CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

POWER CONTROL & CONVERSION

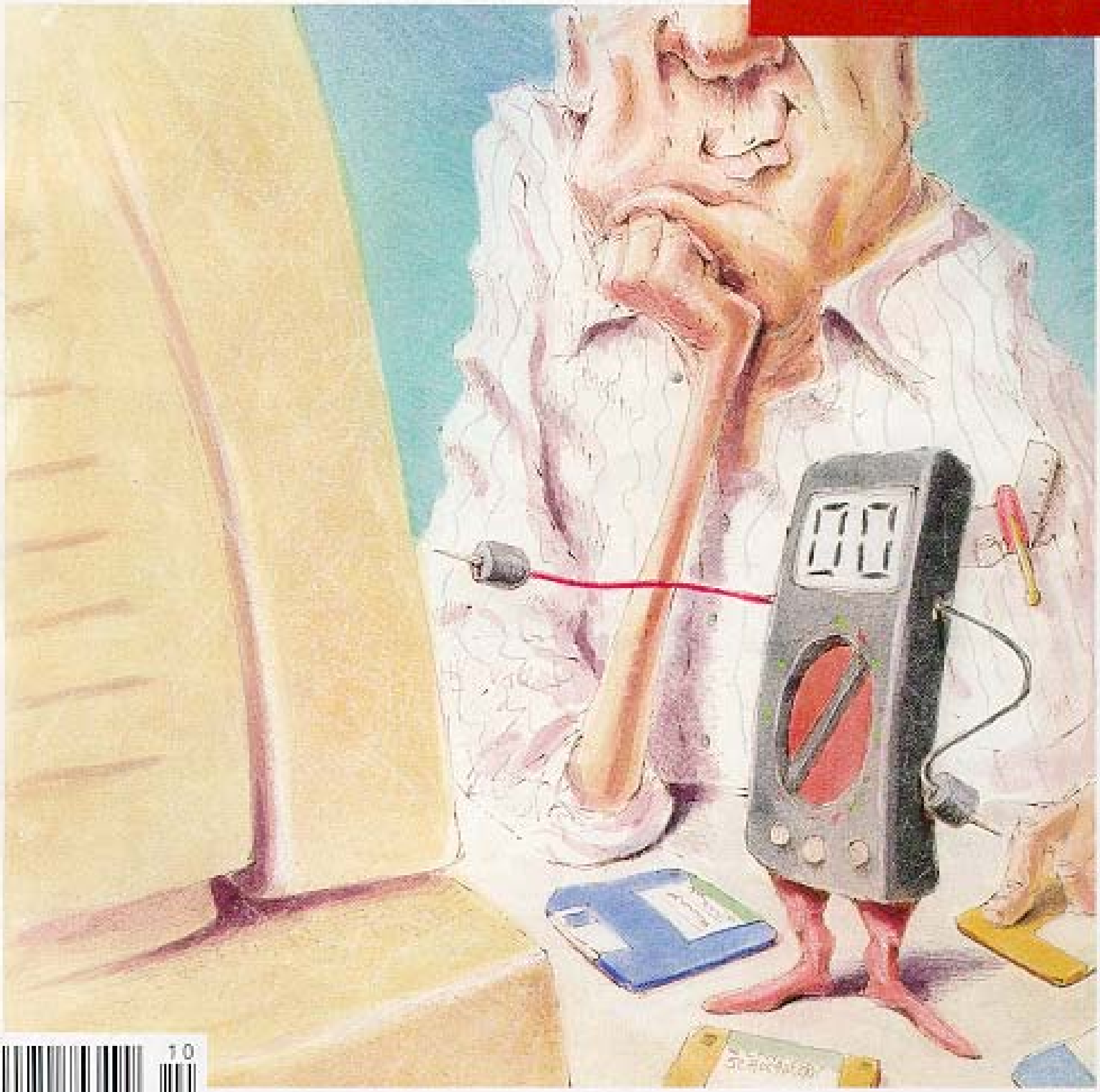
Computer-controlled Gas-fired Foundry Furnace

High-power Stepper Motor Control

Add DTMF and Voice to your HCS II

All-in-one Chips from WSI

October 1993 — Issue #39



10

0 74470 75349 0

$3.95 U.S.
$4.95 Canada

EDITOR'S INK

The Year Ahead

this time of year, most people's thoughts turn to the waning days of summer, back to school, and all-too-short vacations. Here at the *Computer Applications* Journal, though, we're already thinking about 1994. The production schedule is finished and we've chosen the editorial themes for the year. For the moment, let me ask that you remove your reader hat and replace it with your author hat. Take a look at the following list of themes and see if any matches your specialty or latest project:

Month	Issue #	Theme	Proposal Deadline
January	42	Home Automation	911193
February	43	Applications in Light	1011193
March	44	Cross-development Tools	11/1/93
April	45	Graphics and Video	1211193
May	46	Robotics	1/3/94
June	47	Distributed Control	211194
July	48	Communications	311194
August	49	Data Acquisition	411194
September	50	Embedded Applications	512194
October	51	Industrial Control	611194
November	52	Digital Signal Processing	711194
December	53	Computing in Real Time	8/1/94

As I've said before, you don't have to be a professional writer or even have written an article before to write for us. What's important is that you know your stuff and you can describe it in a manner that your fellow engineer can understand. Give me a call (203/875-2199), send a fax (2031 872-2204), write a letter (4 Park St., Vernon, CT 06066) or send me a BBS message (203/871-1988) with your ideas and we'll see where it leads.

In this month's feature articles, we have what I think is a neat project that graphically illustrates an example of real-world interfacing and control. The gas-fired foundry furnace controller collects information from an assortment of sensors and must make an intelligent decision about whether to allow the fire to burn. An error here could cause a bit more damage than a runaway lighting controller.

Next, we've had numerous articles in the past on controlling stepper motors, but now it's time to bring on the big motors. Find out what it takes to drive the motors that require more than a few milliwatts of power to operate.

Finally, Design Contest winner Derek Matsunaga describes how he reverse-engineered a test feature found on several models of Fluke multimeters so he could automatically send meter readings to a data collection computer.

I look forward to hearing from you with your article ideas.

Ken

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

TECHNICAL EDITOR
Michael Swartzendruber

ASSOCIATE EDITOR
Robert Rojas

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
John Dybowski & Russ Reiss

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
Lisa Ferry

GRAPHIC ARTIST
Joseph Quinlan

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

Cover Illustration by Bob Schuchman
PRINTED IN THE UNITED STATES

PUBLISHER
Daniel Rodrigues

PUBLISHER'S ASSISTANT
Susan McGill

CIRCULATION COORDINATOR
Rose Mansella

CIRCULATION ASSISTANT
Barbara Maleski

CIRCULATION CONSULTANT
Gregory Spitzfaden

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Dan Gorsky

CIRCUIT CELLAR INK. THE COMPUTER APPLICATIONS JOURNAL (ISSN 0896-8985) ıs published monthly by Circuit Cellar Incorporated. 4 Park Street, Suite 20, Vernon. CT 06066 (203) 875-2751. Second class postage paid at Vernon, CT and additional offices One-year (12 issues) subscription rate U S.A. and possessions $21.95, Canada/Mexico $31 95, all other countries $49 95. All subscription orders payable in U.S funds only, via international postal money order or check drawn on U.S bank Direct subscription orders and subscription related questions to The Computer Applications Journal Subscriptions, P.O. Box 7694, Riverton, NJ 06077 or call (609) 786-0409
POSTMASTER. Please send address changes to The Computer Applications Journal, Circulation Dept P 0. Box 7694, Riverton, NJ 08077.

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

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

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

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

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

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

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

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

Circuit Cellar INK makes no warranties and assumes no responsibility or liability of any kind for errors in these programs or schematics or for the consequences of any such errors. Furthermore, because of possible variation ın the quality and condition of materials and workmanship of reader-assembled projects. *Circuit* Cellar INK disclaims any responsiblity for the safe and proper function of reader-assembled projects based upon or from plans, descriptions, or information published ın Circuit Cellar *INK*

INSIDE ISSUE 38

READER'S INK

ONE COLLEGE STUDENT'S VIEW

I have been meaning to write to Steve ever since his editorial on the importance of properly educating our future engineers and innovators. Your offer of magazine subscriptions was most generous.

Who am I? I'm going to be a junior in the EE program at Cooper Union in New York City, but since I was in fifth grade, I have been interested in and learning about electronics. Steve's BYTE columns fueled my quest for knowledge and I read them religiously from May 1987 until the founding of *Circuit Cellar INK.*

Your BBS was perhaps my most valuable resource where real engineers helped a high school student find the bug in his first 6808 computer board (A13–A15 were wired backwards to the 'LS138). I did a good deal of work with 8-bit microcontrollers before entering college two years ago. Unfortunately, college took much of my time as did my job in the computer center, so I haven't done any building since high school. My main computer interests are now centered on parallel supercomputing. But I continue to have a soft spot for small 8-bit controllers. My new hobby, radio-controlled airplanes, provides for many fascinating control projects.

Why am I writing to you? It's the whole issue of American education in the technology arena-it's pitiful. What can I say? Much of what we're fed in college is inappropriate and is "taught" (to use the term loosely) by Ph.D.s who are out of touch with industry, creativity, students, and enthusiasm. This past summer, I worked for two months at Sun Microsystems in Mountain View, Calif., doing digital design work. I did this work with knowledge that I taught myself from your projects, from other engineers, from reading, and from experimentation.

Two complete years of engineering college provided me with near zero. A basic circuits course and an analog electronics course accounted for college's contribution to my usable skills. There is hardly any motivation through hands-on engineering in school. So many smart people are turned away from what I love because of what they are presented at school.

This has to change. Fewer theoretical formulas and more wire-wrapping. Less cramming and more soldering. A student should be shown a data book freshman year and learn about real components. Then, after a couple of years of fun learning, you introduce the heavy theory. It's a tough situation.

EE students at my school (and others) do not enter a lab class until the second semester of their junior year. Can you believe that? It takes 2% years to enter a lab. That's crazy!

Therefore, how can I deeply respect American colleges and their requirements? In Steve's recent editorial, he tells us to study and take advantage of our classes. I see little to take advantage of. The way to learn is on your own through books, magazines, BBSs, and experimentation. This must be encouraged by our society. Somehow.

I'm an academic "squeaker" as you put it. My overall GPA is 3.1 and my major is 3.4. I'll maintain those, but no more unless by luck. The way I see it, I can keep those averages with minimal impact on my life. That means plenty of time for my supercomputer research, for airplane building and flying, for my interests-my sanity. It is my belief that I will succeed through hard work on my own as I pursue my interests and gain valuable skills. We'll see.. .

Thanks so much for your columns, your magazine, your BBS, and Steve's personal answers to a 13-year-old's questions about TTL fanout. People like you are true educators.

Mark **Balch**
Brooklyn, N.Y.

After discussing your letter in the office among several engineers with backgrounds similar to yours, we can sympathize with your frustration. "Why don't they just get on with it and teach us something we'll use in the real world!" We've all been through it. Contrary to what you believe right now, though, those "inappropriate" courses that you think are a waste of time are actually laying the groundwork of knowledge that you will rely upon, consciously or unconsciously, for the rest of your life.

College is not for learning about and playing with the latest chips or processors currently on the market. If that's all you graduated with, you'd be worthless ten years down the road when your sum total knowledge is woefully out of *date.*

College is where you learn how to learn. It teaches you *the problem-solving techniques and gives you the basic skills that you'll need once you make it into the real world. Hands-on learning and experience are something you'll get in your first few years on the job. There just isn't enough manpower, resources, or time to include that kind of teaching in a four-year program.*

READER'S INK

And until you decide what you want to specialize in, it really isn't even appropriate.

You trusted Steve as a 13-year-old to answer your question from his many years of experience. Trust us again on this one. Go to class, do the work, and put your heart and soul into it. You won't have this opportunity again. Once you graduate, you'll have plenty of time for supercomputers and model airplanes. Now is the time to prepare for the future.

-Editors

SOFTWARE CONTROLLERS ON THE MAC

Regarding the letter from Fred Johnson in the July issue in which he asks about development tools for the Macintosh: HyperCard! It's the cat's meow. I'm not a programmer, yet was able to easily develop a very robust graphic robotic controller, UGRC@, using HyperCard and communicating via the built-in serial port. I even built a radio link to sever the cord. The real beauty is that HyperCard comes with the Mac.

If Fred has questions, I'm Sparky3 on America Online.

Gene Simmons
Harvest, Ala.

CORRECTION

On page 12 of the September issue, the correct phone number for Acuity Research Inc. is (408) 252-9639. We regret any inconvenience this error may have caused.

We want to hear from you!

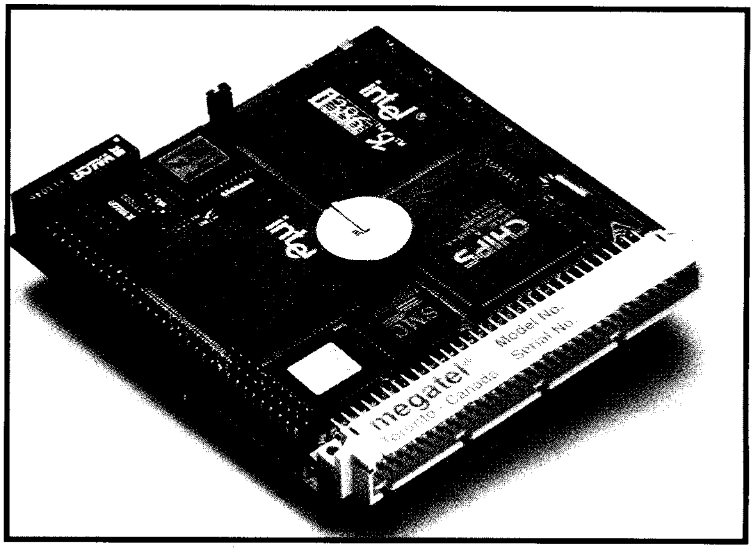
We encourage our readers to write letters of praise, condemnation, or suggestion to the editors of the Computer Applications Journal. Send your letters to:

The Computer Applications Journal
letters to the Editor
4 Park St. • Vernon, CT 06066

NEW PRODUCT NEWS

Edited by Harv Weiner

EMBEDDED PC FEATURES LOCAL BUS SUPER VGA



The embedded marketplace continually pressures PC manufacturers to decrease size while increasing functionality. Megatel has responded with the PC/II+, an i80386SL-based PC compatible with a multitude of features on a 4″×4″ PC/104 format board.

The PC/II+ includes a 25-MHz processor with up to 16M bytes of interleaved user DRAM, 256K bytes BIOS flash memory, SCSI host adapter, floppy disk controller, super VGA video/LCD controller, and an AT-compatible BIOS. Standard I/O features include two RS-232 serial ports, an RS-232 high-speed serial port, general-purpose parallel I/O port with BIOS support as a printer port, a real-time clock with battery backup, and a 16-bit ISA I/O bus. CMOS technology is extensively used to reduce power consumption to approximately 6 watts at +5 V.

The PC/II+ also includes an Ethernet interface using the Intel 82595 single-chip Ethernet controller and its buffered slave architecture. Megatel supports both 8- and 16-bit operations, both fully 802.3 compliant with the AUI and TPE serial interface. Software for the Ethernet interface includes full Novell IPX/SPX support.

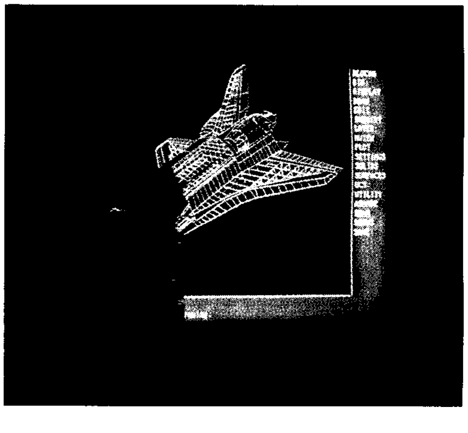
The performance of the on-board display controller has been increased by implementing a local bus, Chips & Technologies 65530 super VGA controller with a full 1M byte of video RAM. This allows the use of many of the higher resolution super VGA modes that are popular with multimedia applications.

Megatel provides a complete legal BIOS in flash memory. It will boot standard versions of PC-, MS-, or DR-DOS. An on-board ROM DOS is also offered. The PC/II+ will run most of the popular PC software packages including Windows 3.1. A high-performance flash-file subsystem, which provides on-board capability for a 2.25M-byte solid state disk, is also included.

The PC/II+ Developer's Kit sells for $995. with 2M bytes of memory. The kit includes a passive backplane, transition I/O board, cable set, board jacket, and user/technical manual set.

Megatel . 125 Wendell Ave.. Weston, Ontario . **M9N 3K9** Canada . (416) **245-2953** . Fax: (416) **245-6505** **#500**

PC VIDEO ON TV



A low-cost PC-compatible plug-in card and software that will allow the display of computer screen images on a TV or their storage on an SVHS VCR has been introduced by International Computers. The **VideoOut** is capable of displaying both VGA text and graphics on both black-and-white and color screens. VGA cards capable of interlaced video signals will be displayed on a TV set with perfect 480-line resolution. Noninterlaced video signals are also accommodated with excellent results. The device can produce flicker-free images from Windows.

VideoOut supports both text and graphics modes. Standard VGA modes are used by all video card manufacturers, but some cards use unique extended modes, with resolutions far beyond the original IBM definitions. VideoOut features a lookup table to determine the proper addressing and mode number for the VGA card installed.

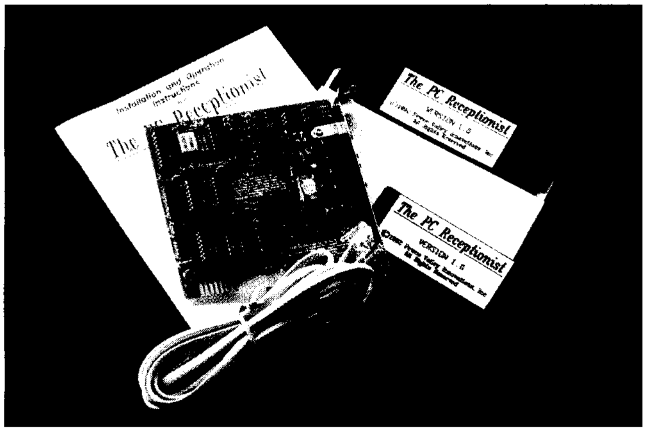
A user's manual and a video output cable to connect the unit to a TV or VCR are provided. The VideoOut includes the PC card, 2 video cables, user's manual and software diskette, and sells for $99.

International Computers . 12021 West Bluemound Rd.. Wauwatosa, WI 53226 • (414) 764-9000 . Fax: (414) 281-3522 **#501**

NEW PRODUCT NEWS

CALLER ID ACCESSORY

Pewee Valley Innovations has introduced a multipurpose, PC-based Caller ID accessory that offers convenient and foolproof solutions to telephone-related access and security problems. The **PC Receptionist** can be used to restrict access to modems and fax machines, track telephone usage, log customer and client calls, and make efficient use of pager facilities.


Installation and Operation Instructions
The PC Receptionist
The PC Receptionist VERSION 1.0
The PC Receptionist VERSION 1.0

Unlike many security systems that automatically answer a call and wait for the caller to enter an access code and/or password, the PC Receptionist uses Call Block/Pass technology to completely eliminate any chance of a hacker gaining access to the system. Unauthorized or unwanted calls are never answered and the system can be configured to completely block calls from predefined or unknown numbers, or to pass calls only from particular numbers, or both. The PC Receptionist is transparent to outgoing calls.

The PC Receptionist includes an easily configured eight-bit slot adapter card for IBM-compatible computers, DOS software, users manual, and a modular extension cord. For full operation, Caller ID service must be available from the local telephone company and subscribed to by the user. The PC Receptionist hardware is FCC Part 15 and Part 68 certified and is warranted for one year. The PC Receptionist Kit for DOS sells for $149.95. Windows-based software is $30 additional.

Pewee Valley Innovations, Inc.. 6601 Old Zaring Rd.
Crestwood, KY 40014 • (502) 241-4295 **#502**

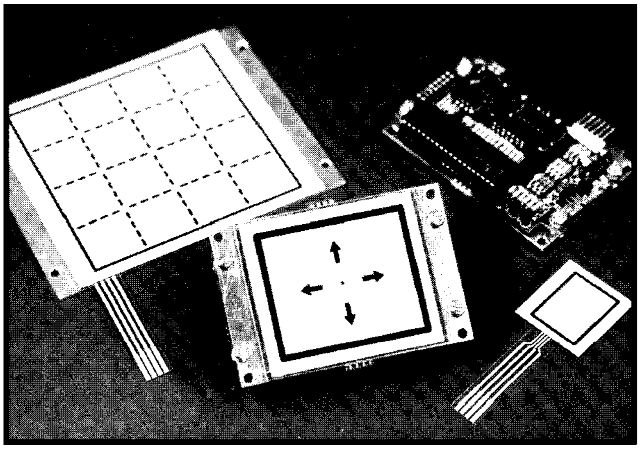
TOUCHSURFACE DEVELOPER KITS

The **TouchSurface** is a powerful, low-cost, input or control technology that senses both position and pressure continuously when touched with a finger or stylus. Intelligent Computer Music Systems has announced three developer kits for product design and experimentation.

The TouchSurface Demonstration Pointing Device is a desktop computer peripheral that provides the fastest evaluation of TouchSurface technology. It is suitable for developing product concepts and evaluating the TouchSurface as a pointing device or multimedia controller with capabilities well beyond mice and trackballs. It offers Microsoft two-button mouse compatibility in MS-DOS and Windows, and is fully user configurable in Windows.

The Panel Point is an embedded pointing device module designed for mounting in a panel or in an instrument chassis. It interfaces through a serial port and is ideal for industrial controls, mobile computer products, or other applications where control or pointing device functions are needed, but a mouse or trackball is inappropriate. Typical applications include character or graphical user interfaces, real-time systems controls, and absolute or relative positioning systems.

The Analog Interface Kit gives the product designer or experimenter a TouchSurface sensor with an analog voltage output. Applications for such a device include product development with embedded microcontrollers and systems that already incorporate analog-to-digital converters, or industrial or laboratory computers incorporating data acquisition systems.



The TouchSurface Demonstration Pointing Device and Panel Point Developer Kit sell for $250 each, and the Analog Interface Kit sells for $150.

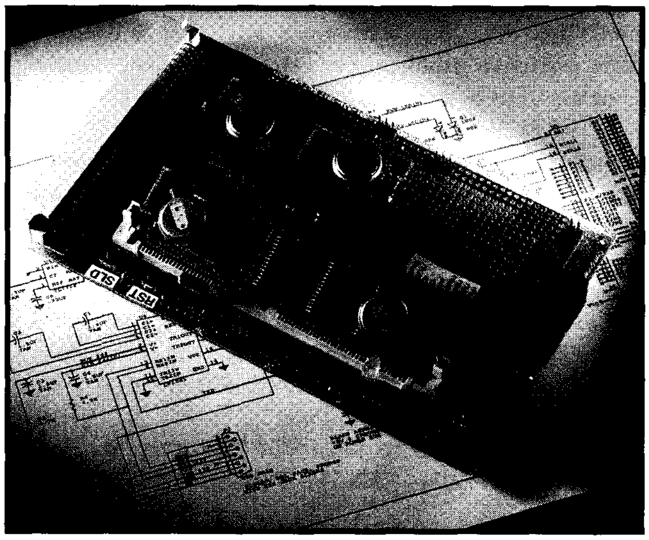
Intelligent Music.116 North Lake Ave.
Albany, NY 12206-2710 • (518)434-4110 . Fax: (518) 434-0308

#503

NEW PRODUCT NEWS

LOW-COST PROTOTYPE BOARD

A low-cost, easy-to-use prototype board and prototyping kit has been introduced by Intellix. The **DPB1** board allows 0.1" and 0.050" SIMM, DIP, and PLCC/PGA devices to be easily prototyped. The kit also supports Dallas Semiconductor SIPSTIK devices.



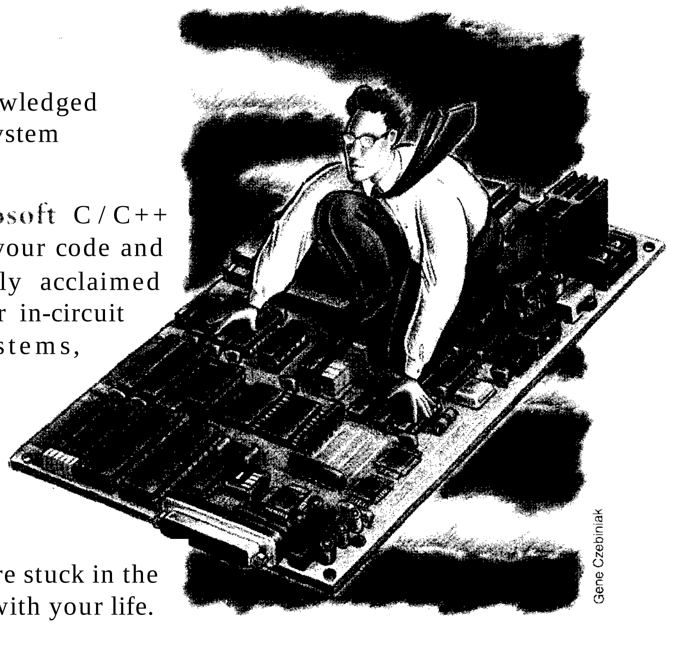
The DPB1 prototype board measures 100 mm x 160 mm (3.9"×6.3") and features three rows of SIMM patterns for up to 80-contact devices. It also provides 30 rows of holes on a 0.1" grid for any size DIP and up to 84 PLCC/PGA devices. Patterns for DB9 and RJ11connector, and various headers are also provided.

A novel feature of the prototype board is the staggered and straight hole patterns on 0.050" and 0.1" centers. This allows combinations of DIPs and SIMMs to be accommodated on the same board.

The DPB1 prototype board is also available as a complete kit that simplifies prototyping of Dallas microcontrollers, particularly the SIPSTIK packages. The kit includes an RS-232 link, reset chip, reset and serial load PLD, complete schematics, and all needed components. The bare board sells for $39, and kit pricing starts at $98.

Intellix • 555 South 300 East, **#21** • P.O. Box 526398
Salt Lake City, UT 84152-6398
(801) 534-1017 • Fax: (801) 534-1019

#504



NEW PRODUCT NEWS

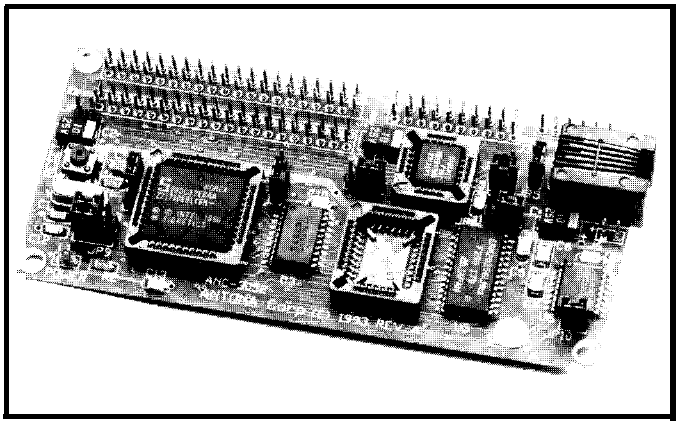
8032 EMBEDDED ADAPTER

The model ANC-3052 from Antona combines the best features of a prototyping adapter with the most commonly needed circuitry to support the Intel 8032 microcontroller. This Embedded Adapter approach provides the circuit designer with a low-cost, time-saving method to wire-wrap prototypes, or stand-alone systems, based on the 803 1 or 8032 microcontroller.

The adapter includes a crystal-controlled 80C32 processor, 32K of on-board RAM, a 32-pin socket (strappable for up to 64K of user supplied PROM), and an RS-232 level double-buffered serial interface which is baud rate programmable up to 38.4 kbps. There are two interrupt inputs, two counter/timer inputs, and eight digital I/O lines accessible through the combination wire-wrap and ribbon cable jacks. All microcontroller pins are accessible through both wire wrap and ribbon cable connectors. A 20-pin auxiliary connector provides seven decoded I/O strobes and the upper 8 bits of the address lines for external circuit control. A power-on LED and reset circuits are also included on the card.

The adapter occupies just under nine square inches of board space and provides wire-wrap and test points for each of the microcontroller pins. This allows the designer to attach oscilloscope or logic analyzer probes to the component side of the card. The adapter pins are on 0.1" centered rows to allow use on a variety of prototype boards or, by using ribbon cables, as a single-board computer. The RS-232 interface is accessed through either a 6-pin, single in-line connector or a modular RJ-11 connector. By using low-power surface-mount technology components, the card draws under 150 mA at 5 volts.

As an aid to the designer in using the adapter, the user's manual includes an adhesive-backed pin numbering sheet to guide the wire-wrapping of the prototype and a template of the adapter which the designer can use as a signal-to-pin designation map. The adapter is available with either 0.025-inch square, 3-level wire-wrap pins, or gold machine pins. The ANC-3052 sells for $146.



Antona Corp.
1643% **Westwood** Blvd.
West Los Angeles, CA
90024
(310) 473-8995
Fax: (310) **473-7112**

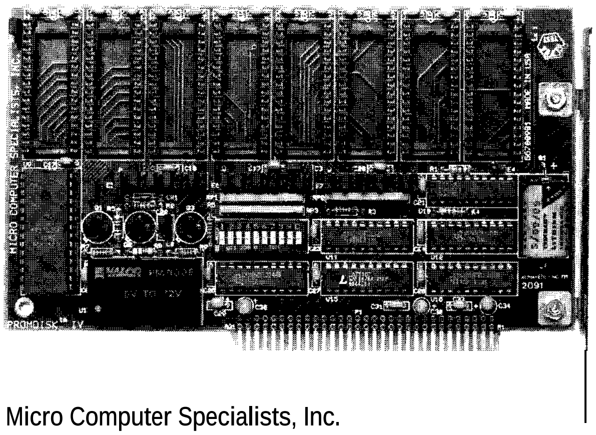
#505

SOLID-STATE DISK EMULATOR BOARD

MCSI has introduced a 4M-byte disk emulator board for IBM PC (ISA) compatible computers. The PROM-DISK IV can emulate up to three read-only or read/write fixed or floppy disk drives with capacities ranging in size from 32K bytes to 4M bytes, using 28- or 32-pin JEDEC standard EPROMs, flash memory, and SRAMs. A program that is developed using MS-DOS, DR-DOS, or QNX operating systems will operate without a physical disk in a target application system.

PROMDISK IV can typically be configured as one 4M-byte drive; two 2M-byte drives; or one 2M-byte drive, one 1.5M-byte drive, and one 5 12K-byte drive. In addition, PROMDISK IV will work with other physical drives in the system.

The PROMDISK IV Disk Emulator Board comes with the on-board control firmware, users manual, and utility disk and sells for $299.

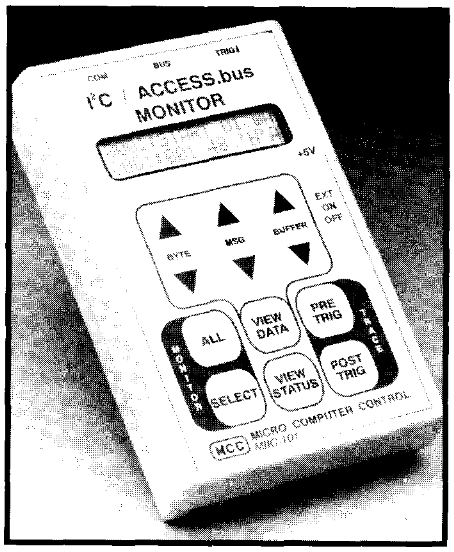


Micro Computer Specialists, Inc.
2598-G Fortune **Way** • Vista, CA 92083
(619) 598-2177 . Fax: (619) 598-2450

#506

NEW PRODUCT NEWS

MONITOR FOR I^{2}C OR ACCESSBUS



Developers working with I^{2}C or ACCESS.bus systems can now collect bus traffic in real time with a new instrument available from Micro Computer Control Corporation. The **101 Bus Monitor** is a stand-alone troubleshooting tool for the InterIntegrated Circuit (I^{2}C) serial bus developed by Philips Semiconductors, and the ACCESS.bus developed by the ACCESS.bus Industry Group (ABIG).

I^{2}C is a low-cost network for connecting microcontrollers and a variety of integrated circuits. ACCESS.bus is an open industry standard for connecting multiple input/output devices to a PC via a single port. When connected to an I^{2}C bus or ACCESS.bus network, the 101 Bus Monitor can collect, display, or upload information on all bus activity. The battery, wall transformer, or bus powered unit includes a display and keypad for stand-alone operation, and a built-in RS-232 interface.

In Trace mode, the unit can collect bus messages directed to one or all bus-device addresses. Data collection can start or stop with an external trigger input to synchronize the unit with network events. Internal storage can buffer up to 2700 messages. Remote operation can upload messages to a PC while collecting network traffic.

Applications include software and hardware troubleshooting, manufacturing and quality control testing, and field service diagnostics. The 101 Bus Monitor sells for $667.

Micro Computer Control Corp. • P.O. Box 275 • Hopewell, NJ 08525 • (609) 466-1751 • Fax: (609) 466-4116 #507

NEW PRODUCT NEWS

INTELLIGENT DVM

DeltaQuest has introduced The Intelligent DVM, a digital voltmeter that interfaces to a standard IBM PC compatible. The unit consists of a module that connects to a standard printer port and an innovative software program that allows the user to rescale or convert measurements automatically. The DVM is ideal for remote applications with a portable computer.

The Intelligent DVM possesses powerful recording capabilities. Initiation, sampling, and termination are controlled by various combinations of time, trigger, count, and displayed value. All significant parameters may be recorded. The software provides powerful graphing options for recorded and real-time data. Graphs may be cut and pasted into other Windows applications, or sent directly to the printer. In addition, the software allows instant conversion from volts to decibels, degrees Celcius, PSI, or any other user-definable scale.

Two digital outputs allow the unit to control peripheral equipment, enabling it to operate as a dedicated process server. DVM functions are controlled externally by two digital inputs, an on-screen push button, and the DVM displayed value. Resolution of 5% digits allows for simple interface to low-level signal sources such as thermocouples and pressure and strain gauges.

The Intelligent DVM autoranges from t1.2 V to ±120 V and features an input impedance of 10 megohm (>100 megohm for the 1.2-V range), and a conversion rate greater than 10 conversions/second. The 3.75" x6.3" xl .41″ unit requires 9 VDC at 5 mA.

System requirements are a PC compatible with Microsoft Windows 3.1 and a free parallel port. The Intelligent DVM is available in three models: The Professional Model at $329.95, the Basic Model at $289.95, and the Single Range Model at $229.95.

DeltaQuest • 4960 **Almaden** Expressway , Suite 238 . San Jose, CA 95118 . (408) 997-8644 • fax: (408) 997-6730 **#508**

#108

#109 See us at *Embedded Systems Booth #1005*

FEAT'URES

Fire Control for Foundry Furnaces

FEATURE ARTICLE

David McFalls

David unleashes the powers of Apollo to tame the raging maelstrom of Prometheus and creates a link between the bronze age and the age of silicon.

I'm a mechanical engineer who has been trying to build a working foundry for a long time. My first experience containing high temperatures came sometime in intermediate school when I discovered I could power a carbon arc from the wall plug using a salt-water rheostat. The body of the furnace was an old ceramic flower pot with electrodes removed from D-cell carbon zinc batteries. I've built a variety of different furnaces and ovens since then, each teaching new lessons beyond finding the shortest route to the circuit breaker. To generate heat on a large scale, air-fuel combustion is wonderfully cost effective, but potentially dangerous. I hope to illuminate many of the safety issues of combustion monitoring and control for those of you who might enjoy experimenting with yet another computer peripheral device. Please be advised that situations dangerous to life and property are possible (probable) if you experiment with combustion, and that the system described in this article cannot guarantee your safety.

OVERVIEW

The oven hosting the combustion system has a working volume of 45 ft^3. It is a steel-shelled behemoth weighing a couple of tons. It is used to prepare ceramic shell molds for foundry work, and is designed to fire pottery as well. The oven is fed by two burners, each with its own flame monitoring system, and its own ignition'system. The air-fuel ratios are manually adjustable at the burners, with source air and fuel under combustion system control. The oven uses forced-draft so that the working volume is at a slight positive pressure. Complicating burner issues, unfortunately, the oven and the combustion systems are outdoors, which stresses many components with exposure to dust and rain. Additionally, despite expensive insulation and careful design, the oven provides the potential for serious radiant and/or conductive heat transfer to several system elements.

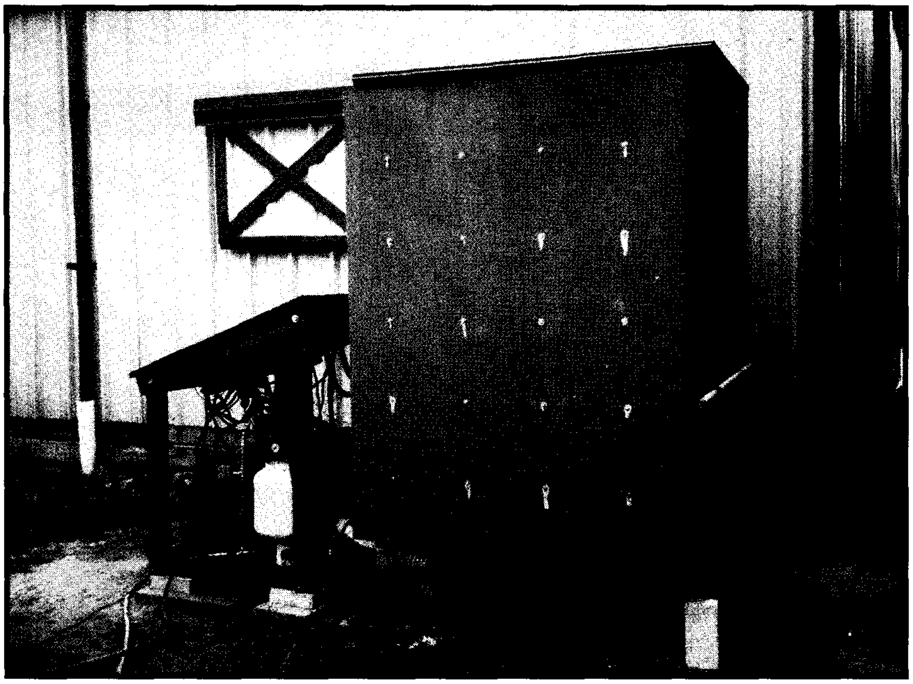


Photo I-The *combustion control system (left) is located next to the oven (right). Each subsystem is contained in its own aluminum box under the top of the control panel.*

The combustion control system should provide simple, reliable, and safe control of the oven in operation. In my case, these goals are somewhat constrained by concerns for time available, cost, and ease of use. Combustion control is normally defined to include flame monitoring, ignition, and control of fuel and oxidizer mass flow rates to achieve desired heating effects. Real-time control of the air-fuel ratio by monitoring exhaust gases for 0,, CO, or CO, and continuous smooth throttling of the burners are features I couldn't afford for this oven. I think the simpler system described in this article will suit the needs of most experimenters. This system monitors the burner and ignites it when necessary. A PC controls the temperature of the oven by switching the burners on and off.

