

CIRCUIT CELLAR

I N K[®]

THE COMPUTER APPLICATIONS JOURNAL

May 1994 — Issue #46

ROBOTICS

Laser Range Finder
System

Issues in Robot Design

Robotic Motor Control

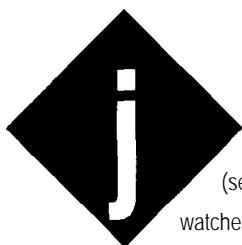
Add Wireless
Remote Control to
Ciarcia's AVmux



\$3.95 U.S.
\$4.95 Canada

EDITOR'S INK

Robocult



Just as Steve has pondered the robotics industry (see "Steve's Own INK" on page 96), so too have I watched the industry and enthusiasts with a certain amount of wonder. It seems that while there is a lot of

interest in the subject by a lot of electronics enthusiasts, the technology just isn't available yet for making the "robotutler" that Joe Public envisions when he thinks of a robot. The lack of a market for anything but mundane industrial robots has kept virtually all companies out of the consumer market (or has killed all those who've tried).

I devoted most of a "ConnectTime" column a few months ago to the subject of robotics and had an outpouring of praise and excitement (and was one reason why we chose to have a "Robotics" theme issue this year). The devotion of most robotics enthusiasts can be almost cult-like in its intensity. No Design Contest winning project has generated more interest than the Laser Range Finder used on a robot designed for the popular Micromouse competition. I'm happy to say that we're finally running an article detailing the design of the Range Finder, and it leads off our feature section this month.

For those not yet indoctrinated into the robotics family, our next article is a short robot design seminar that steps through some of the more basic decisions that must be made when designing a mobile robot. It's not just a matter of mounting a few motors to a square base.

Next, Technical Editor Michael Swartzendruber presents a simple motor control system that could be the basis of a robot's drive system. Low cost and ease of operation were the key parameters here.

Finally, Steve wraps up his audio/video multiplexer with the design of a wireless (RF or IR) remote for changing the AVMux's settings. Some off-the-shelf modules go a long way toward eliminating the need to reinvent the wheel.

In our columns, Ed concludes his LCD panel discussion by covering what is necessary to generate characters on a panel that only does bit-mapped graphics. Jeff describes the software side of Steve's AVMux and provides a look at the "brains" of the operation. Tom breaks out his magnifying glass to get a better look at a complete '386-based PC the size of a credit card. Last, John explores the options available for storing nonvolatile program code on embedded controllers.

CIRCUIT CELLAR **INK**®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

EDITOR-IN-CHIEF
Ken Davidson

TECHNICAL EDITOR
Michael Swartzendruber

ASSOCIATE EDITOR
Rob Rojas

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

WEST COAST EDITOR
Tom Cantrell

CONTRIBUTING EDITORS
John Dybowski & Russ Reiss

NEW PRODUCTS EDITOR
Harv Weiner

ART DIRECTOR
Lisa Ferry

GRAPHIC ARTIST
Joseph Quinlan

CONTRIBUTORS:
Jon Elson
Tim McDonough
Frank Kuechmann
Pellervo Kaskinen

Cover Illustration by Bob Schuchman
PRINTED IN THE UNITED STATES

PUBLISHER
Daniel Rodrigues

PUBLISHER'S ASSISTANT
Sue Hodge

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

HAJAR ASSOCIATES NATIONAL ADVERTISING REPRESENTATIVES

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

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

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

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

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

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

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

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

Entire contents copyright © 1994 by Circuit Cellar Incorporated. All rights reserved. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

- 1 4** Scope Out the Laser Range Finder
by Tom Ward
- 2 2** A Robotics Design Seminar
by Robert Angelo
- 3 0** YAMCI-Yet Another Motor Control Interface
by Michael Swartzendruber
- 3 6** Wireless Remote Control of the AVMux
by Steve Ciarcia
- 4 6** Firmware Furnace
All Text is Graphics: Characters for the '386SX
Project's Graphic LCD Panel
Ed Nisley
- 6 0** From the Bench
Audio/Video Traffic Control
Jeff Bachiochi
- 6 8** Silicon Update
Honey, I Shrunk the PC
Tom Cantrell
- 7 4** Embedded **Techniques**
Embedded Programs
John Dybowski

INSIDE ISSUE 46

- 2** Editor's INK
Ken Davidson
Robocult
- 6** Reader's INK
Letters to the Editor
- 9** New Product News
edited by Harv Weiner

- 84** **ConnectTime**
Excerpts from
the Circuit Cellar BBS
conducted by
Ken Davidson
- 96** **Steve's Own INK**
Steve Ciarcia
The Market That Was
Never Born, Refuses to
Die, and May Yet Live
- 81** **Advertiser's Index**

READER'S INK

Open Your Mouth and Say, "Ahh"

In response to reader Bert Schneider ("Reader's INK," February '94) regarding his request for diagnostic codes and electrical interfaces on Ford computers:

There are two categories of diagnostic codes in use (by *all* manufacturers):

1. Legislated diagnostics (required by law in California). Describes five diagnostic modes.

2. Enhanced diagnostics (not required by law).

Describes a much expanded set of diagnostic modes and reserves space for manufacturer-specific modes. All the enhanced diagnostics modes are optional

The legislated diagnostics are covered by an SAE document (J1979), while the enhanced diagnostics are covered by SAE J2190. (The latest version of SAE 71979 Legislated Diagnostics provides two additional diagnostic modes, for a total of seven. This version has not been published yet, however, so it is not official.)

It is not enough to know the codes-you also have to know how to request the codes and how to massage the data the codes give you. Plus, you have to know how to electrically interface to the vehicle's computer bus. The protocol describing this is covered by other documents-SAE J1850 (Vehicle Communications, Class B), and SAE 72178 (Message Format). There are also SAE documents describing the external scan tools used. The material covered by these documents is very complex and it is difficult to grasp the whole picture.

Additionally, the enhanced diagnostics are, for the most part, manufacturer specific (read "proprietary"), and vehicle manufacturers are understandably reluctant to reveal what they have or have not implemented in their vehicles. (If, as is likely, Bert's '93 Probe has not implemented the J1850 protocol yet, but is still using the older "UART" protocol, then the diagnostic codes used are completely proprietary. Bert will have a difficult time obtaining them.) The thought of a noncertified technician (read "hacker") trying to plug in to their computer bus and attempting to access the computers sends spasms of horror through a manufacturer's body-not to mention what it does to the vehicle's warranty. I would not recommend that anyone try it. As vehicle computerization technology progresses, there will be scan tools available on the market that a noncertified person can use. I would highly recommend waiting.

Jeff Stineburg

Senior Engineering Instructor

Ascent Technologies

Ann Arbor, Mich.

stine@asc-tech.com

Programming Aid with a Twist

When I learned Pascal, I learned it in the context of structured programming. When I learned C a year later, I learned it in the context of structured programming. However, when I learned assembly language for the IBM PC later that same year, my professor made sure I thoroughly understood the fact that assembly language programming was unlike anything I had ever done before. I was practically told to forget everything I knew and get ready to learn it all over again.

I was not about to learn a new programming style and saw no reason to forget the one I already knew, so I adopted a compromise style very similar to what Hank Wallace described in "An Assembly Language Programming Aid" in the February '94 issue. If you combine his suggestions with what I have to add here, you will be writing assembly language faster than ever. But more importantly, five years from now, when you look at your code, you will know what every line was meant to do.

Quick review

The main thrust of Mr. Wallace's article was that if you first express your algorithm in a high-level language, debug it, then hand-compile it by writing it in assembly code right along side the high-level code, you will be able to write more quickly and efficiently, all the while producing code that is portable and fully documented.

How can assembly language be portable? It cannot truly be portable, but if you simply strip off the assembly code, as suggested by Mr. Wallace, you will have a working program in a high-level language that could then be recompiled to the new machine, either with commercial software or by hand.

Two steps further

The style I have been using goes two steps beyond what Mr. Wallace suggested. In fact, he barely mentioned one of my techniques and flat out discouraged the use of my other. I maintain that if you use our techniques combined, your assembly code itself will be self-elucidating as to its meaning and only augmented by the high-level documentation off to the side. After all, the point is to be able to read, write, and understand assembly language quickly and efficiently.

There are two primary differences between my style and the style that Mr. Wallace described. Although these two ideas are fairly simple (and actually go hand-in-hand with what Mr. Wallace was saying), they can make a world of difference in the quality of code that you are able to write. The first suggestion is you should use meaningful labels to simulate high-level concepts when

READER'S INK

writing your code. The second is you should use meaningful indentations to organize your code in the same structural format used in high-level languages.

Meaningful labels

Mr. Wallace realized that statement labels would have to be placed on the lines beside the major branch destinations in the assembly language code. However, what he suggested you do is use the form *Lnnn*, where *nnn* is any three-digit number. In fact, he discouraged the use of any other type of label because he thought that this uniform method would eliminate confusion.

What Mr. Wallace and a lot of programmers tend to forget is that labels do not have to be meaningless numbers. There are certain "cryptic" words that every programmer recognizes. If we use labels whose meanings are recognized by every programmer, the labels themselves will be self-documenting. In fact, we can emulate the functions of every high-level construct by using meaningful labels.

So, what are these magical, cryptic, little words? The answer is simple: if, then, else, endif, while, do, repeat, and so forth. If we are going to have to place labels at every branch destination anyway, why not make them labels worth writing? See Listing 1 for an example of how we can use labels to illuminate the meaning of assembly language code.

Listing 1—Meaningful labels

```
DO_1:   lodsb
IF_1:   cmp al, 'a'
        jb  ENDIF_1
        cmp al, 'z'
        ja  ENDIF_1
THEN_1: sub al, 32
ENDIF_1: stosb
WHILE_1: loop DO_1
```

Meaningful indentations

Mr. Wallace mentioned the fact that indenting blocks of code was a useful way to indicate function. It also helps the reader's eye flow from section to section. Although he mentioned this, he suggested the indentations should be present in the high-level code off to the side. To me, a much better way to code is to indent the assembly language code itself (along with the comments off to the side).

Every programmer knows how to use indentation to logically separate high-level programming constructs. Every programmer also knows how to pick out the

important code blocks as well as the overall flow of a program by simply scanning the code and noticing the ifs, whiles, and dos. The natural thing to do is to implement these same constructs in assembly language using the same programming style. See Listing 2 to see how much easier it is to read.

Listing 2—Meaningful Indentations

```
DO_1:
    lodsb
    IF_1:   cmp al, 'a'
            jb  ENDIF_1
            cmp al, 'z'
            ja  ENDIF_1
    THEN_1: sub al, 32
    ENDF_1: stosb
WHILE_1: loop DO_1
```

Bringing it all together

By using meaningful labels and indentations, high-level programmers have been creating efficient, modifiable code for years. If assembly language programmers apply these same techniques to our code, it will be just as efficient and modifiable as anything written in C.

The only thing these two techniques will not do for us is give us portable code. Since portability is important and nearly all high-level languages are portable, all we need to do is couple our programs into a high-level language. This is easily accomplished by using Mr. Wallace's wonderful idea of writing high-level code alongside assembly code as comments.

The differences that my style had from Mr. Wallace's was the order in which I wrote the code. However, after further examination, I am sure everyone will agree that Mr. Wallace's order of events is much more efficient than mine (before I adopted Mr. Wallace's). Before reading Mr. Wallace's article, what I used to do was 1) express the algorithm in pseudocode, 2) translate the program to assembly, and 3) decompile the code to a high-level language (usually C). Now, I see that it is better to do as Mr. Wallace suggested and 1) express the algorithm in pseudocode, 2) translate the program to a high-level language (still C), and then finally 3) hand compile the code to assembly.

By following the second order of events, we can code our algorithm into a high-level language (and then fully debug it)! It is so much easier to debug an algorithm in a high-level language than one in assembly. In the final

READER'S INK

rogram, we can be sure that all of the bugs arose from our implementation of the high-level constructs and not from faults in the algorithm. See Listing 3 to see how much easier it is to read a program using Mr. Wallace's idea of using high-level implementation as comments.

Jeff Mills
English, IN

Listing 3—Bringing it all together

```
DO_1:                                ; do l
    lodsb                             ; a[i] = getc(buffer);

    IF_1:  cmp al, 'a' ;   if (islower(a[i]))
           jb  ENDIF_1;
           cmp al, 'z';
           ja  ENDIF_1;

    THEN_1: sub al, 32 ;   toupper(a[i]);

    ENDIF_1: stosb

    WHILE_1: loop DO_1                ;} while (a[i] != EOF);
```

Contacting Circuit Cellar

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

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

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

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

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

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

A STROKE OF GENIUS FROM MOTOROLA: THE 68306

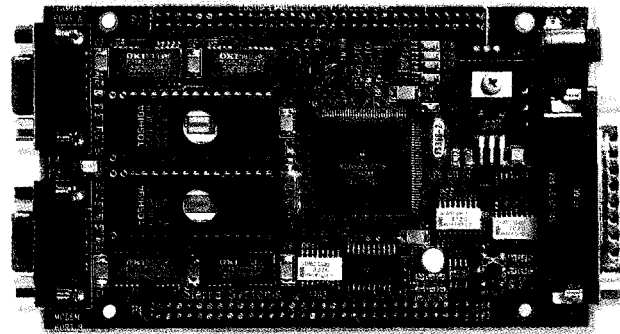
- 16 MHz 68EC000 CORE • DRAM CONTROLLER
- SOFTWARE PROGRAMMABLE CHIP SELECTS AND INTERRUPT LINES • COUNTER/TIMER
- TWO 8-BIT BIDIRECTIONAL PARALLEL PORTS
- WATCHDOG TIMER • TWO SERIAL CHANNELS

ALL ON ONE LOW-COST CHIP

And from Sierra Systems: The Complete Hardware/Software Solution for only \$450

Sierra Systems is offering a complete PC based development system that includes a 68306 CPU card, power supply, 68306 Configuration Utility, and a restricted use license to the Sierra C™ Compiler and QuickFix™ source level debugger.

Whether you are developing software, evaluating the 68000 family, looking for an engine to drive prototype hardware, or need a processor card in an OEM system, this kit is the ideal solution.



6728 Evergreen Ave • Oakland, California 94611 • Tel (510) 339-8200 • Fax (510) 339-3844

8 0 0 - 7 7 6 - 4 8 8 8

NEW PRODUCT NEWS

Edited by Harv Weiner

CODEC DEVELOPMENT BOARD

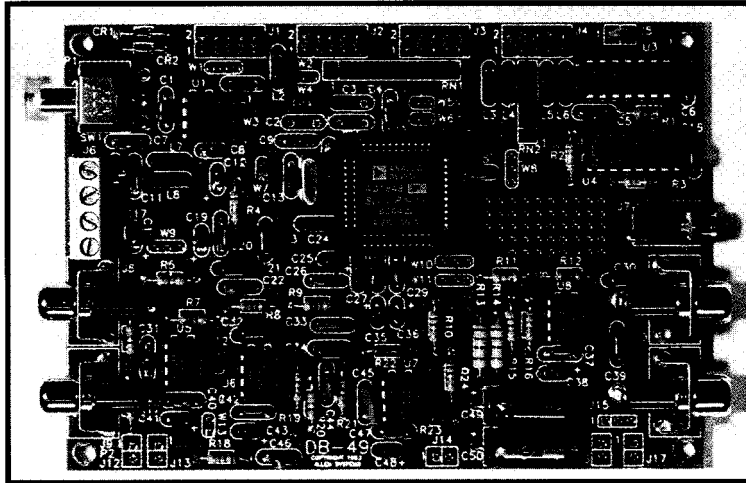
Allen Systems has introduced a development board based on the AD 1849 16-bit stereo codec. Called the **DB-49**, it facilitates development of applications based on the AD1849. A main feature of the codec is its ability to handle high-quality audio in multimedia and other applications.

The AD1849 offers numerous features including sigma-delta A/D and D/A converters, multiple channels of stereo input and output, programmable gain and attenuation, and sample rates of up to 48 kHz. A primary feature of the AD1849 is its serial interface for communicating with the host DSP. This serial approach simplifies the connection to the DSP. In addition, it allows the AD1849 to be used with a variety of DSP devices.

The DB-49 board includes an assortment of digital and analog circuits to facilitate design of AD1849-based products. On the digital side, the board provides reset-generation logic. Circuitry to drive the D/C (data/control select) input is also available on the DB-49. This simplifies the codec/DSP interface even further since the DSP is not required to generate the D/C control line.

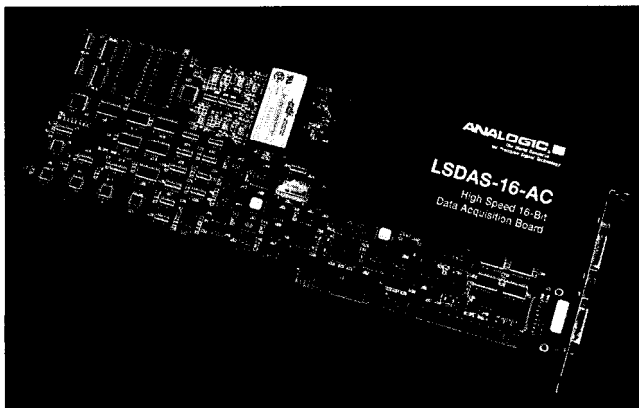
A number of analog circuits are also provided on the DB-49. The board can generate an analog +5-volt supply source from its digital +5-volt supply, if desired. Alternatively, a separate +5-volt analog supply may be tied to the board. Also provided in the DB-49 are a line-level input circuit, a "phantom-powered" microphone input circuit, a line output circuit, and a headphone output drive circuit. Standard analog connectors facilitate board installation.

The DB-49 is designed to support daisy chaining of two or more boards. Header connectors are available to simplify this type of system configuration. The DB-49 sells for \$200.



Allen Systems • 2346 Brandon Rd. • Columbus, OH 43221 • (614) 488-7122

#500



HIGH-PRECISION 16-BIT DAS BOARD

Analogic Corporation has announced the **LSDAS-16-AC**, a low-cost, 16-bit, AT-compatible data acquisition board. The board features 16 programmable analog inputs capable of acquiring up to 50,000 samples per second. This board can be used for high-performance PC-based instrumentation applications such as spectroscopy,

copy, chromatography, audio, benchtop testing, and multichannel data acquisition.

The LSDAS-16-AC offers a selection of analog input modes. The user can program the board to accept either 16 single-ended or 8 differential inputs, and can select one of six unipolar or bipolar full-scale input ranges. The board maintains low noise specifications by combining expert circuit design and a multilayer circuit board. In addition to 16 digital I/O lines, one 16-bit counter/timer input is available for event counting or for dividing an external signal. An expansion multiplexer will support up to 256 analog inputs and a variety of signal conditioning options.

The LSDAS-16-AC is priced at \$995. Complete documentation is provided with the board and includes setup routines and data acquisition utilities. C language libraries are available at an additional cost of \$295.

Analogic Corp. • 360 Audubon Rd.
Wakefield, MA 01880
(508) 977-3000 ext. 2388 • Fax: (617) 245-1274

#501

NEW PRODUCT NEWS

PCMCIA CARD ADAPTER

Curtis Inc. has introduced a new PCMCIA computer accessory product for use with IBM-compatible ISA bus computers. The ROMDISK PC **CardSharkII**

CardSharkII is 3.5-inch

CardSharkII

CardSharkII.

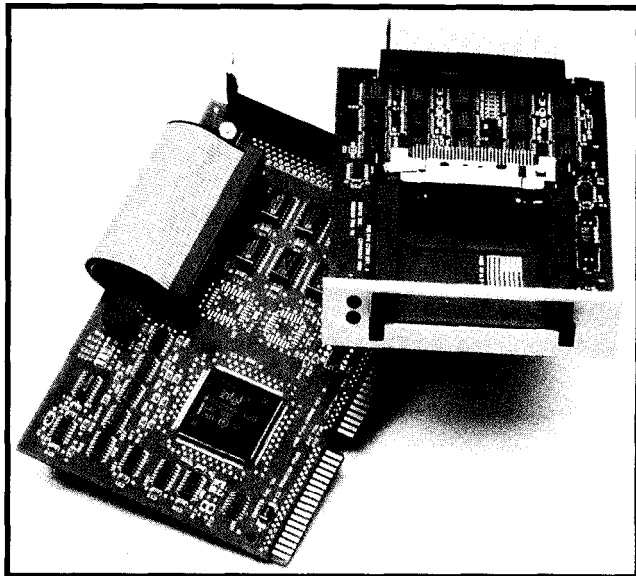
CardSharkII

and II Flash cards, Intel, Sundisk, and rotating media PCMCIA IDE/ATA drives. The PCMCIA sockets are fully isolated and buffered so cards may be "hot swapped" while another card is active in another socket. Both FAT and the Flash file systems are supported for compatibility with any notebook or palmtop PC.

The CardSharkII fits into a standard 3.5-inch drive

connections are required. Extra heavy duty power

control circuitry is provided for #502 high-current PC cards



autoboot CIA hard drives. An optional capability for SRAM and Flash PC cards is available. Extended ROM BIOS support is provided on-board. The CardSharkII internal version sells for \$299; the external version sells for \$349.

Curtis, Inc.

418 West County Rd. D • St. Paul, MN 55112

631-9512 •

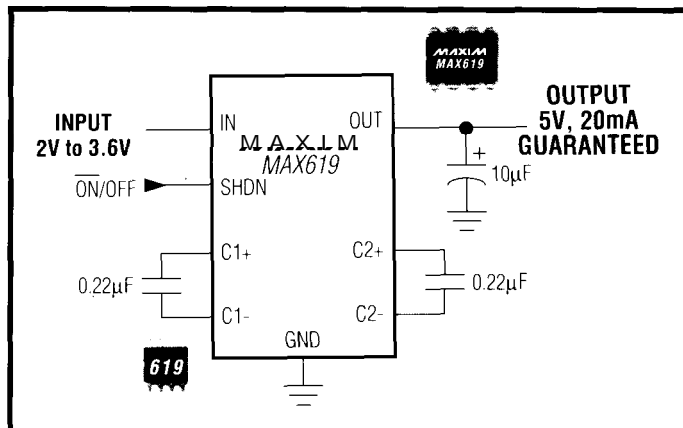
631-9508

mA charge pump converter that provides a regulated 5-V output at up to 20

3-

V-only systems.

8-pin (DIP or SO) package uses internal charge pumps to generate the $\pm 4\%$ by a pulse-skipping controller. The complete M in²



μA 0
kHz)

capacitors. Its low shutdown supply current (10 μA max) completely disconnects the load from the input, preventing battery drain. The MAX619 sells for \$2.10 in OEM quantities.

Maxim Integrated Products

120 San Gabriel Dr.

Sunnyvale, CA 94086

(408) 737-7600

#503

NEW PRODUCT NEWS

LOW-COST DATA ACQUISITION SOFTWARE

The Notebook/LE software from Omega is a low-cost version of the popular Labtech Notebook software. It is ideally suited for straightforward data acquisition and control applications with relatively low channel requirements.

With Notebook/LE, users can collect and display data to the screen in a variety of graphic formats, including trend charts, bar graphs, and meters. Calculation functions allow both linear and nonlinear

scaling of the data, as well as a host of other capabilities. Data may be stored to disk in a condensed binary format or in ASCII format for compatibility with most spreadsheet programs.

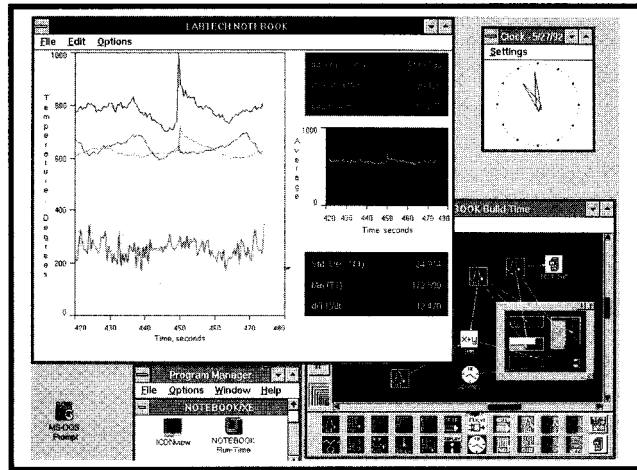
Notebook/LE supports on/off and PID control. Setup is simple and intuitive. By interconnecting icons or graphic symbols, users can quickly implement the data acquisition task. For example, connecting an analog input icon to a log icon creates a data-logging system. Applications can be created or modified very rapidly.

Notebook/LE supports a variety of plug-in boards from Omega. The Note-

book/LE (SWD-LTNLE) sells for \$495 and includes both DOS and Windows versions.

Omega Engineering, Inc.
One Omega Dr.
Stamford, CT 06907-4047
(203) 359-1660
Fax: (203) 359-7700

#504



ATTEND THE SECOND ANNUAL
ECC
EMBEDDED COMPUTER CONFERENCE
& EXPOSITION
JUNE 8-10, 1994
SANTA CLARA
CONVENTION CENTER
SANTA CLARA,
CALIFORNIA

Hardware and Software Solutions for Embedded Computer Applications

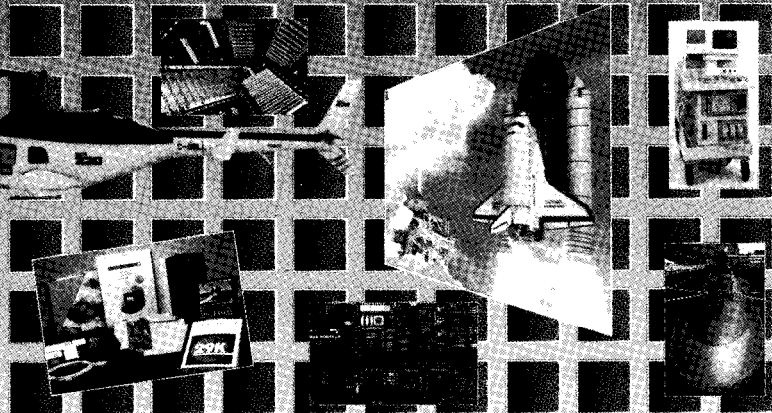
Who Should Attend Hardware and software engineers who put computing power to work in embedded applications.

Technical Program "32 bits or better" The demand for higher performance, smaller size, lower cost and faster time to market, is forcing the move to more powerful and advanced solutions such as 32- or 64-bit RISC or DSP for processing; ASICs and multichip modules; high-performance backplane buses and more...

You'll find two and a half days packed with a combination of theoretical, practical, and application orientated sessions on incorporating high-performance embedded computers and developing embedded realtime software for complete systems and subsystems.

Exhibit Hall A comprehensive mix of leading vendors will talk with you about solutions to your specific embedded and dedicated applications.

See The Latest In: microprocessors single-board computers peripheral boards operating systems realtime kernels and compilers development systems CASE and debugging tools embedded PCs and more...



To receive detailed information on seminars and attending ECC, please complete and return the coupon below. FAX TO: (203) 838-1447, OR CALL (203) 831-9444



MAIL COUPON TO: ECC / BAV Expositions, Inc.,
85 Washington Street, Norwalk, CT 06854

Send Me Information On Attending Send Me Information On Exhibiting

Name _____

Title _____

Company _____ # of Employees _____

Address _____

City _____ State _____ Zip _____

Phone _____ Fax _____

Sponsoring publications: Computer Design, EE Times, Military & Aerospace Electronics, VMEbus Systems Associations: MMG, PC/104 Consortium, SIG32, STDMG

CPJ

Photographs courtesy of Siemens Medical Systems, NASA, Electric Boat, BCC/Gen Electronics, Inc., Evace Computers, Inc., Advanced Micro Devices, Inc., Apple Computer, Inc., Bell Helicopter

NEW PRODUCT NEWS

LOW-COST DEVELOPMENT SYSTEM

Highlands Electronics has announced a new type of PC card that eliminates the high cost and complexity of standard computer emulators. The **eBoard** uses the actual CPU during development. Because the cost of the eBoard is a fraction of an traditional in-circuit emulator, the unit can be left in place to run the application.

The eBoard is supported by a development system that consists of a cross-

assembler, an integrated developer's environment, and board-resident Forth language and kernel. Development is very fast because the cross-assembler and debugger are PC resident and do not burden the target CPU's memory. Breakpoints can be examined on the fly for real-time debugging, and the eBoard's memory and devices can be modified while the application is running. The development system also comes in library form to allow development of the user interface as part of the debugger. Switching between the debugger and application is eliminated.

The eBoard has a central input and output kernel supporting an object-oriented method of development. A user can work as if a device, such as a keyboard or port, were actually attached to the target computer. These devices would be real with one library and virtual with another. A board-resident Forth language provides the user with a quick and simple means for adapting to changes in the application.

The eBoard features a 6502 microprocessor and is externally powered for continuous data acquisition and control. It comes with a

breadboard, cable, experimenting supplies, and instructions for a series of experiments. The board also includes a 6522 Versatile Interface Adapter with its I/O lines and selected CPU lines brought out to a DB-37 connector. The eBoard sells for \$249.95 including postage and handling.

Highlands Electronics
13720 Lake Shore Dr.
P.O. Box 927
Clear Lake, CA 95422-0927
(707) 994-1024
Fax: (707) 994-5823

#505

MULTIPLE VIDEO WINDOWS FOR WORKSTATIONS

RGB Spectrum has announced **SuperView**, a multiple video windowing system that displays up to four real-time video windows on a single high-resolution monitor. Each video window can be positioned, scaled to full screen, overlaid with computer graphics, or overlapped with other video windows. SuperView can be used for video conferencing, command-and-control, surveillance, simulation, and robotics applications.

SuperView is a third-generation system based on a proprietary design that guarantees real-time video performance under all conditions without burdening the host CPU or graphics controller. It was developed for applications that require the simultaneous display of computer graphics

with multiple video sources. It accepts NTSC (or PAL) composite video and Y/C (S-Video) signals from up to four cameras, tape recorders, video disc, and teleconferencing systems simultaneously. In addition, it will accept various high-line-rate video signals from FLIR and medical imagers.

The system supports software control to manipulate the video windows, adjust video parameters, and control graphics overlays.

Optional X.TV software provides full integration under X Windows.

RGB Spectrum
950 Marina Village Pkwy.
Alameda, CA 94501
(510) 814-7000
Fax: (510) 814-7026

#506



NEW PRODUCT NEWS

COMPACT TIME-DELAY RELAY

A compact, block-style, time-delay relay featuring a "delay-on" timing function has been announced by ISSC/Kanson. The solid-state 2110 series incorporates two voltage options for operation from 24 to

240 VAC/DC and is capable of switching up to a one-ampere load.

Three time-range options are available to permit time settings from 0.1 seconds to 10,230 seconds by means of DIP switch programming. The 2110 timers offer repeat accuracy of $\leq 0.5\%$ with a timing tolerance of $\pm 5\%$ and

a reset time of ≤ 50 milliseconds.

The encapsulated unit measures 2" x 2" x 0.75" and offers a simple two-wire installation. The unit features normally open contacts, and the voltage drop at one ampere is typically 2.5 volts. The load may be connected to either side of the line.

ISSC/

Kanson Electronics, inc.
4700 Raycom Rd.
P.O. Box 170
Dover, PA 17315-0170
(717) 292-5631
Fax: (717) 292-1696

#507

POWER PROTECTION CATALOG

A free catalog from Best Power Inc. on the topic of power protection helps users of sensitive electronics define and solve power problems. It shows how to save money by protecting equipment from power problems such as surges, sags, spikes, noise, brownouts, black outs, and lightning.

The catalog is a power protection reference manual. Along with brief descriptions and specifications of products, the catalog offers the following sections:

Power Problem Analysis and Solutions: This section gives a brief review of many common power problems, symptoms that make their identification simple, and a brief discussion of which type of power protection technology will best solve the problem and why.

How to Buy a UPS/SPS: For the uninitiated, power protection choices can be overwhelming. The catalog makes selecting the right power protection as easy as asking a few very basic questions about the system that needs protection. The reader can then select the appropriate level of protection based on their answers.

Latest Advances in UPS Communication Software: The catalog explains how, with communication software, UPS users need never worry that their system will crash because of unattended operation during long outages.

The **Best Power Protection Catalog** can be ordered by calling (800) 356-5794.

Best Power Technology, Inc.
P.O. Box 280
Necedah, WI 54646
(608) 565-7200
Fax: (608) 565-2221

#508

TRANSIMPEDANCE AMPLIFIER

The **PTA-100** from Portable Technologies is a compact, battery-powered, precision transimpedance amplifier that converts low-level DC and AC currents into high-level voltages. It is designed to be used in conjunction with a variety of voltage-sensitive instruments such as oscilloscopes, lock-in amplifiers, multimeters, chart recorders, and data loggers, permitting them to measure weak current levels ranging from microamps down to femtoamps.

The PTA-100 has five sensitivity ranges covering 10^{-5} A/V to 10^{-9} A/V with 1% accuracy. It features a very low current noise of 4 fA/Hz^{1/2} at 10^{-9} A/V, a 20-kHz bandwidth at 10^{-5} A/V, a virtual ground input, and a full range output of ± 3 V peak to peak. Input and output offset voltages are less than 0.1 mV and input leakage current is less than 0.1 pA.

Because of its small size and its ability to drive long cables, it can be conve-

niently located close to the signal source to minimize electrical noise and interference. Low power consumption allows the PTA-100 to operate up to 500 hours on a Y-volt alkaline battery. Applications include the measurements of signal currents from sensors (e.g., photodiodes, photomultipliers, and ion gauges), leakage currents, tunneling currents, and high-value resistance.

An optional BNC-mounted silicon photodiode, **PD-1**, is also available. It connects directly to the PTA-100, transforming it into a wide-range photodetector with noise-equivalent-power as low as 10 fW/Hz^{1/2}. The PTA-100 sells for \$179 and the PD-1 for \$39.

Portable Technologies
P.O. Box 20763
Castro Valley, CA 94546
(510) 537-4954

#509

FEATURES

14

Scope Out the
Laser Range Finder

22

A Robotics
Design Seminar

30

YAMCI-Yet Another
Motor Control Interface

36

Wireless Remote Control
of the AVMux

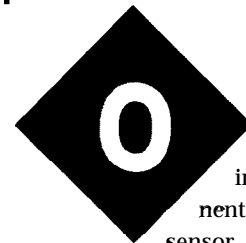


FEATURE ARTICLE

Tom Ward

Scope Out the Laser Range Finder

We've seen many kinds of sensor systems for robots, including ultrasonics, IR LEDs, video...the list goes on. This Design Contest Winner that uses a laser instantly caught the judges' eyes, as it likely will yours.



One of the most important components of a robot is the sensor system. Whether it is an unmanned lunar rover or an industrial cleaning robot, accurate and reliable measurements of distance are absolutely essential.

The requirements of a sensor system are essentially the same for all robots. The distance between the robot and objects of interest must be measured quickly and accurately, even when confronted with an unexpected environment. This means the device must not be affected by ambient light and sound conditions, it must be able to detect objects of varying composition, and differentiate between objects that are closely spaced. Power consumption and size are always at a premium with robots, so the range finder should be compact and low powered.

BACKGROUND

The sensor system described in this article was initially intended for micromice: experimental autonomous robots designed purely for the task of navigating and solving mazes. Micromice must negotiate their way through a 2.88-meter square maze composed of 16x16 cells. The unit walls are 5 cm

high, 18 cm long, and painted white on the sides and red on top. The function of a micromouse sensor system is, in its crudest form, to detect walls for the purpose of maze mapping and avoiding collisions. Although this may sound like a simple task, good performance necessitates the consideration of many different criteria. The micromouse maze also serves well as a general model of a robotic environment, albeit greatly simplified. Figure 1 illustrates the most common micromouse sensor system: the retroreflective infrared variety.

In this system, pairs of transmitters and receivers detect the presence of a wall underneath by the reflectance of a portion of the transmitted infrared waveform. This configuration is simple to operate, but suffers from several major drawbacks. First, the resolution is limited to the minimum separation distance achievable between the sensor pairs, and the complexity of circuitry required for reception (and consequently the number of input bits to the controller) is in direct proportion to the number of sensors.

Second, the sensing range extends only as far as the outstretched "arm" is extended, limiting the maximum distance to a few inches.

Third, the complete sensor assembly is mounted high off the ground, raising the center of gravity and making fast turns harder to perform. Additionally, and perhaps most importantly, this sensor system can only be used

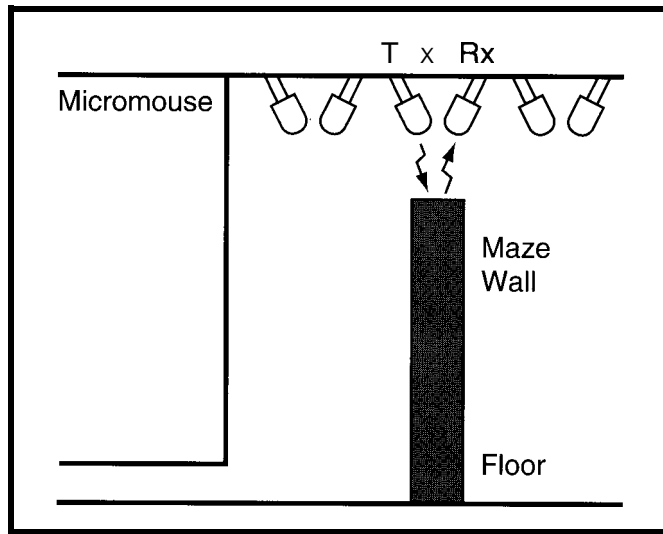


Figure 1—One common micromouse sensor system uses IR LEDs to sense the tops of the maze walls.

in a known environment such as a micromouse maze.

Another popular ranging technique for general robotics is sonar. The time of propagation for an ultrasonic wave to reflect from a distant object is measured, and from this, a range can be calculated. Sonar has the advantage of allowing longer-range measurements, so the presence of an obstacle in the distance can be detected early. In the case of micromice, long-range measurements mean we can plan the path of long runs before they are started, accelerating to high velocity, and then decelerating again before impact! Unfortunately, one of the main problems with ultrasonics is the cancellation and reinforcement of

waveforms caused by unwanted reflections from adjacent walls and objects. Related to this problem is the difficulty of producing a narrow "field of view" sonar system that is able to differentiate between multiple closely spaced objects.

