifier are shown in Figure 7.

NOISE REJECTION

One of the principal reasons to use an isolation amplifier is to reject electrical noise. The circuits presented thus far are of a single-ended design. The common mode rejection, CMR, of these circuits is set by the CMR of the coupler and the bandwidth of the output amplifier. The typical common mode rejection for the IL300 is shown in Figure 8.

PHOTOVOLTAIC ISOLATION AMPLIFIER

The schematic of a typical positive unipolar photovoltaic isolation amplifier is shown in Figure 9.

The input stage consists of a servo amplifier, ICI, which controls the LED drive current. The servo photodiode is operated with zero voltage bias which is accomplished by connecting the photodiodes anode and cathode directly to *IC1's* inverting and noninverting inputs. The servo photocurrent is linearly proportional to the input voltage, $I_{P1} = V_{IN} / R1$. The servo photocurrent resulting from the LED emission keeps the voltage at the inverting input of *IC1* equal to zero. The output photocurrent, I_{P2}, results from the incident flux supplied by the LED. The output voltage is proportional to the output photocurrent I_{P2} and is equal to the product of the output photocurrent times the output amplifier's transresistance, R2. The composite amplifier transfer gain (V_{OUT}/V_{IN}) is the ratio of two products; The first is the output transfer gain, K2 x R2, and the second is the servo transfer gain, *K1 x R1.* The amplifier gain is the first divided by the second. The result is shown in equation (16).

$$\frac{V_{OUT}}{V_{IN}} = \frac{K2 \times R2}{K1 \times R1} \qquad (16)$$

Equation 17 shows that the composite amplifier transfer gain is

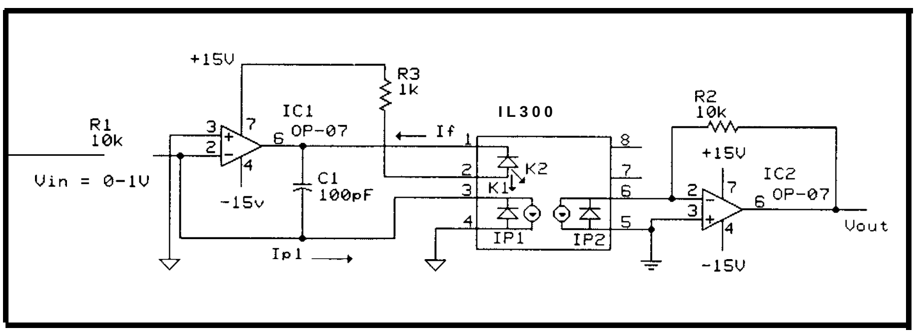


Figure **9**—*Positive unipolar photovoltaic isolation amplifier.*

independent of the LED forward current. The $K2/K1$ ratio reduces to IL300 transfer gain, K3, which is included in equation 14. This shows the composite amplifier gain is equal to the product of the IL300 gain, K3, times the ratio of the output to input resistors.

$$\frac{V_{OUT}}{V_{IN}} = \frac{K3 \times R2}{R1} \qquad (17)$$

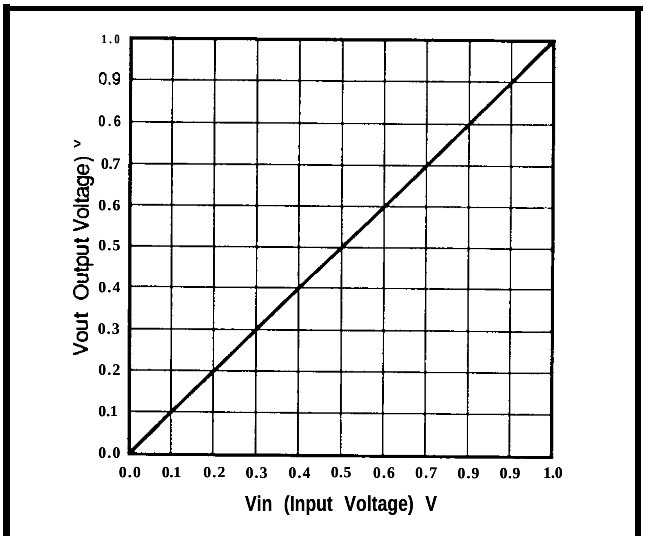


Figure 10—*Photovoltaic amplifer transfer gain.*

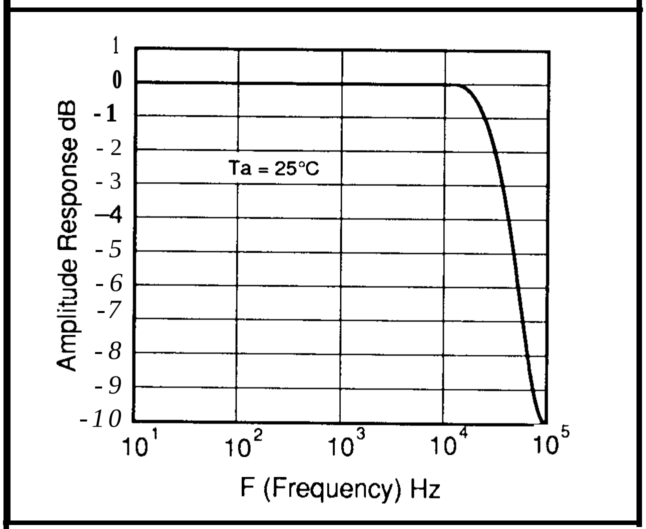


Figure 11—*Photovoltaic amplifier frequency response.*

The amplifier has the gain transfer and linearity characteristics shown in Figure 10. The frequency response is shown in Figure 11. This amplifier has a small signal bandwidth of 45 kHz. Figure 12 shows a photovoltaic isolation amplifier with prebiased amplifiers.

The dynamic input range of the amplifier and the input resistor, $R1$, set the quiescent operation point, I_{P1Q}. The bias is introduced into the inverting input of the servo amplifier, $IC1$, and forces the LED to provide photocurrent, I_{P1}, to servo the input back to a O-volt equilibrium. The bias source can be as simple as a series resistor connected to Vcc. Best stability and minimum offset drift are achieved when a good-quality current source is used. Figure 12 shows an amplifier using two modified Howland current sources. The first source prebiases the servo amplifier, and the second source is connected to $IC2$'s inverting input which matches the input prebias.

This amplifier is designed to respond to a bipolar signal of ±1 V. Given $R1$ = 10k results in a quiescent point of 100 µA. The maximum servo photocurrent, I_{P1}, occurs with a + 1.000-V input. Under this input condition, the photocurrent will be 200 µA. The maximum LED current is calculated using the the servo gain, $K1$, and maximum servo photocurrent, I_{P1}, of 200 µA. With a typical $K1$ of 0.007, the maximum LED current will be 28 mA. The OP-07's output current must

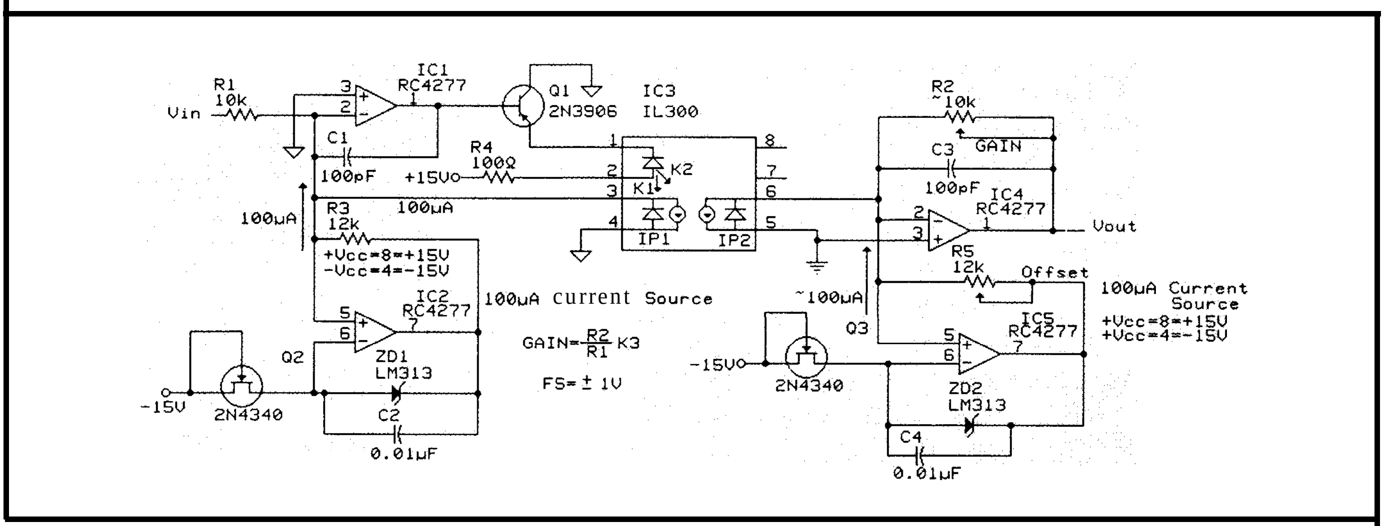


Figure 1 [redacted] *isolation amplifier.*

be boosted in order to handle this LED drive requirement and is accomplished by using a 2N3906 PNP as a buffer transistor between the output of *IC1* and the LED. The buffer transistor has the added effect of reducing the OP-07's thermal drift by reducing the OP-07 internal power dissipation and allowing it to provide the maximum LED current of 28 mA.

The amplifier's output signal is a combination of input voltage times the voltage gain and an offset voltage. This is shown in equations (18) and (19).

$$V_{OUT} = V_{IN} \times \text{Gain} + \text{offset} \qquad (18)$$

$$V_{OUT} = V_{IN} \times K3 \times (R2/R1) + R2 \times (K3 \times I_{b1} - I_{b2}) \qquad (19)$$

To calibrate this amplifier, first adjust the offset (*R12*) resistor for V_{out} = **0** V while the input resistor (*R1*) is grounded. Next apply a stable 1.000-V voltage to the input and adjust the gain resistor (*R2*) for V_{out} = 1 .000 V.

The previous circuit offers a DC- and AC-coupled bipolar isolation amplifier. After calibration the output will be zero volts for an input of zero volts. This circuit exhibits exceptional stability and linearity and has demonstrated compatibility with 12-bit A/D converter systems. The circuit's common mode rejection is determined by CMR of the IL300.

CONCLUSION

The analog design engineer now has a new circuit element that will make the design of isolation amplifiers easier. The preceding circuits and analyses show the variety of isolation amplifiers that can be designed. As a guide, when highest stability of gain and offset is needed, consider the photovoltaic amplifier. Widest bandwidth is achieved with the photoconductive amplifier. Lastly, the overall performance of the isolation amplifier is greatly influenced by the op-amp selected. Noise and drift are directly dependent on the servo amplifier.

The IL300 also has utility in the digital environment. The pulse response of the IL300 is constant over time and temperature. In those critical designs where LED degradation and pulse distortion can cause system failure, the IL300 will eliminate this failure mode. ❑

CONTACT

Siemens Components, Inc.
Integrated Circuit Division
219 1 Laurelwood Road
Santa Clara, CA 95054
Tel: (408) 980-4500

SOURCE

The IL300 is available from:
Pure Unobtainium
Tel: (203) 870-9304

Bob Krause is an applications engineer with Siemens Components Inc.

IRS

413 Very Useful
414 Moderately Useful
415 Not Useful

DEPARTMENTS

FIRMWARE FURNACE

Ed Nisley

Two-Way Power Line Communication

The major portion of most of today's home control systems is power line communication. Ed shows us his version of an X-10 power line interface that can be used with the HCS II or in other applications.

Contrary to what you might suppose, Steve really can be lured out of town. But guess who baby-sits his Home Control System. Right. I get a page of X-10 module addresses, the HCS reload sequence, and a backup diskette. I'm supposedly prepared for anything short of plate tectonic activity.

The day after he left on the last jaunt, his main garage door latch clattered onto the floor. No fried electronics, no smoking power drivers, not even a program bug: a piece of pot metal with remarkable timing picked that day to end its service life.

Oft' the best laid plans...

X- 10 remote control modules are much the same. They are utterly reliable for years, then snap on at 2:30 A.M. every morning for a week, and revert to normal without a hint as to what changed. Steve happened to be acquainted with this feature and designed his original HCS to remind the modules of their state in life every four minutes; a glitched module is reset by the next refresh command.

Elsewhere in this issue you will find Steve's description of Circuit Cellar's new generation home control system. Ken's doing hard work on the supervisory controller (SC) and command compiler, and I'm supplying a few widgets. This column describes the bidirectional power line interface (PL-Link) that the new HCS II uses to control the X- 10 modules scattered throughout the house. The PL-Link has a simple job: convert the SC's orders into X- 10 command sequences and keep the X- 10 modules in line by resending those commands every few minutes.

"The Smart X-10 Controller PL-Link" earns its moniker because it listens to the AC line and maintains a table of module states based on what it hears. Turn a lamp on using a standard X-10 control unit, and PL-Link takes note and refreshes that lamp ON until you (or the HCS II's supervisory controller) turn it off. The SC can extract module states from PL-Link and adjust itself to manual overrides without special one-time program changes, so the whole system is, if not truly intelligent, at least dimly aware.

Because we have covered the X- 10 specs (in gory detail) in previous issues, I will describe how to coerce C into producing X-10 messages. The language is Dave Dunfield's Micro-C, which has sprouted three new memory models and a host of other features since the last issue.

```
D     Dump program status (debugging use)
E     Show and clear error flags (debugging use)
Ln    Set logging mode (bit mapped)
            L            report current mode
            LO           disable (default)
            L1           show received X-10 messages
            L2           show transmitted X-10 messages
            L4           show refresh changes
Nn    Set network/interactive mode
            N            report current mode
            NO           set interactive mode
            N1           network mode (no error messages) (default)
            N2           network mode with command echo (no err msgs)
P     Report power failure status: n,n
            first digit = 1 if power is currently OFF
            second digit = 1 if power failed since last P command
Cl    Query X-10 module status
            Q            report all modules for all housecodes
            Qh00         report housecode h: h=sss... (mod 01 first)
                               O=off    1=on/dim/bright    X=not used
            Ohmm         report module mm for housecode h: hmm=ss
                               OF=off ON=on/dim/bright  XX=not used
R     Set refresh period
            R            report current period in seconds
            Rmm          set period in minutes
            RSss         set period in seconds
            RCmm         clear refresh buffer, set period in minutes
S     Send X-10 message
            Shmmff             send function ff to housecode h module mm
            Sh01 AF            send ALL UNITS OFF command to housecode h
            Sh01 AN            send ALL LIGHTS ON command to housecode h
            Shmmffrr     send function ff to housecode h module mm
                                     function message repeated rr times
            (may be strung together: Shmmff,hmmffrr,hmmff)
```

Figure 1—*The PL-Link* responds *to* the *above* commands sent *over* the serial interface. *In* normal mode, each command must be preceded by the *network* header and controller's address *(!_X-10_). Note that* the *underscores* are *actually* spaces in the message sent

TAKING COMMANDS

The new HCS II supervisory controller communicates with its peripherals through bidirectional RS-485 transceivers using a single twisted pair of wires. The peripherals are all passive until the SC sends them a command or status inquiry; only after decoding a properly formatted and addressed command may a peripheral send anything to the SC. In fact, the peripherals cannot echo command characters because RS-485 communications are half-duplex.

The RS-485 messages must include a header and a fixed-format address field to select the appropriate peripheral, so even though the commands use plain ASCII text, they are fiendishly difficult to type (macro keys help!). PL-Link includes a demo mode that eliminates the address overhead and echoes each character as it is received, giving you manual control using a standard communications program. A text file included with the downloadable code shows how to activate "human" mode.

Figure 1 shows PL-Link's command set. There are only three really useful commands: send an X- 10 message, query the status of a house code or single module, and clear the refresh tables. The remaining commands are handy, but the Big Three do most of the work.

PL-Link uses serial interrupt handlers, ring buffers, and interface routines similar to those I have presented in the past and don't require description here. For me to say that serial interrupts are set to low priority allowing the high-priority power line interrupts to occur without delay is enough. The timer and AC line sync interrupts require only 100-200 microseconds, so they do not cause lost characters.

The Micro-C run-time library includes serial port drivers, but they are not appropriate for the task at hand because they assume ordinary "human" serial input instead of a robot network. Such is firmware....

Listing 1 shows the code invoked by a "Send" command. The outer `while` loop picks off repeated X- 10 commands following the "S" character, extracts and validates each set of parameters, and sends the appropriate X- 10 bit sequences to the power line using the `PwrSendSeq( )` function. The `RefNewEvent()` function enters the module's ON/OFF state in the refresh table.

A complete X- 10 transmission consists of a house and device code to select a particular module, followed by a house and function code to tell that module what to do. The house/device and house/function messages are normally sent twice, except for the DIM/BRIGHT functions, which may be sent up to 32 times. There should be no delays between repetitions, but there must be a delay between the house/device and house/function groups.

The PwrSendSeq() function shown in Listing 2 translates the house, device, function, and repeat values into X- 10 transmission sequences (as I understand the rules, anyway!). In most cases, there will be four X- 10 transmissions for each PL-Link command, each with a different bit pattern or length. PwrFormatMsg() createsthebit patterns and inserts them into a ring buffer, which is emptied by an interrupt routine that sends each bit at the right time.

Before I discuss how bits are sent, an examination of where the bits come from and go to in the hardware is necessary.

BURSTS TO BITS AND BACK

As you can see from Figure 2, there are two big hardware blocks: the TW523 bidirectional power line interface and a COMM-Link microcontroller. You may substitute your favorite microcontroller (with some code changes, of course), but the TW523 is essential. In effect, it is a modem that converts digital data into X- 10 power line signals and returns digital data when it hears a valid X- 10 command on the power line.

The TW523 provides optical isolation between your computer circuitry and the power line's AC voltages and currents. Unlike the earlier transmit-only PL513, the TW523 has two open-collector outputs: in addition to the 60-Hz square wave for zero-crossing synchronization a new bit presents X-10 data pulses received from the power line. Both are open-collector transistors driven by optoisolators, which require external pull-up resistors. The internal 8051 pull-ups are entirely satisfactory for this purpose.

Driving the TW523 data input requires a buffer transistor because an 8051 I/O bit can neither source nor sink the 4.5 milliamps required to drive the optoisolator LED. The PL-Link board shown in the photos has a 2N2907 connected to INT1 for this purpose.

In addition to those three "mandatory" I/O pins, the firmware also provides for three optional LED outputs: a heartbeat to indicate the main loop is running, an indicator that goes on when an X-10 message is being sent, and a power failure indicator. Steve's HCS II runs from +12-volt gel cell batteries, so the X- 10 controller is the only unit that knows when the power goes off!

The PL-Link's RAM and EPROM configuration depends on the C program's memory model. For a Micro-C Compact model program, the EPROM is at OOOO-7FFF and the RAM at 8000–9FFF, with Code and Data spaces overlapped. I edited the start-up code to define those addresses, but your compiler may require different contortions. In any event, make sure the code matches the hardware!

So much for the machinery. Now for the tricky part.

Listing 1—*PL-Link's* **command** *decoding uses a* ***simple*** *switch* ***table to identify*** *each single-character command. This shows* ***what*** *the PL-Link does in* ***response*** *to a "Send" command.*

```
case 'S' :                              /* send X-10 command             */
      while (*pChar){                               /* handle multiple parms */
         ParseErr = CmdGetXArgs(&pChar);
         if (ParseErr){
            Nputstr("*** syntax error\n");
         }
         else {
            PwrSendSeq(CmdHouseNum,CmdModuleNum,CmdFuncNum,CmdRepeats)
            RefNewEvent(CmdHouseNum,CmdModuleNum,CmdFuncNum,CmdRepeats);

      break;
```

PRECISION PULSES REDUX

The key to transmitting and receiving X- 10 messages is maintaining synchronization with the AC power line. The TW523 provides a 60-Hz signal that identifies both zero crossings, but the firmware will be interrupted only by the falling edge. Further, there are tight specs controlling when the three X- 10 transmitted bits must occur within each half cycle and when to sample the TW523 data output for received bits.

Photo 1 shows how the firmware produces the right bits at the right times. The primary synchronization comes from the 60-Hz signal shown in the upper trace, which triggers the 20 interrupts from Timer 0 shown on the lower trace. There are three groups of three interrupts in each half cycle that mark the beginning, middle, and end of each X-10 data bit, and a tenth interrupt to handle "setup" functions for the next half cycle. All of these interrupts occur every power-line cycle, so there are 20 x 60 or 1200 interrupts per second.

A twenty-first timer interrupt will occur if (when?) the 60-Hz input disappears. A state machine triggered by that interrupt times out after eight missing cycles and flags a power failure; there must be eight consecutive good cycles before power becomes "OK" again. This feature copes with power line frequency variations down to 55 Hz as well as glitches without flagging too many errors. The HCS SC can find out whether the power is currently off and whether it has failed since the last status report.