A high-level diagram of this system is in Figure 1. The PC performs the management chore of deciding when to fire the burners. A program called MOLTEN interprets oven command files, communicates with the thermocouple interface module, and logs temperatures. Most other systems are configured into two channels, one for each burner; there are two flame sensors, two ignition systems, and a

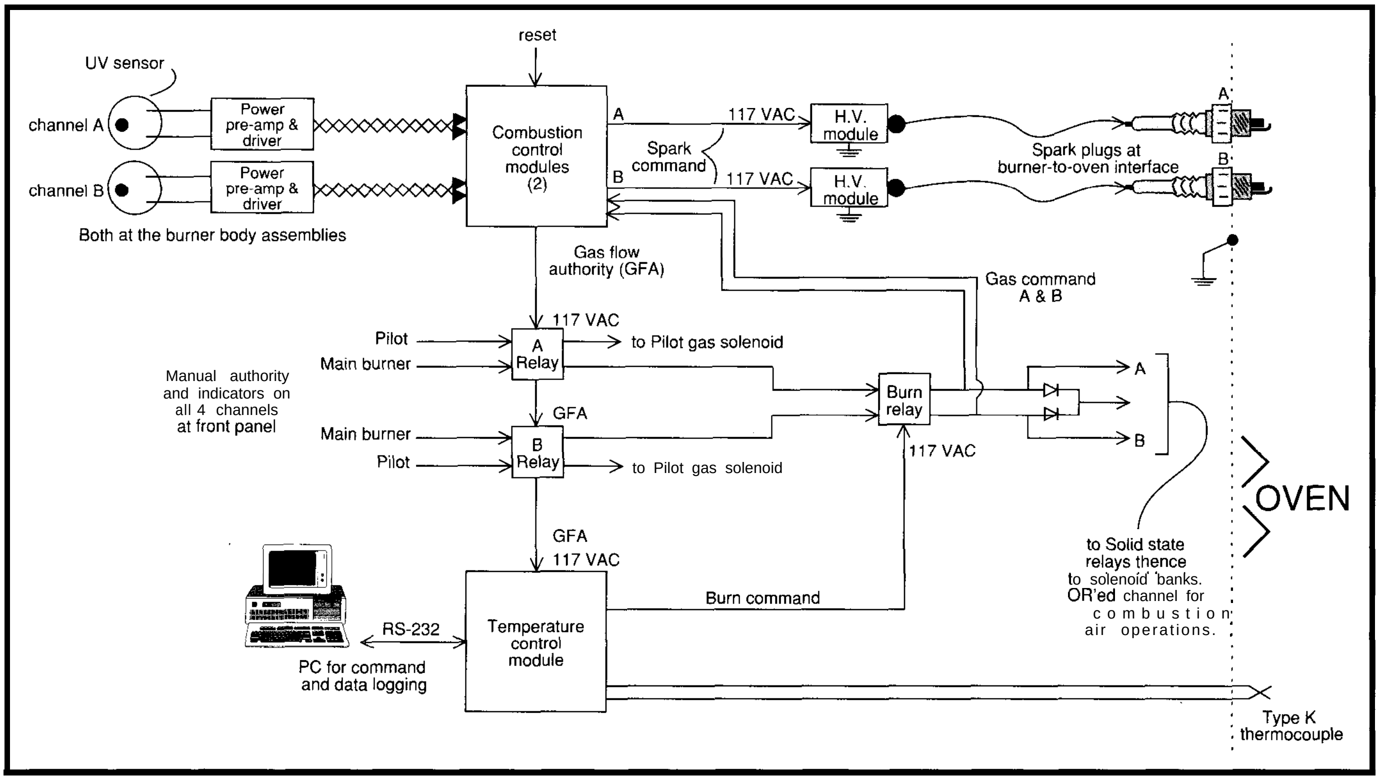


Figure *I-An overview of the electrical systems for fhe oven controller. There have been several revisions over the years, creating a mixture of bought, built, and found components.*

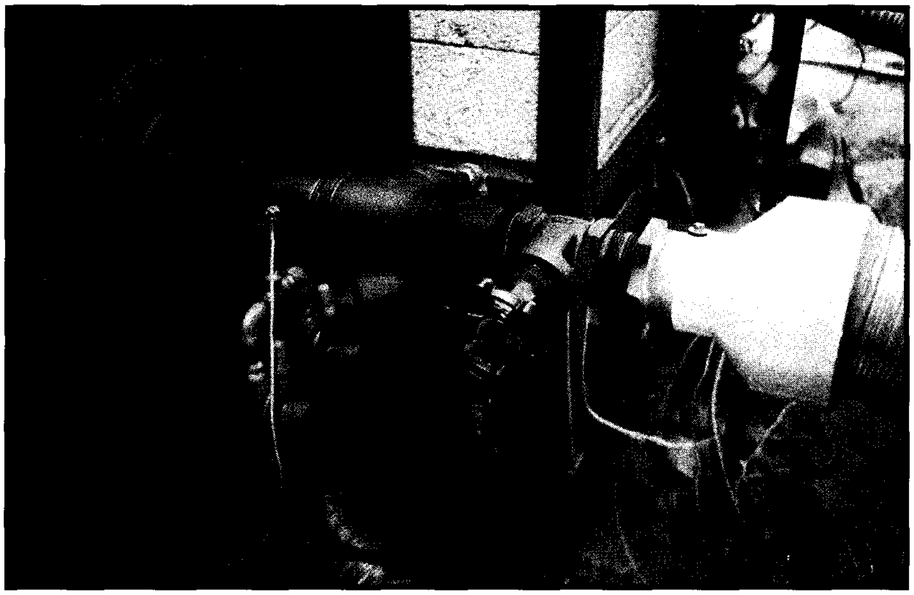


Photo 2—*The air intake, UVtron, fuel supply line, and spark* plug *assembly. See Figure* 3 *for a labeled diagram of the components shown here.*

variety of parallel command paths controlling gas and air flow. Typically, where dual channels exist, they are known as A (red) and B (blue). There are also two pilot channels through the control relays. The pilots were used before the completion of the flame control system to light the main burners, but have since been used to preheat molds.

There are a variety of safety features incorporated into the overall design. All gas flow solenoids are normally closed in case of a power outage. There are also manual gas shut-off valves at a variety of choke points along the gas feed route. For the main burners to have gas flow, gas flow must be authorized from the front of the control panel, the combustion control system must authorize gas flow, and the PC must request the burners to be lit. Any of these three controlling agents may deny the burn. If you have enabled gas flow at the front panel and the PC has requested heat, the flame monitoring circuitry allows five seconds for a good light in both burners before removing gas flow authority. This may seem lengthy, but when the oven is cold, lean air-fuel mixtures are difficult to keep in steady combustion in the burners. The ignition time-out period should be set based on the circumstances encountered in a particular installation.

From a packaging standpoint, most of the electronics fit into three small aluminum boxes mounted to the underside of the control panel. Cables make long runs from the panel to the flame sensors, the ignitor modules, the thermocouple, and the PC.

MONITORING COMBUSTION IN THE BURNERS

Combustion detection techniques range from simple mechanical means to sensing flame emissions in the electromagnetic spectrum. When burners are sealed into a hot chamber, techniques such as mechanical or gas-based thermostats, thermocouples, and temperature sensing IR pyrometers can't readily be used. That leaves detecting combustion by monitoring changes in conductivity, emissions in and around the visible spectrum, and acoustic signature.

One method of conductivity monitoring is the flame rod, which senses current flowing in the ionized species of the flame. This means the rod must be in the flame. Potential maintenance problems include fouling or drooping of the electrode and leakage currents affecting the flame sensing decision. The flame rod is

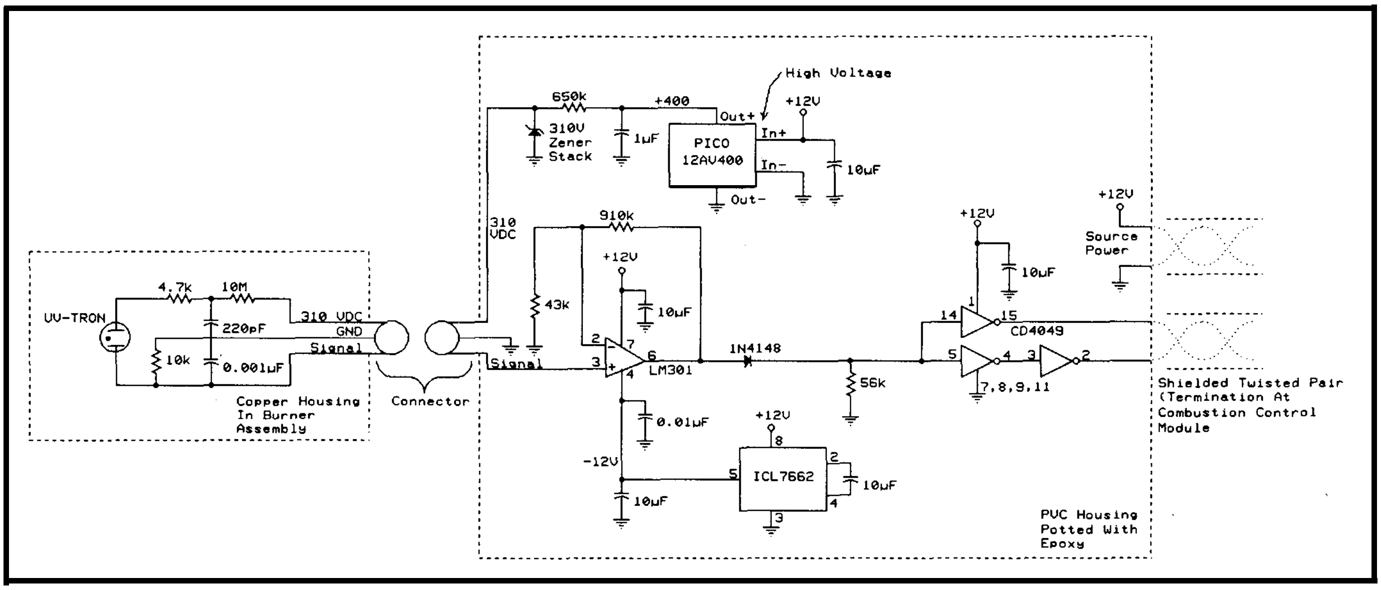


Figure 2—*Heat, size, and weathering were the primary concerns for the UV defection components at the burners. The UVtron support electronics fit info small PVC plumbing fixtures which attach to the base of the phenolic housing mounted in the burner.*

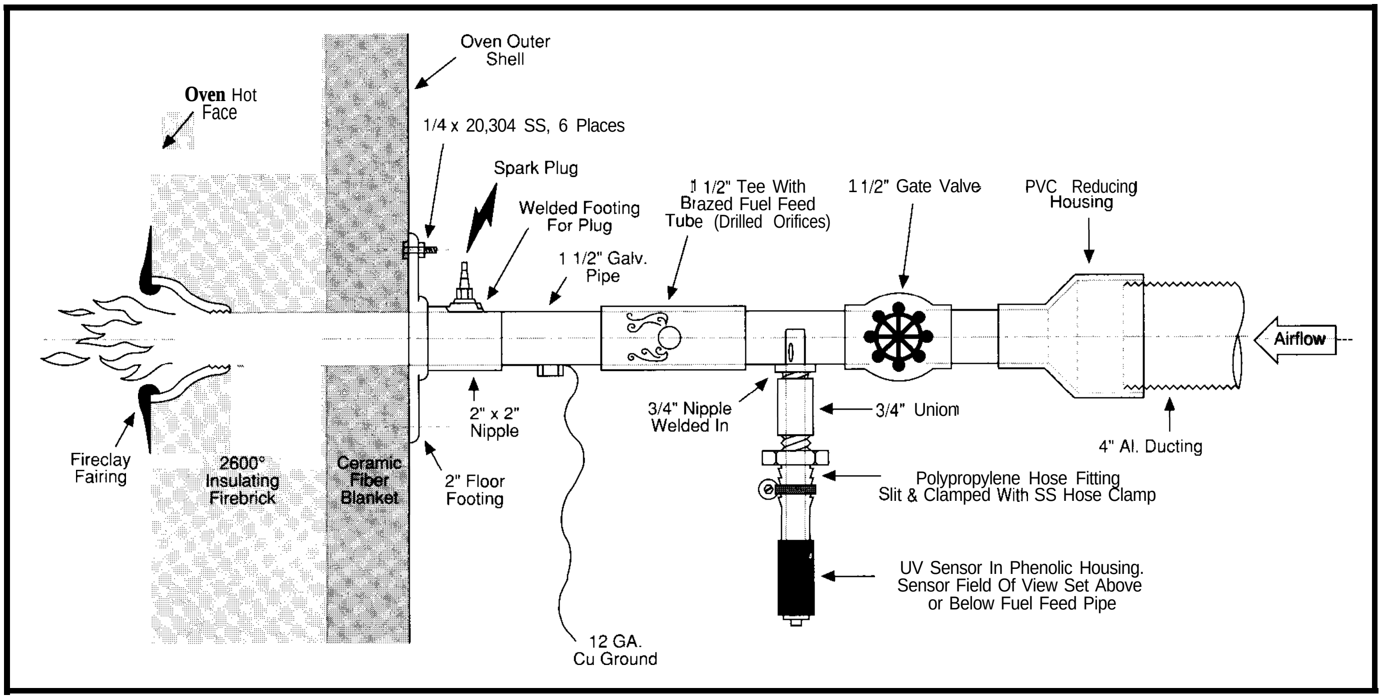

Oven Outer Shell
Oven Hot Face
1/4 x 20,304 SS, 6 Places
Spark Plug
Welded Footing For Plug
1 1/2" Galv. Pipe
1 1/2" Tee With Brazed Fuel Feed Tube (Drilled Orifices)
1 1/2" Gate Valve
PVC Reducing Housing
Airflow
3/4" Nipple Welded In
3/4" Union
4" Al. Ducting
2" x 2" Nipple
2" Floor Footing
Fireclay Fairing
2600° Insulating Firebrick
Ceramic Fiber Blanket
Polypropylene Hose Fitting Slit & Clamped With SS Hose Clamp
UV Sensor In Phenolic Housing. Sensor Field Of View Set Above or Below Fuel Feed Pipe
12 GA. Cu Ground

Figure 3—*The UV detector assembly slips info the airstream just ahead of the propane feed. Alignment is performed by adjusting the detector assembly while looking down fhe burner bore from inside the oven.*

simple technology, but I prefer to put system components outside the raging inferno for longevity.

While acoustic monitoring seems an intriguing possibility, I decided to build a system based on technology the industrial sector uses everyday. For noncontact combustion monitoring, systems typically use infrared emissions and/or ultraviolet emissions. For hot chambers, there are IR systems that look at the AC component (less than 1 kHz) of the radiation to detect the chaotic flame front. These systems are sometimes used while burning materials which are strongly ultraviolet absorbing, such as hydrogen sulfide. For simplicity, it is hard to beat monitoring the ultraviolet spectral emissions of combustion.

You can go out and buy "purple peepers" and hook them to flame managers for a couple of kilobucks, but where's the fun in that? Solar blind UV detectors are relatively cheap at $35, and are easy to work with. While there are some solid-state detectors out there, the large installed base of products surrounds the gas discharge tube. The careful choice of spectral response of the tubes makes them sensitive to UV flame emissions, but unable to detect solar illumination, most domestic light sources, or the black body type emissions of a very hot oven. I chose the Hamamatsu R2868 tube for this project, a tube they describe as a *UVtron*. It is sensitive from 185 to 260 nm and is small enough to get into the tight spaces of my burner assembly.

In use, the tube has a 300350-volt potential across it. The photoelectric effect causes the cathode to emit electrons when the incident photons have enough energy to excite them off the surface. After the electrons are emitted from the cathode, they accelerate in the electric field toward the anode. They excite some of the low pressure gas molecules they find along the way, creating ions and more electrons. The ionized pathway in the tube created by this avalanche conducts nicely, with the current flow signifying that UV radiation has been

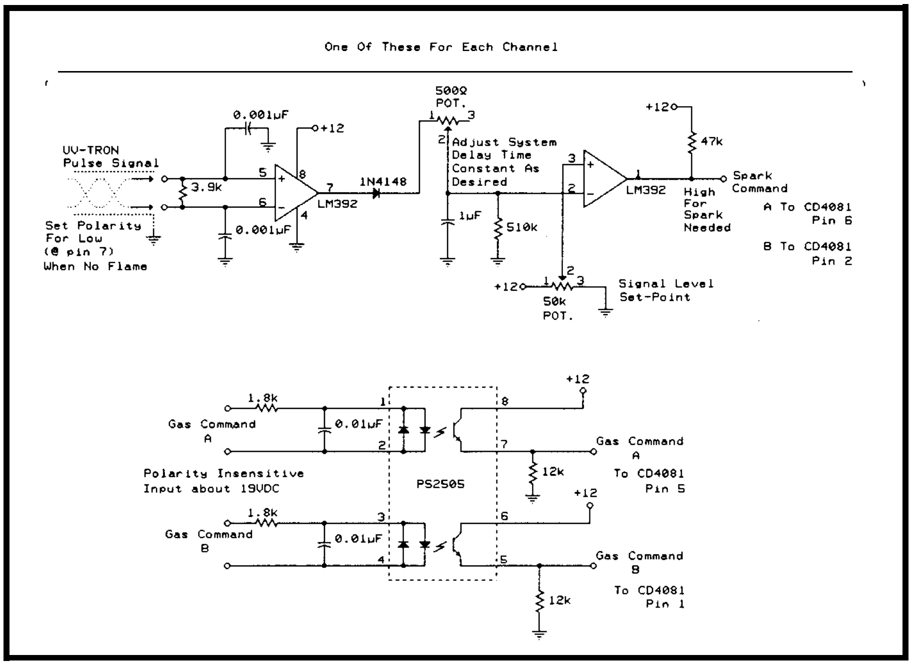


Figure 4-*The LM392 performs nicely as a differential amp and comparator for the incoming pulse train from fhe UVtron.Incoming signals subject to solenoid spikes were isolated as necessary.*

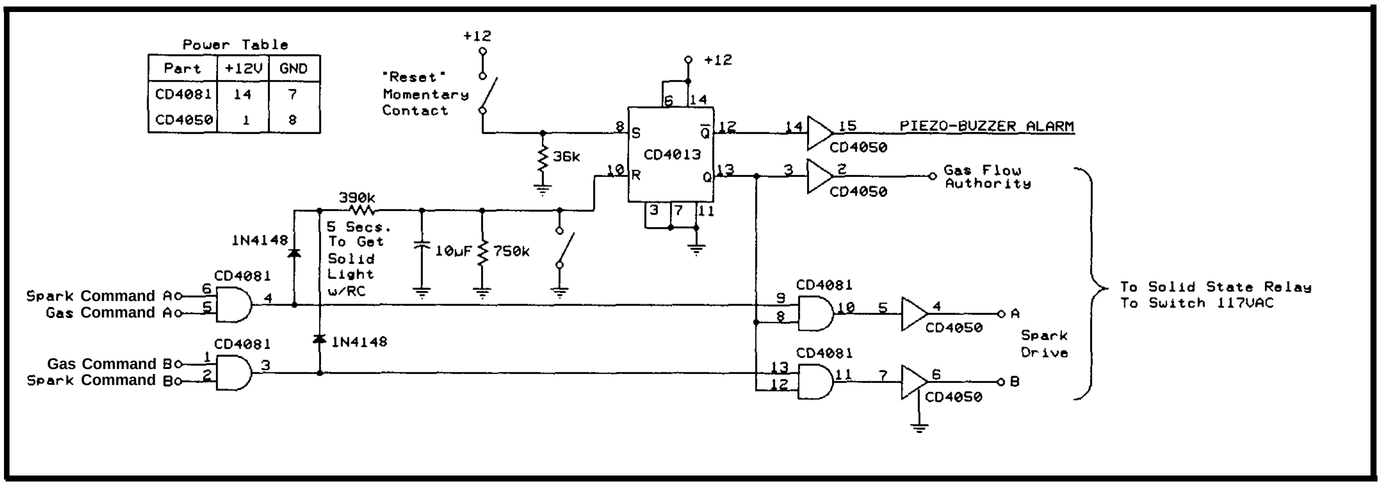


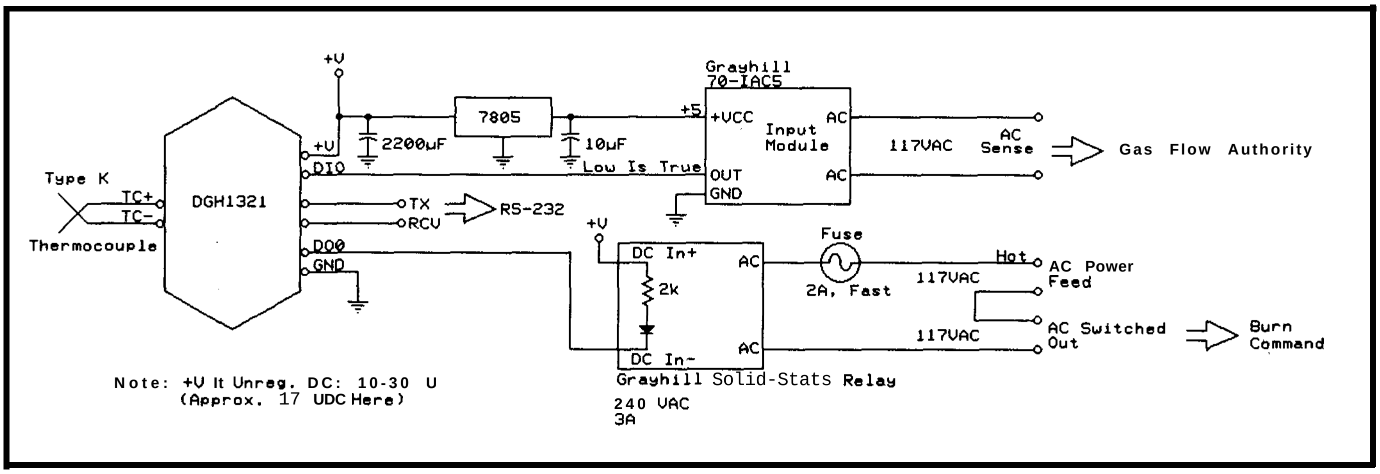
Figure 5-This timeout *circuitry features an annoying alarm notifying* the user when Gas *Flow Authority has been revoked. The most recent problem invoking this circuitry* involved a *fuel* regulator frozen *solid with ice in 95"* heat

detected. The tube is not self-quenching, so the driver circuitry must reduce the voltage across the tube after discharge starts until discharge has ceased. This is simply done with an RC network as shown in Figure 2. The 220-pF capacitor defines the limit of the total energy available to the discharge process so damage to the tube does not occur. The specifications for the R2868 recommend an average discharge current of only 100 µA (1 mA max.). Under operating conditions, the circuit shown generates a pulse train between 0.5 and 1 volt in amplitude, which is then available for further processing.

Coming up with power for the UVtron isn't as difficult as it might seem at first. You need tenths of a milliamp between 300 and 350 volts for the R2868. I found that the flash units in disposable 35mm cameras generate 325 volts. The units store that voltage in a capacitor until discharge time. The supply had no trouble with the UVtron current requirements. I also built a voltage quadrupler from 120 VAC using diodes and capacitors in the standard ladder configuration. This system, with filtering and zener regulation, also works well. A variety of companies also sell small pulse transformers which could be used to construct a high-voltage supply. I chose to use the microminiature Rico (12AV400) high-voltage supply. It was encapsulated and had plenty of reserve capacity.

The UVtron is quite sensitive. Hamamatsu claims the R2868 can detect a lighter's flame from over sixteen feet. My own testing shows that discharge events were triggered by such a flame from a broad area in the small workroom I use. The thresholds used in the current design are set to detect a lighter at about five feet. At five feet, the tube is very actively discharging with a slight purple light. Note that tubes should not be allowed to see each other discharging as the event is UV rich. The best source of UV in the flame is the root area at the burner's mouth. The UVtron can be looking coaxial to the burner boresite, or looking obliquely at the burner entry point into the oven. In my system, I chose to mount the UVtron within the burner airstream. This reduces the number of oven hot face penetrations.

The pulse-forming circuitry and the UVtron fit into a slender phenolic tube which slides into the burner assembly as shown in Figure 3. The UVtron looks down the bore into the oven. Previous mounting configurations used copper tubing, with



provisions for forced-air cooling. I discovered that I didn't need the internal cooling, and the metal contact with the ignition circuitry was causing unusual behavior from the signal conditioning circuitry. I found the phenolic tubing ideal since it is easy to work with and is a good insulator.

Cooling is now externally supplied by fans when the burner is off, and is supplied by the feed air when the burner is on. The base of the phenolic tube connects to a small PVC housing which contains UVtron support electronics. Within the small housing is a Pico miniature 400-volt power supply, an amplifier, and the CD4049 inverter, which I used as a "differential line driver" to minimize data artifacts. Each of the four devices in the housing was wired separately (no circuit board) with very close component placement. After successful testing, many layers of clear nail polish were applied to each circuit assembly. The insulated pieces were then stuffed into extremely tight quarters with the feed cable and epoxy filled to seal out mother nature. The feed cables to both flame sensors are about twenty feet in length. The cables run to the control panel through occasional mud.

If you haven't guessed by now, the electrical environment is very noisy. There are nine solenoids clacking, three big cycling motors, a pair of spark plugs, and a sad host of 60-Hz cabling everywhere. The first-generation system suffered some noise problems due to poor attention to proper grounding and shielding termination techniques. The current design has proven successful in the harsh environment.

The cables running back from the UV sensors have two shielded twisted pairs of 22 AWG wire. The shields are terminated at the signal conditioning circuitry. One pair runs 12 volts to the sensor package, while the other returns the 12-volt square wave pulses from the tube. Figure 4 shows the integration processing of the pulses. I used an LM392 for the convenience of having an op-amp and a comparator in a single 8-pin DIP package. With my cabling and termination, the gates were delivering a clean 6-volt signal at the resistive load. [While this is entirely adequate for this application, longer cable runs would probably demand a "real" line driver and receiver.)

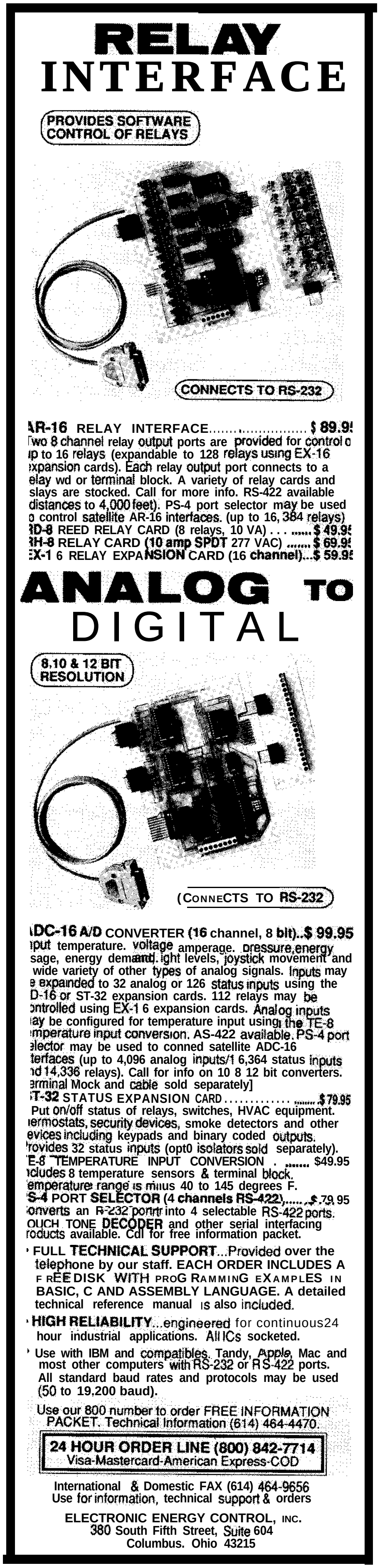
The rest of the circuit integrates and thresholds to provide a Boolean flame signal.Ignition is required when the output of the comparator is high. The time constants used in the integrator are not critical. The signal ramps up quickly and decays within a half second with the values shown. The values chosen for your application could vary, depending on the stability of the burners, for instance. Make sure the UVtron can't see the spark plug gap or the system might oscillate on and off even though the gas isn't burning! Sparks generate UV.

Figure 4 also shows input isolation from the console. The DC used in much of the control panel operations is 19 volts. This voltage was mandated by a box full of surplus gas solenoids, but it is also conveniently compatible with the solid-state relays. Front panel commands are converted to 12 volts through the isolator and fed to the simple time-out logic shown in Figure 5. The signal shown as Gas Command is the front panel switch ANDed with the PC burn request signal using a mechanical relay.

When gas flow is requested at either burner, and there is no UV detected at the burner, then the system has roughly five seconds to establish steady combustion. After that time, the signal Gas Flow Authority is removed by the CD4013 flip-flop, and an obnoxious alarm goes off. The signals Gas Flow Authority, Spark Drive A, and Spark Drive B run out to the power supply box, where all 60-Hz operations are performed. Three small, solid-state relays switch the AC to drive 120-VAC drop-out relays and the solid-state ignitors. The signal processing box has no AC power or signals in it.

IGNITING THE BURNERS

The burners are lit with standard spark plugs. I threaded steel pipe and welded into the 1 S-inch burner tube. The spark plug cavity sits very close to the back wall of the oven. It should be



"upwind" but reasonably close to where you find the flame attachment point to be on your burner. I used a Lenox-MacLaren ignitor module to fire the plugs. It has a standard automotive plug fitting, is completely encapsulated, and isolated from the line voltage. I went to a performance automotive store and had a pair of eight-foot cables made with spark-plug fittings on both ends. These are high-temperature, solid-conductor cables that were expensive, but worth it. They sit in the mud and still perform. Standard feed wire for a neon sign is good for experimentation, but would require a relatively cool installation for longevity.

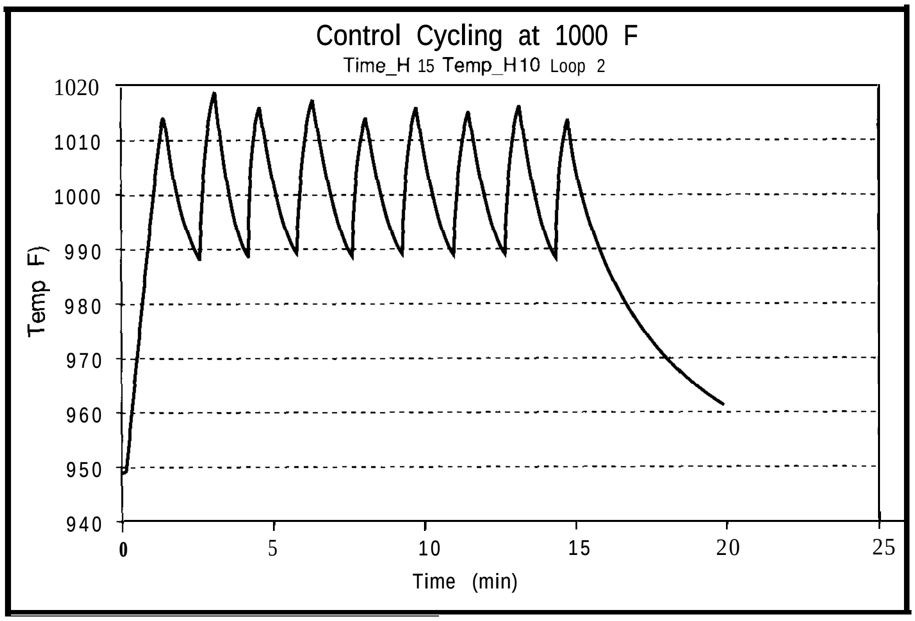
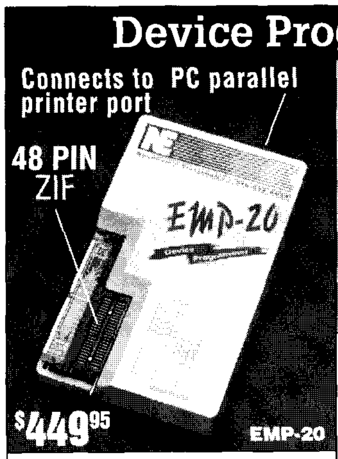
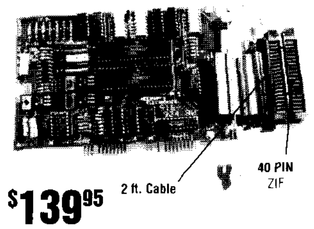


Figure 7—*Control* cycling for fifteen minutes at 1000°F overshoots the intended target.

The ignitor modules are screwed to a bracket which is mounted to the heavy steel of the chimney frame, which in turn is bolted to the frame of the oven. The ignition current from the ignitors runs through the burner assembly into the shell of the oven, where it eventually finds the chimney. In my zeal to reduce noise during early experiments, I chose to provide a very clean return path for the ignitors. The chimney, the oven frame, both burners, and the ignitor bracket were tied together with 12-gauge copper wire. These lines join at a common point, which is a six-foot copper grounding rod sunk behind the oven. If nothing else, the oven is better prepared for lightning. The ignition wires broadcast quite a bit of noise, and should be as short as possible. Most commercial flame control literature describes similar remedies for noise interference and grounding





problems. Proper shielding and grounding practices must be observed.

DECIDING WHEN TO BURN

The last box under the control panel is the temperature control module. It contains two I/O modules: one 120-VAC input and one 120-VAC output. The I/O modules support a small microprocessor-based thermocouple interface device made by DGH. The DGH D1321 is a small (fits in your palm) general-purpose module designed to support process control. This particular module uses RS-232 for communication and reads type-K thermocouples. It also has a couple of input and output lines, and could actually have performed a temperature controller function in a stand-alone mode. I chose to use it only as a remote I/O device for my PC.

The PC is located in a nearby building that is air conditioned. The temperature control box and the PC are linked with a 100-foot RS-232 link, running at 9600 bps. At one point in the development, I considered building up the remote I/O capability from one of the many single board computer options. I opted for immediate satisfaction and bought the D1321.

The temperature control module reads the Gas Flow Authority line through the 120-VAC input module (see Figure 6). A 5-volt signal is sent to the D1321 as long as the flame monitoring circuitry believes safe conditions exist at the burners. At PC instruction, the D1321 issues the Burn Command, which is translated to 120 VAC and output for relay use.

The D1321 also reads the 10-gauge, type-K thermocouple embedded in the wall of the oven whenever the PC requests the read. The bead of the thermocouple just extends into the oven volume, with a fine layer of fireclay bonding it to, the oven wall. This helps to protect the thermocouple, but at lower temperatures, it also keeps the system from rapid cycling. Rapid cycling occurs when the thermocouple is responding to the hot combustion gases and not the actual oven hot-face temperature.

The fire coupling to the thermal mass of the wall essentially low-pass filters the temperature data. Some of you may be able to afford an alumina ceramic sheath for the thermocouple. This would perform some filtering in addition to providing an ideal environment for the long-term survival of the thermocouple. Careful placement of the thermocouple and a little fireclay can perform well, and the price is right.

A note about thermocouples. If you plan to have unusual combustion atmospheres at very high temperatures (corrosive or strong oxidation), you should have a few spare thermocouples. It is difficult to make your

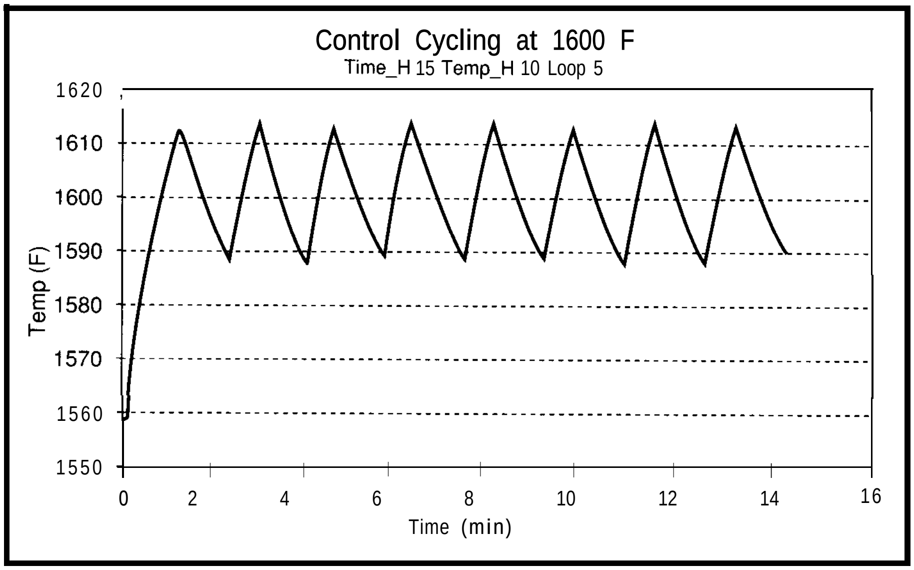


Figure 8—*Control* cycling for fifteen minutes at *1600°F* results in a slower moving, easier to control oven temperature.

own heavy-gauge thermocouples, but companies like Omega Engineering sell them affordably. For applications which do not require temperatures close to the melting point of the thermocouple, you would do well to buy a good-sized spool of the smaller-gauge pair. Type-K thermocouples with the woven ceramic insulation should suffice for most experiments. The beads can be made with any acetylene torch, or you may spot weld them with a big capacitor.

The program MOLTEN runs on the PC to coordinate the whole oven control problem. MOLTEN reads in command files and attempts to make the oven perform as directed. It monitors, but does not command, the flame maintenance operations. The program requests burns to be performed, and monitors temperature inside the oven. It uses the DGH module as an intelligent peripheral in a harsh remote location. MOLTEN was written in C to provide the control needs of the oven, meaning that it does not have a slick user interface. MOLTEN uses the interrupt-driven serial communications routines supplied by DGH with their software and samples disk. The software is otherwise without complexity.

The control scheme I chose for this first-generation MOLTEN is often referred to as bang-bang control. The state of the system (on or off) is mandated by system position relative to two limits. Most home ovens work this way, cycling between the hysteresis boundaries around some chosen control point. The command Temp_Hyst is used to define the acceptable hysteresis for the system.

Time_Hyst	num		(seconds)
Temp_Hyst	num		(degrees F)
Logfile	filename		(file put in local directory)
LogLoopTime	num		(seconds between data samples)
Wait	num		(minutes, log active)
Enable			(burner use authorized)
Soak	numl	num2	(numl = temp F, num2 = minutes of soak)
Disable			(burner use no longer authorized)

Figure 9a—*The MOLTEN control program reads ifs commands from a prepared command file. The program supports enough to allow intelligent control of the burners*

```
* this is a command input file for MOLTEN.EXE
Time_Hyst 15
Temp_Hyst 10
LogLoopTime 5
Logfile dsm7.out
Enable
Soak 1600 120
Disable
Wait 180
```

Figure 9b—*A MOLTEN command file used to generate oven cooling curves is shown. The file DSM7.OUT would contain a header of useful information followed by two columns of data containing time in seconds and temperature at that time. This data is easily imported into spreadsheet programs for analysis.*

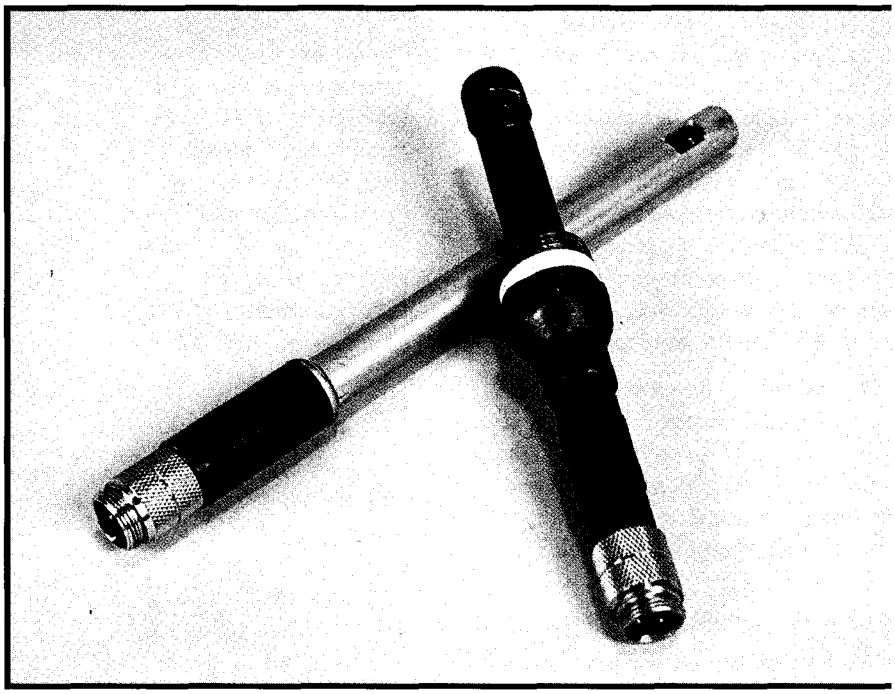


Photo 3-The *UVtron* sensor is built info pieces of *tubing* to ease its installation. The sensor actually peeks out the *window at the top of each tube.*

I have the additional constraint that the system can't be quickly cycled. There is a substantial fraction of a second required to open the four-inch butterfly valve which provides the feed air to the burners. An artifact of this slight delay is a rich mixture for lighting, which is beneficial. It takes even longer for the valve to close, perhaps ¾ second. The delay results from the large pneumatic actuator the valve uses, and is not unreasonable. What this means, though, is that I really can't cycle the burners on and off at a high rate of speed. There is also wear and tear to be considered. To account for mechanical limitations, the command Time_Hyst (temporal hysteresis) gives the system a grace period after a transition from either state.