So exactly what are we looking for in a micromouse sensor system? We need a long-range, high-resolution, rugged range finder which is compact enough to be mounted "below the wall" and has a narrow field of view. Ideally it should allow

measurements at both close range (for wall tracking) and long range (for path planning). It is no coincidence that a sensor system with these characteristics is equally in demand for other robotics applications as well.

OPERATION

Although a laser range finder is not a new idea, most existing devices use either mechanical scanning arrangements, interference measurements, or time-of-propagation techniques, and are optimized for long-distance (like surveying) applications. This results in bulky, expensive units with many moving parts.

I decided to investigate the possibility of designing an experimental laser range finder that was small enough to be hand held and allowed distances between a few centimeters and a few meters to be measured. The design described here fulfills these requirements and delivers the data in a standard form (in this case, 9600 bps RS-232) so performance can be verified easily and the module can be

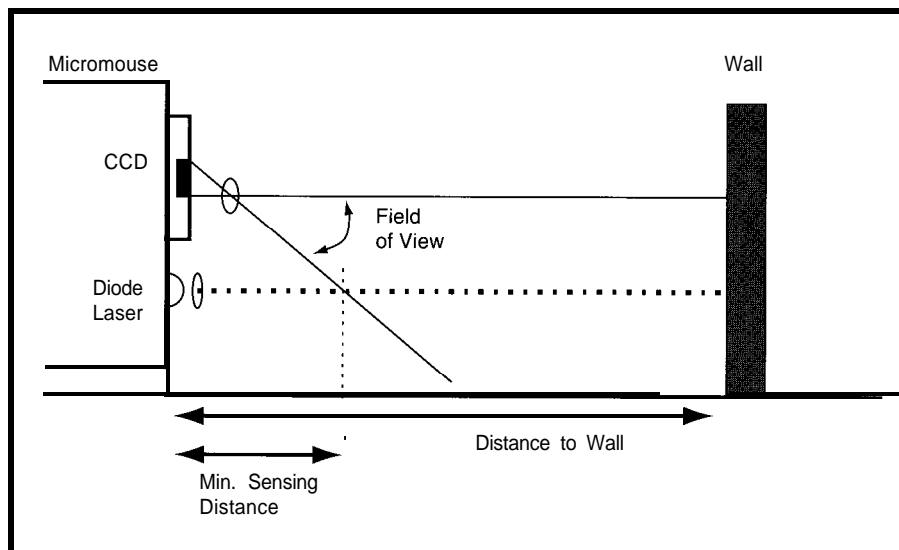
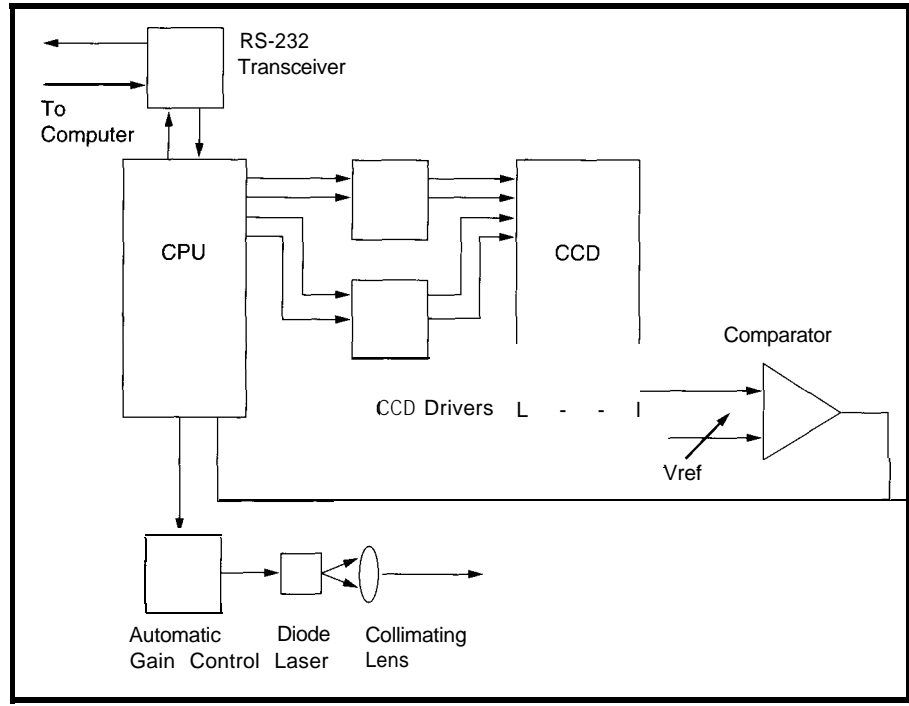


Figure 2—The laser range finder works by producing a spot on the distant wall, then sensing the position of its reflection on a CCD array.

Figure 3—The laser range finder uses a PIC processor to drive the CCD array, receive feedback from the CCD, drive the laser diode, and communicate with the outside world through a bit-banged serial port.

directly connected to existing robotic vehicles. The sensor unit can be made for under \$150 and is not just restricted to robotic applications but anywhere where short distances need to be measured accurately.

Figures 2 and 3 show the method of operation of the laser range finder. A collimated laser diode produces a spot of light on the object to be measured (in this case a maze wall). A focusing lens produces an image of this spot on the surface of a CCD (charge-coupled device), the contents of which are then examined to determine the position of the spot. (A CCD acts essentially as a shift register for analog signals, with the light intensity falling on each pixel producing the analog signal which is shifted along, and finally out the other end of the device.) As the object is moved toward the range finder, the spot stays in the same position on the wall, but the position with respect to the field of view of the CCD changes, giving rise to the method of distance measurement. This



setup can be thought of as a one-dimensional camera looking at an object that is slightly off-center in the field of view.

Note that this method requires only the comparison of each CCD pixel with a threshold value, so a single comparator can be used instead of a high-speed analog-to-digital converter.

IMPLEMENTATION

Several functions must be performed by the circuitry controlling the range finder. First, the CCD must be provided with a set of rather complex clock waveforms to allow the pixels to be shifted out and the video output compared with a threshold value. The resulting measurements must then be converted into a suitable output

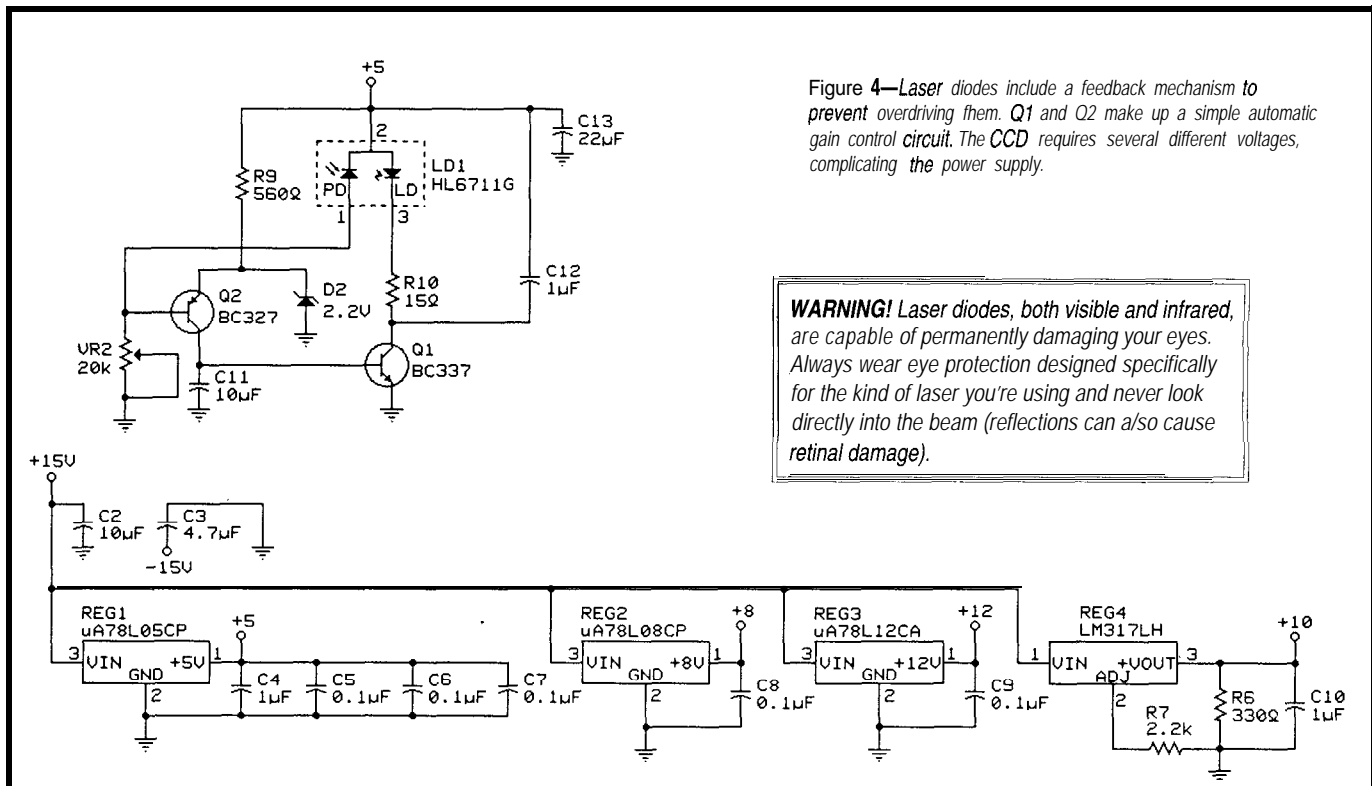


Figure 4—Laser diodes include a feedback mechanism to prevent overdriving them. Q1 and Q2 make up a simple automatic gain control circuit. The CCD requires several different voltages, complicating the power supply.

WARNING! Laser diodes, both visible and infrared, are capable of permanently damaging your eyes. Always wear eye protection designed specifically for the kind of laser you're using and never look directly into the beam (reflections can also cause retinal damage).

format (in this case, RS-232) for delivery to the host computer. Additionally, I decided the host computer should be able to specify the sampling rate remotely, so the controller must allow bidirectional serial communication.

Given that a variety of functions need to be performed, I decided to use a dedicated microcontroller, and in this case the trusty PIC series proved to be the best choice. These devices produce fast enough execution to allow 1000 distance samples per second at a 16-MHz clock rate, and are inexpensive, low power, and available in a small footprint device (18-pin DIP for the 16C54). Although on-chip communication functions are not provided, it is a simple task to code serial transmit and receive routines in software.

The CCD I chose for the job is the CCD111A from Loral Fairchild. This device is a 256-element linear CCD array in an 18-pin DIP and provides the imaging function of the system. A CCD typically requires several clock waveforms (in this case, four signals in the range of 0-8 volts), and these are provided by the ICL7667 drivers (low-power pin equivalents of the DS0026).

The laser diode is an Hitachi HL6711G 5-mW, 670-nm (visible red) unit; the type commonly used in laser pointers. The two transistors (Q1 and Q2 in Figure 4) provide a simple automatic gain control function for controlling the optical power output of the laser. In this case, I preadjusted the power to 3.5 mW by way of VR1. It is worth noting that diode lasers are very sensitive to both static electricity and excessive operating current, so care must be taken when handling them and when adjusting the power.

I initially thought that an infrared diode laser might be preferable for this system, but decided against it for several reasons. When setting up and calibrating a sensor system like this, the visual feedback from the red laser beam is invaluable. There is also a great advantage in knowing exactly where the measurement is being taken. It is also worth noting that for safety reasons that infrared lasers are

particularly dangerous. The near infrared can penetrate the human eye, but is not actually registered on the retina, so damage can unknowingly be caused by direct viewing of the radiation. Although safety is still a concern with visible lasers, damage is rarely caused by a low-power visible device due to the in-built blinking action of the human eye when strong light is present. All lasers must be handled with care, however, and the system should display a warning sticker for a class IIIB Laser product, informing the user not to look directly into the beam.

The mechanical aspect of this design is just as important as the electronic. A stable mechanism is required for the attachment of the imaging lens to the CCD and the collimating lens to the diode laser. In this case, I constructed two holding arrangements from brass stock available from a hobby supplier. The lenses I used have a 4.25-mm focal length, are 6 mm in diameter, and were extracted from an expired CD player. Alternatively, they can be bought directly from scientific suppliers.

The cost of the complete sensor is dominated by the CCD, currently around \$45, the laser diode (about \$40), and the optical filter discussed later (\$38), bringing the total to just under \$150.

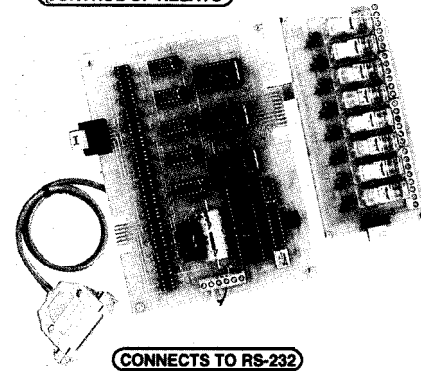
RESULTS

Figure 6 shows the test results of the laser range finder. It can be shown that the image formed on the CCD occurs at a distance $d = (f \times h)/x$, where f is the focal length, h is the distance between the CCD and the laser, and x is the distance to the object. The CCD in this case has an active area of 3.328 mm, so graphing the above function from 0 to 3.328 mm in 256 steps will produce the predicted curve of output versus distance as shown in Figure 6.

The differences between the predicted and experimental curves are almost entirely due to the inaccuracies in alignment of the imaging lens along the axis of the CCD elements. This causes a small, constant error that can easily be subtracted from the readings,

RELAY INTERFACE

PROVIDES SOFTWARE CONTROL OF RELAYS

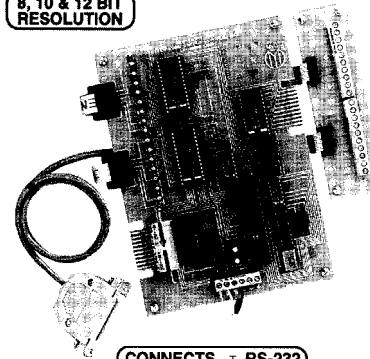


CONNECTS TO RS-232

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

ANALOG TO DIGITAL

8, 10 & 12 BIT RESOLUTION



CONNECTS TO RS-232

ADC-16 CONVERTER* (16 channel/8 bit).....\$ 99.95
ADC-8G A /CONVERTER* (8 channel/10 bit).....\$124.90
Input varieties and a wide RS-422/RS-485 A/D for info on other

ADC-8Ech).....\$ 139.95
Includes 16 m. block channel).....\$ 99.95 on/off

sensor detectors, and other devices.
STA-8D \$ 134.90

Allows callers to select control functions from any phone, PS-4 PORT SELECTOR (4 channels RS-422).....\$ 79.95 port into 4 selectable RS-422 ports.
CO-485 (RS-232 to RS-422/RS-485 converter).....\$ 44.95

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

SUPPORT...provided & disk including test software

RELIABILITY...engineered applications

RS-485...use TO RS-232, RS-422 or IBM and compatibles, Mac

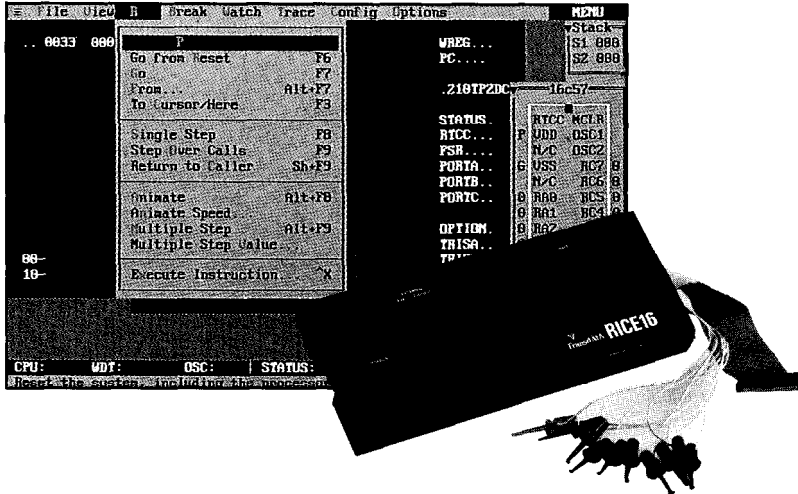
Use our 800 number PACKET. Technical Information (614)

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

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

ELECTRONIC ENERGY CONTROL, INC.
380 South Fifth Street, Suite 604
Columbus, Ohio 43215.5438

RICE16-5X

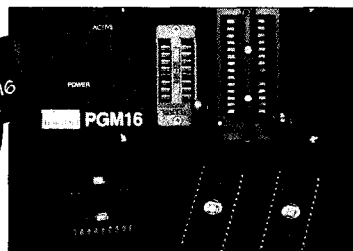


Real-Time Emulation to 16MHz Complete system for 16C5x only \$895.00'

Introducing RICE16-5x, real-time, non-intrusive in-circuit emulator for the PIC16C5x family microcontrollers: a feature-filled development system at an affordable price. **Suggested Retail for US only*

- Real-time Emulation to 16MHz
- Non-intrusive Operation
- PC-Hosted via Parallel Port
- 8K Program Memory
- 8K deep by 24 bits wide Trace Memory
- Source Level Debugging
- Unlimited Breakpoints
- External Trigger Break with either "AND/OR" with Breakpoints
- *Trigger Outputs on any Address Range
- 12 External Logic Probes
- User-Selectable Internal Clock from 40 frequencies or External Clock
- Easy-to-use windowed software
- Single Step, Multiple Step, To Cursor, Step over Call, Return to Caller, etc.
- Search Capability in Source Code, Program Memory and Trace Buffer
- On-line Assembler for Patching Instruction
- Support; 16C71 and 16C84 with Optional Interchangeable Probe Cards
- Comes Complete with TASM16 Macro Assembler, Windowed Software, Power Adapter, Parallel Adapter Cable and User's Guide
- 30-day Money Back Guarantee
- Made in the U.S.A.

Call our BBS at (214)980-0067 to download DEMO: RICE16.ZIP



* for a limited time only



Advanced **Transdata Corporation** Tel (214)9802960
14330 Midway Road, Suite 104, Dallas, Texas 75244 Fax (214) 980-2937

or alternatively steps could be taken to align the system more accurately. The graph is not linear due to the $\frac{1}{x}$ term in the above equation, but this is not a problem in most applications. A lookup table could be provided on the PIC, or the distance could be calculated directly from this equation. The output will always be of higher resolution for close measurements than for longer ranges, but this graph can be made more linear (and longer range) by increasing the separation between the CCD and the laser, at the expense of the minimum sensing distance.

Many robotic applications can actively exploit this nonlinear characteristic, as it is often important to be able to measure short distances with high accuracy, while requiring only approximations for objects that are a long way off (as in the case of the micromouse).

Changing the lens also allows the ranging distance to be correspondingly altered. A longer focal length lens has a narrower field of view and allows operation over a greater distance. However, a narrower field of view also increases the minimum sensing distance. In this case, $f = 4.25$ mm and $h = 4$ cm, allowing a minimum sensing distance of 5.25 cm. This can be calculated by evaluating the distance x in the equation above for d equal to 3.328 mm (the end of the CCD).

It is worth mentioning that this ranging system was designed primarily to fit into a small space, but the availability of high-resolution CCDs (up to several thousand pixels) allows the potential of higher-resolution, longer-distance sensors if desired. With a 256-element CCD, short-distance measurements are accurate to within fractions of a millimeter and the useful sensing distance extends to several meters with a resolution of about 15 cm at 2 m.

It is interesting to note that the color of the object has little effect on the output versus distance. I performed tests on the ranging system with both white (90% reflectivity) and mid-gray (18% reflectivity) objects, and the output was consistent within 1 LSB. It

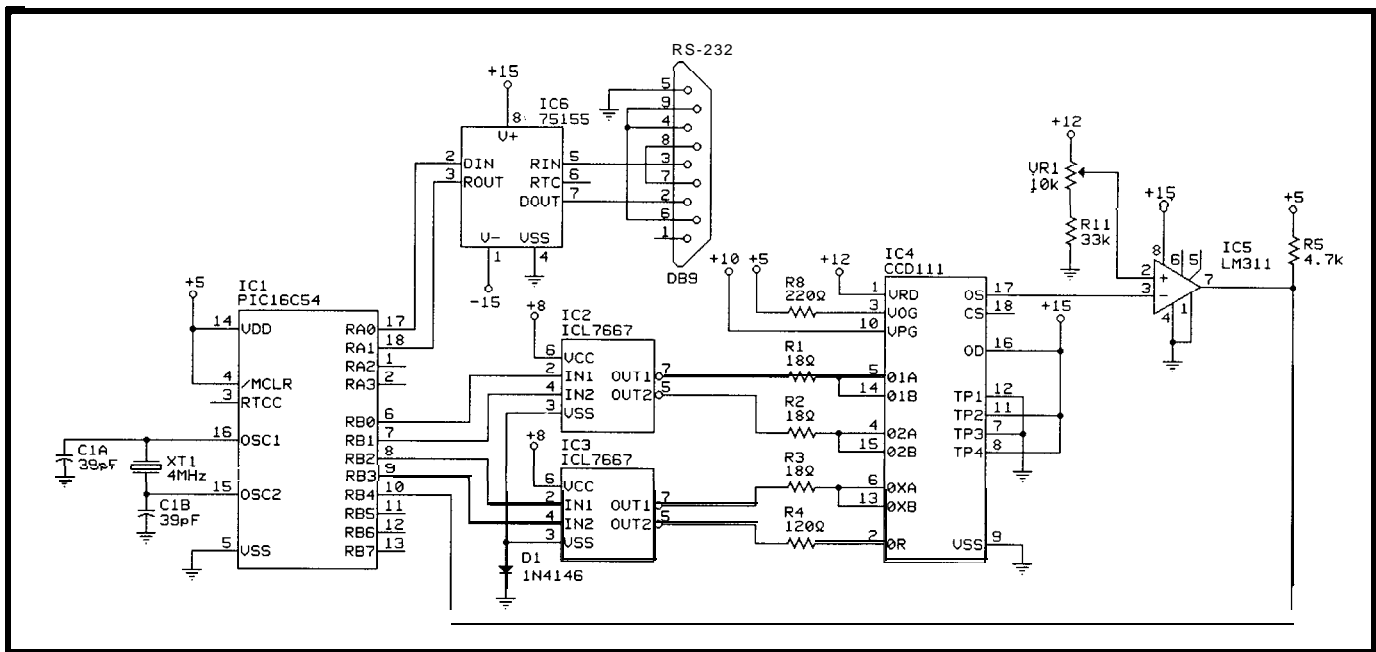


Figure 5—The PIC processor spends most of its time generating waveforms for the CCD array. A simple LM311 comparator helps detect the presence of the laser spot's reflection. A standard RS-232 interface talks to a host computer.

is surprising how consistent the results are, even with a matte black object, although the maximum sensing distance is reduced if the intensity of reflected laser light is substantially decreased.

Extremes of lighting conditions can affect the output, but this can be reduced to negligible amounts by covering the CCD lens with a suitable filter. I used a 671-nm (± 10 -nm band-

pass) optical filter from Edmund Scientific here, and with it in place, the output did not change when I directed a 150-W spotlight onto the object at 1 meter!

The angle of the unit to the surface of an object is also not particularly important. Tests showed that the object can typically be adjusted to $\pm 75^\circ$ off-center while maintaining a correct reading.

Another great advantage of this system is the speed of distance measurements. With a 4-MHz crystal, 250 samples/second can be taken, but this limit is primarily the software overhead. Replacing this with a 16-MHz crystal should allow 1000 samples per second to be taken, allowing excellent motion control loops to be maintained (e.g., when aligning the robot with a wall when running down a corridor). Note that this sampling speed is equal to the inverse of the integration time of the CCD (the time that we are collecting light), so consideration must be given to the amount of laser light reflected back for very fast sampling. I obtained the experimental graph above at 200 samples per second.

Power consumption is just over 100 mA continuous, dominated by the laser diode's requirement of 85 mA. If power consumption must be reduced further, the laser diode could be turned off between measurements or pulse-width modulated by the PIC, and the voltage regulators could be replaced with lower quiescent-current devices.

Total power consumption could be reduced to around 20 mA or less if measurements are required only around 20 or 30 times per second rather than several hundred.

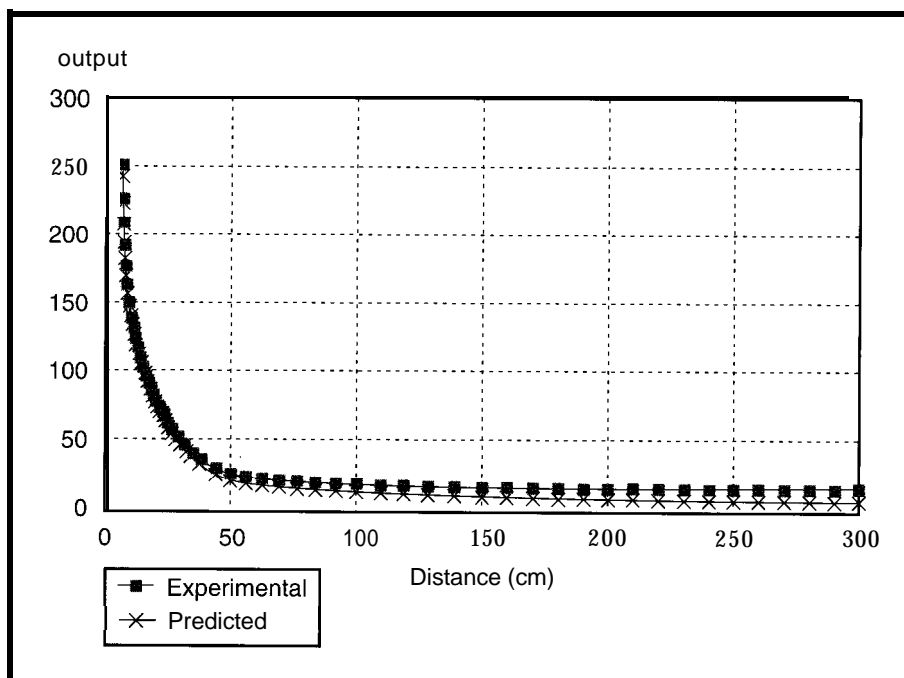


Figure 6-A comparison of the predicted and experimental results shows virtually the same graph. Any inaccuracies are due to alignment of the imaging lens along the axis of the CCD elements.

A final note about the lens arrangement. Normally a lens is placed directly over the center of an imaging element (in this case the CCD), but for our application the dot from the laser diode is always at one side of the field of view, so this configuration would result in only half of the CCD being used. A simple adjustment made by moving the lens over the end of the CCD closest to the diode laser (look back at Figure 2) results in the full 256 pixels being used for the measurement. We could have angled the CCD or laser diode, but the shifting of the lens is much simpler to do mechanically.

SOFTWARE

The software is in two parts: the embedded PIC code (available on the BBS) and the IBM PC example program (Listing 1). The PC is really only acting as a serial terminal, displaying the distance measurements (directly in terms of which one of the 256 pixels is activated first). A facility is provided by the software to change the number of samples per second acquired. The user runs the program (CCDREAD) with the number of samples per second as the first command line parameter. This is sent to the PIC at startup, which then continuously samples the CCD at this rate, returning the distance information at each sample.

THE FUTURE

A CCD-based laser ranging system has been designed that is compact, low-powered, and has no moving parts. It is useful in a wide range of environments with varying object color, angle, and lighting conditions. The angular field of view is limited only by the ability to collimate the diode laser, and it can be constructed cheaply with readily available components. There are some situations, however, where the device will not function well. The main problem is with highly reflective or transmissive objects (mirrors and glass in particular). This could be avoided in critical applications by employing a combination of other sensing techniques such as sonar.

Several design changes are currently being made to this device. These include the use of a higher-

Listing 1—CCDREAD.C is IBM PC code to set up the CCD range finder and display the result of the measurements.

```
i/include <stdio.h>
#include <stdlib.h>
#include <bios.h>
#include <conio.h>

#define port 0          /* Serial port for range finder */

main(int argc, char *argv[])
{
    unsigned freq, count;
    double period;

    if (argc<2){
        printf("Usage: CCDREAD <Frequency>\n");
        exit(0);
    }

    /* Convert freq into delay time based on clock rate of PIC */

    freq = atoi(argv[1]);
    period = (((1.0/freq)-0.004941)*1E6)/3;
    count = (unsigned int) period;
    if (period<0){
        printf("Maximum sampling rate exceeded\n");
        exit(0);
    }

    bioscom(0, 0xe3, port); /* Set serial port for 9600,N,8,1 */
    bioscom(1, 0x00, port);
    bioscom(1, 0xf0, port); /* Signal start of transmission */

    /* Now send the delay for the specified sampling rate */

    bioscom(1, (char)(count & 0xff), port);
    bioscom(1, (char)(count >> 8), port);

    /* And print the current range returned until a key press */
    while (!kbhit())
        printf("%d\n", (bioscom(2, 0, port) & 0xff));
}
```

resolution CCD, the improvement of the laser diode AGC circuitry, and the design of an adjustable and stable optical arrangement. The device as described here was constructed as an experimental unit, but for continuous mobile use, the mechanical stability of the laser diode, CCD, and optics are important. Multidimensional adjustment facilities must also be provided for the imaging and collimating lenses to allow for focusing and the placement of the image within the active area of the CCD.

Patents are pending on this design, and due to the large response that I received from the original publication, I plan to arrange the production of sensor modules with a manufacturer in the near future. ☒

Tom Ward is an electrical engineer with particular interest in embedded control. He is resident in Australia, and can be reached by answering machine or fax at 61-7-871-0921.

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.

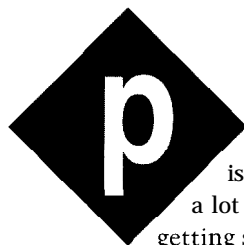
IRS

401 Very Useful
402 Moderately Useful
403 Not Useful

A Robotics Design Seminar

FEATURE ARTICLE

Robert Angelo



Personal robotics is of interest to a lot of people, but getting started can be a daunting task. In this article, I will try to give you a starting point by investigating some key factors in designing a robot for personal use. The topics include base design, drive motor control, a little about a control computer, and how to give the robot eyes.

The first thing you need to determine is what the robot will do, or better yet, what you want to learn from it. Should this be built for experimenting, or should it really do some sort of useful work? My suggestion is to consider this as an experimental project—the “useful work” part is to learn some useful things for future use. I’ll describe an expandable system here and give some thought to future enhancements.

BOTTOMS UP

Base design is more than a minor question. I have designed and built some 20 robots and have updated and/or repaired a number of others. Most of the robots I have worked with were used for entertainment purposes, while others were for Hollywood studios. It has been my experience that base design has seldom been given enough preplanning or thought. Let’s look at some possibilities.

A square base may seem simple, but brings some maneuvering prob-

lems into the equation. A round shape is better for maneuvering, but can be tougher to build. The best compromise I have found is an octagonal shape. Examining the maneuvering issue for a minute, collision recovery is the biggest problem. A square base can get into situations that are hard to recover from, since a turn recovering maneuver often won’t work. Another situation arises as the robot tries to go past a chair. If the robot is square and is not quite far enough away from the chair, a collision is probable. If the base was round or octagonal, it would likely just slide by the chair. This same scenario would apply for going through a doorway. See Figure 1.

Therefore, start with an octagonal shape for the base. It provides straight sides for mounting things, so makes construction a bit easier. This shape still offers angled sides for collision recovery, and can also use a counter-rotational maneuver to recover from being stuck in a corner (like a round base could).

MO, MO, MO’ MOTORS

The next issue of concern is the drive system and wheel placement, with steering and weight being the two biggest considerations. A three-wheel versus four-wheel design is an obvious question. A three-wheel robot will almost always have all the wheels in contact with the ground, where a four-wheel robot can easily have one wheel lose contact with the ground. Consider moving from a carpeted floor to a tiled floor or vice versa. If the robot is not

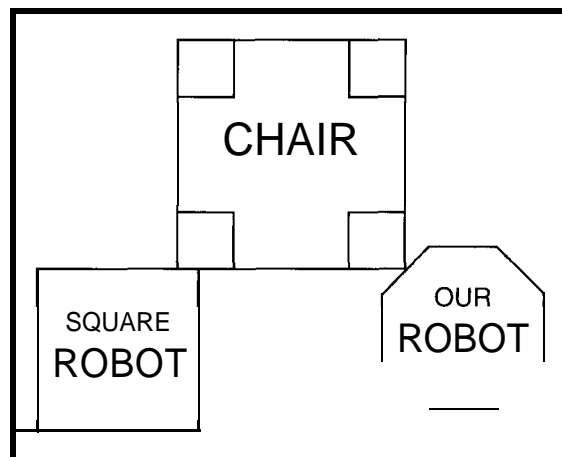


Figure 1—One advantage to having an octagonal-shaped robot versus one that is square is its ability to get past objects and around corners.

For some, robotics is a topic that conjures up images of fully automated servants ready to fulfill your every whim. In reality, that may be in the future, but for now, get started with something a bit simpler.

perfectly square with the threshold, it can very easily lift one wheel (Murphy says it will usually be a drive wheel) for some period of time, causing some degree of turn. A three-wheel design is far less susceptible to this problem. If we are building a small robot, weight being in the range around 45 pounds, a three-wheel design is quite good. It is simple in concept, easy to build, and simple to control. Refer to Figure 2.

Steering is next. If you drive two of the three wheels and have a simple caster as the third wheel, steering is simply the switching of motor direction. If you drive one wheel and try to steer with the same wheel by rotating the vertical axis, you need two stronger motors than would be necessary if driving two wheels. A related problem is maneuvering. With two drive wheels, you can counter-rotate your way out of a collision. This maneuver also works with a single drive/steering wheel design. With a four-wheel design, you would also drive two wheels and have the same off-center rotational axis.

The size of the base is another concern that must be addressed very early in the robot's design. Most interior doorways are between 28 and 31 inches wide. If we make a robot that is about 17 inches wide, it should be able to make it through doorways easily.

Taking all of the above into account, I'd suggest a three-wheel design. Referring to Figure 2 again, two drive wheels are placed at or near the center line, which can be drawn from front to rear, and a caster is positioned at the rear. The battery is mounted between the wheels and the control electronics are mounted in front. I believe this layout offers the best compromise between ease of construction and controllability.

I use a "clam shell" design in this robot, allowing the addition of side panels which can be used to make the assembly whatever height is needed or desired. One feature of this design is the use of a motor and wheel combination, wherein the wheels are mounted directly on the gear case output shaft. This eases some of the burden from a

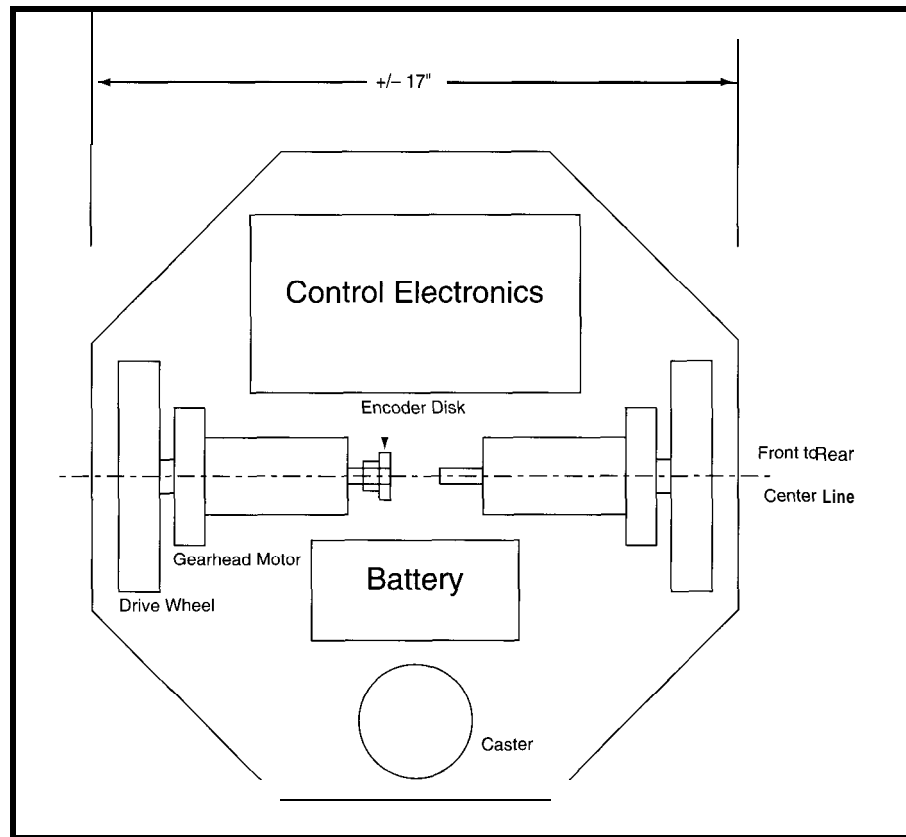


Figure 2—One of the more popular and easy-to-build designs is the three-wheel robot. In this case, two wheels are motor driven while a caster serves as the third wheel.

mechanical design standpoint and saves a lot of space.

The motors I use are Brevell's model number 780-982959. These are 12-V motors that have nice mounting tabs for easy attachment. The wheels have a V-shaped tread design providing good forward traction and low turning drag. This design allows approximately one inch of clearance under the robot. Using this drive system, the finished robot can weigh as much as 45 pounds and still maneuver over carpet.

The battery is a sealed lead-acid gel cell. This type of battery has shown very good power density and is much easier to keep track of. NiCd batteries don't let you know when they are about to die, they just do it. In contrast, sealed lead-acid batteries have a discharge curve that gives you a warning before they are about to run out of power. I chose a 12-volt, 6.5-AHr battery due to its availability and relative low cost. This type of battery is used in many of the security systems as the battery backup source, so check with your local alarm installer for more information about

where you can find them. A somewhat larger battery may also be used if more power is needed in your robot.

MUST HAVE BRAINS

How can you control your new robot? This is where the real fun began for me. I am using an 8051-based microcontroller for my robot. There are plenty of generic boards that I could have used with plenty of software development tools to boot. But, being an enterprising sort, I designed a control board that uses the 80C535 microprocessor from Signetics. This microprocessor has some very nice features, including an eight multiplexed A/D converter, compare registers, two 8-bit I/O ports, a serial port, and three counter/timers.