During power failures the timer interrupts free-run at about 55 Hz, with the twenty-first interrupt filling in for the 60-Hz signal. The rest of the firmware is unaware of the change, although X-10 messages don't go anywhere and there is no received data. The program in the supervisory controller decides how to recover from the failure and what to tell the X-10

Listing 2—*Each PL-Link order can produce several X-10 commands. This function adds the proper commands to the transmitter ring buffer for each Send" order.*

```
/*-----------------------------------------------------------------------*/
/* Create X-10 message given all the information...                      */
/* This produces several messages with varying repeats for each part...  */
/* House. device, function are all indexes into the ACLINE tables        */

PwrSendSeq(House,Device,Func,Repeats)
WORD House,
WORD Device,
WORD Func;
unsigned int Repeats;
{

   PwrFormatMsg(House.Device.XTMSGSHORT.1):      /* no pause after msg   */
   PwrFormatMsg(House.Device.XTMSGBITS.1);       /* short pause          */

   Repeats = max(1,Repeats);                     /* ensure good repeats  */
   Repeats = min(255,Repeats);

   if (1 == Repeats) {
      PwrFormatMsg(House,Func,XTMSGBITS,1);      /* ensure pause         */
   }
   else {
      PwrFormatMsg(House,Func,XTMSGSHORT,Repeats-1);      /* no pauses   */
      PwrFormatMsg(House,Func,XTMSGBITS,1);      /* short pause here     */
   }

}
```

modules when they wake up. When power returns, the timer interrupts will lock to the 60-Hz interrupt and normal operation resume.

The interrupt handlers are written in 805 1 assembly language using Micro-C's in-line assembler to embed the code within C functions. Suffice it to say that the code picks a bit from the tail of the transmitter's ring buffer entry, sends it during each of the three bit times, then steps to the next bit. A variable determines how many bits are sent in each message, and when the bits are finished the code chucks up the next ring entry and starts over again.

The firmware also samples the TW523 receiver output at the middle of the first bit time in each half cycle, even while transmitting a message. A state machine finds the start sequence (four successive half-cycles with the otherwise invalid bit pattern 11 10), validates the next 18 half-cycles to assemble a complete message, and adds it to the head of the receiver's ring buffer.

The X-10 transmitter and receiver ring buffers are located in External RAM. The interrupt routines use Internal RAM variables to speed access to the bits and touch External RAM only when a message is complete. This timing provides space for many incoming and outgoing messages while not adding excessive overhead on time-critical parts of the code.

The TW523 receiver is independent of the transmitter, so outgoing messages are echoed back to the PL-Link board. PL-Link updates its refresh table based on both transmitted and received messages to ensure the table always has the latest information. The refresh table remains current even when the TW523 is disconnected or the power fails since outbound messages need not be echoed.

Photo 2 shows a complete X-10 transmission as seen at the PL-Link I/O pins. There are two pairs of X- 10 messages: the first two set the house and device and the second two set the house and function. Remember the 2N2907 transistor inverts outgoing data bits and the TW523 presents inverted bits to INTO; in both cases, a "low" voltage indicates a "1" bit.

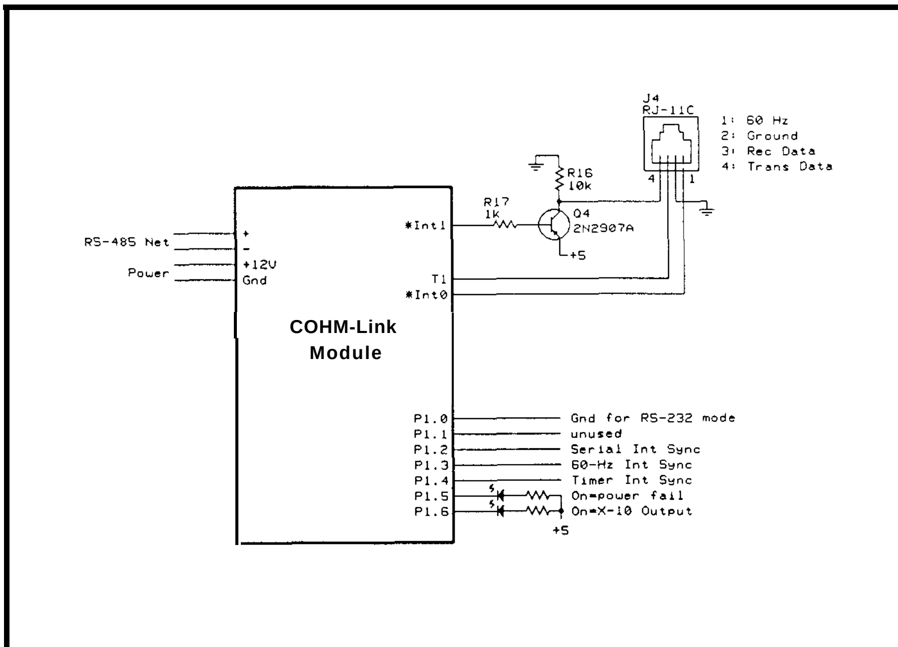


Figure 2-The *PL-Link module uses the standard COMM-Link as ifs basis and adds a TW523 interface. Note that the "heartbeat" LED connected to P1.7 is on the COMA-Link board, so isn't shown here.*

Interestingly, the TW523 does not behave quite like the documentation would have you believe. The receiver output is actually a stored copy of the most recent valid X-10 message; in effect, the TW523 records a message and plays it back immediately, regardless of what happens on the power line. It cannot record and play simultaneously, so you must send each message twice to prevent data loss.

For example, if you send one house/device message (say, "N" and "02") followed by a single house/function message ("N" and "ON"), the TW523 will return only the house/device message. X-10 modules do not have this restriction, so the lamp at NO2 will turn ON. Spending a little time sending various combinations of messages to see how the TW523 reacts is worthwhile.

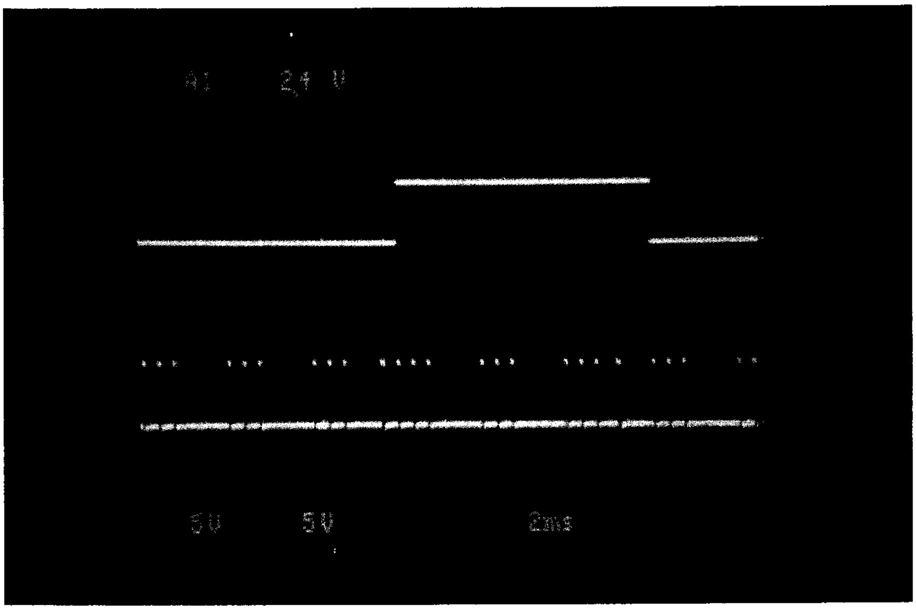


Photo 1—During an X-10 transmission, the primary synchronization comes from the *60-Hz signal shown* in be upper trace, which *triggers* the *20 interrupts from Timer 0 shown on the lower trace.*

MODELING MEMORY

The key to writing good firmware is keeping track of your variables. The 805 1 architecture provides three address spaces in which to store variables: Internal RAM, External RAM, and (for constants only!) in EPROM with the program code. In assembly language, you appreciate full well the differences between the three, but C language syntax masks the subtleties.

In addition to the program's variables, you need RAM to hold subroutine return addresses, save CPU registers during interrupts, and so

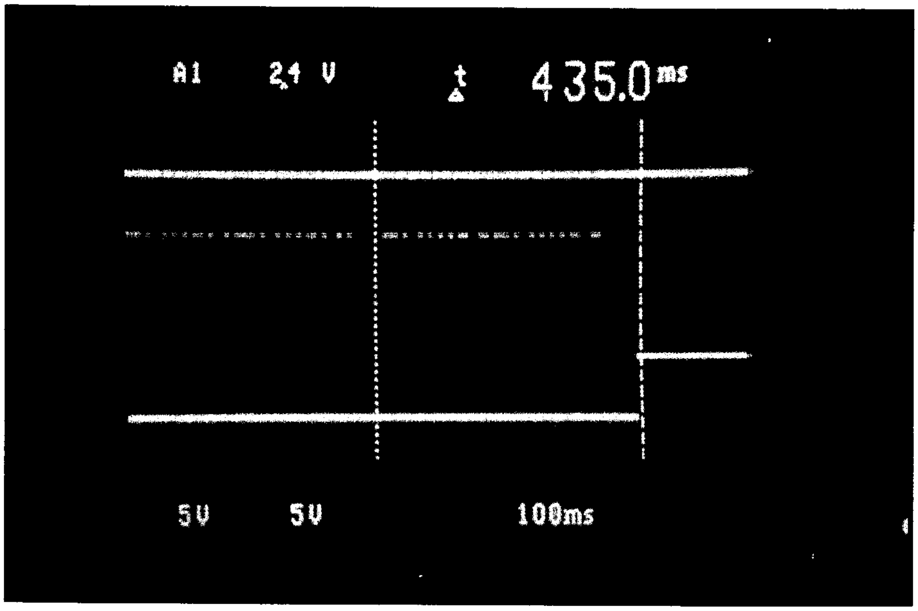


Photo 2-This *complete X-10 transmission consists of hvo pairs of X-10 messages. The first two set he house and device and the second two set the house and function.*

forth. C needs space to pass function arguments, evaluate expressions, and allocate heap storage. The memory model you use tells the compiler where to put everything, but given the sheer number of choices it is no wonder memory models are often confusing.

Micro-C now supports five memory models: Tiny, Small, Compact, Medium, and Large. Although the names may sound familiar to 80x86 C programmers, the meanings are not obvious. Worse, 8051 memory models vary by compiler and "Compact" may mean two different things for two different 8051 compilers.

Remember: it's not what you don't know that hurts you, it's what you think you know that ain't so.

Tiny model is the simplest; everything must fit into the 8051's Internal RAM. Given that there are only 128 bytes of RAM (256 bytes in 8052/32 CPUs), Tiny model programs are easily overwhelmed. Nevertheless, the BLinkBOX and COMBO programs in the last two columns were both Tiny programs, even if BLinkBOX needed 256 bytes of Internal RAM.

Small model allows you to put variables into either Internal or External RAM, but keeps local variables (those defined within C functions] on the Internal RAM stack. Small model may be precisely right for your programs if you forego local variables.

Compact model moves the local variables to a simulated stack in

External RAM, so code using those variables becomes larger and slower. However, this model is a must if your functions twiddle large character arrays or use many local variables. The PL-Link uses Compact model because Internal RAM is chock-full of time-critical global variables, leaving precious little room for local variables on the stack.

All three of these models enforce a nonstandard restriction on your code: any initialized variable is treated as a constant stored in EPROM. While this may sound like a terrible disadvantage, your code ought to have an initialization section anyway. In point of fact, putting true constants in EPROM makes a great deal of sense because they cannot be changed by a glitch or program bug.

In addition, these models require combined Code and Data spaces, which is usually accomplished by ORing the CPU's -PSEN and -RD outputs with an '08 gate. This process limits the total address space to 64K bytes total and forces RAM and EPROM to cover different address ranges. You can now see the reasoning behind PL-Link's peculiar addressing with EPROM from 0000 to 7FFF and RAM from 8000 to 9FFF.

Medium model is similar to Small model, but with separate External RAM and Code spaces. The C start-up code copies all the initialized variables from the Code EPROM to the Data RAM before passing control your C program, which can then alter the values as needed. All local variables must still fit on the Internal RAM stack, so Medium model may not be particularly useful.

Large model is the least restrictive of the five because it uses separate Data and Code spaces and puts local variables in External RAM. Of course, the resulting code can be much larger and slower than any other model, but it's the only way to get 64K of Data, 64K of Code, and not run out of Internal RAM.

In all cases, the CPU return address stack is in Internal RAM because that's where the hardware must find it. If you have allegedly portable C code that twiddles the stack

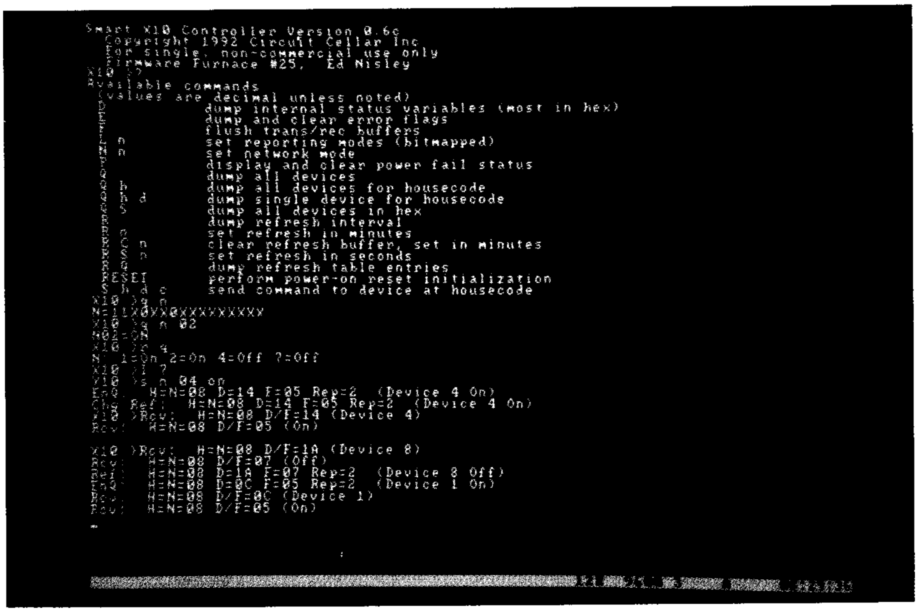


Photo 3-A screen shot of a session with the PL-Link shows what he "human" mode of operation looks like

directly, it won't work correctly in all models because the Internal stack grows up and the External stack grows down.

Many 80x86 C compilers implement the storage class keyword near to identify variables located within the default segment and far for those variables in other segments. Micro-C slightly abuses the register keyword to designate Internal RAM variables. I prefer the former convention, even though it is not directly applicable to 8051 CPUs, so my 8051base.h file has a #define statement to replace all NEAR variables [note the capitalization) with register variables.

Micro-C has a separate start-up file for each memory model. The code defines the starting address and size for the system's RAM and EPROM. I also modified the Compact start-up code to leave room for the interrupt vectors; the changes are detailed in SMARTX10.C.

SUMMING UP

SMARTXIO as implemented on the PL-Link hardware can form the basis of a powerful home control system because it relieves the main supervisory controller of the grubby details (and critical timings) required to send and receive X-10 messages. It uses either RS-485 or RS-232 communications, so you should be able to adapt it to a wide range of systems; I'd be interested in hearing what you come up with.

The hex file required to create a PL-Link EPROM can be downloaded from the Circuit Cellar BBS. PL-Link's source code will not be placed on the BBS, but may be licensed from Circuit Cellar Inc.

The BBS files do, however, include the complete source code for MON X 10, an X-10 monitor program that reports what it hears from a TW523 X-10 interface over the RS-232 serial link. MON X 10 will show you how to create a Micro-C program with in-line assembler and bit manipulations.

Next issue: a people tracker. ❑

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

Ed Nisley is a Registered Professional Engineer and a member of the Computer Applications Journal's engineering staff. He specializes in finding innovative solutions to demanding and unusual technical problems.

IRS

416 Very Useful
417 Moderately Useful
418 Not Useful

Simulating Dynamic System Responses

There are many parallels between mechanical and electrical systems. Understanding how to model dynamic mechanical systems can often increase understanding of other kinds of systems.

PRACTICAL ALGORITHMS

Charles P. Boegli

finding the time response of a mechanical system is a simple matter of four steps. First, determine the transfer function of the system. Second, calculate the Laplace transform of the input. Third, multiply the two together. Fourth, find the inverse Laplace transform of the product. The transfer function arises from physical considerations of masses, spring constants, and damping. The unit step and unit impulse are used extensively as "standard" inputs.

These steps are useful where the transfer function and input can be expressed analytically, but they serve less well where the system is nonlinear or the input transient is arbitrary. When the mechanical system has arbitrary transients, one way to find the response is by numerically solving the differential equations. Several FORTRAN routines are available, among them $TR ES P [1]. Another alternative is simulating the transient response of a mechanical system with spreadsheets. The response is found without solving differential equations, and it proceeds directly from basic and easily comprehended mechanical considerations. The effects of the changes in spring constants and damping are immediately obvious, making these solutions unusually appealing.

Differentiating between *analysis* and *synthesis* of a system is important. A typical synthesis is: given an arbitrary pulse and a desired output response, determine the transfer function of the system to realize the response. This problem is far more difficult than analysis, which starts with the input and the transfer function and then finds the output. In designing mechanical systems, more often trial-and-error analyses are repeated to obtain the desired output than attempting synthesis. A spreadsheet program facilitates such searching, and what it lacks in accuracy it more than makes up in rapidity and convenience.

SINGLE DEGREE OF FREEDOM FIXED SYSTEMS

A typical single degree of freedom fixed linear system is depicted in Figure 1. A mass is suspended by a spring, with its freedom to move controlled by a fluid damper fixed to a support. The differential equation describing this system's behavior is

$$mx'' + cx' + kx = F(t)$$

in which x is the displacement of a mass *m*, *c* is the damping, and *k* is a spring constant. The quantities x' and

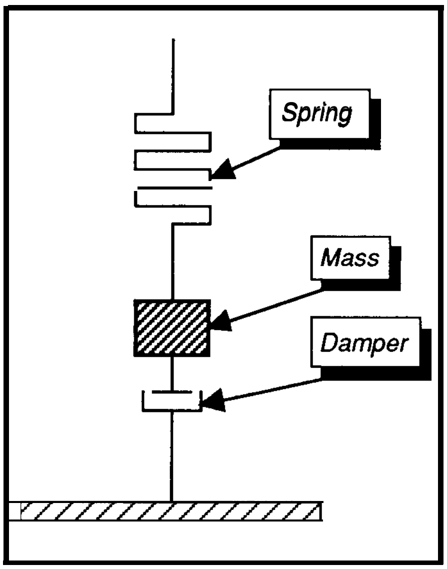


Figure 1—*A typical single degree of freedom fixed linear mechanical system includes a damped mass suspended from a spring.*

x" refer to the first and second derivatives of x with respect to time. The driving function *F(t)* represents the input to the system.

This equation is merely a force summation. Its first term measures the force needed to accelerate a mass. A viscous damper exerts force propor-

Figure 2—*A spreadsheet can be used to simulate the mechanical system in Figure 1. The cell equations used in his spreadsheet are described in the text.*

	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)	(I)
	SINGLE DEGRFE OF FREEDOM FIXED SYSTEM								
	CHARLES BOEGLI								
		MASS	1.000 SLUG			MAX DISPL	1.163		
		SPRING	9.000 LB/FT						
		DAMPER	3.000 LB/(FT/SEC)						
	TIME SEC	INP DISPL FT	SPRING FORCE LB	DAMPER FORCE LB	ACCEL FT/SEC2	VELOCITY FT/SEC	OUT DISPL FT	VEL PREDICT	DISPL PREDICT
1	-0.05	0.000	0.000	0.000	0.000	0.000	0.000		
2	-0.04	0.000	0.000	0.000	0.000	0.000	0.000		
3	-0.03	0.000	0.000	0.000	0.000	0.000	0.000		
4	-0.02	0.000	0.000	0.000	0.000	0.000	0.000		
5	-0.01	0.000	0.000	0.000	0.000	0.000	0.000		
6	0.00	1.000	9.000	0.000	9.000	0.000	0.000	0.000	0.000
7	0.01	1.000	9.000	0.000	9.000	0.090	0.000	0.000	0.000
8	0.02	1.000	8.987	-0.864	8.123	0.176	0.002	0.288	0.001
9	0.03	1.000	8.960	-0.930	8.030	0.256	0.004	0.310	0.004
10	0.04	1.000	8.935	-0.770	8.165	0.337	0.007	0.257	0.007
11	0.05	1.000	8.904	-1.258	7.647	0.416	0.011	0.419	0.011
12	0.06	1.000	8.863	-1.492	7.371	0.492	0.015	0.497	0.015
13	0.07	1.000	8.815	-1.682	7.134	0.564	0.020	0.561	0.021
14	0.08	1.000	8.762	-1.898	6.864	0.634	0.026	0.633	0.026
15	0.09	1.000	8.702	-2.108	6.594	0.701	0.033	0.703	0.033
16	0.10	1.000	8.636	-2.298	6.338	0.766	0.040	0.766	0.040
17	0.11	1.000	8.564	-2.484	6.080	0.828	0.048	0.828	0.048
18	0.12	1.000	8.487	-2.663	5.824	0.888	0.057	0.888	0.057
19	0.13	1.000	8.404	-2.834	5.570	0.945	0.066	0.945	0.066
20	0.14	1.000	8.317	-2.997	5.320	0.999	0.076	0.999	0.076
21	0.15	1.000	8.224	-3.153	5.072	1.051	0.086	1.051	0.086
22	0.16	1.000	8.128	-3.301	4.826	1.100	0.097	1.100	0.097
23	0.17	1.000	8.026	-3.442	4.584	1.147	0.108	1.147	0.108
24	0.18	1.000	7.921	-3.576	4.345	1.192	0.120	1.192	0.120
25	0.19	1.000	7.812	-3.703	4.109	1.234	0.132	1.234	0.132
26	0.20	1.000	7.699	-3.823	3.876	1.274	0.145	1.274	0.145
27	0.21	1.000	7.583	-3.936	3.647	1.312	0.157	1.312	0.157
28	0.22	1.000	7.463	-4.042	3.421	1.347	0.171	1.347	0.171
29	0.23	1.000	7.340	-4.141	3.199	1.380	0.184	1.380	0.184
30	0.24	1.000	7.215	-4.234	2.981	1.411	0.198	1.411	0.198
31	0.25	1.000	7.086	-4.320	2.766	1.440	0.213	1.440	0.213
32	0.26	1.000	6.956	-4.400	2.556	1.467	0.227	1.467	0.227

tional to the velocity of movement, as described by the second term, while the third term shows a spring exerts force proportional to its extension or compression. So long as the system is at rest, all the terms are static. If the system is disturbed, then it responds to restore equilibrium in a manner satisfying the equation.