The command LogLoopTime sets the **time** period to elapse between temperature samples in seconds. Each time the D1321 is called, the state of Gas Flow Authority is reported, and the temperature is reported. There is also error checking during reporting: **MOLTEN** and the D1321 processor check command and response messages against generated checksum bytes appended to messages. Presumably, the LogLoopTime could be set to fractional values for sampling several times a second, but my system doesn't need that kind of data bandwidth, and the log files get huge.

If a log file has been declared, the commands Wait and Soak will generate new data for the log file at the LogLoopTime interval. Otherwise, the time and temperature will only be logged to the screen. The command Wait is specifically for taking data, letting the oven coast while logging temperature.

The Soak command actually forces the oven into action. It is a time-based loop that uses the PC internal clock to gate its actions. The values given to Time_Hyst, Temp_Hyst and LogLoopTime control the actions of Soak. Examine Figures 7 and 8 to see the results of the Soak command on the oven. Both plots show fifteen-minute burn schedules around a target temperature.

From examining the two charts, it is apparent that the slopes of attack and decay are different for different temperatures. The higher the slope, the more the overshoot in this simple control scheme. At 1000°F, the overshoot can be almost 10” beyond the hysteresis band. At higher temperatures, the oven temperature moves slowly and is easier to control.

My applications don't require accuracies near ±10° of a target temperature, but I'm pleased the system is that nimble. For ceramic shells, I work in the 1600-1900” range where that kind of tolerance can easily be maintained. I have not yet fired pottery in the oven, but at a cone 10 firing temperature (from 2345 to 2381 °F depending on heating rate), the oven should be very cooperative (slow) to control.

A complete list of the commands that are currently processed from the input file are shown in Figure 9a. Numeric arguments are of type **INT.** Figure 9b shows a typical command batch file. Lines are parsed into tokens separated by white space. If the first token in a line is a known command, the line is interpreted. Otherwise the line is a comment line. It requests two hours at 1600°F followed by three hours of data logging. Data is taken every five seconds.

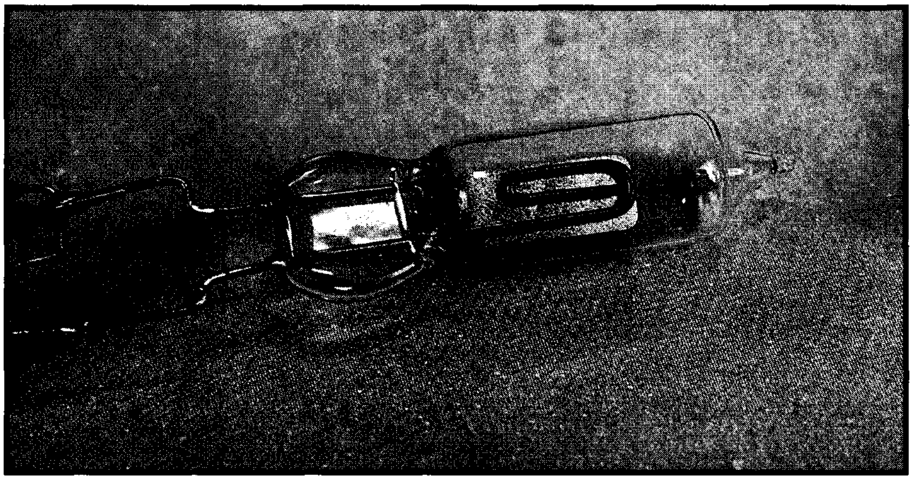


Photo 4-A closeup of **the** *UVtron* **shows its construction.**

ADDITIONAL THOUGHTS

There is one safety feature I should add. In retrospect, it would be good to monitor the spark command line at pin 1 on the LM392 amps. Both lines could be inverted and ORed and fed to the remaining input on the D1321. In this way, testing of the UVtrons could be performed occasionally. One of the failure modes of the UVtron is constant discharge. Whenever the burn command is removed, the spark command lines should show no combustion. This would also test some relays in the system. Some commercial systems have a self-checking feature which, is a shutter that covers the tube at a regular interval. The processing circuitry then expects to see a regular pattern of notched data from the tube.

I will probably add a Ramp command to the available commands in MOLTEN at some point in the future. That would allow controlled gross temperature changes, which is currently only possible by taking steps with the Soak command. Something large in the oven would warrant controlled ramping to reduce the chances for cracking. While it might be fun, I probably won't add auto-tuning PID control algorithms to MOLTEN since the current temperature control scheme meets my needs. This project has switched from something to improve into something to use. My first bronze castings were poured last spring, I intend to make more. ❑

David McFalls spent seven years in the Machine Perception Section of the Robotics and Automation Department at the Southwest Research Institute. He now ***pursues*** *other interests including alternative energy and sculpting.*

SOFTWARE

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

Omega Engineering, Inc.
P.O. Box 4047
Stamford, CT 06907-0047
(203) 359-1660
Fax: (203) 359-7807

DGH Corp.
P.O. Box 5638
Manchester, NH 03 108
(603) 622-0452
Fax: (603) 622-0487

Hamamatsu Corp.
P.O. Box 6910
360 Foothill Rd.
Bridgewater, NJ 08807
(201) 231-0960
Fax: (201) 231-1218

IRS

401 Very Useful
402 Moderately Useful
403 Not Useful

FEATURE ARTICLE

Tom Dahlin

Stepping Up in Performance

Driving High-power Stepper Motors

Moving disk drive heads and mobile robots is easy with small stepper motors and their low power requirements. When you want to move mountains, though, you have to bring in the big guns. Here's how.

regular readers of the Computer Applications Journal have been treated' to several articles over the years on stepper motor applications and drive circuits. In all cases to date, the stepper motors used in those articles were of the low-current type, which I will characterize as requiring under 1 amp per motor coil. Several ICs exist which are capable of directly controlling motors in this class, the most common being the UCN5804 from Allegro Microsystems. Such devices make interfacing a microprocessor to stepper motors a snap, conserve board space, and generally make life easier for the system designer.

You may find it helpful to dig out and review material on stepper motor construction (see references), bipolar and unipolar drives, and step formats. To save space here, I'm going to assume you have that stuff down pat.

What do you do when the need arises for a motor that requires more current than can be provided by the UCN5804? Stepper motors are available in torque ratings of over 500 inch-pounds, making it possible to easily control some really neat (albeit bordering on scary] stuff. For instance, think of a computer-controlled X/Y table with a wood router as a pen. The only thing separating your computer from those heavy-duty applications is a method of driving the larger motors.

In this article, I will discuss drive circuits that are appropriate for medium to large motors requiring from one to four amps of current. I will start by adding a power-boosting stage to the UCN5804. I will then contrast that circuit with a more efficient chopper type. Finally, I will conclude by showing some low-chip-count implementations of chopper-based stepper motor drive circuits.

MOTOR RATINGS AND SIZES

Manufacturers of stepper motors usually group them by size. It is common to find motors listed as *NEMA* size 17, 23, 34, and so forth. NEMA refers to the National Electric Manufactures Association which has [among other things) standardized the base dimensions for motor cases and mounting hardware. You can get a feeling for the size of the base by dividing the NEMA code by 10. For instance, a NEMA 34 motor has a base of roughly 3.4 inches on a side. Table 1 shows a *rough* guide to what is available and the cost on the surplus market.

DRIVER TYPES

The two most popular methods for providing current to stepper motors are L/R [pronounced "L over R") and chopper drivers. Each of these drive methods can be used in a unipolar or bipolar configuration, but most L/R drivers tend to use a unipolar drive while the chopper types most often use a bipolar drive.

NEMA code	Base Size (in./side)	Torque Range (oz.-in.)	Current Range (amps)	Surplus Price
17	1.65	15-30	0.2-i .2	$8–$12
23	2.22	40-I 50	0.2-4.0	$1 0–$30
34	3.27	150-300	1–8	$20–$70
42	4.20	500-I 300	2-I 0	$40+

Table I--The *National Electric* Manufacturers Association *lists common stepper motors available in today's market.*

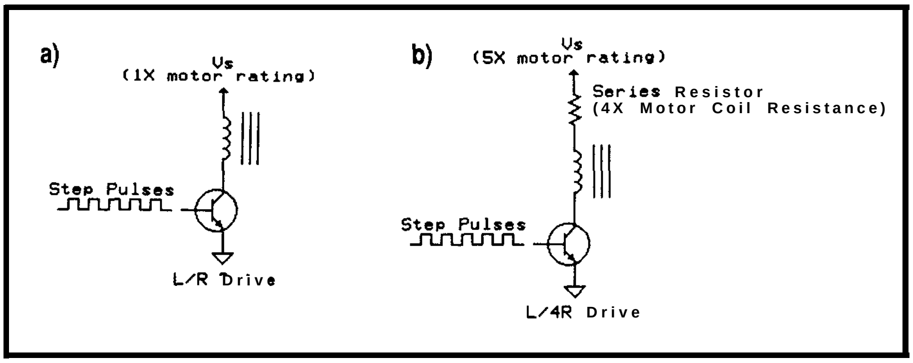


Figure 1-a) *In a simple L/R driver, the voltage source, V_s, is chosen to be equal to the motor's rated operating voltage, and the current is limited by the coil resistance. Adding an external series resistor shortens the L/R time constant of the circuit and improves performance at higher stepping speeds. b) In an L/4R scheme, the series resistor is four times the resistance of the motor coil, and the supply voltage is increased to five times the motor's rated value. Note that ⅘ of the drive power is going into the resistor when this scheme is used.*

You may also have heard the term "microstepping" driver. This is a technique used to produce many intermediate steps between the motor's natural steps. Remember how we can produce a half step position in the motor by turning on both coils at the same time? A microstep driver carries this principle further by using two DACs to produce sine and cosine drive waveforms to the motor instead of on/off step pulses. Current control is usually accomplished using a chopping technique for microstep drives, but I won't get into that technique here.

L/R DRIVE

The L/R method uses the motor's internal resistance, sometimes in conjunction with an external series resistor, to limit the current through the motor's coils. It is most often used to drive small stepper motors using a unipolar drive. The series resistance of the circuit limits the current flow to prevent the motor coil from burning up. This circuit is shown in Figure la.

The L/R method starts to present problems as more performance is demanded from the motor. The two main problems are loss of torque at high speeds and poor power efficiency.

Let's address the first issue. The loss of torque is due to the current rise/fall time limiting effect caused by the inductance in the motor coils. The current in an LR series circuit requires about five time constants to reach a steady state once an impulse (i.e., a step) is presented and another five time constants to decay back to zero when the impulse is removed. This time constant (in seconds) is calculated as the motor's coil inductance (in henrys) divided by the circuit's series resistance (ohms]. As shown in Figure 2, at low speeds [less than 100 steps per second) the LR time constant is usually not an appreciable amount of the duty cycle. But as the step rate is increased, we eventually reach a point where the current has not yet reached a steady-state maximum before the drive current is switched off. In the same way, the current cannot decay to zero before the next step is applied.

To overcome the limitation caused by the LR time constant, we can increase the resistance of the circuit by using an external series resistor (R_s) and increase the voltage applied across the series pair as shown in Figure lb. The value of the series resistor is typically chosen to be two to four times more than the motor's internal resistance. In an L/4R driver, the series resistor is four times the internal resistance of the motor and the applied voltage is increased to five times the motor's rated value. This has the effect of reducing the time constant by a factor of four, thus allowing faster step rates. The tradeoff is that the series resistor is using up most of the power going into the circuit. Consider that a 10-ohm series resistor with 3 amps going through it will dissipate 90 watts! Better not touch that little radiator with your bare hands.

A HIGH-POWER L/R EXAMPLE

Figure 3 shows a circuit that adds a power booster stage to the UCN-5804. This is done by using four Darlington transistors. While having the advantage of being easy to understand, it requires good construction and power wiring techniques. It is particularly important to place the protection diodes near the Darlingtons and to provide a good, low-inductance current path for the Darlington emitters. Pay attention to the power dissipation of the series resistors, R_s. These are best mounted directly on a chassis, rather than on a circuit board. My breadboard of this circuit is shown in Photo 1.

CHOPPER DRIVES

Chopper drives offer an improvement in efficiency and speed performance as compared to L/R drivers. A basic chopper drive is illustrated in Figure 4. Here, a supply voltage is used that is five times that of the motor's rating. If the power transistor were turned on and left on, the current

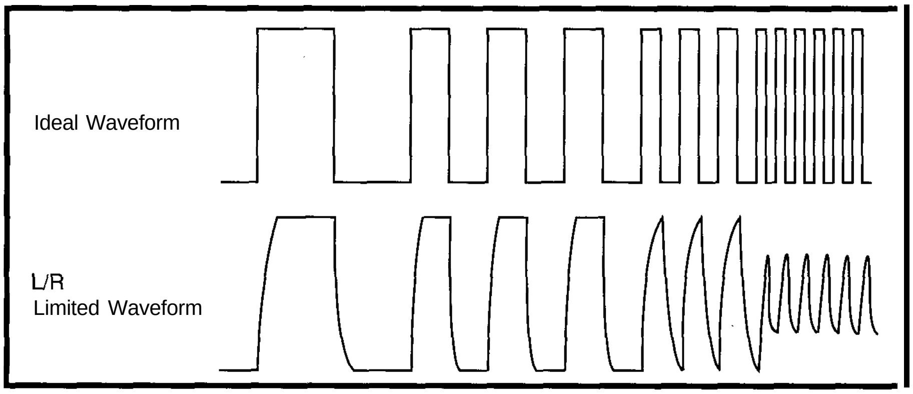


Figure 2-M *time constant limits rise/fall times. Average current available to the motor coils decreases as the stepping frequency increases.*

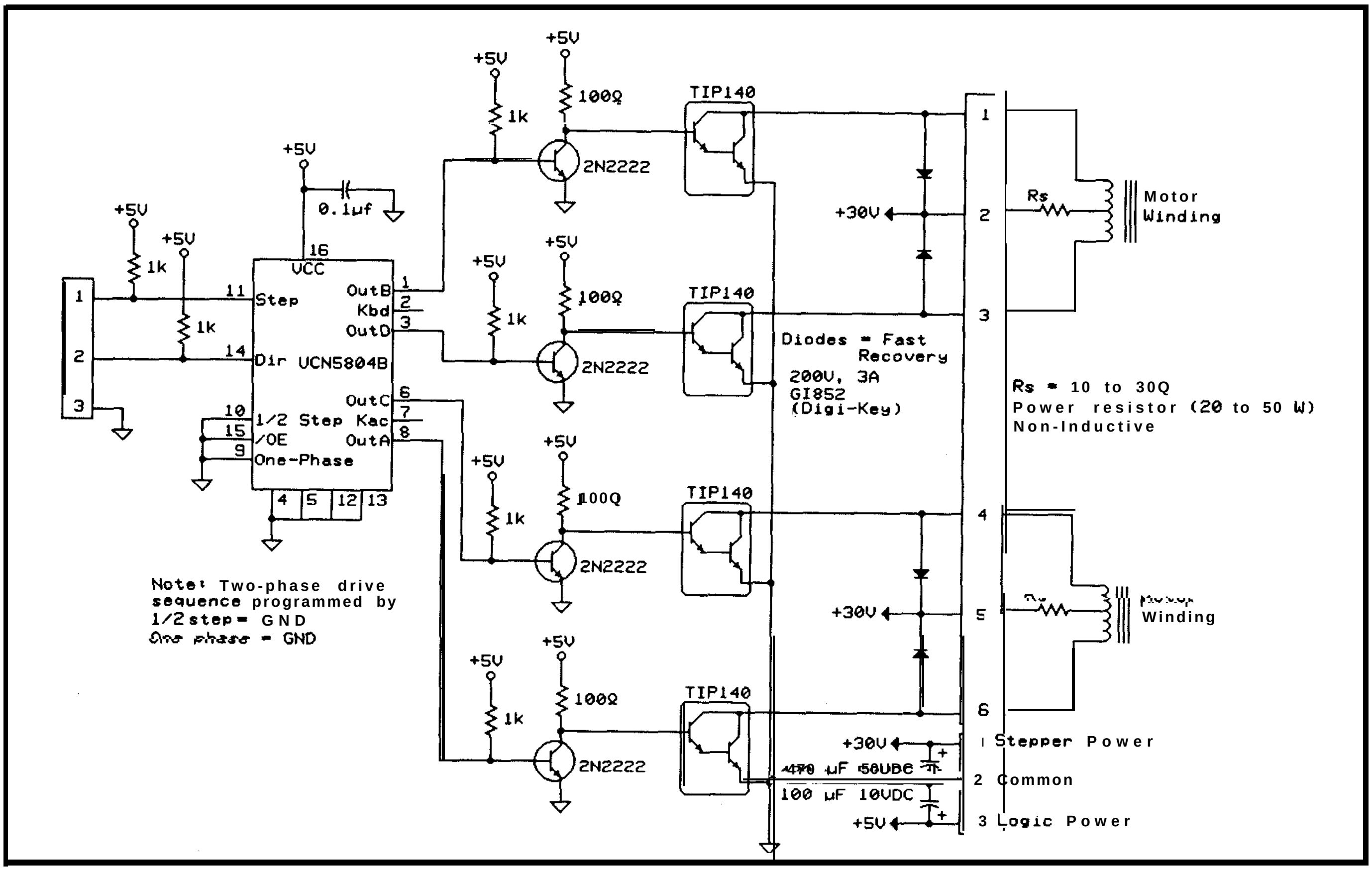


Figure 3—*High-power L/H driver* doubles as a bagel toaster. While *offering the advantage of a simple* design, *this circuit is wasteful or power due to the voltage drop across the series* resistors. *For best speed performance, use a 24–30-volt supply for the stepper motor and select R_s resistors to limit the motor current to the rated value.*

flowing through the collector circuit would eventually damage or destroy the motor. To prevent this from happening, a circuit designed to monitor the motor current with a low-resistance [typically 0.1 to 1 .O ohms) sense resistor is placed in the collector circuit. The voltage across the resistor is used to trip a comparator. The comparator's threshold is set with a reference chosen to equal the voltage across the sense resistor at the desired maximum current.

Why is this approach better than an L/R scheme? The answer is that it eliminates the power-wasting series resistor and still retains a fast current rise time by increasing the applied voltage to the coil.

CHOPPING CHIPS

SGS-Thomson offers several chips designed for stepper motor applications. Of particular interest are those that provide high-power drive capability with a minimal component count.

I am going to discuss the L297 controller chip, which, when combined with either the L298 or two L6203 driver chips, can provide up to 4 amps of drive current per coil.

OVERVIEW OF THE SGS CHIPS

I'll start with the L297 controller chip (shown in block diagram form in Figure 5). Like the UCN5804, this IC generates a set of four drive signals in half-step, full-step, and wave-drive modes. An on-chip PWM chopper circuit provides current sensing and control. Your microprocessor needs to provide step (called CLOCK hereafter}, direction, and a few control signals discussed later. The device is packaged in a 20-pin DIP.

The outputs from the L297 are designed to interface to a driver device that is used to switch the motor

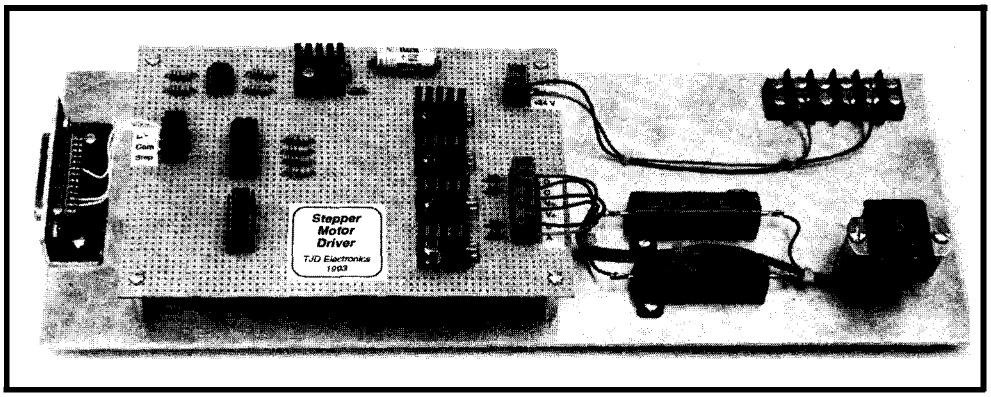


Photo I--This *high-power L/R driver breadboard uses four Darlington transistors to increase the drive capacity of fhe UCN5804 to over 5 amps. A 5-volt, 2-amp motor was driven by the circuit. Note the 50-watt power resistors are mounted off the circuit board and use the aluminum base as a heat sink. This approach has the advantage of being cheap and easy to build, but is extremely power wasteful.*

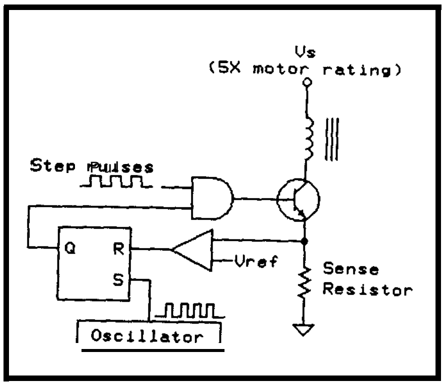


Figure 4—*The chopper drive circuit uses a voltage source to provide an overcurrent to the motor if not controtted. Control is accomplished by using a comparator to sense voltage across a low-resistance sense resistor. Chopping action occurs at the oscillator frequency, typically 20 kHz.*

current. Two useful devices that can be applied here are the L298N and L6203. The L298N [see Figure 6) is a dual full-bridge driver capable of providing output current up to two amps per coil (steady state) at up to 45 volts. It uses bipolar transistor drivers in its output stage and is packaged in a 15-W power tab package (a lower power version, the L293E, is available in a 20-pin DIP). The emitter connections of the lower power transistors are brought out to allow the insertion of a low-value current sensing resistor for use by the L297. Eight fast protection diodes are required for transient protection. I have used the 1N4935 (available from Digi-Key}, which is a 1 -amp, 200-volt type, with good success.

The L6203 (Figure 7) is the higher power member of the chip family whose members include the L6201, L6202, and L6203. The L6203 is a DMOS device capable of passing 4 amps at up to 48 volts. It is a single H-bridge device, so two are required for a stepper application. Because DMOS switches are used, an intrinsic diode exists across the source/drain connection that can act as a freewheeling diode to protect the device from transients. An RC snubber circuit across the motor coil is used to limit the voltage rise time to allow the intrinsic diode sufficient time to act.

Figures 8 and 9 show how the L297 controller is interfaced to the L298 and L6203 devices. I will take no credit for these designs, they are straight from the SGS data book. I have, however, built and tested them.

APPLICATION NOTES

A full discussion of the SGS devices is beyond the scope of this article and would essentially duplicate information existing in the manufacturer's data sheet and application notes. Obviously you should get these from SGS and study them in detail before plowing into a circuit design. What I hope to provide here is some amplification of subtleties.

Referring back to the L297 block diagram in Figure 5, note the block labeled "translator." This block generates the phase sequences for half-step, one-phase-on (full-step], and two-phase-on (full-step) operation. It is essentially a state machine driven by a three-bit counter that is clocked by the

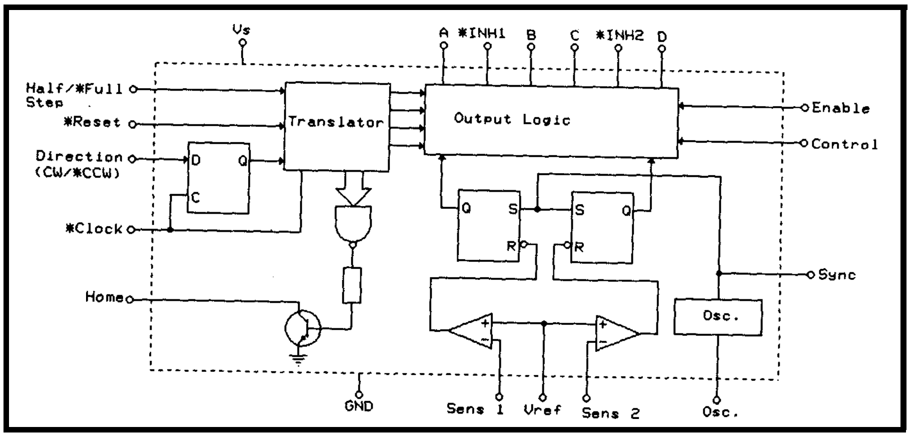
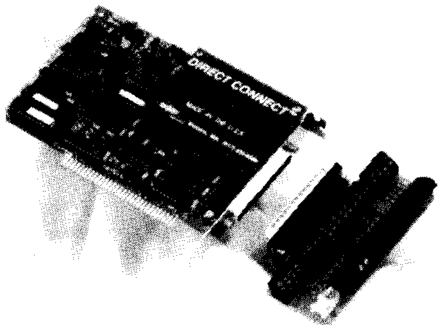
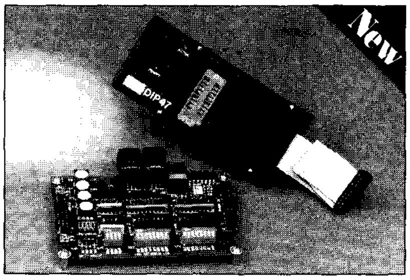
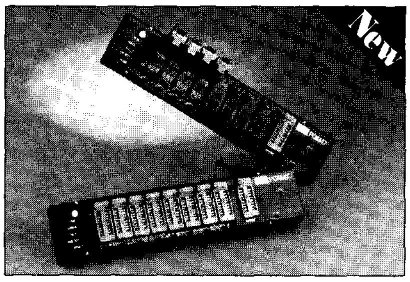


Figure 5—*The L297 includes aft logic required to translate step and direction signals from your microprocessor to the drive sequences needed to control an external output stage. Comparators, flip-flops, and switching oscillator needed to implement chopper control of motor current are all built in.*







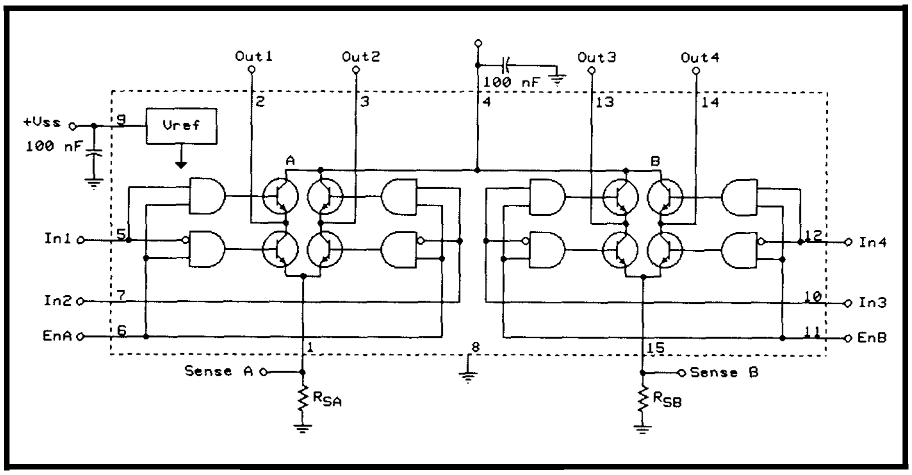


Figure 6—*The L298* **is a** *dual full-bridge driver capable of passing 2 amps '(DC) at up'ro '46 volts. Externai' sense* **resistors are used to provide current sense voltages for the** *L297.*

CLOCK* input. The translator output has eight different states. When you select the half-step mode by driving the HALF/FULL* input high, the translator cycles through all eight states as the CLOCK+ input is clocked. When you select the full-step mode by bringing HALF/FULL* low, the translator skips every other state to generate either a one-phase-on or two-phase-on drive sequence. Which one you get is dependent on where the state machine was when you drove HALF/FULL * low.

You can force the translator's state machine to its first state by pulsing RESET* low. If HALF/FULL* is low at this time, or is brought low before a CLOCK' pulse clocks the state machine, you will generate a two-phase-on full-step sequence. To get a one-phase-on full-step sequence, you need to begin with the half-step mode selected (HALF/FULL* = high), reset the state machine to the first state (pulse RESET* low), advance to the second state by applying one clock pulse to the CLOCK* input, and then select full-step mode by bringing HALF/FULL* low. All of this should be done on initialization by your host microprocessor.

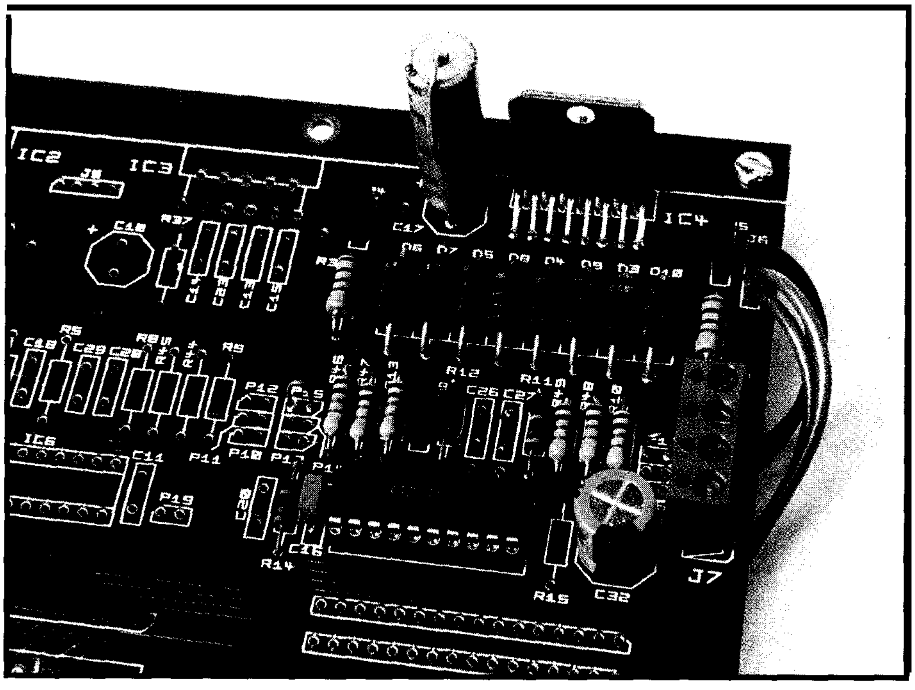


Photo 2-A *photo of the SGS-Thomson HWPC6203/298 evaluation board shows an idea/ way to try out the SGS 1297, L6203, and L298. Three different circuits are laid out on the board including the L297/L298, L297/L6203 combinations and a DC motor driver using the L6203. The L297/L298 combination has been populated in this* photo.

The CONTROL input to the L297 determines whether the chopper will act on the phase outputs (A,B,C,D) or on the inhibit outputs (INH1*, INH2*) to control the drive current. Bringing CONTROL to ground tells the L297 that you want the chopper to act on INH1. and INH2 *. Nailing CONTROL high results in the phase outputs A,B,C,D being chopped. Normally you would want to have the chopping occur on INH1* and INH2* because it would provide for a faster current decay in the motor windings.

The ENABLE input to the L297 must be high for normal operation of the device. When ENABLE is low, the INH1*, INH2*, A, B, C, and D outputs are all brought low. This provides a convenient way of disabling the motor drive current when you want to go to a standby mode.

The sense inputs on the L297 (SENS1 and SENS2) are high impedance and are particularly susceptible to glitches. I found that using a 22k and 100 pF low-pass filter between the sense resistor and these inputs goes a long way to keeping the circuits stable.

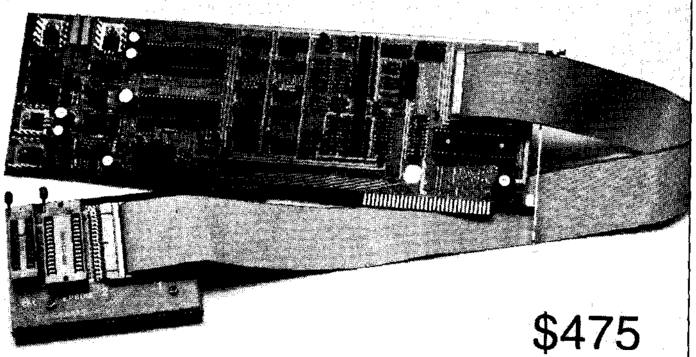
EVALUATION KITS

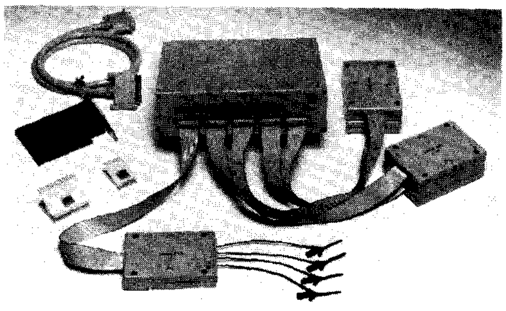
SGS offers several evaluation kits for their power devices. These kits come complete with two-sided printed circuit boards, application notes, and ICs. You need to add the discrete components and solder them up.

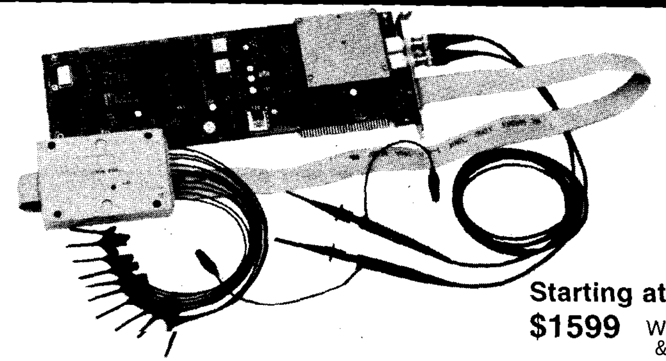
The HWPC6203/298 evaluation board contains three separate circuits. Photo 2 shows the board as assembled using the circuit in Figure 8 (the L297/ L298N combo]. Photo 3 shows the implementation of the L297/L6203 circuit of Figure 9.

The evaluation board is designed to run from a special PC-based driver board. That board, in conjunction with SGS driver software, allows you to control the L297 from the PC. I did not use the driver board, as I felt the cost (≈ $300) was inappropriate to the application. Instead, I used toggle switches to control the L297 inputs and a function generator to provide the step pulses.

I *strongly* encourage you to get one of these kits as a means of experi-







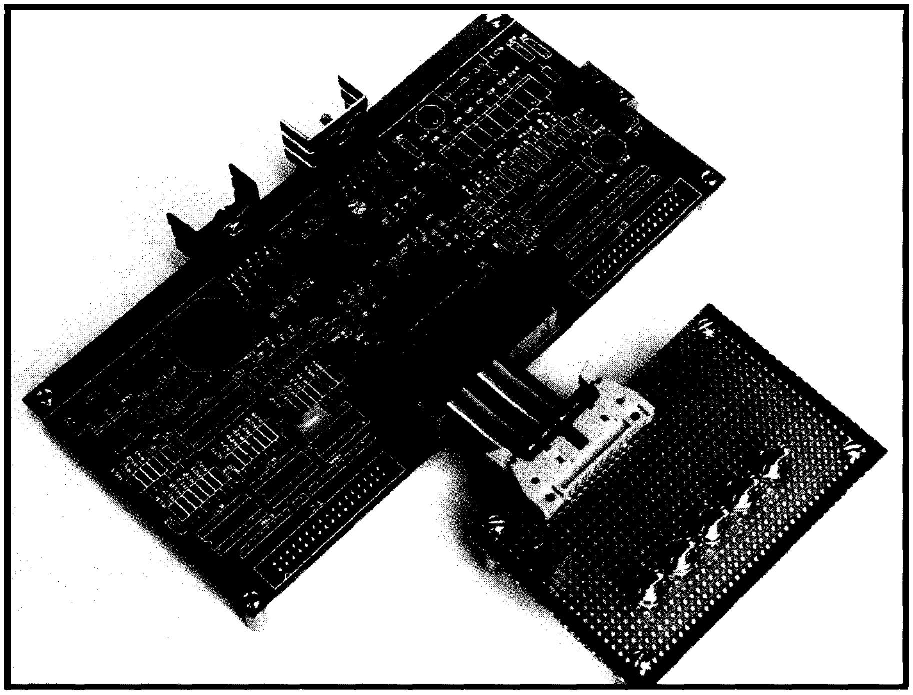


Photo 3—*The same evaluation board as in Photo 2, but this time only the L297/L6203 circuit has been populated.*

menting with the SGS chips. Even with the evaluation kits, it is not a simple matter of implementation. In any circuit that is switching heavy currents through an inductor, there is going to be a tendency for glitches and noise. Trying to stitch wire and then debug these circuits is an exercise for the masochistic. It is much better to start with a proven design and a clean board layout and prove to yourself that the circuit really works. This allows you to quickly evaluate the circuit, tweak what you need to, and then do a cut and paste into your application.

SOME USEFUL BUILDING BRICKS

Before closing, I should also mention that you can simply buy a chopper driver module. Several companies manufacture “bricks” which only require a step and direction pulse on the input side, a motor on the output side, and a 30-VDC power supply. They are commonly available up to the 6-amp size in prices starting at $150. A typical unit, rated at 2.5 amps, costs about $250. Most of the suppliers also offer microstepping versions as well. See the resource section for addresses and phone numbers.

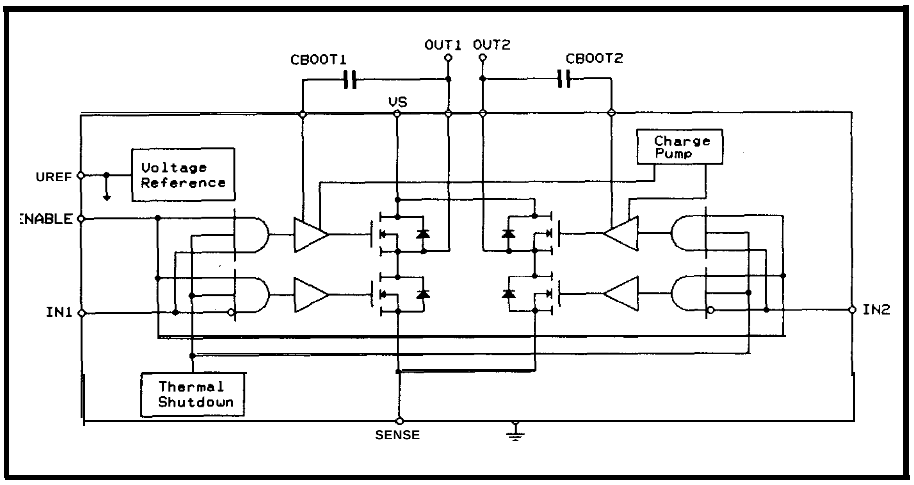


Figure 7—*The L6203 IC* uses *DMOS* power *transistors to* switch *up to 4 amps of current at 48 volts. The* **on** resistance of the *transistors is typically 0.3* ohms. If is packaged in ***an*** eleven-lead *power tab package.*

Tom Dahlin is employed as an Electronics and Software Design Engineering Specialist by the 3M Company in Saint Paul, Minnesota. His background includes 15 years of analog and digital design, motion** control,* ***and electronic imaging.

REFERENCES

Steve Ciarcia, Ed Nisley, “Circuit Cellar Stepper Motor Scanning Sensor : A look at the logic and control of stepper motors,” Circuit ***Cellar INK,*** July/August 1988.

Tim McDonough, Dennis Grim, “Stepping Out: A robot arm that demonstrates microprocessor control of stepper motors,” ***Circuit*** Cellar ***INK,*** July/August 1988.

Jeff Bachiochi, “Simple Steps for Easy Positioning,” ***Circuit Cellar INK,*** February 1993.

TRADE MAGAZINES

Here are two useful trade magazines that are offered as freebees to those in the industry. Phone and ask for a subscription request.

Motion Control-The Magazine For Motion Control Applications and Technology. Tower Media Corporation, 800 Roosevelt Rd., Bldg. C, Ste. 206, Glen Ellyn, IL 60137, (708) 858-1888.

Motion-The Guide To Electric Motion Control, Motion Corporation,.Production Division, 2030 Hillman Cir., Orange, CA 92667, (714) 974-0200.