I also ported the MCS-51 BASIC (BASIC-52) to this processor. While I was knee deep in the code, I added some robot-specific commands. The extended command list is shown in Table 1. I also deleted some of the generic commands that would not be of use to a robot, but did keep the

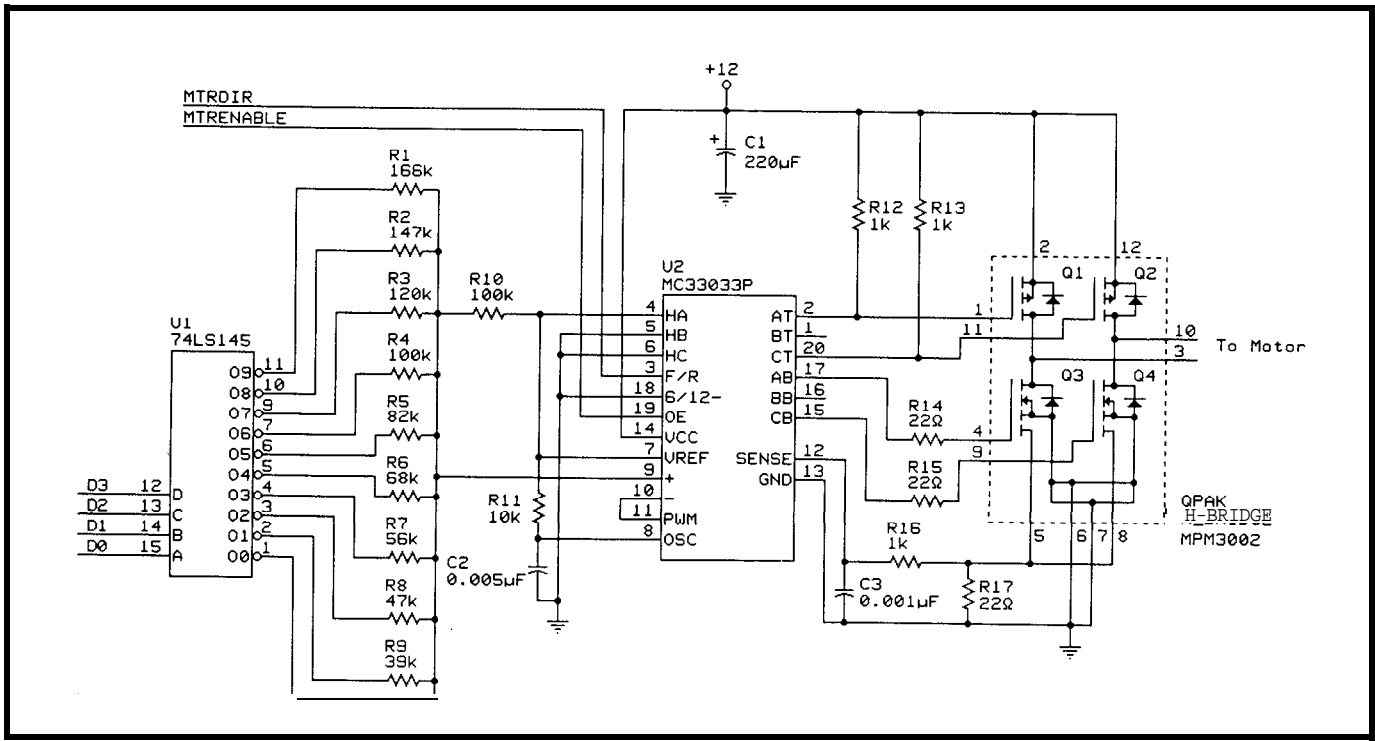


Figure 3—Motor direction control for the octagonal robot comes via an MPM3002 full H-bridge, while pulse width modulation is supplied by an MC33033P PWM controller.

floating-point math since I could envision an eventual use for it.

Since this controller was destined to become a robot controller, I added hardware to the controller board that would be specifically needed by a robot. For instance, the PWM motor control system is on the controller board, as is the ultrasonic ranging interface (I'll discuss both in a bit). I provided a small prototyping area and two expansion connectors for adding Micromint BCC bus boards. Three memory sockets are provided so many different RAM and EPROM combinations are available.

I use one of the "extra" 8-bit ports for on-board needs, while the remaining port's signals are on a separate connector. The A/D converter signals are also available on a separate connector.

There is also an encoder disc on the left motor shaft extension. I used an optical encoder to keep track of distance or turning angles. Because the disc is connected to motor shaft, it

rotates at the speed of the motor shaft rather than the velocity of the wheel. I made this decision because it makes distance measuring more accurate. By connecting the encoder's output to one of the microprocessor's counter inputs, we can set up the software so the counter provides an interrupt when the robot has gone a predetermined distance. With one slot in the disc, the resolution is 1/8" when using the wheels shown. The only rule that now needs to be in force is all turns are counter-rotational with motors turning in opposite directions. The on-board A/D converter is used to keep track of battery condition. The other analog inputs are free for any future application.

Motor control is provided with a current-controlled Pulse Width Modulation (PWM) circuit. I selected power MOSFET transistors in this circuit because of their low voltage drop and, consequently, low heat dissipation. As a rule of thumb, when heat is generated in a circuit, energy is being wasted, and energy is precious in a battery-operated system. I selected an H-bridge as the final control stage for the robot's main drive motors.

SPECIFICALLY SPEAKING

The H-bridge circuit allows current reversal for motor direction control. Figure 3 shows a sample of the circuit used in this design. I could have used simple relays for this purpose and

some kind of voltage control for speed control, but this becomes bulky and the voltage control circuit would also waste battery power.

If bipolar Darlington transistors were used instead, they would drop between 0.7 and 1.4 volts. This

MOV XXX, YYYY	XXX = direction FWD, REV, CW, CCW YYYY = distance in inches or degrees of turn
MOV FWD, 100	Move forward 100 inches
MOV CCW, 90	Turn left 90 degrees
SCALE XXX	If a different drive system then suggested, a scale factor must be given prior to using these commands
RANGE	Returns distance to nearest object, to the nearest inch
PORT5	Read/assign I/O port 5
MUX#	Select analog mux
ANA	Returns value of A/D converter

Table 1—The system includes several new robot-specific BASIC-52 commands in place of some less useful ones.

would no longer be applying pulse width modulation to the motor and would be using more battery energy than is necessary. The output of the DAC is fed to the MC3033P.

The MC33033P is a current-controlled pulse width modulator circuit. MPM3002 is a full H-bridge in a single package that also provides a superior current-sensing output. This device uses a "sense-FET" since the lower N-channel transistor that doesn't add any current loss to the circuit. The MC33033P chip was designed as a brushless motor controller, however it also works well in this application. What we are really doing is setting the current value for the motor, thereby determining its speed.

You'll need to experiment to determine the relationship between current value and the robot's speed under various conditions. One big influence is the floor surface the robot is running on. The motor stall point is also manageable with this circuit. Much as I would like to take the credit for this design, it is straight out of one

of Motorola's engineering bulletins. The MC33033P PWM controller also gives a motor enable and direction control inputs. It may sound complicated, but it does work well and provides a very efficient computer-controlled motor control system.

MURPHY'S LAW OF NEWTONIAN PHYSICS: MOVING OBJECTS WILL COLLIDE

It would probably be nice to have some idea if there is something in front of the robot, so I included a Polaroid ultrasonic system in the robot for this purpose. A processor-controlled system has the drawback of not accurately timing echo response due mainly to things like interrupt latency. A not-so-obvious conclusion is to offload the task from the processor as much as possible. See Figure 4 for the circuit design I used. Signals INIT and ECHO are connected to the Polaroid ranging module. The 555 timer is used to clock the 74LS590 counter. The timer is set to clock the counter at about 6.3 kHz, selected because it is

roughly equal to the round trip per inch time interval (148 microseconds, more or less). The timer can be fine tuned to account for the temperature of the environment, since temperature is the biggest influence on accuracy of this system.

The 74LS590 is an 8-bit counter with clock- and output-enable functions built in. This allows connecting its output directly to the computer's data bus. The 74HCT74 flip-flop is used as a control register. On power-up, the flip-flop is cleared, which sets INIT off and disables the counter.

The timing diagram in the schematic shows the relationship of the signals. SONARGO starts the process by clearing the counter and asserting INIT to the ranging module. Signal ECHO from the ranging module goes low after INIT goes high, which enables the counter to begin counting. When an echo is received, ECHO goes high, the counter is disabled, and flip-flop sets INIT back low.

Signal SONARTO is asserted if no echo has occurred prior to the counter

PLIX™ HARDWARE X-10™ TRANSCIVER CHIP

Micromint introduces its new Power Line Interface for X-10. Plix is an 18-pin ASIC chip that automatically handles all the specialized X-10 timing and bit-shuffling between a computer and a TW523 power line module.

- ✱ Complete interface between parallel port and TW523 or PL513 modules
- ✱ Performs all X-10 transmit and receive functions
- ✱ Detects AC power loss
- ✱ Simple interface and timing - operates even in interpreter BASIC
- ✱ Low power - only 1.8 mA @ 5V

Plix chip and data sheet
100 qty. OEM

**\$20
\$12**

PLUS SHIPPING

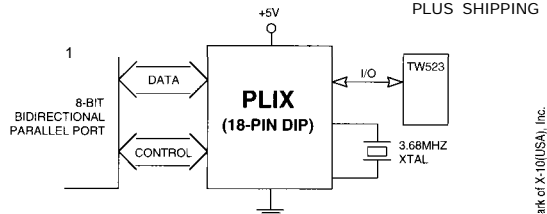
Call **1-800-635-3355**

or Write for a Plix data sheet

PLIX EVALUATION BOARD

To speed your evaluation, we have produced a low-cost Plix interface that plugs into a parallel printer port. The Plix-EKit includes a Plix chip and data sheet, PCB with all components, application note, and PC-compatible software on diskette.

Add \$30 for TW523 X-10 transceiver module **\$39.00**
PLUS SHIPPING



MICROMINT, INC.
4 PARK ST. VERNON, CT 06066
(203) 871-6170 FAX: (203) 872-2204

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

ACSBUS HCL1 & 8052

\$287 (Qty 1)

\$169 (Qty 1)

HAVE BUS with Immediate Availability of the following "Connections"!

CEBus, CONTACT IN/OUT, A.C. SENSE, 64-BIT HIGH CURRENT OUT TRANSDUCER, TRIAC OUT, PRECISION MOTION CONTROLLER, CR1 USER INTERFACE (LCD Display, Keypad, Indicators, Panel Printer, Parallel Printer RAM EXPANSION, DIGITAL AUDIO, A/D, D/A, MODEM & FACSIMILE Remote Program Loading, Windows™ Remote Control Software.

More Immediate Connections Available Every Day.
Call or Write for more info!



On The Cutting Edge of Technological Evolution

4276 Lago Way • Sarasota, FL 34241
Ph. (813) 377-5775 • FAX (813) 378-4226



reaching its maximum count [which is, of course, 255]. An interrupt can be generated if ECHO and SONARTO are logically ORed together, and a status port can be read to see if an echo or timeout has happened.

So what exactly is the scenario at this point? A SONARGO signal is generated by the processor, INIT is asserted, the counter begins counting and waits for the ECHO signal to come from the ranging board. The counter can count to a maximum of 255 inches, which is equivalent to 21 feet, 3 inches. If no echo is received prior the counter timing out, SONARTO is asserted and the process can be aborted, which means either there is no object in front of the robot or there is no detectable object in front of the robot. Software would be used to determine whether or not to continue on the present course or turn the robot a small amount and look again. Another possibility would be to mount the transducer on a shaft that could be driven by a servo or stepper motor. We could then "look" around as needed.

YES, BUT IS IT A HOVERCRAFT?

We now have a robot base shape design that should meet our needs, at least to begin with, and a motor drive system that makes efficient use of the battery. We also have the means to see the environment and maybe map it, and a computer that offers enough power to manage the system. In short, it is the basis of a system that can grow with your needs and desires. The possibilities are limited only by your imagination. Some thoughts to ponder: add a training leash or wireless communication link to a host computer, IR links, or maybe radio links. I would enjoy hearing about your ideas and how your project grows. □

Robert Angelo has been involved with research and development for 15 years in the electronics field. For the last 8 years, he has consulted independently in the concept-to-production of PCBs for robots, which has included projects for Hollywood and trade shows.

SOURCES

AUROtronics
11920 N. 76th Dr.
Peoria, AZ 85345
(602)486-4773

Wheels described in article, robot controller (with or without MCS-5 1 BASIC), and PWM dual motor controller. Send SASE for info.

Johnstone Supply, Inc.
(503) 256-3663

Breve1 Motors.

Micromint, Inc.
4 Park St.
Vernon, CT 06066
(203) 871-6170

Polariod ranging systems, BCC
Bus control and peripherals.

I R S

404 Very Useful
405 Moderately Useful
406 Not Useful

The Soft-Scope[®]/CSiMON debugger and CSi-Locate[™] combined with your existing compiler create a complete embedded systems development environment

Absolute or Hex files

New **CSi-Locate** builds absolutely located files or hex files from applications compiled using Microsoft, Borland, or Watcom tools. Soft-Scope also supports Intel, Metaware, and Symantec tools.

Easy Configuration

Configure a **CSiMON** ROM monitor in minutes for most targets.

Versa tile

For the entire 8086 family of processors, in real or protected mode.

Cost effective

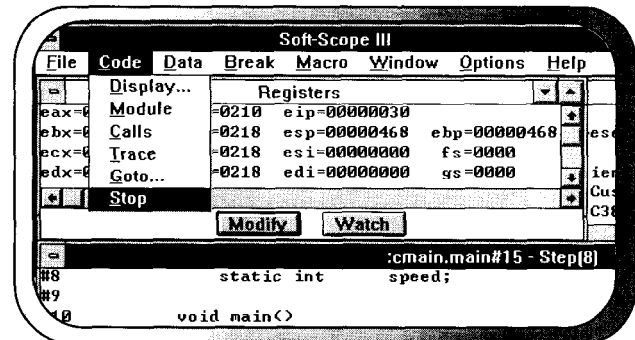
Include the *royalty-free* CSiMON ROM monitor as part of your product.

Real-time support

Intel's iRMX, Industrial Programming Inc's. MTOS-UX and JMI Software Systems, C Executive fully support **Soft-Scope**.

Established

Concurrent Sciences developed the original source-level debugger for the 8086 processor in 1983, and is known industry-wide as the expert in 32-bit protected-mode development. Soft-Scope fully supports the Intel386 EX processor.



“ We have about 750,000 lines of source code in our product, running on an 80486 in both 16- and 32-bit protected mode, and WE are very pleased with how well Soft-Scope deals with multitasking and mixed 16/32-bit code.

Gerd Hoeren, Integrated Measurement Systems Beaverton OR

CONCURRENT SCIENCES INC.

PO Box 9666 Moscow ID 83843 • (238) 882-0445 • FAX (208) 882-9774

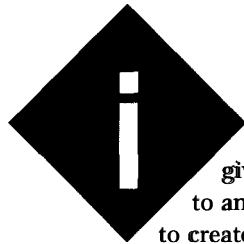
Soft-Scope is a registered trademark and CSi-Locate is a trademark of Concurrent Sciences Inc. All other trademarks are the property of their respective owners.

FEATURE ARTICLE

Michael Swartzendruber

YAMCI-Yet Another Motor Control Interface

If you've been looking for a motor control system for your next robotics project, look no further. Michael covers the details of a basic setup that's perfect for any level of expertise.



If you were to give an assignment to any two engineers to create a circuit to

perform a certain task, and these two engineers had no communication with one another and shared no information during their design process, how likely is it that their designs would be identical? Given the number of competing technologies, components, and suppliers, you would think the odds would be quite small. In the end, other factors (cost, manufacturability, supply issues, reliability, testability, packaging constraints, etc.) will decide which design is dropped and which is retained. The whole point is there are many roads to Rome, only some are better for different reasons.

The same idea applies to motor control circuits. There are as many possible implementations as there are control technologies. You could opt to build an H-bridge from a small number of amplifier circuits constructed entirely from discrete components, or you could opt for a fully integrated single chip controller, such as those offered by Motorola or National Semiconductor, or you could choose a method that falls somewhere between.

When I set out to design this motor control interface, I considered

what would be most important on the list of requirements, and came up with the following list:

- *Direction control (forward or reverse)
- *Fine speed control through the use of PWM
- Two coarse speed ranges: high and low
- *Motor "lock out" to prevent the motors from being accidentally activated when the system powers up
- Low chip count, but without using a costly single-chip controller

The criteria shown above could be sorted into a number of areas. For instance, the "lock out" requirement is viewed as a safety-enhancing feature to keep the motorized device from uncontrolled and unintended operation during the power-on sequence. Since the controller might not get around to setting up the motor's control port for some undetermined amount of time while it gathers its wits, a lockout ensures that the motors won't have the potential to damage the system or any other object in the instrument's environment during the power-on sequence.

The forward and reverse control as well as the PWM inputs could be viewed as a necessity. After all, what use is a motor control system that would not provide even this simple level of control?

The coarse speed control is an extra item that I wanted to add to the list since I considered it to offer a great deal of versatility to the control circuit. By combining a coarse speed control with a PWM input, you get some of the advantages of a multiple-speed, multiple-torque system without having to supply the additional costs, design time, or mechanical overhead of a multiple-gear transmission.

BLOCK BY BLOCK

Figure 1 contains the schematic of the motor control interface I created. As you can see, it is a pretty straightforward design that combines a lot of well-understood technologies into a single system. The system consists of a lockout control circuit, a logic

circuit that provides control signal "decoding" and other combinatorial logic functions, a motor current switching and control circuit, and a set of current driver outputs that are used to drive relays for the battery circuit.

The lockout function is provided by a garden-variety LS14 that is wired as simple reset circuit. In one sense, the circuit acts as a delayed locking switch. The purpose of this circuit is to hold the RDY signals at a low state for a period of three seconds. This arbitrary delay factor can easily be changed to any other time delay that you consider more appropriate by selecting new RC components. As long as this signal is in the low state, there is no way that battery power can be applied to the motors. So even if the control lines come on in some unknown state, the motors cannot be accidentally powered. By providing access to the

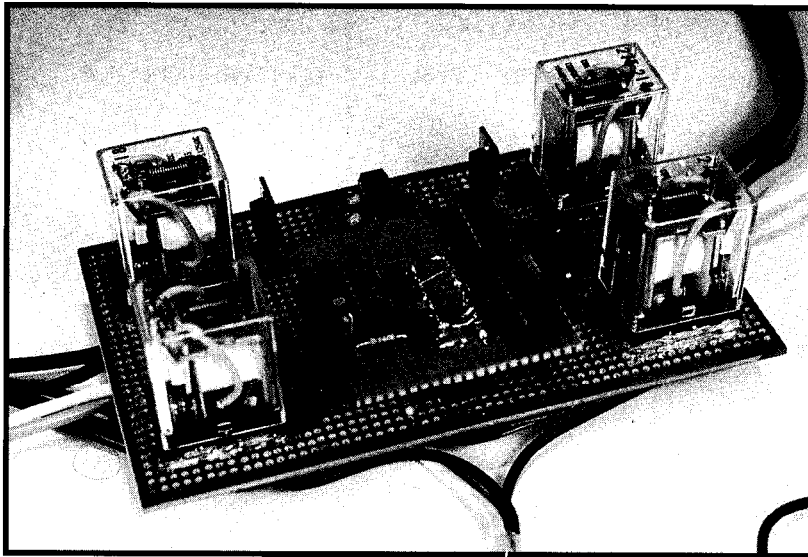


Photo 1--The motor control interface includes four 12-volt, 10-amp relays and a "mezzanine card," used to replace a tempermental PEEL chip with discrete logic.

discharge path of the RC circuit, the system can lock out the motors at any time by pulling these lines low. This could give the processor time to "think" about what to do next without having to "worry" about damaging anything.

The RDY signal is input into two other devices in the system. First, the signal is routed to a 7400 where it is mixed with the CSPD (Coarse Speed Control) signal. The outputs of the NAND gate are inverted and used to

provide control signals to the motor speed relays. This provides you with a way to lock out motor speed commands, or you can use this logic to disconnect the battery power from the speed control relay while you are "changing speed." This could be a useful feature, but you don't have to drive it that way.

The second place the RDY signal goes is directly into the

relay driver chip which is used to control the relay that connects/disconnects battery power from the motor circuit.

The RDY signal is fed to a MOSFET driver chip that is used to control a relay. The relay is connected in the battery power circuit and is wired such that the only time battery power is made available to the motor circuit is when the relay coil is activated. The relays I used for this project are Radio Shack #275-218.

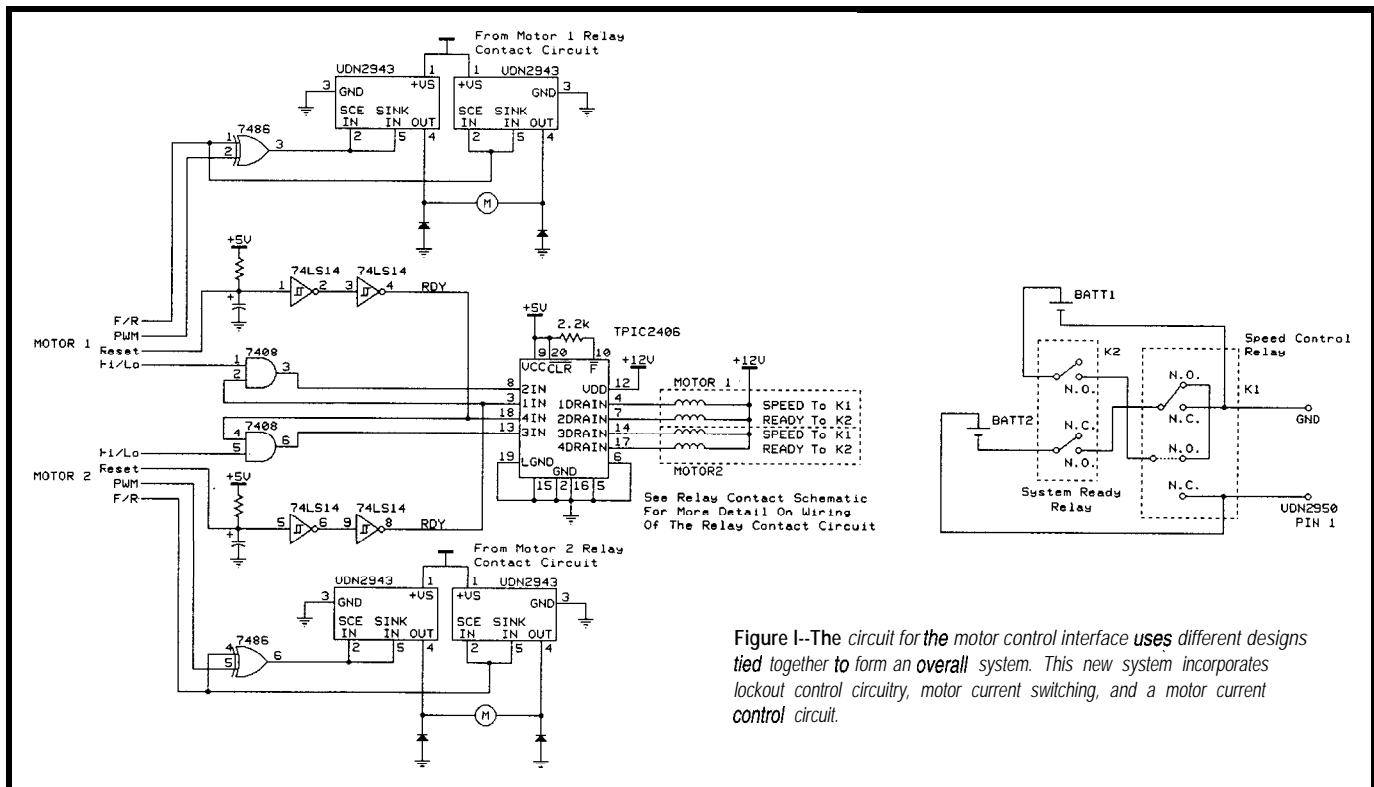
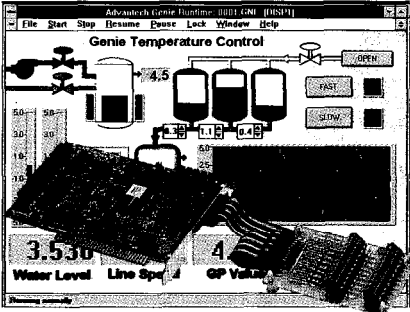


Figure 1--The circuit for the motor control interface uses different designs tied together to form an overall system. This new system incorporates lockout control circuitry, motor current switching, and a motor current control circuit.

Data Acquisition & Control Solutions

Temperature Measurement
High Speed Data Streaming
Remote Data Acquisition
Strip Chart Recording



Temperature Measurement & Control Solution Package ~\$895

Key Features:

- ❖ For J, K, S, T, B, R, E thermocouples
- z+ 8 thermocouple channels, expandable to 32 channels
- ❖ 0.1°C resolution with 12-bit A/D
- Z+ Up to 250 samples per second
- ❖ 16 D/I and 16 D/O channels
- ❖ Powerful Windows SCADA software
- ❖ Icon based, no programming required

1 General Purpose DAS Card 8 A/D, 1D/A, 16/16 DIO, wiring kit	\$295
1 Multifunction DAS Card 16 A/D, 2 D/A, 32 DIO, counter	\$335
⌈ High Performance DAS Card 100KHz, programmable gain	\$595
⌈ 24 Ch Digital I/O Card	\$100
1 24 Ch Digital Input w/Interrupt	\$180
⌈ 32 Ch Digital I/O Card	\$170
⌈ 32 Ch Digital Input w/Interrupt	\$225
1 144 Ch Digital I/O Card	\$235
1 8 Relay & 8 Digital Input Card	\$210
⌈ High Speed Data Streaming Pkg. 100KHz data streaming to disk	\$555
⌈ PC Strip Chart Recorder Pkg. 16 channel recording at 15KHz	\$635
⌈ Remote DA&C Starter Kit RS-485 based, up to 4000 feet	\$435

Free
120-page Solution Guide
for quality minded,
budget conscious
Engineers
1-800-800-6889



Industrial & Lab Automation with PCs
ADVANTECH.

750 East Arques Ave. Sunnyvale, CA 94086
Tel: (408) 245.6678 FAX: (408) 245-8268

FWD/REV	PWM	state	Chip1 Pin2	Chip1 Pin5	state	Chip2 Pin2	Chip2 Pin5
0	0	L	0	0	L	0	0
0	1	L	0	0	H	1	1
1	0	H	1	1	H	1	1
1	1	H	1	1	L	0	0

FWD/REV	PWM	Chip1 Pin2	
0	0	0	
0	1	0	follows FWD/REV
1	0	1	
1	1	1	

FWD/REV	PWM	Chip1 Pin5	
0	0	0	
0	1	0	follows FWD/REV
1	0	1	
1	1	1	

FWD/REV	PWM	Chip1 Pin2	
0	0	0	
0	1	1	XOR of inputs
1	0	1	
1	1	0	

FWD/REV	PWM	Chip1 Pin5	
0	0	0	
0	1	1	XOR of inputs
1	0	1	
1	1	0	

Figure 2—The motor control signals can be described in black-and-white terms using truth tables. These are then used to define the discrete logic on the board.

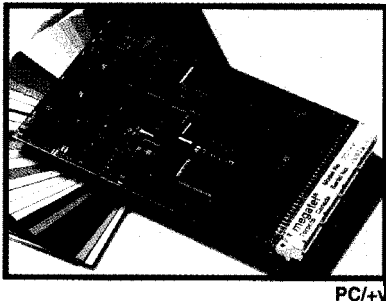
These particular relays have a 12-volt coil and the contacts are rated at 10 amps. You could probably choose any other relay in these positions. I chose them because there were available and had more than enough head room in the current handling capacity of the contacts. The coil resistance of these particular relays is 150 ohms. So, if these are driven with a 12-volt source, and with no other resistance in the coil circuit, the current in the coil circuit is 80 mA. This falls well below the 770-mA-per-channel capacity (nominal) of the MOSFET driver chip.

The power MOSFET device is a TPIC2406 from Texas Instruments. I chose this device to simplify the design of relay coil driver a great deal. It provides a logic-compatible interface for the control signals and also

provides output drivers that are easily capable of driving the relay coils used in the motor control stage of the system. The drivers in this device can pulse control up to 3 amps per channel, with up to 12 amps total pulsed through the device. This system requires four such drivers, and the fact that this device contains that many is another plus in its favor. Pushing the logic-to-coil interface into a single device goes a long way in reducing the parts count for this circuit. The TPIC2406 is designed specifically for this kind of application and includes built-in protection from transients caused by inductive back-EMF. In fact, each channel can safely absorb up to 50 millijoules of inductive energy. Even though this device can latch (in what is present on the input pins (in

Embedded PC

**"The new megatel
PC/+v offers on-board
VGA in a small rugged
form factor."**



On-board Features Include:

- 16 MHz V-40 CPU
- 640KByte User DRAM
- On-Board VGA Video/LCD display controller
- Power Consumption under 2 watts, +5 volt only operation
- 3 RS-232 Serial Ports
- Floppy, SCSI, Keyboard Ports
- Parallel, Printer Port
- PC Compatible BIOS
- Real Time Clock
- Only 4" x 6" (100 x 160mm)
- ISA and PC/104 Bus Options

**For a list of our International Distributors
please contact our head office at:**

megatel computer corp.

125 Wendell Ave.

Weston, Ont. M9N 3K9 Canada

Fax: (416) 245-6505

(416) 245-2953

ECC

EMBEDDED COMPUTER CONFERENCE
& EXPOSITION

JUNE 8-10, 1994

SANTA CLARA CONVENTION CENTER
SANTA CLARA, CA.

megatel®

other applications, this feature can be used to save the processor from having to baby-sit the chip, not to mention saving a latch), I am using it in a transparent mode since I'm mostly interested in its drive capabilities.

To control the current through the motor, I used the UDN2950 from Allegro. These devices also integrate a logic-compatible interface with high-current source and sink devices in the motor output stage. They also have integrated transient protection, so the need for external diodes is eliminated. These devices also contain protection circuitry that prevents shorts in the device caused by faulty control signals or by rapid switches in motor direction. In addition to these features, these devices also include overvoltage, overcurrent, and overtemperature protection. In short, they are designed to protect themselves, making them more robust and reliable. Barring the catastrophic, these devices are well designed for the typical motor control application.

Even though these devices are specified as half-bridge controllers, by placing a pair of them as shown across a motor load, you can make a full-bridge implementation. The trick to running these devices in this configuration is to provide them with the right mix of control signals. To derive them, I set up the simple truth table shown in Figure 2. I knew that the direction control signal would determine which of these devices would be sourcing or sinking current. I also knew that I only wanted motion when the PWM input was high (that's the way it's supposed to work, right?). After I mapped out the truth table for all of the control signals, I then split the table apart so I could look at each control signal separately. This allowed me to see the logic functions that would be needed for each of the control pins on the two devices.

I also used the same process to derive the control logic of the newer UDN2943. This device (also from Allegro) performs the same general function as the 2950, but use a different mix of control signals. I have worked out the control logic for these chips, and it is shown in Figure 3.

The last bit of cleverness I built into this interface has to do with the relay that is driven by the CSPD (Coarse Speed Control) control line. The wiring of the relay contacts changes the ways the batteries are wired to one another when the relay changes states. In one setting, the batteries are in series with each other, while in the other setting the batteries are wired in parallel. This little trick is how I provide for two coarse speed ranges with this interface. The PWM input can be used in both of these speed ranges to provide additional control over the motor speed.

You will also notice that the motor supply is completely isolated from the logic and interface power delivery systems. This is done in an effort to keep motor noise from getting into the supply systems for the controller's logic. It might be more complex, but you don't want your controller getting scrambled bits every time a motor fires.

I originally tried to incorporate all logic functions required by this circuit into a PEEL, but had some problems, so fell back to using discrete jelly beans. The circuit now operates in a rock solid fashion. In fact, this is the reason for the little "mezzanine card" on the board. Since I had already wired the prototype board for the PEEL, and I didn't want to rewire the circuit card, I created a card that would just plug into the socket on the main board.

ENHANCEMENTS

The one drawback I can see to this design is that it is an open-loop system. The motor control devices themselves are rather minimal, and do not provide any current sensing output. In order to implement a closed-loop system, you may consider one or both of the following options.

One option would consist of a current-sensing circuit in one leg of the connection between the motor and the control devices. This circuit could be as simple as a low-value resistor providing input to an additional circuit used to the monitor voltage dropped across this same resistor. Something as simple as a comparator could be applied in this voltage-drop

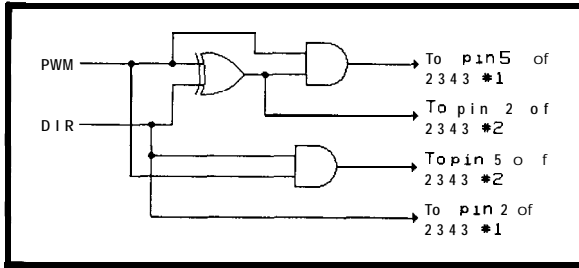


Figure 3-Control logic for the UDN2943 device was accomplished using the same truth table technique used for the other board logic.

sensing circuit if you were only interested in monitoring for a predetermined current level/voltage drop through the resistor.

The other option would be provide an encoder feedback on the motor's output shaft to measure/monitor the rotational velocity of the motor. Using these two methods you could provide sensory input that would allow the control system to sense the speed of the device (useful for closed-loop PWM systems) and monitor for stall currents.

Like many circuits, it can be further enhanced to suit your own particular requirements. For instance, another nice enhancement to this

control circuit would be to add latching capabilities to the input circuit by inserting a latch between the inputs to the control logic and the control lines from the microprocessor.

A final enhancement worth considering is to add a dedicated PWM generator that could be set by the controller and left to run without requiring any further attendance. In certain applications, freeing the processor from having to generate PWM can be an architectural improvement. But, as many controllers offer a PWM output, it really becomes a tradeoff between the cost requirement of the additional components and the performance gain that comes from adding a dedicated PWM source.

WINDING DOWN

My intention in illustrating this circuit was to implement the core of a rather sophisticated motor control circuit with readily available and low-

cost technologies. I think I have been successful in that objective. One of the primary strengths of this design is that the parts count is kept relatively low. While it could be argued (probably successfully) that I could have lowered parts count even further with a fully integrated controller, I would counter argue that this solution is a compromise that attempts to maintain a low parts count while trying to bypass the single source and cost issues that could arise with a fully integrated, single-chip controller. □

Michael Swartzendruber is an engineer with experience in network and communications design and Windows and Macintosh programming. He is also a Technical Editor for the Computer Applications Journal. He may be reached at michael.swartzendruber@circellar.com.

IRS

- 407 Very Useful
- 408 Moderately Useful
- 409 Not Useful

REMOTE POWER CARD!

3 VERSIONS:

RESET
WAKEUP OR RESETS PC IF IT TRIES TO BOOT FOR HARDWARE OR SOFTWARE REASONS

PHONE
TURN ON PC WITH PHONE, SHARE VOICE / MODEM LINE, CONTROL AC APPLIANCES

TIMER
WAKEUP OR SHUTDOWN PC, LATE NITE BACKUP / MODEM, CONTROL AC APPLIANCES

95\$ 275 OEM

ALL MVS CARDS INCLUDE ASM, BASIC, AND C SOURCE FOR PC OR SBC

8 CHAN ADC

DATA ACQUISITION, SERVO CTL, AUDIO
9-BIT RESOLUTION 22KHZ SAMPLE RATE
SHARP CUTOFF ANTI-ALIAS FILTER
CREATE STEREO BLASTER (VOC) FILES

95\$

2 CHAN DAC

ALARMS, CIVILTY
VOICE MAIL, MUSIC
8-BIT RESOLUTION 44KHZ SAMPLE RATE
PLAYS MONO / STEREO BLASTER FILES
FUNCTIONS AS DIGITAL ATTENUATOR TOO

75\$

MVS MVS BOX 850 MERRIMACK, NH (508) 742-9507

5 YEAR LIMITED WARRANTY
PING IN USA

LOW COST DEVELOPMENT TOOLS

We have a complete line of 'C' compilers (MICRO-C) for: 68HC08, 6809, 68HC11, 68HC16, 8051/52, 8080/85/Z80, 8086 and 8096 processors. Cross assemblers for these plus others. Source code and porting packages are available!

Development Kits: \$99.95 + s&h
Includes C compiler, Cross assembler, ROM Debug monitor, Library source code, Editor, Telecomm program and everything else you need to do 'C' and Assembly language software development for your choice of processor.

Emily52, a high speed 8052 simulator: \$49.95 + s&h
High speed (500,000 instructions/sec on 486/33). Hardware emulation mode accesses real ports/timers on your target system. Includes PC hosted "in circuit" debugger.

BD52, a complete 8052 Development System: \$249.95+ s&h
Everything you need for 8052 development, including: Hardware: 8032 based single board computer with 32K ROM, 32K RAM, RS-232 port, Hardware debug support. Software: DDS MICRO-C Developers Kit, EMILY52 Simulator, PC Hosted "in circuit" debugger and kernel in ROM.

loaded BD52: \$299.95 + s&h
As above, with extra 32K RAM, 4 A/D + 1 D/A, 2K EEPROM, 7 line relay driver, watchdog and power monitor.,

Call or write for our free catalog.

DUNFIELD DEVELOPMENT SYSTEMS

P. O. Box 31044
Nepean, Ontario Canada
K2B 8S8
Tel: 613-256-5820 (1-5pm EST, Mon - Thu)
(BBS & Catalog Requests 24 Hour with Touch Tone)
Fax: 613-256-582 1 (24 Hour)

Wireless Remote Control of the AVMux

FEATURE ARTICLE

Steve Ciarcia

Oast month I detailed the design of my newly updated audio/video multiplexer (AVMux). Using a couple of new chips from Maxim and Analog Devices, the AVMux facilitates effortless switching of up to eight video channels and up to eight sets of stereo audio channel

pairs. Using the AVMux, I can effortlessly attach and reconfigure the connections between multiple VCRs, CD players, a Pro Logic decoder, a laserdisc player, and various other audio/video sources to the same set of amplifiers or in any number of different electronic combinations.