Imagine the mass of Figure 1 is held in place while the far end of the spring is moved a unit distance away, stretching the spring and exerting force on the mass. At time $t = 0$ the mass is released, and under the influence of the spring and damper it seeks to attain a new position. The equation's solution describes the movement of the mass after its release. Figure 2 is the first page of a spreadsheet simulating this problem. Although quantities are expressed in English units [feet, pounds, seconds), any consistent set of units can obviously be used.

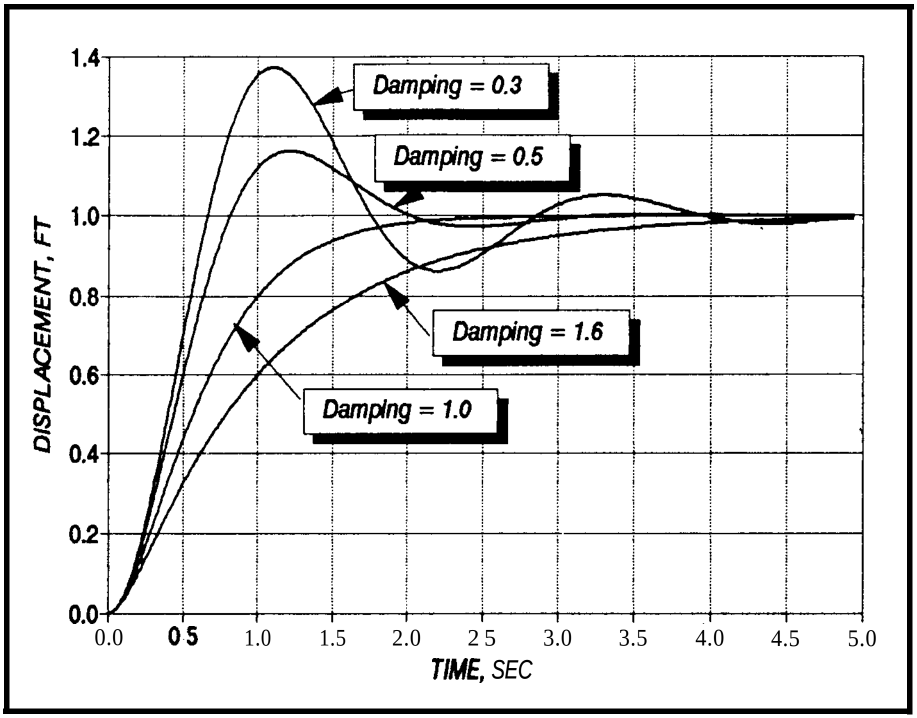


Figure 3—*Responses of the fixed system to a unit step input*

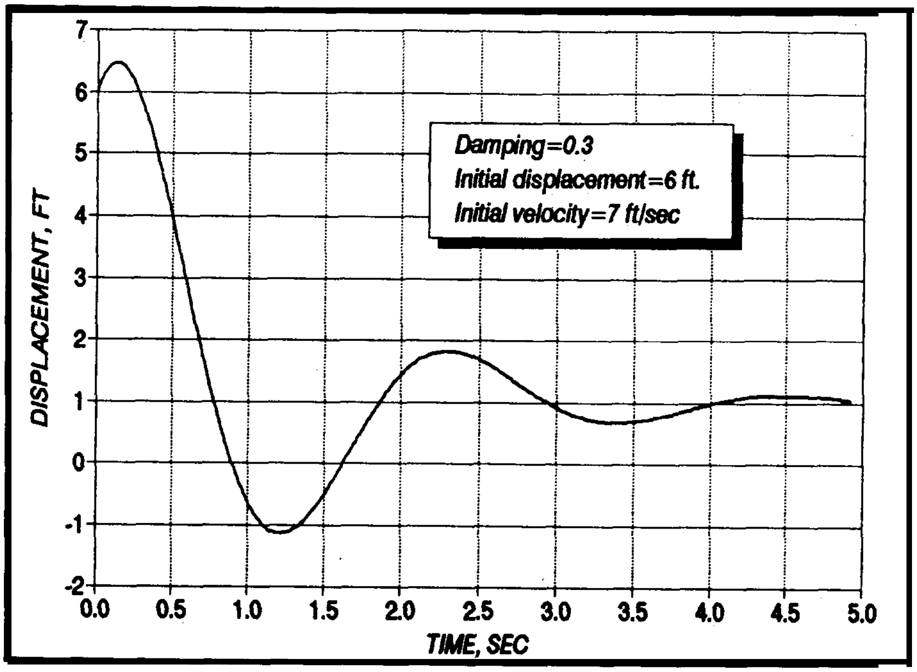


Figure 4—*Adding an initial velocity and displacement produces a modified unit step response.*

Within the body of the spreadsheet, each row describes the acceleration, velocity, and position of the mass at a particular interval in time. The first column (A) contains the time at intervals short enough that accelerations, velocities, and displacements do not change too much (the meaning of "too much" will be clarified later). If such division is impossible (e.g., because the input is not defined for short intervals), then expedients like interpolation can fill the gaps, but poorly interpolated values may introduce errors into the calculation. Several initial rows in all columns are reserved for $t < 0$ to allow the system to be initialized and still have an input transient start at $t = 0$. The time intervals are 0.01 sec, so the first value for time in Figure 2 is $t = -0.05$ sec.

The second column (B) lists the data for an input pulse, which I usually enter item by item. The spreadsheet must be longer than the total number of data points in the input pulse because transients will. continue for some time after the pulse has ended. The rows for $t \leq 0$ contain the initial conditions in which acceleration, velocity, and position are usually zero.

The third column (C) contains instantaneous spring forces, which you find by multiplying the extension of the spring by the spring constant. Because the seventh column (G) contains the output displacement while column (B) lists the input displacement, the value in column (C) is the difference between the values in column (G) and column (B) multiplied by the spring constant; for instance,

```
C7 = (B7-G7)*(spring constant)
```

This entry in the spreadsheet results in a "circular reference" error because I attempted to calculate a quantity from the effect it will have.

Similarly, the fourth column (D) tabulates the forces exerted by the damper, which are proportional to the velocity of movement of the mass relative to the support. The sixth column (F) will list these velocities, so the formula used in column (D) is

```
D7 = -F7*(damping constant)
```

The negative sign is present because the force opposes the motion. This formula also causes a circular reference error.

You can avoid circular reference errors in several ways. One example is to make the time intervals short enough that very little change occurs during any of them. For instance, the contents of cell C7 (for instance) are then approximated by

```
C7 = (B7-G6)*(spring constant)
```

the same device being used for the velocities.

A better way is to use a "predictor" to estimate the content of a cell from the known contents of preceding cells. One predictor I favor is to fit five points with a third degree polynomial by least squares and extraoolate to the sixth point [2]. This has the form

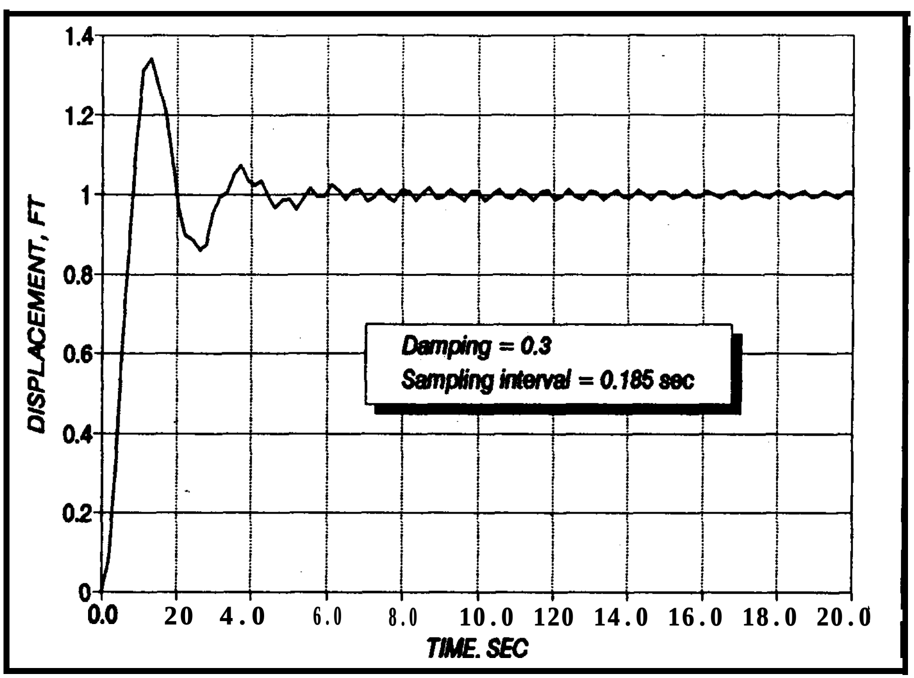


Figure 5—Long *sampling periods often result in instability which shows up as oscillations.*

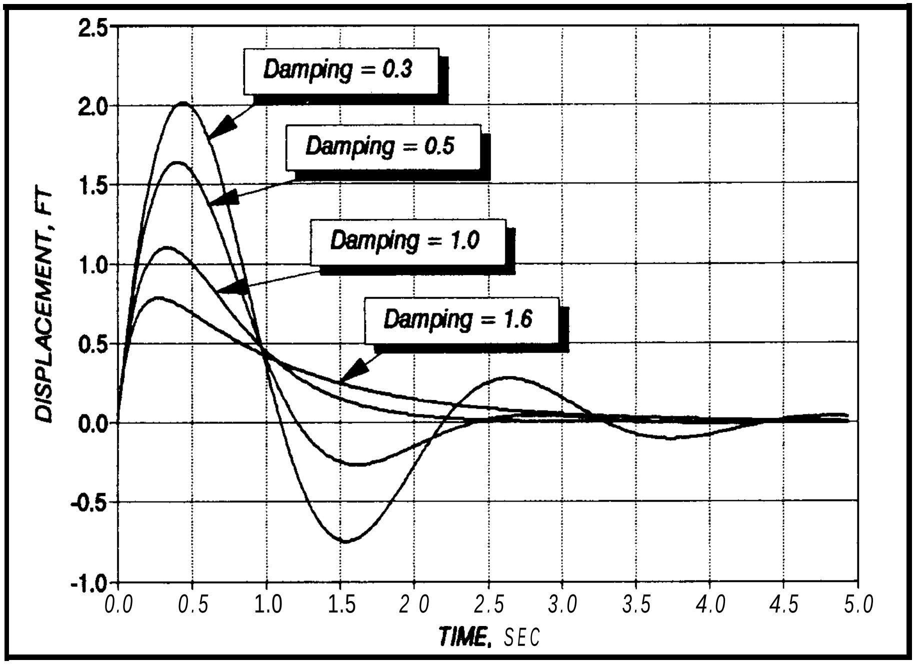


Figure 6—*Unit* **impulse responses** *of the same fixed systems shown in Figure 3.*

$$y_6 = (1/5)(-4y_1 + 11y_2 - 4y_3 - 14y_4 + 16y_5)$$

in which $y_1 \ldots y_5$ are known points and y_6 is the extrapolation, and it requires data points at uniform intervals.

Two added columns (H and I) in Figure 2 contain predictors for velocity and displacement. Displacement predictor calculation, which starts at $t = 0$, requires five preceding displacements and is the reason for including five rows prior to $t = 0$. Column (C), the expression for instantaneous spring force now becomes

```
C7 = (B7-I7)*(spring constant)
```

and that for damper force,

```
D7 = -H7*(damping constant)
```

neither of which causes a circular reference error.

The fifth column (E) expresses the mass acceleration. If an initial value is placed into column (B), then the initial acceleration, determined by that value and the spring constant, will appear in the corresponding row of column (E). Subsequent rows are calculated by

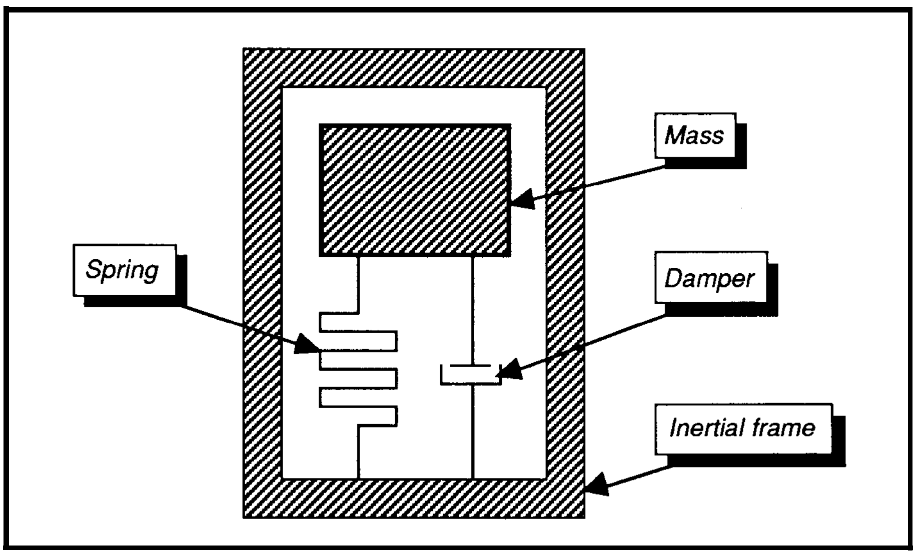


Figure 7—*Moving* **We** *spring* **so it** *is attached to the same support as the damper and allowing the support to move in space results* **in different** *behavior.*

	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)	(I)	(J)	(K)
	GENERAL	INERTIAL	SYSTEM WORK SHEET								
	CHARLES	BOEGLI 9/17/91									
			MASS =		1.00		MAX REL DIST =		1.381		
			SPRING CONST =		9.00		MIN MASS ACCEL =		-13.155		
			DAMPER CONST =		1.80		MAX MASS ACCEL =		0.000		
		<== FRAME			==><			MASS			==>
	TIME SEC	ACCEL FT/SEC2	VELOCITY FT/SEC	DISTANCE FT	VELOCITY FT/SEC	REL VEL FT/SEC	DISTANCE FT	REL DIST FT	SPRING LB	DAMPER LB	ACCEL FT/SEC2
1	-0.050	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2	-0.040	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
3	-0.030	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
4	-0.020	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
5	-0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
6	0.000	-9.000	-0.090	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7	0.010	-9.000	-0.180	-0.001	0.000	0.180	0.000	0.001	0.000	-0.324	-0.324
8	0.020	-9.000	-0.270	-0.004	-0.003	0.267	-0.000	0.004	-0.032	-0.480	-0.512
9	0.030	-9.000	-0.360	-0.007	-0.008	0.352	-0.000	0.007	-0.060	-0.633	-0.693
10	0.040	-9.000	-0.450	-0.011	-0.015	0.435	-0.000	0.011	-0.095	-0.782	-0.878
11	0.050	-9.000	-0.540	-0.016	-0.024	0.516	-0.000	0.015	-0.138	-0.929	-1.067
12	0.060	-9.000	-0.630	-0.022	-0.035	0.595	-0.001	0.021	-0.188	-1.071	-1.260
13	0.070	-9.000	-0.720	-0.028	-0.047	0.673	-0.001	0.027	-0.245	-1.211	-1.456
14	0.080	-9.000	-0.810	-0.036	-0.062	0.748	-0.002	0.034	-0.309	-1.347	-1.656
15	0.090	-9.000	-0.900	-0.045	-0.078	0.822	-0.002	0.042	-0.380	-1.479	-1.859
16	0.100	-9.000	-0.990	-0.054	-0.097	0.893	-0.003	0.051	-0.457	-1.607	-2.064
17	0.110	-9.000	-1.080	-0.064	-0.118	0.962	-0.004	0.060	-0.541	-1.732	-2.273
18	0.120	-9.000	-1.170	-0.076	-0.140	1.030	-0.006	0.070	-0.630	-1.853	-2.483
19	0.130	-9.000	-1.260	-0.088	-0.165	1.095	-0.007	0.081	-0.726	-1.971	-2.696
20	0.140	-9.000	-1.350	-0.101	-0.192	1.158	-0.009	0.092	-0.827	-2.084	-2.911
21	0.150	-9.000	-1.440	-0.115	-0.221	1.219	-0.011	0.104	-0.934	-2.194	-3.128
22	0.160	-9.000	-1.530	-0.130	-0.253	1.277	-0.013	0.116	-1.046	-2.299	-3.346
23	0.170	-9.000	-1.620	-0.145	-0.286	1.334	-0.016	0.129	-1.164	-2.401	-3.565
24	0.180	-9.000	-1.710	-0.162	-0.322	1.388	-0.019	0.143	-1.286	-2.499	-3.785
25	0.190	-9.000	-1.800	-0.180	-0.360	1.440	-0.022	0.157	-1.414	-2.593	-4.006
26	0.200	-9.000	-1.890	-0.198	-0.400	1.490	-0.026	0.172	-1.546	-2.683	-4.228
27	0.210	-9.000	-1.980	-0.217	-0.442	1.538	-0.030	0.187	-1.682	-2.769	-4.450
28	0.220	-9.000	-2.070	-0.238	-0.486	1.584	-0.035	0.202	-1.822	-2.850	-4.673
29	0.230	-9.000	-2.160	-0.259	-0.533	1.627	-0.040	0.219	-1.967	-2.928	-4.895
30	0.240	-9.000	-2.250	-0.281	-0.582	1.668	-0.046	0.235	-2.115	-3.002	-5.117
31	0.250	-9.000	-2.340	-0.304	-0.633	1.707	-0.052	0.252	-2.267	-3.072	-5.339
32	0.260	-9.000	-2.430	-0.328	-0.687	1.743	-0.058	0.269	-2.422	-3.138	-5.560

Figure 8— *The inertial system in Figure 7 can be simulated with a spreadsheet. As before, the cell equations are described in the text.*

summing the forces acting on the body and dividing them by its mass; the formula is

```
E7 = (C7 + D7)/m
```

The next column (F) lists mass velocities. The velocity in any cell is the velocity in the preceding cell modified by the acceleration. The first cells hold the initial velocity that, again, may be arbitrary but is usually zero. For subsequent cells

```
F7 = F6+(E6+E7)/2*(A7-A6)
```

The last term takes into account the duration of the acceleration.

The last column (G) lists mass displacements. The first entries are the initial displacement, usually zero. Subsequent values are found from

```
G7 = G6+(F6+F7)/2*(A7-A6)
```

All these formulas are copied into the remaining cells of the spreadsheet.

To use the spreadsheet, enter values for the mass, spring constant, and damping constant, as well as initial values for mass displacement [column G) and velocity (column F). You must edit the formulas to include proper values for mass, spring constant, and damping constant; the calculation of the shock response is automatic.

Figure 3 shows the motions of the mass in a system composed of a mass of 1.0 slug and a spring rate of 9 lb/ft when the initial velocities and displacements are zero. The curves are for damping forces of 1.8, 3.0, 6.0, and 9.6 lb/(ft/sec). Plotting the responses of the same system with other initial conditions is a matter of changing numbers in the $t \leq 0$ rows of the spreadsheet. For example, Figure 4 represents the response of the system with a damping force of 1.8 lb/(ft/sec), when the initial velocity is changed from 0 to +7 ft/sec and the initial displacement from 0 to +6 ft. The input transient remains a unit step function, so the ultimate value of the output is still 1.0, but the mass settles down to this value quite unlike it does with zero initial values.

ACCURACY

This technique simulates the behavior of the system from moment to moment, calculating the behavior of the mass as the forces exerted on it change. That small errors in each calculation lead to a very large error in the final result has been known for a long time, and has given rise to a number of more complicated methods of synthesis aimed at eliminating accumulated errors.