SEMICONDUCTOR DATABOOKS

The following manufacturers’ products were mentioned or used in this article. Detailed device information and application notes are available in the following books:

Allegro Microsystems, Inc.
115 Northwest Cutoff, Box 15036
Worcester, MA01615
(508) 853-5000

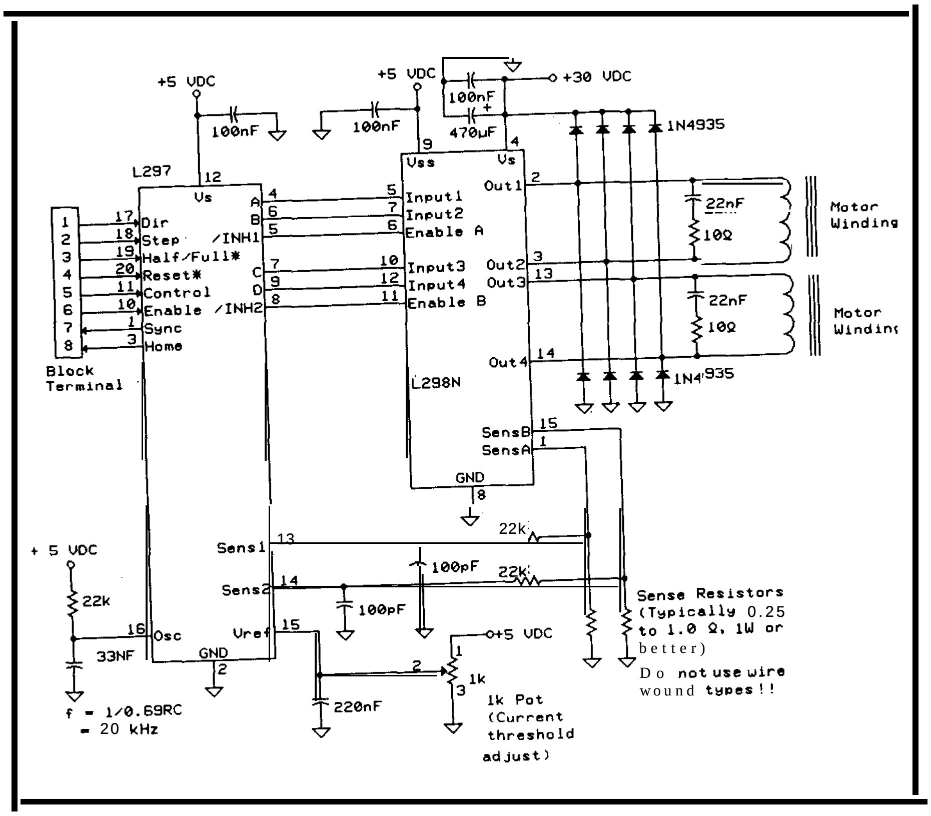


Figure 8—*L297/L298* combination provides up to 2 amps per coil. Using *only two chips and a handful of discretes, this circuit can significantly* boost performance over that of a simple resistance limited driver, *both* in *speed and* Dower efficiency.

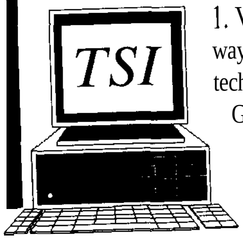
Industrial ***and Computer Peripheral***
ICs Databook ***(2nd Ed.)***
Designer's Guide To Power Products Application Manual (2nd Ed.)
SGS-Thomson Microelectronics
1000 East Bell Rd.
phoenix, AZ 85022
(602) 867-6100

MOTOR VENDORS

The following vendors all publish catalogs containing specifications for their products. Some, like Airpax, include tutorial application sections.

Airpax ***Stepper Motor Handbook***
North American Philips
Airpax Mechatronics Group
300 West Main St.
Northboro, MA 01532
(508) 393-1919

Slo-Syn AC Synchronous, Gearmotors, and DC Stepper Motors Catalog
Superior Electric Company
Bristol, CT 06010
(203) 582-9561



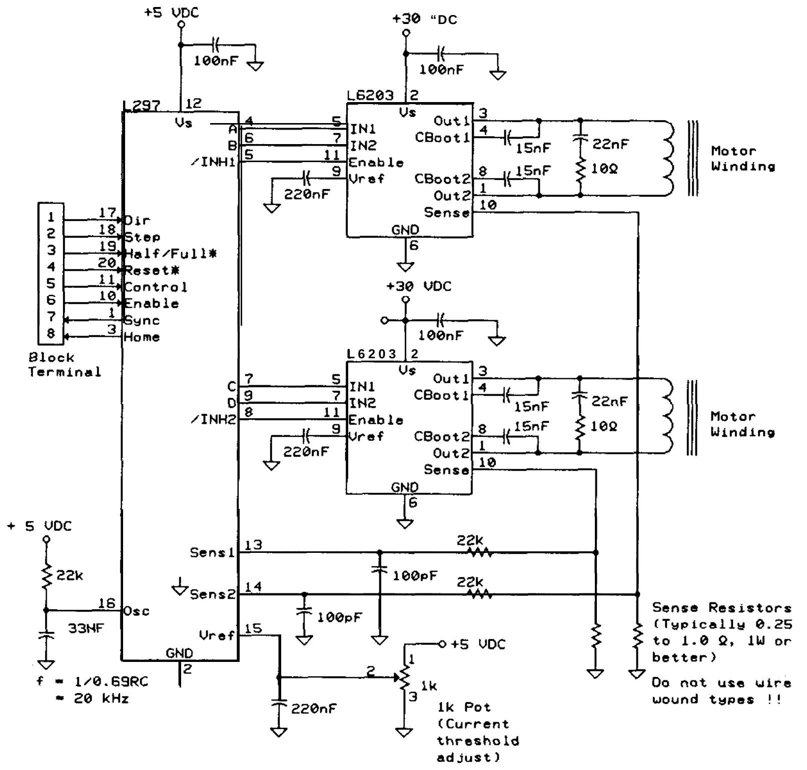
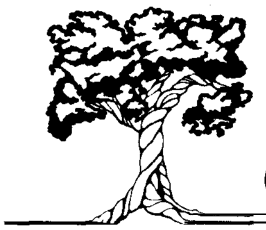


Figure 9—*L297/L6203* combination *provides up to* 4 amps per coil. Special care *needs to be taken on power trace wiring and board layout The values of the sense resistors are chosen such that the I×R drop across them will be about 1 volt at the operating current of the motor.*



Oriental Motor USA
(Vextra Brand Motors)
2701 Plaza Del Amo, Ste. 702
Torrance, CA 90503
(213) 515-2264

American Precision Industries
RapidSyn Division
3229 Roymar Rd.
Oceanside, CA 92054
(619) 439-7500

SURPLUS HOUSE

C&H Sales
2176 East Colorado Blvd.
Pasadena, CA, 91107
(800) 3259465

BOOKS

Theory and Application of Step Motors, Kuo, B.C., West Publishing, 1974.

Stepping Motors: A Guide To Modern Theory and Practice, 2nd Edition, P.P. Acarnley, Peter Peregrinus LTD, London, UK, 1984.

Motors and Controls, James T. Humphries, Merrill Publishing Company, Columbus Ohio, 1988.

READY-TO-GO STEPPER DRIVERS

The following companies offer ready-to-go stepper motor drivers in the form of packaged modules.

Intelligent Motion Systems, Inc. (IMS)
5 11 Norwich Ave.
Taftville, CT 06380
(203) 889-8353

Semix, Inc.
4160 Technology Dr.
Fremont, CA 94538
{510} 659-8800

General Controls, Inc.
2350 Brickvale Dr.
Elk Grove Village, IL 60007
(708) 595-2152

IRS

404 Very Useful
405 Moderately Useful
406 Not Useful

DESIGN CONTEST WINNER

FEATURE ARTICLE

Derek Matsunaga

Data Logging Meters are No Fluke

Digital multimeters have everything necessary for data logging applications except one thing: a computer interface. That didn't stop one intrepid designer, who found a little-known feature of certain Fluke meters.

Despite the relative age of the 80 series of Fluke multimeters, they remain one of the most versatile and popular hand-held DMMs available today. Their rugged design, ease of use, and wide range of features makes them a first choice among many engineers. Unfortunately, these meters are not known for their data acquisition capabilities.

After removing my Fluke 87 from its familiar yellow holster, I noticed the "quick reference card" molded into the back of the meter. One of the features listed on this card is labeled "Ultrasonic Data Output," which is enabled by pressing the HOLD button while turning the meter on. With this feature enabled, I noticed that the meter emitted an audible buzz each time the display was updated. As it turns out, the Fluke 80 series multimeters have the capability to audibly transmit information via their internal speakers. A project was born!

I immediately assumed that the buzzing noise was numerical data representing what the meter was measuring. After building an audio detection circuit, I began to analyze the transmitted information to determine the data format. I started by looking for some sort of BCD or similar format-to no avail. After many hours of staring at the data, I finally figured out that each bit of data represents an active segment on the meter's display.

Once I determined the data format, I added a 68HC705K1 to the detection circuit and wrote software to translate the information from the meter's format into ASCII. With the completed circuit and software, all information on the meter's display is transmitted via RS-232 to a host. This data can be viewed remotely on a dumb terminal, printed on a printer, or logged to disk for later processing. In this way, the Fluke 80 series becomes a front end to a computer-based measurement system. Not bad for free!

I should note that Fluke does not support this function as an end-user application. They do not disseminate any information about this output capability or its data format. They've told me that the ultrasonic output capability is used strictly for final testing of the unit at the factory and is not intended for data acquisition. However, it's already built into your 80 series, so why not use it?

I'll begin by explaining the basics of the meter's data format. The details of the data format are, unfortunately, pretty complicated and cannot be fully described in such a short article. Following the data format, I'll discuss the signal detection scheme and the microprocessor hardware. The hardware discussion will be followed by an outline of the software. I'll conclude by describing some of the applications for this project.

THE 80 SERIES DATA FORMAT

With the ultrasonic output enabled, the meter transmits the displayed data via its internal speaker. Naturally, all other audio indicators are nonfunctional when the meter is in this mode. The data output is heard as a buzzing noise which occurs about twice every second, depending on the measurement mode of the meter. When in Ultrasonic Output mode, the measurement rate of the meter is reduced by about half. Each occurrence

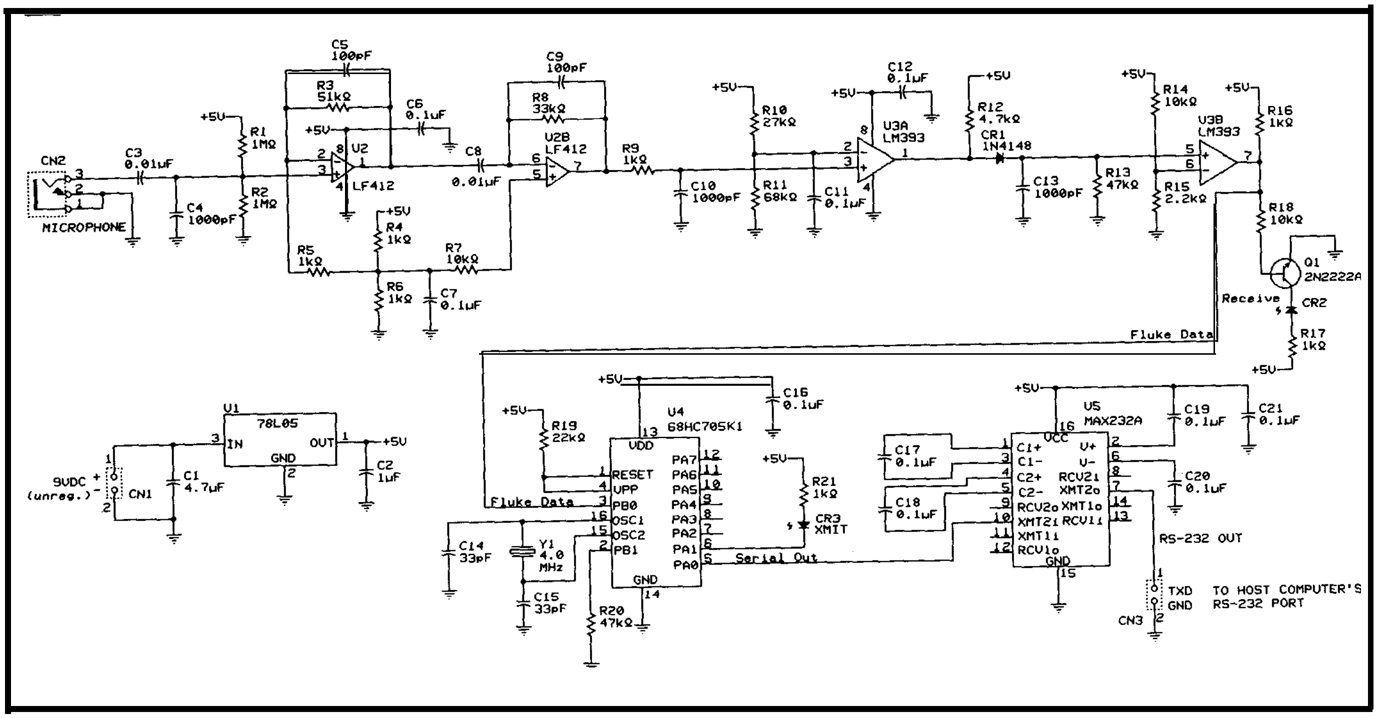


Figure 1—*By using* a single-chip microcontroller for the brains, the bulk of fhe Data Holster circuitry consists of the *ultrasonic* front end and *smoothing* circuits.AMAX232 *provides the RS-232* interface to allow *5-volt-only* operation. If the interface is plugged directly into the meter, the microphone front end *(U2) may be* eliminated.

of the buzzing sound represents a string of thirty-six nybbles that contain information about which segments and annunciators are active on the meter's display.

The output signal is modulated on a 16.6 kHz carrier. The bit time is about 1.3 ms. Each nybble is lead by a start bit (logical 1) and terminated by a stop bit (logical 0). This allows fairly accurate asynchronous detection because a word counter can be reset every time a start bit is received, thus eliminating timing error stack-up.

With the start bit, the stop bit, and four data bits each taking about 1.3 ms, the total time to transmit a single word is 7.8 ms. Since there are 36 words per data string, each string will require a total of about 280 ms of transmission time. The time between data strings varies with the meter's measurement mode, but it is always at least one word length. I used 11 ms (about 1.5 word lengths) to detect dead time between measurements.

Each string of data contains a three-word header and a one-word trailer, which leaves 32 words for actual measurement information. I'm not sure what the function of the header is, but the trailer word is a [illegible] for all previously transmitted words in the string. Perhaps the header could be used for synchronization in case the bit time were to drastically change.

Each bit of the 32 data words (128 bits total) indicates the status of a specific segment or annunciator on the meter's display. The bar-graph segments are included in the transmission. Unfortunately, the meters in the 80 series do not share a common data format, For example, the percent indicator (%) on the Fluke 85 translates into the Hertz (Hz) symbol on the Fluke 87. Clearly, the two are not interchangeable. Because of the ROM limitations of the 68HC705K1, I wrote separate software for each meter in the 80 series to address the differences in data format.

HARDWARE DESCRIPTION

In designing the hardware for this project, my major constraint was to build a circuit using common off-the-shelf components while minimizing the circuit's physical size. My ultimate goal was to build a self-contained unit which could easily attach to the meter with no electrical connection (see Photos 1 and 2). Additionally, I wanted to use a DC wall transformer as a power source. The size constraints and power source requirements lead me to choose the 16-pin Motorola 68HC705K1 for a microcontroller, and the 16-pin MAX232 for driving RS-232. In addition to their 16-pin footprints, these two devices require just a few external components to function. Luckily, I had the 68HC705K1 evaluation kit sitting idly in the lab...it was a processor looking for a project!

SIGNAL DETECTION

The key to signal detection in this project is the use of a suction cup microphone to pick up the meter's signal. This microphone, typically available at Radio Shack, is designed to be attached to a telephone handpiece for conversation recording. Its unidirectional characteristics inherently filter background acoustical noise and require the signal source to be in close proximity. However, this microphone can be excited by electrical noise from all directions. I'll discuss the effects of electrical noise on the detection circuit after describing the system hardware. For now, assume the microphone will only pick up acoustic signals.

The schematic for this project is shown in Figure 1. With the microphone attached to the back of the ing software description, I'll explain how I dealt with some of the Kl's limitations and how all of this functionality was crammed into 496 bytes.

Due to the Kl's small ROM space, it was necessary to select which of the meter's annunciators would be decoded and which would not. Since each numeral and each annunciator requires at least one software comparison and one ASCII character, some of the least-used annunciators had to bc eliminated. Of all possible numerals and annunciators that the meter is capable of displaying, I selected to decode and transmit the following:

*numerals 0 through 9 and decimal points
*polarity signs (+ and -)
•all available magnitude prefixes (u for micro, n for nano, M for mega, etc.)
•all available unit suffixes (V for volts, F for Farads, Hz for Hertz,etc.)
*the AC and DC annunciators

These arc probably sufficient for most applications. Among the annunciators which I had to eliminate are the 32-segment bargraph, the low-battery indicator, the beeper indicator, and the MIN-MAX-AVG-RECORD indicators. Most of the eliminated annunciators do not have significance pertaining to the actual measurement on the meter. After making these sacrifices, the code was ready to be written. See Figures 3a and 3b for the software flowchart.

The K1's program counter is vectored to address 0200 hex upon an external reset or power up. At this point, the ports are configured and the necessary variables are initialized. The program then loops until about 11 ms of dead time at port B bit 0 is detected. In other words, the program will not start acquiring a data record from the meter unless it can acquire the entire record rather than just a record fragment. Since the K1 executes an instruction every 500 ns (remember the 2-MHz internal clock), a 16-bit register is required to count 11 ms of dead time. The K1 lacks such a register, so I did the counting by incrementing the lower eight bits and, upon overflow, incrementing the upper eight bits. Although this task is fairly trivial, it consumes precious ROM space!

When about 11 ms of dead time has been detected, the program proceeds to acquire the 36 four-bit words from the meter. Each time a start bit is detected, a counter is reset and the software looks at port B for a 1.3-ms duration. The number of highs and lows detected during this time are stored in separate locations. This enables the software to "filter" noise which may be caused by the meter's deviation from the 1.3-ms bit time. At the end of the 1.3-ms bit time, the software checks to see if there were more highs than lows. If so, the bit is considered to be asserted.

If the most recently received bit is a start bit or a stop bit, it is ignored and the software waits until the next bit is received. Otherwise, the bit is rolled onto an 8-bit byte and stored in RAM for later processing. This process is repeated until all 36 words have been received. Although the process of converting 4-bit words into the 8-bit structure of the K1 is fairly straightforward, the required housekeeping activities eat up many bytes of ROM...and we haven't even started to decode the data yet!

After receiving the 36 words sent by the meter, the software runs an algorithm on the data and compares the result with the meter's checksum. If they do not match, the software loads the ASCII value for an exclamation point (21 hex) and sends it out to

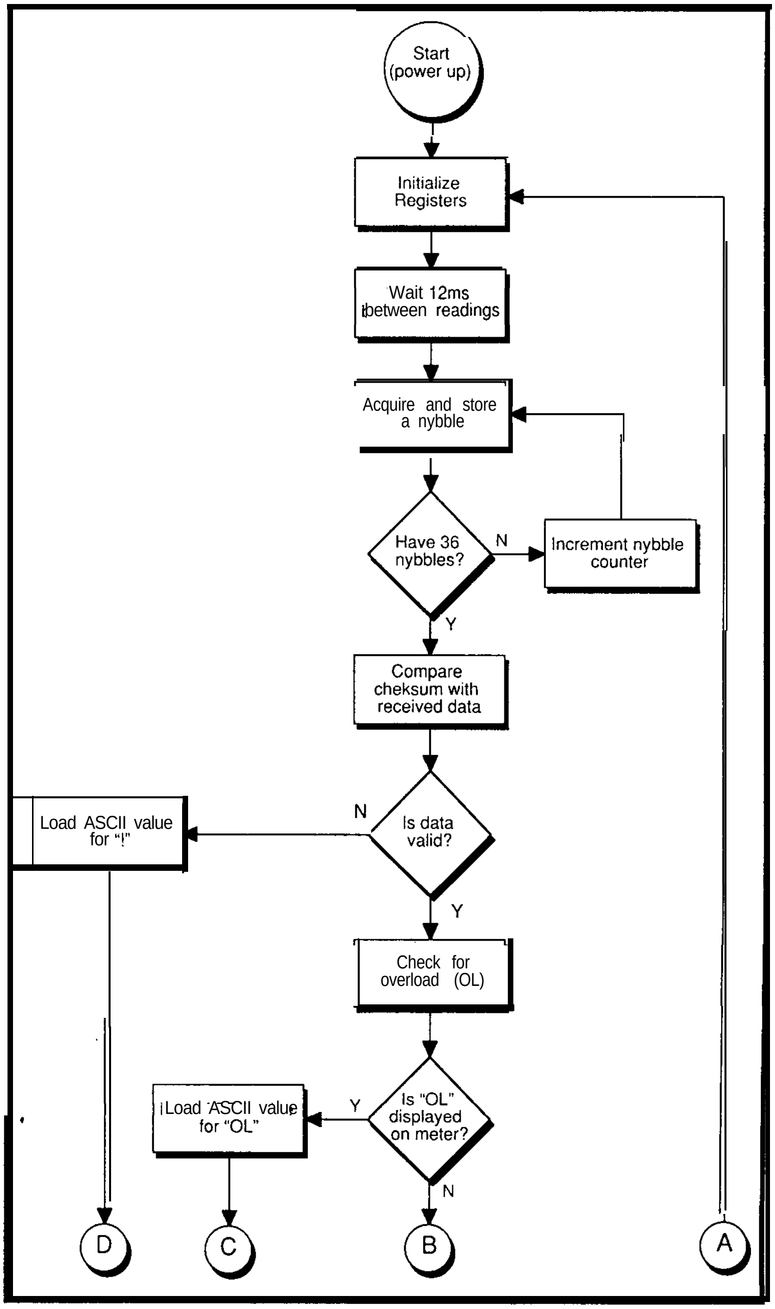


Figure 3—*The DataHolster* code takes up *all* 496 bytes of *ROM* and 32 bytes of RAM contained in fhe *MC68HC705K1*. Due *to* the limited nature of *the microcontroller, most things are done brute force* with straight-line code.

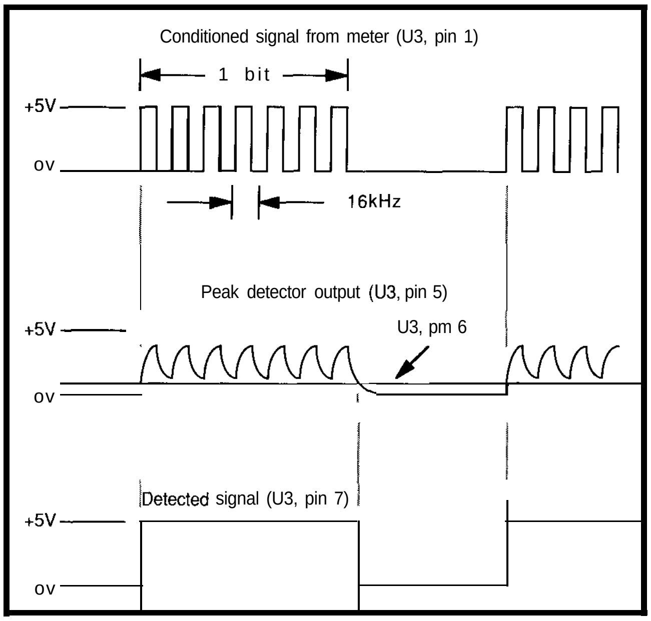
meter, the 16-kHz modulated signal is routed through CN2 to conditioning amplifier U2a. This amplifier detects changes in the pickup's impedance, eliminates any DC which may be present, and amplifies the signal. The amplified signal is then sent through another gain stage in U2b. The resistor divider formed by R4 and R6 provides a DC reference at $^{Vcc}/_2$ to the inputs of U2, allowing AC-coupled operation of the op-amps with a single +5-V power supply. Small capacitors in the feedback paths reduce the frequency response of the op-amps to prevent high-frequency oscillation. After the microphone's signal has been amplified, it is lightly filtered by R9 and C10.

At this point, the signal is approximately 1.5 volts (peak to peak) riding on the $^{Vcc}/_2$ offset. Comparator U3a is used to square-up the 16 kHz signal so that it swings between ground and Vcc. The "squared-up" signal then feeds the peak detector formed by CR1, R13, and C13. The cutoff frequency of this peak detector is set to about 21 kHz by Cl3 and R13 while the peak voltage is limited by the divider formed by R12 and R13. This ensures that the peak detector will never charge to +5 V, but will discharge fast enough to prevent stretching out the bit time. Figure 2 shows the critical signals in a typical bit detection sequence.

The output of the peak detector is "squared-up" by U3b to 0–5-volt levels. The detected signal is then fed into port B, bit 0, of the 68HC705K1. Transistor Q1 lights CR2 every time a bit is detected. I added this to the design for troubleshooting and visual effect purposes.

MICROCONTROLLER HARDWARE

The 68HC705K1, herein after referred to as "K1," is an extremely simple 8-bit microcontroller. With its built-in RAM, ROM, and ports, it requires very few external components to function. Namely a crystal, some capacitors, and a few resistors.



Conditioned signal from meter (U3, pin 1)
1 bit
+5V
ov
16kHz
Peak detector output (U3, pin 5)
+5V
U3, pm 6
ov
Detected signal (U3, pin 7)
+5V
ov

figure 2—*The signal from the meter is first squared up, then passed through a peak detector. The final output is then used bv the microcontroller to decode the data transmission.*

The crystal frequency of the K1 is 4 MHz. The K1 divides this frequency by two internally for a resulting operating frequency of 2 MHz (500 ns per cycle). Interrupts (either software or hardware) are not used in this project.

The K1 reads the detected signal at port B bit 0, decodes it, and translates the decoded data into ASCII. The ASCII data is then placed on port A bit 0 via a software shift register at 9600 bps, 8 data bits, one stop bit, no parity. The serial data is then passed to U5 [the MAX232) for level shifting. The LED on port A bit 1 flashes every time a data string is sent out.

ADDITIONAL HARDWARE NOTES

As I mentioned earlier, the suction-cup microphone can be excited by electrical sources. Although I do not have complete data on the microphone (it came from Radio Shack), I assumed that its internal construction is similar to that of a loop antenna. This presents a problem when the microphone is in close proximity to video monitors, computers, or other strong sources of EMI. To eliminate this, I have experimented with adding filters (both active and passive) to the front end of the detection circuitry. This turns out to be a fairly difficult task because of the sharp rolloffs required to attenuate all noise sources while only passing the 16-kHz carrier. A bandpass filter centered around 16 kHz with attenuation of maybe 40 or 50 dB at 15 and 17 kHz would probably eliminate most of the electrical interference. However, I don't believe these specifications can be reasonably met without using DSP techniques. Whether analog or digital, an effective filter would add a significant amount of complexity to the system.

A simpler way to eliminate noise problems is to hardwire the meter's speaker output to the decoding circuit. To use a hardwire connection, you must install a connector in parallel with the meter's speaker. Casual users will probably not want to modify their meters in this way, but it is a must for serious users. I used a $^3/_{32}$″ phone jack mounted just above the V+ jack on the meter (refer to Photo 1b). I wired my meter such that when a $^3/_{32}$″ phone plug is inserted into the jack, the internal speaker of the meter is disabled. This makes for silent and reliable operation (the continuous audio information emitted by the meter can become quite annoying after a while).

If a hardwire connection is used, the microphone front end of the decoding circuit (U2) can be eliminated. However, it is important to electrically isolate the meter from the decoding circuit to protect the circuit in case the meter fails to provide isolation between its input jacks and the speaker. I used a pulse transformer in a DIP package for this purpose. The output of the pulse transformer can be connected directly between ground and U3 pin 3. In doing this, be sure to protect the input of U3 with diodes because the transformer's output will swing below ground. Also, the threshold voltage at U3 pin 2 may have to be adjusted depending on the turns ratio of your transformer.

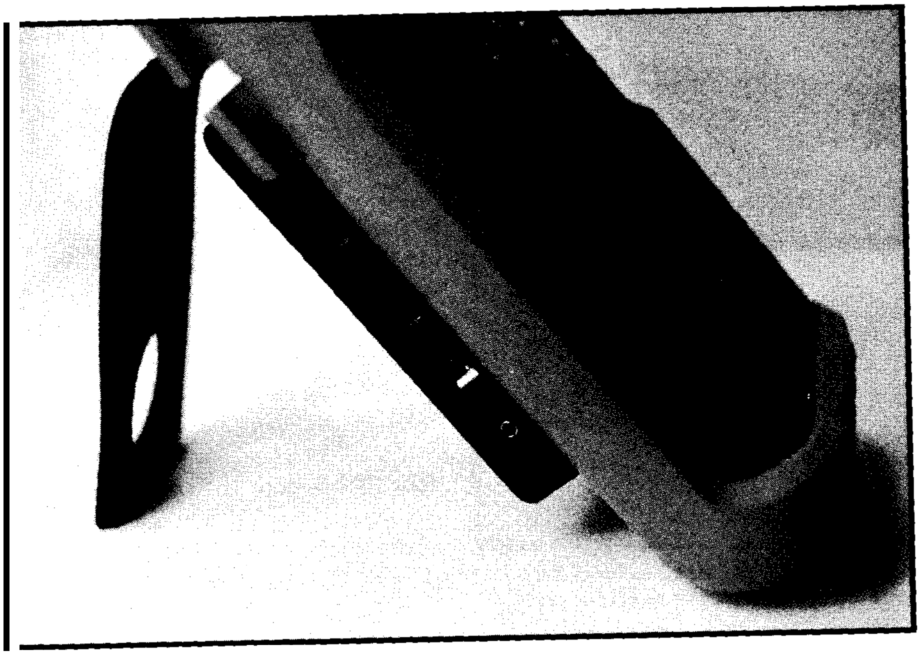


Photo 1a—*The DataHolster fits* comfortably **behind the** *yellow shockcase of the 80-series meters with no electrical connection to the meter. From top to bottom* along the side of the DataHolster are the receive indicator, *transmit* **indicator,** power indicator, *power input jack, and RS-232* output *jack*

The easiest way to eliminate noise problems is to move the meter and decoding circuit away from the suspected noise source. If the microphone is swamped with noise, the LEDs (CR1 and CR2) will not flash because the peak detector is probably pegged high. The decoding software in the K1 uses the meter's checksum to detect intermittent noise. If a bit gets corrupted during transmission, the software will transmit an exclamation point to the host computer to indicate that an error has occurred.

On my initial prototype, I used the ±10-V supplies generated by the MAX232 to power the LF412. At the time, it seemed like an elegant (tricky) way to provide a dual rail supply to the op-amp. This scheme worked quite well until I replaced the Maxim part with one of its alternate sources. As luck would have it, the source impedance of the non-Maxim ±10-V supply

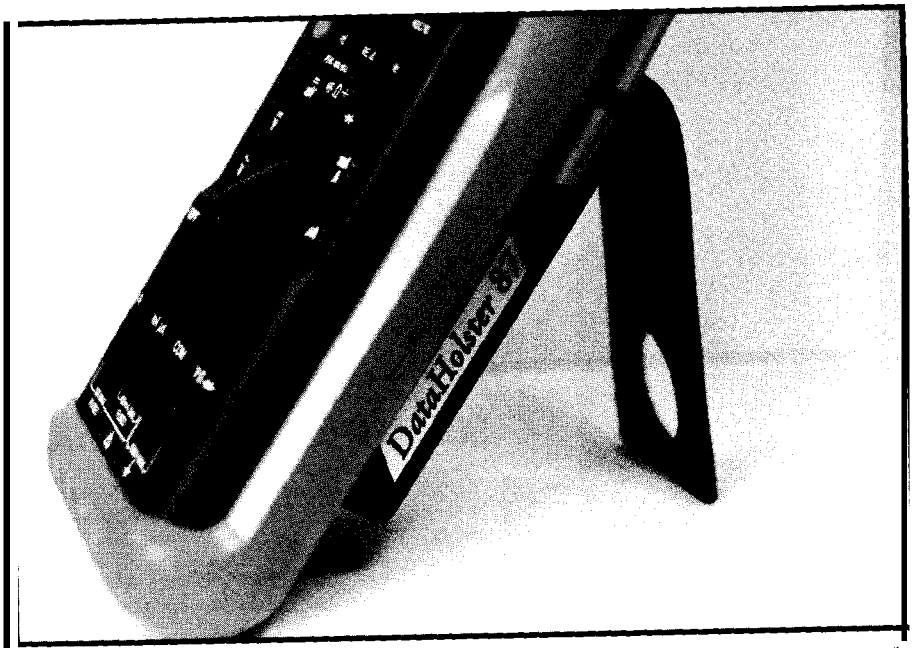
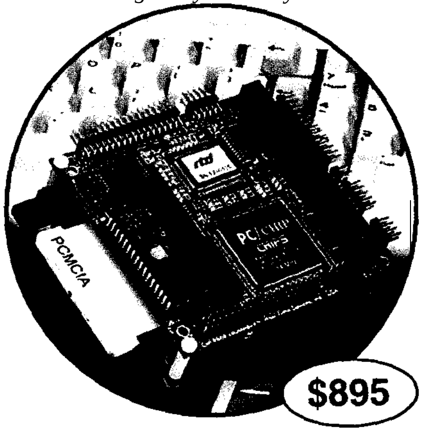


Photo 1b—*If you decide that the* ultrasonic connection is too unreliable or *annoying, you may elect to install a* $\frac{3}{32}$" *jack on the front of the meter to allow direct* **connection of the** DataHolster. *The* **jack can be Seen** *just* **above**



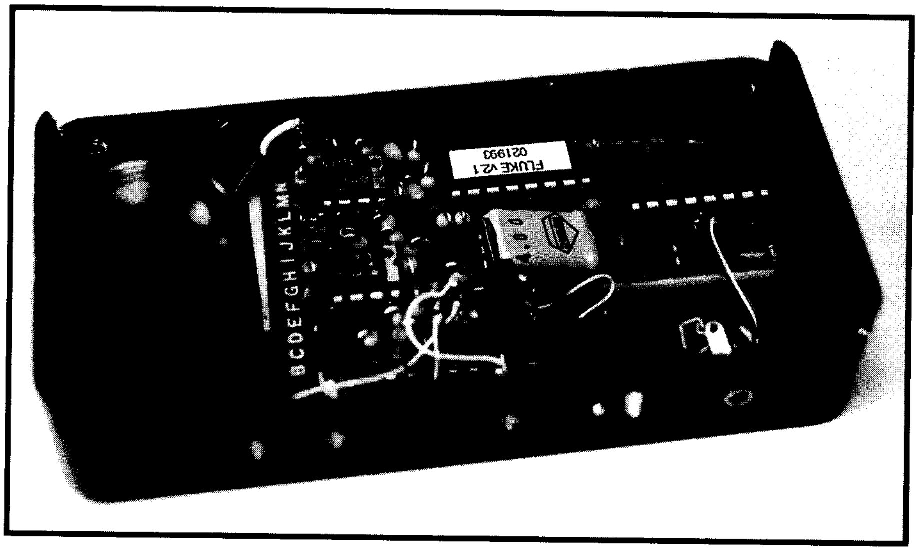


Photo 2—*A view of the inside reveals compact and neat internal construction of the DataHolster. The suction-cup microphone is contained inside the unit (on the left side) and picks up the signal through the back of the meter.*

is too high to reliably power the op-amp. Hence, I abandoned this idea and redesigned the system so that all components use the +5-V supply. The moral of this story is to always test parts from alternate sources, especially if you're doing "tricks" with the components.

THE SOFTWARE

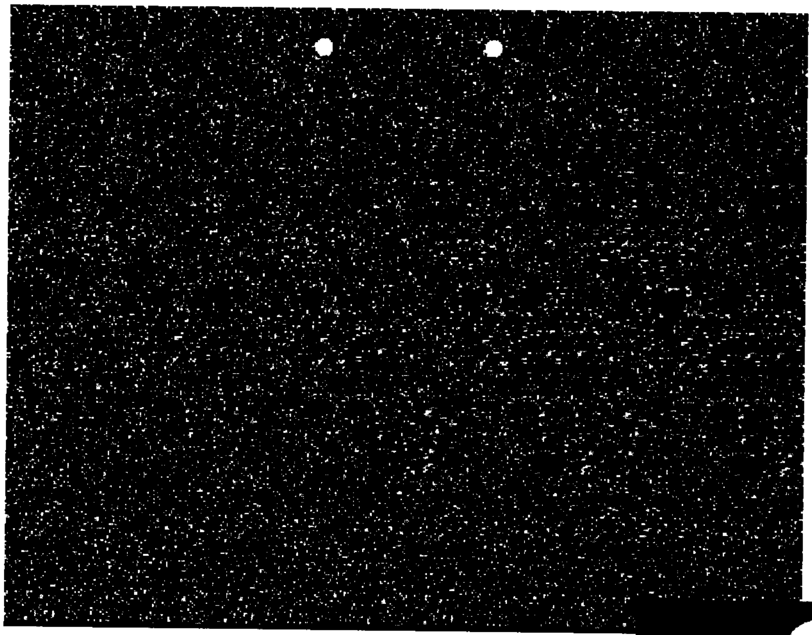
The software for this project is, in general, pretty straightforward. The required tasks can be broken down into four major segments:

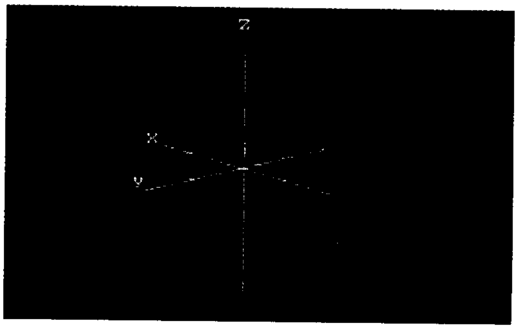
- asynchronously acquire the data from the meter
- verify the checksum
- process (decode) the data
- serially transmit the processed data to the host computer

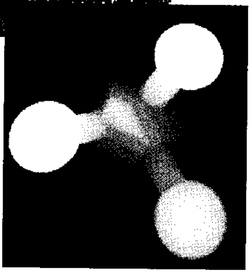
The major software challenge lies in accomplishing all of these tasks in the K1's 496 bytes of program ROM. For all intents and purposes, this precludes writing code in anything but assembly language. In addition to the small ROM, the following limitations must be considered:

- The K1 has only 32 bytes of RAM, some of which is used for storing the stack.
- The K1 does not have any 16-bit registers.
- The K1 has only two 8-bit registers (accumulator and X).
- The K1's internal counter cannot be reset by software.
- The K1 has no built-in serial port.

This version of the software uses all 496 bytes of program ROM and all of the 32 bytes of RAM! In the follow-







ing software description, I'll explain how I dealt with some of the Kl's limitations and how all of this functionality was crammed into 496 bytes.

Due to the Kl's small ROM space, it was necessary to select which of the meter's annunciators would be decoded and which would not. Since each numeral and each annunciator requires at least one software comparison and one ASCII character, some of the least-used annunciators had to be eliminated. Of all possible numerals and annunciators that the meter is capable of displaying, I selected to decode and transmit the following:

*numerals 0 through 9 and decimal points
*polarity signs (+ and -)
•all available magnitude prefixes (u for micro, n for nano, M for mega, etc.)
•all available unit suffixes (V for volts, F for Farads, Hz for Hertz,etc.)
•the AC and DC annunciators

These are probably sufficient for most applications. Among the annunciators which I had to eliminate are the 32-segment bargraph, the low-battery indicator, the beeper indicator, and the MIN-MAX-AVG-RECORD indicators. Most of the eliminated annunciators do not have significance pertaining to the actual measurement on the meter. After making these sacrifices, the code was ready to be written. See Figures 3a and 3b for the software flowchart.

The Kl's program counter is vectored to address 0200 hex upon an external reset or power up. At this point, the ports are configured and the necessary variables are initialized. The program then loops until about 11 ms of dead time at port B bit 0 is detected. In other words, the program will not start acquiring a data record from the meter unless it can acquire the entire record rather than just a record fragment. Since the K1 executes an instruction every 500 ns (remember the 2-MHz internal clock), a 16-bit register is required to count 11 ms of dead time. The K1 lacks such a register, so I did the counting by incrementing the lower eight bits and, upon overflow, incrementing the upper eight bits. Although this task is fairly trivial, it consumes precious ROM space!