With the possible exception of the actual wiring chore itself, the basic multiplexer and control unit is quite straightforward and easily constructed. Unfortunately, solving the basic switching problems only served to create further design necessities. Let me explain.

The primary problem with commercial multiplexers (when they used to be available) is that they are housed in a box much like traditional stereo equipment with all the input/output jacks on the back. Such shortsightedness on their part also requires

What good is having ultimate control over your virtual audio/video environment if you have to get out of your chair to change the setup? Outfit your home theater in style by adding an RF interface to the AVMux.

Circuit Cellar Audio/Video Multiplexor Audio connections

from 3 - ADS CD Player Output	to 1 - Nakamichi Preamp In1
from 5 - JVC VCR Audio Output	to 2 - Nakamichi Preamp In2
from 4 - Dolby/SFproc Output	to 3 - Subwoofer Amp Input
from -----	to 4 - JVC VCR Audio Input
from 2 - Sony VCR Audio Output	to 5 - Mitsubishi Proj Ext1
from -----	to 6 - Mitsubishi Proj Ext2
from 7 - Laserdisk Player Out	to 7 - Dolby/SFproc Input
from 1 - Nakamichi Preamp Out	to 8 - DAT Recorder Input

Choose (A)udio, (B)oth, or (V)ideo

Photo 1--Making cross-connections is as easy as picking the device from the list displayed on the TV screen.

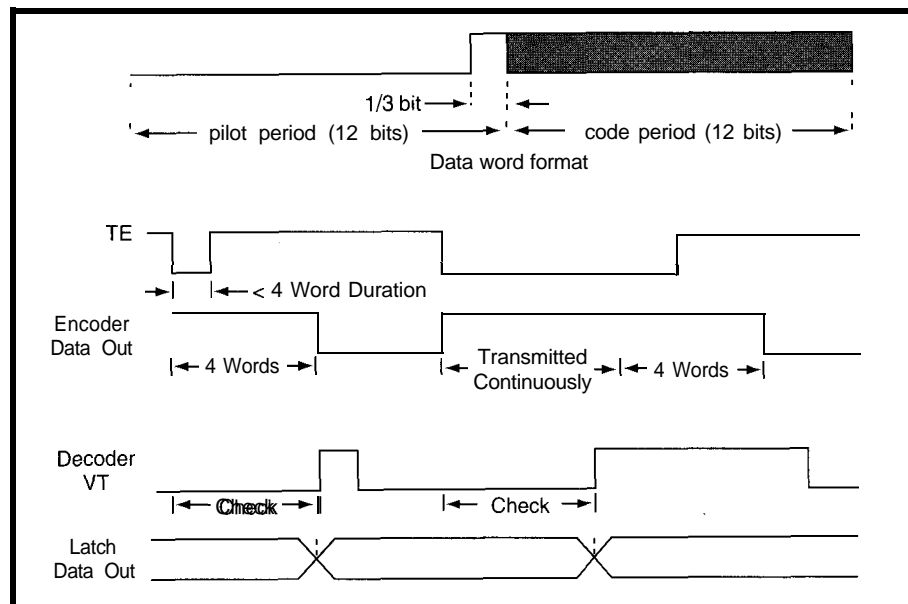


Figure 1-1 The Holtek Hi-12E and HT-12D RF chips use a 12-bit data format to communicate information

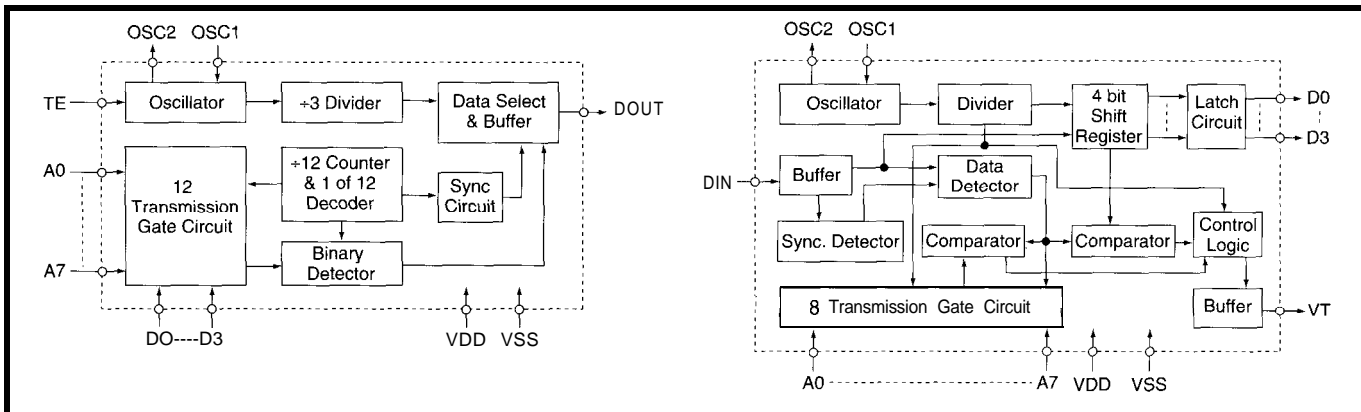


Figure 2—The Motorola H1-12E transmitter (left) and H1-12U receiver (right) chips take much of the hassle out of adding an HF data interface.

taking a chainsaw to your expensive CWD oak stereo cabinets to widen the minuscule wire access holes to actually route all these wires.

The front to the box isn't much better. It usually has an LED display which indicates the From-To connections. Of course, if you are sitting farther than 10 feet from the equipment, you need to dim the lights and use a pair of binoculars to read the display. Making a new program setting requires rolling out of the easy chair.

Being a designer with a decidedly sedentary disposition greatly influenced the configuration of the new AVMux. First, the multiplexer electronics and the audio/video connectors are housed in a single unit which is fully independent of display or control entry devices. The 16" x 5.5" x 2.5" multiplexer enclosure (pictured last month) can be physically mounted on the back of the entertainment system cabinet where the wires actually are.

I isolated the front display control panel and keypad entry functions as a separate hard-wired remote controller which can sit right on the arm of the easy chair. Pictures and schematics of this remote box are contained in Jeff Bachiochi's "From the Bench" article this month. In his presentation, Jeff explains the inner workings of this remote as well as the software specifics of the other interfaces I'm about to present here in a high-level way.

WHEN IS IT GOOD ENOUGH?

As I mentioned, my original AVMux concept was just the multiplexer box and the hard-wired remote

display/control out next to the easy chair. No fuss, no more getting up to change formats. No need for expensive field glasses.

AND I SHOULD HAVE STOPPED RIGHT THERE! EXCEPT...

The first consequence of this arrangement is that the Froms and the Tos are still listed by number. Audio output from the VCR may be connected to AVMux Audio Input 1 and the surround-sound decoder's input attached to AVMux Audio Output 7. Connecting them is as simple as punching in From 1 To 7. Remembering what is connected to all these

inputs and outputs and which are audio or video, however, requires a physical list. In short, I might now have a remote device in my hand that displayed a pretty XY matrix of connections, but unless I had a real good memory, I'd still need a listing next to the box to remind me what functions these numbers represented.

Soon the solution to the problem became evident. Since we were using an 80C52-BASIC chip as the controller for the AVMux, we had the performance of a full BASIC available at our disposal. BASIC-52 has among its commands the ability to reconfigure one of its PORT1 processor lines as a

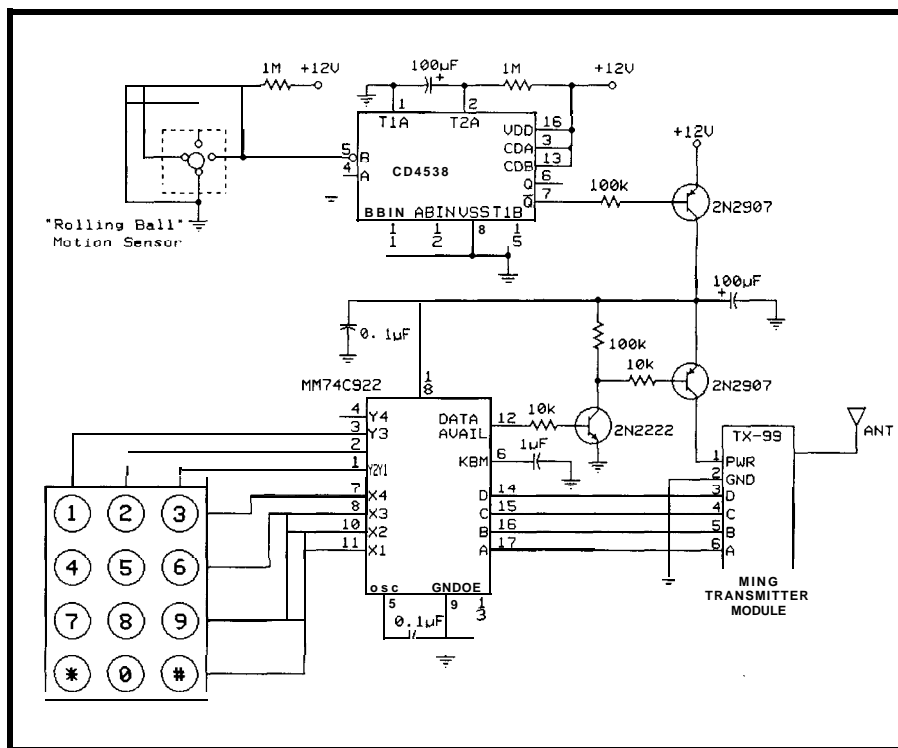


Figure 3—Adding a keypad, keypad controller (MM74C922), motion detector, and power control one-shot (CD4538) to the Ming RF transmitter module is all it takes to create an RF remote.

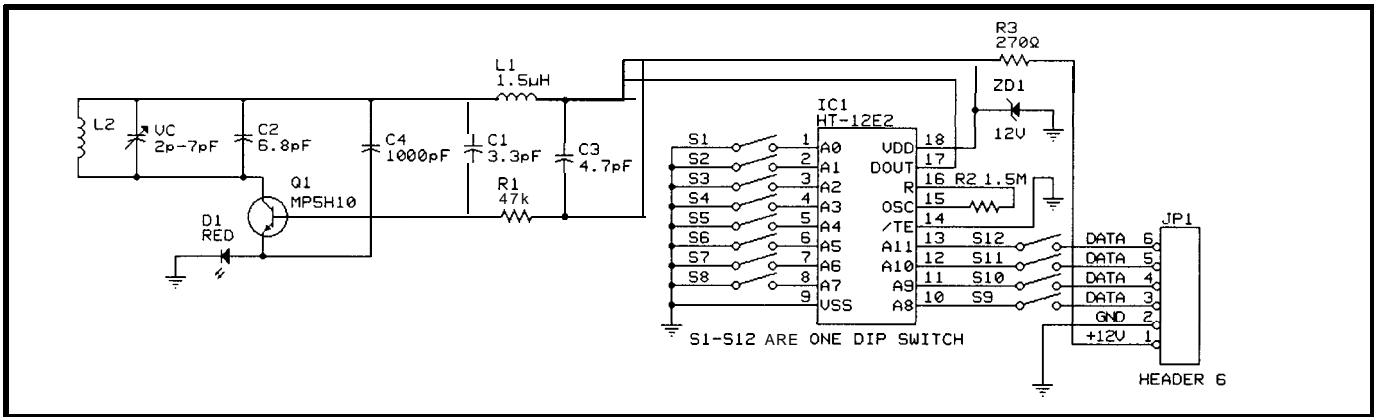


Figure 4—The Ming TX-99 transmitter module uses DIP switches to set an 8-bit address, while 4-bit data comes from an external device.

serial printer output port. PRINT# (LPRINT in other flavors of BASIC) dumps a formatted output string to this port as fast as 4800 bps.

In Jeff's control program, he allocated memory (battery-backed) so the user can physically enter (and store) an ASCII label for each input and output. For example, the From 1 audio connection could be designated "Laserdisc Audio Out" and the To 6 audio connection labeled "Pro Logic Decoder In."

Based on the configuration switch settings, when the system is initialized or any setting is changed, the control program actuates the connections, adjusts the LED display, and enumerates the connection list to the printer port. Connect a serial terminal to the port and you have an electronic listing.

I connected the AVMux printer port to a single-board ASCII terminal with NTSC video output. My intention is to apply this video to the external input of the projection TV.

When I change the AVMux settings, the video listing can be presented on the TV in a Picture-in-Picture box.

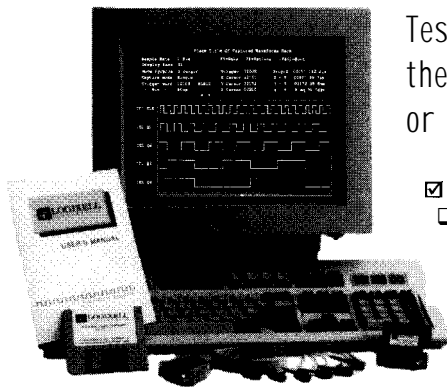
Once presented with a connection list or matrix that could be read from the easy chair, it seemed redundant to run a hard-wired remote display/keypad as well. All I really needed was the entry device if the display was on the TV. Perhaps I should make a wireless keypad?

WIRELESS KEYPAD

The AVMux remote control keypad has 12 keys which can be represented with a 4-bit code. We accompany the 4 bits with a data-valid strobe signal. Making a wireless keypad is just a matter of conveying these 4 bits to the input of the AVMux controller, and Jeff's program is ready to accept this connection via the 82C55 PPI.

Regardless of the transmission medium, wireless transmission typically involves scanning the keypad for the relevant 4-bit key-pressed code, combining it with a unique identifier, and transmitting it as either an infrared or RF serial bit stream.

The Real Logic Analyzer



Test your Logic circuits with the printer port of your IBM or compatible computer!

- 5 Input capture channels via printer port
- High Speed 64K input capture buffer
- Glitch capture and display
- Full triggering on any input pattern
- Automatic time base calibration
- 4 cursors measure time and frequency
- Save, print or export waveforms (PCX)

The Real Logic Analyzer is a software package that converts an IBM or compatible computer into a fully functional logic analyzer. Up to 5 waveforms can be monitored through the standard PC parallel printer port. The user connects a circuit to the port by making a simple cable or by using our optional cable with universal test clips. The software can capture 64K samples of data at speeds of up to 1.2uS (Depending on computer). The waveforms are displayed graphically and can be viewed at several zoom levels. The triggering may be set to any combination of high, low or Don't Care values and allows for adjustable pre and post trigger viewing. An automatic calibration routine assures accurate time and frequency measurements using 4 independent cursors. A continuous display mode along with our high speed graphics drivers, provide for an "Oscilloscope-type" of real time display. An optional Buffer which plugs directly to the printer port is available for monitoring high voltage signals,

LOGIXELL
ELECTRONICS
61 Piper Cr.
Kanata, Ontario
Canada K2K2S9

Requires 286, or higher with EGA or VGA display.

LA20 Software Only \$79.95us
Software With Test Cable \$99.95us
BUFF05 Buffer \$39.95us

Tel: (613)599-7088
Fax: (613)599-7089

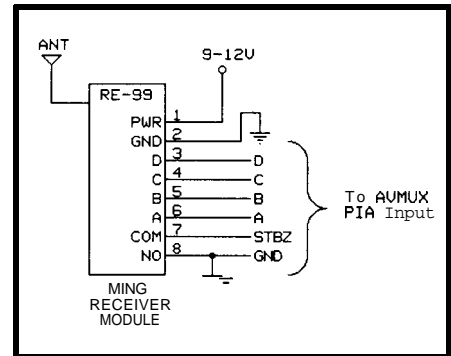


Figure 5—The Ming receiver module connects directly to some input pins on the AVMux controller.

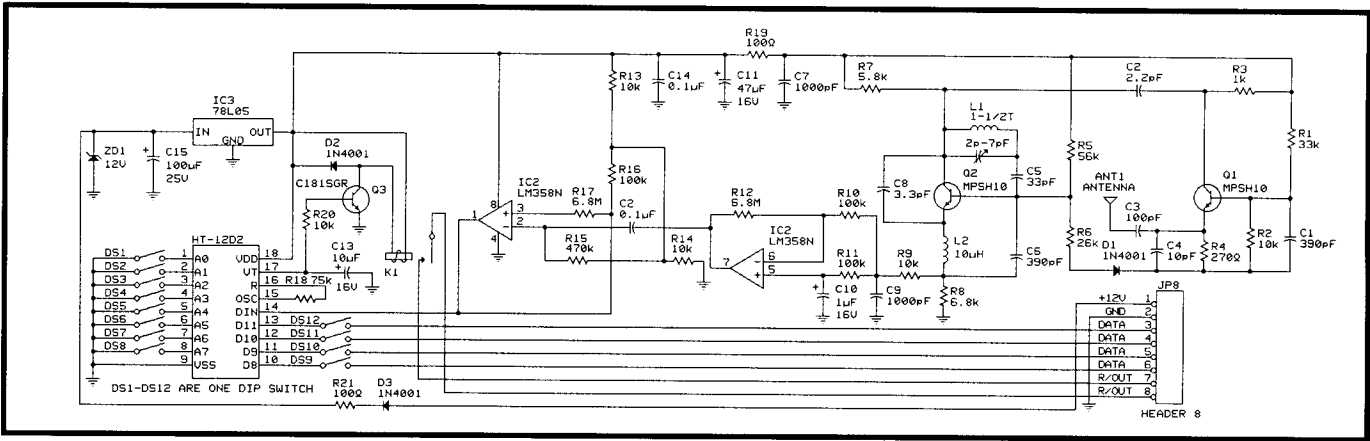


Figure 6—The Ming RE-99 receiver module uses the companion Holtek HT-12D chip. It watches the airwaves for an 8-bit address that matches its DIP switch settings and outputs the accompanying data to the AVMux.

Holtek Inc. makes a pair of chips, the HT-12E transmitter and the HT-12D receiver, which fit the bill exactly. The HT-12E/12D pair send and receive 12-bit words timed as shown in Figure 1. The first eight bits are designated as "the address" while the remaining four bits are "the data." The HT-12E transmits the word four times. When the HT-12D receives this word three times, it outputs the data bits and a data-available strobe. Figure

2 shows block diagrams of the two chips.

The two most practical wireless techniques are RF and infrared. Coincidentally, Ming Engineering and Products manufactures an inexpensive RF transmitter/receiver pair which incorporates the Holtek chips. Figure 3 is the schematic of a hand-held keypad incorporating the Ming RF transmitter. Figure 4 shows the RF section of the Ming transmitter. Figure 5 is the

schematic of the receiver connected to the AVMux controller, and Figure 6 is the Ming RF receiver circuit.

Making a wireless keypad involves some tradeoffs. The Holtek chip is designed to transmit a single 4-bit data value. It is not designed to scan a keypad like many of the newer remote control chips, so we must synthesize these functions.

As with any remote control, battery life is the primary consider-

Little Star™

NEW

30 I/O lines
Only \$195

Newest in Z-World's line of C-programmable miniature controllers, the Little Star™ has 16 protected digital inputs, 14 high-current driver outputs, RS232/RS485, battery-backed RAM and real-time clock, programmable timers, watchdog, and more. The Little Star is also available with enclosure and LCD/keypad, expansion cards for additional I/O, and optional 18 MHz clock. Our easy-to-use, yet powerful Dynamic C™ development system (\$195) integrates an editor, compiler, debugger, and dozens of software libraries all in source code. The Little Star is ideal for OEM control applications, manufacturing automation, test and data acquisition.

1724 Picasso Ave.
Davis, CA 95616
916.757.3737
916.753.5141 FAX

24-Hour AutoFAX 916.753.0618. Call from your FAX. Request catalog #18.

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

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

508-369-9556
FAX 508-369-9549

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

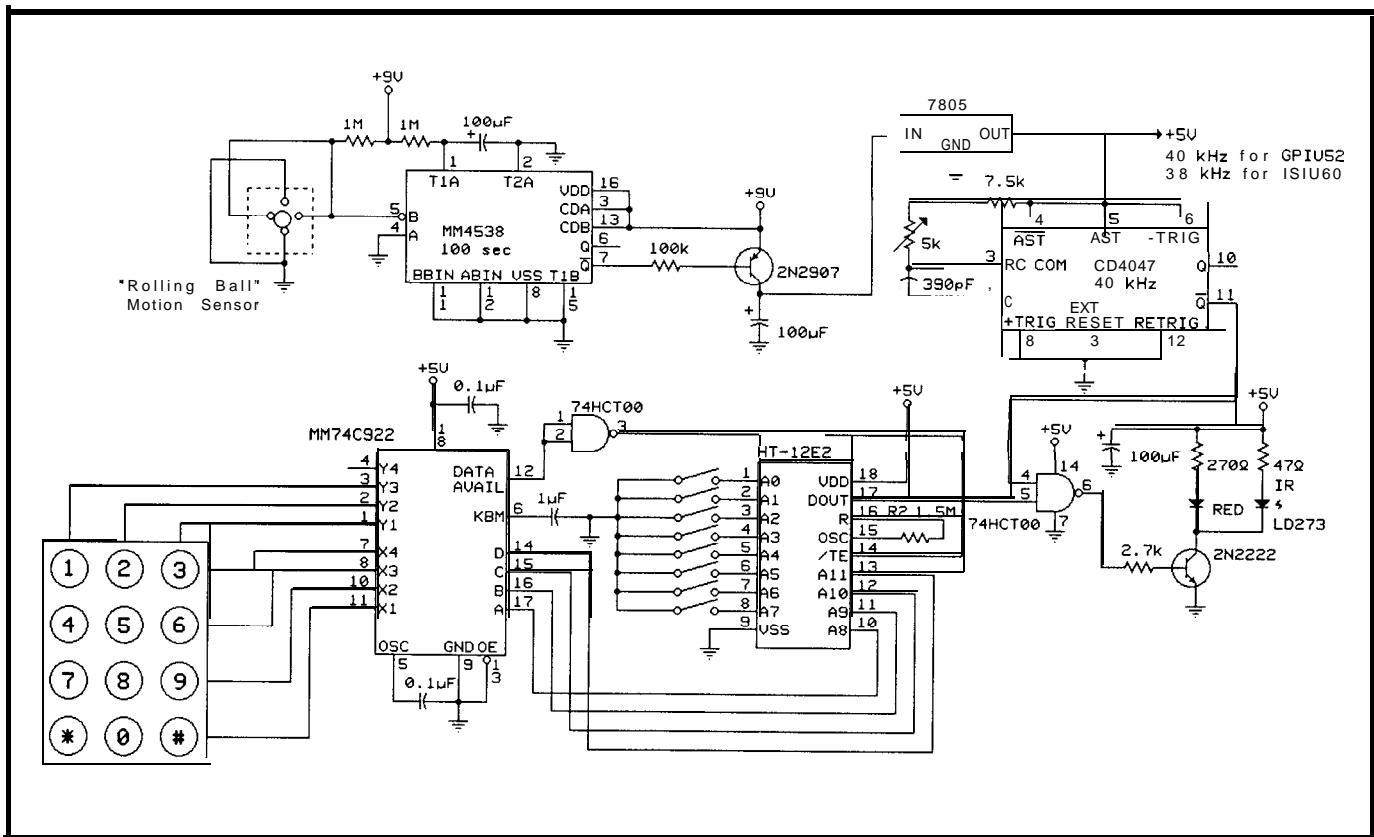
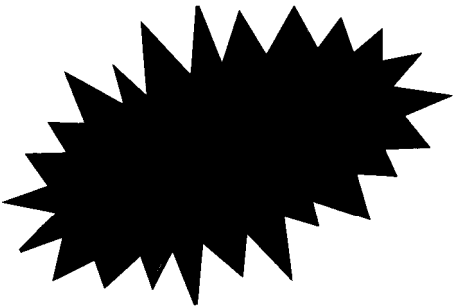
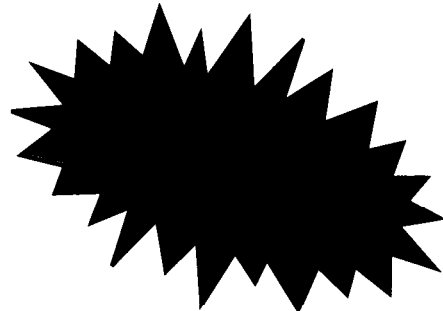


Figure 7—If you prefer IR over RF, the RF complexity can be eliminated and the same Holtek chips can be used to transmit and receive the data. A CD4047 creates the raw 40-kHz subcarrier while the HT-12E gates the signal for an IR LED.

TURBO-1 28 SINGLE BOARD BASIC DEVELOPMENT SYSTEM



The Photronics Research T-128 **SBC** features Dallas Semiconductor's new **8051-compatible DS80C320** high-speed microcontroller. With its 2X clock speed (**25 MHz**) and 3X cycle efficiency an instruction can execute in 160ns: an 8051 equivalent speed of **62.5MHz!!!**



The T-126 board **NEEDS NO EPROM EMULATOR, PROGRAMMER OR ERASER.** With our unique on-board memory management circuitry, a 70ns 128Kx8 SRAM is configured into any nonvolatile/write protectable combination of CODE, DATA, or OVERLAID mapping.

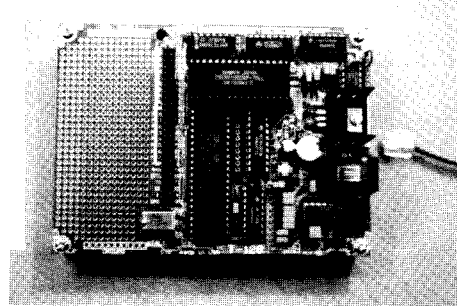
- All RAM is programmed on-board
- Dual Cell. Embedded Lithium Backup Technology
- Crashproof

NEW ON CHIP FEATURES INCLUDE:

- Programmable Watchdog Timer
- Early Warning power monitor
- Two full-duplex serial ports
- Dual data pointers
- 13 interrupt sources (6 ext., 7 int.)

ACCELERATED BASIC-52

- Modified Basic-52 now fast enough for new applications
- Stack BASIC programs in RAM and autorun
- 6 bit ports and deviceeedy I/O
- Only 5" x 3-3/4" includes 500-hole proto area
- One 4-wire telephone cable connects console/power
- The powerful combination of an accelerated Basic/Assembly Interface easily approaches 16 bit capacity
- 8-MIP "NITRO" upgrade available soon

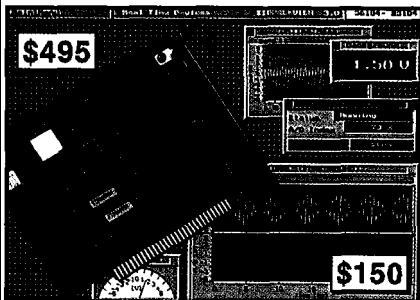


With assembler, editor, terminal emulator, utilities
1 year warranty • Free technical support
Manuals and power adapter included • Ready to Run!
INTRODUCTORY PRICE \$239

Photronics Research, Inc. • 109 Camille St. • Amita, LA 70422 • (504) 748-9911

Data Acquisition and Control Without Compromise

Programmable Scan & Burst
RTD's Advanced Industrial Control
boards set a new performance
standard for general purpose
industrial & laboratory applications.



ADA2210 with SIGNAL*VIEW
A cost-effective solution

ADA2210 features:

- | 125 kHz XT throughput
- 16 S.E. / 8 DIFF 12-bit analog inputs
- | $\pm 5, \pm 10, 0-+10V$ selectable input range
- programmable auto channel scan
- programmable burst mode
- | software & external triggers, pacer clock
- on-demand DMA transfer
- | programmable gain: 1/2/4/8 (1/10/100/1000)
- 3 cascadable 16-bit counters
- | 16 programmable digital I/O lines
- two 12-bit analog outputs, selectable range
- rtdLinX Universal TSR DOS driver
- | Labtech Notebook driver

HARDWARE SOLUTIONS

Our AIC family also includes **AT analog** I/O boards, **+5V** only A/D & control boards for portable PCs, 4-20 **mA current** loop outputs, and opto-22 compatibility.

SOFTWARE SOLUTIONS

Select the power and performance of **SIGNAL*VIEW™** or **SIGNAL*MATH™** from our library of application programs for monitoring, data acquisition and analysis, control, DSP and 3D graphics.

*For more information on these and other
ISA bus and PC/104 products,
call, write or fax us today!*

**Place your order now and receive
SIGNAL*MATH and
SIGNAL*VIEW for
\$195!**



Real Time Devices, Inc.
P.O. Box 906

State College, PA 16804
(814) 234-8087 ■ Fax: (814) 234-5218

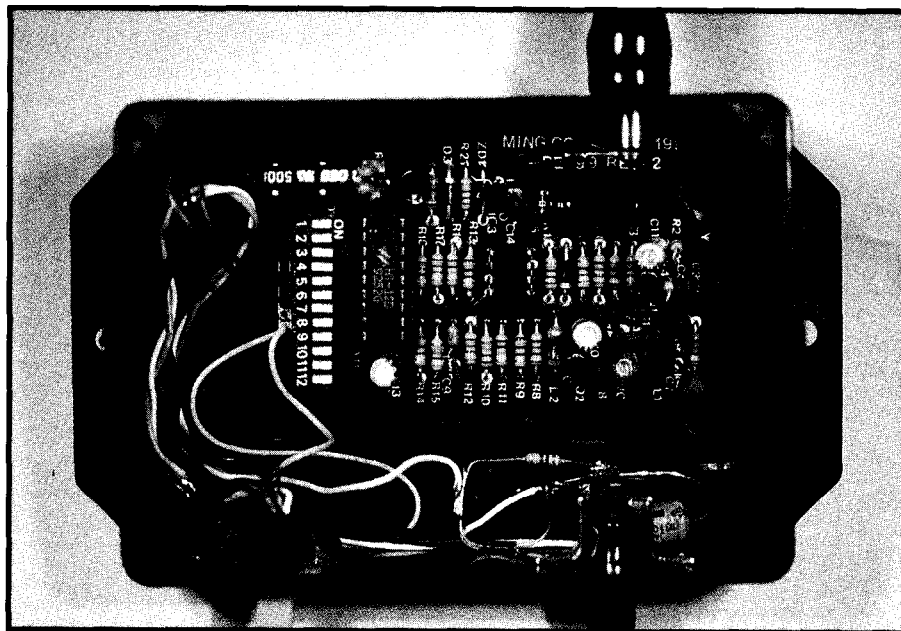


Photo 2-The RF receiver board has its own enclosure that can be mounted behind the equipment stack next to the AVMux.

ation. Both the IR and RF remotes share the same basic power control and scanning circuit. In the normal inactive state, only the CD4538 one-shot is powered. In the unit incorporating the Ming TX-99, this power is 12 volts. The power consumption is approximately 1 μA .

A "Rolling Ball Motion Sensor," typically used in toys, fires the one-shot when the remote keypad is picked up. Its duration is 100 seconds and is

retriggered as long as it is moved. Once triggered, power is applied to a 74C922, which is a 16-key keypad scanner. If any key is pressed during the one-shot period, that key code is sent to the 4-bit transmitter input. Simultaneously, the "data available" output from the 74C922 turns on the power to the TX-99 and the key code is sent. The only stipulation is that the key be pressed long enough to transmit the 4-word packet, about 50 ms.

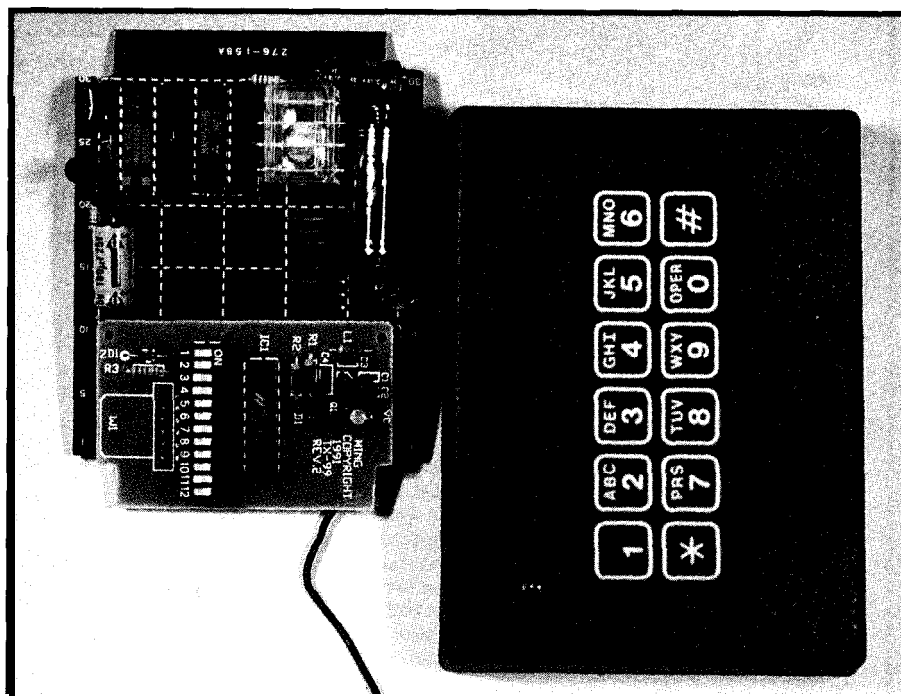


Photo 3-The hand-held RF transmitter unit uses a "rolling ball" motion detector with some power control circuitry to extend battery life.

The IR transmitter (Figure 7) works essentially the same except that it is powered from a 9-V battery and has to have the Holtek HT-12E as part of the circuit. The circuit also includes the 8-bit address selection switches which are already on the Ming transmitter. Also, like the RF approach, we have a modulated transmission. Here, the serial output from the HT-12E modulates a 40-kHz (or 38-kHz) clock before it is applied to the IR LED.

The RF receiver is simplicity itself. The RE-99 receiver has all the necessary attributes. Once properly addressed, any transmitted keycode will appear on the 4-bit output simultaneously with a relay contact closure which can be wired as a data available strobe. The IR receiver (Figure 8) is equally unembellished and consists only of the Holtek chip and an integrated IR receiver module.

The choice of whether to use IR or RF will depend upon a variety of things including local interference. The Ming units operate at about 300 MHz and their range can be as much as 100 feet (the IR unit is good for about 20-30 feet). This range can be increased by adding antennas to the transmitter, receiver, or both. A frequency of 300 MHz translates to a wavelength of 1 meter. A 1/4 wavelength antenna would, therefore, be 1/4 meter or 9.8 inches.

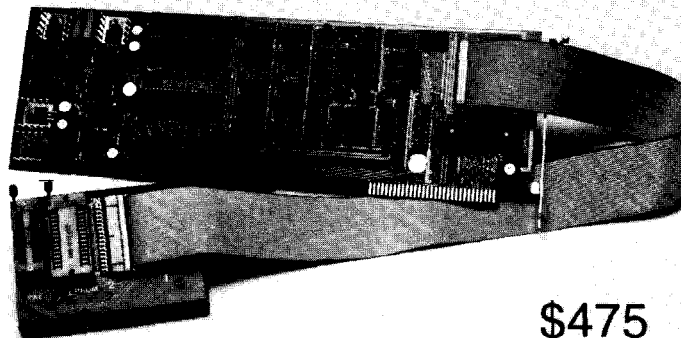
If you live in the wide open spaces, you probably won't have any problems. If you live in a 5000-person high-rise in the middle of a city, you'll find lots of 300 MHz floating around. The problem might not be receiving false keys but rather none at all if most transmissions get trampled in the noise. Experiment and see.

IN CONCLUSION

Well, I've got my new AVMuX up and running and I don't have to leave the chair except to go to the refrigerator. Now that this is done, I can get back to wiring more things to run under HCS control. Of course, since this little wonder is directly compatible, perhaps I should carefully document the connection interfacing. Who knows? It might even make a good article. □

Universal Device Programmer

PAL
GAL
EPROM
EEPROM
FLASH
MICRO
87C51
PIC
93C46
XC1 736
PSD 3xx
5ns PALs

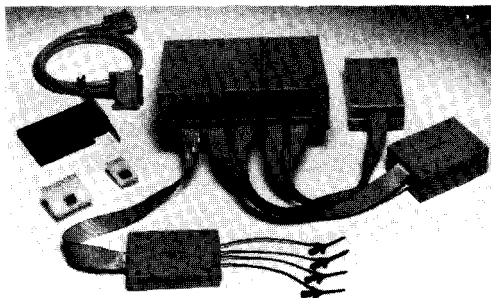


\$475

Free software updates on BBS
Powerful menu driven software

400 MHz Logic Analyzer

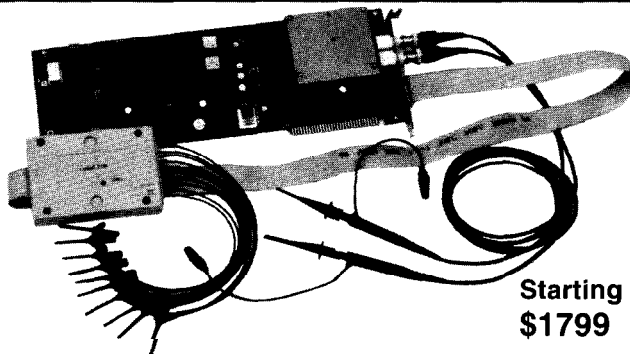
- up to 128 Channels
- up to 400 MHz
- up to 16K Samples/Channel
- Variable Threshold Levels
- 8 External Clocks
- 16 Level Triggering
- Pattern Generator Option



\$799 - LA12100 (100 MHz, 24 Ch)
\$1299 - LA32200 (200 MHz, 32 Ch)
\$1899 - LA32400 (400 MHz, 32 Ch)
\$2750 - LA64400 (400 MHz, 64 Ch)

Price is Complete
Pods and Software
included

200 MSa/s DIGITAL OSCILLOSCOPE



Starting at
\$1799 With Pods
& Software

- 200 MSa/s sampling Rate
- 2 Analog Channels (2 ch. Digital Oscilloscope)
- 8 Digital Channels (8 ch. Logic Analyzer)
- 125 MHz Single Shot Bandwidth
- up to 128K Samples/Channel

Call (201) 808-8990



Link Instruments

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

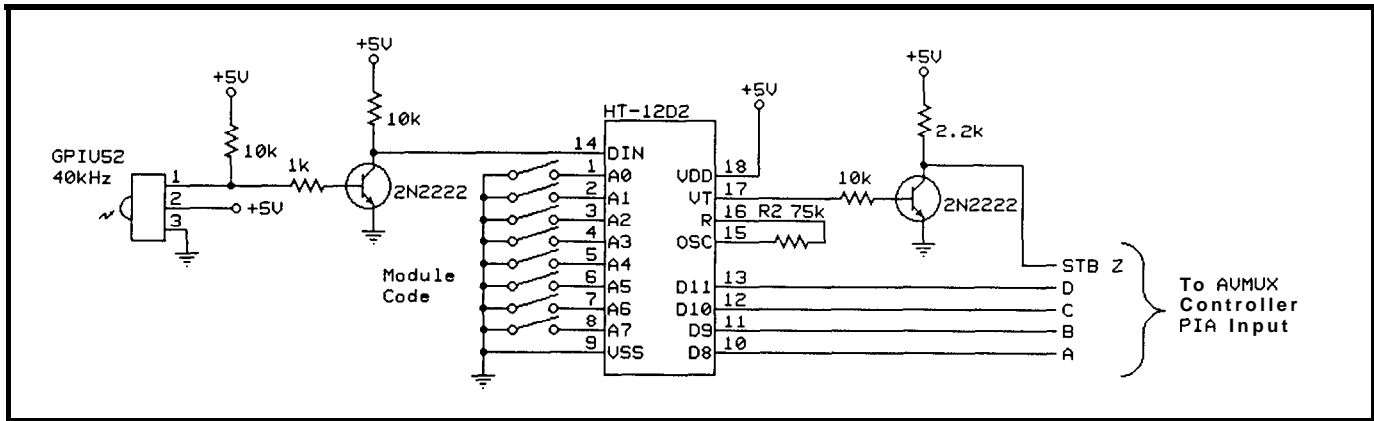


Figure 8—The Sharp GPIU52 (40 kHz) and IS1U60 (38 kHz) IR receiver modules put all the touchy IR receiver circuitry inside a single device, making the receiver unit a piece of cake.