On the other side of the spreadsheet coin lie two facts, the first being these programs carry out calculations with high accuracy, retaining a number of decimal places rare in the days of mechanical calculators. Arithmetic errors are reduced almost to the vanishing point, and those remaining are from the formulas themselves. Quite often, you will know the final state of the system, after transients have died away, which serves to check the accuracy of the simulation. For instance, Figure 3 shows the problem stipulates the mass must rest ultimately at a displacement of 1.0.

What remains is the transient behavior of the system in which inaccuracies often cannot be estimated. The advantage of this simulation method is it reduces to child's play determining the behaviors of systems complex enough to be almost impossible to find analytically. The question, whether an approximate solution is better than no solution at all, must be answered affirmatively.

Simple first-order systems are readily solved analytically. Several rapid checks are available to verify the accuracy of behaviors calculated by numerical simulation. The calculated dampings of the systems in Figure 2 are 0.3, 0.5, 1.0 [critical damping), and 1.6, respectively. Underdamped system responses rise to a peak and then decay both exponentially and sinusoidally. The height of the first peak is one measure of accuracy. (For systems with critical or greater damping, the time to reach some percentage of final position is often used.)

For the underdamped systems, those having a damping of 0.3 rise to 1.372 times the final rest position before beginning to decay, while those having a damping of 0.5 rise to 1.163. Agreement between these figures and a published graph [3] appears excellent.

This method of simulation occasionally exhibits instability when a steady-state value is maintained for some time. It is easy to detect because successive values of displacement oscillate rapidly (usually at the sampling frequency) about the mean output; transients oscillating about zero are rare. Usually, you can delay or eliminate the instability by reducing the time intervals.

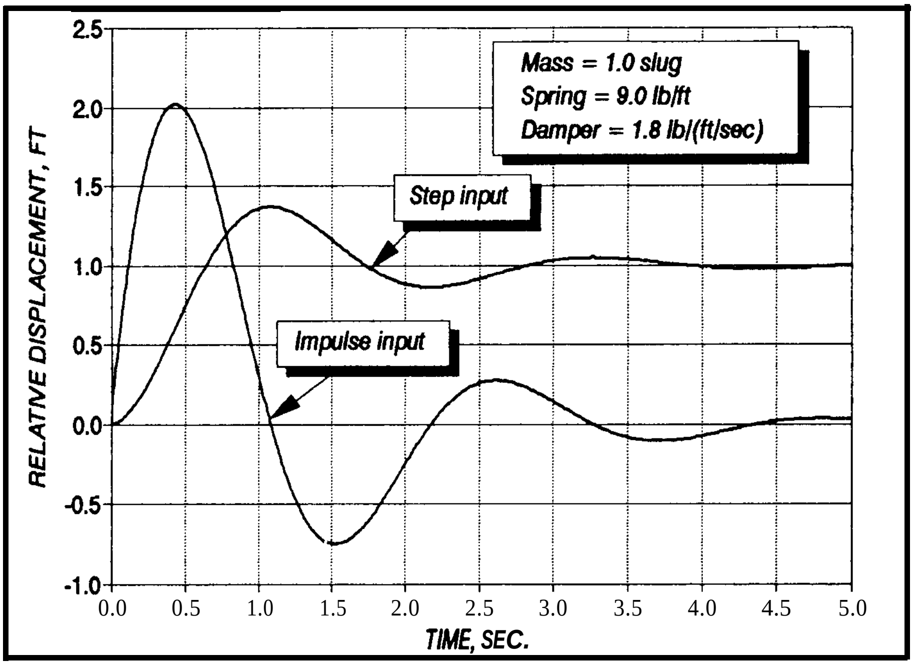


Figure 9—The motion of the maw relative to the frame in the inertial system is practically identical to that shown in Figure 3 for the fixed system.

To give you an idea of how large the time intervals can be to obtain accurate transient response and avoid instability, I repeated the calculations for damping equal to 0.3 with larger time intervals. Figure 5 shows the onset of instability with a sampling interval of 0.185 sec; however, even with this coarse interval, the peak transient amplitude dropped only slightly, to 1.343.

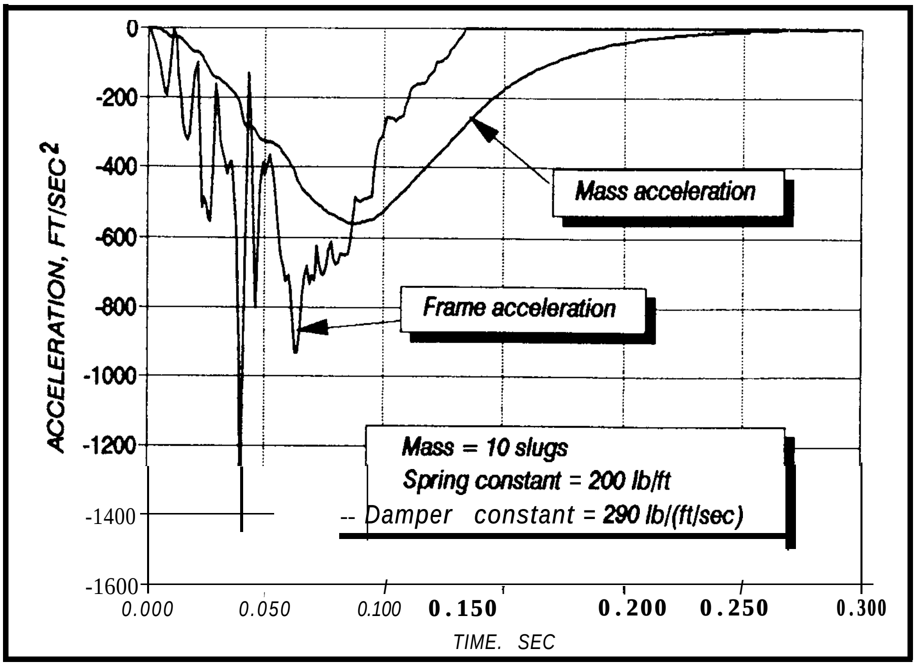


Figure 10—The inertial system response to an arbitrary acceleration applied to the system.

The unit impulse, a spike of infinite height, infinitesimal width, and area of 1.0, cannot be easily entered into a spreadsheet. However, you can find the response to a unit impulse by differentiating the response to a unit step. In the spreadsheet of Figure 2, I obtained the displacement by integrating the velocity. Differentiating the displacement is pointless because the derivative is already available in the velocity column. The displacement response to a unit impulse is the same as the velocity response to a unit step. Figure 6 shows unit-impulse responses for the systems used in Figure 3.

SINGLE DEGREE OF FREEDOM INERTIAL SYSTEMS

The damper in Figure 1 is fastened to a fixed support. If it is instead attached at the same point as the spring, as depicted in Figure 7, the system is not the same as that shown in Figure 1 because displacement of the support acts on the damper as well as the spring. However, the system of Figure 7 lacks any fixed supports and can move independently in space.

Accelerating the inertial system of Figure 7 is the same as moving the spring end in Figure 1. The spreadsheet of Figure 8 [4] verifies this fact by simulating the motion of the mass relative to the inertial frame. In this sheet, column (A) is devoted to time, column (B) to an arbitrary acceleration of the inertial frame, column (C) to its velocity, and column (D) to its displacement. The formulas for row 7 are

C7 = +C6+B7*(A7-A6)
D7 = +D6+C7*(A7-A6)

Columns (E) through (K) refer to the suspended mass. Column (E) calculates the velocity of the mass from the previous velocity and acceleration using

E7 = +E6+K6*(A7-A6)

In column (F) the relative velocity [the velocity of the mass in relation to that of the inertial frame) is

F7 = +E7-C7

Similarly to column (E), the distance of the mass in found in column (G) from

G7 = +G6+E7*(A7-A6)

while column (H) computes the relative distance of the mass using

H7 = +G7-D7

In column (I) the force exerted by the spring is

I7 = -H7*(spring constant)

while in column (J) that exerted by the damper is

J7 = -F7*(damper constant)

The last column presents the acceleration of the mass, found by

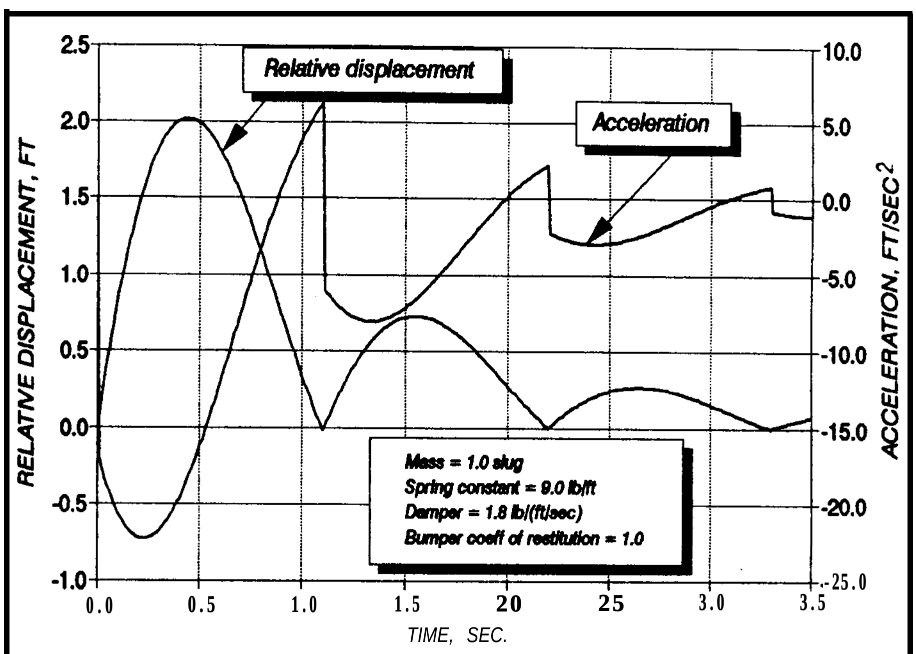


Figure 11 -The inertial system response to an impulse acceleration applied to the system.

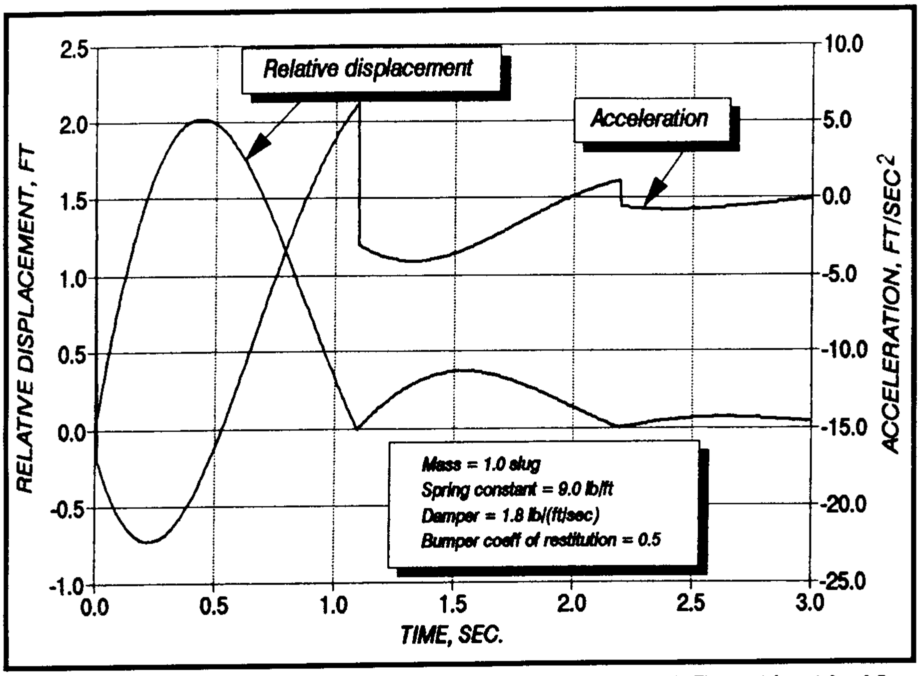


Figure 12—*Reducing the coefficient of restitution of the rubber bumper in the system in Figure 11 from 1.0 to 0.5 produces a somewhat more damped response.*

summing the forces and dividing by the mass m:

```
K7 = (I7+J7)/m
```

For quick reference, I set in the header the peak relative distances and maximum mass acceleration.

The spreadsheet in Figure 8 uses initial values of zero for all columns, a mass of 1.0, a spring constant of 9 lb/ft, and a damping constant of 1.8 lb/(ft/sec), just as in the previous sheet. The -9 ft/sec^2 step input of acceleration produces a final deflection of 1.0 foot to the mass. You can also obtain directly the relative displacement curve for impulse input by inserting an acceleration of -900 ft/sec^2 into the spreadsheet at $t = 0$. Figure 9 shows the motion of the mass relative to the frame is practically identical to that in Figure 3 for the fixed system.

I obtained all of these response curves without solving any equations, merely by introducing mechanical conditions into the proper columns.

ARBITRARY INPUT PULSES

The spreadsheet can simulate the response of an inertial system to an arbitrary deceleration. Actual impact-test deceleration readings were manually entered in column (B), with a time increment of 0.001 second. Figure 10 graphs the input (frame) and output (mass) decelerations. The suspension reduced the peak deceleration of the mass from 1450 to 550 ft/sec^2.

NONLINEAR SYSTEMS

Much work has been done during recent years on damping, and devices obeying different laws than those used in the equation are common. In fact, linear dampers are the exception. The force of fixed-orifice hydraulic dampers (e.g., shock absorbers) is proportional to the square of the velocity.

Springs that are not linear, or those in which force does not even vary with extension, are articles of commerce. Such devices, which introduce severe difficulties into the analytical solutions of the response equations, are easily handled by these spreadsheet programs. For instance, a quadratic (square law) damper requires the formula for damper resistance to be changed to

```
J7 = +F7*@ABS(F7)
        *(damper constant)
```

where the absolute value, which preserves the sign of the damper force, is necessary because the damper opposes motion equally in either direction.

Figure 11 shows the response of a system similar to that of Figure 9 to a 9 ft/sec^2 pulse. However, in this system, a perfectly resilient bumper is set against the mass at rest position. The mass can move away from the bumper freely, but when it encounters the bumper going in the other direction it bounces away with the same velocity as that with which it struck. I had to reduce the sampling interval from 0.01 sec to 0.005 sec, starting at 2 sec, to avoid the oscillation I mentioned earlier.

Figure 12 is for the same system, but now the coefficient of restitution of the rubber bumper is reduced from 1.0 to 0.5. Figure 13 is for the same system, but with a preloaded spring for which the force is 5 + 8 x (deflection) pounds.

In the last three examples, the expression in El6 (for example) is

```
@IF(H15<0, [A]*C16 - (E15
   + [B]*K15*(A16 - A15)),
   E15 + K15*(A16 - A15))
```

in which *[A]* is 1.0 plus the coefficient of restitution, and *[B]* is the coefficient of restitution. This statement translated into English is: if the relative distance is is less than zero, then the mass velocity is (one plus the restitution coefficient) times the frame velocity minus (the previous mass velocity plus the restitution coefficient times the acceleration times the time interval), otherwise it is the previous mass velocity plus the (acceleration times the time interval). As you've probably guessed, setting up these expressions can be a headache.

The final acceleration in Figure 13, -5 ft/sec^2, represents the force of the prcloaded spring on the mass when the mass rests against the bumper.

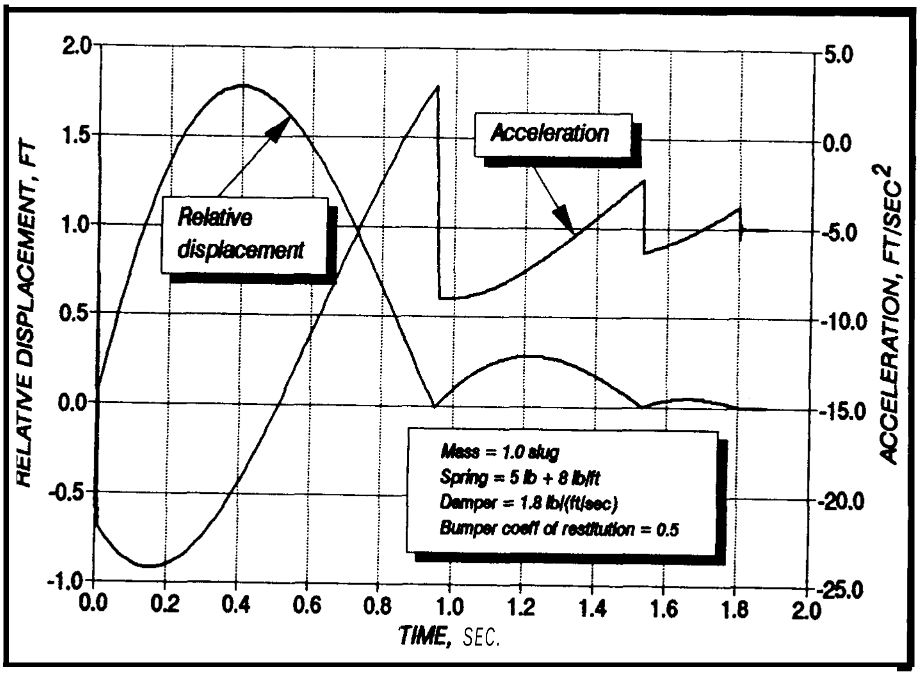


Figure 13—*Changing the system in Figure 12 to use **a different spring yields a slightly different graph.***

CLOSING REMARKS

Spreadsheets like those used in this paper are of great utility in analyzing systems. You can more rapidly write the nucleus of a high-level language program to carry out the analysis, with assurance that the mathematical expressions and operations work properly; the spreadsheet eliminates most of the debugging.

I originally intended to include two degree of freedom systems [which as a class are more important than single] and to also consider vibration, which may introduce a new set of problems. I quickly realized that the range of possibilities opened by each incursion into a new facet of the technique would be a many months' task.

With proper care, spreadsheet simulation can be a valuable tool for analyzing shock responses. It is especially worthwhile with systems that have severe nonlinearities and are difficult or impossible to handle by classical techniques. But where this method appears to show an unexpected phenomenon, view it with some skepticism and subject the results to a number of tests (such as reduction of the sampling time interval).

On the other hand, the method works well even when input data are not available at fixed time intervals. Particularly because it reduces the calculation of shock response to a technique understandable to any high-school student, I hope it will be of value in the development of new devices, where the ability to deal easily with nonlinear systems may permit exploring a wider range of design possibilities.

ACKNOWLEDGEMENT

The encouragement of my friend and fellow computer enthusiast Jeff Spray is gratefully acknowledged. ❑

REFERFNCES

1. Tse, Morse, and Hinkle, ***Mechanical Vibrations/Theory and Applications,*** Second Edition (Boston, Allyn, and Bacon Inc., 1978), Appendix C.
2. Charles P. Boegli, "Functions of Tabulated Experimental Data," unpublished ms.
3. John G. Tuxal, ***Control System Synthesis*** **[New** York: McGraw-Hill Book Company, 1955), 39.
4. This spreadsheet does not employ predictors because they do not work well with the nonlinear systems I will address shortly. For this reason the sampling time intervals must often be shorter than in the spreadsheet of Fig. 2.

Charles Boegli is president of** Randen **Corporation in Blanchester, ***Ohio.*** *Randen **is a small company offering services in technical computer programming and analog circuit design.***

419 Very Useful
420 Moderately Useful
421 Not Useful

Does It Come With a Memory... Standard?

What happens when a standard is written to be everything to everybody? Share Jeff's adventures as he wades through the emerging credit-card-size module standards.

FROM THE BENCH

Jeff Bachiochi

Just Say NO! Isn't that what we tell our kids? When you begin to feel uncomfortable with something, "Just say NO." Well, I'm feeling a bit uncomfortable right now. I'm unsure of a decision I made based on conversations with a few manufacturers' reps. You see, what I thought was, may not be. And if it isn't, then it just doesn't make sense. You know? Let me try to explain.

One of the problems facing both software and hardware engineers using microcontrollers is the limited amount of address space available for data collection. Whether it's a fast sampling rate or just logging over a long period of time, storage space is often at a premium. Although I might start with 64K of space available, it quickly gets whittled down to 8K or 32K at best. Disk drives are an alternative for logging data. Unfortunately, they are heavy and require a reasonably clean environment to perform well. Transporting between systems is a pain because most operating systems have their own unique recording format.

If you use a RAM disk as a fast storage device, you know the benefits of the speed it has over mechanical drives. You also know its downfall if you've shut down your computer without saving its files to either a floppy or a fixed disk. A mass storage device made with nonvolatile RAM would be just the ticket for logging data and easily transporting it between environments.

IT'S NOT YOUR AVERAGE MEDIUM

Many times new products are based entirely upon new parts. Newer technologies produce faster, more powerful, and (hopefully) cheaper parts. Designing around new parts is dangerous, especially if they are only manufactured by one source. You can end up with a super widget, but go down the drain because you can't get the new part in any quantity, the price goes through the roof, or worse yet, it gets discontinued altogether.