When about 11 ms of dead time has been detected, the program proceeds to acquire the 36 four-bit words from the meter. Each time a start bit is detected, a counter is reset and the software looks at port B for a 1.3-ms duration. The number of highs and lows detected during this time are stored in separate locations. This enables the software to "filter" noise which may be caused by the meter's deviation from the 1.3-ms bit time. At the end of the 1.3-ms bit time, the software checks to see if there were more highs than lows. If so, the bit is considered to be asserted.

If the most recently received bit is a start bit or a stop bit, it is ignored and the software waits until the next bit is received. Otherwise, the bit is rolled onto an 8-bit byte and stored in RAM for later processing. This process is repeated until all 36 words have been received. Although the process of converting 4-bit words into the 8-bit structure of the Kl is fairly straightforward, the required housekeeping activities eat up many bytes of ROM...and we haven't even started to decode the data yet!

After receiving the 36 words sent by the meter, the software runs an algorithm on the data and compares the result with the meter's checksum. If they do not match, the software loads the ASCII value for an exclamation point (2 1 hex) and sends it out to

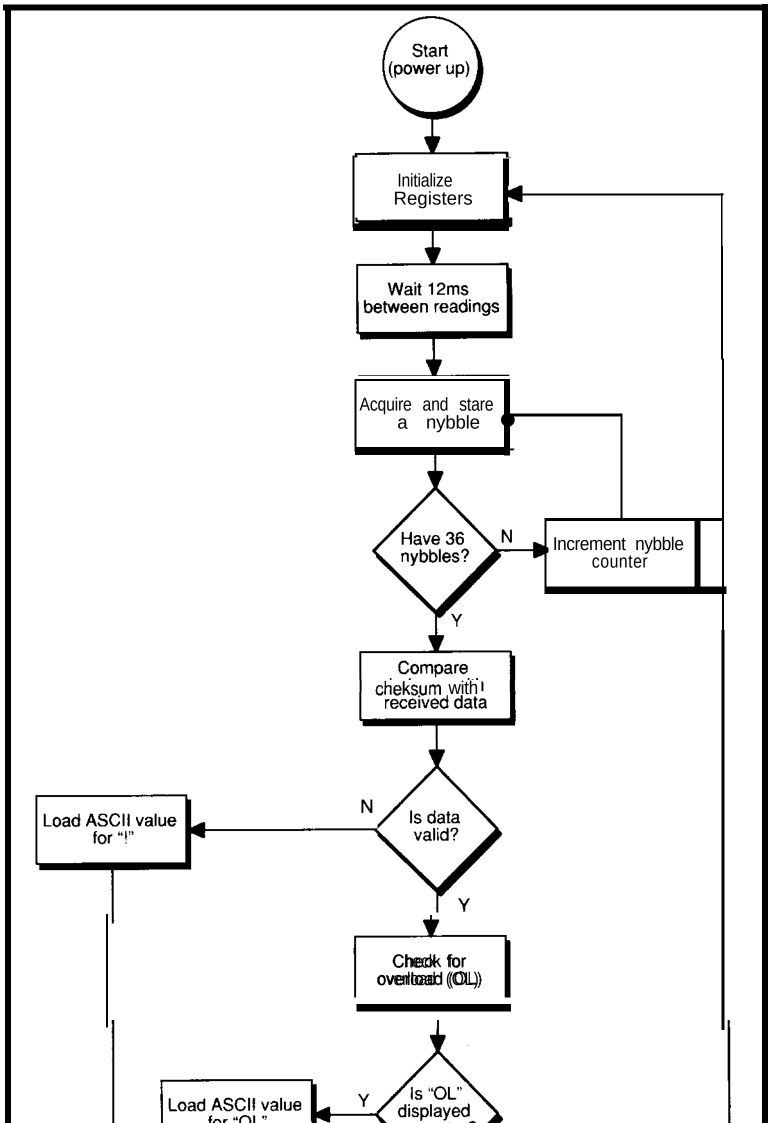


Figure 3—*The DataHolster* code takes up *all* 496 bytes of ROM and 32 bytes *of RAM* contained in the *MC68HC705K1*. Due to the limited nature of the microcontroller, most things are done brute force with straight-line code.

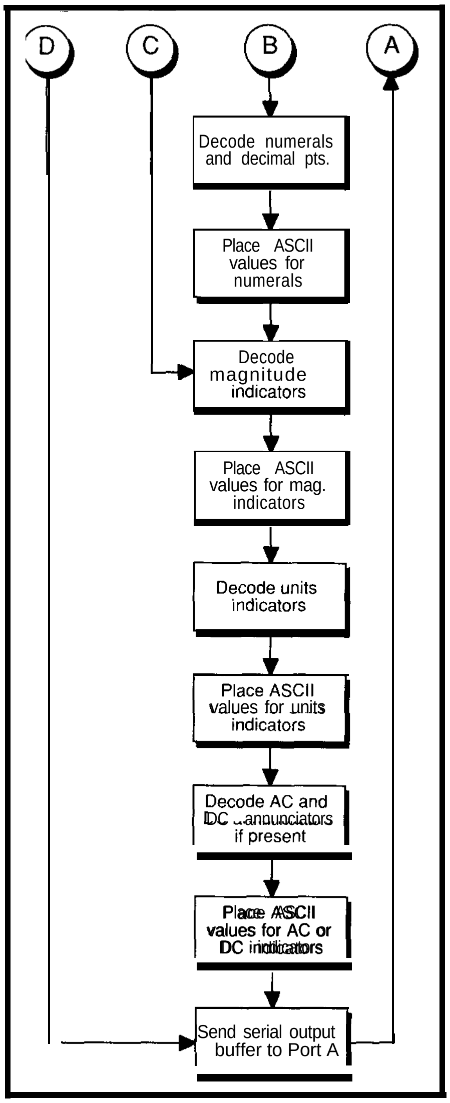

D
C
B
A
Decode numerals and decimal pts.
Place ASCII values for numerals
Decode magnitude indicators
Place ASCII values for mag. indicators
Decode units indicators
Place ASCII values for units indicators
Decode AC and DC annunciators if present
Place ASCII values for AC or DC indicators
Send serial output buffer to Port A

Figure 3—*continued*

the host computer. I could not use the word "error" because it requires 5 bytes of ASCII and some overhead to transmit, so I used the exclamation point instead.

If the checksum algorithm determines that the data is valid, the software proceeds to decode the meter's display. It begins the decoding by checking for an overload condition on the meter. This condition is indicated by the letters "OL" on the meter's display. If the software detects the seven-segment signature of "OL," it will not try to decode the polarity sign or any numerals.

The software decodes the data in the same order that you would read the meter...from left to right. The polarity sign is decoded, followed by the ten-thousands digit, followed by the thousands digit, followed by the hundreds digit, and so forth. After decoding the polarity and the numerals, the software looks for magnitude indicators and units indicators. Finally, the "AC" and "DC" annunciators are decoded. This decoding method allows the appropriate ASCII values to be placed in the serial output buffer in the order in which they are to be transmitted. So, a reading such as "-1.234mVDC" will have the ASCII value for a minus sign at the top of the serial buffer and the letter "C" at the bottom. This makes the serial output routine fairly simple.

After decoding all of the data and storing the appropriate ASCII values in the serial output section of RAM, the software enters the serial output routine. This routine simply loads a character from the serial output section of RAM and shifts it, bit by bit, into port A. When the bit shifting is complete, a stop bit is sent and the process repeats until all characters in the serial output section of RAM have been sent. The serial output routine sends a carriage return after the last character is sent. When the RS-232 transmission is complete, the program loops back to the beginning and starts over as if power were just applied to the K1.

Embedded within the software are a couple of neat tricks which I used to reduce the program size while maintaining full functionality. Unfortunately, they are too detailed to explain here. A complete listing of the Fluke 83 version of the software, including comments, is available on the Circuit Cellar BBS.

APPLICATIONS

The applications for this project are numerous because the Fluke 80 series is an extremely versatile (and popular) hand-held multimeter capable of making true RMS measurements. This project, affectionately named the "DataHolster," can be used as an automated data logger for environmental measurements such as temperature (with a thermal probe], humidity, motor RPM modulation, long-term power line monitoring, or any other slow-moving signal. By using a computer with RS-232 communications software, the data from the meter can be captured as a function of time. This data can be stored for off-line postprocessing in a spreadsheet, graphing program, or database.

CONCLUSIONS

This project illustrates how to use the little-known ultrasonic output feature of the Fluke 80 series for data acquisition. Since the introduction of the 80 series, many hand-held meters have been introduced which sport some kind of computer output capability. However, the measurement capabilities, accuracy, and reliability of these other meters may not be on par with that of the 80 series. Using the Fluke 80 series with a "DataHolster," you get the best of both worlds...a first-class, time proven, true RMS, hand-held multimeter, and data acquisition capabilities! ❑

Derek Matsunaga holds a BSEE and is currently a product engineer for a medical device design/manufacturing company. His interests include circuit design and analysis, product development, and signal processing. He may be contacted at the address below OK on the Circuit Cellar BBS.

SOURCE

A complete technical description of the Fluke 80 series data format with drawings and examples may be ordered by sending a check or money order for $9.95 (U.S.) to:

Derek Matsunaga
P.O. Box 19615
Boulder, CO 80308-26 15

A hardcopy of the commented sourcecode will also be included. A limited quantity of programmed and tested 68HC705K1Ps for this project is available. If interested, please inquire when ordering technical information.

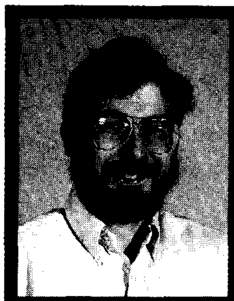
IRS

407 Very Useful
408 Moderately Useful
409 Not Useful

DEPARTMENTS

IDs and LCDs

The '386SX Gets Positive Identification and a Small Display



Just when you thought you were safe from hardware keys and locks, Ed adds a clever little serial number chip to the Firmware Development Board. Also in the hopper is a small LCD display to impress your friends.

FIRMWARE FURNACE

Ed Nisley

Some years ago the PC press pushed hardware serial numbers so software could be keyed to a single PC. The predictions were that software piracy would die overnight, profits shoot up, prices drop down, and all would be right with the world.

Another prognostication gone awry.

At the moment I suppose I'm one of those software pirates, because I have two PCs: a dying Model 80 from which I'm transferring files to my new system. If the PCs had unique serial numbers I'd be out of luck even though I've got only one pair of hands on one keyboard at any one time...and the Model 80 can no longer run the programs.

Actually, it gets worse. The first new system didn't quite work, so this is actually my second attempt to transfer the files. It's never been clear what the software transfer fee might be in this situation, but I'm sure paying it twice wouldn't improve my disposition at all.

Nevertheless, this month I'll describe how to give your Firmware Development Board a unique ID based on the Dallas DS2400 Silicon Serial Number. If you combine the (almost trivial) hardware with the BIOS extensions from last month's column, you can produce a system that won't even boot with an incorrect serial number, let alone run a program. Just don't try to charge me for your code, OK?

We'll also add a small, character-LCD panel to display status messages without a terminal or video monitor. If

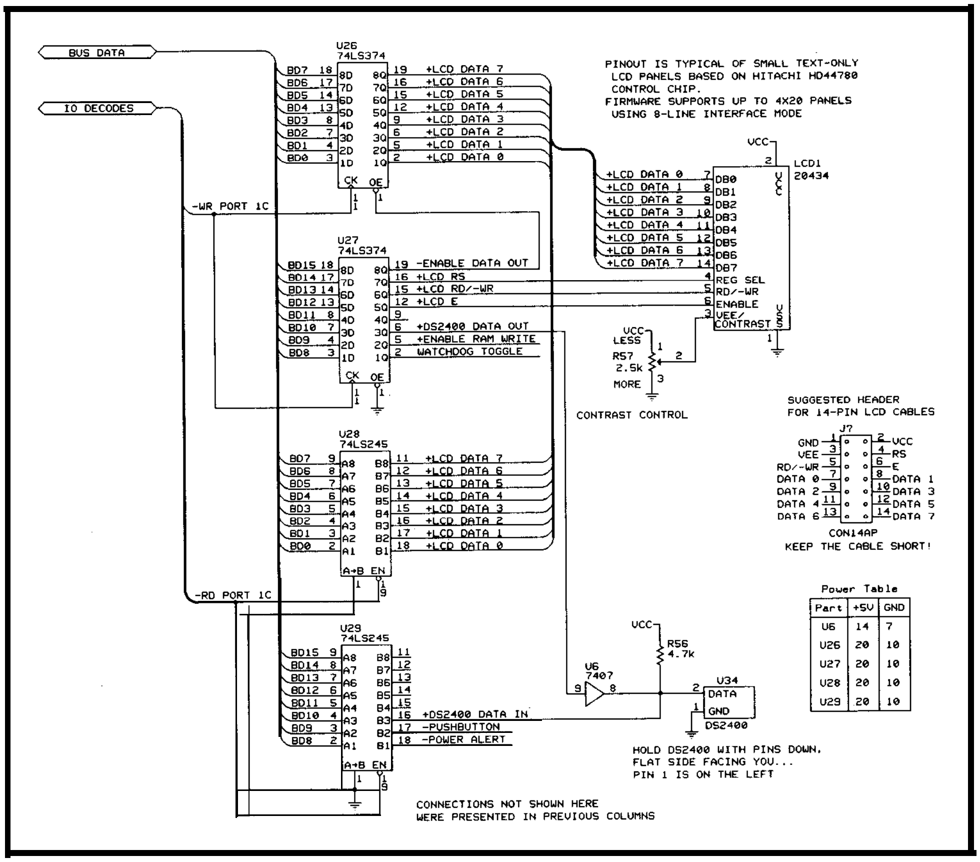

BUS DATA
IO DECODES
U26
74LS374
BD7 18 8D 8Q 19 +LCD DATA 7
BD6 17 7D 7Q 16 +LCD DATA 6
BD5 14 6D 6Q 15 +LCD DATA 5
BD4 13 5D 5Q 12 +LCD DATA 4
BD3 8 4D 4Q 9 +LCD DATA 3
BD2 7 3D 3Q 6 +LCD DATA 2
BD1 4 2D 2Q 5 +LCD DATA 1
BD0 3 1D 1Q 2 +LCD DATA 0
CK OE
-WR PORT 1C
U27
74LS374
BD15 18 19 -ENABLE DATA OUT
BD14 17 16 +LCD RS
BD13 14 15 +LCD RD/-WR
BD12 13 12 +LCD E
BD11 8 9
BD10 7 6 +DS2400 DATA OUT
BD9 4 5 +ENABLE RAM WRITE
BD8 3 2 WATCHDOG TOGGLE
PINOUT IS TYPICAL OF SMALL TEXT-ONLY
LCD PANELS BASED ON HITACHI HD44780
CONTROL CHIP.
FIRMWARE SUPPORTS UP TO 4X20 PANELS
USING 8-LINE INTERFACE MODE
VCC
LCD1
20434
DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7
REG SEL
RD/-WR
ENABLE
VEE/
CONTRAST
VCC
LESS
R57
2.5k
MORE
CONTRAST CONTROL
SUGGESTED HEADER
FOR 14-PIN LCD CABLES
J7
GND VCC
VEE RS
RD/-WR E
DATA 0 DATA 1
DATA 2 DATA 3
DATA 4 DATA 5
DATA 6 DATA 7
CON14AP
KEEP THE CABLE SHORT!
U28
74LS245
BD7 9 A8 B8 11 +LCD DATA 7
BD6 8 A7 B7 12 +LCD DATA 6
BD5 7 A6 B6 13 +LCD DATA 5
BD4 6 A5 B5 14 +LCD DATA 4
BD3 5 A4 B4 15 +LCD DATA 3
BD2 4 A3 B3 16 +LCD DATA 2
BD1 3 A2 B2 17 +LCD DATA 1
BD0 2 A1 B1 18 +LCD DATA 0
A→B EN
-RD PORT 1C
Power Table
Part +5V GND
U6 14 7
U26 20 10
U27 20 10
U28 20 10
U29 20 10
U29
74LS245
BD15 9 11
BD14 8 12
BD13 7 13
BD12 6 14
BD11 5 15
BD10 4 16 +DS2400 DATA IN
BD9 3 17 -PUSHBUTTON
BD8 2 18 -POWER ALERT
VCC
R56
4.7k
U6
7407
U34
DATA
GND
DS2400
HOLD DS2400 WITH PINS DOWN,
FLAT SIDE FACING YOU...
PIN 1 IS ON THE LEFT
CONNECTIONS NOT SHOWN HERE
WERE PRESENTED IN PREVIOUS COLUMNS

Figure l--The *DS2400* uses a single *open-collector* output bit *and one input bit, but the LCD pane/ needs a* *full* *byte of* data *and several control lines. All of the pulse* timings *are* produced *in firmware, so these* devices *are not* connected *directly* to *the PC's ISA bus connector. Previous columns have presented the* *support* *circuitry behind these ICs.*

naught else, it can point out a wrong serial number...or perhaps show an unchanging "33" so everyone knows your PC's clock rate.

The two devices require completely different interfaces: the DS2400 is bit-serial, while the LCD uses a byte-wide parallel port. We'll use the Firmware Development Board's 8254 timer chip to provide CPU-independent timings, plus a dollop of test code to make sure everything is working.

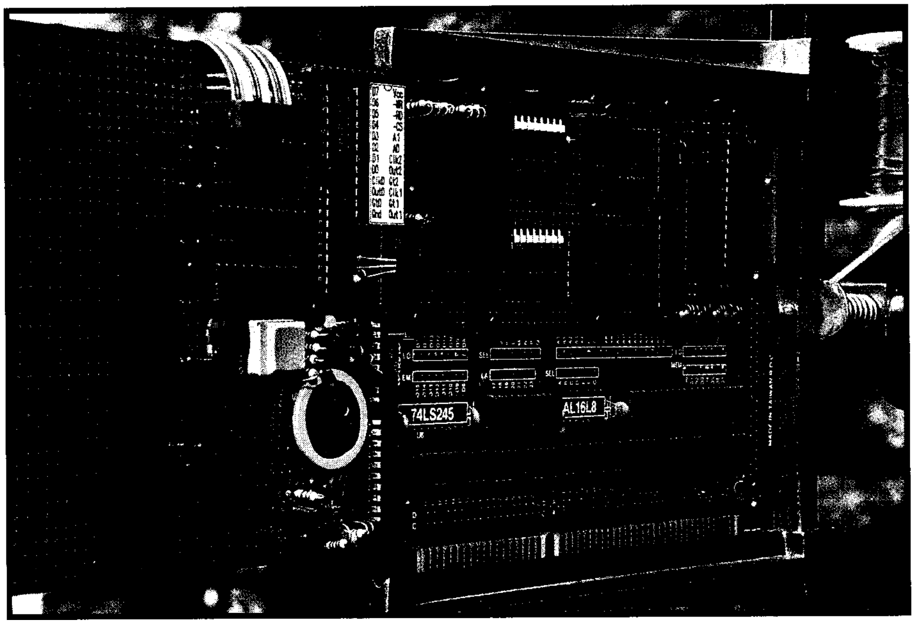


Photo l--The *DS2400* looks *like a* plastic *transistor just below the 8254, while the LCD port connects to the ribbon cab/e going off the top of the board.*

SIMPLE CIRCUITRY

The DS2400 and character LCD interfaces require very little additional hardware, as you can see in Figure 1. As before, I've omitted the ISA bus connections and support chips described in previous columns, so you'll need to refer back a few issues to get the complete details.

Photo 1 shows the Firmware Development Board as it looks today. The DS2400 looks like a plastic transistor in a TO-92 case just below the 8254, while the LCD port connects to the ribbon cable snaking from the top of the picture. The lithium cell, MAX691, and the clump of discrete parts provide backup power for the SRAM chip located in the upper right comer of the board.

The DS2400 claims to use a one-wire interface. You need a ground connection, of course, but one signal wire provides bidirectional data and timing. Both the 7407 open-collector driver and the DS2400 can pull the wire to ground, while the 4.7k pullup resistor supplies the logic-high state when the driver transistors are off.

As you might expect from a CMOS part, the DS2400 does not use TTL voltage levels. Its V_{IH} rating is 3.0 volts, which is well above the 2.4 volt V_{OH} of normal TTL. The part is always used with an open-collector driver and pullup, so this has no effect on most circuits, but keep it in mind if you're trying something truly bizarre.

The LCD interface has more wires and a contrast-control trimpot, but it's fairly simple. In fact, you can probably connect the V_{EE} terminal to ground for acceptable contrast without a pot. If your LCD panel requires a negative bias voltage, connect the pot between the +5 and -12 power supplies. You should increase the pot to about 10k to reduce its power dissipation or, better yet, follow your LCD's data sheet recommendations.

A perennial BBS question involves interfacing these little LCD panels directly to a microcontroller bus. The

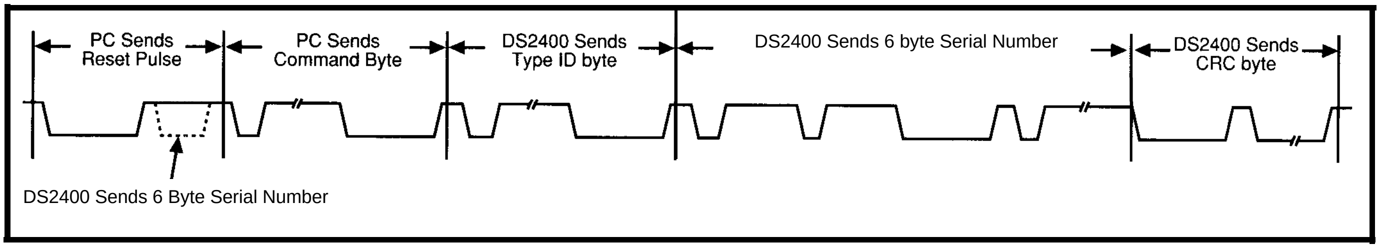

PC Sends Reset Pulse
PC Sends Command Byte
DS2400 Sends Type ID byte
DS2400 Sends 6 byte Serial Number
DS2400 Sends CRC byte
DS2400 Sends 6 Byte Serial Number

Figure P-Reading a *DS2400* requires a reset pulse *followed by* 72 time *slots for the bidirectional* data *flow. Each half of the* *reset* *pulse is at least* 480 *µs and the* *data* time *slots are between 60* and *120 µs* each *with* a mandatory *1-µs delay* *after* *each bit. A complete* transaction *thus* occupies less than 10 ms.

Hitachi HD44780 LCD controller is really designed for the 6800 family bus, so it takes a bit of hocus-pocus to adapt it to an Intel bus. Rather than confront that issue, I elected to use simple port I/O and be done with it. If you need just an LCD panel without all the rest of the Firmware Development Board, the references I mentioned in Issue 3 1 should give you a good head start on your bus interface design.

Be careful when you wire up the LCD's '245 and '374 data lines. It's quite easy to get one group "backwards" and produce some truly baffling bugs. The LCD software includes a counting sequence that should help pin down that problem, but it's better to avoid it entirely. Note that the '374 driving the LCD's data lines has its Output Enable line controlled by the high-order bit from the other '374 so your code can shut it off when reading data from the LCD.

With the hardware out of the way, on to the code....

BIDIRECTIONAL BIT BANGING

Each Dallas Semiconductor DS2400 Silicon Serial Number chip is laser-personalized during production so that, unlike ordinary ICs, every one is unique. The data includes an 8-bit type number, a 48-bit serial number, and an 8-bit cyclic redundancy check (CRC) value to verify the data.

Most ICs transmit and receive data using at least two wires, one of which is a clock defining when the other is valid. The DS2400 takes a minimalist approach: all information is transmitted by pulse-width-modulated blips on a single wire. Using PWM simplifies the hardware, but requires moderately complex firmware. Fortunately, firmware is cheap once you've figured out how to make it work and costs nothing to reproduce, so it's a fair tradeoff.

Incidentally, the DS2400 is ideally suited to the 8051 CPU's bidirectional I/O pins. We need a 7407 open-collector driver on the Firmware Development Board, but an 805 1 system can get along with just a wire. You can't get any cheaper than that! (Well, you can if you embed the serial number in the CPU, but that's another topic.)

The DS2400 timing specs are particularly critical because it does not have a power connection: its CMOS circuitry runs from charge stored on a tiny internal capacitor while the signal line is high. The timing specs ensure that the capacitor will not discharge during a transmission, so you must be careful to observe the minimum and maximum pulse width limits. The CRC will tell you if you received bad data, but it's better to design the interface correctly than depend on happenstance.

Figure 2 outlines a complete transaction: the PC resets the DS2400, sends a command word, and then clocks the type identifier, serial number, and CRC bits from the chip, The initial reset pulse discharges the internal capacitor and clears the DS2400's circuitry to ensure each

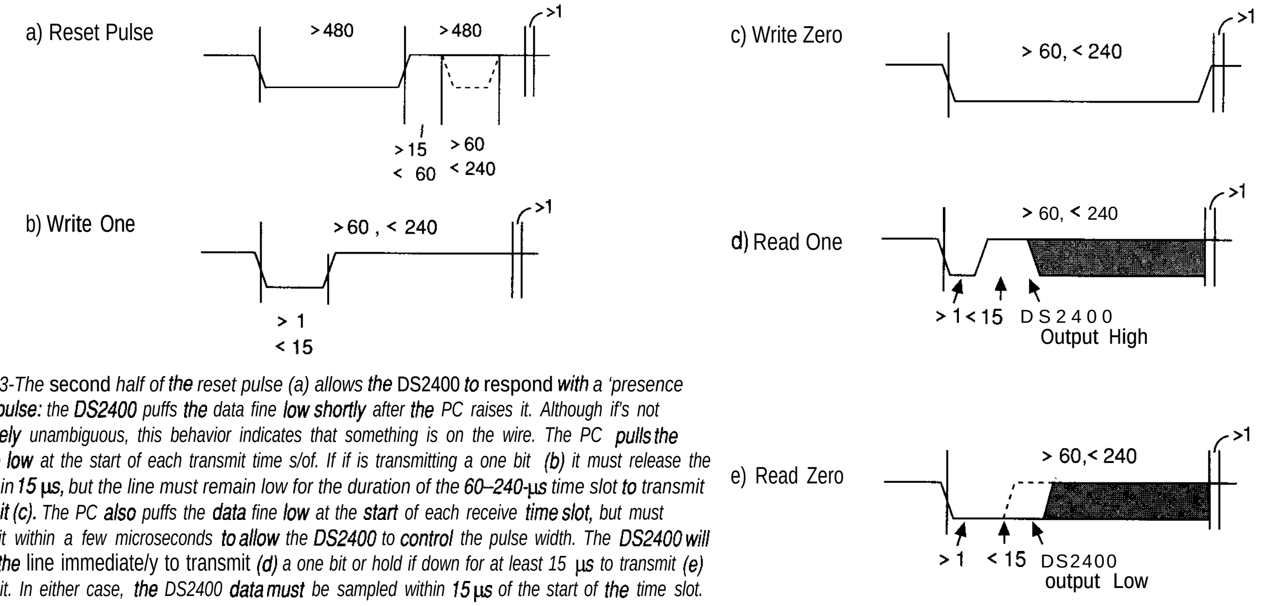


Figure *3-The* second *half of* *the* *reset pulse (a) allows* *the* DS2400 *to* respond *with* *a 'presence* *detect" pulse:* *the* *DS2400* *puffs* *the* *data fine* *low shortly* *after* *the* *PC raises it. Although if's not* *completely* *unambiguous, this behavior indicates that something is on the wire. The PC* *pulls the* *data* *line* *low* *at the start of each transmit time s/of. If if is transmitting a one bit* *(b)* *it must release the line within* *15 µs*, *but the line must remain low for the duration of the* *60–240-µs* *time slot* *to* *transmit a zero* *bit (c)*. *The PC* *also* *puffs the* *data* *fine* *low* *at the* *start* *of each receive* *time slot*, *but must release it within a few microseconds* *to allow* *the* *DS2400* *to* *control* *the pulse width. The* *DS2400 will* *release* *the* line immediate/y to transmit *(d)* *a one bit or hold if down for at least 15* *µs* *to transmit* *(e)* *a zero bit. In either case,* *the* *DS2400* *data must* *be sampled within* *15 µs* *of the start of* *the* *time slot.*

Listing I--The PC and *DS2400* communicate *by* means *of pulse-width-modulated signals. This routine produces low and high pulses without depending on the CPU clock rate. It a/so enforces a minimum 1-µs idle time between each pair of pulses. Because read and write signals are identical, this code samples the data line shortly after the rising edge; the caller decides whether the bit is useful or not.*

```
int PulseDS2400(LowTime,HighTime)
WORD LowTime;
WORD HighTime;


WORD TestValue;
int RetValue;

      TestValue               /* keep compiler happy */
      HighTime;
      LowTime;

asm {
      CL1                     ; first pulse needs precise timing

      MOV  DX,#SYNC_ADDR_A    ; show sync on printer port
      IN   AL,DX
      OR   AL,#$02
      OUT  DX,AL

      MOV  DX,#I8254_BASE_A+6; set, mode 0, 16-bit timer, chan
      MOV  AL,#$B0
      OUT  DX,AL

      MOV  DX,#I8254_BASE_A+(DS_CTR*2); set low time
      MOV  AX,6[BP]
      OUT  DX,AL
      XCHG AH,AL
      OUT  DX,AL

      MOV  DX,#CTLS_ADDR_A              ; se DS2400 data bit low
      MOV  AX,#0
      OUT  DX,AX

      CALL WaitTimer

      MOV  DX,#I8254_BASE_A+(DS_CTR*2) ; se read delay interval
      MOV  AX,#TIME_RD
      OUT  DX,AL
      XCHG AH,AL
      OUT  DX,AL

      MOV  DX,#CTLS_ADDR_A ; set DS2400 data bit hiqh=floatiny
      MOV  AX,#DS2400OUT_A
      OUT  DX,AX

      CALL WaitTimer

      MOV  DX,#STAT_ADDR_A    ; read output from DS2400
      IN   AX,DX
      AND  AX,#DS2400IN_A     ; isolate the data bit
      MOV  -2[BP],AX

      MOV  DX,#I8254_BASE_A+(DS_CTR*2); set rest of high time
      MOV  AX,4[BP]
      SUB  AX,#TIME_RD
      ADD  AX,#7              ; plus mandatory recovery time
      OUT  DX,AL
      XCHG AH,AL
      OUT  DX,AL

      STI                     ; re-enable interrupts

      CALL WaitTimer
```

(continued)

Listing l-continued

```
      JMP PulseDone

;- wait for timer interval to expire

WaitTimer EQU *
      MOV DX,#I8254_BASE_A+6; latch & read back counter 2 status
      MOV AL,#$E8
      OUT DX,AL

      MOV DX,#I8254_BASE_A+(DS_CTR*2); read counter status
      IN  AL,DX

      TEST AL,#$80             ; wait for it to go high again
      JZ   WaitTimer
      RET

PulseDone EQU *

      MOV  DX,#SYNC_ADDR_A     ; clear sync pulse
      IN   AL,DX
      XOR  AL,#$02
      OUT  DX,AL
}

      return !!RetValue;       /* return Boolean value */
```

transaction starts from the same known state.

In general, the PC will read the serial number only once during its power-on reset sequence. The DS2400 has no power dissipation specification, but leaving the signal line high at the end of the transaction eliminates the few milliwatts dissipated in the pullup resistor...which is surely orders of magnitude more than the DS2400!

Figure 3 details the five waveforms needed for the DS2400 interface. The reset pulse is easy, because it has only minimum times: no less than 480 µs for both the low and high parts. The transmit and receive pulses are more critical because they are only a few microseconds long and the PC must accurately measure the received pulses to decide what they are.

When a project's timing is denominated in microseconds and you're charged with writing the pulse measurement code, get ready for some serious thinking! The first question is whether it's even possible, and if you decide it is then the next question is what's the best way to pull it off.

As an aside, sometimes you really will be asked to do the impossible. One recent BBS thread involved a project that was supposed to sample an analog voltage at something like 20 MHz and store 16-bit values direct to disk using a custom PC ISA bus card. If you've been paying attention to this series, you know that can't ***possibly*** work (why?) but the requirement was proposed in all seriousness. My advice was to present the ISA bus limits, then hide behind a bush if the project continued in defiance of reality.

Fortunately, it was canned when the engineer explained how the PC world was really put together....

HARDWARE TIMING

If you use the DS2400 in an 805 1 microcontroller system you can determine pulse widths by simply counting instruction cycles and writing a timing loop with a known duration. Not so with the 80386! In fact, 80x86 CPUs encourage programming as an experimental science: figure out how long a loop should last, write it, measure the results, then tweak the loop count to make the answer come out right. With any luck, a few iterations will converge on the desired value.

If your luck goes the other way, though, changing PC board vendors, moving the program from RAM to EPROM, or just twiddling a few BIOS setup values will clobber your well-tuned loop. Some of the tricks you use routinely in 8051 code just don't apply anymore: delay loops are in the "better to avoid if at all possible" category, although 1'11 show you a good use for one a little later on.

The 8254 timer chip on the Firmware Development Board was intended for just such applications: it can measure time intervals up to 9 ms with 139-ns resolution. It's connected to the 14.3-MHz clock, so the time intervals do not depend on the CPU's clock frequency or system board hardware, which means you'll get the same results every time.

Remember, however, that there are some limitations. If you are trying this on an old 8088 system, it may not execute fast enough to keep up with the short intervals produced by the 8254. The pulses will be timed by a single pass through the timer code because the hardware will be ready long before the code checks it. On the other hand, the Firmware Development Board won't work in an 8-bit ISA slot, so you'll have a few other problems before getting to the time delays.

Because the DS2400's read and write pulses are so similar, I combined the code into the routine shown in Listing 1. PulseDS2400 is written in 8086 assembler in a Micro-C wrapper, so the input arguments and local variables are on the stack rather than static storage locations.

Although the 8254 timers can produce hardware interrupts, I use simple software polling to detect the end of each pulse. Remember that hardware interrupt handlers must start by saving the CPU registers and setting up the segment registers. When you are dealing with microsecond pulses, that takes far too long.

There are sneaky ways to streamline hardware interrupt handlers under very special, carefully controlled, conditions, but 1'11 save those for a later column. You've seen some of the tricks already: the key is to know exactly when the handler will be active so you can preload the registers. The DS2400 didn't call for such gymnastics, so I could get away with polling. Whew!

The first timing loop measures the low part of the pulse. If the mainline code is writing a zero bit, this will last for 90 us, but when it is reading data from the DS2400, the pulse is only 5 µs long. PulseDS2400 also sets a sync bit in the parallel printer port to mark its entry into this code; if that bit never goes low again, you know you have hardware problems on the Firmware Development Board. If you plug in the LED-and-switch box I described in Issue 31, you can watch the firmware in action, which is often reassuring.

The 8254 output pin goes low during the pulse and returns high at the end. The delay loops load the timer count registers to start the interval, then poll the output status bit. Polling involves writing a latch command to the 8254 and reading the status register, so it's slower than a pure input bit. A 33-MHz '386SX is fast enough that this doesn't add too much delay, but if time were really critical you could route the timer output bit back through an input port to poll it directly.

After the low-pulse time ends, the code writes a one to the DS2400 output bit and sets a two-microsecond delay. When that interval expires, the code samples the DS2400 input bit and records the value; if this is a read pulse, the DS2400 controls the signal line. For bits written from the PC, the sampled value is simply ignored.

Following the sample, the code loads the timer with the rest of the required high time plus the mandatory one-microsecond idle time and waits for it to expire. Because the DS2400 output is now high, additional delays are OK, so the code also enables interrupts. The DS2400 spec sets the minimum idle time, but does not specify a maximum so a protracted delay will not cause a data error.

The code this month includes routines that use PulseDS2400 to write the 8-bit command word (always OF) and read back all 64 bits from the

Listing 2—*The DS2400 includes an d-bit CRC computed over its Type ID and serial number data. If the byte you compute using this algorithm matches the one read from the DS2400, the odds are pretty good that no errors occurred during the transfer. This code is based on the 8-bit algorithm shown in the DS2400 data sheet, but uses 16-bit registers to shuffle the bits because the 8051 and 80x86 CPUs do no set the ALU flags in the same way.*

```
int CRCDS2400(pData)
BYTE *pData;


int CRC:

   pData = pData:              /* keep compiler happy */
   CRC  =  0:                  /* clear both bytes */

asm {
   MOV     AH,#0               ; set up initial CRC
   MOV     SI,4[BP]            ; set up data pointer
   MOV     CX,#7               ; set up byte counter
OuterLoop EQU *
   PUSH    CX                  ; save byte counter
   MOV     CX,#8               ; set up bit counter
   MOV     BL,[SI]             ; fetch data byte
CRCLoop EQU *
   MOV     AL,BL               ; set up data byte
   XOR     AL,AH               ; combine low bit with old CRC
```

(continued)

DS2400. It's quite straightforward, except for one little, teeny, tiny detail: you must transmit and receive bytes low-order bit first. Believe it or not, that critical fact isn't mentioned anywhere in the data sheet!

CHECKING YOUR ID

Once you read the DS2400 data, you should verify that it is correct by using the CRC byte. You may, if you like, assume that nothing can go wrnog (oops!) with such a simple interface, but I'd suggest checking the CRC value just to be sure. The CRC is computed over the Type ID byte (which is always 01) and the six bytes of data.

The DS2400 data sheet presents the CRC algorithm as an 8051 assembler subroutine. It's tempting to just transliterate it into 8086 assembler, but there's a gotcha: the two CPUs set the flags differently for some ALU operations, so a simple line-for-line conversion won't work. Listing 2 shows the 8086 code I use to compute the CRC; it takes advantage of the fact that we can handle 16-bit values with no extra effort.

Just when you think you've got everything right, though, there's a gotcha lying in wait. I recall a BBS thread from some years back where somebody simply could not get the right CRC value. They were using the Dallas algorithm on an 8051 CPU, so there was no question of a translation error, but the CRC byte read from their DS2400 chips simply did not match their computed values.

After considerable headscratching and many calls to Dallas' tech support, they found that their DS2400s were wrong! There was a little glitch in the code driving the production line laser: it burned a unique serial number into the chip, then computed and burned *the wrong CRC value*! Talk about hard-to-find bugs....

Needless to say, that was fixed PDQ so all current DS2400 chips are OK. However, if you have a stash of old DS2400s check them very carefully just to be sure....

I've said it before and I'll say it again: hell hath no fury like that of an unjustified assumption!

Listing **2**—*continued*

```
    ROR    AX,1              ; put result into high bit of CRC
    JNC    Zero              ; and into carry flag, too
    XOR    AH,#$0C           ; stir in preshifted constant factor
Zero EQU *
    SHR    BL,1              ; set up next data bit
    LOOP   CRCLoop           ; repeat for all bits
    INC    SI                ; aim at next byte
    POP    cx                ; fetch byte counter
    LOOP   OuterLoop         ; and repeat for each byte
    MOV    -2[BP],AH         ; set up return value

    return  CRC;
```

The sernum.c program available on the BBS reads back the DS2400 data, calculates the expected checksum, and displays everything. To judge from the DS2400s I have on hand, Dallas produced about 400,000 parts before I bought these, at least if they're assigning the serial numbers sequentially.

If your circuit doesn't work the first time, sernum also writes the command byte in a tight loop so you can scope the DS2400 data pin to see what the trouble may be. The circuit is simple enough that a solder splash should be the extent of your troubles, but you never can tell....

LCDs REDUX

The Firmware Development Board sports a pair of seven-segment LED digits that we've used to report error conditions and display status information. The messages tend to be cryptic, but for our purposes they suffice. If you want to dress your system up for company, though, you need at least a smidgen of legible text output. The small LCD panels we've used on 8051 systems are equally handy on PCs; it's a simple matter of firmware to get one working.

Back in Issue 8 I did an in-depth review of these displays, so I'll refer you there for the details of how they

Listing **3**—*This code writes a sing/e byte to the LCD pane/. The Mode argument determines whether the byte is a character for the display buffer or a command to the LCD controller. The BlipEnable routine pulses the LCD's Enable input high and low to strobe the byfe into fhe controller.*

```
LCDSendByte(Data,Mode)
WORD  Data;
WORD  Mode:

WORD PortValue;

     if (!(Mode & SENDFORCE))
     {                                /* force without wait? */
          LCDWaitBusy();


     PortValue = (Mode & SENDDATA) ? LCD_RS : 0;
                                      /* cmd/data, -WR, enabled */
     PortValue |= Data:               /* combine data */
     outpw(CTLS_ADDR,PortValue);

     BlipEnable(PortValue);

     PortValue |= LCD_RW;             /* disable -Wr line */
     outpw(CTLS_ADDR,PortValue);
```

work and what all the interface lines mean. For now, I'll concentrate on the specifics of the Firmware Development Board's application.