Steve Ciarcia is an electronics engineer and computer consultant with experience in process control, digital design, and product development. He may be reached at steve.ciarcia@circellar.com.

SOURCES

Schematics of the TX-99 and RE-99 boards are reprinted by permission of

Ming Engineering and products Inc.
17921 Rowland St.
City of Industry, CA 91748
(818) 912-9469

The Holtek encoder/decoder ICs and Data Manuals are available from

Digi-Key Corp.
701 Brooks Ave. S., P.O. Box 677
Thief River Falls, MN 56701-0677
(800) 344-4539

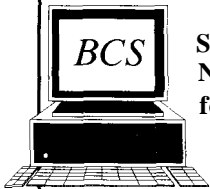
The Maxim and Analog Devices chips used in the AVMUX are available from

Pure Unobtanium
13109 Old Creedmoor Rd.
Raleigh, NC 27613
Phone or fax (919) 676-4525

IRS

- 410 Very Useful
- 411 Moderately Useful
- 412 Not Useful

EARN \$5000 PER MONTH With your home computer!



Spending too much on your computer? Now it's time to let it earn good money for you! Our two successful Software Packages, MoneyMaker Volume-1 and MoneyMaker Volume-2 will give you all the insider information you need to start and succeed with your own PROFITABLE BUSINESS Part-Time at home!

Your cost for all this extra income?

Only \$29.95 for MoneyMaker Volume-1

For BOTH VOLUMES NOW send just \$39.95 !!!!
AND GET NEW 300 PAGE MANUAL FOR 1994

Includes FREE Talking Demo Disk and
3 BIG Catalogs full of more Money Making ideas!

* We always pay postage

Please specify disk size or we ship 3.5"

- Check, Credit Card, or M.O. to:

BUSINESS COMPUTER SYSTEMS, INC.

46-16 65TH PLACE -Dept M2

WOODSIDE, N.Y. 11377

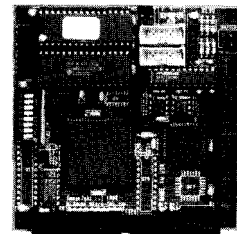
AMEX, VISA, MC ORDERS
FAX (718) 898-3126
PHONE (718) 803-3638

Quality Software for the PC since 1986

We're Small, We're Powerful, And We're Cheaper.

MMT-188 EB

- 2 serial I/O ports
- 3 programmable parallel I/O ports
- 1 Meg RAM/ROM capable
- power fail detect interrupt and reset
- counter-timers
- watch dog timer
- expansion connector



Only \$191⁰⁰ (Qty 100)

ALSO AVAILABLE: MMT-Z180, MMT-196, MMT-HC11, MMT-EXP

In fact, you'll get the best product for about half the price. If you're interested in getting the most out of your project, put the most into it. For the least amount of money.

Call us today for complete data sheets, CPU options, prices and availability.

Custom Work

Welcome. Call or fax for complete data sheets

2308 East Sixth Street

Brookings, SD 57006

Phone (605) 697-8521

Fax (605) 6974109



WE'RE SMALL BUT WE'RE POWERFUL!

DEPARTMENTS

46 Firmware Furnace

60 From the Bench

68 Silicon Update

74 Embedded Techniques

84 ConnectTime

FIRMWARE FURNACE

Ed Nisley

All Text is Graphics: Characters for the '386SX Project's Graphic LCD Panel



A graphic panel that displays lots of

dots is fine until you need some text. It turns out the raw resources are already in your PC, put in place to support the old CGA. Harness those resources to produce text on your LCD display.



Have you noticed how much DOS text-mode programs now resemble Windows apps? Otherwise sensible companies spend precious programming time cross-dressing their programs...as though 2000 characters allowed enough screen real estate for fancy borders and pushbuttons. Ah, well, such is progress.

This month is, perhaps, a step in the same direction. We're now going to turn a perfectly serviceable bit-mapped graphics display into a moderately intelligent Glass Teletype with ANSI cursor positioning and attribute controls. This will give us a way to display status information without using a serial port or the video screen; it'll come in handy in a few months.

I'll start by showing where the character bitmaps come from, then cover the assembler code that shuffles them into the LCD Display RAM. The TTY routines that decode the standard ASCII control characters to determine the location of the next output character come next and I'll finish with the top level: the ANSI controls.

FILCHING FOSSIL FONTS

The Original IBM PC had two display options: the text-only Mono-chrome Display Adapter and the text/graphics Color Graphics Adapter. You could have both cards installed in the same system with the MDA showing quite high-quality text on one screen and the CGA displaying moderately bad graphics on another.

The catch was that in graphics mode, the CGA hardware didn't support text output. The CPU had to draw every character, dot by dot, whenever a program called for text. Even though those functions really belonged with the CGA hardware, the PC's designers included them in the System BIOS. The architecture for installable BIOS extensions came somewhat later when the IBM PC/XT hit the scene.

Now, nearly 15 years later, the PC Compatibility Barnacles dictate that every PC clone's BIOS must include those CGA functions even though the standard video card is a Super VGA with its own BIOS extensions. You may have the latest SuperTurbo 50 Million WinMark video accelerator, but the system board BIOS remains ready for the day when you return to that good old CGA!

The CGA presented 640x200 dots in one color on a background of another color, so anything it could do maps directly to our 200-line LCD panels. In particular, the BIOS has 8x8-dot character bitmaps which, while

they may not be the nicest looking characters in the world, lie waiting for a little firmware to put them to good use.

Those bitmaps were located at address F000:FA6E in the Original IBM PC and the Compatibility Barnacles guarantee that address remains correct for every PC clone ever built. The bitmap for a particular character starts at:

$$F000:FA6E + 8 * (\text{character value})$$

and extends for the next eight bytes. You can see how the process works in Listing 1, which copies a single character from the bitmap into the LCD Refresh Buffer.

The BIOS bitmaps include only the first 128 characters (0x00 through 0x7F), so LCDWriteGlyph substitutes a question mark if the character isn't in the table. It then iterates through the bitmap's rows, copying them one byte at a time. In this case, FontHeight and FontWidth are both eight dots.

If you need large, small, or very detailed characters, you must modify this routine to copy the appropriate bits from your custom bitmap table to the Refresh Buffer. This will also require changes in the LCDWriteByte routine so that you can write partial bytes if your glyphs are not a multiple of eight bits wide.

The BIOS ROMs are not the only source of bitmap characters in a stock

PC: any graphics card other than the CGA includes a BIOS with the fonts appropriate for it ready for our use. The EGA and VGA standardized a method for locating these bitmaps, so if your embedded PC has such a card installed in it, you have a wider choice of characters. Because the LCD code is entirely independent of the built-in graphics code, you can use any of the bitmaps without regard to the current video graphics or text mode.

The bad news is that there's no tidy way to decide if the system has no video card, an MDA, a CGA, or something more useful. The code shown in Listing 2 checks the Int 1F vector, which points to the bitmap for character 0x80 if the font can support more than 128 characters. That vector should be zero with no card or a CGA and nonzero with an EGA or VGA. The vector does not point to an interrupt handler, although some BIOSs may aim it at an unexpected interrupt handler during the power-on self tests.

If the Int 1F vector is nonzero, the code issues Int 10, function 11, subfunction 30 to get the address of one of the card's font bitmaps. Because VGA cards include both 8x8 and 8x16 fonts, I use the latter for 400-line LCD panels just to show that, if you have more dots, the characters need not look quite so ugly. The code then sets several variables detailing the font's height, width (which is always 8 bits), and the number of characters in the bitmap.

This should work with most VGA video cards, but I won't be surprised to find that it fails in some systems. In particular, EGA cards don't include the 8x16 font table, so the results will look bizarre. I much misdoubt any of you have an EGA, but if so you must tweak the code if you also have a 400-line panel. Uncomment the stub of code to dump the register values returned from the Int 10 call to see what went wrong if strange things happen.

Incidentally, I had to use the int 8 function rather than the more familiar int 86x because the BIOS function returns the font pointer in ES:BP.

Listing 1—This C function writes a character bitmap (a.k.a., the character's glyph) into the LCD Refresh Buffer. Because the assembly language routines handle all the panel-specific bit shuffling, this routine is the same for all the LCD panels. The font width is always eight bits to keep things simple.

```
void LCDWriteGlyph(int CharCode, int DotRow, int DotCol){
int ScanLine;

if ((CharCode < 0) || (CharCode >= NumFontChars)){
CharCode = '?'; /* an obvious ASCII filler character */

CharCode *= FontHeight; /* point to start of char */

for (ScanLine=0; ScanLine < FontHeight; ++ScanLine){
LCDWriteByte(*(pFont+CharCode+ScanLine),
DotRow+ScanLine, DotCol, BlinkMode);
```

Check your compiler doc, but as near as I can tell, `in tr` is the only way to get BP without resorting to in-line assembler.

You can short-circuit the entire font search routine with your own code to install a completely custom font. Keep in mind that the rest of the code assumes that the bitmaps are eight bits wide, but the sky is the limit in the other direction.

Remember that the LCD Refresh Buffer contains the bitmap patterns, not the ASCII character values. The CGA included a readback function that searched the font bitmap table to find the pattern that matched the one at the cursor location, but I didn't attempt that. If you must read characters back from the LCD Refresh Buffer, I suggest allocating a 2000 (or 4000, or whatever) byte buffer and storing the character codes "in parallel" with the bitmaps. Trust me, it's a whole lot easier!

Now that we have a source of character bitmaps, the next step is to put them into the Refresh Buffer. The

CGA had a simple bit layout, but most LCD panels are not nearly so accommodating.

DIVVYING UP THE DOTS

Listing 3 shows `LCDWriteByte`, which writes a single byte into the DMF651 Refresh Buffer. The character coordinates are given in terms of dots, but the column must be a multiple of eight to position the character correctly. The row may be any value, but, as you'll see later, I use only multiples of the font height to produce a regular spacing.

The Graphic LCD Interface supports blinking on the DMF651, so the incoming byte is split in half and stored in the low nybbles of two successive buffer addresses. The bit layout allows us to write four bits at a time, which means we don't need the direct dot plotting code from last month.

The panel displays bits 4:7 alternately with bits 0:3 at the blink rate. The high and low nybbles in each byte must be identical to suppress

visible blinking. If the high nybble is zero, the dots in the low nybble appear to blink on and off against a blank background. You can implement other blinking schemes if you like, as well as other effects such as inverse characters.

If you need nonaligned or proportionally spaced characters on your LCD panel, you must handle incoming bits that don't fit so neatly. In fact, a single bitmap byte may be split among three different Refresh Buffer bytes. The code gets particularly tricky at the right edge of the panel where a character may fall off the end of the line.

Aligned characters simplify the routine that calculates the Refresh Buffer address. Listing 4 shows the few lines of `LCDMakeCharAddr` for a DMF651 panel. The dot-drawing `LCDMakeAddr` routine I covered last month sets `CL` to the amount needed to move the dot from the low-order bit, but here we always set the entire nybble, so no further shifting is necessary.

Adapting this code to other panels is reasonably straightforward. For example, the 640x400 LG64AA44D panel accepts eight bits on each Dot Clock, so blinking isn't supported. Bits 0:3 of each byte appear on rows 0 through 199, while bits 4:7 fill rows 200 through 399. The trick is to insert the new data without disturbing the existing bits in the "other" nybble of the byte.

Listing 5 shows the `LG64AA44D` `LCDWriteByte` function. In this case, `LCDMakeCharAddr` sets `CL` to align the new nybble in either the low or high half of the Refresh Buffer byte depending on which row it's in. The code reads the existing byte from the buffer, strips out the old bits, inserts the new ones, and finally writes the modified byte back into the buffer. Although it's slightly slower than the DMF651 code, you won't notice the difference in real life.

So much for a single character. Now for the useful stuff!

GAZING IN THE GLASS

First of all, a confession: I have not implemented a real blinking character

AVOCET SYSTEMS, INC.

C and Pascal Compilers

Simulators

Assemblers

In-Circuit Emulators

EPROM Programmings

Intel
8051
8048
8085
8096

Hitachi
64180
H8

Western
65816

Rockwell
6500
6502

RCA
1802

Motorola
680x0
68300
68HC11
6800
6801
6802
6803
6804
6805
6809

Zilog
Z8
Z80
Z180

TI
32010
32020

Source For Quality Embedded Systems

100 Main Street, P.O. Box 490, Rockport, ME 04856
For outside U.S., call (207) 236-9855
In Maine call (207) 236-6713

Call today:
1-800-448-8500

Listing 2—This routine sets up the font bit map for the rest of the code. If the system has no video card, the `Int1FVector` should be 0000:0000 to indicate that the default 128-character CGA font is in effect. That vector is nonzero if an EGA or VGA card is installed, in which case `Int 10`, function 11, subfunction 30 returns a pointer to the desired font table. I used the 8x8 font for 200-line panels and the 8x16 font for larger panels to illustrate the fact that more dots can produce a less-ugly font. If you prefer more information to prettier characters, use the 8x8 font regardless of the panel size.

```
void LCDInitFont(void){
struct REGPACK rp;
void far *Int1FVector;
int VidMode;

/*- figure out what character table is available */

puts("Initializing character generator...");

Int1FVector = (void far *)*(long far *)MK_FP(0x0000,4*0x1F);
printf(" Int 1F vector is %Fp\n",Int1FVector);
if ((void far *)NULL == Int1FVector){ /* have chars 80..FF? */
puts(" Using ROM BIOS CGA font");
pFont = (BYTE far *)MK_FP(0xF000,0xFA6E);/* no, use BIOS ROM */
FontHeight = 8; /* which is 8x8 only */
FontWidth = 8;
NumFontChars = 128; /* basic ASCII only! */
}
else {
puts(" Using video card font");
VidMode = *(BYTE far *)MK_FP(0x0040,0x0049);
printf(" . . . resetting video mode: %u\n",VidMode);

rp.r_ax = (0x00 << 8) | VidMode; /* Reset video mode */
intr(0x10,&rp);

rp.r_ax = 0x1130; /* Get Font Pointer info */
if (NumDotRows > 200){
rp.r_bx = 0x0600; /* BH = 06 for 8x16 font */
FontHeight = 16;
}
else {
rp.r_bx = 0x0300; /* BH = 03 for 8x8 font */
FontHeight = 8;
}
printf(" . . fetching %u line font pointer\n",FontHeight);
intr(0x10,&rp);

#if 0
printf("ax %04X bx %04X cx %04X dx %04X ds %04X es %04X bp
%04X\n",rp.r_ax,rp.r_bx,rp.r_cx,rp.r_dx,
rp.r_ds,rp.r_es,rp.r_bp);
#endif

pFont = (BYTE far *)MK_FP(rp.r_es,rp.r_bp);
FontWidth = 8;
NumFontChars = 256;
}

printf(" pFont=%Fp, FontHeight=%u, NumFontChars=%u\n",
pFont,FontHeight,NumFontChars);

/*- set up screen control variables */

NumCharRows = NumDotRows / FontHeight;
NumCharCols = NumDotCols / FontWidth;
VScrollAllowed = 1;
printf(" Panel has %u character rows and %u character cols\n",
NumCharRows,NumCharCols);

return;
}
```

cursor for the LCD panel because I expect to use it as an output-only status device. Nevertheless, we need a name for the spot where the next character will appear and "current cursor location" is as good as any I can think of. If you need a visible cursor at the current cursor location, well, it's a simple matter of firmware.. .load a blinking block and be done with it!

The ASCII character set includes about 32 control characters that are not ordinarily displayed. Instead, they affect how the remaining 96 characters appear on paper, which was the current rage when they were defined. The control characters include old friends Carriage Return and Line Feed, as well as obscure relatives like End of Medium.

Which control character functions you implement and exactly what they do depends on how thorough you are. Nobody (well, hardly anybody) uses paper output, so the mappings tend to be somewhat arbitrary. The term "Glass Teletype" pretty well describes the level of control you can achieve: no matter how hard you try, the results still look a lot like a roll of paper.

Listing 6 shows the bare-bones implementation I chose for the LCD panel code. Line Feed and Carriage Return do pretty much what you'd expect, Form Feed clears the panel, Back Space moves the cursor one space to the left, Tab moves it to the next multiple of four, and Delete erases the character at the cursor position. Everything else is treated as a displayable character and passed to the LCD panel.

The `CurDotRow` and `CurDotCol` variables contain the current cursor location in terms of the dot coordinate of the upper-left corner. Although I force the characters to use a grid the size of the font bitmap, you can tweak the code to handle multiple character sizes and other effects.

The code supports vertical scrolling, so the panel will always show the last 25 (or 50; or whatever) lines. I won't show the code here, but, contrary to what you might think, you can't just use a big rep `MOVS b` instruc-

What is the 80C552

R's a high integration, 8051 with:
8 ch. 10 bit A/D **2 PWM outputs**
Cap/cmp registers **16 I/O lines**
RS-232 port **Watchdog**
 W'e've made the **552SBC** by adding:
2-RS-232/485 multi-drop ports
24 more I/O **Real-time Clock**
EEPROM **3-RAM & 1-ROM**
Battery Backup Power Regulation
Power Fail Int. **Expansion Bus**
 Start with the Development Board - all the peripherals, power supply, manual and a debug monitor for only \$949. Download your code and debug it right on this SBC. Then use the \$149 and up OEM boards for production. or have us make a custom board for you. Call now for a brochure!

188SBC

Use Turbo or MS 'C'
Intel 80C188XL
Two 1 meg Flash/ ROM sockets
Four battery backed, 1 meg RAM
16 channel, 12 or 16 bit A/D
8 channel, 12 bit D/A
2 RS-232/485 serial, 1 parallel
24 bits of opto rack compatible I/O
20 bits of digital I/O
Real-time clock
Interrupt and DMA controller
8 bit, PC/104 expansion ISA bus
 Power on the quiet, 4 layer board is provided by a switcher with watchdog and power fail interrupt circuitry.

It's got "EIEIO"

The 188SBC is also available with Extended Interface Emulation of I/O - a Xilinx Field Programmable Gate Array and a breadboard area. You can now define and design nearly any extra Interface you need. 188SBC prices start at \$299. Call right now for a brochure!

\$149 8032 ICE
Still Available!

8031 SBC as low as **\$49**

Call for your custom product needs. Quick Response.



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

Since 1983

(619) 566-1892
 70662.1241 @compuserve.com

Listing 3—The fundamental character operation is writing a single byte into the LCD Refresh Buffer. This DMF651 routine computes the buffer address, splits the byte into two nybbles, and sets the blinking bits. The code is considerably simplified by aligning characters on d-dot column boundaries!

```

PUBLIC C LCDWriteByte
PROC    LCDWriteByte
ARG     CharCode: WORD, DotRow: WORD, DotCol: WORD, Blinking: WORD
USES   ES,DI

MOV     AX,[DotCol]      ; set up column number
AND     AX,NOT 0007h     ; . . . force byte alignment
MOV     BX,[DotRow]     ; set up row number
CALL    LCDMakeCharAddr

LES     AX,[pLCDBuff]   ; get buffer base pointer
ADD     DI,AX           ; . . . update byte pointer

;-- insert the left nybble and blinking pattern

MOV     AX,[CharCode]   ; fetch new byte value
MOV     DL,AL           ; save high bits for later
SHR     AL,4            ; align nybble, clear blink bits

CMP     [Blinking],1
JE      SHORT @@1
MOV     AH,DL           ; no blinking, copy data bits
AND     AH,0F0h
OR      AL,AH

@@1:    MOV     [ES:DI],AL ; write blinkng + data to buffer

;-- insert the right nybble and blinking pattern

INC     DI              ; step to next buffer byte
AND     DL,00Fh        ; extract nybble, clear blink bits

CMP     [Blinking],1
JE      SHORT @@2
MOV     BH,DL          ; no blinking, copy data bits
SHL     BH,4
OR      DL,BH

@@2:    MOV     [ES:DI],DL ; write blinkng + data to buffer

RET
ENDP    LCDWriteByte
  
```

Listing 4—This routine for a DMF651 panel computes the LCD Refresh Buffer address corresponding to a character's location given as a dot column in AX and a dot row in BX. The characters are always aligned on an 8x8-dot grid, so there is no need to compute a shift amount

```

PROC    LCDMakeCharAddr

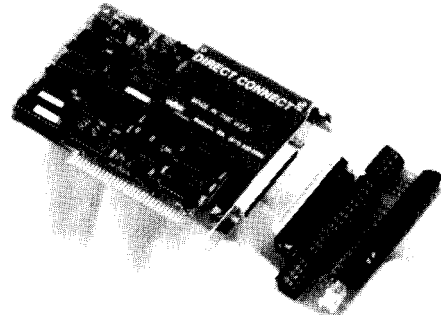
ADD     BX,BX           ; make word table index
MOV     DI,[RowStarts+BX]

DIV     [BYTE ColModulus] ; col / modulus
MOV     CL,0           ; force shift amount to zero
MOV     AH,0
ADD     DI,AX          ; DI points to the byte

RET

ENDP    LCDMakeCharAddr
  
```

16 BIT DATA ACQUISITION?



*Don't Settle For Less
Than The Cutting Edge...*

HIGH PERFORMANCE

Guaranteed 16 bit accuracy
8 Channel A/D Board CI 295

- 16 bit A/D resolution
- 16 bit accuracy
- 50 kHz throughput
- DMA
- 8 lines digital I/O
- 3 channel counter/timer

LOW COST

16 bit 8/16 Channel
A/D Board \$895

- 16 bit A/D resolution
- 15 kHz throughput
- DMA
- 8 lines digital I/O
- 3 channel counter/timer

OPTIMUM CONVERSION™

DT2801/5716 compatible
A/D Board \$1395

- 16 bit A/D resolution
- 16 bit accuracy
- DMA, Prog. Gain
- 16 lines digital I/O
- 3 channel counter/timer
- 2 D/A channels

Cut through the specs - each of ADAC's 16 bit boards have been evaluated against every competing model. On noise performance, speed, ease of use, and price, ADAC's leading technology wins every time.

See for yourself - Call for an evaluation board today.

1-800-648-6589

We've been making data acquisition boards for longer than anyone in the world.



70 Tower Office Park, Woburn, MA 01801
FAX (617) 938-6553 TEL (617) 93.56668

Analog & Digital I/O, Industrial PCs,
and High Channel Count Systems

Listing 5—The LG64AA44D LCD Refresh Buffer layout is more complex than the DMF651. Blinking isn't supported because the 400-row panel accepts and displays eight bits on each Dot Clock. This routine splits a byte into two nybbles and inserts them into the appropriate parts of the two target bytes. As with the DMF651, the character's column address must be a multiple of 8.

```

PUBLIC C LCDWriteByte
PROC   LCDWriteByte
ARG   CharCode: WORD, DotRow: WORD, DotCol: WORD, Blinking: WORD
USES  ES,DI

MOV   AX,[DotCol]           ;set up column number
AND   AX,NOT 0007h         ;... force byte alignment
MOV   BX,[DotRow]          ;set up row number
CALL  LCDMakeCharAddr

LES   AX,[pLCDBuff]        ;get buffer base pointer
ADD   DI,AX                 ; . update byte pointer

;- insert the left nybble

MOV   AX,[CharCode]        ; fetch new byte value
MOV   DL,AL                 ; save high bits for later
SHR   AL,4                  ; set up for alignment
SHL   AL,CL                 ; align to target nybble
MOV   BH,0F0h              ; set up bit mask
ROL   BH,CL                 ; .. align to strip old bits

MOV   AH,[ES:DI]           ; fetch target bits
AND   AH,BH                 ; strip old bits
OR    AL,AH                 ; tuck in new bits
MOV   [ES:DI],AL           ; put 'em back in the buffer

;- insert the right nybble
; this uses the same nybble mask because it's in the same
; screen half

INC   DI                    ; step to next buffer byte
AND   DL,00Fh               ; extract low nybble
SHL   DL,CL                 ; align to target nybble

AND   BH,[ES:DI]           ; fetch target bits, strip old
OR    DL,BH                 ; tuck in new bits
MOV   [ES:DI],DL           ; put 'em back in the buffer

RET

ENDP   LCDWriteByte

```

Listing 6—Emulating a "glass Teletype" is a matter of picking off control characters that affect either the cursor location or the LCD Refresh Buffer contents. This code handles a basic set of controls that suffice for simple text output applications. The VScrollAllowed variable defaults to TRUE, so a linefeed at the bottom of the panel scrolls the panel contents up by one character row.

```

void LCDSendChar(int CharCode){
    switch (CharCode){
    case LF:
        if (CurDotRow >= (NumDotRows FontHeight)){/* on last line? */
            if (VScrollAllowed){
                LCDScrollUp(FontHeight);/* yes, scroll up a line */
                CurDotRow = NumDotRows
                FontHeight;          /* force to last line */
            }
        }
    }
}

```

(continued)

tion to schlep the bits around. Consider sorting out the dots on the LG64AA44D, where the same byte can hold both the source and the target dots!

The Del character requires a routine to delete a rectangular block of dots. In this case, you must copy a group of dots on a single row, then repeat that for each row in the character while filling the right end of the line with blanks. I did not include a corresponding Insert function, but you can add it fairly easily if your application needs it.

Because I expect most of the output to come from C strings, there is also an LCDSendString function defined to handle standard zero-terminated strings. You can embed control characters in the strings or send them directly to LCDSendChar as needed.

The code also includes several utility functions to enable and disable vertical scrolling, wrapping at the end of each line, and so forth. The source code is available on the BBS if you're

Listing 6—continued

```

else {
    CurDotRow = 0;          /* no scroll, wrap to top */

else {
    CurDotRow += FontHeight; /* no, so step down */
}
break;

case CR :
    CurDotCol = 0          /* to leftmost col */
    break;

case FORMFEED :
    LCDClear();
    break;

case BS :
    if (CurDotCol >= FontWidth) /* if not at edge */
        CurDotCol -= FontWidth; /* back up one column */

    break;

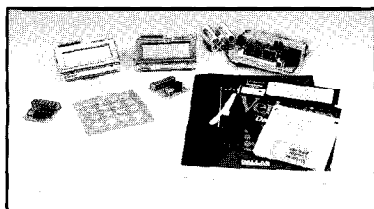
case TAB:
    /* tab to next stop */
    do {
        LCDSendChar(' ');
    } while (((CurDotCol/FontWidth)+1) % TABINTERVAL);
    break;

case DEL:

```

(continued)

Batteries Included



Mid-Tech's ec.25 offers the ultimate in flexibility and extensibility for your specialized embedded computing needs. Specifically designed for battery operation, this small (4" by 4" by 1") computer is capable of extremely long run times due to its built-in power management capability.

The included PC hosted IPL utility allows serially downloading executable programs. A number of ready-to-run applications are included on diskette along with source code for all peripheral drivers and a number of useful utilities written in both assembler and C.

The ec.25 accommodates an unparalleled complement of I/O options for a variety of data collection, monitoring, and control tasks.

- DS2250 Soft Processor (8031 compatible)
- 64K Lithium-backed RAM
- Bootstrap Loader with PC-based IPL Utility
- RS-232 and CMOS RS-485 Ports
- 12C LCD/Keypad Port
- RTC/Timer with 256 bytes RAM
- 512 Bytes E²PROM
- 8-channel 8-bit ADC
- 2-channel 8-bit DAC
- 16-bit Bidirectional DIO
- * Dual Switch-Mode/Pass-Mode Voltage Regulators
- * Full Featured Battery Manager/ Fast Charger
- 120 Binary I/O Points over Twisted Pair
- NiCd Battery and Line Power Options Included

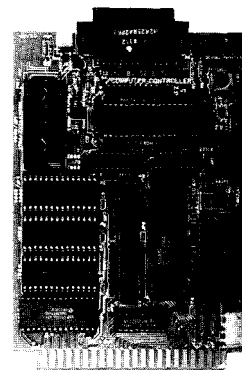
The full-up ec.25DK developers kit is \$495.00 (single quantity). Call for OEM pricing and availability of special configurations.

Mid-Tech Computing Devices
P.O. Box 218, Stafford, CT 06075-0218
Voice/fax: (203) 684-2442

BCC52 BASIC-52 COMPUTER/CONTROLLER

The BCC52 controller continues to be Micromint's best selling single-board computer. Its cost-effective architecture needs only a power supply and terminal to become a complete development system or single-board solution in an end-use system. The BCC52 is programmable in BASIC-52, (a fast, full floating point interpreted BASIC), or assembly language.

The BCC52 contains five RAM/ROM sockets, an "intelligent" 2764/128 EPROM programmer, three 8-bit parallel ports, an auto-baud rate detect serial console port, a serial printer port, and much more.



PROCESSOR

- 80C528-bit CMOS processor w/BASIC-52
- * Three 16-bit counter/timers
- * Six interrupts
- * Much more!

Input/Output

- Console RS232 - autobaud detect
- Line printer RS-232
- * Three S-bit parallel ports
- EXPANDABLE!
- Compatible with 12 BCC expansion boards

MEMORY

- 48K RAM/ROM, expandable
- Five on-board memory sockets
- Either 8K or 16K EPROM

To Order Call 1-800-635-3355
Tel: (203) 871-6170
Fax: (203) 872-2204

BCC52	Controller board with BASK-52 and 8K RAM	\$189.00	Single Qty.
BCC52C	Low-power CMOS version of the BCC52	\$199.00	
BCC52I	-40% to +85°C industrial temperature version	\$294.00	
BCC52CX	Low-power CMOS, expanded BCC52 w/32K RAM	\$259.00	

CALL FOR OEM PRICING



MICROMINT, INC. 4 Park Street, Vernon, CT 06066
in Europe: (44) 0285-658122 • in Canada: (514) 336-9426 • in Australia: (02) 888-6401
Distributor Inquiries Welcome!

Listing 6—continued

```

LCDScrollLeft(CurDotRow, CurDotCol, FontHeight, FontWidth)
break:

default :
LCDWriteGlyph(CharCode, CurDotRow, CurDotCol); /* display it
*/
CurDotCol += FontWidth; /* tick column counter */
if (CurDotCol > (NumDotCols - FontWidth)) { /* past eol?
*/
CurDotCol = 0; /* reset to left edge */
if (!HWrapAllowed) {
LCDSendChar(LF); /* and step to next line */

```

interested in this sort of thing, but I won't spend any more space on them here.

One thing you can't do with ASCII control codes is "back up" the position of the next output character without affecting the existing text. Once you reach the last row of the screen, you

can't get back to the top without clearing everything, which is a bit extreme. Fortunately, the ANSI standards committee set down some rules to handle cursor control on ordinary terminals that we can adapt to our graphics LCD panels.

ASSUME THE POSITION...

Longtime INK readers are familiar with ANSI control sequences; most recently they appeared in the HCS II LCD-Link terminal described in Issue 27. For our present purposes, I'll simply say that they allow the same output string to produce similar results on anything that can interpret the commands.

Figure 1 summarizes the control sequences used in the graphic LCD character interface. Each sequence starts with the same two characters: 0x1B (Escape) and 0x5B (the left square bracket: "["). Next come any numeric parameters represented by ASCII digits and the terminating letter that identifies the command. The case of that final letter is significant: ESC[2j puts the cursor at row 2, column 1

All control sequences begin with two common characters:

ESC - ASCII 27, the escape character
 [- ASCII 91, the left square bracket

There are no blanks within the command strings.

Numeric parameters are decimal and default to 1 if omitted.

Row and column numbers start with 1 at the upper left.

The upper/lower case of the trailing letter is important!

Command	Example	Function
ESC#A	ESC[2A	Cursor up # rows (up 2)
ESC#B	ESC[B	Cursor down # rows (down 1)
ESC#C	ESC[10C	Cursor right # columns (right 10)
ESC#D	ESC[5D	Cursor left # columns (left 5)
ESC[#;#H	ESC[H	Set cursor to row;column (default 1,1)
ESC[#;#f	ESC[1;2f	Set cursor to row;column (1,2)
ESC[#;#j	ESC[3j	Set cursor to row;column (3,1) (H,f, and j are all synonyms)
ESC[s	ESC[s	Save current cursor location (1 level)
ESC[u	ESC[u	Restore saved cursor location
ESC[2J	ESC[2J	Clear display and home cursor
ESC[K	ESC[K	Clear from cursor to end of row
ESC[#h	ESC[7h	Set display mode, ignored except for Wrap at end of each row
ESC[#l	ESC[7l	Reset display mode, ignored except for no wrap at end of row
ESC[#m	ESC[0m ESC[7m	Set display attributes, ignored except for Disable all attributes Set blinking attribute

Figure 1—ANSI Control Sequences

AMX™

The Real-Time Multitasking Kernel

680x0, 683xx
 80x86/88 red mode
 80386 protected mode
 i960® family
 R3000, LR330x0
 Z80, HD64180

Features

- Full-featured, compact ROMable kernel with fast interrupt response
- Preemptive, priority based task scheduler with optional time slicing
- Mailbox, semaphore, resource, event, list, buffer and memory managers
- Configuration Builder utility eases system construction
- InSight™ Debug Tool is available to view system internals and gather task execution statistics
- Supports inexpensive PC-hosted development tools
- Comprehensive, crystal clear documentation
- No-hidden-charges site license
- Source code included
- Reliability field-proven since 1980

Count on KADAK.
 Setting real-time standards since 1978.

For a free Demo Disk
 and your copy of our excellent AMX
 product description, contact us today.

Phone: (604) 734-2796
 Fax: (604) 734-8114



KADAK Products Ltd.
 206 - 1847 West Broadway
 Vancouver, BC, Canada V6J1Y5

AMX is a trademark of KADAK Products Ltd.
 All trademarked names are the property of their
 respective owners.

Listing 7—Decoding the ANSI control sequences depends on a state machine to track all the allowable parameters and command terminators. This function is invoked for each new character sent to the LCD panel. It extracts the numeric parameters and passes control to the function decoder when the final letter occurs.

```

void LCDAnsiDriver(int NewChar){

ANSIBuffer[ANSICharCtr] = NewChar & 0x7F; /* strip & save */
++ANSICharCtr;
ANSICharCtr = min(ANSICharCtr,ANSI_MAXLEN-1);

switch (ANSIState){
case ANSI-WAIT :           /* waiting for ESC char */
    ANSIParamCtr = 0; /* force buffer restart */
    if (ESC == NewChar){
        memset(ANSIBuffer,0,sizeof(ANSIBuffer));
        ANSIState = ANSI-MODE;
    }
    else {
        LCDSendChar(NewChar); /* just dump it */
    }
    break;
case ANSI-MODE :           /* expect bracket as next char */
    if ('[' == NewChar){
        ANSIState = ANSI_PARAM;
    }
    else {
        LCDAnsiNull(NewChar); /* ditch sequence */
    }
    break;
case ANSI_PARAM:
    if ('=' == NewChar){
        break; /* gobble this one.*/
    }
    else {
        if (isdigit(NewChar)){
            ANSIState = ANSI_NUM; /* still a number */
        }
        else {
            if (isalpha(NewChar)){
                LCDAnsiGetNumber0; /* get number if any */
                LCDAnsiDoCmd(NewChar); /* end of sequence */
            }
            else {
                LCDAnsiNull(NewChar);
            }
        }
    }
    break;
case ANSI_NUM:
    if (':' == NewChar){ /* end of param digit */
        LCDAnsiGetNumber0; /* save prev param */
    }
    else {
        if (isdigit(NewChar)){
            ANSIState = ANSI_NUM; /* still a number */
        }
        else {
            if (isalpha(NewChar)){
                LCDAnsiGetNumber0; /* get number if any */
                LCDAnsiDoCmd(NewChar); /* end of sequence */
            }
            else {
                LCDAnsiNull(NewChar);
            }
        }
    }
    break;
default :
    LCDAnsiInit(); /* error, restart decoder */
}
}

```

while ESC[2] clears the panel and homes the cursor.

Decoding these sequences is made somewhat difficult by the fact that the numeric parameters for some commands may be duplicated or omitted. The state machine shown in Listing 7 tracks the possible combinations and calls a decoder routine when it encounters the final command character. Every character sent to the display must pass through this routine, so there is a moderate performance penalty associated with implementing ANSI controls.