I grumble about standards whenever I don't have what I need. How many different spark plug or oil filter wrenches do you own! Ever noticed

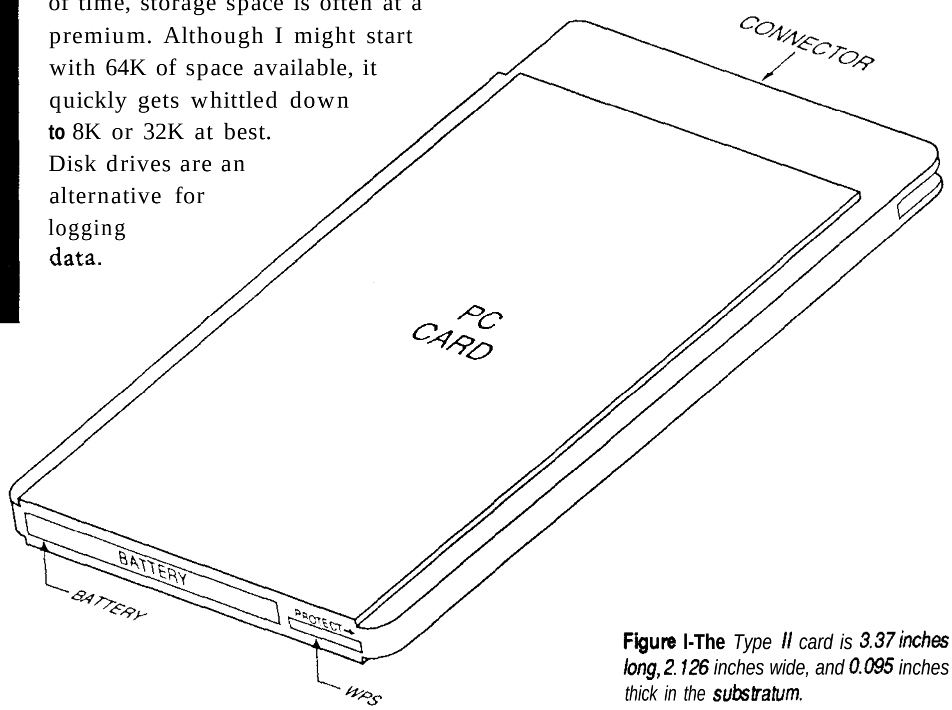


Figure 1-The *Type II card is 3.37 inches long, 2.126 inches wide, and 0.095 inches thick in the substratum.*

that no matter what height you set your bookshelves to, there is always a book that is too tall to fit? Have you gone to the store to pick up replacement watch batteries or vacuum cleaner bags and stood there dumbfounded, staring at the choices? Give me a break!

My search (this time) was for some kind of mass storage device, similar to a 3.5” diskette, but solid state. PCMCIA kept popping up in all the trade journals, along with ads for SRAM, FLASH, EPROM, ROM, and OTP cartridges. I’d seen credit card memories, but every manufacturer seems to have their own connection interface. It was a fad trying to take root. What I needed was a little standardization. Reps were telling me PCMCIA was doing just that.

It seems a memory card standard has been released. In fact, the hardware form factors were defined back in 1985, although the first release (1 .O} of the standard was not published until May, 1990. I requested information on the released standard and started gathering parts so I could begin prototyping as soon as the design popped out of the LaserJet as a schematic wiring print.

SECOND THOUGHTS

In about a week a package appeared on my desk. It was a two-volume set of references from PCMCIA (PC Memory Card International Association). At this point I knew I’d gotten more than I bargained for. I stared blankly at the two volumes, and visions of vacuum cleaner bags raced through my mind. When I regained consciousness, I cracked open volume one.

PCMCIA was formed to handle the technical aspects of the new memory card standard as well as the marketing and promotional activities to support it. Working closely with JEIDA (Japan Electronics Industry Development Association) ensured compatibility between U.S. and Japanese releases. The primary goal of this standard was to enable system and card manufacturers to build compatible products without regard to “what’s under the hood.” Although

Pin	Signal	I/O	Function	Signal	I/O	Function
1	GND		Ground			
2	D3	I/O	Data bit 3			
3	D4	I/O	Data bif 4			
4	D5	I/O	Data bit 5			
5	D6	I/O	Data bit 6			
6	D7	I/O	Data bit 7			
7	CE-1	I	Card Enable			
a	A10	I	Address bit 10			
9	OE	I	Output Enable			
10	AI 1	I	Address bit 11			
11	A9	I	Address bit 9			
12	A8	I	Address bit 8			
13	AI3	I	Address bit 13			
14	AI4	I	Address bit 14			
15	WE/PGM	I	Write Enable			
16	RDY/BSY	0	Ready/Busy	IREQ	0	Interrupt Request
17	vcc					
18	Vpp1		Programming Supply Voltage 1			
19	AI6	I	Address bit 16			
20	AI5	I	Address bit 15			
21	AI2	I	Address bit 12			
22	A7	I	Address bit 7			
23	A6	I	Address bit 6			
24	A5	I	Address bit 5			
25	A4	I	Address bit 4			
26	A3	I	Address bit 3			
27	A2	I	Address bit 2			
28	A1	I	Address bit 1			
29	A0	I	Address bit 0			
30	D0	I/O	Data bit 0			
31	D1	I/O	Data bit 1			
32	D2	I/O	Data bit 2			
33	WP	0	Write Protect	IOIS16	0	IO Port is 16 bit
34	GND		Ground			
35	GND		Ground			
36	CD1	0	Card detect			
37	DII	I/O	Data bit 11			
38	D12	I/O	Data bit 12			
39	D13	I/O	Data bit 13			
40	D14	I/O	Data bit 14			
41	D15	I/O	Data bit 15			
42	CE2	I	Card enable			
43	RFSH	I	Refresh			
44	RFU		Reserved	IORD	I	IO Read
45	RFU		Reserved	IOWR	I	IO Write
46	AI7	I	Address bit 17			
47	A18	I	Address bit 16			
48	A19	I	Address bit 19			
49	A20	I	Address bit 20			
50	A21	I	Address bit 21			
51	vcc					
52	Vpp2		Programming Supply Voltage 2			
53	A22	I	Address bit 22			
54	A23	I	Address bir 23			
55	A24	I	Address bit 24			
56	A25	I	Address bit 25			
57	RFU		Reserved			
58	RESETI		Card Reset			
59	WAIT	0	Extend bus cycle			
60	RFU		Reserved	INPACK	0	Input Port Acknowledge
61	REG	I	Register select	REG	I	Reg Sel & I/O Enable
62	BVD2	0	Battery voltage detect 2	SPKR	0	Audio Digital Waveform
63	BVD1	0	Battery voltage detect 1	STSCHG	0	Card Statuses Changed
64	D8	I/O	Data bit 6			
65	D9	I/O	Data bit 9			
66	D10	I/O	Data bit 10			
67	CD2	0	Card detect			
68	GND		Ground			

Figure 2-The signals on the *left side* of *the table* are found *only* on the original memory card interface. The signals on the *right* side are those added or changed *for an I/O* and memory card *interface*.

Figure 3—*The Card Information Structure is a variable-length linked list of data blocks called tuples.*

Code	Name	Description
0	CISTPL_NULL	Null tuple—ignore
1	CISTPL_DEVICE	The device-information tuple (Common Memory)
2-7	CISTPL_CHECKSUM	(Reserved for future, upward-compatible versions of the device-tnformation tuple.)
8–0Fh	CISTPL_LONGLINK_A	(Reserved for future, incompatible versions of the devcie-information tuple.)
10h	CISTPL_LONGLINK_C	The checksum-control tuple
11h	CISTPL_LINKTARGET	The long-link control tuple (to Attribute Memory)
12h	CISTPL_NO_LINK	The long-link control tuple (to Common Memory)
13h	CISTPL_VERS_1	The link-target-control tuple
14h	CISTPL_ALTSTR	The no-link-control tuple
15h	CISTPL_DEVICE_A	Level 1 version/product-information tuple
16h	CISTPL_JEDEC_C	The alternate-language-string tuple
17h	CISTPL_JEDEC_A	Attribute Memory devcie informatron
18h	CISTPL_CFIG	JEDEC programming information for Common Memory
19h	CISTPL_ENTRY	JEDEC programming informatron for Attribute Memory
1Ah	CISTPL_DEVICE_OC	The configurable-card tuple
1Bh	CISTPL_DEVICE_OA	The configurable-entry tuple
1Ch	CISTPL_VERS_2	Other operatrng conditions device information for Common Memory
1Dh	CISTPL_FORMAT	Other operating conditions device information for Attribute *Memory*
1Eh–3Fh	CISTPL_GEOMETRY	(Reserved for future standardization)
40h	CISTPL_BYTEORDER	The Level-2 version tuple
41h	CISTPL_DATE	The format tuple

the initial focus was IBM PC-compatible systems, the spectrum has widened to include all computer types and noncomputer consumer products.

Sounds like these people have their act together. So where's the monkey wrench?

The two-volume set I received was Release 2.0, issued September 199 1, and along with it were a few surprises. The Electrical Specification of Release 1.0 has been altered. New specifications allow the cards to contain hardware other than memory (e.g., fax, modem, or other I/O devices). The interface is beginning to look more and more like a PC's expansion slot. The new I/O devices can even execute code directly from within, providing the host hardware implementation supports it. Does any of this information sound familiar? I thought it was supposed to be a universal interchange medium. These higher level functions sound processor specific to me. What originally was a simple beauty is slowly turning into a complex beast. Is it all too much? Are you feeling a bit uncomfortable? Let's back up a bit here and briefly look at the three parts that make up the PCMCIA Standard.

THE PHYSICAL STANDARD

This section defines the memory/I/O card's physical dimensions, the connection system, and qualifying test parameters. All cards are 3.37 inches long by 2.126 inches wide. The Type I card is 0.065 inches thick. The Type II card is slightly thicker, 0.095 inches in the substratum [Figure 1). Other physical standards include labeling specs and an optional WP (write protect] switch and battery location.

Socket contacts are used on the cards, while the host connector uses pins. The interesting part of the host connector's design is the length of the pins. There are three lengths used, and they sequence the connections whenever a card is inserted or removed. The longest pins are power and ground, which ensure the card is powered prior to applying signals to the card's I/O connections, the second length pins. Finally, the shortest pins are engaged, which signal to the host that insertion is complete. Card guides built into the host connector are keyed to ensure proper insertion and engagement.

Mechanical performance specs include a guaranteed number of insertions, insertion and ejection forces, vibration and shock immunity, contact resistance, and current carrying capacity. Environmental performance specs include resistance to moisture, shock, temperature, chemicals, electrical discharge, X-rays, ultraviolet light, and electromagnetic fields. An interesting note here: "harsh environment" is defined as no air-conditioning or humidity control. Hooey, I wouldn't want to work in a place like that!

THE INTERFACE STANDARD

The PC Card interface features a 64M-byte address space (AO-A25) using either an 8- or 16-bit data bus. As in other 16-bit systems, A0 is not used with 16-bit words because each

Byte \ Bit	7	6	5	4	3	2	1	0
0	TPL_CODE		CISTPL-DEVICE (Olh) or CISTPL_DEVICE_A (17h)					
1	TPL_LINK		Link to next tuple (at least m-l)					
2			Device Info 1 (2 or more bytes)					
			Device Info 2 (2 or more bytes)					
			(etc.)					
			Device Info n (2 or more bytes)					
m			FFh (marks end of device info field)					

Figure 4-*Each tuple begins with a tuple code byte, which defines the type* of information *held within the tuple block*

Byte\Bit			
	Device Type Code	WPS	Device Speed
	Extended Device Speed (if Device Speed Code equals Eh, otherwise omitted)		
	Additional Extended Device Speed (if bit 7 of Extended Device Speed is 1, otherwise omitted)		
	Extended Device Type (if Device Type Code equals Eh, otherwise omitted)		

Figure 5a—*The* **device info** *field can be from one to low bytes long.*

Code	Name	Meaning
0	DTYPE_NULL	No device. Generally used to designate a hole in the address space. If used, speed field should be set to 7h.
1	DTY PE_ROM	Masked ROM
2	DTYPE_OTPROM	One-time programmable PROM
3	DTY PE_E PROM	UV EPROM
4	DTYPE_EEPROM	EEPROM
5	DTYPE_FLASH	Flash EPROM
6	DTYPE_SRAM	Static RAM (JEIDA has nonvolatile RAM)
7	DTYPE_DRAM	Dynamic RAM (JEIDA has volatile RAM)
8-Ch		(reserved for future use)
Dh	DTYPE_IO	I/O Device
Eh	DTYPE_EXTEND	Extended type follows
Fh		(reserved for future use)

Figure 5b—*Within the device info field is the device type code,* **which defines** *one of sixteen types.*

address consists of two bytes, one at an even address and one at the next odd address. Two ● CE (card enable) inputs control the width of the data bus. If the host uses one ‘CE, the data bus is defined as eight bits wide. ● OE (output enable) and *WE (write enable) control the direction of data flow to and from the memory card. RDY/*BSY signal informs the host when the memory card is ready for the next access. BVD1 and BVD2 reflect the status of the backup battery and the validity of the data within the memory card. VPP1 and VPP2 are the programming voltage inputs for cards requiring a supply other than VCC. You may have seen some ads recently for a Maxim chip that provides this supply from 5 V. Figure 2 shows all the pin definitions.

Five pins designated as reserved on Release 1.0 have been defined for Release 2.0. RESET clears the card’s configuration options, returning it to the default power-up state. *WAIT delays the completion of a memory access or I/O cycle in progress. ‘IORD and ● IOWR signals handle data transfers after the host has configured the card as an I/O device. *INPACK (input acknowledge) signals the host that the card can respond to an I/O read cycle presently being addressed.

Four pins previously defined in Release 1.0 can be redefined by the host in Release 2.0 by writing to the configuration register. WP becomes ● IOIS16 (I/O is a 16-bit port] and informs the host that the I/O card can handle 16-bit data transfers. RDY/● BSY becomes ● IREQ (interrupt request) and requests the host to perform a software service. BVD2 becomes ● SPKR, a source of digital audio waveforms. Lastly, BVD1 becomes ‘STSCHC (status changed), which indicates the OR’d conditions of all the original signals now redefined by the host and presently only available through the cards status register.

All memory device types are supported. In addition to the common memory or I/O address space, a separate attribute memory area can be selected for card identification purposes. Not all memory cards presently under production have this attribute memory used for card configuration, and there are some still unresolved issues that deal with the programming specifications of EPROMs and EEPROMs.

THE SOFTWARE STANDARD

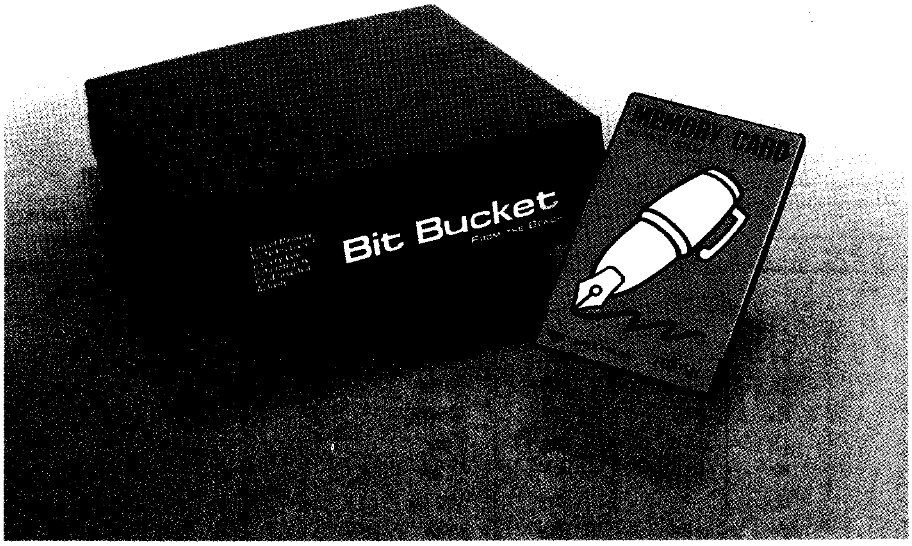
In order to support many incompatible data-recording formats, a metaformat hierarchy was developed. As of Release 2.0, five layers have been defined from the simplest (0) to the most complex (4). Layer 0 specifies the interface characteristics but does not require any configuration information.

The Basic-Compatibility Layer 1 cards must have attribute memory containing some primary information about the card’s size, speed, and type. This data is recorded using a Card Information Structure (CIS) starting at address 0 in the attribute memory. The CIS is a variable-length linked list of data blocks called “tuples” [see Figure 3). Each tuple begins with a “tuple code byte” [Figure 4) which defines the type of information held within the tuple block. For instance, tuple code 01H describes device information about the common-memory space used in the card. The second byte of each tuple is an offset pointer to the next tuple and in doing so implicitly defines the length of the tuple. The remaining bytes of the tuple contain pertinent information. In the case of tuple code 01H, this information would be in a device type and speed description byte [up to four bytes for some nonstandard device descriptions), followed by a device size byte (see Figures 5 and 6).

Layer 2 specifies the Data-Recording format, which falls into one of two categories. The first is the disk-type format in which data blocks are formed (like disk sectors) and blocks are written and read as a whole. The second is the memory-type format where each byte is directly addressable. As with disk-type formatting, block data checking can be implemented using checksums or CRCs. Tuple codes for Layer 2 start with 40H.

The Data-Organization Layer 3 defines how the data in a particular partition is organized. It may contain a DOS or other OS-type file system. It may be a memory-type or an application- or vendor-specific file system. This layer uses tuple 46H and is intended to help the host choose the appropriate file system. ASCII text descriptions of the organization are implemented allowing the host to notify the operator of the card’s file system even if it is not supported.

Layer 4 describes the System-Specific standards, which are an interchange format for all cards formatted with a DOS FAT-based file system, a standard for directly executable programs, and a way of interpreting older cards not using CIS. The interchange format would ensure free interchange of information between DOS systems as long as they are supported, but were not limited to, this basic format. The basic format is defined as a disk-type format of 512-byte blocks with no error checking. Two types of support are defined for the directly executable XIP (execute In Place) programs: the LIM 4.0 standard and the 386 extended-addressing mode. To ensure the older memory cards not supporting the attribute space or CIS are still useful, a procedure is defined for interpreting what is found on a card and using the interchange format as a minimal foundation.



THE END (OF VOLUME ONE)

OK, everyone take five.. . has your head stopped reeling? At this point I know much more than I need to or even want to. I was originally concerned with the possibility of designing in a device that could be extinct tomorrow. After having digested volume one, I feel comfortable knowing this reference material isn't the kind prepared for a product likely to see an early death.

For those of you who are curious as to what's in volume two: it provides the software developer with enough information to create a "Socket Services" handler specific to the

Code	Name	Meaning
Oh		(Reserved—do not use)
Ih	DSPEED_250NS	250 ns
2h	DSPEED_200NS	200 ns
3h	DSPEED_150NS	150 ns
4h	DSPEED_100NS	100 ns
5h–6h		(Reserved)
7h	DSPEED_EXT	use extended speed byte

Figure 5c—*Also within the device info field is the device sped. Four popular speeds are currently defined with room for more.*

7	6	5	4	3	2	1	0
EXT	Speed Mantissa				Speed Exponent		

Mantissa Code	Mantissa Meaning	Exponent Part Code	Exponent Part Meaning
Oh	Reserved	Oh	1 ns
Ih	1.0	Ih	10 ns
2h	1.2	2h	100 ns
3h	1.3	3h	1 µs
4h	1.5	4h	10 µs
5h	2.0	5h	100 µs
6h	2.5	6h	1 ms
7h	3.0	7h	10 ms
8h	3.5		
9h	4.0		
Ah	4.5		
Bh	5.0		
Ch	5.5		
Dh	6.0		
Eh	7.0		
Fh	6.0		

Figure 5d—*When none of the predefined device speeds apply, an extended device speed may be defined.*

system's hardware interface. This volume will allow you to develop higher-level software without regard to the system-specific hardware.

Is my fear without foundation? I thought I was getting a standard interface that would accept memory cards of differing types and sizes. What I actually got was a functionally expanded interface with specifications requiring expanded software support to identify and implement anything imaginable. My plans for these memory cards do not include fancy formatting and file handling capabilities, although they could be implemented. At this point, I am satisfied with a simple sequential-access nonvolatile mass-storage device. Present SRAM cards top out at about 5 12K bytes, but I can take advantage of every new device size as it's released by designing with the PCMCIA interface standard.

CREATING A BIT BUCKET

I know a logging device of such capacity will be handy in future

7	6	5	4	3	2	1	0
# of address units – 1					Size Code		

Code	Units	Max Size
0	512 bytes	16K
1	2K	64K
2	8K	256K
3	32K	1M
4	126K	4M
5	512K	16M
6	2M	64M
7	Reserved	Reserved

Figure 6—*Within* **he device** *ID structure,* **following** *the device speed/type information, is a* **device size** *byte*

situations, so it must be as universal as possible. Because simplicity is paramount, only four modes of operation are necessary: serial record and playback, and parallel record and playback. Using serial and parallel interfaces will allow this device to be used by most computers or controllers. The serial modes will be of a fixed data rate (configured in software) with no handshaking; just a serial stream of data limited only by the character transmission speed. The parallel modes make use of strobe and acknowledge signals to handshake the data into or out of the system. Initially, there will be no provision for recording multiple files on one memory card. All data is simply streamed contiguously into *(or* **out** of) the bit bucket (memory) until the system is turned off or the last available address is accessed.