The common denominator of all these panels is the Hitachi HD44780 LCD controller that provides their (admittedly limited) intelligence. Most of the character LCD panels you'll find use this controller, but it's a good idea to get the data sheets along with the hardware just in case you wind up with an oddball. Drop a note on the Circuit Cellar BBS and we can probably walk you through getting your display working.

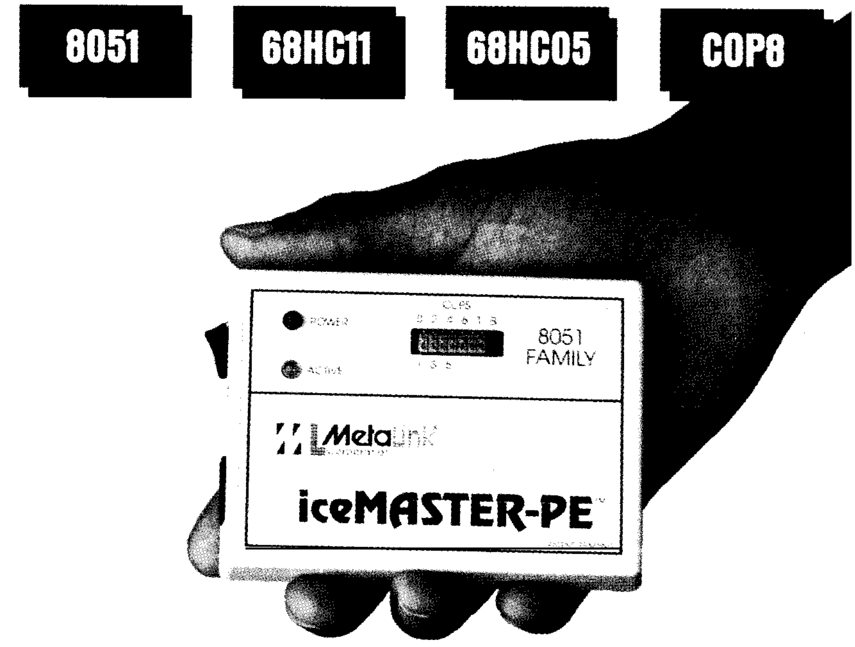
The code this month can handle any HD44780 display up to the maximum of 80 characters; it arrives set for four lines of twenty characters. A pair of constants define the number of visible rows and columns so, after changing two lines and recompiling, you're up with your new display.

As shown in Figure 1, the LCD panel interface uses port I/O rather than a direct PC ISA bus connection. Although a bit-banged interface is considerably slower than a bus hookup, the firmware can still update the entire display faster than the liquid crystals can respond. So that's probably fast enough.

Listing 3 shows the few lines of code required to write a byte to the display. The LCD's Enable line latches the data into the controller, so the code writes the 16-bit port three times: once to set up the data and control lines with Enable low, once to set Enable high, and a final write to lower Enable.

Because the port's data cannot be read back into the CPU, the code holds them in the PortValue variable between writes. This simple trick suffices for now, but I'll describe some of the problems that can arise in real-world code after we get the LCD panel on the air.

Listing 4 is the routine that reads status and data from the LCD controller. This code is used in two places: to check the controller's (Busy) status flag before sending a new command or data byte and to read the display buffer during vertical scrolling. Both of these functions can be simulated in firm-



ware, but at the cost of one ‘245 buffer. I decided to use Steve’s favorite programming language: solder.

Despite my tirade about software delay loops, you’ll find one buried in the heart of LCDReadByte just after the code raises the Enable line. Some of the LCD panels I’ve used need more time to drive data back to the Firmware Development Board. The value I settled on creates a 150-µs pause that is about four times the delay my worst panel needs, but if you get erratic vertical scrolling, you’ll know what the problem is and where to fix it.

This may be related to the ribbon cable layout that comes naturally to these panels, although I don’t see a glitch on the scope. You might want to experiment with different cable layouts to see if bracketing the LCD’s Enable or the data lines with pairs of quiet ground lines helps. Trust me on this: if you haven’t used firmware to fix hardware before, it’s time you got started....

The remaining LCD code adds cursor positioning, string output, simple scrolling and some control character processing. It’s straightforward code available on the BBS if you’re interested. The lcdtest.c program provides several test loops with scope triggers, as well as a routine that copies incoming serial characters to the LCD so you can check displays out by hand.

Extra credit project: you can add ANSI cursor positioning support fairly easily. The description of the LCD-Link project in Issue 27 should get you started.

Listing 4—*This routine reads a byte from the LCD controller; the Mode argument determines whether it comes from the display buffer or the controller. The loop in the asm{} section compensates for delays caused by cable capacitance and must be tuned for each system...so there is no need for a precise measurement.*

```
int LCDReadByte(Mode)
WORD  Mode:
{
WORD  Data;
WORD PortValue;

 outp(SYNC_ADDR,inp(SYNC_ADDR) | 0x02);  /* scope sync */

 PortValue =(SENDDATA == Mode) ? LCD_RS:0;/* read data or addr? */
 PortValue |=LCD_RW + LCD_NODRV; /* LCD rd. and disable drivers */
 outpw(CTLS_ADDR,PortValue);

 PortValue |= LCD_E;                     /* strobe data fr. LCD */
 outpw(CTLS_ADDR,PortValue);

asm {
 MOV     CX,#RD_DELAY              ; delay allows data settle time
?1pl LOOP ?1pl


 Data =inpw(STAT_ADDR) & 0x00FF; /* fetch and isolate LCD data */

 PortValue &= ~LCD_E;                    /* remove LCD strobe */
 outpw(CTLS_ADDR,PortValue);

 outp(SYNC_ADDR,inp(SYNC_ADDR) & ~0x02); /* sync off */

 return  Data:
```

SHARED BITS

The PortValue variable let LCDSendByte toggle the LCD Enable line while holding all the other bits steady. Unfortunately, LCDSendByte has no way to preserve the existing bits because the port can’t be read back. That has no effect on the demonstration program, but in real life it’s a killer problem.

For example, the watchdog timer routines I discussed in the last column send a new bit to the watchdog about six times a second. That interrupt handler cannot read back the existing bits, so if you are writing a byte to the LCD controller, it will clobber your data while toggling the watchdog. When the LCD write completes, it returns the favor by clearing the watchdog bit.

This problem is easy to solve in a single program: all your routines must use a single global PortValue variable to hold the current state of the output port. If an interrupt handler can change the port, you must protect your noninterruptable code by disabling and enabling interrupts around the sections where you twiddle PortValue.

An oversight here can cause some subtle bugs. When your mainline code loads PortValue into a register to change the bits, as most C compilers will, your interrupt handler can get the “old” PortValue from the variable, change a few other bits, and write it back; your mainline code will then destroy the handler’s bits when it writes its PortValue back into the variable.

It’s worth working through a few examples on paper to convince yourself that you’re in a lot of trouble. Look for those instants in time when PortValue differs from the actual port bits: that’s when the interrupt handler will strike. Then make sure only one routine can alter PortValue at any time and always keep the variable in sync with the hardware port.

I’ll be working through these issues in the next few months, but I figured I should at least point out the land mine right now. This problem is known and the solutions are documented in the multitasking literature. You’ve surely read about semaphores, critical regions, atomic locking, and threads of execution by now; if not, well, back to the books.

RELEASE NOTES

The downloadable code this month has two Micro-C programs that exercise the DS2400 and LCD panels. You can combine the routines, but pay attention to the “shared bits” problem described above. If you stir in the watchdog code from the previous issue, it gets still more complex!

If you don't like the notion of a serial-numbered PC, you're in good company. But if you're writing code that must know for sure that it's running on authorized hardware, now you've got a good idea of how to start. I'll leave the encryption and hacker defenses to other writers, but I will make use of the serial number in a few dangerous projects.

If serial numbers ***really*** annoy you, though, I'm sure The Great Dybowski can come up with an 805 1 -based DS2400 emulator to produce any serial number you like with the push of a button...and I might even help him out!

Next month: despite my best intentions, I'm going to talk about embedding an entire Micro-C program as a BIOS extension. You can't fit too much into 32K, but sometimes a little code is better than none at all. ❑

Ed Nisley, as Nisley Micro Engineering, makes small computers do amazing things. He's also a member of the Computer Applications *Journal's* ***engineering staff. You may reach him on CompuServe at*** *74065,1363* ***or through the Circuit Cellar BBS.***

SOURCES

DS2400 Silicon Serial Numbers are available from the usual mail-order suppliers. LCD panels based on the Hitachi HD44780 controller make frequent appearances in electronic surplus catalogs, so you can save quite a bit of money if you need just one or two. Check the supplier listing in the February 93 column for suggestions.

Pure Unobtainium has the complete Firmware Development Board schematic, as well as selected parts. Write for a catalog:

Pure Unobtainium
13 109 Old Creedmoor Rd.
Raleigh, NC 276 13
Phone/fax: (9 19) 676-4525

I R S

410 Very Useful
411 Moderately Useful
412 Not Useful

Updating TIM for the HCS II

FROM THE BENCH

Jeff Bachiochi

Add the Touchtone Interactive Monitor to your home control system

You've waited and waited, and it's finally here. The HCS II Telephone Interface offers a registered DAA and complete DTMF transmitter/receiver. There's even an optional text-to-speech interface available for it.

Hardware for the HCS II telephone interface has taken an unexpected detour. The original plan for adding the telephone interface link to the existing HCS network was unexpectedly canceled. Stay calm now, I said the orginal plan was canceled, not the new plan. Let me explain.

Production boards for the HCS II's network links are single-board layouts of prototyped hardware. Prototyping a new I/O link is easy because only the new I/O circuitry in question needs to be built. Although the production board is a single-board layout, to lower prototyping costs, we always build the new interface circuitry on top of an existing RTC31 controller board to test it out (*Circuit Cellar INK,* April/May '89). The RTC processor board contains the engine, communications interface, and program storage. A piggyback-style expansion bus offers I/O interfacing and plenty of prototyping area.

This approach reuses a cornerstone building block (80% of the circuit) leaving only the untested bits of silicon to be manually connected. The intake of tin/lead/flux fumes is therefore kept to a minimum. (Wouldn't it be nice if someone would produce flux in different fragrances. Anyone willing to invest in such a venture?) This shortcut allows you to quickly get down to the software necessary to make the hardware jump through the proverbial hoop.

INITIALLY, A SIMPLE CONCEPT

As with the other HCS link modules, the telephone interface would talk to the Supervisory Controller through the twisted-pair network. The hardware should allow a connection to be established between the telco pair and the telephone link interface board.

It all starts with a legal connection to the phone system allowing the HCS to monitor the phone line, answer a ring, and recognize and produce DTMF signals. In addition, the appropriate circuitry for injecting and retrieving

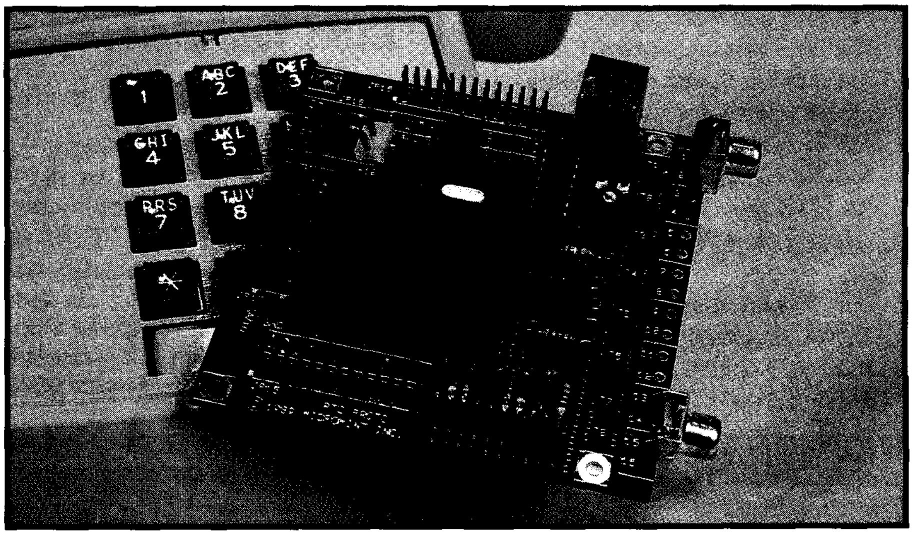


Photo 1—*With* ifs new *telephone interface, the HCS II can originate* ***calls*** *or receive* calls and in ***both*** cases allows ***the*** homeowner ***to*** interact ***with the*** system remotely.

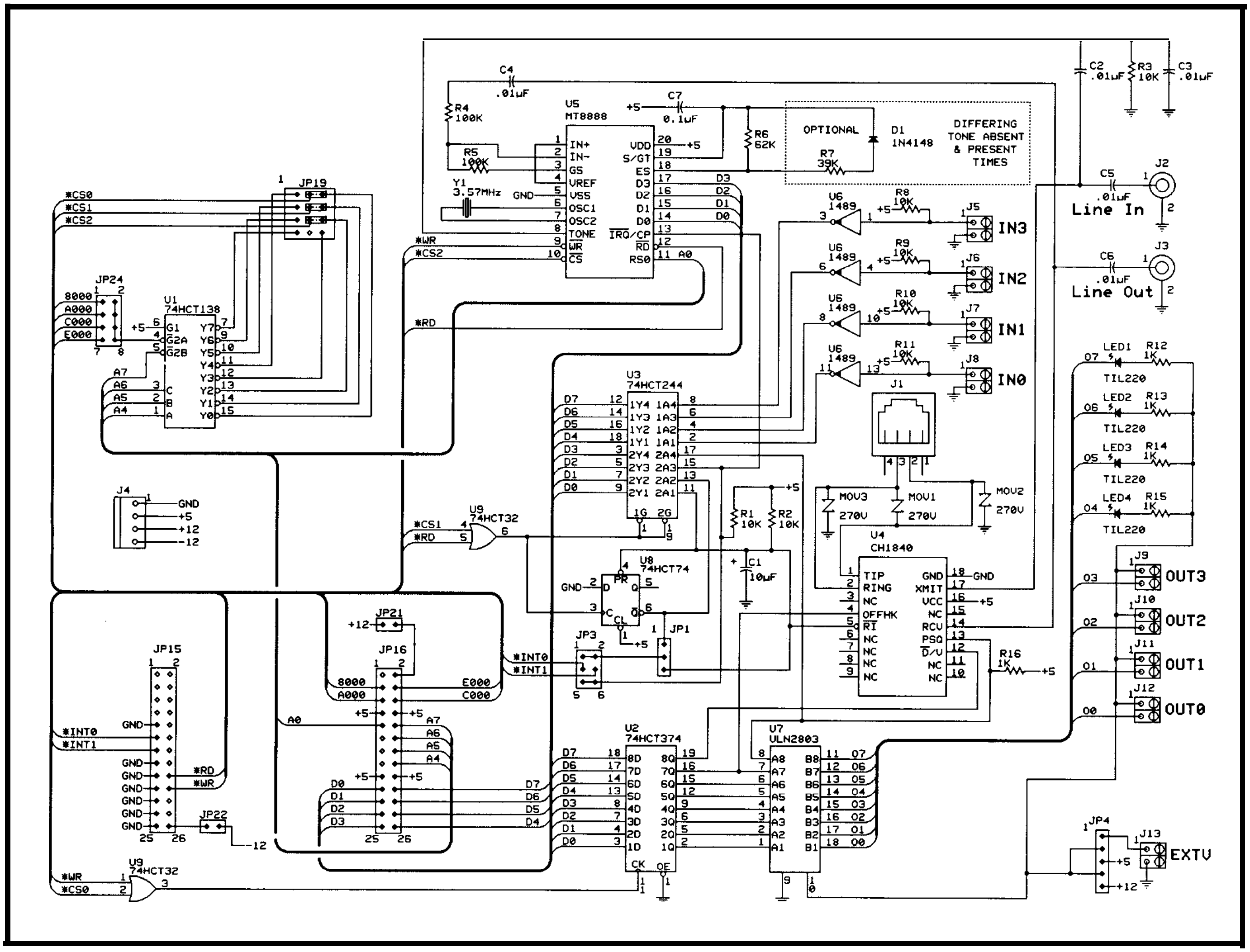


Figure 1—*All activity* on the *DTMF* board centers around *the M8888 transmitter/receiver chip. The supporting cast of components includes Cermetek's CH1840 DAA chip and some glue logic.*

other audio offers some additional benefits. And what network link would be complete without some high-voltage I/O: just a few ±30-volt inputs and a modest offering of open-collector outputs. The schematic for the newest member of the RTC/HCS family is shown in Figure 1.

DAA

The FCC requires that any device connected to telco equipment conform to basic safety standards (Part 68) protecting both telco and the consumer. Of course, you must prove you comply, or you can use a preapproved front end or DAA (direct access arrangement). DAAs are not overly complex, nor are they inexpensive, which doesn't seem to make sense. If you've ever looked for a DAA, you will most certainly have come across the name Cermetek. In fact, it is probably the only name you have found. In the whole electronics industry I don't know of another market in which competition seems to disappear so silently.

To comply with Part 68 your circuitry must:

*Prevent supplying currents greater than 10 mA and voltages greater than 70 volts to the telco line.
*Limit strength of transmitted signals between 100 Hz and 1 MHz.
*Not create an imbalance between the two telco connections (tip and ring] and ground.
*Limit both "on hook" (DC] and "off hook" (AC or ringer) impedance.
*Provide protection against service theft.

If, after severe electrical surge, vibration, temperature, and humidity tests, the device continues to comply, you may then submit an application to the Federal Communications Commission. Proving compliance is best left up to FCC testing laboratories. For a fee, starting at about $1500, they will test your device and suggest changes necessary to bring it into compliance. They will submit the proper paperwork, on your behalf, to the FCC.

The Cermetek CH1840 complies with all requirements for voice and data use through V.32. It is Part 68 tested and registered. This registration is passed on to the user as long as simple trace and isolation criteria are obeyed.

Refer to Figure 2 for a block diagram of the CH1840. This DAA requires only +5 volts, which goes a long way toward simplifying the power supply design. Only two signals are necessary for telco connection. The

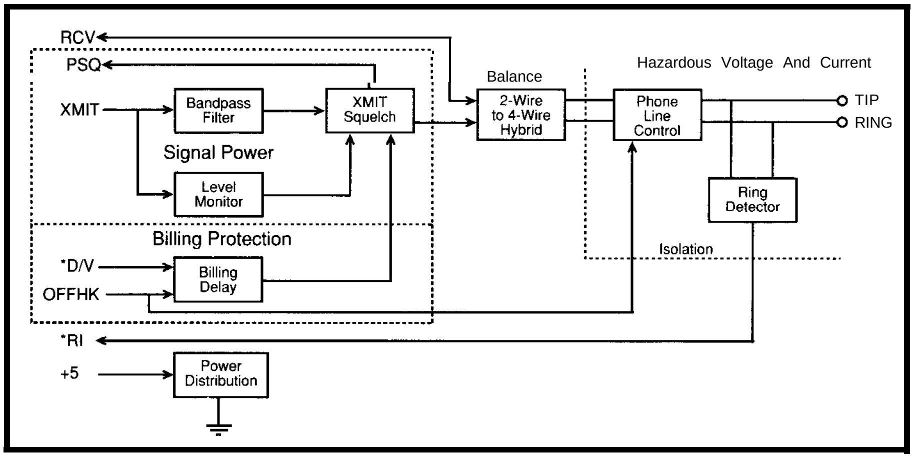


Figure 2—*The preregistered* **DAA** *provides a legal* **interface to** *the telephone line. It connects audio to the telco's tip and ring lines plus* **provides ring** *detect and off-hook control. An optional forced billing delay a/so helps comply* **with** *regulations.*

Tip and Ring signals (named after parts of the phone jacks used to manually connect customers on the first telephone switchboards) carry both incoming and outgoing audio. These signals are not referenced to ground. All remaining signals are user connections. The Rcv and Xmit connections are ground referenced and provide the user with separated audio in and out. *Data/Voice and Offhk inputs enable the billing delay and take the line off hook. *RI and Psq outputs indicate the presence of a ring signal and a squelch of the audio input.

DTMF

One of the first transmissions of data in a digital format was the telegraph. Dots and dashes (zeros and ones) are grouped to represent alphanumeric characters. Today we are all too familiar with sounds of modem or fax conversations, especially when connecting with one unexpectedly. Telephones based on dual tone, multifrequency (DTMF), also known by the trademarked name Touch Tone, are rapidly replacing the old rotary dial telephones. This is allowing central offices to upgrade to electronic switching equipment. The advantages of an electronic system over a mostly mechanical one are many: power savings, increased MTBF, and flexibility just to name a few. It's easy to see why these advancements were eagerly (read "slowly") embraced by the phone company.

Low tone	High tone	Digit
697Hz	1209 Hz	1
697Hz	1336 Hz	2
697Hz	1477 Hz	3
697 Hz	1633 Hz	A
770 Hz	1209 Hz	4
770 Hz	1336 Hz	5
770 Hz	1477 Hz	6
770 Hz	1633 Hz	B
852Hz	1209 Hz	7
852Hz	1336Hz	8
852Hz	1477 Hz	9
852Hz	1633 Hz	C
941 Hz	1209 Hz	0
941 Hz	1336 Hz	•
941 Hz	1477 Hz	#
941 Hz	1633 Hz	D

Tablel-Each *of the* **16** *valid DTMF combinations* **is** *made up of a* **pair** *of tones.*

DTMF tones are triggered by a matrix of four columns (three columns on your telephone) by four rows that can produce sixteen (twelve on your phone) possible key positions. Each column and each row has a different tone assigned to it, giving sixteen (twelve) distinct tone pairs. Each key combines the appropriate row and column tones. In this fashion, sixteen possible combinations (a nybble of information) can be passed at one time using only eight discrete frequencies. The eight tones were carefully selected to fit within the narrow 3-kHz passband of the telephone system and so the tone harmonics would fall between other tones so as not to cause a false image. See Table **1** for the valid tone-pair combinations.

Notice the digits A-D are not on your phone, however these are legal DTMF codes and can be both recognized and produced by existing DTMF circuitry. The ability to receive and transmit DTMF is not always needed in many products, but when it is, one of the most widely chosen ICs is the M8880 (from Teltone, Mitel, and others). Since the M8880 is specifically designed to interface with a 6800 series microprocessor, it doesn't interface easily to 803 **1** -style architecture, the one used on the RTC3 **1.**

Sometimes manufacturers do listen to designers though, because the M8880 has mutated just this year into the M8888. Same core, but now can be directly interfaced "Intel style." DTMF tones are produced by writing a nybble to the M8888's data register. Recognized codes are reported in the same nybble format. An interrupt pin can flag completion of an event (sending or receiving a DTMF signal) or a status register can be polled for this information. The M8888 will produce DTMF signals with the proper duration and spacing as well as reject incoming tones which do not fall within acceptable limits. All this integration makes DTMF a snap. It is the call progress interpretation which will cause the headaches. For a look at the level of functionality this chip brings to the table, have a look at Figure 3.

Call progress tones fall into a separate passband of 225–550 Hz. The M8888's call progress mode switches in a special bandpass filter which passes only these signals to the ● IRQ pin. It is left up to the user to determine the state of the call from this signal. See Table 2 for typical call progress signals.

If you look closely at the frequencies and durations, you will see that there are actually two ways of distinguishing between call progress tones. By comparing On/Off timings and by determining frequency content. See the scope shots in Photos 2a–d for signal comparisons.

AUDIO

Up to this point, everyone at Circuit Cellar had pushed the HCS telephone interface in the same direction. Although we all agreed voice response is necessary, tempers flew to new heights whenever the interface topic turned to speech.

Digitally recorded speech is hot right now. New compression methods are popping up everywhere. Everyone seems to have their favorite, and they're willing to defend it to great limits and with considerable zeal.

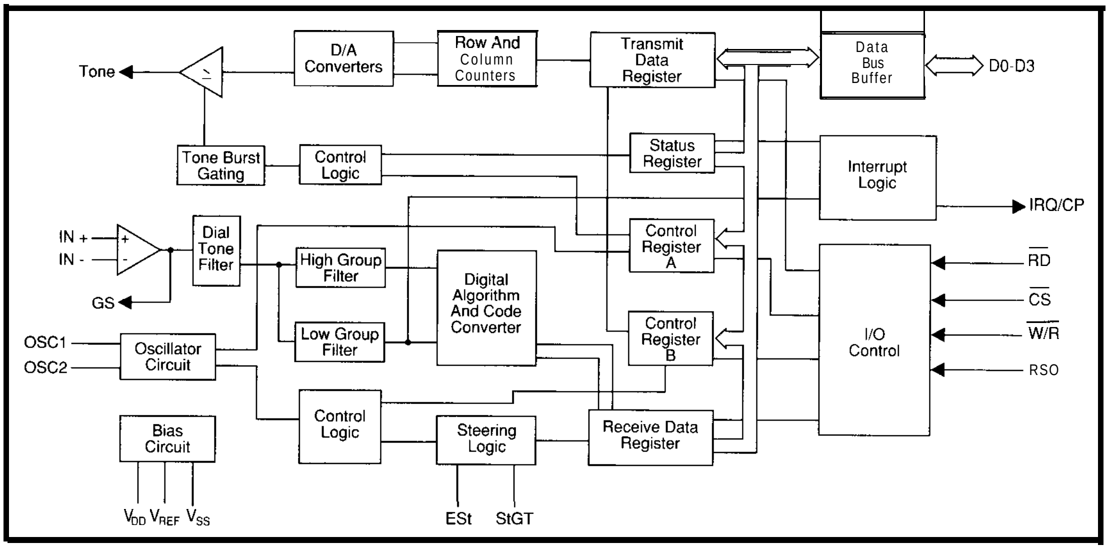
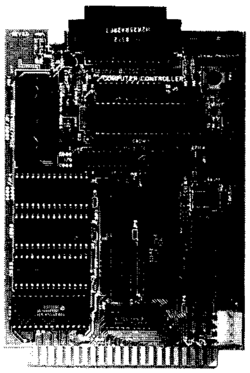


Figure 3—*The M8888 DTMF transceiver takes care of all the details necessary for receiving and generating DTMF signals. It also provides limited call progress information for further processing by the host computer.*

No matter what the algorithm, some facts can't be ignored. To store large vocabularies, digital speech requires lots of memory. It also requires extra hardware to record, edit, and store the phrases. Canned speech severely limits the potential responses you may wish to give. Think of all the possibilities just to say, "The outside temperature is x degrees."

So where does this leave us! Speechless? Not exactly. Early speech synthesizers were based on the 30 or so phonemes (sounds) that all words are composed of. Text-to-speech firmware dissected each word into reproducible phonemes. The big advantage here was unlimited vocabulary without the need for banks of memory or recording hardware. The disadvantage was that it would produce almost unintelligible speech.

Speech quality has improved since then; more processing power has helped create smarter algorithms. Exception tables help pronounce even the screwiest of English words correctly.



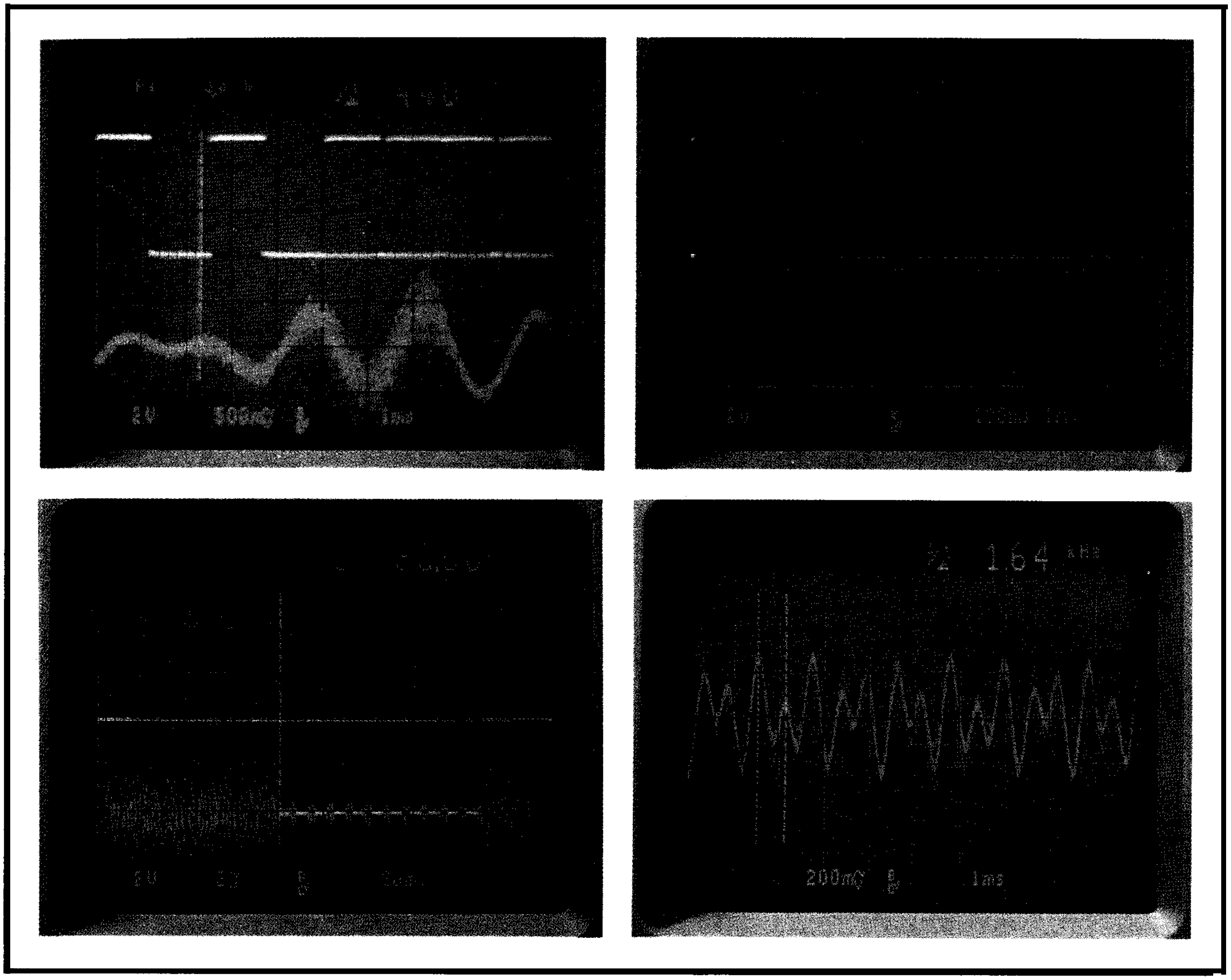


Photo 2—*Call progress tones can be distinguished by either comparing on/off timings or finding frequency content The upper left screen shows a dial tone, upper right is a busy signal, lower left is the signal produced when the receiver is left off hook, and lower righf shows a DTMF digit.*

After evaluating many forms of speech reproduction, the text-to-speech approach won hands down due to its reasonable cost and flexibility in our application. To help speed production, we chose to license the firmware but lay out the circuitry on another RTC expansion board.

When the circuitry for the DTMF interface is combined with the speech synthesizer, the processor, and network communications, the single link board size expands beyond what we thought was practical for a network node. This implementation was beginning to exceed optimum. Stepping back far enough for a look at the forest gave all of us the insight necessary to see the trees in a different light. Why bog down the network with text and dialing strings when a better solution was staring back from the prototyped interface? The RTC expansion headers designed into the 8031 board I used to test these prototypes were also part of the HCS Supervisory Controller. A separate link was not necessary. The HCS can handle these two boards (DTMF and speech) directly. All heads turned to Ken, "It can, can't it?" A pause of the appropriate length was inserted here to provide just the right amount of suspense. "Makes sense to me," was the answer we all wanted to hear.

Function	Frequency (Hz)	On Time [seconds)	Off Time (seconds)
Dial	350 + 440	continuous	
Busy	480 + 620	0.5	0.5
Ring Back	440 + 480	2	4
No Such Number	200 to 400	continuous FM @ 1 Hz	
Left Off Hook	1400 + 2060 + 2450 + 2600	0.1	0.1
Congestion	480 + 620	0.2	0.3
Reorder	80 + 620	0.3	0.2
Ring Back PBX	440 + 480	1	3

Table P-*Knowing the frequencies and cadences used by typical call* **progress** *signals is necessary to write code to support the call progress feature of the M8888 DTMF transceiver.*

I couldn't be happier. Not only does the HCS get its telephone interface, but I get a couple of new expansion boards I can use with my RTC boards.

TYPICAL HCS CONVERSATION

Ringggg... Rin<click> "Hello, Circuit Cellar INK."

"This is the HCS II calling for Jeff Bachiochi. Jeff, please enter your four-digit access code."

<hold pressed>

<extension called>

"Jeff, some machine is asking for you on line 12"

"Thanks, Nancy!"

<line picked up> bleep-blurp-burp-bip <DTMF tones entered>

"Jeff, I know you plan to be home soon, would you like me to turn on the air conditioning for you? It is 82 degrees in the family room."

bleep <"N" pressed>

"It is now after 3:00 P.M. The security system is still active. No one has entered the premises since it was activated at 9:30 A.M. I am worried about the children. Do you have any additional commands for me?"

bleepity-bloopity-blump <Touch Tone sequence pressed>

"I understand. Record channel 4 at 8:00 P.M. for 1 hour. Do you have any additional commands for me?"

bleep <"N" pressed>

"Don't worry. I have everything under control. Goodbye."

<called ended>

"Who *was* that?" questioned Nancy, our receptionist.

"Oh, just my house checking in. I forgot to tell it we were all going out to celebrate our anniversary tonight! It was just following orders." 1 grinned at her sheepishly as I left for the evening. It felt good to know that someone was watching the house while I was out. Now, did I remember to tell the HCS where I could be reached tonight? ❑

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.*

CONTACT

Cermetek Microelectronics
1308 Borregas Ave.
Sunnyvale, CA 94089
(408) 752-5000

Teltone Corp.
22121-20th Ave. SE
Bothell, WA 98021
(206) 487-1515

Mite1 Semiconductor
2321 Morena Blvd., Ste. M
San Diego, CA 92110
(619) 276-3421

HCS II DTMF Telephone Interface (RTC-DTMF) kit or assembled
HCS II Text-to-Speech Interface (RTC-Voice) assembled only
Contact Circuit Cellar Inc., 4 Park St., Vernon, CT 06066, (203) 8752751 for more information.

413 Very Useful
414 Moderately Useful
415 Not Useful

Swiss Army Chip

SILICON UPDATE

Tom Cantrell

Although this family of all-in-one chips from WaferScale Integration doesn't have a cork screw or scissors, it will still get you out of some tight spots when board space is at a premium and you need maximum functionality.

I've found that a Swiss army knife is indispensable gear for camping trips. My 'little red gadget has proven to be a life saver on more than one occasion, such as the almost ill-fated expedition in which we found our entire beverage supply came from a primitive country that hadn't yet mastered twist-off technology.

Like a Swiss army knife, there are times where a single chip combines a variety of useful functions that come in quite handy.

Indeed, the bright idea of "combo chips" that combine RAM, ROM, and I/O is almost as old as the microprocessor itself. Figure 1 shows the circa-'70s offerings from the micro pioneers, Intel and Motorola.

The three-chip solution from Intel split the RAM and ROM/EPROM into separate chips. The 8156's 22 I/O lines combined two 8-bit ports and a six-bit port into a single device. The 16 I/O lines of the 8355/8755, complemented the 8 156's byte-oriented ones, and are individually bit programmable as input or output. Note that both chips include the latch needed to demultiplex the ADO-AD7 lines of the 8085.

Motorola took a different tack, putting the RAM on the 6802 CPU and the rest (ROM, I/O, timer] on the 6846. Joining memory are eight bit-programmable I/O lines (plus two handshake lines) and a 16-bit timer with dedicated clock, gate, and output lines.

Though they sounded like a good idea at the time, these combo chips ultimately failed. What went wrong?

The answer is that integration in the VLSI era was, and remains, a tricky proposition. Too little and you're easy picking for competitors-too much and you end up with an expensive "jack of all trades" chip that's a "master of none."

Those old combo chips were squeezed to death between the emerging 8-bit single-chip microcontrollers (like the 805 1 and 6801}, and high-density, aggressively priced, byte-wide RAMs and ROMs.

A good idea, like music and fashion, is all a matter of timing. Something that wouldn't fly yesterday may be all the rage tomorrow. Will the combo chip rise Phoenix-like from the ash heap of IC history? Or is it a another case-like disco and bell-bottoms-of "a bad idea whose time has finally come."

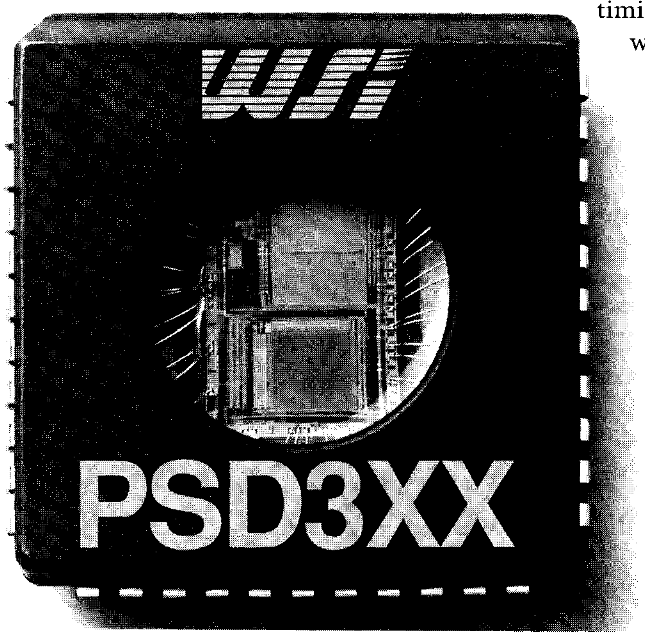


Photo 1—*By packing EPROM, RAM, and I/O into ifs 44-pin chip, the WaferScale PSD resembles combo chips of the past.*

DÉJÀ VU ALL OVER AGAIN

The WaferScale PSD (Programmable System Device] echoes the combo chips of old by packing EPROM,

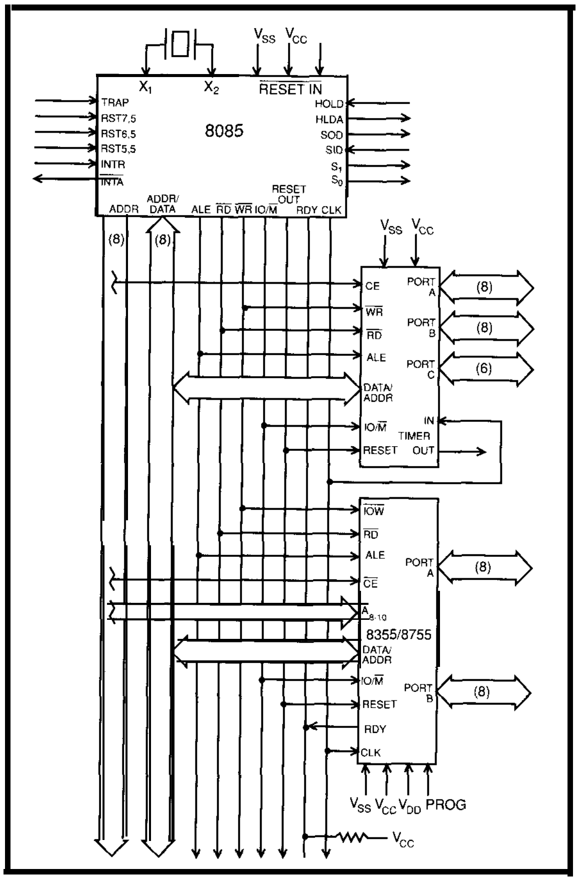


Figure la-O/d *Intel combo chips such as the 8156 and 8755 were sef aside for a three-chip solution. The 256 byfes of RAM in the 8156 and 2K bytes of ROM in the 8355 were split info different chips.*

RAM, I/O and some glue logic into its 44-pin plastic (OTP), or ceramic (windowed), leaded chip carrier. Ceramic PGA (pin grid array) and plastic quad flat pack (PQFP) packages are also available (no DIPs though).