The actual ANSI function routines are nearly anticlimactic, as you can see in Listing 8. Most are just a line or two of code that adjusts the cursor position or sets a control variable that affects subsequent characters. Adding new commands is easy: a new case clause and the requisite code are all you need.

One conflict between ANSI cursor controls and Glass TTY mode occurs when you write the character in the extreme lower right corner: row 25, column 80 for a 200-line display. In TTY mode, the panel scrolls upward by the font height to make room for the next line of characters. Because ANSI controls are generally used to create a static display image, perhaps with a tidy Wind-oid border around the whole screen, you don't want vertical scrolling.

You can enable line wrapping by sending ESC[7h and disable it with ESC[7l around the offending characters, but that's a nuisance. Because I plan to use the panel as a static display the ANSI initialization routine simply disables vertical scrolling. I can't find a control sequence to handle this, but I won't be surprised if one of you folks comes up with the Official Doc on it.

So, there you have it: a reasonably fast, large, and cheap character output device that leaves all the standard PC hardware untouched. Now you can run a program that uses the serial port or video display and still get intelligible diagnostic information out!

THE REST OF THE STORIES

The 640x200 TLY-365-121 panel is similar to the DMF651, although the

Listing 8—This excerpt from the ANSI function decoder shows that handling the commands is mostly a matter of a few calls to routines in the LCDChars module that twiddle the cursor location. Most ANSI functions are ignored so the LCD panel isn't confused by functions it cannot perform; the same character string sent to a "real" ANSI terminal might set the display color, for example.

```
void LCDAnsiDoCmd(int NewChar){
  BYTE Counter;
  ANSIState = ANSI-WAIT;      /* by default...*/

  switch (NewChar){

  case 'A' :/* cursor up */
    LCDSetCharCursor(LCDGetCharRow()-
      ANSIParams[0],LCDGetCharCol());
    break;

  <<< cases omitted >>>

  case 'H' :/* set cursor position */
  case 'f' :/* set cursor position (synonym) */
  case 'F' :/* set cursor position (synonym) */
    for (;ANSIParamCtr < 2; ++ANSIParamCtr){
      ANSIParams[ANSIParamCtr] = 1; /* force defaults */
    } /* fix origin! */

    LCDSetCharCursor(ANSIParams[0]-1,ANSIParams[1]-1);
    break;
```

(continued)

electrical rows are 1280 dots long. Tweak the DMF65 1 character address routine and it'll work fine.

The 480x128 LM215, as always, presents a challenge. Because the four dots in each byte appear in different quadrants, you must set one bit at a time. Drawing a single 8x8 character thus involves 64 calls to the `LC D S e t D o t` function described last month. Modify that code to support blinking if you really need it. All this for 960 characters?

The 640x400 LM64015T should be a piece of cake because each bitmap byte shows up in the same Refresh Buffer byte. I didn't write code for it because nobody can track down the CCFT backlight inverters; I have no desire to build a kilovolt power supply.

If you need help tweaking the code to draw characters on your panel, drop me a note on the Circuit Cellar BBS and our collection of experts will help push you over the top. I trust you see how easy it is by now, though.

RELEASE NOTES

The BBS files this month include source and binary files to check out the Glass TTY and ANSI code on both the DMF65 1 and LG64AA44D panels, which are the "best of breed" in my collection. The code is written in Borland C and processed through Paradigm Locate. You'll need the usual diskette with our boot sector loader to use the files.

Paradigm's Locate demo has a 16K file size limit, which hasn't been a problem up until now. I made liberal use of `printf`, `scanf`, and a few other storage-hogging functions to simplify these programs, so the resulting code is slightly larger than 16K. Locate produces binary files in power-of-two increments, which means a file one byte over 16K uses 32K. If you're using the Locate demo, you can reduce the program size by replacing those big functions with simpler ones; think of it as practice for when your program exceeds 640K.

Several readers found stashes of 640x480 VGA-class panels and asked how they might work with the Graphics LCD Interface. As I mentioned in Issue 43, the main problem is

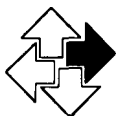
Multi-Axis

Motion Control



1 to 8 Axis Motor/Motion Controllers

- DCXcontrollers incorporate state-of-the-art DSP, RISC, and FPGA technologies to insure maximum performance at minimum cost
- DCXcontrollers provide many sophisticated ON-BOARD control features such as MULTI-TASKING (the ability to perform independent tasks simultaneously), COMPLEX-CONTOURING, and the ability to programmed directly in USER-UNITS such as inches, meters, etc.
- DCXcontrollers can operate totally STAND-ALONE, using their own on-board "computing + memory + I/O" capabilities, or EMBEDDED in a machine. They can also be installed in an ISA-bus (PC/XT/AT), or in a VME-bus (6U)
- DCXcontrollers have a MODULAR ARCHITECTURE for creating a limitless variety of Application Specific MULTI-AXIS controllers. They are created, in minutes, by installing one or more intelligent *DCX function modules* on an intelligent *DCX motherboard*
- DCXcontrollers are supplied with (free) EASY-TO-USE, POWERFUL SOFTWARE for set-up, programming and operating. These include all major HLL interface programs, with examples and source code, a CNC control user-interface with G-code and HPGL capability, and an interactive Servo-Tuning utility



Precision MicroControl
CORPORATION
8122 Engineer Rd. • San Diego, CA 92111

TEL (619) 565-1500
FAX (619) 565-1511

#137

the 54-Hz refresh rate produced by my choice of a 480-ns Dot Clock: the excellent backlights on these panels produce an annoying flicker. They also require 38,400 bytes of Refresh Buffer, so a piggyback RAM expansion is in order, too.

Several other readers bought LCD panels with no accompanying documentation: a bad idea! The fact is, if you don't have doc for a panel when you buy it, you probably won't be able to use it. Unless it's one that I've described here, make sure you see the doc before you spend any significant money on a graphic LCD panel! 📄

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

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.

SOURCES

Endicott Research Group (ERG) has several lines of inverters for backlight applications. They're the factory source and don't want to deal with small orders, but if you happen to need a few kilobucks of inverters they can help you out. Fax Scott Barney at (607) 754-9255 for information.

Pure Unobtainium has the complete Firmware Development Board schematic, as well as selected parts. Write for a catalog: 13109 Old Creedmoor Rd., Raleigh, NC 27613. Phone or fax (919) 676-4525.

IRS

413 Very Useful
414 Moderately Useful
415 Not Useful

Listing & continued

```

case 'J' :/* erase display & home cursor */
  if (2 == ANSIParams[0]){
    LCDClear();

    break;
case 'K' :/* erase to end of line */
  Counter = NumCharCols  LCDGetCharCol();
  while (Counter) {
    LCDSendChar(DEL);
    --Counter;

    break;

<<< cases omitted >>>

case 's' : /* save cursor location */
  ANSISavedRow = LCDGetCharRow()+1;
  ANSISavedCol = LCDGetCharCol()+1;
  break;
case 'u' : /* restore cursor location */
  LCDSetCharCursor(ANSISavedRow-1,ANSISavedCol-1);
  break;
default :
  LCDAnsiNull(NewChar);/* dump all others...*/

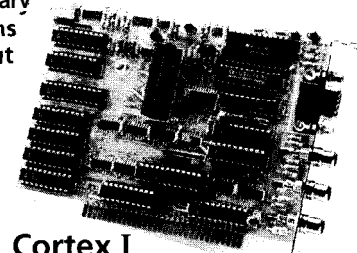
```

Video Frame Grabber

- \$495 Including Software with "C" Library
- Half Slot Card for Compact Applications
- Real Time Imaging with Display Output
- 8 Bit (256 Gray Levels)

FEATURES:

- Single 512 X 484 Image or Four 256 X 242 Images
- External Trigger
- Input Look Up Table
- Low Power Option Available
- Elegant Software Interface
- <10 nsec Pixel Jitter Means Accurate Digitization
- EISA (PC) Bus and STD Bus Products Available
- RS-170 and CCIR Video Formats Available
- Binary and TIFF File Formats



Cortex I

The Cortex I is used in machine vision, industrial control, medical, security and scientific applications around the world.

ImageNation strives to delight customers with quality products and personal service at a competitive price.

Call today for volume pricing or to discuss your application.

ImageNation Corporation
Providing Imaging Solutions
FAX (503) 643-2458 • (800) 366-9131

P.O. Box 276
Beaverton, OR 97075
(503) 641-7408

Audio/Video Traffic Control

FROM THE BENCH

Jeff Bachiochi



When Steve announced his plans for a new "entertainment room," I knew there was bound to be a flurry of new equipment purchases. He knows what he wants and if he can't find the right piece of equipment, he'll build it. The day Steve divulged his intentions for this project, I had mixed feelings.

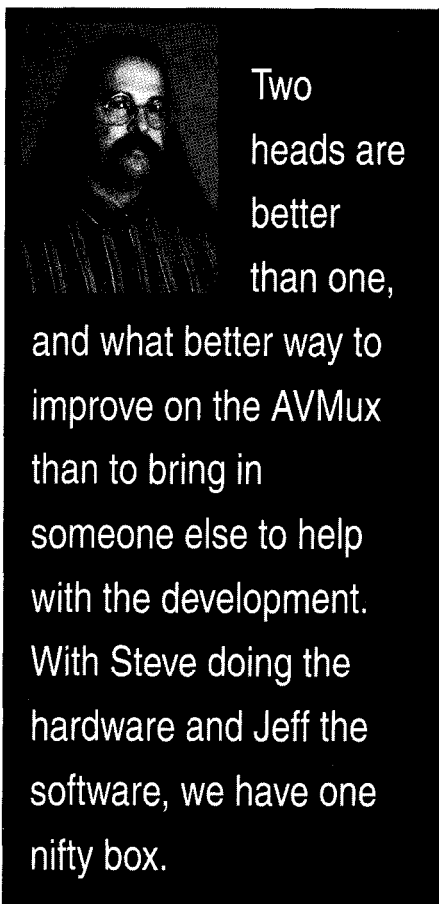
You see, I've been meaning to put together a video switcher since I built our addition three years ago. At that time, I installed a coax line to each of the rooms. I figured this would eventually allow me to choose from various video sources from each remote location. If I didn't have a hand in this project, I may not be getting the features I was looking for. On the other hand, if I collaborated, it could end up being a better project all around.

By the time Steve and I had exchanged our wish lists, it was clear that we were going in two entirely different directions. Steve wanted independent audio and video switching capabilities. I was looking for remotely accessed video-only switching. Our modular design approach allows one to trim away unwanted functions

without having to redesign the entire system.

Although prototyped on the same PC board, the video and audio sections are actually separate designs and could be separate PC boards. With this approach, the multiplexer could include a video switcher, an audio switcher, or both.

As Steve previously explained, control of the multiplexer can come from a variety of sources. First, a serial terminal connection gives the user an access port for program maintenance. With the mode switch set in local mode, a matrix of video and/or audio connections is presented to the terminal, along with instruction prompts for changing the matrix connections. In remote mode, ASCII-coded matrix data is transmitted and is intended for use with the remote LED matrix display. The display indicates the present video and/or audio connections on an 8x8 LED array and prompts the user to provide additional commands using the attached keypad. A



Two heads are better than one,

and what better way to improve on the AVMux than to bring in someone else to help with the development. With Steve doing the hardware and Jeff the software, we have one nifty box.

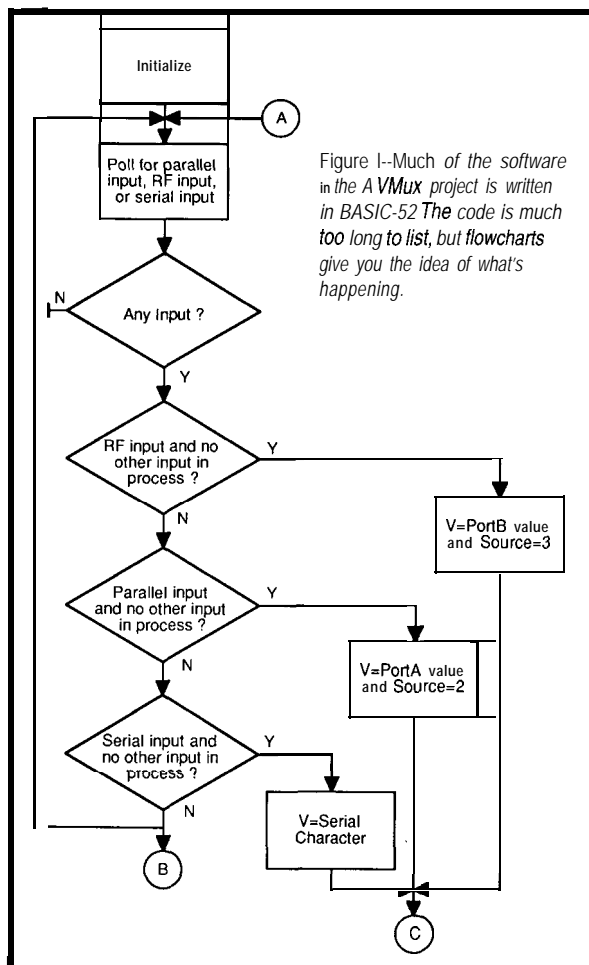


Figure 1--Much of the software in the AVMux project is written in BASIC-52. The code is much too long to list, but flowcharts give you the idea of what's happening.

second (auxiliary) serial output contains connection information displayed in a from/to listing using the actual names of equipment connected to the AVmux's I/O jacks.

Input can come from two additional sources (other than the serial input). An 8255 is configured as a pair of strobed parallel input ports. The first port is used like a printer port, accepting parallel data from a PC or DIO-Link. The second port interfaces to Steve's RF receiver (see his article on page 36 of this issue) to decode remote keypad pushbuttons.

Two configuration switches are used. The first, as stated earlier, selects a "local mode," where commands can arrive from a serial terminal, or an "LED matrix mode," where commands can arrive from the hand-held LED matrix and control terminal. The

second switch enables or disables the auxiliary serial output, which is meant for display through a video monitor.

A/V SOFTWARE

The main code is written in BASIC-52 with a short assembly language call used to bit bang the array data into the video multiplexer and audio multiplexer. Both devices use a clocked-serial bit stream as input with a load pulse to make the final parallel transfer of data from the internal shift registers. If the audio multiplexer did not have a minimum clock rate of 20 kHz, I could have skipped the assembly stuff, but the need for speed mandated it. Parameters are passed to the routine through a few internal user registers prior to making the call. The routine grabs data from one of the two data tables held in protected memory

above MTOP. The data is transferred by the CALLED routine and shifted into both multiplexer devices. Depending on which device is the destination, the appropriate load strobe is pulsed.

The assembly language routine could have been permanently set in place, but I chose to embed the routine into the BASIC program as DATA statements and XBY (poke) it into protected memory when the BASIC program is run. This allows all the code to exist in a single easily maintained program. See Figure 1 for a gross overview of this program.

The BASIC code begins with a little house cleaning and then reads the DATA statements into protected memory. A few checks are made at this point. First, a checksum is calculated to ensure the DATA statements were read correctly. Next, the

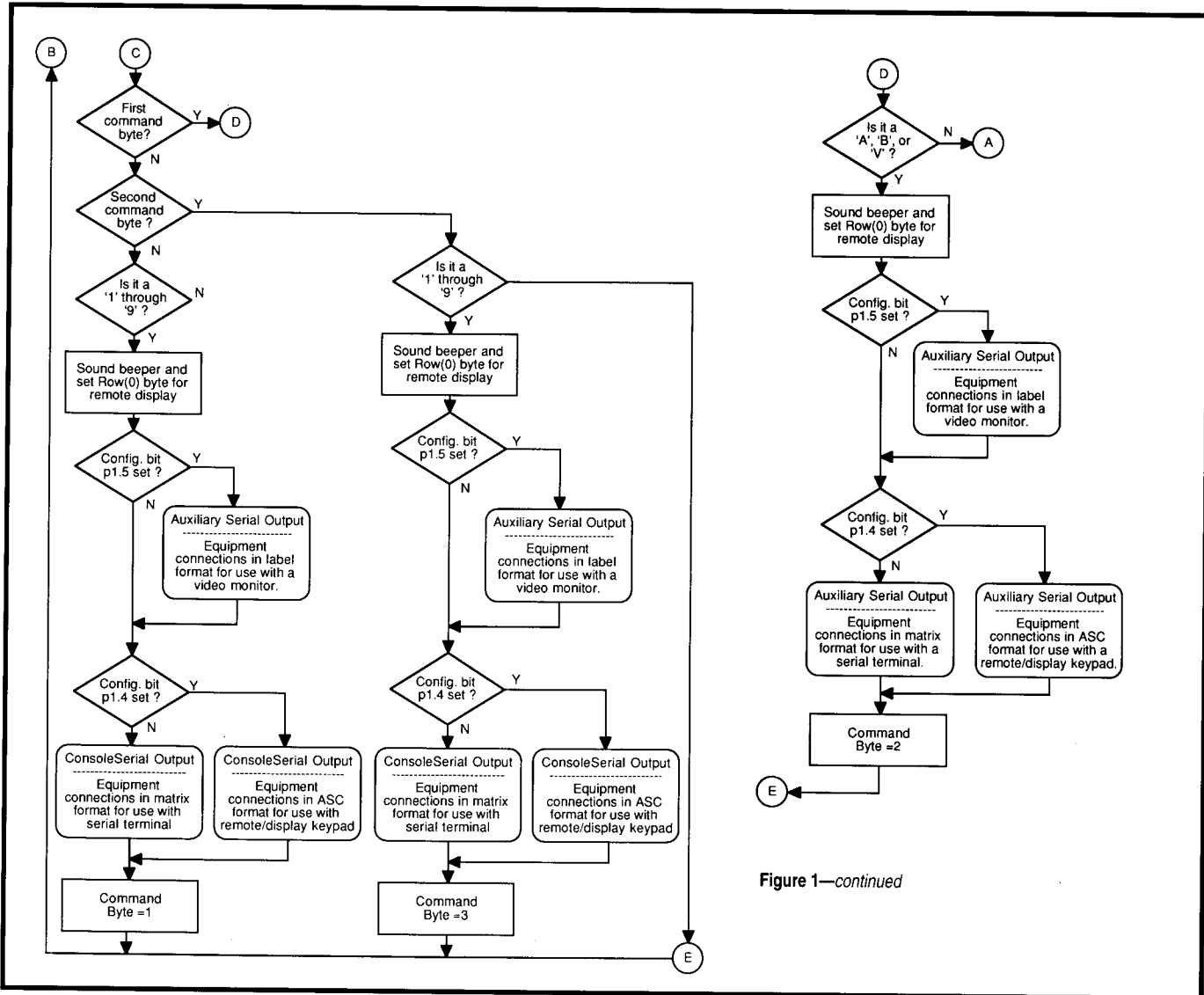


Figure 1—continued

program verifies that there was protected memory in the system to accept it. We wouldn't want to make a CALL without being sure we could get back.

Four tables are kept in protected memory just below the above routine. Table 1 has four bytes of video-array data. Each byte holds the connection data for two output channels, one in each nybble. Legal nybble values are 0-7 (connected to input channel 1-8) or 9 (which means a grounded input).

Table 2 holds 32 bytes of audio-array data. Each word (two bytes) holds a 16-bit data value that indicates whether any of the 16 inputs is connected to a particular output. A "1" in a particular bit position (bits 0-15 equate to inputs 1-16) signifies that that input is connected to the output associated with that output word (here 0-15 means outputs 1-16). Because the audio matrix is actually 16x16 and we want to use it as a dual [stereo] 8x8 matrix, the matrix is divided up into

four pieces: X1-8 (left inputs), X9-16 (right inputs), Y 1-8 (left outputs), and Y9-16 (right outputs). Notice that some connections are considered illegal even though they can be logically made (i.e., left input 1 is not allowed to be connected to right output 1).

Table 3 is a list of labels for each of the 16 video and 16 audio I/O connections (24 characters maximum each). I choose not to prompt for these labels from within this program. But instead, since you would have to connect a terminal device up to enter the data anyway, I created a short program containing the labels (which you edit directly in the program before downloading it). When the program is run, your labels are inserted into protected memory. This program can be easily edited and run if you ever make connection modifications.

Table 4 is a simple 2-byte flag that indicates if the program has been run

before. Once the program has been run, we don't want to initialize the A/V multiplexer with no connections, because in all likelihood we want to initialize all connections to their prior state before the power was removed [the last time it was on].

By checking Table 4, we know how to initialize the audio and video multiplexers. The data from Tables 1 and 2 are transferred using the CALL routine and we are now ready for user input. The first legal input is restricted to either an "A" (audio), a "B" (Both), or a "V" (video). This could come in through the serial port or through the 8255 PPI (recall that this is set up as two strobed input ports). The 8255 handles both parallel inputs, one from parallel port and one from the RF receiver. Once a legal character is received from one device, the others are locked out until the command is completed. This is fulfilled by receiving two additional characters in the

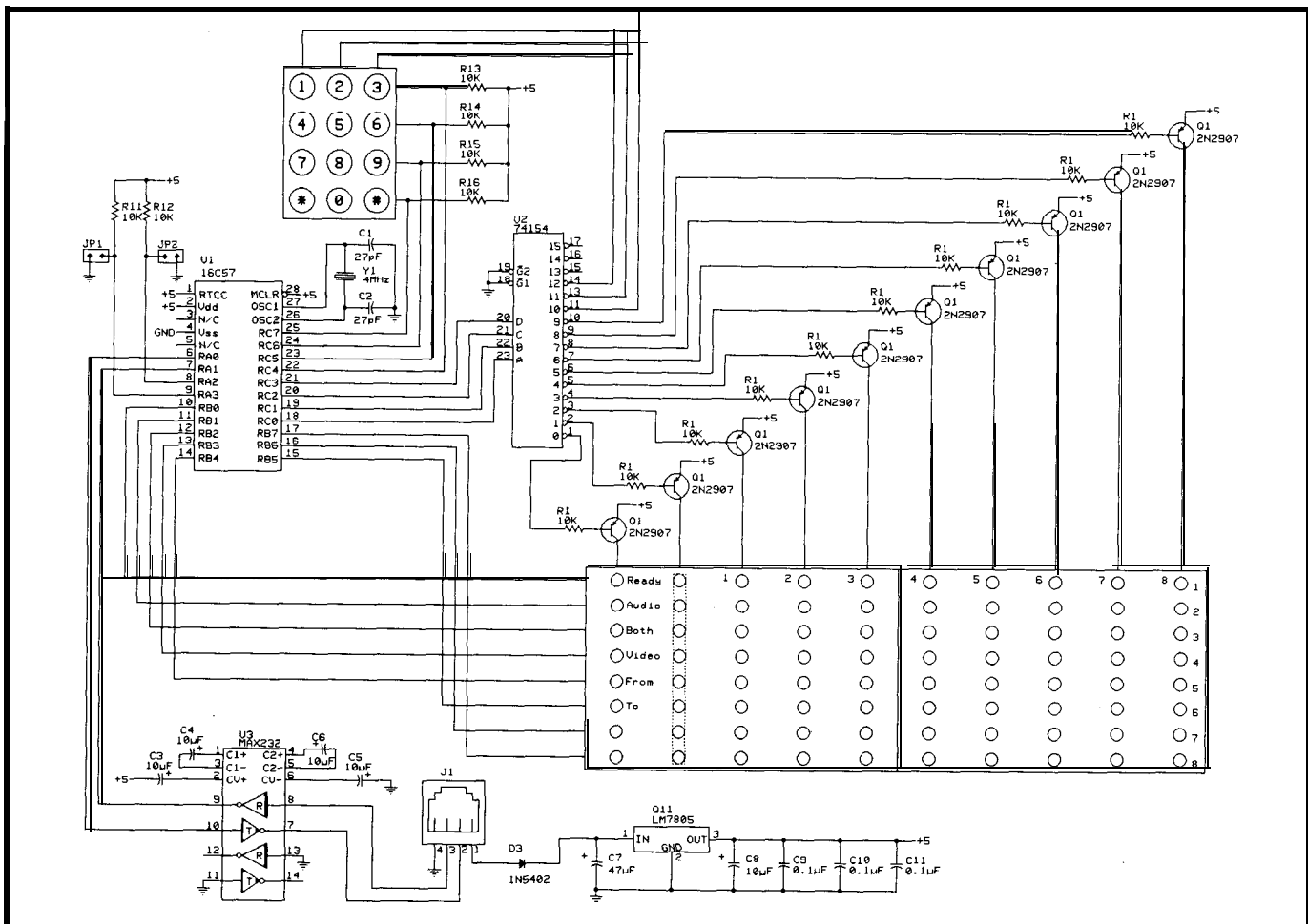


Figure 2-At the core of the AVMux remote is a PIC controller chip. To maximize the use of the PIC's I/O lines, the keypad scanning is multiplexed with the displays. A typical MAX232 is used to talk to the AVMux's 80C52 processor serially

range of "1"- "9." The first character selects where the audio or video signal is coming from, which can be input 1-8 or 9 (which is equivalent to none). The second character determines which output the input is assigned to, which can be output 1-8 or 9 (to cancel the command).

After the reception of each legal character, the configuration switches determine which BASIC subroutines are used to output the appropriate feedback to one or more of the display devices.

REMOTE LED MATRIX AND KEY PAD

Perhaps you want to view the I/O connections selected for the audio and/or video multiplexer without the

Photo 1--The 8x8 LED matrix makes for an *easy-to-read* panel when trying to discern which and what type of I/O connections have been made.

hassles of a serial terminal or video monitor attached. An 8-column by 8-row matrix of LEDs will do nicely in this event. Each column represents an output or signal source while each row designates an input or component destination. A glowing LED at the intersection of a source column and a destination row designates a connection. See Photo 1.

Individual LEDs could be used, however I had a few 5x8 LED arrays left over from an earlier FTE project (the scrolling LED display). Two of these make a 10x8 matrix; the first

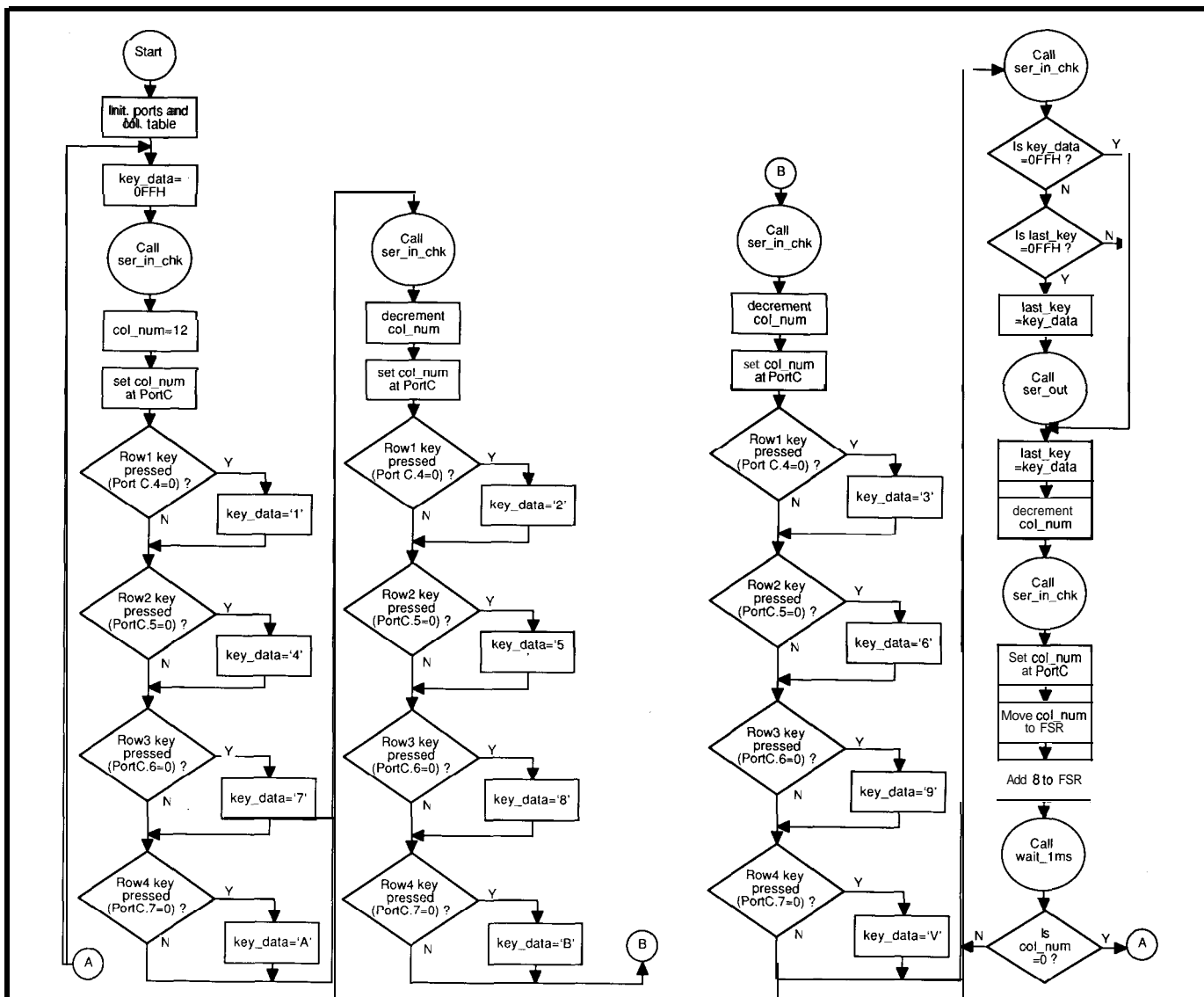


Figure 3—The PIC must simultaneously scan the keypad, scan the display, and watch for serial data from the AVMUX. The lack of interrupts necessitates lots of polling, which can make the code look more complicated than it is.

two columns are not used for matrix indicators, but for prompting input from the user. Six LEDs in the first row are labeled as Ready, Audio, Both, Video, From, and To. When a command is finished, the "Ready" LED prompts the user to choose from either the Audio, Video, or Both matrices. If entered correctly on the keypad, the "Audio," "Video," or "Both" LED comes on along with the prompting of the "From" LED. The 8x8 LED matrix is updated to show the present connections of the user's chosen matrix. When a column number is entered on the keypad by the user (1-8 or 9 as none), the "To" LED is turned on prompting the user for a final entry, which will be the destination. A row number completes the command (1-8 or 9 to cancel) and updates the 8x8 LED matrix with the new connection data.

You may wish to have this display out front with the rest of your component controls or closer at hand. Therefore, its design incorporates a serial transmission scheme. The umbilical cord, a modular phone cable, carries RS-232 transmissions using three conductors-transmit, receive, and ground-while the fourth conductor carries power for the I/O display. This allows the remote LED display unit to be a considerable distance from the switching electronics or simply on a shelf near the equipment.

CODED ASCII MATRIX DATA

Only ASCII [printable] characters are transmitted to and from the remote LED display, making debugging easy when using a serial terminal. The keypad simply transmits ASCII O-9, A, B, or V whenever the appropriate key is pressed.

The received LED data is a bit more complicated. There are in fact 10 possible columns (O-9) of data which could be passed to the display. To keep things straight, I use a two-byte ASCII transmission for each half column. The first byte is the column number (O-9)+20h. The second byte is the nybble value (O-F)+40h (for the lower nybble of data), or (O-F)+50h (for the upper nybble of data). This might seem a bit complex, but notice each byte is a

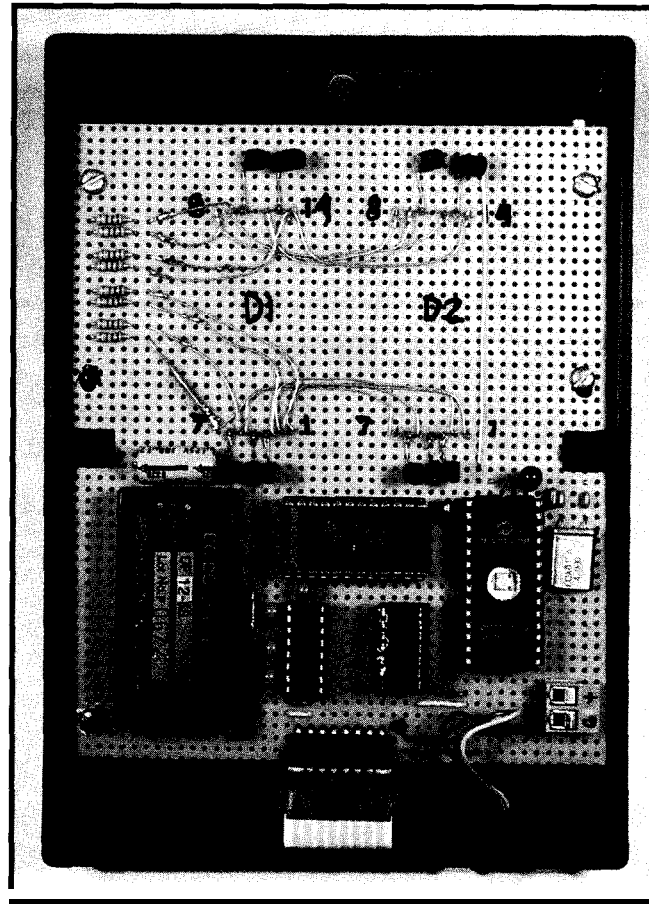


Photo 2—The back of the remote shows the PIC (right), multiplexer (center), and DC-DC converter (left) that efficiently converts the 12 volts supplied to it to the necessary 5 volts. The schematic in Figure 2 shows the use of an LM7805, which is a cheaper, smaller, and hotter solution.

printable character and you can tell at a glance what the data is and where it goes.

The display can actually be updated very rapidly and may bring to mind other uses for this technique from information displays to electronic art.

A PINCH OF HARDWARE

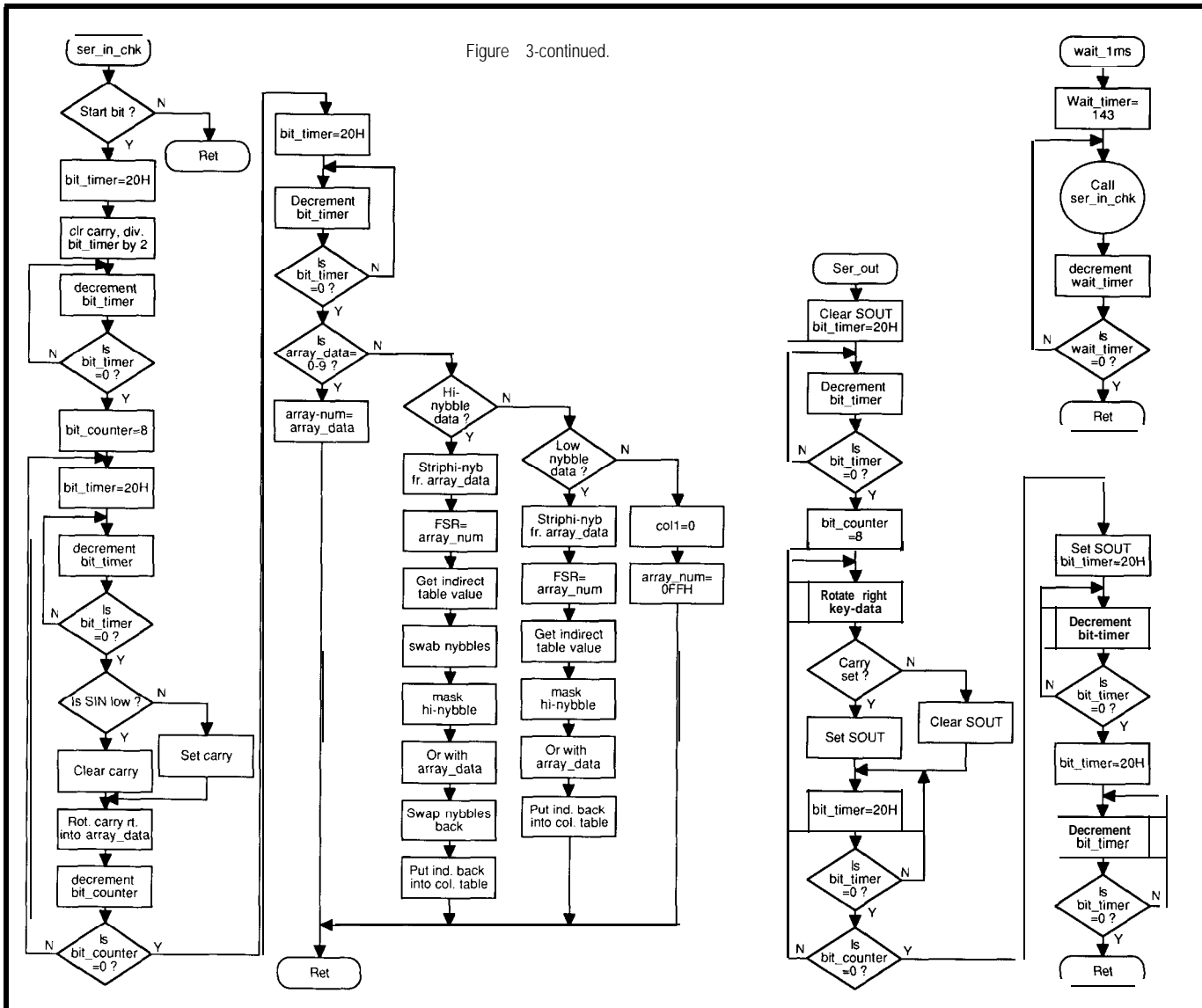
Little is needed beyond a micro, a MAX232, and a 4-to-16-line decoder to handle serial data, keypad scanning, and LED matrix multiplexing (see Figure 2). The 10 columns of LEDs and 3 columns of the keypad are enabled one at a time by the 74HCT154. The LED's "row data" is output on Port B of the micro. The lower half of Port C drives the '154 to select which column is enabled. The upper half of Port C uses the keypad rows as inputs (they are held high with pull-ups.) Port A is used for communication (RX and TX). See Figure 3.