Next time we'll look at the "Bit Bucket" in detail. I'll define both the hardware and software necessary to collect mass quantities of data using the PC Card memory devices. At least now I have the confidence to say "Yes" to the use of the physical and interface standards. As far as the software standard goes, let's run it up the flagpole and see who salutes.

SOURCE

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

Jeff ***Bachiochi (pronounced "BAH-key-AH-key") is an electrical engineer on the Computer Applications Journal's engineering staff. His background includes product design and manufacturing.***

IRS

422 Very Useful
423 Moderately Useful
424 Not Useful

Cheap Chips

Lean and Mean PIC Machines

What do you do when discrete gates aren't enough and a regular microcontroller is just too much? Pick a PIC and come out just right.

SILICON UPDATE

Tom Cantrell

One of the first things an aspiring marketer learns is to choose one's words carefully. As a junior chip peddler (er...sales consultant), I remember being told that "cheap" was a word to be avoided. Period. Pick your euphemism ["cost sensitive," "high volume," "cost over performance") but for heaven's sake, not "cheap"!

Along the same lines, simply stating your product's features is considered amateurish. You must instead relate the "benefits" that will accrue to those customers wise enough to choose you over competitors. With apologies to Woody Allen, the ultimate sales pitch is something like:

FEATURE: Super-duper See-Thru Look-ahead Pipe-dream
BENEFIT: you a) will be loved and b) will never die.

Gee, where do I sign.

Maybe I'm burned-out from the ever-escalating hype permeating the high-tech biz, but now is time for a new era of glasnost marketing. I want to see a data sheet like:

*Faster than a bat out of hell!
*Easier to use than a politician!
*Cheap!
*Cheaper!
*Cheapest!

Here are some chips from Microchip Technology that fill the bill.

PIC AND CHOOSE

Let me just say up front, continuing with my no-bull approach, the PIC 16C5*x* series has what are probably the cheapest micros around. I'm talking as little as $2 in volume for a complete OTP (One-Time Program; i.e., EPROM in a no-window package) micro with RAM, EPROM, and I/O.

Many times, I have been faced with the need for a little piece of logic to perform some fairly mundane task many times. Usually, the choices boiled down to either wiring up a few 74*xx* TTLs or using one of the popular 80*xx* or 68*xx* micros.

The problem is often neither approach is ideal. The TTL-gates approach may be best for high-volume applications (or for those who agree with our illustrious leader, Steve Ciarcia, when he says, "my favorite programming language is solder"), but wire-wrapping and debugging more than a few boards quickly becomes tiresome. Furthermore, you're faced with diving back into the rat's nest when you change your mind (inevitably] about how it should work.

The classic 8-bit micro may be the easiest approach, but it can be expensive. A "single-chip" micro was traditionally either ROM based (i.e., minimum feasible order size is thousands) or windowed EPROM (expensive; e.g., $10 for an 8748). Another alternative is using a "non-single-chip" micro, with low-cost external RAM and EPROM, but then I'm back to the wire-wrapping blues (or more often than I care to admit, just throwing a $100 single-board computer at what should be a $5 application).

Admittedly, recent technologies have closed the gap. On the gate side, PALs can easily consolidate 5-10 74*xx*s. Meanwhile, the classic 8-bit EPROM-based single-chip micros are beginning to appear in OTP versions at lower cost.

Despite this convergence on the needs of low cost and volume applications, these approaches always leave me feeling a little uncomfortable about the waste involved because I like to squeeze every bit of functionality out of a given technology. The gates approach is often speed overkill (i.e., you don't need a 50-MHz PAL to toggle an LED) while the micro

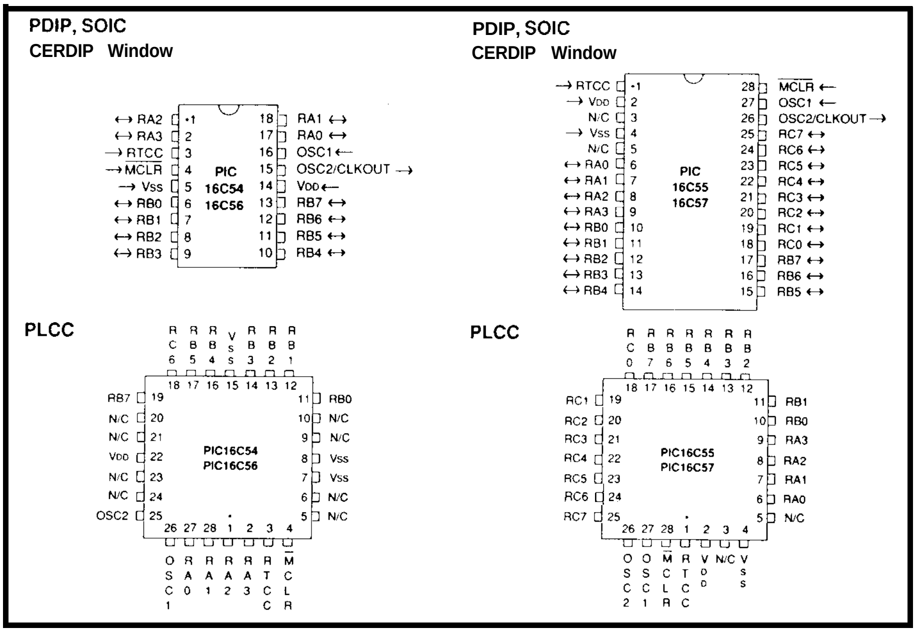


Figure 1—*PIC processors* **are available in a** *number* **of configurations that** *fit* **a range of** *applications.*

approach is complexity overkill (e.g., an 8K-byte, 32-I/O-bit micro with 7.5K bytes of NOPs and 24 no connects). Read on to see how the PICs uniquely fill the gap between gates and regular 8-bit micros.

CHEAP RISC

Figure 1 shows the pin-outs for the two basic versions (18-pin and 28-pin) of the PIC. Right away the dual personality of the chip becomes apparent; it is an 8-bit micro in small gate-like package (especially the 18-pin DIP version). Why pay for more if you don't need to?

With so few pins, explaining their function becomes blessedly easy. Note the only I/O difference between the 18-pin and 28-pin versions is the latter has an additional 8-bit I/O port (RC0–RC7). In addition to the general-purpose I/O lines, MCLR' is a reset input while OSC1 and 0SC2 are the CPU clock lines. The oscillator is unique because it can accept a simple RC (Resistor/Capacitor) as well as traditional crystal or TTL clock source. Save money by using the RC option if you're willing to accept less speed and accuracy as shown in Figure 2. RTCC is an input that can clock an on-chip 8-bit counter [with optional 1:2, 1:4, ... 1:128 prescaler). Alternatively, the on-chip counter and prescaler can be driven by the internal clock divided by four [for example, 5 MHz for a 20-MHz PIC).

The RISCy nature of the PIC becomes clearer moving on-chip (Figure 3). Let me check off some of the traditional RISC criteria against the PIC.

*Fixed Length Instructions: Yep. Every PIC instruction is 12 bits, and different models offer 512 (16C54/55), 1K (16C56), or 2K (16C57) instruction capacity. Programming is easy because there are only 38 different instructions, which is about as reduced as possible.

Cext	Rext	Fosc @ 5.0V, 25°C	
20 pF	5.1k	3.33 MHz	±22%
	10k	1.85 MHz	±22%
	100k	189 kHz	±38%
100 pF	5.1k	1.18 MHz	±15%
	10k	668 kHz	±15%
	100k	67.8 kHz	±26%
300 pF	5.1k	46 kHz	±10%
	10k	254 kHz	±13%
	100k	25.1 kHz	±21%

Figure 2—*If lower* **speeds and less accuracy are acceptable** *tradeoffs for lower cost, a simple RC circuit may be used* **in place of a crystal.**

•Pipelined, Single-Cycle Execution: OK, with the understanding that a single cycle is four clock periods [as shown in Figure 4). With a two-level pipeline (fetch and execute overlapped), the PIC's "5-MIPS" performance (at 20 MHz) leaves most other 8-bit micros eating dust.

*Load/Store: The PIC fulfills this most fundamental of RISC precepts, subject to a little handwaving. All PIC instructions reference "W," a File Register (FR), or both. In one interpretation, W is a type of "accumulator" while FRs (the '54, '55, and '56 offer 32 while the '57 offers 80 FRs) are "RAM." In this light, the PIC isn't Load/Store because instructions can operate directly on "memory" (the FRs). However, if you consider W as a "temporary register" and FRs as "regular registers," the instructions only work on "registers" per the Load/Store criteria. Actually, this limitation doesn't matter because Load/Store, which addresses the issue of "slow" external memory accesses versus "fast" on-chip register accesses, becomes a rather meaningless concept when all "memory" is on-chip.

*Harvard Architecture: this criterion refers to the use of separate bus and memories for instructions and data, which the PIC exploits. Though not really a RISC tenet, Harvard architecture is hyped as such on complex RISC chips like the AMD 29000 and Motorola 88k.

Of **course**, the PIC does not have cache (another meaningless concept when all memory is on-chip), delay slots, branch prediction, speculative execution, superscalar, superpipeline, or superanything. These omissions are not surprising because the PIC doesn't even have interrupts and it barely has a stack (two levels only). That's OK, because not only do you get what you pay for but you pay solely for what you need.

You do get some handy features like a power-on-reset timer that eliminates the external RC usually required, something that should appeal to designers who are really cheap.. . oops!... who are concerned about minimizing system cost. You also get a watchdog timer with a clever feature:

it operates off a clock circuit separate from the CPU. This aspect is handy because the watchdog remains vigilant even if the CPU clock is stopped. Stopping the clock is something you might want to do because the PIC is "static" and includes a sleep mode. These features make very low speed, voltage, and power [i.e., battery] operation possible. For example, a PIC consumes a miserly 3 volts at 32 microamps with 32-kHz clock—wow!

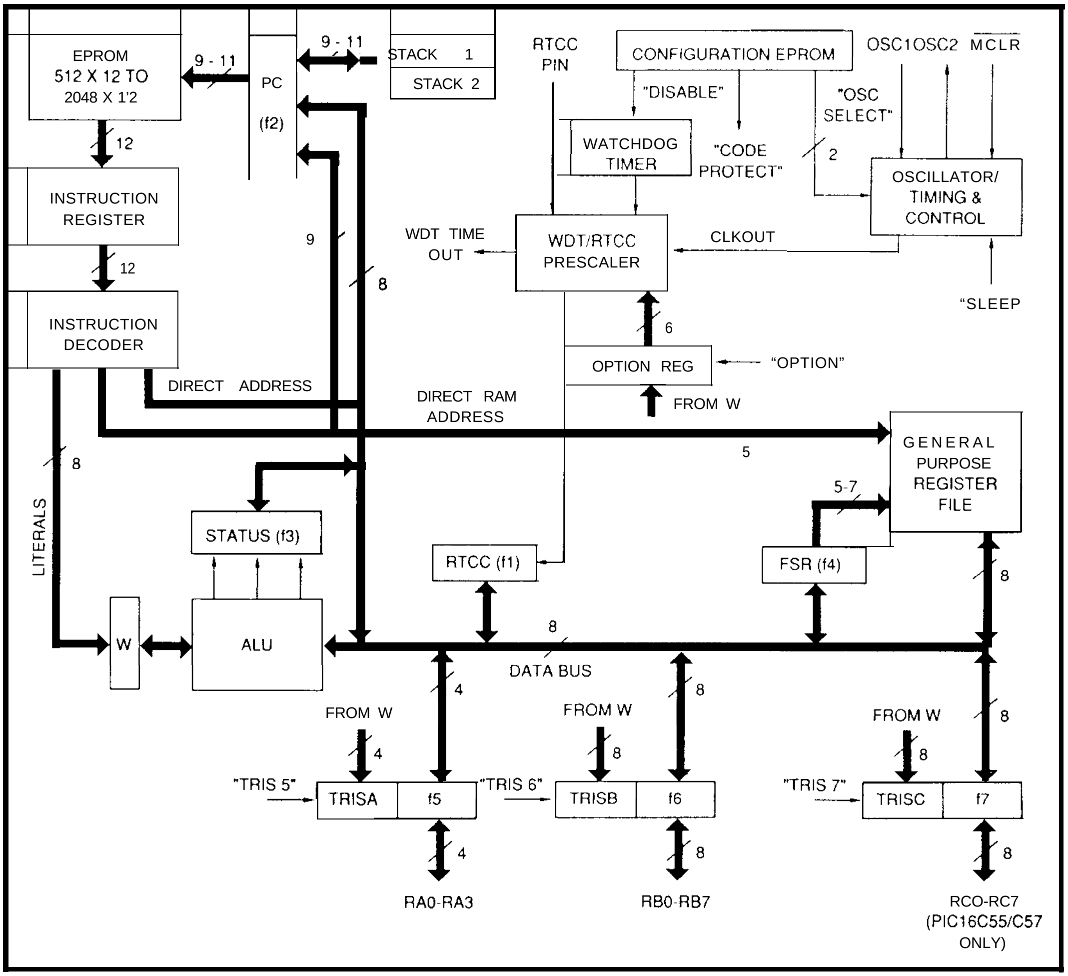


Figure 3—*RISC-like features of We PIC processor include fixed-length instructions; pipelined, single-cycle execution; load/store; and Harvard architecture. Other useful features include a power-on-reset timer, a watchdog timer which operates* off *a clock independent of the processor's, and a sleep mode.*

CHEAP TOOLS

I tend to take up-front tool cost for granted because I have all kinds of assemblers, simulators, emulators, and programmers gathering dust. Thus, I haven't been in the market recently, but I guess the minimum tools setup for a PAL or 8-bit micro runs at least $1000, and I know it can get much higher.

Cheap chips need cheap tools, and here Parallax Inc. fills the bill. They offer a complete PC-based development package, including assembler, "emulator," and device programmer for only $449. You can also buy small quantities of the PIC at very decent prices (as low as $3) directly from Parallax.

I say "emulator" because it isn't the classic emulator with expensive features like real-time trace or source-level debug. Rather, it more closely corresponds to an EPROM emulator you might use with a regular micro.

The Parallax setup basically implements a downloadable RAM version of the PIC with a few extras, like power supply, switch-selectable clock source, and so forth. These features allow quick code change/debug iterations without the cost and time to burn an actual EPROM version of the chip.

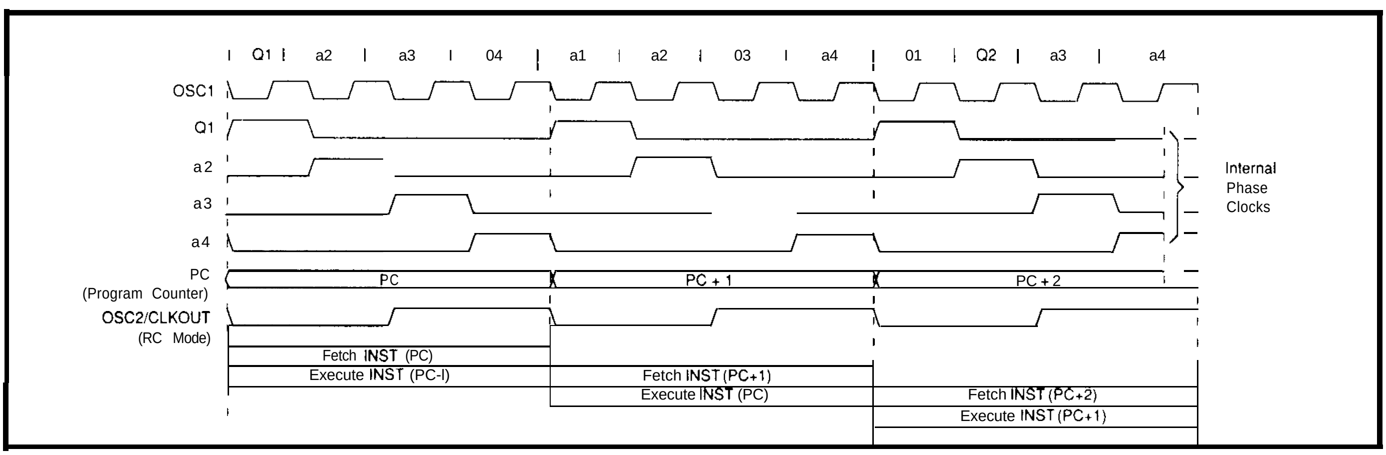
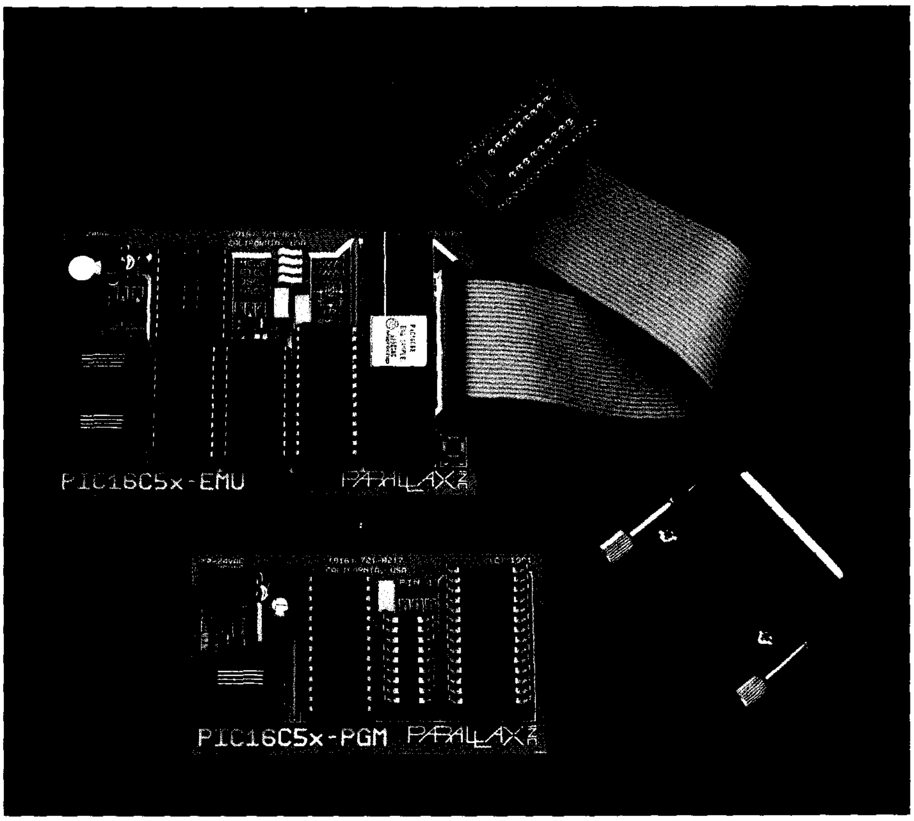


Figure 4-The *PIC operates with a four-c/o&period cycle and overlaps instruction execution with the fetch of he next instruction.*

The emulator/programmer setup (combined: a tidy 20 sq. in.) connects to the PC printer port on one side and to an 18- or 2%pin DIP header on the other using a 6” ribbon cable, which plugs into the target PIC socket. Each board is powered by a small wall-mount transformer, so they can be used together or separately.

Entering a program on the PC using your favorite editor, and then assembling that program using PASM . EXE completes the developmental procedure. Next, you can use **PEP. EXE** to download the object code to the emulator, run the program, and after everything is working, burn a PIC using the device programmer. The latter includes LIF (low insertion force) sockets for both the 18- and 28-pin PICs.