On the memory front, both speed and density have improved over the years. Every chip in the PSD family includes 16K bits of SRAM with members differentiated by EPROM capacity; 256K bits for the PSD3 11, 512K for the '312, and a whopping one megabit for the '3 13. As shown in Figure 2, the lineup includes variants supporting 16-bit access, and speed selections from 120 ns to 200 ns are offered.

Nineteen I/O lines are partitioned into 8-bit ports (A & B), and a 3-bit port (port C). Each line's direction is programmable, and lines defined as outputs on ports A or B feature TTL or open-collector drivers.

On the surface, the PSD may appear little more than a combo chip of old on bit-bulking steroids. However, looking inside the chip (Figure 3) reveals a lot of glue logic tailored to meet the needs of today's small system designs.

PADDED ROOM

Allocating memory and I/O space in a typical design calls for a few TTLs (or a PAL) devoted to address decoding and control signal generation. Also, the processors that use a multiplexed address/data bus need at least one latch. The PSD sweeps all this inside, making it possible to build powerful systems from just two chips.

The PSD diplomatically deals with most designers' preferred bus interface, whether it be Intel {*RD, *WR, ALE] or Motorola (AS, E, R/*W} by incorporating both. Even the RESET input is programmable to be active on high or low reset pulses. Like the many other options on the PSD, these selections are put in the EPROM along with the rest of the program code.

The four combinations of 8- or 16-bit data buses, and multiplexed or nonmultiplexed operation, are also programmable. When configured for 16-bit operation (PSD '30x}, the ● BHE/ ● PSEN pin takes on the *BHE (Bus High Enable] function and supports both 8- and 16-bit access to the memory (remember, most 16-bit CPUs offer byte access). When it is in 8-bit mode, *PSEN [Program Space Enable) functions as, and connects directly to, the 8031 pin of the same name.

The PSD goes all out when it comes to address decoding by incorporating two PADs [Programmable Address Decoders, see Figure 4). One of them (PAD A) is for mapping access to internal resources, and the other one [PAD B) generates up to eleven chip selects (via port B & C} for external devices.

Though the first PSD (the '3x1 with 34K total) didn't push the 64K limit of 8-bit micros, the PSD was designed from the start to face reality by dividing the internal EPROM into eight blocks (for example, each 128K

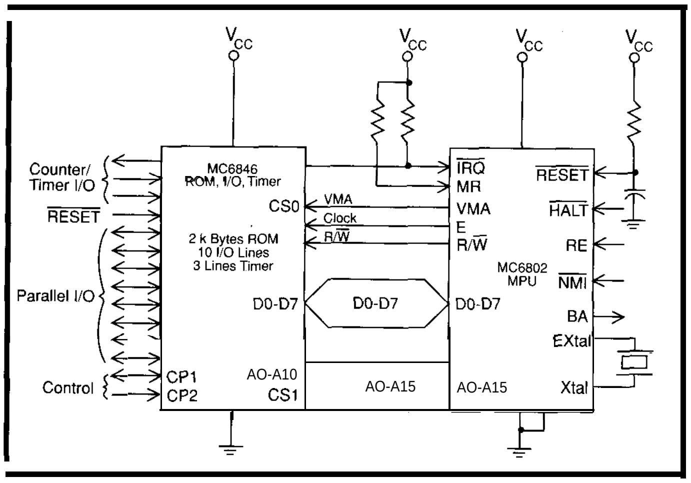


Figure 1 b—*Motorola put 128 bytes of RAM on fhe 6802 CPU, while ROM, I/O, and the timer were put on fhe 6846.*

Part No.	Description	EPROM	x8/x1 6
PSD301	Programmable Microcontroller Peripherals with Memory; x8/x16;	256Kb	x8/x16
PSD311	256Kb—1 Mb EPROM; 16K SRAM; PAD; System Features.	256Kb	x8
PSD302		512Kb	x8/x16
PSD312		512Kb	x8
PSD303		1Mb	x8/x16
PSD313		1Mb	x8

igure *2-The PSD lineup includes a standard 16K bits of SRAM for all members, while EPROM capacity varies with each chip.*

bits on the 1M bit '3x3). A built-in page register controls access to each block via page select inputs (P0–P3) to the PAD. A block in which the page inputs are PAD programmed as "don't cares" becomes global and thus a safe place to put bank switching code and interrupt vectors.

Besides serving as I/O or chip selects, port C goes beyond the call of duty by optionally acting as high-order (A16–A19) address inputs. Al9 can be further configured to function as an external ● CS input, placing the PSD in low-power standby mode when inactivated. Finally, the three bits of port C can even serve as generic PAD I/O pins, possibly eliminating a little random logic from your design.

Port A is no slouch either. Besides I/O, the low-order address inputs of the PSD can be passed through port A for connection to other chips. When used with a nonmultiplexed processor, port A serves as the data bus.

Figure 5 sums up the myriad "configuration bits" that define PSD

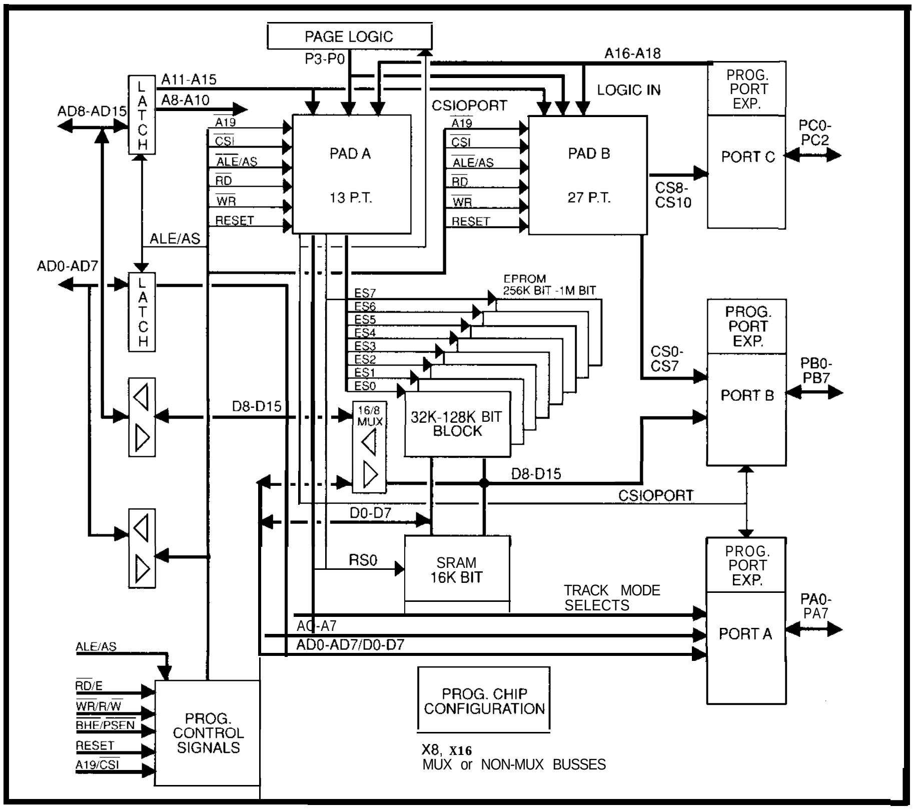


Figure 3—*Although* the *PSD* may appear to *be the same as its cousins from the past, the block diagram reveals much more glue logic to support the demands of today's small system designs.*

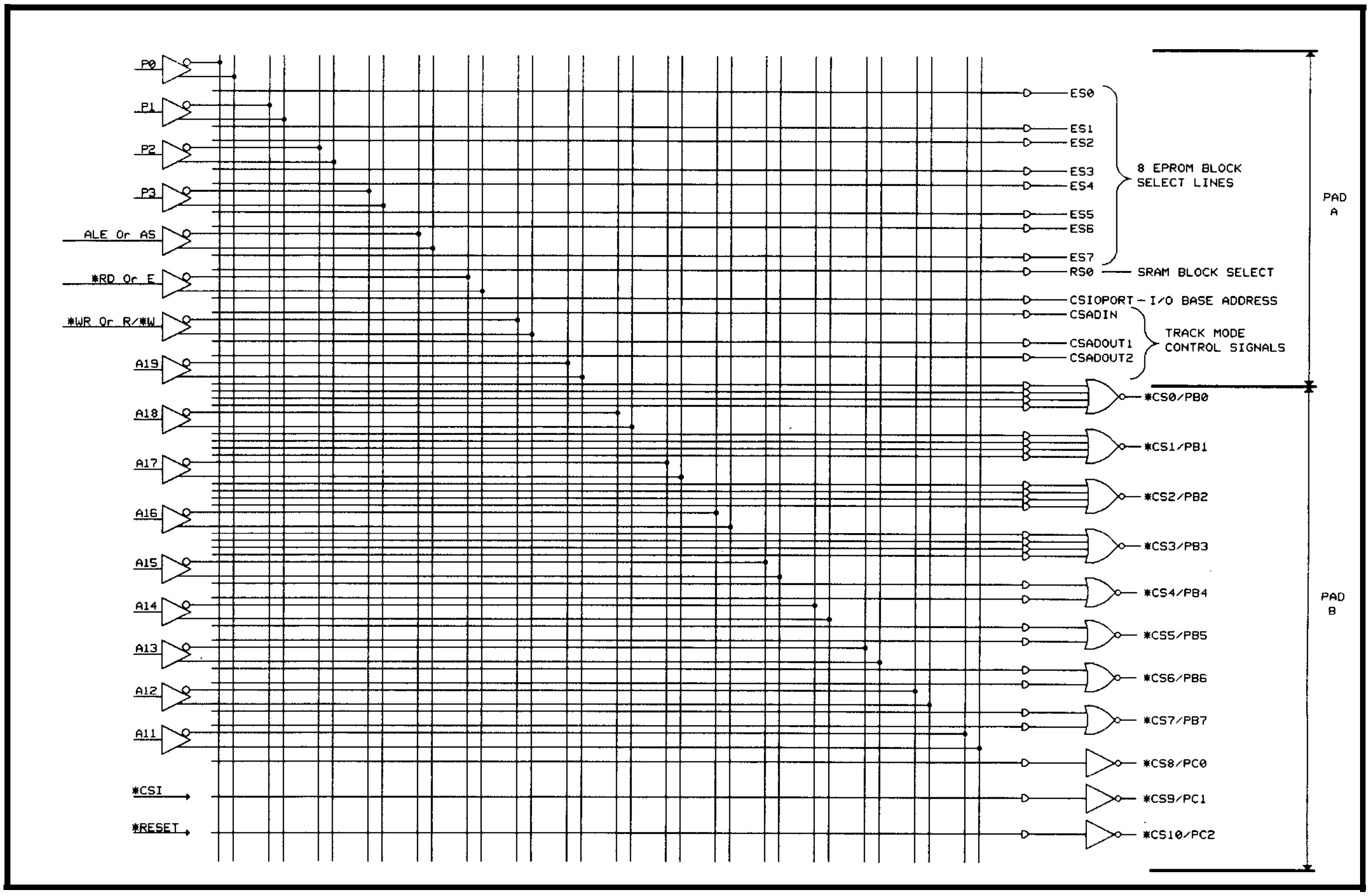


Figure 4—*Two programmable address decoders (PADs)* on the *PSD are used for mapping access to internal resources (PAD A) and generating up to eleven chip selects for external* devices *(PAD B).*

operation. There is even a security bit that prevents the configuration and PAD fuse map from being read.

The vast number of option permutations allows easy connection to almost any CPU. Figure 6 shows the connection of two likely candidates—the Intel 80C31 and Motorola 68HC11. The ***Programmable Peripherals Design and Applications Handbook*** from Waferscale shows the connections and options settings for many other popular CPUs including the '186, '286, '196, '451, Z80, 6809, 68000, and the 68302.

PERILOUS PROGRAMMING

As for any EPROM, burning your bright ideas into a PSD is a two-step process. First, you have to create the hex file, and then you have to program the chip. However, neither is especially straightforward for the PSD.

The PSD hex file contains much more than your program code. You've got to correctly set and locate all fifty configuration bits as well as the PAD fuse-map. Putting everything together by hand is only a job for the very committed (or those who should be).

When you're finally ready to blast the bits into this brainiac, V_{PP} is raised to 12.75 V, but that's where the similarity to a regular EPROM ends. The programming algorithm is quite intricate and involves forcing the chip into different "modes" in order to program and verify the PAD's configuration bits and EPROM. Worse, V_{CC} must change with the modes across a five-step range from 4.5 to 6.25 V! All in all, rolling your own programmer seems a rather messy proposition.

Some higher end programmers from third parties can support the PSD. If yours doesn't, WaferScale offers a nice setup, though admittedly a little pricey at $1765.00, it's called a PSD-Gold. This package includes a programmer, and PC-based software called MAPLE. If your programmer does support the PSD, you can get just the software by ordering the lower cost ($495) PSD-Silver package.

MAPLE is highly recommended since it makes the otherwise tedious task of configuring the PAD and its many options quick and easy.

Config. Bits	# bits	Function

CADDRDAT		ADDRESS/DATA Multiplexed (separate buses)
		CADDRDAT=0, nonmultiplexed
		CADDRDAT=1, muliplexed

CA1 9/*CSI		Al 9 or ● CSI CA19/*CSI=0, enable power-down CA1 9/*CSI=1, enable Al 9 input to PAD

CALE		Active HIGH or active LOW
		CALE=O, Active high
		CALE=1, Active low

CRESET		Active HIGH or active LOW
		CRESET=0, Active low reset
		CRESET=1, Active high reset

'COMBISEP		Combined or separate Address Space for SRAM and EPROM 0=Combined, 1=Separate

CPAF2		PortA ADO-AD7 (address/data multiplexed bus)
		CPAF2=0, address or I/O on Port A (according to CPAFI)
		CPAF=1, address/data multiplexed on Port A (track mode)

CADDHLT		Al 6-A19 Transparent or Latched
		CADDHLT=0, Address latch transparent
		CADDHLT=1, Address latched (ALE dependent)

CSECURITY		SECURITY On/Off
		CSECURITY=0, off
		CSECURITY=1, on

CLOT		AO-A15 Address Inputs are transparent or ALE dependent in nonmultiplexed modes
		CLOT=0, transparent
		CLOT=1, ALE-dependent

CRRWR CEDS	2	Determine the polarity and control methods of read and write cycles. CEDS CRRWR 0 0 'RD and *WR active low pulses 0 1 R/*W status and high E pulse 1 1 R/*W status and low ● DS pulse

CPAF1	8	Port A I/O or A0–A7
		CPAFI =0, Port A pin is I/O
		CPAF1=1, Port A pin is Ai (i is between 0 and 7)

CPACOD	8	Port A is CMOS or Open Drain Output
		CPACOD=0, CMOS output
		CPACOD=1, open drain output

CPBF	8	Port EI is I/O is *CS0–*CS7
		CPBF1 =0, Port B pin is *CSi (i is between 0 and 7)
		CPBF1=1, Port B pin is I/O

CPBCOD	8	Port B is CMOS or Open Drain Output
		CPBCOD=O, CMOS output
		CPBCOD=1, open drain output

CPCF	8	Port C A16–A18 or *CS8–*CS10
		CPCF=0, Port C pin is Ai (i is between 16 and 18)
		CPCF=1, Port C pin is *CSi (i is between 8 and 10)

CADLOG	4	Al 6–A19 Address or Logic Input
		CADLOG=0, Port C pin or A19/*CSI is logic input
		CADLOG=1, Port C pin or Al 9/*CSI is Ai (i is between 16 and 19)

Figure 5—*The vast number of configuration* bits *in* the PSD can make ***programming it*** a ***nontrivial task*** unless you have tools designed *for the job.*

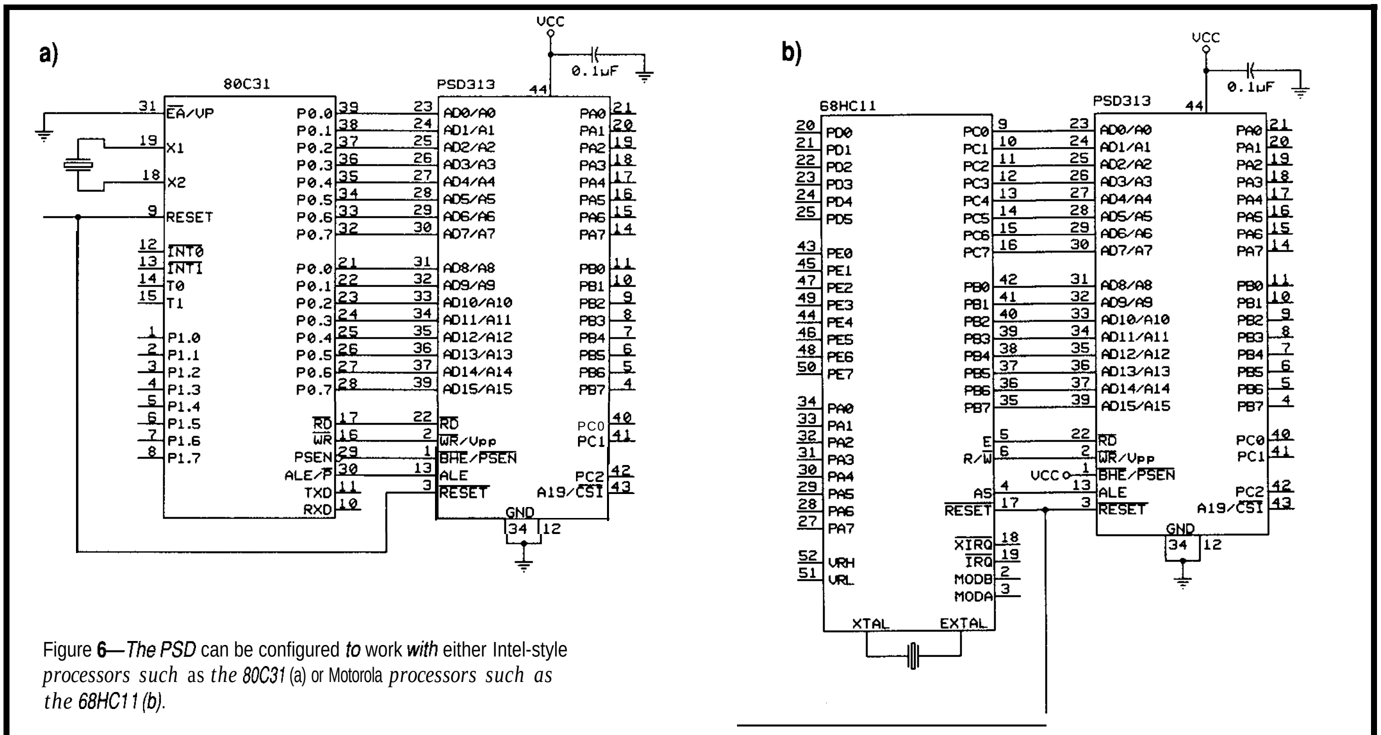


Figure 6—*The PSD* can be configured *to* work *with* either Intel-style *processors such* as *the 80C31* (a) or Motorola *processors such as the 68HC11 (b).*

PRICE PROVISO

No doubt the PSD is a neat chip, but success depends on a variety of factors that ultimately boil down to a simple question-is it a "good deal?"

The question may be simple, but the answer isn't since the optimality of a particular PSD-based design is quite application dependent.

For instance, a minimum system using a 16-bit multiplexed bus CPU (like the '186 or '196) takes full advantage of the PSD's on-chip latches and 16-bit access option. In this case, the PSD solution is far more streamlined than a traditional byte-wide memory design which would call for two latches, two 8-bit EPROMs, two 8-bit SRAMs, and some TTL/PAL glue logic.

On the other hand, nonmultiplexed CPUs are not as well served since the separate data bus consumes port pins (both port A & B in a 16-bit design, leaving only the 3-bit port C).

Another consideration is sourcing, the sole nature of which was surely a factor in the demise of the earlier combo chips. Besides the well-known effects of competition on price, supply reliability shouldn't be overlooked. As the old saying goes, "!@#$ happens." A friend of mind recalls the day long ago when, as a young chip marketer, he got a call from a frantic sole-sourced customer that went something like, "...our overseas guys just called and said they're roasting marshmallows at your factory-where are my parts!" "Fear and loathing" aptly describes the mood of customers who get hung out to dry. Fortunately, WSI recently announced that the PSD is to be second sourced by Philips, so you can design it in and still sleep at night.

Yes, priced from $7.92 (PSD311, 200 ns) to $11.21 (PSD313, 120 ns) in 100s, the PSD does command a higher price per bit than commodity memory chips. However, if you add up the cost of the equivalent EPROM, SRAM, latch, glue, and so forth, you'll see the PSD isn't out of line at all. In fact, the premium is likely offset by PCB cost and reliability improvements associated with fewer chips, traces, and pins. For example, compare a '196+PSD setup that needs only 112 pins versus the 200 or so needed for a traditional '196+(2)EPROM+(2)SRAM+glue design. The PSD solution is also smaller and lighter and thus especially well-suited for portable applications.

All in all, I think the PSD successfully walks that fine line between being "too little too late" and "too much too soon." It fills the gap between the high-end single-chip CPUs, and the typical half-dozen or so commodity chip solution.

Unlike disco and bell bottoms, it's time to give "combo chips" like the PSD another chance. New end graph WSI isn't standing still either. Tune in next month for a look at their brand new line of bigger and better PSDs. ❑

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

CONTACT

WaferScale Integration, Inc.
47280 Kato Rd.
Fremont, CA 94538
(510) 656-5400
Fax: (510) 657-5916

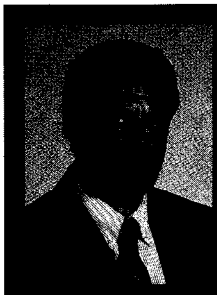
Philips Semiconductors
8 11 East Arques Ave.
P.O. Box 3409
Sunnyvale, CA 94088-3409
(408) 991-3737

416 Very Useful
417 Moderately Useful
418 Not Useful

Small Memories

EMBEDDED TECHNIQUES

John Dybowski



Have you ever needed a few hundred bytes of RAM, but didn't have the board space or budget for another full-size chip? Check out some offerings from Dallas Semiconductor and Signetics that could make your life easier.

Many embedded applications require only small amounts of read/write memory. In most cases, the required memory comes contained within the microcontroller that you're using. Even some fairly elaborate applications simply don't require the acquisition and processing of significant amounts of data. These applications typically fall into the domain of small single-chip controllers. Sometimes, however, you need more storage capacity than a particular controller provides. In this case you may be forced to add external memory in the smallest denomination available to meet your design objectives. Now, although necessary to completing the task at hand, no one will argue that such an endeavor constitutes a singularly uninteresting topic. And, anyway, how much can you say about a 2K-by-8 RAM chip?

Fortunately that's not the kind of memory I plan to talk about here. The types of small memories I will discuss, unlike the 2K-by-8 RAM chip, have some rather interesting characteristics. This interest stems not so much from the memories themselves, but from the way they can be used, and the way they are connected to the host controller. It makes sense to use a serial interface to keep the package size under control since doing otherwise would defeat the whole idea. This, of course, results in a small memory in a large package. Obviously a pointless exercise.

In past articles I've covered various two- and three-wire serial RAM and E^2PROM devices as well as serial real-time clocks that also happened to include varying amounts of built-in RAM. I will try to avoid being repetitious and briefly describe some two- and three-wire RAM chips that I haven't covered before. Then I'll focus on some memory devices that take interconnection economics to the limit; devices that carry all their communications over a single wire.

CLOCKED MEMORIES

Although E^2 devices have the rather desirable attribute of retaining data with no power whatsoever, there

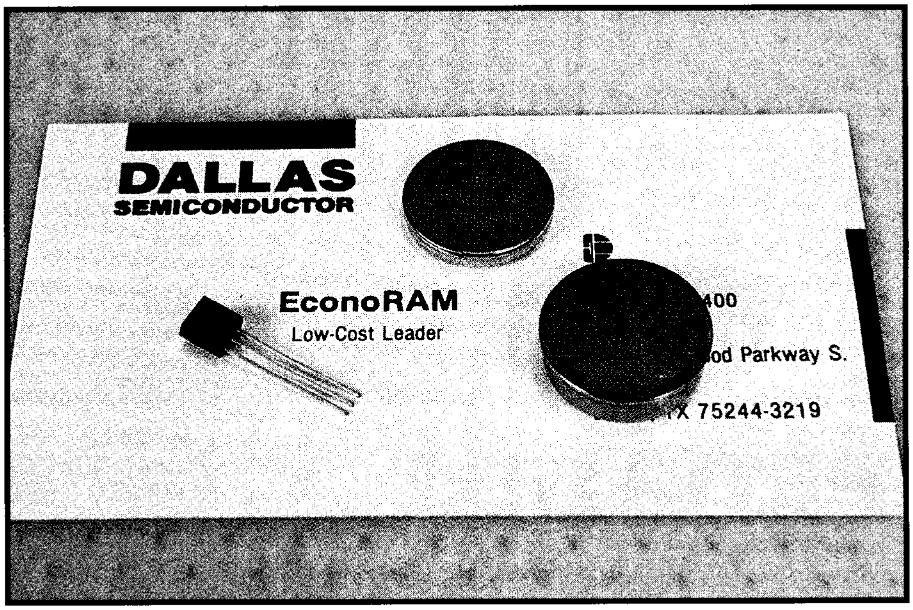
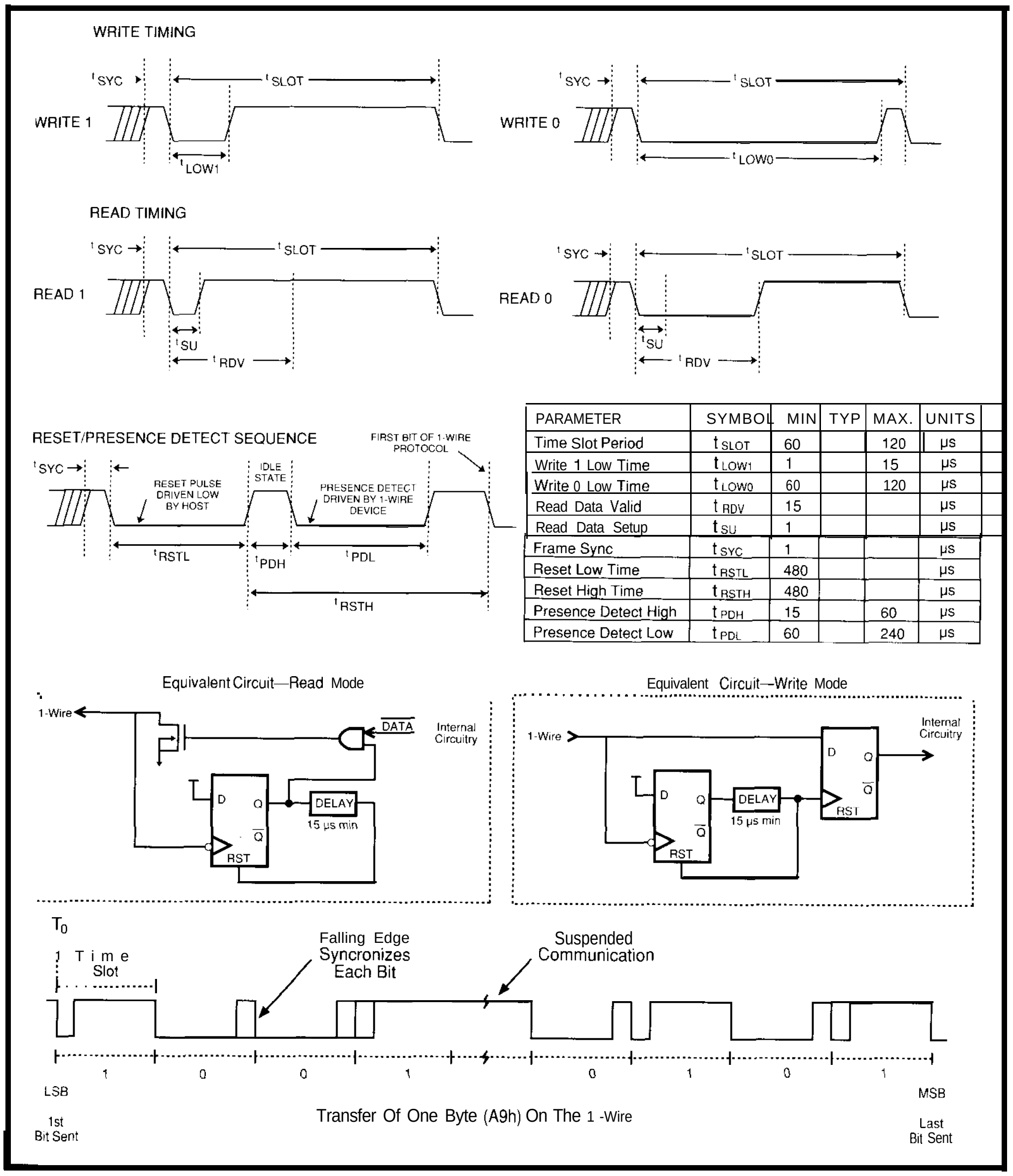


Photo I-Da//as Semiconductor has two types of touch memory packages available: one *intended for fixed PCB mounting (if resembles a transistor package) and the other known as a transportable container. These stainless steel holders resemble lithium coin cells.*



PARAMETER	SYMBOL	MIN	TYP	MAX.	UNITS
Time Slot Period	t_{SLOT}	60		120	µs
Write 1 Low Time	t_{LOW1}	1		15	µs
Write 0 Low Time	t_{LOW0}	60		120	µs
Read Data Valid	t_{RDV}	15			µs
Read Data Setup	t_{SU}	1			µs
Frame Sync	t_{SYC}	1			µs
Reset Low Time	t_{RSTL}	480			µs
Reset High Time	t_{RSTH}	480			µs
Presence Detect High	t_{PDH}	15		60	µs
Presence Detect Low	t_{PDL}	60		240	µs

Figure **1**—*I he* SIX *definedphases of a one-we transaction include reset, presence defect, bus command, device command, data transfer, and CRC. During data transfer, a self-clocking scheme is used.*

is a wear-out mechanism that you must contend with when using these parts. The limiting factor associated with E^{2}PROMs is the write cycle endurance. This is defined as the number of times you can write to a single bit before it stops working. Although progress has been made in this regard, and devices with write cycle endurance in the 100,000 range are common, some applications will simply not tolerate this limitation.

It's easy to determine if an E^{2}PROM is an appropriate fit for a particular application. But you must keep in mind that a firmware anomaly, or a system upset, can possibly lead to unintentionally performing a lot of write operations that can quickly lead to the part's destruction. This is not so much a problem when using serial devices, since the access method itself involves some very specific data transitions and timing sequences that narrow the window on such a catastrophic event. That's not to say there are any guarantees, but the failure mode would require the code to enter into a loop that continually, and explicitly, wrote to the poor memory device. Because of the protocols in place in these kinds of systems, this is an extremely unlikely event.

The situation is more touchy with parallel devices where the program can inadvertently enter into a tight "write loop" without much difficulty and consume the E^{2}PROM in a matter of seconds. In any event, there are applications that do require deliberate, and frequent, updating of nonvolatile parameters, and for these a RAM-based approach makes the most sense.

If your design already supports a three-wire serial device such as the Dallas DS 1202 real-time clock, and you find that its 24 bytes of nonvolatile RAM is not sufficient for your needs, then you may consider Dallas's DS 1200 serial RAM chip. Access to the RAM is via a three-wire interface scheme that is electrically identical to that used in the DS1202 RTC and is composed of \RST (reset), CLK (clock), and DQ (a bidirectional data) lines. The cost of adding this part to an existing design involves allocating an I/O pin to function as the RAM's reset line since the data and clock lines can be shared with the DS1202.

Possessing 1024 bits of RAM organized as 128 bytes, the DS1200 provides the mechanisms to access this RAM either randomly or sequentially. Random addressing lets you single out a particular memory location to read or write. The sequential addressing mode allows you to move data in and out of the chip in bursts with address incrementing handled automatically by the internal address generator at the completion of each byte transfer. The use of this burst capability is particularly useful since the overhead associated with each access amounts to 24 clocks. These overhead clocks serve to transfer the address and command information prior to any actual operations on the chip's internal RAM area.

Available in a 10-pin DIP package, the DS 1200 provides separate power pins for the main (5-volt) supply as well as the battery connection that is

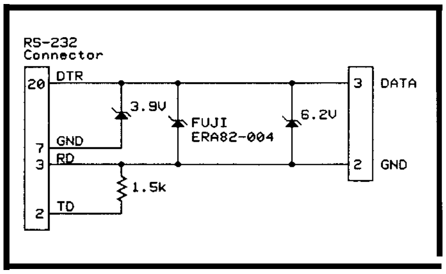


Figure 2-A *standard RS-232 port can be configured to interface with the Dallas Semiconductor one-wire devices.*

required for attaining nonvolatility. This dual supply connection saves you the external, low dropout, mixing diodes and is a feature I'd like to see more of, especially on chips that have pins designated as no-connects.

If I^2C is your preference then you may wish to consider the PCF8570 or PCF8571 RAMs from Signetics. These parts provide 256 and 128 bytes of RAM, respectively. Unlike the DS1200 and other 3-wire devices, I^2C imposes no penalty in the way of requiring extra pins when new devices are added to the bus. The chip selection is handled entirely through the protocol exchange over the data (SDA) and clock (SCL) lines. The device address consists of two portions, where the four upper bits comprise a fixed base address with the three low-order bits providing a user-programmable field that can be set by jumpering pins on the IC. Presumably, with this arrangement, you could have up to eight of these RAMs on the I^2C bus at the same time, but depending on the system configuration this may not necessarily be true.

It turns out that certain other common device types such as RTCs and E^2PROMs share the same base address as these RAM chips. This reduces the amount of RAM devices that can simultaneously be accommodated. Worse, the larger E^2PROMs use some of the bits of the programmable part of the address for paging (due to the limited scope of the I^2C's addressability), which further reduces the actual number of bus members that can be accommodated at certain base addresses. Luckily, this problem has obviously been encountered and solved; the PF8570C defines a different base address than the PCF8570, PCF8571, and the other contending peripheral chips.

SPARTAN BIT FIELDS

By now, my feeling about all the proprietary three-wire serial buses on the market is probably well known. To summarize: I think the idea of having a serial standard that requires a chip select (or reset) line for each added peripheral is, well...rather restrictive. In all fairness, there are a lot of applications where the extra wires required by such an approach are not a problem. Nonetheless, for any nontrivial serial peripheral array, the chip select lines do tend to add up. It all boils down to your perspective, your intentions, and how many I/O pins you have handy.

Another irritating piece of fallout associated with the use of a proprietary interface, especially when using parts from various manufacturers, is the fact that often you will need multiple (mildly tweaked) firmware bit bangers to cope with the slight bit stream variations that these parts want to see. A well-thought-out serial bus like I^2C is clearly superior. If only all the peripherals we needed were available with such an interface.. .

It turns out that an apparently similar amount of effort went into

Listing 1-A *handful of low-level routines make writing code to access DS1990 one-wire ROM device easier.*

```
;Public entry points
        public  Rd_DS1990
;Defined I/O bit
;
DQ      equ     p1.7

;Internal data
        rseg    IDATA
SN_Buff ds      8

        rseg    CODE
;Local...
;1-wire initialization routine.
;output: cy=0 if 1-wire device is present

Init:
        push    0
        clr     DQ              ;emit reset pulse
        mov     r0,#250
        djnz    r0,$

        setb    DQ              ;float the line
        setb    c
        mov     r0,#125
i_loop:
        anl     c,DQ            ;device driving low?
        djnz    r0,i_loop

        pop     0
        ret
;Local...
;Routine to read a byte from a 1-wire device.
;output: data byte is returned in acc
Read&Byte:
        push    0
        mov     r0,#8           ;8 bits to read
rb_loop:
        setb    DQ              ;idle the line
        nop
        clr     DQ              ;start time-slot
        nop
        setb    DQ              ;release the line
        nop
        mov     c,DQ            ;read the data bit
```

(continued)

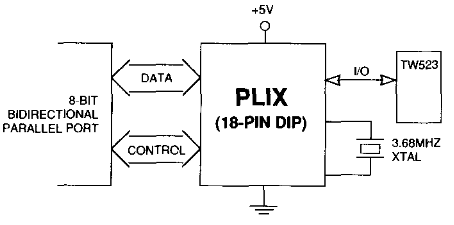
Listing 1-continued

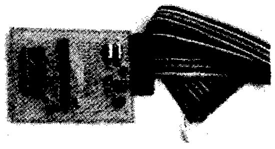
```
        rrc     a

        push    0
        mov     r0,#30
        djnz    r0,$            ;pad out time-slot
        pop     0

        djnz    r0,rb_loop      ;loop until done
        setb    DQ              :termiate time-slo
        pop     0
        ret
;Local...
;Routine to write a byte to a l-wire device.
;input: byte to write is in acc

Write-Byte:
        push    0
        mov     r0,#8            8 bits to write
wb_loop:
        setb    DO               idle the line
        nop
        clr     DQ               start time-slot
        nop
        rrc     a
        mov     DQ.c             emit the data bi

        push    0
        mov     r0,#30
        djnz    r0,$            ;pad out time-slot
        pop     0
```

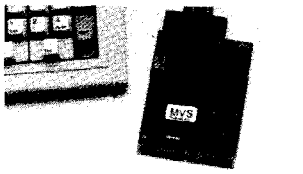
(continued)

crafting yet another serial interface. This interface method, devised by Dallas, is based on just one signal wire (and a return lead). This method is used in their low-cost, one-wire RAMs and ROMs as well in their touch memory product line. Defining not only the actual signaling between the controller and the attached peripherals, this methodology also encompasses such issues as multidropping multiple slave devices, sequential transfers of data, and a clever address searching scheme for determining what devices are actually present on the wire at any given time. If you already like the I²C concept, then read on and perhaps you will agree with me that maybe there's room for two serial peripheral standards: a two-wire standard and a one-wire standard.

Operating a serial interface over a single wire is not uncommon but there are surrounding issues associated with such an approach. Remember that this single wire must do more than transport bidirectional data. It must also allow various higher level func-







tions such as the passing of command, status, and address information. Random or sequential transporting of data must also be accommodated.

It should be obvious by now that a very major difference exists between a serial method that uses a discrete clock signal, and the self-clocking mode necessitated when using just one wire. The main difference is the discretely clocked scheme is synchronous by nature [and generally can operate down to DC), but the one-wire interface employs an asynchronous technique and mandates that strict, although well defined, timing parameters be adhered to. If the controller cannot keep up with these restrictions, device synchronization will be lost and the transfer sequence will abort. The situation is eased by the fact that synchronization is accomplished for each bit cell and as a result communications can, in fact, be suspended between bit transfers.

Dallas describes everything from detecting when devices connect to the bus, the actual bit timing for registering ones and zeros, command sequences, the definition of the various transaction sequence phases, and even a file structure. As with most things, you can make it as simple or as complex as you desire. Electrically it's another matter; the interface consists of nothing more than a single wire that is passively pulled up to 5 volts via a 4.7k resistor. Due to the multidropped, bidirectional nature of the bus, all bus members (including the master controller) must use open-drain drivers to prevent electrical contention problems. And with the careful management of line timing, up to fifteen feet of wire can be driven under most operating conditions.

Let me proceed by first describing what to do at the bit level to transfer ones and zeros between the master controller and the slave devices.

First of all, data-bit transfers occur in discrete time intervals called *time slots* which are started by a falling edge on the line as driven by the master controller. This sync pulse must be held for at least 1 us.

This protocol defines ones or zeros by short or long low times within a

Listing I-continued

```
        djnz    r0,wb_loop      ;loop until done
        setb    DQ              ;termiate time-slot
        pop     0
        ret
;Local...
;Calculate CRC
;clear CRC register (r7) before initial call to this routine.
;input: byte to calculate into CRC is in acc
;output: new CRC is in CRC register (r7)

;local storage
CRC     set     7
Calc_CRC:
        push    0
        push    acc
        mov     r0,#8
cc_loop:
        xrl     a,CRC
        rrc     a
        mov     a,CRC
        jnz     ccl
        xrl     a,#18h
ccl:
        rrc     a
        mov     CRC,a
        pop     acc
        rr      a
        push    acc

        djnz    r0,cc_loop
        pop     acc
        pop     0
        ret
;Public...
;Read a string from the DS1990 silicon serial number
;input: rl points to destination buffer (iram)
;output: if acc=0 then:
; the destination buffer contains 8 bytes where:
; byte 0 contains the family code
; bytes 1 6 contain the serial number
; byte 7 contains the CRC
;
Rd_DS1990:
        call    Init
        jc      rds_error       ;jump no device

        mov     a,#0fh          ;read rom command
        call    Write-Byte
        mov     r7,#0           ;clear CRC register
        mov     r0,#8           ;8 bytes to read
rds_loop:
        call    Read-Byte
        mov     @rl,a           ;store next byte
        call    Calc_CRC        ;compute CRC
        inc     rl
        djnz    r0,rds_loop     ;execute read loop
        mov     a,r7            ;get computed CRC
        jnz     rds_error       ;... should be 0
;normal exi
rds_exit:
        ret
;error exit
rds_error:
        mov     a,#-1
        ret
```

time slot. A one is registered by returning the line to a high state within the sample window. A zero is written by holding the line low for the entire duration of the sample window.