The upper three enables [of the '154] are not needed, so its addressing begins with address 12 (decimal),

which is the first keypad column. If a key in the first column is being pressed, the associated row will be pulled to ground. The code looks for a low on any keypad input during the time in which the column is enabled. If a key press has been detected during any of the three column enables and no key was down during the last check, then we jump to the serial output routine and transmit one character at 9600 bps. If no keys were pressed, then the code moves on to scan the LED column drivers. Row output data is updated as each of the columns are enabled. A short pause of 1 millisecond is executed between each column enable to allow enough time to pass so as to let the illuminated LED stimulate the eye's retina enough to provide for persistence of vision. Once all the columns have been scanned, we go back and look for a key press once again. This cycle time also acts as a debouncing to the switches.

One more thing. We have to pause every now and then [actually quite

Figure 3-continued.



often] to look for a start bit coming into the micro. No interrupts are available, so polling must take place in a timely manner. By checking the RX pin after every column enable (and also during the 1 millisecond pause) we are assured of capturing the serial data within the allotted time frame. After the receipt of a character, a determination is made as to what to do with it. It can point to the appropriate column table offset, or if it's data, it can be placed into either the upper or lower nybble in the table. If the character falls outside the legal boundaries, it is tossed out and an error is displayed.

Errors are displayed by lighting the entire column of LEDs between the first column of command prompting LEDs and the last columns which make up the 8x8 matrix. New data

sent to the remote should always turn off the error LEDs to clear any error status.

A LAST LOOK

Did we actually succeed in creating a piece of equipment which satisfied both of our needs? Well, Steve has already completed the integration of the AVmux into his new entertainment room and although I can't smell the popcorn from here, I have seen "Top Gun" vapor trails originating from high atop "Ciarcia Peak." As far as my installation goes, I am looking for a few more video sources to add. I just can't bear to leave those extra video inputs unconnected. 📺

Jeff Bachiochi (pronounced "BAH-key-AH-key") is an electrical engineer on

the Computer Applications Journal's engineering staff, His background includes product design and manufacturing. He can be reached at jeff.bachiochi@circellar.com.

SOFTWARE

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

IRS

- 416 Very Useful
- 417 Moderately Useful
- 418 Not Useful

Honey, I Shrunk the PC

SILICON UPDATE

Tom Cantrell



I've sat in many high-tech product planning meetings in my time, and am always amused by the "reasoning" that goes on when defining a new product. In fact, most profound (and thus, wildly successful) products tend to appear quite "unreasonable" before the fact.

Consider our friend the microprocessor, which was viewed by many as little more than a lowly desk calculator chip. "Why does the world need a computer chip?" said the suits, pointing to the minuscule (by IC company standards) quantity of mainframes and minis sold each year. Lucky for us Intel was still small enough at the time to give the visionaries a chance.

A company's ability to lead and innovate seems directly proportional to the degree they foster risk taking. Good companies will give a zealot rope while bad ones tie them up with it.

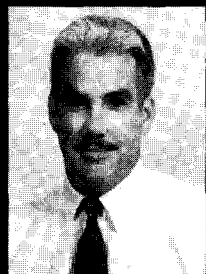
So, I've got to give Epson credit for the CARDIO, a product my "heart" (pronounced as CARDIOvascular) says is exciting, though my "head" hasn't a clue what it's good for.

FORECASTING FOLLY

The fact is, we're in the "food fight" era of product planning: throw any product you can think of into the market and hope it sticks. It's inevitable that many won't make it.

Nevertheless, when facing the administrative mandarins, a hapless product proponent must come up with some "analysis" purporting to "prove" that success is "guaranteed." Typically, the approach is to snow the bureaucrats with an avalanche of market "data," which is presented as insightful "information."

Thus, I had to smile when reading the CARDIO press materials that cited "portable equipment" and "other devices" as target markets. I could just see the product promoter sweating in front of a room of inquisitors, explaining—no doubt with reams of spreadsheets and upward sloping graphs—



First came the ATs, enough to

cover three quarters of the space located on your desk. Then came laptops, and then notebooks, and more recently, palmtops. Could it get any smaller? And to what end? Enter CARDIO...

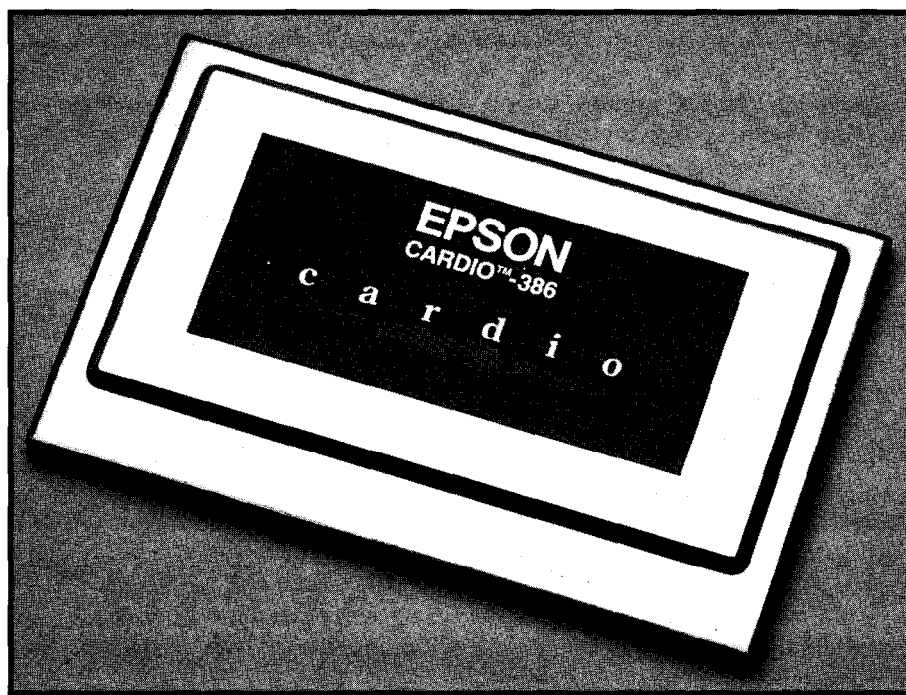


Photo 1—Epson's CARDIO closely resembles a credit card in size and appearance, but packs a lot of power into a small space.

how the “portable equipment” market looks rosy and the “other devices” market is sure to grow. It’s as satisfying as a good curve ball when I think of slipping a pitch like that by.

Fortunately, product planning in the high-tech arena is facilitated by the fact that even the most outlandish technical prophecies are likely to come true at some point, it being more a matter of when, than if. I’ve always subscribed to the approach of defining what will sell and then considering the messy details of if/when/how it can be built. That may sound unrealistic, but it’s better than the opposite approach -if it can be done, it should be done— that has littered the floor with products that could be made, but not sold.

MAKE OR BREAK

In that light, defining the CARD10 [Figure 1 and Photo 1] is quite easy—“Gee, what the world needs is a complete PC on a credit card.” The Marketeers exit left with ear-to-ear grins as they let the manufacturing folks chew on that.

“OK,” the manufacturers say. It turns out it is possible by relying on the latest miniaturized IC packaging techniques including bare die, MCMs (MultiChip Modules), and TSOP (Thin Small Outline Packages). See Photo 2.

Most advanced is the use of bare ‘386SL die relying on a technique known as TAB [Tape Automated Bonding]. Rather than mounting the die in a chip and the chip in the board, why not just dump the middleman and bond the die right to the board?

Actually, there is a reason for concern with TAB, MCM, and other techniques that rely on bare dice. It is referred to in the industry as the “Known Good Die” issue and revolves around the limited testability of bare dice and the impact on module yields.

It turns out that a die is partially tested in wafer form, known as “wafer sort.” Typically, a probe steps and repeats across the wafer, performing cursory tests on each die and marking the bad ones. The idea is to avoid the high cost and waste associated with bonding a bad die into a package.

However, it isn’t the usual practice to attempt exhaustive testing

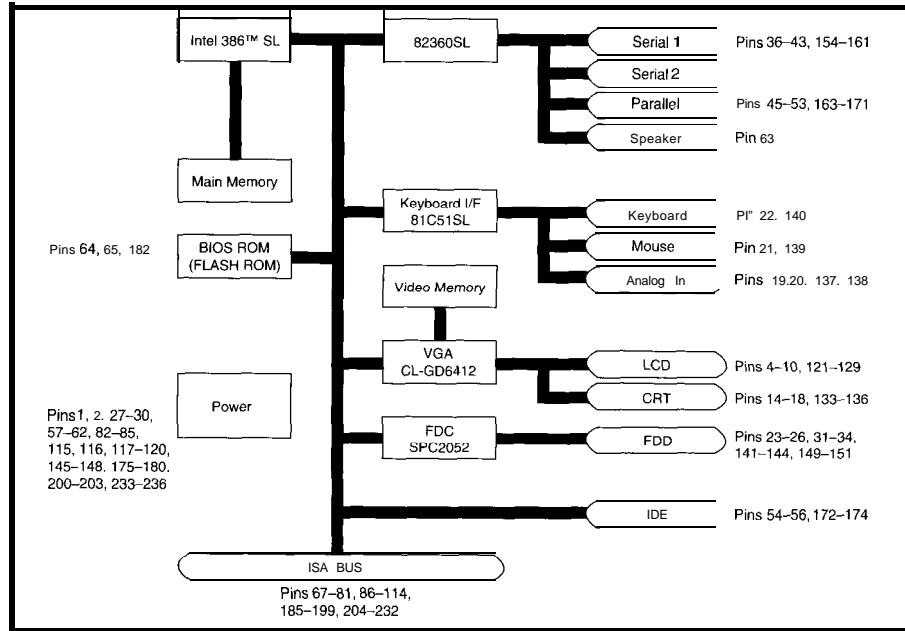


Figure 1—Virtually every signal necessary to interface the CARD10 with a bus and common peripherals is brought to the outside world.

at the wafer level for a variety of technical and cost reasons. Some tests—such as temp cycling—aren’t feasible and the cost and throughput of testing are worse at the wafer level than at the packaged chip level. At some less-than-exhaustive level of test, it becomes cheaper to go ahead and bond the die in the package and hope for the best.

The bottom line boils down to a situation in which a percentage of the dice that pass wafer sort will ultimately prove bad—something unfortunately discoverable only after the entire module is built. Worse, while an individual parts yield may seem satisfactory, the situation quickly

degrades as the number of dice in the module increases.

For instance, what if 10% of the dice that pass wafer sort and are assembled into the module prove faulty? Sounds pretty good, but let’s say the module has four dice, in which case the module yield falls to 66%, meaning a third of the completely assembled modules head for the scrap heap. So far, the option of “repairing” bad modules has proven infeasible.

Epson says that their own internal yield analysis shows that, up to a limit of four to eight chips or so (depending on the specific technology used), bare dice are cost competitive with packaged chips. Perhaps concerns about

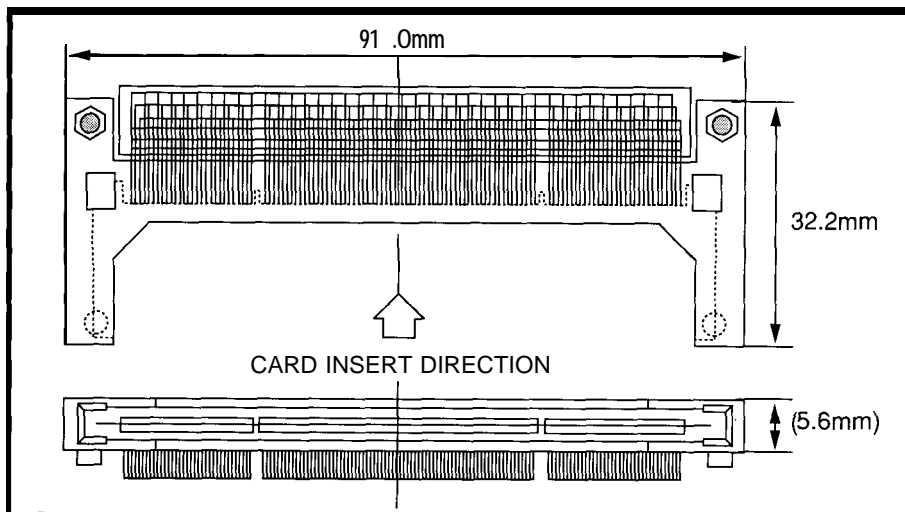


Figure 2—The EASI (Epson All-in-one System Interface) squeezes 236 signal pins into a half-square-inch area

yield are best measured by the price (about \$400 in OEM quantities for the 16MHz/1MB/128KB version) which, though higher than a brand-x motherboard, seems quite reasonable. The good news is costs eventually fall and at each step along the way, a new round of applications become feasible.

EASI DOES IT

OK, so manufacturing falls for it and commits to building the beast. Oh, there's just one question—just which signals should be brought out?

You announce that, after careful consideration of the requirements of the “portable” and “other” device markets, the answer is, “What the heck, how about all of ‘em?”

At this point, it might be wise for you to disappear for a time while the manufacturing folks grapple with your latest brainstorm, lest you be an easy target for their frustration.

Ultimately, the answer is *EASI*—the “Epson All-in-one System Interface”—exploiting an ultra high density connector (Figure 2) to pack a whop-

PRELIMINARY					
		5V Line		3.3V Line	
Clock	Memory	Typical	Maximum	Typical	Maximum
16 MHz	1 MByte	190 mA	348 mA	450 mA	770 mA
16 MHz	4 MByte	190 mA	348 mA	630 mA	1060 mA
20 MHz	1 MByte	190 mA	348 mA	485 mA	815 mA
20 MHz	4 MByte	190 mA	348 mA	695 mA	1105 mA
Suspend	1 MByte	7 mA	13 mA	4 mA	7 mA
Suspend	4 MByte	7 mA	13 mA	5 mA	9 mA

Note: In the Suspend mode power is not determined by the clock speed.

Figure 3—Typical *CARDIO* power draw is on the order of a few watts when running and a few milliwatts in suspend mode.

ping 236 signals into an area of about 1/2 of a square inch!

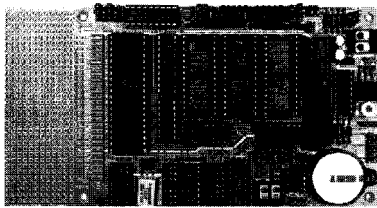
For the most part, the signals are a straightforward one-for-one mapping of the corresponding desktop connectors including ISA, FDD, IDE, CRT, keyboard, mouse, and serial/parallel ports. However, a few additions and differences are worth noting.

One of the most notable extras is the bit-mapped LCD interface consisting of 8 data bits and horizontal, vertical, and shift timing outputs (see Ed Nisley's recent “Firmware Fur-

nace” articles for a complete discussion of the subject). It is compatible with 640x480 mono panels from Epson and can likely interface with other manufacturers' panels as well.

The *CARDIO* requires a 3.3V/5V supply (the DRAM and the 386SL run at 3.3 V, everything else at 5 V), so the AT's +12-V and -5/-12-V supplies aren't needed. However, if you plan to use the ISA interface, keep in mind the extra voltages may be required. Indeed, boards that need them may be damaged if they are missing.

8051 EMBEDDED CONTROLLERS With Lots of Extras!



We offer a full line of low cost 80C32 embedded controllers and software tools which are ideal for developing products, test fixtures and prototypes.

Features Include:

- Low power CMOS design
- Up to 60K of code space and up to 60K of data space
- 5 to 15 volt operation
- Small form factor (3.5" * 6.5") with prototyping area
- System diskette includes application notes
- Start at \$100

Available Options:

- Multifunction Board adds A/D, 24 I/O lines and more!
- BASIC-52 or Monitor/Debugger in EPROM,
- C Compiler \$100 or BASIC Compiler for \$300

Iota Systems, Inc.

POB 8987 • Incline Village, NV 89452
PH: 702-831-6302 • FAX: 702 831-4629

*Linking Microcontrollers.
Breaking Boundaries.*

The 9-Bit Solution

The Cimetrics Technology O-Bit Solution is a complete microcontroller network (μLAN) that supports the 8051, 68HC11, 80C186EB/EC, and many other popular processors. The 9-Bit Solution takes full advantage of microprocessor modes built in to microcontrollers. The 9-Bit Solution allows simple and inexpensive development of master/slave multidrop embedded controller networks.

- 8051, 68HC11, 80C186EB/EC compatible
- A full range of other processors supported
- Up to 250 nodes
- 16 Bit CRC error checking with sequence numbers
- Complete source code Included

**Link Your Product
With A Cimetrics
μLAN Today!
607.273.5715**

120 West State Street • Ithaca, New York 14850
Ph 607.273.5715 • Fx 607.273.5712

If you need the extra voltages and are tempted to use a cheap AT power supply, remember that they usually spec a minimum current draw-well above the few watts (and less than 0.1 W in sleep mode, see Figure 3) typically consumed by the CARDIO-in order to regulate properly. You'll either have to choose a more appropriate supply or add dummy loads.

Note that the CARD10 doesn't include a battery, so an external backup input for the RTC is provided. The lack of a battery may be less problematic than it seems—perhaps the time can be obtained from whatever the CARD10 is plugged into.

Similarly, power control is a two-way street between the CARD10 and EASI. Under BIOS control, the CARDIO outputs four SMOUT (System Manager OUT) signals that can be used to control power to the LCD, disk drives, and so forth. Meanwhile, external power management circuitry can call for reduced power consumption by asserting an EXTSMI [External System Management

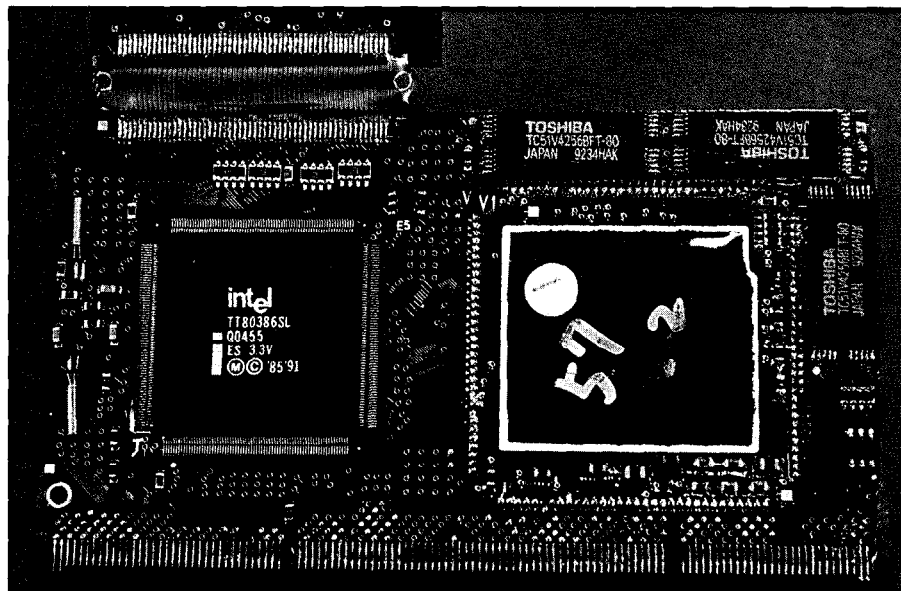


Photo 2—As an example of different packaging techniques used such as bare dice, MCMs, and TSOP, take a look under the CARDIO hood. The other side looks very similar.

Interrupt) input. Other power-control-related inputs include POWERGOOD, BATWRN*, BATDEAD*, and SRBTN* (Suspend/Resume Button).

Permutations of the CARDIO are available with 16/20-MHz CPU, 1–4 MB of RAM, and 128K–256K of Flash

memory. The 128K BIOS, thanks to Flash, is easily updated and the 256K Flash version offers extra space for “personalization” of the card with user-specific information such as ID or passwords. This could be important given that “portable” also translates

STOP!
LOOK!
LISTEN!

Odds are that some time during the day you will stop for a traffic signal, look at a message display or listen to a recorded announcement controlled by a Micromint RTC180. We've shipped thousands of RTC180s to OEMs. Check out why they chose the RTC180 by calling us for a data sheet and price list now.

MICROMINT, INC.
4 Park Street, Vernon, CT 06066
(203) 871-6170 • Fax (203) 872-2204

CALL 1-800-635-3355

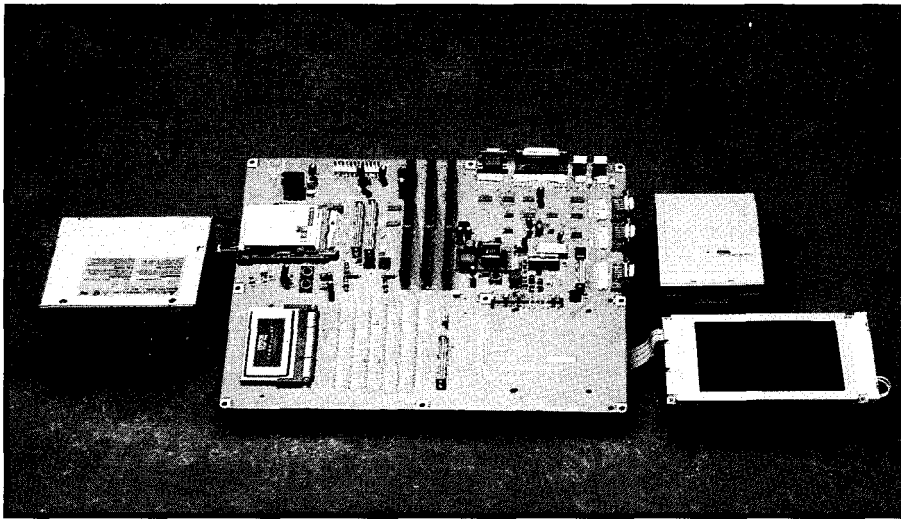


Photo 3—The CARD10 gets lost in the middle of the EVA Kit Evaluation Board, intended for development purposes.

into “losable” and “stealable.” However, accessing the Flash is as easy as hardware (i.e., via the EASI bus connector), so it’s not the place to keep any real secrets.

MOTHER OF ALL BOARDS

For a glimpse of what a single-chip, PC-based motherboard of tom-

orrow might look like, check out the CARD10 EVA Kit Evaluation Board (P/N D105300, \$1000) in Photo 3.

Yeah, the CARD10 looks pretty lonely amidst the connectors, disk drives, and otherwise sparsely populated real estate. Keep in mind the EVA board’s purpose is only to evaluate and prototype CARD10

applications that will presumably take better advantage of its small size.

For convenience, the EVA board makes the EASI bus signals accessible for probing, an otherwise tedious and intricate task. Desktop-compatible connectors are offered for CGA/VGA, two serial ports, a parallel port, mouse, keyboard, and three ISA slots.

Provision is made for a variety of power schemes including single DC supply, AC/DC converter, or PC/AT-compatible plug-in. The inverter required for the LCD backlight is also provided.

While a wide variety of floppy and hard disks are connectable, dedicated space is provided for a 1.44-MB, 3.5” floppy and a 40-MB, 1.8” IDE hard disk. They, along with a 6” 640x480 mono backlit LCD, are offered by Epson as a EVA Peripheral Units kit (P/N D105301, \$1700).

Epson’s done their part and, especially the manufacturing folks, deserve a hand for coming up with the worlds smallest PC. Now it’s up to you entrepreneurial integrators out there to figure out just what “portable equipment” and “other devices” make a good home for CARDIO. □

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.

Real-Time Multitasking with DOS for Microsoff C, Borland C, Borland/Turbo Pascal

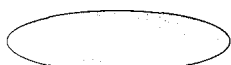
Develop Real-Time Multitasking Applications under MS-DOS with RTKernel!

RTKernel is a professional, high-performance real-time multitasking kernel. It runs under MS-DOS or in ROM and supports Microsoft C, Borland C, Borland/Turbo Pascal, and Stony Brook Pascal+. **RTKernel** is a library you can link to your application. It lets you run several C functions or Pascal procedures as parallel tasks. **RTKernel** offers the following advanced features:

- pre-emptive, event/interrupt-driven scheduling
- number of tasks only limited by available RAM
- task-switch time of approx. 6 μ s (33-MHz 486)
- performance is independent of the number of tasks
- use up to 64 priorities to control your tasks
- priorities changeable at run-time
- time-slicing can be activated
- programmable timer interrupt rate (0.1 to 55 ms)
- high-resolution timer for time measurement (1 μ s)
- activate or suspend tasks out of interrupt handlers
- programmable interrupt priorities
- semaphores, mailboxes, and message-passing
- keyboard, hard disk, and floppy disk idle times usable by other tasks
- interrupt handlers for keyboard, COM ports, and network interrupts included with source code
- supports up to 36 COM ports (DigitBoard, Hostess boards)
- supports protocols XOn/XOff, DTR/DSR, RTS/CTS
- full support of NS16550 UART chip
- supports math coprocessor and emulator
- fast, inter-network communication using Novell’s IPX
- runs under MS-DOS 3.0 to 6.x, DR-DOS, LANs, or without operating system
- DOS calls from several tasks without re-entrance problems
- supports resident multi-tasking applications (TSRs)
- runs Windows or DOS Extenders as a task
- supports CodeView and Turbo Debugger
- Kernel Tracer for easy debugging
- ROMable
- full source code available
- no run-time royalties
- free technical support by phone or fax

RTKernel-C 4.0 \$495 **RTKernel-Pascal** 4.0 \$445
C Source Code: add \$445 Pascal Source Code: add \$375

International orders: add \$30 shipping and handling.
Mastercard, Visa, check, bank transfer accepted.



On Time MARKETING

Professional Programming Tools

In North America, please contact:
On Time Marketing
88 Christian Avenue • Setauket, NY 11733 • USA
Phone (516) 689-6654 • Fax (516) 689-1172
CompuServe 73313.3177

Outside North America, please contact:
On Time Marketing
Karolinenstrasse 32 • 20357 Homburg • GERMANY
Phone +49 - 40 - 43 74 72 • Fax +49 - 40 - 43 51 96
CompuServe 100140.633

CONTACT

Epson America, Inc.
20770 Madrona Ave.
Torrance, CA 90509-2842
(310) 787-6300
Fax: (310) 782-5350

S-MOS Systems
2460 N. First St.
San Jose, CA 95131-1002
(408) 922-0200
Fax: (408) 922-0238

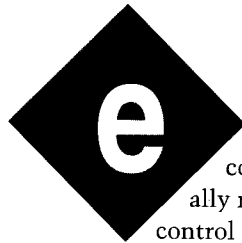
IRS

419 Very Useful
420 Moderately Useful
421 Not Useful

Embedded Programs

EMBEDDED TECHNIQUES

John Dybowski



Embedded computers generally run under the control of stored programs. Most often these programs are stored in EPROMs where they are, for all practical purposes, permanent. Although burning programs into these "stone tablets" is pretty much the standard mode of operation, there are several issues associated with this practice.

For anyone charged with maintaining a battery of embedded instruments, the primary concerns revolve around the amount of time and effort involved in changing EPROMs if, or when, the need arises. Even if you've specified the application accurately and your code is perfect in every way, it's not unreasonable to consider that the application itself may change over time. I'll be the first to admit that there are a great many applications that can be served with a fixed and unchanging feature set. Then again, if you're designing anything that people interact with, odds are system requirements will grow and evolve as users' needs and expectations change. Anyway, whether your program update is planned or unplanned, and whether it's to add features or to fix bugs, it quickly becomes apparent that wrestling with EPROMs is less than optimal for promoting happy customers.

PLIABLEWARE


If you don't want to rip your system apart every time a code change is required, then you need to store your program in something other than a conventional EPROM. You would be better off with something that is

disposed to in-circuit programming. Remember that, by their very nature, embedded computers usually get crammed into out-of-the-way places and often exist in large numbers. These are two good reasons why changing chips tends to be a costly and time-consuming affair. Such hidden expenses are becoming an increasingly important concern as evidenced by more and more customers that are willing to pay extra up front to avoid such entanglements. To satisfy these changing product needs, chip manufacturers are supplying a wide variety of alternate memory options.

Flash technology is perhaps one of the most familiar and is rapidly gaining favor with many engineers. Although differing in detail, the closely related E²PROM devices are also useful in many of the same applications. With the right support circuitry, even battery-backed RAM can be used to satisfy these same purposes. But, regardless of the technology used, it is advisable to make sure an appropriate write protection technique can be applied. It goes without saying that for an embedded computer to erase its executable program would be catastrophic...and inexcusable.

Early E²PROMs addressed this problem somewhat indirectly by requiring various weird voltages to enable the programming function. This "security" feature carried over to Flash devices as many of these parts still require a "high voltage" programming supply. The arrival of Flash and E²PROM devices capable of programming using just the standard logic power supply resulted in a reliance on software-based write-protection algorithms.

Updating program memory in-circuit is, in many applications, the way to go. This can be done using a purely hardware approach where you directly seize control of the memory device by tapping onto its pins using some sort of special programming connector. An alternative, and more attractive, method would allow doing the programming operation under firmware control. Obviously with this approach you'd have to keep at least a



All embedded controllers must have

their main program in some sort of nonvolatile memory, whether it be EPROM, EEPROM, Flash memory, or battery-back RAM. Which you choose depends on the application.

little code on-line so you can acquire and load the new program from the system controller. One way of doing this is to use a small dedicated EPROM that holds your program loader and communications handler.

Another way you could set up your system would be to use an E²PROM where your download and communications utilities could be kept in a separate, out of the way, part of the chip. Apart from the technological differences, one of the more apparent dissimilarities between an E²PROM and a pure Flash device is that the E²PROM can be erased and programmed on a byte-by-byte basis whereas the Flash part must be completely erased before it can be reprogrammed. In this respect it functions just like a standard EPROM

except that the erasure is invoked electrically instead of by using ultra-violet light. Needless to say, having to clear the entire Flash device is bothersome since it mandates a separate storage device for the code that must be permanently maintained.

In response to this objection, sectored parts are now available which contain multiple virtual Flash devices in various groupings on one chip. Some of the so-called "boot block" devices have what essentially amounts to an EPROM sector that can only be programmed by applying a high programming voltage to the chip. That is, these "boot sectors" can only be programmed and erased using device programmers and cannot be altered once they are installed in the system. This provides the needed security for

critical low-level code and eliminates the need for a separate EPROM.

These developments illustrate the various types of nonvolatile storage and their associated stages of security. Even something as conceptually straightforward as nonvolatile memory actually exists in degrees. The distinctions between technologies tend to blur, however, and with the addition of a partitioned nonvolatile controller like the Dallas DS1610, even a static RAM chip can take on some of the attributes of E²PROM and Flash parts. Generally speaking, all of these parts are capable of doing the same thing. And as so often happens in electrical engineering, the details actually determine which technology is appropriate (or most appropriate) for a given task.

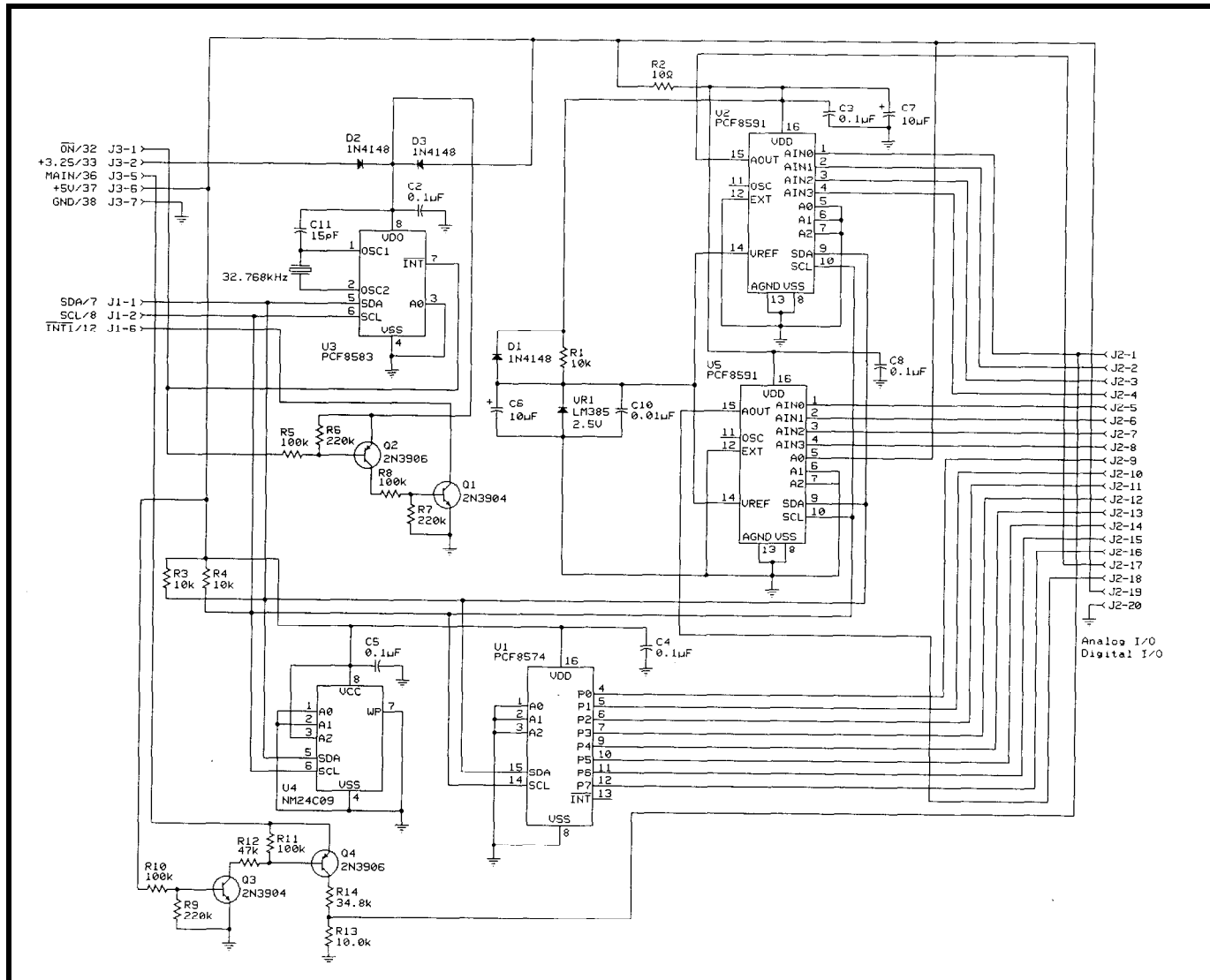


Figure 1—All circuitry including the PCF8583 RTC is located on the I²C card; however, back-up power for the RTC is located off-card

A PROCESSOR WITH PERSONALITY

The DS2250 is a controller that integrates a lithium-backed RAM. This RAM can be partitioned as either program memory or data memory. Partitioning allows you to arrange the built-in memory in a variety of ways and even lets you use it in conjunction with external EPROMs and RAMs as you would with a standard 8031. Most implementations, however, do not use any external memory components since a full-blown system with program and data memory can be configured using just the DS2250. Not only does this make for a compact system in itself, but you retain all your bit-addressable I/O. This saves you parts and eliminates headaches since the on-chip I/O bits are flexible and easy to use.

Most controllers have no personality whatsoever until an executable program is placed in their code store. The situation is different with the DS2250, and you can carry on an intelligible dialog with the DS2250 before any program is transferred to the part whatsoever. The initial implanting of code can be performed with a standard parallel programming algorithm and using any 87C51 programming equipment. The DS2250 tolerates the programming voltages and understands the programming algorithms that these programmers use. Or it can be programmed directly from the host PC via the on-chip serial port with the aid of a built-in Loader ROM. This ROM is normally not accessible by the user, but by pulling RST high and $\overline{\text{PSEN}}$ low, the DS2250 can be placed into bootstrap mode. When the chip is in this mode, the Loader ROM takes control of the DS2250 and performs a number of useful functions besides the initial loading of the executable image. Once bootstrap mode is exited and the DS2250 is released for operation, the Loader ROM relinquishes control and once again becomes transparent to the user having no effect on the memory map.

The Loader ROM understands and responds to a number of commands over the serial link. With the addition

Listing 1—Support services for the PCF8583 FC RTC/timer.

```
#pragma LARGE CODE
#include "reg5000.h"

extern void RTC_IN(char length, char address, char *ptr);
extern void RTC_OUT(char length, char address, char *ptr);

/* Set the BCD RTC date and time and date is at *ptr in the
form: milliseconds, seconds, minutes, hours, date, month */
void SetRTC(char *ptr)
{
    unsigned char i;
    unsigned char DataBuffer[7];
    DataBuffer[0] = 0x80;
    for (i = 1; i <= 6; i++)
        DataBuffer[i] = *ptr++;
    RTC_OUT(7, 0, DataBuffer);
    DataBuffer[0] = 0x0;
    RTC_OUT(1, 0, DataBuffer);
    return;
}

/* Get BCD RTC date and time. Time and date are returned at *ptr
as: milliseconds, seconds, minutes, hours, date, month */
void GetRTC(char *ptr)
{
    RTC_IN(6, 1, ptr);
    return;
}

/* Set BCD RTC date and time alarm. Alarm time and date are at
*ptr in the form: seconds, minutes, hours, date, month */
void SetAlarm(char *ptr)
{
    unsigned char i;
    unsigned char DataBuffer[8];
    DataBuffer[0] = 0x4;
    RTC_OUT(1, 0, DataBuffer);
    DataBuffer[0] = 0xf0;
    for (i = 1; i <= 5; i++)
        DataBuffer[i] = *ptr++;
    RTC_OUT(6, 8, DataBuffer);
    return;
}

/* Set the RTC interval timer where interval = 000 (no timer),
001 (milliseconds), 010 (seconds), 011 (minutes), 100 (hours), or
101 (days), and count = bcd up-count value */
void SetTimer(char interval, char count)
{
    unsigned char DataBuffer[2];
    DataBuffer[0] = 0x4;
    RTC_OUT(1, 0, DataBuffer);
    DataBuffer[0] = count;
    DataBuffer[1] = (interval | 0x8);
    RTC_OUT(2, 7, DataBuffer);
    return;
}

/* Assembler linkage: Input a string over the I2C bus */
static void RTC_IN(char length, char address, char *ptr)
{
    extern void Rec_I2C_String (void);
    #define RBO ((char *) 0x10000L)
    RBO [0] = length;
    B = address;
    ACC = 0xa0;
    DPTR = ptr;
}
```

(continued)

Listing 1-continued

```
Rec_I2C_String();
return;

/* Assembler linkage: Output a string over the I2C bus */
static void RTC_OUT(char length, char address, char *ptr)

extern void Xmit_I2C_String(void);
#define RBO ((char *) 0x10000L)
RBO [0] = length;
B = address;
ACC = 0xa0;
DPTR = ptr;
Xmit_I2C_String();
return;
```

of a PC hosted Initial Program Loader (IPL), a higher-level command shell is placed above the DS2250's command interpreter and the entire configuration, download, and verification operation can be set up to proceed automatically.