A key point is the Parallax **PASM** redefines the instruction syntax, so it differs from the “official” definition by Microchip. Normally, such a move would be considered taboo, but I’m letting it pass for two reasons. First, I don’t have to worry about maintaining

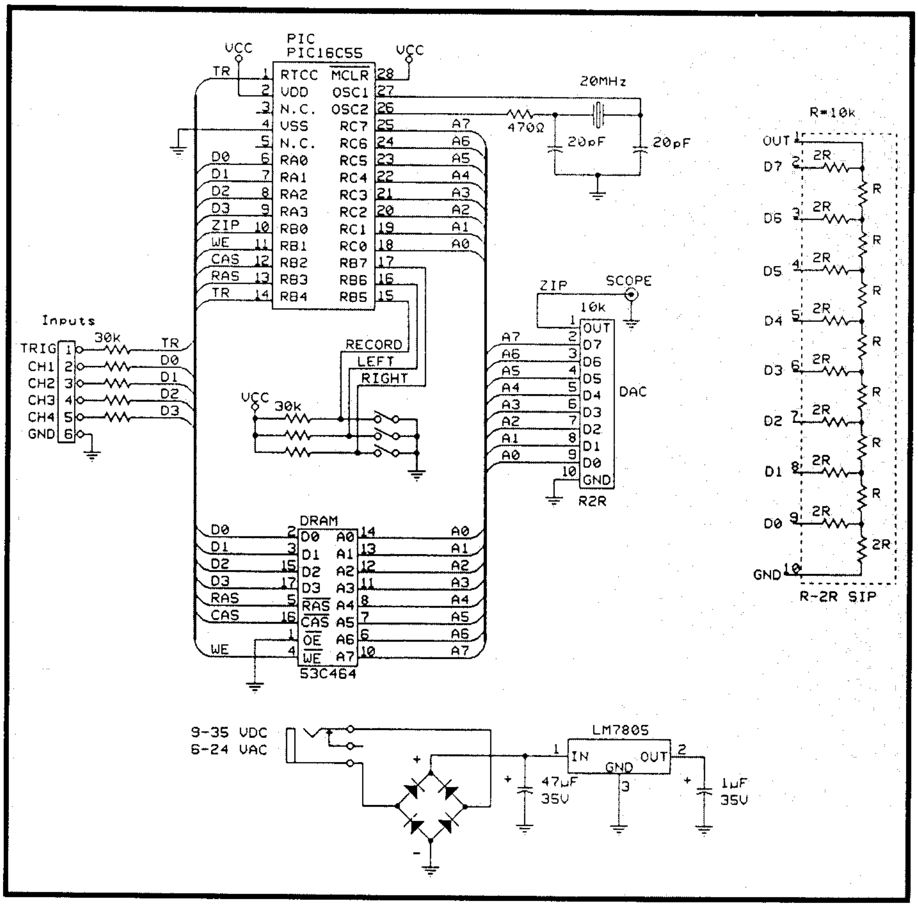


Figure 5—*A simple PIC application records up to four channels of input with time/trigger clocking. The output goes to a regular oscilloscope for display. A basic resistor ladder DAC keeps cost down.*

compatibility with vast libraries of existing official PIC code (I'll bet you don't either). Second, and most important, in my opinion the Parallax scheme is superior.

The Parallax improvements generally fall into two classes: more consistent instruction names and formats, and macroinstructions in which a single `PASM` instruction generates multiple official PIC instructions.

These improvements effectively decrease the already reduced instruction set. For example, Parallax turns ten different official instructions into one generic `MOV` instruction. Furthermore, the Parallax macroinstructions can replace up to four official instructions.

Remember, brain cells start dying when you're in your twenties, and the population of programmers is rapidly aging. The Parallax approach postpones the day when we'll be forced to watch commercials with elderly hackers whimpering, "I've fallen—and I can't remember all the darn instructions!"

CHEAP THRILLS

The "no-bull" approach calls for less talk and more action, so let me discuss an example PIC application.

Figure 5 shows the schematic for a lean, mean, four-channel digital oscilloscope based on the 28-pin 16C55-HS PIC, a 64K × 4 DRAM, and a resistor network configured as a voltage-dividing DAC. Thanks to the low price of the OTP PIC, the total parts cost for the gizmo is less than $20—refreshing isn't it?

When the Record button is pressed, the PIC captures and stores 64K samples from the four input channels in the 64K × 4 DRAM. For display, the PIC drives the resistor ladder DAC in a manner that causes an attached oscilloscope to magically show the four input traces plus a "guide" trace reflecting the position of the scope screen (64 samples) in the trace buffer (1024 screens). The DAC implementation exploits the fact that PIC I/O lines are true CMOS, unlike those of most TTL-compatible micros (i.e., high impedance when configured as inputs and sink/source rail-to-rail when configured as outputs). The Left and Right buttons scroll the scope "screen" back and forth either a sample at a time (position designated by the small guide) or 16 screens (i.e., 1024 samples) at a time (position designated by the large guide).

The display loop starts with a "sync" pulse generated by driving the DAC from 0 to 255 (i.e., 5 volts). Next, the PIC runs through a loop 64 times, outputting the guide trace voltage and then a voltage for each of the four data traces. The latter are determined by adding a voltage representing each sample value (0 or 1) and a position offset voltage corresponding to the channel number.

Three sampling modes are provided for Record. If Record is pressed by itself, the PIC simply captures 64K samples as fast as possible. If Record is pressed and held down, + or – are pressed and released, and then Record is released, the PIC waits for the specified edge on the Trigger input channel and then records 64K samples at full speed. Finally, if Record is pressed and held down, + or – are pressed and held down, and then Record is released followed by + or – release, the PIC records a single sample each time the specified edge is detected on the Trigger input.

A look at the Record portion of the code shows how the PIC determines the triggering mode and performs the data capture. Listing 1 also serves as a good introduction to PIC programming.

The Record routine starts by clearing the DRAM address and a bit designating full-speed or triggered sample mode. Next, the sequence starting at `:debounce` (the ":" makes `debounce` a local label; the same label name can be used elsewhere in the program) is a loop that waits for and debounces the Record switch input. This classical PIC code uses the `SETB` and `CLRB` instructions to set and clear bits and the `SB` and `SNB` to skip the next instruction depending on the

Listing 1— *The Record portion of the PIC code shows how buttons are debounced and how the samples are taken and stored.*

```
; Record pressed get final record command and read 64K
; samples into DRAM

handle-record call    clear-address      ;reset address
              clrb    bit                ;clr bit to flag left/right

:debounce     mov     temp,#100          ;100 cycles of debounce
:wait         sb      left               ;if left button, dir=0
              clrb    direction
              sb      right              ;if right button, dir=1
              setb    direction
              snb     left               ;if either button, set bit
              sb      right
              setb    bit
              jnb     record,:debounce   ;if record, reload debounce
              djnz    temp.:wait         :record not pressed, 100 yet?

              mov     w,#101000b         :get + or edge for option
              snb     direction          ; according to direction
              mov     w,#111000b
              mov     option,w           ;load option with w

              mov     w,#regular_loop    ;if left or right button is
              snb     left               ; still pressed, then do
              sb      right              ; per-sample triggering
              mov     w,#trigger_loop

write-entry   mov     temp,w             ;save loop address in temp
```

(continued)

state of a bit. It looks a little strange, but you can step through it yourself to see how it sets and clears bit [trigger mode/full-speed mode) and direction (rising or falling edge if trigger mode).

The PIC option register is written to configure the RTCC trigger input as rising or falling edge. Note again how the skip instruction makes the common chore of loading a register with a bit-dependent value easy. Similar code sets up a looping address (which is saved in **temp)** depending on the state of the + or – keys **(regular_looportrigger_loop).**

In trigger mode, the PIC spins on the three instructions at **trigger_loop** waiting for the LSB of the RTCC to change from 0 to 1 (i.e., when the RTCC increments in response to the appropriate edge on the trigger input). Meanwhile, the second instruction checks whether the Record key is pressed providing a way to cancel sampling should the trigger input fail to make an appearance.

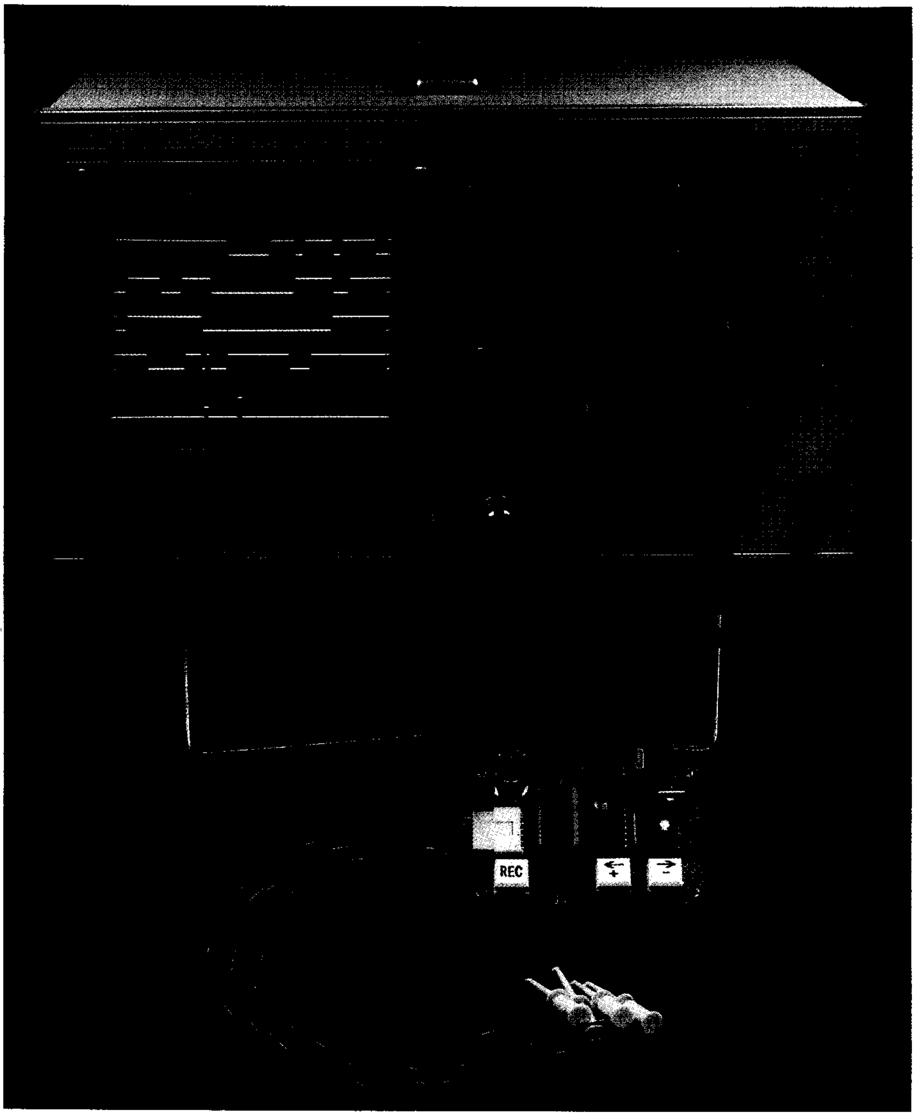
The core data capture routine at **regular_loop** does the DRAM

Listing 1—*continued*

```
              clrb   we                    ;clr we to enable dram write
              jnb    bit,regular_loop :if no button, regular

trigger-loop  clr    rtcc                  ;reset rtcc
:wait         jnb    record,clear_to_end ;if record button, abort
              jnb    rtcc.0,:wait          ;wait for trigger edge

regular_loop  mov    rc,address_low        ;2 cyc  ;write row address
              clrb   ras                   ;1      ;latch into dram
              mov    rc.address_high       ;2      ;write column addr
              clrb   cas                   ;1      ;latch in dram, write
              setb   cas                   ;1      ;end write cycle
              setb   ras                   ;1      ;return ras high, too
              mov    w.temp                ;1      ;get loop addr in w
              inc    address_low           ;1      :inc low address
              snz                          ;1/2s   ; skip if not 0
              incsz  address-high          ;1/2s   ;inc hi addr
                                                   ; skip if 0
                     w                     ;2      ;jmp to loop
                                                   ;(14 cycles/loop)
              setb   we                    ;done, return we high
              jmp    make_ra_input         ;compensate for write-entry
```



boogie by asserting the row address/RAS and the column address/CAS at which point whatever data is sitting on the four DRAM inputs is stored. The last steps increment the address and exit the loop when 64K samples are captured. Note how the final jump instruction loops back to either **trigger_loop** (for sample-per-trigger mode) or **regular_loop**, depending on the address in W (which is saved/restored from variable **temp).**

The core loop takes only 2.8 µs to execute (14 cycles x 4 clocks/cycle x 50 ns/clock) thanks to the fact the PIC is running at 20 MHz; thus, it acquires the data at over 350 kHz. Not bad at all. In fact, it acquires data much faster than any number of more expensive micros that shall remain nameless [lucky for them).

Oh well, the overpriced competitors have one big advantage over the PIC-at least they aren't cheap.

CONTACT

Parallax, Inc.
6200 Desimone Lane #69A
Citrus Heights, CA 95621
(916) 721-8217

Microchip Technology, Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
(602) 963-7373

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

Tom Cantrell holds a B.S. and an M.B.A. from UCLA. He owns and operates Microfuture Inc., and has been in Silicon Valley for ten years working on chip, board, and system design and marketing.

IRS

425 Very Useful
426 Moderately Useful
427 Not Useful

CEBus Goes Coax

Also Visit the Home of the Future

More and more of the CEBus specification is being accepted as an interim standard. Find out the latest on the CEBus front. Also check out another automated home on display.

DOMESTIC AUTOMATION

Ken Davidson

Anyone looking for my regular CEBus update in the form of a full-length feature article is going to be disappointed this time around. It's really nothing to be disappointed about, though. There just isn't as much news to tell you about as there's been in the past. That doesn't mean no work is being done on the CEBus standard; it simply means most of the work is done. Now the waiting begins.

The CEBus specification is being developed and released in stages. It has been broken up into sections including CAL and the upper network layers, each of the physical media, and a number of key pieces such as routers, brouters, and Node 0. [If anything I mention here is at all unclear, I refer you to my previous CEBus articles found in issues #10, #15, and #21 of Circuit Cellar *INK* for more details.)

Once the CEBus core committee has hammered out its best effort on any given piece, it is sent out to the entire membership (and anyone who requests and is willing to pay a copy) for comment. At the end of the comment period, which usually lasts a few months, all suggestions and negative comments are collected and responses are prepared by the committee. All negative comments must be addressed, even if it is to reject them. Any changes the core committee deems worthwhile are submitted to the membership for comment again to make sure nobody has a problem with what was supposedly fixed.

When everybody is happy with the proposed standard, the committee submits it to EIA as an interim standard. This standard is the first time the specifications are fixed enough for someone to design a product. It will remain in place for several years while people produce new designs. Assuming no major problems are discovered with it during that period, the specification finally becomes a real standard.

Right now, most portions of the spec are somewhere between the initial comment phase and acceptance as interim standards. Some have already been dubbed interim standards, while others are having last-minute details worked out before being released for initial comment. Table 1 gives a brief summary of where each of the major portions was as of this past December.

COAX COMES ALONG

One specification that is new since my last article is coax (CXBus). As you can see from the table, the comment period for CXBus ended December 31, so the committee should be up to its ears in comments as you read this.

CXBus is complicated. With the limited space I have back here, I'm going to necessarily simplify the details. My intention is to give you a flavor for how CXBus functions, perhaps only enough to allow you to envision how your house might be wired and how devices interact with each other. For complete specifications, I refer you to EIA where you may obtain a copy of the proposed spec.

CXBus uses a series of RG-6 coax cable pair branches that originate at Node 0. Each branch goes to a different area of the house and terminates in a four-way splitter. Each of the splitter outputs goes to a pair of jacks within the area serviced by the branch. Presumably, one branch would go to, say, the area composed of living room, dining room, and kitchen while another branch would go to, say, the bedrooms. Total cable length between Node 0 and any final tap may not exceed 150 feet.

One of the cables in each pair handles external video while the other handles internal video. External video is anything that originates outside the house such as CATV or an antenna. Internal video is anything generated in-house such as VCR output, cameras, or computer displays. The internal cable also carries the CEBus control channel.

Older devices that aren't CEBus compatible need only attach to the external cable to receive signals like they always have. CEBus-compatible devices must attach to the internal cable to reach the control channel, and may also connect to the external cable for access to outside sources.

Figure 1 shows the frequency assignment on each of the cables.

The CEBus control channel spans 4-6 MHz on the internal cable. The signal is an amplitude-modulated RF carrier. CEBus devices transmit control signals on 4.5 MHz and receive signals that are retransmitted by Node 0 on 5.5 MHz.

CXBus uses the same superior/inferior signaling scheme for the control channel as the other media. The presence of the carrier denotes a superior state while the absence denotes an inferior state. The four CEBus symbols are represented by one of the two states being asserted for multiples of the Unit Symbol Time (UST). A "one" bit lasts one UST, a "zero" bit lasts two USTs, an "end-of-frame" lasts three USTs, and an "end-of-packet" lasts four USTs. A single UST lasts 100 µs, giving a top speed of 10,000 "one" bits per second.

The frequency band from 54 MHz to 150 MHz on the internal cable is reserved for CEBus devices to transmit data. Within that space, 64 channels, each 1.5MHz wide, are defined. When a device needs more bandwidth, adjacent channels may be requested in groups of 2, 4, 8, 16, or 32 and are combined into a single block. The data is received by Node 0 and is retransmitted on either a low band on the external cable (324–420 MHz) or a high band on either the external or the internal cable (450-546 MHz).

That covers the basics. The complete spec is about forty pages long, so I certainly can't do it justice in only a few paragraphs.

IN THE PUBLIC EYE

If you've read my article in this issue about the Bright Home, you already know that I feel public education is a big part of what will make or break home automation's acceptance by the masses. I recently came across another example of exposing the public to what can be done now to automate a home with off-the-shelf devices.

Most anyone who watches PBS knows about "This Old House." A similar show that started appearing on PBS a few years ago is called "Home-

IRBus	Approved as part of interim standard in October, 1990.
TPBus	Approved as part of interim standard in December, 1990.
PLBus	Comments have been addressed. Several issues still need to be resolved.
CXBus	Initial comment period ended December 31, 1991.
RFBus	Several system alternatives are being reviewed by the committee. Final selection should be made by the time you read this.
CAL	Comments to changes are being addressed by the committee.
Router	Approved as part of interim standard.
Node **0**	To have been released for comment in December, 1991.

Table 1—*As* **of December,** 1991, *the* major *portions of the* **CEBus** *spec* **were** *in* **various** *states of acceptance.*

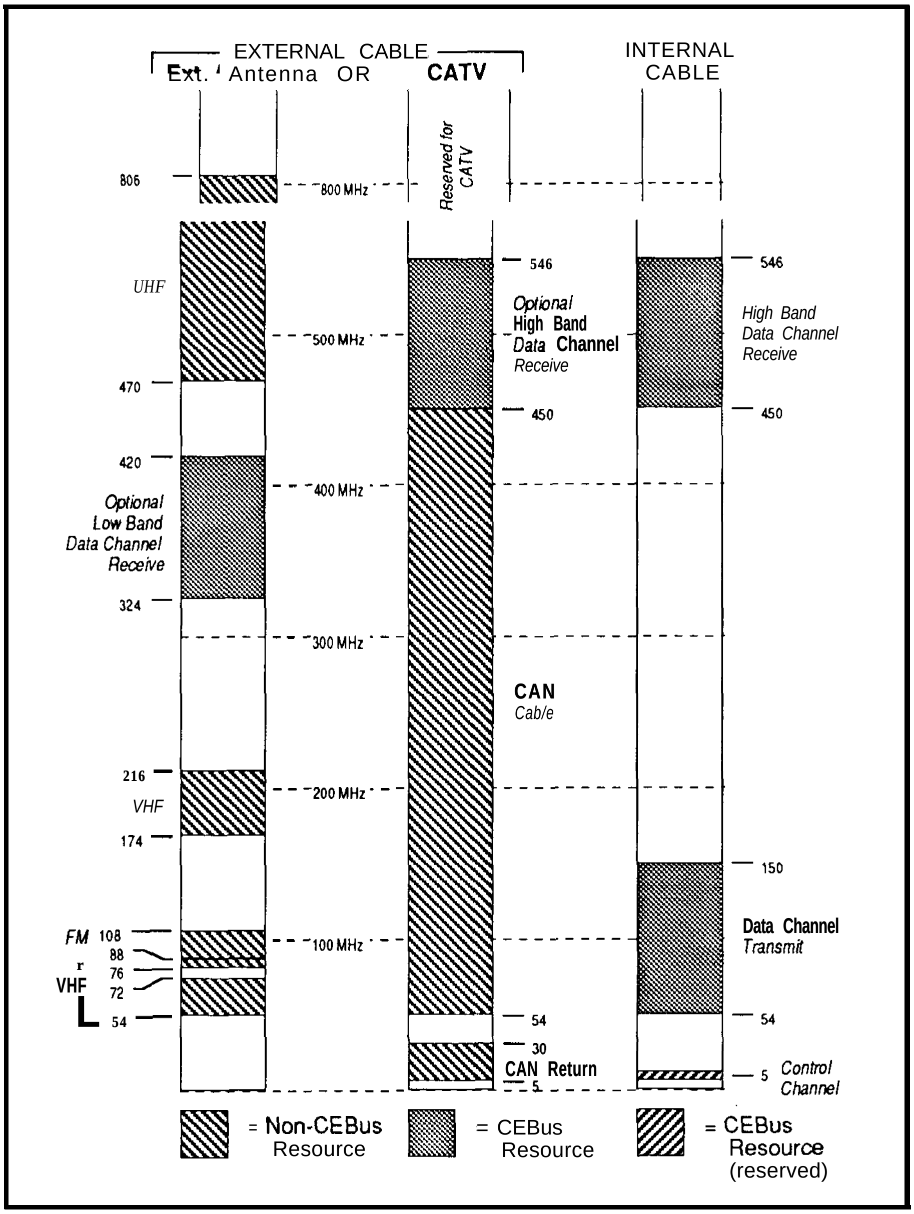


Figure 1—*CXBus* **uses hvo** *cables,* **one for external** *sources* **and one for** *internal sources*

time." This show is hosted by a cheery man and woman duo who make everything look like it can be done on your first try in a matter of minutes, even if it's erecting a two-story log cabin. The projects tackled by "Hometime" don't always center around renovating older houses. They often do all new construction, and sometimes they try to explore the unconventional. For example, they really did build the log cabin I mentioned above.