Ideally, the master controller should read the data line 8 µs from the beginning of the time slot. When the slave device transmits data, the master controller achieves synchronization by pulling the line low during the 1-µs sync time. The slave device controls the line during the remainder of the sample window. Communications can be suspended for any length of time between time slots by the master controller leaving the line high. In this transmission scheme, the data bits are transmitted LSB first and MSB last.

Now, with the low-level bit transfer out of the way, I'll describe the various second-level transaction phases in order to help you gain a better understanding of how the one-wire protocol works.

The six defined phases of the one-wire transaction sequence are reset, presence detect, bus command, device command, data transfer, and cyclical redundancy check. The six transaction phases are discussed below.

*Reset-In order to ensure that all slave devices are listening, the reset sequence is initiated by the master controller. The sequence is started by holding the line low for 480 µs. Following this interval, the master controller allows the line to return high for the 480-µs presence detect interval.

*Presence Detect-Following the reset interval, the slave devices come out of reset. After 15 µs from the rising edge, a slave will drive the line low for 60 µs to signal their presence to the master controller. The duration of this response allows the master controller to distinguish this event from noise on the line.

•One-Wire Command-Once the master controller has established the presence of slave devices on the line, it may issue one of four bus-level commands as follows:

➛Pass Through Mode (CCh)—This command allows the master to access the slave directly without specifying its 64-bit ID number. This command is only useful when there is only one slave present on the line.

➛Read ROM Data (33h or 0Fh)—This command allows the master to read the slave's 8-bit family code, 48-bit serial number, and 8-bit CRC. As with the previous command, this command can only be used if there is just a single slave on the line.

-*Match ROM Data (55h)—This command allows the master to select a specific slave. The master sends out a 64-bit number that includes the family code, the 48-bit serial number, and a CRC. All bits must match for this command to be accepted by the slave.

➛Search ROM Data (F0h)—The use of this command allows the master to interrogate the slaves in order to determine the IDs of all attached slaves. This command functions as a repetition of a three-step sequence that consists of reading a bit, reading the complement of a bit, and writing a bit. (Remember that with their open-drain drivers, the slave devices can only drive the line low.) This operation is performed over the ID bits of each slave until all of the slave IDs are known. As the ID bits are determined in sequence, the progression is controlled by the values that are written as select bits (thereby incrementally selecting slaves). Briefly, the result of doing a read of a true and complemented bit on the multidrop line can yield the following values and their associated assumptions:

✱00—There are still devices on the line that have conflicting bits in this position.
✱01—All devices on the line have a 0 bit in this position.
✱10—All devices on the line have a **1** bit in this position.
✱11-No devices are present.

*Device Command/Data Transaction-Following the transmittal of the device-specific command, a data transfer to or from the slave device takes place. This transfer is carried out in accordance with the host-issued device command.

*CRC-The master controller computes a CRC value over the data as it is transmitted or received. The CRC is part of the permanent ID number that is contained within ROM devices and is managed by the master controller for RAM-based data. In either case, if the CRCs match, the data transfer is assumed to be error free.

Figure 1 describes how the reset and the presence-detect phase operates as well as how the line is managed for sending and receiving ones and zeros. The general principle behind how the internal self-clocking scheme is also shown pictorially here.

Listing 2—*It's possible to communicate with a one-wire device from a high-level language through a standard serial port.*

```
/* Transmit 8 bits onto the 1-wire data line and receive 8 bits
concurrently. The global variable 'com_port' must be set to the
com port that the COM port adapter is attached to before calling
this routine. This port must also be set to 115200 bps, 8 data, 1
stop, and no parity.  This routine returns the uchar 8-bit value
received. If it times out waiting for a character then 0xFF is
returned. */

uchar TouchByte(uchar outch)
{
     uchar inch=0,sendbit,Mask=1;
     uint SPA:
     uint far *ptr = (uint far *) 0x00400000;
     ulong far *sysclk = (ulong far *) 0x0040006c;
     ulong M:

     /* get the serial port address */
     SPA = *(ptr+com_port-1);

     /* Initialize the time limit */
     M = *sysclk + 2;

     /* wait to TBE and TSRE */
     do {} while ((inportb(SPA+5) & 0x60) != 0x60);

     /*flush input */
     while ((inportb(SPA+5) & 0x1))
          inportb(SPA) ;

     /* get first bit ready to go out */
     sendbit = (outch & 0x1) ? 0xFF : 0x00;

     /* loop to send and receive 8 bits */
     do
     {
          outportb(SPA,sendbit);  /* send out the bit */

          /* get next bit ready to go out */
          Mask <<= 1;
          sendbit = (outch & Mask) 0xFF : 0x00;

          /*shift input char over ready for next bit */
          inch >>  1;

          /* loop to look for the incoming bit */
          for (;;)
          {
               /* return if out of time */
               if (*sysclk > M)
                    return 0xFF;
               if (inportb(SPA+5) & 0x01)
               {
                    inch |= ((inportb(SPA) & 0x01) ? 0x80 : 0x00);
                    break
               }
          }
     } while (Mask);
     return inch; /* return the input char */
}
```

ONE-WIRE PACKAGING

Dallas's one-wire products are available in two types of packages: those intended for fixed PCB mounting such as TO92 and SOT223 packages, and transportable containers such as the R3 and F5 metal cans. These R3 and F5 chip carriers are stainless steel holders that physically resemble small lithium coin cells. The stainless cans (called *touch memories* by Dallas) provide the data signal connection on the inner face with the rim used for the return.

RAM- and ROM-based devices are available in both of these packaging configurations. The ROM consists of a factory-lasered 64-bit code that is made up of an 8-bit family code, a 48-bit serial number, and an 8-bit CRC. These ROM devices are inexpensive and can be used to provide serialization and identification information for various types of electronic instruments and equipment. Although often necessary, this type of permanent ID functionality is frequently hard to come by economically, especially when using small 4- and 8-bit controllers with masked ROMs. The Dallas ROMs satisfy this need by providing a unique code in each device-no two are alike.

Perhaps one possible alternative to this serialization problem (for moderate-quantity runs) would be to use OTP PIC processors, where the programming equipment provides direct support for sequencing a serial number with each programming operation. Using this type of a ROM in a metal carrier could also provide a portable alternative to other read-only auto-identification methods such as bar codes and magnetic stripes.

One-wire RAM devices are centered around the same basic ROM core and provide anywhere from 128 to 512 bytes of read/write memory. Some parts provide a single RAM address space whereas others are partitioned such that both password secured RAM areas (for sensitive data), and non-secured RAM areas (for use as general scratch pads) are available. There's even a touch device that contains a built-in real-time clock along with the ROM and RAM.

BIT BLIPPING

Refer to Listing 1 for code that demonstrates what is required to pull off communications with these serial one-wire devices. So as not to complicate matters, I decided to use the simplest device in the touch memory family-the silicon serial number ROM-for this example. The program defines functions to initialize the attached device and determine its presence, tell it to dump its ID information, and to read and deposit this information in an eight-byte buffer. The CRC is also calculated and verified to determine if the transfer completed successfully. Finally, this success/failure information is conveyed back to the calling program.

The main routine first calls the line initialization function that emits a reset pulse then looks for a presence signal on the line. Once a device responds to the reset sequence, the read command, 0Fh, is transmitted via the write routine. Following the issuance of the read command, eight bytes are picked off the line using the read routine and stored to the data buffer, with each byte taking part in a cumulative CRC calculation. If the CRC checks okay, the routine returns to the caller with a completion code indicating success.

As you can see there's not much involved in reading from a one-wire device. You will probably also notice that some of the routines are somewhat rudimentary and could use further refinement. For example, the reset/presence detect routine does not check for a shorted line and only needs to see the line at a low level for one sample period before deciding that a device is connected to the line. Obviously more could be done to check for a stuck line and to enhance glitch rejection. Furthermore, interrupt masking is not handled at this level.

The issue of interrupts is a real concern because of the tight timing constrains imposed by the one-wire protocol. Obviously the timing would be wasted by the occurrence of an interrupt event if it happened at the wrong time.

In a real implementation, a number of different approaches could be taken to resolve this issue and the particular approach would have to take into account the operation of the system as a whole. If interrupts had to be serviced on an ongoing, although infrequent, basis, you could mask interrupts for the duration of the entire transfer sequence. If the interrupts occurred more frequently, the best you could hope for would be to mask the interrupts for each bit transfer, and unmask briefly between bits since, at this time, communications could be suspended without causing problems.

On the other hand, if the interrupts occurred less frequently but randomly and had to be serviced in a timely fashion, there would be no choice but to let it all hang out and not mask the interrupts at all, instead relying on the CRC test to catch corrupted transactions. In this case, the transfer would be continually attempted until successfully completed or until a retry count expired.

BIG AND LITTLE

Having seen the timing constraints involved in communicating with one-wire devices, you may be thinking these small memories would be best serviced using small controllers that are good at processing I/O events in real time. This assumption, although not unreasonable, is one that is not true. After having demonstrated how very small controllers could drive large memory arrays last month, it would be a shame not to turn the tables and connect one of these minuscule one-wire memories to a real processor or to an actual computer for that matter.

It turns out that Dallas supports the one-wire interface on a PC platform and similar tricks can be used to link these devices to most processors using a UART and an RS-232 serial port. Figure 2 shows how a standard RS-232 port can be coerced to electrically accommodate standard one-wire devices using nothing more than a handful of passive components.

The bit manipulations get a bit more involved. If the host has a UART that can operate at up to 115,200 bps, the time-critical signaling can be handled almost entirely in hardware by monkeying with the UART in ways that it was not intended to be used. Of course, you realize this amounts to getting your hands dirty and means you have to be able to operate down at the register level. Listing 2 illustrates how to get the low-level control using a high-level language. The function shown is capable of simultaneously reading and writing eight bits from and to a one-wire device via an RS-232 port.

If we subscribe to the logic that less is more, then we can conclude these one-wire memories amount to a lot. The fact that these devices come in small packages and employ a simple interface can be deceiving since what may not be immediately apparent is the amount of engineering that is actually involved here. This apparent economy of resources has a balance, which comes in the form of ideas. If you want to know more, check out the volumes of information on this subject produced by Dallas Semiconductor. ❑

John Dybowski is an engineer involved in the design and manufacture of hardware and software for industrial data collection and communications equipment.

IRS

419 Very Useful
420 Moderately Useful
421 Not Useful

PATENT TALK by Russ Reiss

Games, for both children and adults, represent ideal applications for small, embedded microprocessors. A micro can often replace myriad mechanical contraptions while offering lower cost, smaller size, and improved performance. This month, I have rounded up nine patent abstracts on games that employ micros. These range from classics like the TI Speak-n-Spell to a couple that represent fundamental technology applicable to many types of games. I think you will see how microprocessors can not only bring the latest technology into the home or amusement center, but can also perform old tasks in better ways.

One of the premier electronic games has been the Texas Instruments Speak-n-Spell. While voice synthesis on computers appeared for a long while to be a solution looking for an application, the TI speech synthesizer employing advanced LPC encoding brought the familiar, completely human-like voice into nearly every home. Abstract 1 is the most recent patent I could find on this technology. Referenced patent 4,516,260 dated 1985 covers the same area. All the other references are to devices such as Federal Screw Works' landmark synthesizer, speaking calculators, and so forth.

Abstract 2 presents one of the critical patents for yet another household word-Nintendo. This patent-on their memory cartridge-represents the key by which clones of Nintendo were kept at bay. The earliest references I could find to their game cartridge are a pair of design patents D292,399 and D294,020 dated 1987 and 1988, respectively. Besides offering market protection for their game, this mechanism also offers the unique feature of permitting the user to continue a game at a later time from the same point where he left off earlier.

The next two abstracts (3 and 4) from SMS Manufacturing represent the same game. The designers here have made a somewhat "flat" video poker game more interesting by incorporating the three dimensional action of bouncing balls. But, as you can see from the wording, the first patent emphasizes the particulars of their specific poker game while the other one attempts to generalize the patent to encompass a wider range of applications. This is a common technique in patents where one wants to attempt both specific and general protection. A reading of the abstracts points up how a single, simple microprocessor coordinates a variety of user inputs, sensors, and actuators. As is so often the case, incorporation of a video display provides a dramatic, yet low cost, user interface.

The next group of four abstracts (numbers 5 through 8) represents a varied collection of games which use microprocessors. I have attempted to present a wide range of different types of games. In order to conserve on space, I've eliminated all but the pertinent information from the listings. What is common among all these games is that the single microprocessor integrates many functions associated with user control, game dynamics, and feedback of results. The use of a microprocessor not only simplifies the design, but also permits modes of operation not otherwise feasible. For example, in some games the computer may act as an opponent. It also serves as a source of randomization, score keeping, and often can control multiple games from a single

1

Patent Number	4,970,659
Issue Date	1990 11 13
Inventor(s)	Breedlove, Paul S.; Moore, James H.; Brantingham, George L.; Wiggins, Richard H., Jr.
Assignee	Texas Instruments Incorporated
US References	3,371,321 3,470,321 3,870,818 4,016,540 4,022,974 4,059,272 4,060,848 4,179,584 4,215,240
US Class	3641513.5 381/51
Title	Learning aid or game having miniature electronic speech synthesis chip
Abstract	An electronic hand-held, talking learning aid is disclosed. The learning aid includes a MOS speech synthesizer chip having an active surface area on the order of 45,000 square mils. The disclosed speech synthesizer chip includes a digital lattice filter, a voiced/unvoiced excitation circuit, a speech parameter interpolator, an input parameter decoder, a digital-to-analog converter, and associated timing circuits. The learning aid is also provided with a microprocessor which functions as a controller for controlling the operation of the unit. A small speaker is driven by the digital-to-analog converter on the speech synthesis chip and a keyboard and display device are strobed by the microprocessor controller. Features include modes in which a speech synthesizer recites instructions or questions to the operator who must properly respond.

PATENT TALK

2

Patent Number	5,014,982
Issue Date	1991 05 14
Inventor(s)	Okada, Satoru; Nishizawa, Kenji
Assignee	Nintendo Company Limited
US References	4,095,791 4,120,030 4,149,027 4,218,582 4,352,492 4,368,515 4,383,296 4,384,326 4,386,773 4,432,067 4,442,486 4,446,519 4,454,594 4,458,315 4,462,076 4,471 ,163 4,485,457 4,492,582 4,500,879 4,562,306 4,575,621 4,575,622 4,620,707 4,738,451 4,752,068 4,757,468 4,858,930
US Class	2731435 2731856 3641410
Title	Memory cartridge and game apparatus using the same
Abstract	A memory cartridge is attachably/detachably loaded to a game machine which includes a microprocessor and a working RAM provided in association with the microprocessor. When the memory cartridge is loaded to the game machine, a program ROM included in the memory cartridge is connected to the microprocessor and the working RAM through data bus and address bus such that the data can be communicated between them. When a game is terminated, a password is generated in accordance with a program stored in the program ROM in advance by modifying game status data including data of score, power, or energy, and the stage number which are stored in the working RAM. The password thus generated is displayed on a display. When the password is inputted by a player prior to consecutively starting a succeeding game in accordance with the display, by restoring the inputted password, the game status data is set in the working RAM, whereby the microprocessor consecutively starts the succeeding game from the game status at the timing when the preceding game was terminated.

controller. This is an area where the computing power of the microprocessor is not of great concern while cost is. In some cases, only simple digital I/O is required, and speed is not paramount. The availability of simple, inexpensive, and low-power controllers like the Microchip PIC series should permit such designs to flourish.

Finally, abstract 9, from Williams Electronic Games Inc., relates to the ever-popular pinball game. This patent attempts to achieve broad coverage for its concept. It represents an interactive playfield-the surface over which the pinball moves and which, in effect, defines the behavior of the game. Their unique use of an image of a human head for the "preferred" playfield is but one embodiment of their broad patent. I particularly like the moving eyes which track the course of the ball on the playfield. Keep an eye out for this one at your local amusement center.

3

Patent Number	5,014,991
Issue Date	1991 05 14
Inventor(s)	Mirando, Salvatore V.; Lee, Walter
Assignee	SMS Manufacturing Co., Ltd.
US References	2,317,506 2,710,756 2,853,304 3,275,322 3,399,896 3,901,511 4,311,311 4,375,286
Title	Amusement game
Abstract	The poker amusement game includes five rubber balls and a playing field with twenty-five openings in which the rubber balls may reside. A microprocessor has inputs connected to a plurality of switches for determining the location of the five balls, a plurality of outputs connected to solenoids to be used for ejecting the balls from the openings, and an output connected to a video display for displaying messages to the player concerning the play of the game, the points received, etc. When the microprocessor detects that a coin has been inserted and that the DEAL/DRAW switch has been pushed, the locations of the five balls are sensed and the corresponding ejectors are energized to eject the balls in a manner that causes the balls to randomly bounce in the playing area. Balls coming to rest in the openings cause switches to close which in turn, will cause the microprocessor to display on the video the identity of the poker card represented by the occupied openings. The player may discard balls by first activating appropriate DISCARD switches and then activating the DEAUDRAW switch. In response, the microprocessor energizes the corresponding solenoids to eject the chosen discarded balls from their openings. The final hand is evaluated when the discarded balls come to rest in their new positions. Tickets or tokens are awarded based on the value of the final hand.

PATENT TALK

4

Patent Number	5,014,988
Issue Date	1991 05 14
inventor(s)	Mirando, Salvatore V.; Lee, Walter
Assignee	SMS Manufacturing Corporation
US References	2,130,123 3,044,778 4,190,251
Title	Poker roll game
Abstract	An amusement/gaming device which provides means for the selective release of game pieces, such as balls, for replay. The amusement/gaming device comprises an array of positions, each position representing a score in a game such as draw poker. Each position includes a hole to hold a ball in that position and an optical detector to identify residence of the ball in the hole. An array of rotatable rods having fingers are mounted below the holes. Solenoids are mounted at the ends of the rods for selectively rotating two intersecting rods to release the ball allowing return of one or more balls for replay. A microprocessor control and a video display interact with the player.

5

Patent Number	5,009,419
Issue Date	1991 04 23
nventor(s)	Streeter, Willie L
Title	Microcomputer controlled rotation game
Abstract	A game device for development of coordination and concentration among children as well as adults is disclosed. The device includes a microprocessor mounted within a housing having indicia representing a pair of hands and a pair of feet. Each participant has access to four switches operable by the hands and feet. The microprocessor generates a sequence of tones or lights, each being associated with a hand or foot on the housing. A pair of arrows may be mounted on the housing to indicate whether the sequence is proceeding clockwise or counter clockwise. Each participant attempts to repeat the sequence by activating the proper switch shortly after each tone or light or both. Play continues until a participant makes an error at which point play ceases. In another embodiment, the game can be played to control the interaction of two participants with one participant generating a first rotation or random sequence to be repeated by the other participant.

6

Patent Number	5050,883
Issue Date	1991 09 24
Inventor(s)	Goldfarb, Adolph E.; Goldfarb, Martin I.
Assignee	Goldfarb, Adolph E.
Title	Self-contained competitive game for developing spatial sense in young children
Abstract	Electronics and a playing method stimulate abstract spatial-relations ability, particularly memory of abstract space, in youngsters-without requiring them to know or spell game commands, or to find keys on a typewriter-like keyboard. The game exploits the competitive instinct by rewarding ability to recall complex geometric abstractions, while yet encouraging play by those who lack that ability. Dedicated manual inputs are used by each player to enter moves-in the pure form of directions in which the player wishes to go, An audio speaker signals which player's move it is, and whether each attempted move is valid. A digital microprocessor is used to define a maze and each player's position in it, and to receive moves from the directional inputs, and to operate the speaker in reply to attempted moves. The processor has no functional connection with any device for displaying a direct pictorial representation of any part of the maze, and indeed no such direct picture is electronically developed or shown. The game does include, however, a playing board on which players can in effect map their own attempts to move through part of the maze-if they are willing to let other players see their maps.

PATENT TALK

SOURCE

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

MicroPatent
25 Science park
New Haven, CT 06511
(203) 786-5500 or (800) 648-6787

Russ Reiss holds a Ph.D. in EE/CS and has been active in electronics for over 25 years as industry consultant, designer, college professor, entrepeneur, and company president. Using microprocessors since their inception, he has incorporated them into scores of custom devices and new products. He may be reached on the Circuit Cellar BBS or on CompuServe as 70054,1663.

422 Very Useful
423 Moderately Useful
424 Not Useful

7

Patent Number	5067,079
Issue Date	1991 11 19
Inventor(s)	Smith, Jay, III; Fort, Jeffrey W.
Assignee	Smith Engineering
Title	Interactive audio baseball game
Abstract	In a microprocessor based interactive audio game for two users to play against one another, or for one user to play against the computer, the game having the ability to store and retrieve sets of statistical data corresponding to the performance of actual players, keys for the users to affect the state of the game, and an audio output for communicating the state of the game or the results of a game play to the users, the audio output for a particular game state or game play result being variable in order to maintain user interest. A plurality of visual indicators are arranged in a grid that represents the flight of a thrown ball through a vertical cross section of the strike zone. Visual indicators are sequentially flashed a predetermined number of times and in a predetermined pattern for a particular variety of pitch.

8

Patent Number	5020,806
Issue Date	1991 06 04
Inventor(s)	Martin, John
Assignee	Arachnid, Inc.
Title	Multiple target electronic dart game
Abstract	A multiple target electronic dart game having shared microprocessor that monitors and services operation of the electronic dart game. The multiple targets are mounted on opposite sides of a column which may be floor-standing or wall mountable. Affixed to the top of the column is visual display video monitor for providing players and observers with game scores and the like. The visual display video monitor is optionally capable of split-screen presentation. The novel design of the multiple target electronic dart game enables players to play dart games not previously available with single target dart games. Moreover, many components utilized by the controls are shared, thus enabling cost savings by eliminating substantial duplication of components.

9

Patent Number	5,149,094
Issue Date	1992 09 22
nventor(s)	Tastad, Gregory J.
Assignee	William Electronics Games, Inc.
US References	3,275,324 4,017,077 4,109,916 4,162,793 4,363,485 4,375,286 4,508,343
Title	Player-operated control system for pinball games
Abstract	A player-operated control system for pinball games includes a plurality of switch assemblies by which the game player communicates information regarding the course of play to a controlling microprocessor. The software in the microprocessor alters the course of play according to the player's command.

CONNECTIM E conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400/9600/14.4k bps
24 hours/7 days a week
(203) 871-1988—Four incoming lines
Vernon, Connecticut

This month, I've chosen a pair of message threads that deal with circuit manufacturing. The first one looks at a problem in manufacturing that is only going to get worse: what kind of solder and flux to use to avoid having to use flux cleaners that harm the environment. The alternatives are looking promising.

In the second discussion, we take a look at some PC board design techniques that help shield the circuit from external RF interference.

Environmentally responsible soldering

Msg#:16298
From: AL **DORMAN** To: ALL USERS

I just purchased another gallon of Kester 5235 flux remover to clean some boards. Last time I paid $18.00 a gallon. This time it cost me $70.00 a gallon. The store says it's because of new EPA regs. There goes the price of my boards.. Unless.. .I heard a while back of a solder that has water-soluble flux. You put the boards in a dishwasher to clean them! Is this true? Who sells this stuff. Is it NASA approved? Does it hold up to heat and/or moisture? Has anyone had any experience with this solder?

Msg#:16327
From: JOHN **MUCHOW** To: AL **DORMAN**

I don't know if you're looking for flux-core solder or bulk flux, but Kester has a few things that might be of interest to you. There are two RMA "No Clean" solder pastes, R-244 and R-239A, and two water-soluble flux solder pastes, R-587 and R-588. They have number 245 Low Residue (no clean) and 33 1 Organic Flux Core solder (water-soluble flux) in 11 to 3 1 gauge spools.

Their bulk No-Clean fluxes are numbers 95 1, 952, 960, 920-CXF, and 932-P4M. Their water-soluble fluxes are numbers 2224-25, 233 1, 233 1 -Zx, 2 120, and 2222.

If you need specs on the ones you're interested in, leave a note or you can request Kester's catalog, phone (708) 297-1600 (their main office). I've had great success with their No-Clean flux solder, #245.

Msg#:16438
From: AL **DORMAN** To: JOHN **MUCHOW**

No Clean? You mean there's no residue? Where does the gunk go that allows the solder to flow? I like my boards to look clean and shiny for my customers. 1'11 get a roll and try it out.

Msg#:16480
From: JOHN **MUCHOW** To: AL **DORMAN**

When the manufacturers say "No Clean," they really mean "You Don't Have To Clean if You Don't Want To" <g>. The no-clean solder fluxes are really just standard solders with very low solids content; they still leave a residue, but it's *significantly* less than that left by conventional solders. And there are no long-term problems with the residue if it's left on the board. I don't know of any cored solders available with water-soluble flux, so the no-clean might be your best bet.

Msg#:16478
From: JIM WHITE To: AL **DORMAN**

Both of the boards I currently have in production are soldered with "no clean" flux solders. One is a through-hole board which is waved. There is some residue, which is supposed to be water soluble for water cleaning. The specs on the flux says it will age well, but the slight tackiness makes me uneasy. The other board is SMT, and the "no clean" solder paste actually leaves a clear hard coating over the solder after IR reflow. This stuff I really like; it is like getting a little conformal coat (which the high-impedance sections like) free.

Msg#:16488
From: AL **DORMAN** To: JIM WHITE

Hmm. I wonder if it would do better or worse in the humidity of Florida where I have some boards installed. Would the coating ooz off the board or actually keep the humidity out. What does the SMT board's coating do when you hold it over a pan of boiling water (in steam)?

Msg#:16811
From: JIM WHITE To: AL **DORMAN**

I don't know. The stuff looks good, but I rely primarily on the manufacturing people's research. Both board vendors are large, high-quality manufacturers who have spent far more researching this stuff than I ever could. Also, their reputations and profit are on the line. I did read the product qualification reports which claim that these no-wash fluxes retain the correct properties for a long time, including temperature and humidity changes.

One guy I know has his business in Boca Raton. He has an adapter board which is assembled there. They used some no-wash flux (through-hole board) which grew some kind of fuzzy-looking crystal in the humidity. The flux in the solder went into solution then crystalized in the high humidity and temperature. The boards worked fine, the crystals were not conductive, but the boards were aesthetically unacceptable.

Msg#:17006
From: JOHN **HARTMAN** To: AL **DORMAN**

I don't know brand, but we use such a solder on our wave line. We used to have a dishwasher to clean up the boards! Now have a larger and more dignified washer.

We also have used water-soluble flux solder for hand solder. In my experience, it is harder to get a good joint, as it doesn't seem to clean as well as rosin. it also eats soldering iron tips, and smells horrid. The production line has no problems, however.

PC board design for **RFI** rejection

Msg#:16317
From: JOHN **MUCHOW** To: ALL USERS

I'm in the process of laying out a small PC board for use in a severe RFI environment. I'm new to PCB design and am using this board to teach myself the things I need to know to go on to more complex layouts.

The board only needs one side for traces and I was wondering if it was better to place a ground plane on the component side (by itself) or the solder side of the board, surrounding the traces. If on the component side, it would be connected to the ground net at each plated-through ground pin (there are 11 of them).

The device will be operated in an arena with dozens of 5-watt UHF and VHF walkie talkies and a few repeaters. Will either of these layouts provide a bit more resistance to RFI, or just more resistance to interference by its own components? I've read Coomb's "Printed Circuits Handbook" and Ginsberg's "Printed Circuits Design" and couldn't find an answer, though I learned a *lot* of other things I needed to know.

Msg#:16319
From: PAUL PETERSEN To: JOHN **MUCHOW**

To some extent it depends on what kind of circuit you're designing. If it's RF up in the megahertz region, then you'll want to consider impedance of copper runs, and so forth. Having a ground plane on the solder side adds capacitance. Actually, if you're worried about picking up stray RF interference, your best bet is to put everything in an RF-tight box using copper screen at the seals and overlapping metal joints. Metal-to-metal with sheet metal screws will only leak RF. Not knowing what your design is, I'd say try it the cheap way and if it works you're home, if not, you have an option to fall back on.

Msg#:16326
From: JOHN **MUCHOW** To: PAUL PETERSEN

My circuit is a small interface being used as a universal front end for several different electronic flash units we use. The interface lets us parallel several units together to simplify cabling. It's probably going to be in a plastic case because of the extraordinary abuse it gets; the small Serpac cases just bounce, no dents. I've only had one occasion where RFI has interfered with the operation of a couple of my hand-wired prototypes, but since its so easy to add ground planes and to backfill with copper, I thought I might as well do as much as I can to prevent that from happening again.

Msg#:16328
From: JOHN **MUCHOW** To: PAUL PETERSEN

Oops, I forgot to mention that it's strictly a low-frequency board; CMOS 555, MOC3023, 2N2222, 2N3906, and some caps and resistors, 9-volt battery powered. The unit sits on standby, drawing about 1 µA and powers up in a couple of microseconds when the input is shorted by the camera's shutter contacts (PC connection). It then fires the flash unit and powers back down. All in 10 milliseconds. The board will fire the flash for 30+ rolls of film a day for 5 years before you have to replace the battery (on paper that is, I'm changing batteries each year <g>).

Msg#:16356
From: JAMES MEYER To: JOHN **MUCHOW**

My experiences lead me to believe that board layout will not make very much difference in reducing the sensitivity of a circuit to externally generated RF. A closed metal box is 99% of the solution. You also have to be very careful with each and every wire that passes into the box. The best shielding is ruined if you connect an antenna to the circuit through a hole in the shielding. Any wire connected to your circuit from the outside must be considered to be an antenna and treated to remove the RF from it.

Msg#:16358
From: JOHN **MUCHOW** To: JAMES MEYER

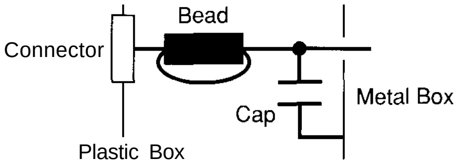
Thanks for the feedback, I'm looking into putting the circuit into a small metal box before assembling everything

CONNECTIME

into a larger plastic box (needed for aesthetic and abuse reasons). My biggest problem is the run of cable up to the unit which is usually 18-gauge zip cord, up to 1000 feet in length. How's *that . for an unwanted antenna problem <g>! We can't switch cable type since that's what every sport photographer uses to hook up his equipment and we have to be compatible with it, including using 2-prong female household receptacles as the input and output connectors! Amazingly, no one has been injured by plugging the wrong cord into the wall instead of a camera or flash pack.

Msg#:16526
From: JAMES MEYER To: JOHN MUCHOW

Ferrite beads and a small ceramic disc cap on each wire that comes out of the metal box should help. If the metal box were enough smaller than the plastic box, then you could put the connector on the plastic box, and add the filter just before the (short) wire went into the metal box as follows:



Msg#:16536
From: JOHN MUCHOW To: JAMES MEYER

In fact, there is a bit of room between the plastic case and the metal one I was considering. The connectors will have to go on the plastic case for aesthetic reasons, so your idea sounds like the best compromise. Does the wire have to pass through the bead more than once (is the attenuation greater if I do loop it?) because I'll need a large bead then, and I don't have . that* much room.

Msg#:16538
From: TODD NICHOLS To: JOHN MUCHOW

When you run that wire through a ferrite bead, you are making a toroidal inductor. Looping the wire increases the inductance (more turns around the "core"). That whole series-L/shunt-C circuit is a 2-pole low-pass filter, and when you increase the series inductance, you're just lowering the cutoff frequency. You can even select the inductance and capacitance for a given level of attenuation at some frequency. If you have some idea of the kinds of hash you have to deal with in the vicinity of your long-wire antenna :-), then you can design your input filters for a certain immunity level. Also, I echo a previous comment about a shield only being effective if you don't let EM1 in through some hole somewhere.

Have you considered using feedthrough capacitors? They are shaped like a small bolt (6-32 or so in size); drill a hole in your metal box, pop one in, tighten the nut, and you have a wire running through the box. The neat thing is the wire runs through a dielectric material, and there is capacitance from that wire to ground-just like the shunt capacitance in the proposed input filter. That's a single-pole low-pass filter. The microwave and RF industry use these a lot. They don't take up much room.

Msg#:16573
From: JOHN MUCHOW To: TODD NICHOLS

I heard of feedthrough caps, but have never used them. I like the idea.

The hash on my input line is pretty well spread out from DC to light<g>! I know I can't block it all, or even most of it, but it would be nice to block as much as I can without spending a fortune protecting these devices. They only cost me $30 or so and an afternoon to build. I don't want to spend $100 protecting them from RFI! They just don't misbehave enough to justify the expense.

Msg#:16635
From: JAMES MEYER To: JOHN MUCHOW

The wire *can* go more than once through the bead if it's big enough, but it doesn't *have* to. Within limits, the more turns you make through the bead, the more filtering. If you need more filtering, you could also just put *two* beads on the wire. Also be careful if you use bare wires. Some beads are conductive. I found out the hard way.

One good source of large and small beads is to tear up old switching type computer power supplies. You can find lots of good parts in dead supplies.

Msg#:16643
From: JOHN MUCHOW To: JAMES MEYER

I'll get a few beads and play with the scope to see how much of a difference more turns make, and how it compares to adding another bead. The computer power supply tip was very timely. A friend of mine just had his systems supply die two days ago and he hasn't thrown it away. It's mine tomorrow! Thanks!

Msg#:16936
From: PELLERVO **KASKINEN** To: JOHN **MUCHOW**

A tall challenge for any filtering you are talking! You say that the hash pretty much covers the spectrum of DC to light. Yet you would want to have your detector see an edge to provide the synchronization. That means you cannot slow down too much with the filtering!

Actually, as much as I like the feedthrough capacitors and ferrite beads in noisy environment, I have a word or two of caution about selecting and using them. I also may have a "real" solution. Let's see.

The feedthrough devices come in two classes, one is just what the name says, with capacitances up to several nanofarads. That in principle adds to whatever capacitance (probably tens of picofarads per foot) your cable has. The other class has less capacitance but also includes a ferrite bead structure. Sort of a distributed series inductance/ parallel capacitance. In other terms, such a thing is called a delay line. However, a delay line is not supposed to attenuate anything, so the ferrite is normally made quite "lossy." In other terms, it has a resistive element in the form of an eddy current path. But all of that still is quite high in the frequency band, several megahertz. Now, to the nature of your wiring and the noise signal.. .

If I understand, you have just plain unshielded, untwisted pair. And you have several of the receivers in parallel along this long transmission line. Yes, transmission line, and not so simple: you also have the third conductor in shape of the ground or the concrete structure or whatever the wires and boxes are laying upon. Thereby you have a ground-referenced antenna of sorts. Two things about this to keep in mind: Your signal should be *differential* while the noise induced into the two wires should be *equal* and *canceling* each other in your differential sense. And second, you have the impedance mismatches to count at every point where you put a receiver. Now, you might start seeing where I am coming to.

The feedthrough capacitors should be minimal in capacitance value, because they again represent an impedance mismatch in addition to potentially slowing your desired signal risetime too much.

The ferrite beads should be used as a common-mode filtering device, that is, both the lines entering the receiver boxes would go through the same opening in the toroid as a pair. This arrangement filters the common-mode noise, but leaves the signal pretty much untouched. It also leaves any noise that already has gotten into your differential domain untouched.

Toward the more proven techniques, you should try to match the line impedance as best you can. That means a terminating resistance of about 100 to 150 ohms at the very far end of the line and attenuating structures at the boxes along the line just like the cable TV systems have. Say something like 600 ohms in, rather than the same 100 to 150 ohms at each place. I think there would be no need to avoid impedance mismatches within each receiver.

Now, all this means that in essence you have to provide a great jolt into the cable as your signal. Say, you put in a pulse of 20 V and your impedance is the lower end of the above-mentioned range, 100 ohms. Then you have to provide a 200-mA pulse. If that does not mask the noise from the antenna, then hardly anything does!

To emphasize, try to keep the receivers as symmetrical as possible relative to earth (your third conductor] because the unsymmetry more than anything is a cause of hash pickup. Therefore, don't think one of the two wires in the cable is the "neutral side" and the other one the hot side. Build everything symmetrical and minimize capacitances and capacitance differences to the inner shielded box. Use optocouplers or whatever suitable devices for maximum benefit in this particular sense. I am somewhat skeptical about your ability to change the governing principles at this time in the project, but at least I have tried to enlighten the universe, haven't I?

We invite you call the Circuit Cellar BBS and exchange messages and files with other Circuit Cellar readers. It is available 24 hours a day and may be reached at (203) 871-1988. Set your modem for 8 data bits, 1 stop bit, no parity, and 300, 1200, 2400, 9600, or 14.4k bps.

ARTICLE SOFTWARE

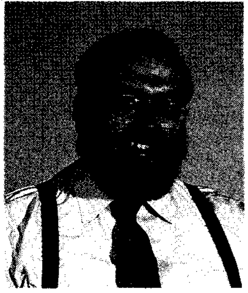
Software for the articles in this and past issues of *The Computer Applications Journal* may be downloaded from the Circuit Cellar BBS free of charge. For those unable to download files, the software is also available on one 360K IBM PC-format disk for only $12.

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) 875-2199. Be sure to specify the issue number of each disk you order. Please add $3 for shipping outside the U.S.

IRS

425 Very Useful 426 Moderately Useful 427 Not Useful

STEVE'S OWN INK



The Race for Power

It seems that about every 18 months we get a new processor. In my experience, most PC buyers can't justify buying a new computer every time a new generation comes along. Of course, there will always be technophiles who must own the latest platform, but the majority of us have hardly touched the performance envelope of the present generation before we are beseeched to join the latest development frenzy and instant obsolescence mentality.

New generations of processors appear so fast that the previous generation becomes "market obsolete" even before most software vendors have figured out how to write code for maximum utilization. In their race for market share, software companies can't afford to take the time to optimize existing code using enhanced instructions every time a new processor comes along. They just have to reformulate what they have to be "code compatible" and shove it out the door. Is this why "entry level" systems require 120-megabyte drives?

Even the compiler folks struggle to stay ahead of the eight ball. So what do we get? A bunch of '486s running 8086 code? 25 MIPS of NOPs? What good is a cache if the programs aren't cache aware and completely flush the cache every few instructions? Sure my old programs run faster, but I would much rather have a program that really used the processor, not just the increased clock speed.

Unfortunately, by the time software tools get debugged and price competitive for a particular generation, that processor is being obsoleted by the marketeers at FastChip Inc. who are telling the developers that it is time to move on. This frenetic pace is exacerbated more when a new processor is just "simply" dropped into an existing architecture (just add a little bit of patching logic). Little thought is given to designing a new system architecture that genuinely capitalizes on the horsepower of the new chip for any particular application.

To offset bandwidth inadequacy of an old bus architecture, designers compensate by unloading low-level processor functions through extensive use of video, sound, math, mass storage, communications, and you-name-it coprocessors. While the idea of distributed processing is a good one, how sensible is it if each of these processors (and the main system processor) is only running at a fraction of its capacity? And in an architecture not akin to even supporting coprocessing to begin with? Now I ask you, does this make sense?

What happens to all of these "obsolete" chips? Look at the number of embeddable '386s and '486s. Unless these things are steering satellites and rockets, it seems a bit of an overkill. It's like using a sledgehammer to drive tacks. The more generations between those "obsolete" processors and the 8- and 16-bit "stone-age" microcontrollers that most of us would actually choose in an embedded control application, the more we feel the need to justify this choice. Do you really need 32-bit CPUs to run a few dozen motors and lights?

I guess my conclusion is that bigger, badder, more potent chips is not a panacea nor does it necessarily improve application software. What we as engineers need to do is slow down long enough to think about better ways to apply what we already have available to us. If that doesn't work, then maybe Tom Cantrell is right. Just give me a processor with about 50 instructions and then you can do whatever you want each generation to make those 50 instructions go faster.

Steve