"Hometime's" latest project is the home of the future. More specifically, what a house might be like 20 or 30 years from now. The entire project is fascinating, even though the home automation system is only a small part of overall scheme. They cover everything from the architecture to the structure to the appliances to the landscaping.

The home automation scheme was designed by Mike Cogbill, who was instrumental in putting together the "The Installer's Guide to CEBus Home Automation" I discussed in the last issue, and Tricia Parks, who wrote the introduction to this issue's Building Automation feature section. The system they used is produced by AMX (Dallas, TX) and was originally designed to automate the boardrooms of large companies. AMX has been seeing a growing application of their system in high-end homes as a complete automation solution.

The home of the future is located in Minnesota and was part of a local "gallery of homes" tour where thousands of people toured the home during a several-day period.

The series aired in five half-hour segments starting last November, resulting in thousands more people being exposed to the ideas behind home automation. You might be lucky enough to find it locally in reruns, or you can buy a complete video from "Hometime" that contains all five segments plus additional material. At $9.99 plus $2.50 S&H, it can't be beat. Also keep an eye out for the April '92 issue of *Better Homes and Gardens* where there will be an extensive write-up about the home. ❑

Thanks to Tom Mock and George Hanover at EIA for their continued CEBus information flow, to Azt Chace of AMX for first alerting me to the Home of the Future series, and to Mike Cogbill for his very helpful last-minute details.

CONTACT

Electronic Industries Association
200 1 Pennsylvania Ave. N.W.
Washington, DC 20006
(202) 457-4975

(Video)
Hometime
Home of the Future
150 N. 6th St.
Philadelphia, PA 19 106
(800) 736-3033

(Comments about the home)
Hometime
Home of the Future
4275 Norex Dr.
Chaska, MN 553 18

I R S

428 Very Useful
429 Moderately Useful
430 Not Useful

CONNECTIME

conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400 bps, 24 hours/7 days a week
(203) 871-1988—Four incoming lines
Vernon, Connecticut

Not only has the magazine been given a new look, but with the installation of the new TBBS 2.2, the look of the Circuit Cellar BBS will also start to change and evolve. Nothing major; just a much-needed facelift. I'm hoping the changes will make the BBS easier to use for all callers.

Also like the magazine, while the look may be somewhat different, the content and quality of the messages posted by our users will remain as good as ever.

In this issue, we 're going to look primarily at stepper motors, with brief forays into using the IBM PC printer port bidirectionally and bow the magnetic stripes on bank and credit cards are formatted. We'll start with the magnetic stripes.

Msg#:44372
From: MATTHEW TAYLOR To: ALL USERS

I am wondering how info is stored on a magnetic stripe. I can fathom reading from and writing to a card if 1) a track is used that contains clock info or 2) a motor is used to drag the card across a head at a predetermined speed.

Do most cards use an encoding scheme that allows clocking info to be recovered, multiple tracks, a combination, or what?

Any help or info on companies dealing in this would be appreciated.

By the way, what sort of data densities are safe to lay down (i.e., can be recorded onto a stripe and withstand a year or so in a wallet)?

Msg#:44481
From: DAVID LAWSON To: MATTHEW TAYLOR

Credit cards have three tracks that can be used for data storage. Track one is recorded at 210 bits per inch with up to 79 characters. It is usually used by the airline industry and carries an account number (up to 19 characters), 2 to 26 characters for a person's name, with the balance of characters for discretionary data.

Track two is used by the American Banking Association and has your credit card information on it. It is written at 75 bits per inch with a maximum of 40 characters. It carries up to 19 digits of account number, expiration date (MMYY), with the remainder any data desired by the card issuer.

Track 3 is used by automatic teller machines. This track is written at 210 bits per inch and has up to 59 characters. Data on track 3 is primary account number, expiration dates, validity dates, service restrictions, and other data the ATM might use.

All tracks are written with phase-coherent FM recording and are synchronous for self-clocking data recovery. There are other flags embedded in the stripe's data like start sentinels, frame separators, end sentinels, and so forth. The actual characters use 5 bits and odd parity (if I remember the parity right).

Reading is fairly easy if you have one of the ICs designed for reading cards. Writing is quite another issue. The problem is getting the data written at a constant density. I have a writer that uses an optomechanical encoder to signal the on-board processor when to write the data.

Msg#:44518
From: MATTHEW TAYLOR To: DAVID LAWSON

Thanks for the wealth of info. I've been currently playing with a single-track card reader I've built. I also have an optical rotary encoder installed.

A very common question we get on the BBS is how to make an IBM PC printer port bidirectional. A hardware modification has been presented in the past, but it's often preferable to use an unmodified port, as is described here.

Msg#:44767
From: JOHN APPLEYARD To: ALL USERS

I'd like to utilize the parallel port on an IBM PC for bidirectional operation. Can anyone tell me the port numbers so I can access the port from BASIC (using INP and OUT). How about pinouts for the port? Thanks.

Msg#:44852
From: ED NISLEY To: JOHN APPLEYARD

Take a quick scan through the message base, as this topic comes up quite often. The DDT-51 project used a bidirectional parallel port and the code is available for downloading.. . admittedly in Pascal, but it's there.

CONNECTIME

There might be other files available that touch on this: have any of you regulars put together the definitive parallel port summary file yet?

Msg#:44857
From: JAKE MENDELSSOHN To: JOHN APPLEYARD

Problem #l-Finding the parallel port:

```
10 DEF SEG = &H40
20 PTR = PEEK(9)*256 + PEEK(8)
```

PTR is the output address as in:

```
30 OUT PTR.128
```

PTR+1 is the input address as in:

```
40 X = INP(PTR+1)
```

Problem #2—What are the output pins?

Pin:	2	3	4	5	6	7	8	9	18
Data Bit:	0	1	2	3	4	5	6	7	Gnd

These outputs are standard DB-25 parallel ports.

Problem #3—What are the input pins?

Pin	Name
10	ACK
11	Busy
12	Out of Paper
13	Select
15	Printer Error

There can be a lot of problems in using these inputs with different computers. These inputs are VERY nonstandard and there are a lot of variations from machine to machine.

For example:

1. Not all computers have all of these inputs connected to the DB-25 (e.g., pin 12: Out of Paper)
2. On some computers these pins float high and on other computers they float low. Don't assume they are one way or the other. Force them where you want with pull-up or pull-down resistors.
3. Here is how to read the bits of the input byte:

Bit	Value	Pin
0	1	Not Connected
1	2	Not Connected
2	4	Not Connected
3	8	15
4	16	13
5	32	12
6	64	10
7	128	11

Special Note: pins 15, 13, 12, and 10 are all noninverted, so their corresponding bits are set when the pins are high. Pin 11, however, is inverted, so its bit is cleared with the pin is high.

4. When you read the input byte with X=INP(PTR+1), you will get a number you can decode to determine which input pin is high and which is low. There are only five inputs and thus the first three bits are not connected to any pins. Be careful, on some computers these three bits are forced low and on others they are forced high.

Msg#:44947
From: BOB PADDOCK To: ED NISLEY

See if you can dig up "Interface Circuit for Printer Port" LAB- 13950, from *NASA Tech Brief:*

Purpose:

"This invention is an electronic circuit, which when used with appropriate software, converts the printer port on an IBM PC, XT, AT or compatible personal computer (PC) to a general-purpose 8-bit data, 16-bit address bus that is cable of interfacing to a multitude of devices."

Summary:

"The purpose of the printer port interface (PPI) is to convert any existing printer port on the IBM PC or compatible to a general-purpose bus. Since the printer port on the IBM PC was designed for the specialized task of outputting data to a printer, the PPI must be able to hold an address as well as provide for data flow in both directions.

In order to output to the external device, the PC printer port sends the address, data, and control values to the PPI one byte at a time. As the PPI receives this information, the PPI latches it on the proper output lines. Once the address and data are latched, the external device is signaled to accept the values.

In order for the PC to read from the external device, it first sends the address and control to the PPI. The external device then responds with eight bits of data on the input lines. Since the printer port has only five inputs, the PPI transfers the eight bits of data to the PC four bits at a time."

It uses several 8-bit latches and buffers. Request LAR-13950 from:

NASA STI Facility
Manager, TU Division
P.O. Box 8757
Baltimore, MD 21240-0757

CONNECTIME

Stepper motors are often used when precise speed and position control are necessary. To get maximum performance from them often takes some tricks, though.

Msg#:46462

From: JAY ALEXANDER ABEL To: ALL USERS

I am working on a stepper motor project, and I seem to be running out of torque. It appears I may be able to eke out what I need by driving the motor with sine waves in quadrature, but I lack a simple circuit for doing this. The same circuit is *usually* used for microstepping applications. If anyone has any experience with the above, or has a simple, elegant solution, please share it.

Msg#:46702

From: PELLERVO KASKINEN To: JAY ALEXANDER ABEL

The amount of torque you can develop is dependent on your motor and only in the second phase of evaluation on the driver circuits, because those circuits are considered selectable or adaptable.

As far as the motors go, you have several issues to consider. For the first, the frame size. For the next, number of poles and number of steps per shaft revolution. And for the third, the magnet material.

The number of poles is usually four, but there is a 5-pole design by Vexta. Anyway, the torque also is not the same in standstill as it is at a high speed. And then there are three resonances, where you tend to almost totally miss any torque at all, in most motors. The 5-pole motors cure two of the three resonances, if I remember correctly.

For any single motor, the torque is proportional to the winding current you apply, within two constraints: overheating and demagnetizing/nonlinear saturation. Normally only the overheating is a real design problem. But for most people, the problem seems to be lack of torque at the higher speeds, or inability to jump from standstill into those higher speeds or even reversal of rotation direction. The acceleration is normally taken care of by ramping the pulse frequency up gradually. You probably know of that already, so I'll move on.

The real high-speed torque is limited by the inability of any practical circuit to provide instantaneous stop of current in one winding and at the same time an instantaneous jump of current from zero to full nominal value in another winding. This is due to the winding inductances and also due to a transformer effect inside the stepper motor. What you can do depends on what you already have done. The sine wave drive as used for microstepping is great at minimizing the inductive effects and the transformer effects as well, if it is properly tuned for the particular motor you are using. But at the same time, it seldom provides for an ability to force a fast change of current for the commutation.

In full or half stepping, you would use two power supplies and current sensing. You feed the winding from the high-voltage supply after each commutation, until the nominal winding current has been achieved and then switch to the lower-voltage.supply for the rest of the step time. The voltage available from the initial high supply determines the maximum speed/torque you can achieve, but does not help if you are limited in the torque at low speeds as well. In such a case, my suggestion would be to get a beefier motor. Or provide artificial cooling to the poor small motor and then increase the supply voltage by whatever amount the added cooling warrants.

Msg#:47617

From: JAY ALEXANDER ABEL To: PELLERVO KASKINEN

Your insights were very useful to me, and in fact your recommendations parallel my own empirical conclusions. Actually, my original message was poorly worded, which I think obscured my question. I'll attempt to clarify.

I am building a magnetic reading device which steps the medium and samples the output from the head, so I not

CONNECTIME

only need resolution, but speed as well. I chose not to use a DC motor and encoder because the physical size and high cost were problematic. The stepper motor in question is very inexpensive. Howcvcr, I discovered for my application, I would need to step at 800 pps to achicvc my other design goals, clearly at the limit of stepper technology.

My solution was to increase the gear ratio so a rate of only 400 pps is required, and to make multiple samples per step. Hut for this to work I need to ensure there is minimal cogging. There are two basic methods to accomplish this: increase inertia or USC sinusoidal drive signals. The former would lengthen the acceleration time, which as Murphy's Law predicts, is also unacceptable.

I then searched for a simple method of generating sinusoidal drivc waveforms. I found no simple, elegant, hardware method. I did, however, find another method which I would like to share with you. By using a higher drive voltage, I am using the enable pins on the bridgc driver to pulse-width modulate the drive to the two coils. For each step, I take ten or so samples of input, and during each step I enable the two motor phases for longer or shorter periods, and, in effect, I get sinusoidal drive currents since the drive signal (4 kHz) is well beyond the mechanical and electrical self-resonant point of which you spoke. As I already had the sampling interrupt service routine, the cost in software is minimal, and the hardware cost is of course nil. Thank you for your timely comments.

Msg#:47710

From: PELLERVO **KASKINEN** To: JAY ALEXANDER ABEL

Looks like you have the most elegant way already, with the PWM at the enable pin. However, just a couple of points to amplify: There is always some kickback when you switch current into and out of a coil. Even more so in a motor, where you have the transformer action as well. The normal first thing people attempt is to put a direct free-wheeling diode over the coils. But that slows down the current change quite badly. What should be done is to have transistors that tolerate a high voltage and then have a suitable series resistor with each of the freewheeling diodes. If your coil current is 0.5 A at 12 V and your transistors can handle 60 V safely, you could allow the kicks to reach as high as 30 to 40 V by putting in a (30–12)/0.5 ohm resistor (36 ohms).

CONNECTIME

Another thing I want to mention is any card readers I know of use a DC motor. When people can make model airplanes containing such motors, I can hardly see it as a size issue. But that is just a shot in the dark, if you prefer the steppers, you do. I have, despite my experience with them, retained a preference of DC motors.

Msg#:48948
From: BRUCE GRAHAM To: PELLERVO KASKINEN

In a two-voltage scheme you can use the flyback voltage for shutting down a coil to charge a capacitor which sits on top of the high-voltage supply. This helps the high-speed performance by keeping the high-voltage supply from dropping at high speed.

Msg#:49099
From: PELLERVO KASKINEN To: BRUCE GRAHAM

Naturally, you are bound to try to use energy-minimizing techniques in the commutation. Trying to preserve the stored energy in a capacitor is an age-old way, with age-old side effects. One is that you have to have higher-voltage-rated transistors. Not a real problem today, but gave plenty of headaches to the early brave experimenters. Another one is that the **voltage level is strongly affected by the frequency of commutation. So, watch out!**

Stepper motors are sometimes "microstepped" to get more resolution from them. Just bow is such a feat accomplished?

Msg#:49301
From: MARC WARREN To: ED NISLEV

How does one microstep a stepper motor? Are there any old postings in the file area? Thanks!

Msg#:49328
From: ED NISLEV To: MARC WARREN

Well, the quick overview was in the Jul/Aug '88 issue of *Circuit Cellar INK* disguised as a sonar scanner using the venerable TI01 ultrasonic range finder.

Basically, if you're using digital drivers you're limited to "half-stepping" which means you turn on both drive phases to position the rotor half way between the normal stopping points. This effectively doubles the number of steps per revolution, but has the nasty side effect of doubling the power dissipation, which may not be a big win.

If you are using analog drivers, the extension is obvious. Power one phase at 25% of normal, the other at 75%, and you position the rotor at a "quarter step" position. The catches are that each phase driver is really a parallel port wired to a DAC instead of a single output bit and you need to run through an internal table of values that give you the right outputs for each position.

OK, you folks who do this for a living.. .now's your chance to chime in with the straight dope!

Msg#:49337
From: MARC WARREN To: ED NISLEY

I get the general idea. But instead of 25/75%, would a sine/cosine relationship be more accurate? [I just started learning about vector arithmetic during this project.) Also, what about using the 8253 as a PWM source for DACing the value? It works great on meter movements, provided the frequencies are high enough.

Msg#:49392
From: FRANK HENRIQUEZ To: MARC WARREN

I'm using an 8254 (bug-free version of the '53) to send pulses to a microstepping driver. I'm looking for a good, simple, constant-acceleration algorithm. It's easy if you're doing a big move, with an acceleration ramp, a constant velocity slew, and a deceleration ramp, but it gets pretty hairy when the moves are smaller than the distance covered by the acceleration ramp. Anyone have any ideas or source code? I'm working with a 68HC11, but any code for any processor would be helpful.

Msg#:49415
From: MARC WARREN To: FRANK HENRIQUEZ

I don't have a tested solution, but here's my thought. Assuming you know how many counts your total ramp takes, and it's the same in acceleration and deceleration, multiply your ramp by two and subtract the total move count. If the result is negative, your move is greater than the total ramp time, and just leave everything as-is. If the result is positive (ramp > move) divide the result by two and clip your ramps at that value. Actually, just the acceleration ramp-the deceleration should take care of itself. I'm making some assumptions here; this is just an off-the-cuff answer.

Msg#:49425
From: ED NISLEY To: FRANK HENRIQUEZ

The technique I've seen for that is you have to look at the total size of the move and use a constant rate instead of a ramp.

The whole purpose behind the acceleration ramp is to get the motor up to a step rate it can't reach from a dead start, then slow down on the other end. If you'll never have time to reach that rate, there's not much point in accelerating...so you use a rate it can handle and skip the frills.

So tell me: is the output of that microstepping driver linear or trigonometric between the motor's "big steps"—or is it something else entirely?

Msg#:49442
From: FRANK HENRIQUEZ To: ED NISLEY

Your suggestion was one of the things I tried. A lot of things work, but the code is inflexible or just plain ugly. I need to calculate values, since I'm using a 10-microstep/step driver. A table would be just as hard to generate. "My" 'scope positioning works out to 0.36" arc per step! Mechanical flexure and other sources of error will probably limit my pointing accuracy to about 1" arc. Since a star is a point source that gets defocused by the atmosphere, I'll consider myself very lucky if I get a night with star disks less than 1" in diameter.

Msg#:49433
From: KURT EHRHARDT To: ED NISLEY

If you can't make the Vmax at the top of the ramp, why not go up the ramp as far as you can and come back down again. That way you're 'ALWAYS' moving with velocities defined in the ramp table. Of course at such slow velocities the need for a ramp is a moot point.

Msg#:49443
From: FRANK HENRIQUEZ To: KURT EHRHARDT

For large motions, I'll be ramping up to a slew speed, slew at a constant velocity, then decelerate. The code to do this is pretty simple; it's the small motions that get ugly. I think the best solution for me is to always use a ramp; if the distance is too small to have a cv slew area, I'll end up with a triangle. At really low speeds [under 1000 steps/sec or so) I'll just skip the ramps and go at that rate. Ramps are useful; they reduce some of the grosser positioning errors and keep the stress on the motors and mechanisms down.

Msg#:49555
From: ED NISLEY To: KURT EHRHARDT

If you do that, the acceleration at the "top" of the peak will be beyond the spec; after all, you're going from maximum tolerable acceleration in one direction to maximum tolerable acceleration in the other, so the net change in acceleration (I think it's called "snap") will exceed the motor's ability to follow.

Of course, if the ramp isn't at the maximum spec then all bets are off. But the only reason to use a ramp is to limit the acceleration, so why limit it to less than the maximum value?

Now we get into picking the slope of the ramp depending on what the endpoints will be, which is how we got into this discussion in the first place!

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 (203) 871-1988. Set your modem for 8 data bits, 1 stop bit, no parity, and 300, 1200, or 2400 bps.

ARTICLE SOFTWARE

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

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

IRS

431 Very Useful 432 Moderately Useful 433 Not Useful

STEVE'S OWN INK

The More It Changes, the More It Stays the Same

Ordinarily, having an editorial page all to oneself would seem to be the perfect vehicle for espousing beliefs and prophesies. If I hadn't already bored you enough about my idea of the "appropriate" electronic house, for example, I could surely have the final word here.

Actually, I feel a little strange writing one page soliloquies which are supposed to be more significant than other pages. This is my page as "editor" but I sometimes wonder if it's just allocated to me as the "old timer" whose supposed to have been around long enough to be a soothsayer of sorts. In reality, all those years building projects in the cellar left little time for espousing beliefs in lieu of cold operable hardware. It was only after coming out of the cellar to start this magazine that performance seems to have generated its own cult following.

As *The* Computer Applications *Journal* enters its fifth year of publication, I can't help but feel we have accomplished something remarkable. We continue to grow despite this adverse economic climate and will easily pass 50,000 circulation this year.

When I talk to some readers, especially the "old timers" (an "old timer" is someone who actually used or owned an Altair, KIM-l, or similar first-generation personal computer), their greatest fear is that success will spoil this publication as it has many others (in their opinion). They point out the nonexistence of many previous technical magazines and the reorientation of others toward computer-specific business interests as examples of the trend.

Editorial reorientation is something you say to advertisers, but to readers it most often means abandoning those precepts which originally attracted you as subscribers and supporters in the first place. Another ploy is to reclassify the reader demographics to prospective advertisers. Readers who one day were avid "solder sniffing" technical types suddenly find that advertisers expect them to be "volume purchase influencers," ready and willing to buy anything.

I was asked if we'd change as we hit 10,000, again at 20,000, and so forth. I only hope the question is still being asked at 100,000.

The most meaningful response I can give is there is no reason to change. The three guiding rules of Circuit Cellar—make it real, make it work, and support it-are easy to follow because they make sense.

When the engineering team at *The* **Computer** *Applications Journal* presents a home control system, we are directly part of its design, installation, and evolution. We don't just describe technical events, we participate in them.

Perhaps because we are of one mind so often, job interest and execution overlap for most of us. For that reason we're still having fun publishing The *Computer Applications Journal.* Given that we have the luxury of enjoying what we do, it certainly doesn't make sense to change it.

Steve