When the IPL is initially invoked to download a program, it builds a configuration file that is subsequently used to configure the DS2250 during the download sequence. Naturally, the most important information contained in this configuration file is the partition information that dictates where program memory ends and data memory begins.

Although ultimately very useful, the DS2250's partitioning capability is somewhat primitive compared to newer Dallas controllers such as the DS2251 and DS2252. The inherent limitation with the DS2250 is that, although separate program and data regions may be defined, they must be contiguous. That is, data memory must begin where program memory ends. Since the DS2250 only supports up to a maximum of 64K, this isn't really much of a problem. Certain applications could, however, benefit if it were possible to define overlapping program and data memories. In any case, once the program is loaded and the partition information is set, the program region is automatically protected. This protection results from the simple fact that the 8031 architecture provides no instructions that are capable of writing to program memory.

BRAIN TRANSPLANTS

Having the ability to download executables directly from a PC is a step in the right direction. However, if you're using the bootstrap loader for this purpose, you have to tolerate the Loader's particular data format and communications protocol (or lack thereof). In many cases, a more useful approach would allow program updating without the need for removing the unit from service. To accomplish downloading without unduly disrupting the installation, you would at least want to perform the operation within the confines of the system's native command syntax and communications protocol.

The designers of the DS2250 anticipated this need and provided the means of accomplishing a program download entirely under firmware control. Using the timed access method I described last month, you can gain access to the memory control register and define a new memory partition. This allows you to temporarily redefine program memory as data memory, which essentially grants you write access. Once you do this, data can be received from the system host in any format desired and can be written to data memory. On completion of the transfer, all that remains to be done is to partition the newly loaded region as program memory and begin executing the new program.

As a safety feature, when memory is partitioned under firmware control, the lowest 2K cannot be defined as

data memory. This restriction is intended for your own safety. Needless to say, it would not bode well if you switched out all of program memory. Naturally, you must ensure that the program does not leave this 2K region until memory has been "normally" partitioned or you'll terminate your download real fast.

TWO BIT I/O

Last month I described how to connect to remotely located LCDs, keypads, and other general-purpose I/O devices using two wires over the I²C bus. The ec.25 uses these very same two wires for communicating with a number of local peripherals as well. Although the I²C card provides the bulk of the system I/O (both digital and analog), it also handles the system timekeeping and timing and provides two separate nonvolatile areas using RAM and E²PROM devices.

This card accommodates eight analog inputs and two analog outputs using two PCF8591s. A PCF8574 handles the digital I/O and provides eight bidirectional bits of digital I/O. Nonvolatile storage is contained in a 24C04 5 12-byte E²PROM. A PCF8583 contains a clock/calendar and a fairly elaborate timer subsystem as well as 256 bytes of battery-backed RAM. It is through the use of the timing facility contained in the PCF8583 that the ec.25 achieves its capacity for very long battery run times due to its capability for intermittent operation.

The schematic in Figure 1 shows the circuitry contained on the I²C card. U3, the PCF8583 RTC, being the only battery-backed peripheral on the card, gets its backup power from a source that is located off-card. Using an off-card backup source allows you to select the nonvolatile backup power at the system level. And having centralized backup power can lower costs, especially if you pick up a bunch of functions that ultimately need to have nonvolatile capabilities. This doesn't mean critical system peripherals such as large capacity RAM cards can't have local backup batteries, simply that it's optional.

I showed you last month how the power manager stepped down the

primary power using a regulator dedicated to providing backup power at 3.25 V. In an alternate configuration you may find yourself not needing to manage battery power at all and the system might be configured with a supply card that is optimized for line-powered operation. Naturally, if you don't have constant power in the first place, then the scheme I just outlined is of no use. The obvious solution would be to provide a centrally located backup power source such as a battery. Depending on the system's functional requirements and operational parameters, the backup source could be one of many different battery chemistries, either primary or secondary, or you could forego the battery and go with something such as a Supercap.

The system provides backup power at a nominal 3.25 V. You'd most likely want to operate somewhere around this level regardless of the backup method that you used. This backup power is combined with the main logic supply using mixing diodes (D2 and D3) fed into the V_{DD} pin of U3. U3's interrupt pin (\backslash INT) can be driven by the RTC alarm or the timer alarm. As such, it can respond to a clock/calendar alarm and can function equally well as a fully programmable interval timer. Because of these capabilities, the PCF8583 proves to be well suited for providing the types of signaling necessary to control the ec.25's power sequencing.

As configured in the system, \backslash INT connects directly to the power control circuitry located on the power manager card. The signals generated on \backslash INT are controlled using firmware and can range from hundredths of a second to hundreds of days when running in timer mode. Due to this flexibility and the reasonably high repetition rates attainable, \backslash INT is buffered by Q1 and Q2 and brought out to the DS2250 for general timing purposes. This signal can either be polled by the DS2250 or can function as an interrupt source.

Although the DS2250 is available with an integral RTC, I'm sure you can see why I elected to forego that particular feature based on what the PCF8583 offers. Even if I didn't need all the capabilities of the PCF8583, I'd

Listing 2—Support services for the 24C04 I²C EEPROM.

```
#pragma LARGE_CODE
#include "reg5000.h"

extern void EEPROM_IN(char length, char address, char *ptr);
extern void EEPROM_OUT(char length, char address, char *ptr);

/* Write to the EEPROM, data is at *ptr. */
void WrEEPROM(char length, char address, char *ptr)
{
    EEPROM_OUT(length, address, ptr);
    return;
}

/* Read from the EEPROM, data is returned at *ptr.*/
void RdEEPROM(char length, char address, char *ptr)

    EEPROM_IN(length, address, ptr);
    return;

/* Assembler linkage: Input a string over the I2C bus */
static void EEPROM_IN(char length, char address, char *ptr)
{
    extern void Rec_I2C_String(void)
    #define RBO ((char *)0x10000L)
    RBO [0] = length;
    B = address;
    ACC = 0xa8;
    DPTR = ptr;
    Rec_I2C_String();
    return;
}

/* Assembler linkage: Output a string over the I2C bus */
static void EEPROM_OUT(char length, char address, char *XferPtr)
{
    extern void Xmit_I2C_String(void);
    #define RBO ((char *)0x10000L)
    RBO [0] = length;
    B = address;
    ACC = 0xa8;
    DPTR = XferPtr;
    Xmit_I2C_String ();
    return;
}
```

have to think twice before giving in to the kind of clock that you get with the DS2250. Curiously, if you elect to use the DS2250T's built-in RTC, what you get is a DS1215 which is a "serial phantom" type of clock. This device resides in the same address space as other memory components, thus the name phantom. Enabling this particular clock requires a 64-bit pattern matching sequence that must be performed serially. Although I've used phantom devices in general and the DS 12 15 in particular with decent results, I've never been especially fond of them; there's just too much overhead. When you're stuck for an RTC in an existing design, they're great for getting you out of a jam. On the other

hand, I find it perplexing that Dallas would actually incorporate something such as this into an original design. Maybe Dallas agrees with my opinion, since their newer microcontrollers (such as the DS225 1) have abandoned this questionable approach and they now use the DS1283 instead which is a "standard" byte-wide device.

Listing 1 shows the fundamental support package for the PCF8583. Although only tapping a fraction of the PCF8583's capabilities, the functions contained within this program provide the basic services you'd require of an RTC and interval timer. Functions are shown that set the RTC, read the RTC, set the clock/calendar alarm, and set the timer alarm. Hopefully the

main functions are pretty self explanatory, but I should say a few words about the low-level assembler linkage.

I've already stated my unwillingness to monkey with the assembly level I²C driver. This month I'm calling the I²C string transfer routines, which means I have a couple of new registers to load up. DPTR along with the other SFRs are defined in a special header file so I don't have to worry about them. RO and the other general-purpose registers, however, are not supported at the C level. Using a 24-bit pointer (RBO) explicitly defines register bank 0 as residing at location 0 and of residing in the data segment. When I want to set up RO, I end up moving the data byte indirectly to location 0 of RBO. Usually you work with your software products; sometimes you have to learn to work around them. If you're interested in looking at the PCF8583 from a couple of different angles, you might want to refer back to my columns in issues 35 and 42.

The 24C04 is a 4K-bit EEPROM. As is usual for parts of this type, data is organized in blocks of 256 bytes. The 24C04 has two such blocks. The 256-byte size is a result of the 8-bit addressing constraint inherent in the I²C implementation. If you have more than one block, essentially what you have to do to get around this restriction is handle each block as a sort of "virtual chip."

As you probably already know, all I²C peripherals have two components that make up their device address. One part of the address is fixed and is assigned on the basis of device type. There is also a programmable part that is selected by strapping address pins on the chip to either V_{CC} or V_{SS}. The ability to select the programmable part of the address is what allows you to have multiple chips that share the same base address on the same bus at the same time. What happens with the 24C04 is you give up one programmable address bit to accommodate the second 256-byte data block. Although A0 is tied to ground in the schematic [and you really must terminate it], it performs no address-related function. The 24C04 locates the first 256-byte block of data at its selected address

Listing 3—Support services for the PCF8.591 I²C ADC/DAC.

```
#pragma LARGE_CODE
#include "reg5000.h"

extern void AD_IN(char length, char address, char *ptr, char chip);
extern void AD_OUT(char length, char address, char *ptr, char chip);

unsigned char DacStatus[2];

/* Enable the selected DAC output */
void EnableDac(unsigned char channel)
{
    DacStatus[channel] = 0x40;
    return;
}

/* Disable the selected DAC output */
void DisableDac(unsigned char channel)
{
    DacStatus[channel] = 0;
    return;
}

/* Set selected DAC channel */
void SetDAC(unsigned char DacData, unsigned char channel)
{
    unsigned char chip;
    unsigned char DataBuffer[2];
    chip = channel;
    DataBuffer[0] = DacData;
    DataBuffer[1] = DacData;
    AD_OUT(2, DacStatus[channel], DataBuffer, chip);
}
```

(continued)

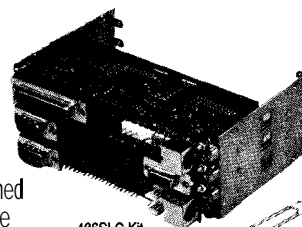
Good Things Come In Small Packages.

Extremely Small Packages.

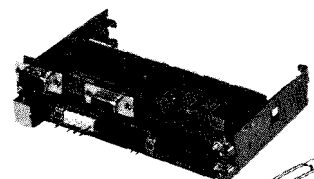
E.S.P. is a miniature, modular, PC/XT/AT product line designed specifically for power and/or space constrained applications. All E.S.P. modules are 1.7" x 5.2", and are ISA bus compatible. Available from Dovatron and other vendors, E.S.P. offers 8086, 386, and 486 processors, DC/DC power supplies, and a wide variety of I/O functions, including:

- *PCMCIA
- Private Eye
- *SCSI
- *Color TFT LCD
- ♦Flash
- ♦Voice Recognition
- *Ethernet
- AtoD
- D to A

Call! 800-848-1148



486SLC Kit
Measures 5.3" x 4.1" x 2"
Includes 4 expansion slots.



8680 XT Kit
Measures 5.3" x 4.1" x 1"
Includes 2 expansion slots.



DOVATRON
INTERNATIONAL

1198 Boston Avenue • Longmont, CO 80501 • 303-772-5933

and the second block at the selected address plus one. This might not sound like a big deal until you consider that some I²C²PROMs get pretty big. In fact, the limiting factor seems to be the number of blocks that can be assigned based on the three programmable address pins allotted for this purpose. The 24C16, for example, defines 16K bits arranged as 8 blocks of 256 bytes. In this case, none of the programmable address bits are significant. If all your system needs is a bunch of E²PROMs, then you're fine, but beware that other peripherals may also be using the same base address as the big E²PROM. Listing 2 shows a minimal driver for the 24C04 and other I²C²PROM devices.

The analog I/O is centered around two PCF8591 converters (U2 and U5). Each IC has four 5-bit analog inputs and one 8-bit analog output. All conversions are referenced to 2.5 V developed by R1 and VR1. Since the system is amenable to various configurations, battery power is monitored using channel 0 of the ADC rather than with a hardwired comparator.

The main power feed is derived from the 10-V preregulator when running from line power or from a battery when line power is not present. Using a PNP transistor (Q4) as a saturated switch, the main feed is attenuated through a divider composed of R13 and R14 and is presented to ADC channel 0. Q4 is switched on by Q3 only when V_{CC} is presented to the system in order to limit the battery's current drain and to safeguard the ADC when it is not powered.

Firmware support, shown in Listing 3, is provided for functions such as enabling and disabling the DACs, writing to the selected DAC, acquiring data from all ADCs, and acquiring data from a specified ADC. In order to conserve power, the DACs are defaulted off until they are turned on by program code. Since the DAC enable bit is transferred as part of the main control byte used for initiating any action in the analog subsection, a global variable containing the DAC enable mask bit is allocated for each channel. The enable and disable routines manipulate these mask bits.

Listing 3—continued

```

    return;
}
/* Get the ADC conversion for channels 0 through 7 */
void GetADCs(unsigned char *ptr)
{
    AD_IN(1, (0 | DacStatus[0]), ptr, 0);
    AD_IN(4, (0x5 | DacStatus[0]), ptr, 0);
    AD_IN(1, (0 | DacStatus[1]), ptr+4, 1);
    AD_IN(4, (0x5 | DacStatus[1]), ptr+4, 1);
    return;
}
/* Get the ADC conversion from the specified channel */
void GetADC(char *ptr, unsigned char channel)
{
    unsigned char chip;
    unsigned char address;
    unsigned char c = 2;
    if (channel < 4){
        chip = 0;
        address = (channel | DacStatus[0]);
    }
    else {
        chip = 1;
        address = ((channel-4) | DacStatus[1]);
    }
    while (c--) AD_IN(1, address, ptr, chip);
    return;
}
/* Assembler linkage: Input a string over the I2C bus */
static void AD_IN(char length, char address, char *ptr,
                 unsigned char chip)
{
    extern void Rec_I2C_String(void);
    #define RBO ((char *) 0x10000L)
    RBO [0] = length;
    B = address;
    if (chip) ACC = 0x92;
    else ACC = 0x90;
    DPTR = ptr;
    Rec_I2C_String();
    return;
}
/* Assembler linkage: Output a string over the I2C bus */
static void AD_OUT(char length, char command, char *ptr,
                  unsigned char chip)
{
    extern void Xmit_I2C_String(void);
    #define RBO ((char *) 0x10000L)
    RBO [0] = length;
    B = command;
    if (chip) ACC = 0x92;
    else ACC = 0x90;
    DPTR = ptr;
    Xmit_I2C_String();
    return;
}

```

Setting a DAC's analog output consists of selecting the channel and passing a binary value. Acquiring data for all eight ADC channels consists of nothing more than setting up a destination pointer and calling the ADC service routine. Most of the work is performed by the I²C driver since it

is capable of independently handling the string transfers over the I²C bus. Note that a dummy read is performed prior to the actual string transfer. This is because the PCF8591 uses the I²C clock as the conversion clock and, because of this, doesn't have the requested conversion data available

Listing 4—Support functions for handling digital I/O over the I²C based byte-wide expansion port.

```
#pragma LARGE_CODE
#include "reg5000.h"

extern void DO_OUT (char c);
extern char DI_IN (void);

unsigned char DoMask;

/* Input the value of the DI port */
char GetDi (void)

    unsigned char c;

    c = DI_IN ();
    return (c);

/* Output the value to the DO port */
OutDo (char DoData)
{
    DoMask = DoData;
    DO_OUT (DoData);
    return;

/* Set the specified bits on the DO port */
SetDo (char DoData)

    DoMask |= DoData;
    DO_OUT (DoMask);
    return;

/* Clear the specified bits on the DO port */
ClearDo (unsigned char DoData)
{
    DoMask &= (~ DoData);
    DO_OUT (DoMask);
    return;
}

/* Assembler linkage: Output a byte over I2C bus */
static void DO_OUT (unsigned char c)

    extern void Xmit_I2C_Byte (void);

    ACC = 0x40;
    B = c;
    Xmit_I2C_Byte ();
    return;

/* Assembler linkage: Input a byte over I2C bus */
static char DI_IN (void)
{
    unsigned char c;
    extern void Rec_I2C_Byte (void);

    ACC = 0x40;
    Rec_I2C_Byte ();
    c = B;
    return (c);
```

until the current transfer is completed. That is, data moved on the initial transfer is always from the previous conversion and it is during this interval that the specified conversion takes place. Getting data from an ADC

channel follows the same general procedure except that the chip and channel are isolated before the transfer is initiated. As before, two transfers are required to ensure that the selected channel is actually returned.

The PCF8574 handles the I²C-based bidirectional digital I/O. Listing 4 shows supported functions for reading from the port, writing to the port, set a bit (or bits), and clear a bit (or bits). You could just as well do these operations in-line rather than calling these support functions with their additional overhead. In some cases, though, it's a good idea to maintain control over even such seemingly trivial functions such as these. For instance, should I start accessing I/O from an interrupt level, it would behoove me to have a set of well-behaved routines that understand proper interrupt masking.

These drivers exemplify the need-to-do mode of operation. That is, they provide minimum levels of performance and functionality at a correspondingly minimal outlay of time and effort. They can always be tweaked if it turns out they are not up to the task. Surprisingly, it often turns out these tweaks are not required. This principle is closely related to the need-to-know principle which provides similar savings in time and energy.

Next month: BIONet. •J

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

SOURCES

For elements of this project, contact:

Mid-Tech Computing Devices
P.O. Box 218
Stafford Springs, CT 060750218
(203) 684-2442

Individual chips are available from

Pure Unobtainium
13 109 Old Creedmoor Rd.
Raleigh, NC 27613
Phone/fax: (919) 676-4525

IRS

422 Very Useful
423 Moderately Useful
424 Not Useful

CONNECT TIME

conducted by Ken Davidson

The Circuit Cellar BBS
300/1200/2400/9600/14.4k bps
24 hours/7 days a week
(203) 871-1 988-Four incoming lines
Internet Email: @circellar.com

How many times have you been trying to debug a program or circuit and said, "But that just can't happen." I/O ports that randomly change their own configuration is one of those times. The first thread takes a look at such an occurrence and some possible solutions.

Next, the analog-versus-digital issue rears its ugly head when we try to come up with the best way to monitor and control the brightness of a light. It turns out there are issues that apply equally to both methods.

Finally, everybody has seen "secret" circuits potted in an attempt to prevent unauthorized circuit snooping. In the last thread, we discuss some tactics for creating such modules, cover potential problems, and list some sources of raw materials.

Self-reconfiguring I/O ports

Msg#:34827

From: ELI WALTER To: ALL USERS

Would anybody have any idea on why the Motorola 68705P3S would change certain of its port pins from input to output at random? My software sets them to inputs and after running the micro for a time, I notice that certain pins were changed to output. Could noise coming from the input that is connected to the pins cause this? Any information on this would be appreciated. Thank you

Msg#:34832

From: GARY CORDELLI To: ELI WALTER

You got that about right. With a number of tiny micros you often see app notes that suggest you reset the direction register regularly to avoid this problem. Sounds cheesy, but that's from the manufacturer.

Msg#:35129

From: ELI WALTER To: GARY CORDELLI

I have never seen that in any data books. That is what I'm going to do. I'd like to find out why such a problem should happen.

Could you give me any clue if an op-amp like the LM324 or LM358 could burn out if the output is hooked to the micro's input? The op-amp is configured to act like a comparator, going high and low, and all of a sudden the micro reverts to output mode. When looking at the schematics of the op-amp, I notice a transistor output with the

emitter to ground and the collector to output. If this transistor is turned on and my micro is trying to source a high, wouldn't I have a direct short to ground, drawing excessive current? I'm destroying op-amps very randomly.

Msg#:35324

From: GARY CORDELLI To: ELI WALTER

Well, I'd be surprised if the micro outlasted the op-amp. I'd expect that any "excessive" current flow would destroy the micro's output drivers before the op-amp self-destructed. Most micros are not built for driving that much of a load directly-especially if they are sourcing current rather than sinking it.

Msg#:35487

From: JAMES MEYER To: ELI WALTER

There could be flip-flops used as registers to control the data direction that are located in the circuit and on the chip-very close to the output/input pins. If you get enough noise back in on these pins, then you *might* be able to override the processor's setting. These registers could be set once by the processor and perhaps it doesn't go back to see if something else like noise has reset them.

If you're seeing this problem, then resetting them at regular intervals is *one* solution. I'd strongly suggest, however, that you attack the problem at the source and take a look at reducing the noise at the pins with RFI filters or even small R/C networks.

The other way that data directions can be reset is by some code routine that is running wild and storing information, not where you *want* it to be stored, but where you *told* it to be stored. Check, recheck, and have someone else re-check your code. If you haven't already done so, run the code on a software simulator or emulator. If the simulator messes up, then you *know* who to blame. 8-)

Msg#:35717

From: ALAN COOK To: ELI WALTER

I had a similar problem with another Motorola micro in an automotive application. It helps if you add some filtering to the power and ground pins of the micro. I used a balun from Vcc to the Vcc pin with a 47- μ F and a 0.1- μ F cap from Vcc to ground, as close to the Vcc pin as possible. I also used the same arrangement in the ground circuit.

CONNECTIME

Msg#:37538

From: PELLERVO KASKINEN **To:** ELI WALTER

Maybe, just maybe, you are falling a victim of the infamous SCR effect. With few exceptions, all normal integrated circuits exhibit an inherent SCR structure that is triggered whenever any signal pin (input or output) is pushed beyond the power supply rail, either above V_+ or below V_- . This can be a simple transient, just microseconds long or even shorter, or it can be because the different power supplies powering two interconnected ICs come on or turn off at a different speed. Or you can have there some extra "noise filtering." Actually, I have bad experiences using *any* capacitor connected directly to a signal pin of an IC. For several years now, if I need such a filtering capacitor, I add another resistor between the capacitor and the IC pin.

To give you an idea what can happen, here is one scenario. We have an ordinary CMOS NAND gate such as 4011 that we want to use for a time delay. So we hook a 1-megohm resistor to one input and a 1-nF capacitor from that input to common, getting a 1-ms delay circuit. Now we power the thing up with a typical power supply response that is not monotonic. Chances are, whatever feeds our 1-meg resistor gets up with the initial power swing, charging the capacitor. Then we have the fallback transient of the power supply, before it comes truly on. During that time, the charge in the capacitor exceeds the power supply voltage, triggering the SCR behavior of the 4011. And now the beefy power supply comes really on and burns up our poor 4011.

The ICs have built-in protection and some data sheets give values for maximum current that the protection network can pass without triggering the SCR effect. Typical values are below 1 mA. Therefore, in a 5-V circuit, a series resistor of 5k or more from the capacitor to the signal pin will eliminate the destruction process covered above. Wanting to be safe, I normally use a 100k resistor in this situation.

If you have an op-amp output connected to a digital IC pin without a current-limiting resistor, you again have to pay attention to the power supply and the bypassing or even RC filtering of the op-amp power. That may just delay the op-amp power rise long enough so that the digital IC pin may feed a triggering charge. I emphasize that the normal CPU pins, even when uninitialized to outputs, could not overwhelm the output of the op-amp while things are up and running, but now the op-amp may not be up and running. The amount of current on these signal pins is not what does the actual destruction, it just triggers the SCR inside the op-amp and the power supply then does the rest. Again, a series resistor may be helpful.

Msg#:42773

From: ELI WALTER **To:** PELLERVO KASKINEN

When you suggest a series resistor between the op-amp and micro, are you saying on the op-amp output, or on the power supply to the op-amp. I do have a series resistor from the capacitor to the op-amp input.

Msg#:43765

From: PELLERVO KASKINEN **To:** ELI WALTER

I am definitely talking about the op-amp output terminal (i.e., between the op-amp and the CPU or other chip where I suspect you are now connecting directly). If you have already a resistor in that place, then maybe you would need a little higher value to limit the current to below the triggering threshold. Of course, all this is sort of speculation and there can be any of the ordinary glitch/transient situations, but you seem to not describe the power arrangements enough for me to get the full picture. Anyway, I just thought I would add something to the general or public knowledge about my experiences regarding these inherent SCR issues. If it is applicable to your case, good. If not, I hope you eventually find the actual culprit.

Bulb brightness control

Msg#:49867

From: JOHN MUCHOW **To:** ALL USERS

I haven't been able to find any info to help me out on a project and I was wondering if anyone might have some input on the best way to approach this.

I need to control the brightness of a DC halogen bulb (150 W, 24 VDC) very accurately ($\pm 1\%$), compensating for aging to hold the brightness steady, and have the ability to set the brightness to any one of several values within a narrow range (SO-100% full brightness).

I was considering the use of a CdS photocell on an ADC and a PIC16Cxx (or perhaps the 16C71, but I've never used that model) to acquire a brightness value and then outputting a value to a DAC that would vary the voltage on the ADJ pins of a pair of paralleled LM338 voltage regulators to change the voltage across the bulb. There would be some calibration involved, perhaps using EEPROM to store the value(s). The PIC would also respond to switch settings, or if I get *really* ambitious, an RS-232 message, to set the different brightness levels.

Might there be a better way of doing this? The accuracy/repeatability is the most important specification; cost is secondary (*sort * of <grin>). I hesitate to use DC PWM or phase-controlled AC because the brightness isn't constant and this bulb is being used to expose photographic

CONNECT TIME

film. I would sincerely appreciate any guidance you can give me.

Msg#:50547

From: JAMES MEYER To: JOHN MUCHOW

A microprocessor is overkill for your application. While you *could* duplicate all the functions that you need with a micro, all you really need to do is to put the lamp in a feedback loop with the sensor.

If you do it that way, you need to be aware of a few "gotchas." The sensor needs to have the same spectral response as the film that you're exposing. The feedback amplifier needs to have its response speed adjusted to match the thermal time constant of the bulb that you're using.

Use analog techniques. It will be a lot less trouble.

Msg#:50662

From: JOHN MUCHOW To: JAMES MEYER

I think I'm still going to need a microprocessor to set the different brightness levels needed even if I use an analog amplifier (digitally programmable gain?). The signal that's sent to my device to indicate which brightness level to select arrives via RS-232. With the machine we're modifying to use, we don't have any other choices.

I appreciate your list of "gotchas"! I was initially going to use a CdS cell, but have been warned that it can drift in value with time and temp. A silicon photodiode, looking at the 400-500-nm and 500-600-nm bandpass-filtered light that is exposing the film (it passes two different light sources), should do the trick.

If we have up to one second to set the light to our different brightnesses needed and the only other changes will be to compensate for bulb dimming as it ages (assuming we have a rock-steady power supply), does the response speed of the feedback amplifier and thermal time constant of the lamp still have to be matched? Or is it just good circuit design to do it anyway!

Msg#:50954

From: JAMES MEYER To: JOHN MUCHOW

If the electronic control is not at least approximately matched to the bulb, then you could get an oscillation in the bulb brightness. The average light value might be OK, but there could be some "flicker." There is also the possibility of having quite a bit of "overshoot" when you change brightness values. You will need to make the supply voltage somewhat greater than the normal operating voltage for the lamp, and that means with a large overshoot, you could shorten the bulb life. A properly "tuned" control loop will adjust the bulb voltage relatively slowly and still provide a good degree of stability.

BTW, whether the control loop is analog as I have suggested, or digital, the same loop stability parameters apply.

You will need to measure the light for either approach. I suggest you work on the measuring circuitry first. Try to get a large, linear signal from whatever you use to do the measurement. Then decide which approach you want to use for the control portion of the loop. Digital or analog should make little difference in the final outcome.

I suggested analog because the changes to the loop can be quickly made with trimpots and changing capacitors, rather than trying to analyze the parameters of a digital system and rewrite software for each change. Once the analog system is working, it should be easy to convert the system to a completely digital one. Analog is easier to develop and digital is the way to go for a commercial system that will be in volume production. It's cheaper that way.

Encapsulating electronics

Msg#:47878

From: RANDY JENKINS To: ALL USERS

I am trying to find information regarding the encapsulation of electronic components using an epoxy resin. I have tentatively selected 3M DP-270 as the epoxy I will be using mainly because it was recommended by 3M. However, I need to mold the epoxy in a square rectangular box around several components on a circuit card and need info on the mold material and sealing the mold against the circuit card. I have entertained the use of a machined polypropylene mold fence or possibly a metal mold fence coated with a Teflon material.

The purpose of the encapsulation is to provide a physical tamper-resistant barrier around the chips which contain encryption key information.

Any information on the matter would be greatly appreciated.

Msg#:48137

From: LYNDON WALKER To: RANDY JENKINS

A few years ago we did the same thing using a 2-part epoxy to encapsulate a 1.5"x0.5" PCB. Nothing fancy was used, just EL-cast epoxy. For potting molds we used flexible plastic containers and just peeled them off after the mold hardened. With a connector sticking out., the whole assembly looked like a black (licorice?) lollipop. The assembly was a hardware copy protection device for a desktop publishing system. We did a few thousand this way.

Problems:

CONNECT TIME

1) The unmolded epoxy tends to have `_sharp_edges_`—be sure your mold has rounded corners or be prepared to sand the edges.

2) Some “customers” simply cut, sanded, drilled, melted, or dissolved the epoxy and then simply copied the circuit to pirate the software—remember, the only consistent effect of copy protection is to annoy people and make them work harder to copy your software!

Msg#:48378

From: RANDY JENKINS To: LYNDON WALKER

Thanks for the info.. .

I am interested in the potting molds used for encapsulation. I need to encapsulate a rectangular portion of the top of the circuit card. How did you manufacture the potting molds and apply them to the circuit cards?

As the engineer responsible for this, I need to be able to understand the whole process. I will certainly get a phone call if it doesn't work!

Msg#:48412

From: LYNDON WALKER To: RANDY JENKINS

We didn't manufacture our molds. My boss at the time simply purchased a few thousand small plastic caps about 2" in diameter. They were used once then discarded.

Msg#:48776

From: DAN HOPPING To: RANDY JENKINS

Main company for encapsulating cups and packaging is

Plasmetex Industries, Inc.
1425 Linda Vista Dr.
San Marcos, CA 92069
(619) 744-8300

They should be able to direct you to the best encapsulant manufacture.

You may also want to try:

Ciba-Geigy: Ren Plastics Div.
49 17 Dawn Ave.
East Lansing, MI 48823
(517) 351-5900

Plasmetex sells the potting shells.

Here's another Idea.. .

If your unit will be visible at all, you can make a really fancy potting “dam” out of silicone rubber. By fancy I mean it can have an exact 3-D replica of your company logo and any patent or copyright info in raised letters. It's not really all that hard and makes a slick, professional-looking product.

Briefly, you get a rubber stamp made (backwards) that contains any raised information you want displayed on your potted product and mount that to a block of wood the size of your final potted device. Then you pour the special silicone rubber mold material around this master. In 24 hours you peel the mold away from the master and you have a mold for all your potted products. There are different durometers of the RTV silicone rubber available. If you are interested and have any questions, leave me a message and we can talk on the phone. The RTV silicone mold compound is available from:

Alumilite Corp.
225 Parsons St.
Kalamazoo, MI 49007
(616) 342-1259

Dow Corning
2200 W. Salzburg Rd.
Auburn, MI 486 11
(5 17) 496-4000

Msg#:50922

From: PELLERVO KASKINEN To: RANDY JENKINS

I can confirm the experiences other people responding to your message have had. The epoxy is handy when used in the disposable plastic caps or cups. In fact, we did a long time ago some of those, having the plastic cups vacuum formed from ordinary plastic sheet. After the mold was cured, we just tore the flimsy plastic cup to pieces when removing it. But you can get any number of off-the-shelf plastic cups or caps from the plastics sales outlets for pennies or from places like Ca-Plugs, possibly for even less.

Now, there is something I want to add to the story. I suggest you choose your epoxy with some `_filler_` material. The fillers add normally opaqueness to the product, which you probably would like in this case. But they also have a beneficial effect to the curing process. Epoxy without the fillers shrinks or expands during the curing and can damage some components with flimsy leads. Proper filler composition eliminates this danger. Then, I hope you do not need any sockets inside your mold. Solder connections are fine, but the uncured epoxy creeps along all surfaces and can open any socket or jumper header connection if there are such within reach.

As to the filler materials I prefer, use quartz sand filler and the guy who tries his pocket knife on your epoxy will have a surprise.

More about the history: We ended up needing to open those molded packages every now and then to replace a transistor. We found that it can be done with as simple a solvent as acetone, but it is slow! Speak of a couple of

CONNECT TIME

weeks! No wonder we changed over to permanent plastic cases with legs and filled it after the small board was in with silicone rubber. That could be carved open in a few minutes, though it took a while before we realized the way to get the block out of its casing was to drill a tiny hole in the bottom and then use shop air to "lubricate" the block/wall interface while pulling from the open end terminals.

Msg#:48037

From: DAVID TILLMAN To: RANDY JENKINS

Hmmm, interesting. This is how we did it once upon a time at a company I once worked for:

We used a fine sanding head on an electric drafting eraser to remove all the chip data (manufacturer, family, etc.). We then painted the tops of the chips that we had just sanded. Then we "potted" the really "Top Secret" circuits that we had a patent on in potting shells with epoxy. These circuits were mounted on individual 1"x½" boards and looked more or less like a DIP after potting. Then these were mounted on the "motherboard." The potting shells cost us pennies each and were easy to work with. We did somewhere between 8,000 and 10,000 of these.

Having said all that, let me say that it was a really lame idea. One day for kicks I dug out a Dremel tool and found that it was quite easy to get down to the pins on the chips. Also we got the epoxy mix wrong on a couple of batches. The internal temp on the units would reach 120°F. One day we started receiving units back from the customer with black goopy stuff leaking out of them.

Let me make it clear that this was *not* my idea. It was demanded by the senior "engineer."

Msg#:48376

From: RANDY JENKINS To: DAVID TILLMAN

Thanks for the response. I would like not to have to pot these components, but due to the nature of the customer and the business we are in, we are required to meet ANSI x3.24, and ANSI x3.92 (I believe these are the correct spec numbers). These specs tell us that we have to provide a tamper-resistant secure area for any data stored in the clear (unencrypted) and that if the physical security of the system is compromised, it must readily evident by inspection that the system is no longer secure and that the data may have been accessed. The potting works well for this last requirement because if it is ground off or chipped off, it will certainly be obvious to the casual observer.

Let me say that I am more worried about the nature of the chemicals involved in the potting process. The epoxy resin material seems to me to be a dangerous substance requiring a great deal of care in handling and usage.

Just one question: how did you manufacture the potting molds and how were they applied to the circuit cards? The

circuit card I have designed will be stamped and loaded in sheets of about 50 (they are pretty small) and I would like to pot the components on all 50 before they are snapped apart.

Msg#:48715

From: DAVID TILLMAN To: RANDY JENKINS

If you gotta, you gotta. We didn't manufacture the molds, we bought them from a supplier by the bag. Imagine the potting shell (mold) looking somewhat like a single hollow domino, open on one side. We filled the shell about halfway with the compound, inserted our circuit, and filled the shell the rest of the way. This left seven legs sticking out that were inserted into the motherboard and soldered.

The casual observer would have had no problem noticing that the modules had been tampered with. Something we used in conjunction with this was a tamper indicator that went by the brand name "Torque Seal." We applied it to the screw heads on the chassis. When it dries it is a dull brown color, but if it is chipped, broken, or cracked it shows a brilliant orange.

As far as handling precautions on the epoxy, the workers mixing it and filling the shells wore rubber gloves and worked in a room with two 48" exhaust fans mounted on the wall. I don't know if those were the proper precautions, it could be that they should have worn respirators.

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) 871-2988. Set your modem for 8 data bits, 1 stop bit, no parity, and 300, 2200, 2400, 9600, or 14.4k bps. For information on obtaining article software through the Internet, send Email to info@circellar.com.

ARTICLE SOFTWARE

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

To order Software on Disk, send check or money order to: The Computer Applications Journal, Software On Disk, P.O. Box 772, Vernon, CT 06066, or use your VISA or Mastercard and call (203) 875-2199. Be sure to specify the issue number of each disk you order. Please add \$3 for shipping outside the U.S.

425 Very Useful

426 Moderately Useful

427 Not Useful

STEVE'S OWN INK

The Market That Was Never Born Refuses to Die, and May Yet Live



What is happening in the robot market? Have you looked at it lately? It seems to be a hodge podge of dichotomies. I mean on the one hand you have the giants of industry dabbling around with their Pumas, and on the other hand you have a bunch of wild eyed candle burners creating an amazing array of interesting creations,

I respect the captains of industry that are pulling the rest of us along into the brave new world of tomorrow. They have forged ahead with implementing new manufacturing methods that have made better products, more productive workplaces, safer working environments, and still managed to survive the storms of labor unions. I think the executives who demanded that these large-scale robots be installed should be commended for their vision. And I hope that these brave experiments are rated on something other than the bottom line.

I suppose you can also consider the fantastic robots that NASA and other world space agencies launch. I salute the many engineers and scientists that in all likelihood put their family just a little lower in priority so these machines would be ready for launch. In some ways, I think if I had a chance to work on these machines, there would be many occasions when the janitor would be chasing me out.

But I think I reserve my warmest fondness for the lone tinkerers that hone their craft without billion-dollar budgets. In some way, these artistic individuals stand up defiantly in the face of everything and participate in this activity for little other reason than just the sheer joy of the activity. Every so often the BBS will flare up with robot aficionados and zealots. Their salvos of intellectual chatter are some of the threads I really enjoy reading and following. The range of their interests and the widely flung topics of these discussions cover just about every discipline that can exist in the world of electrical engineering. And they sometimes leave me leaning back in my chair with images of 10,000 untethered mechanizations of various shapes and sizes charging down the streets of some unsuspecting suburb.

I suppose I feel this way because it reminds me a little bit of my own roots. When I first became a devotee to computers and personal computers, there were a few really big companies selling really big systems to other really big companies. Then there were the rest of us. The guys with the breadboards, the salvaged teletypes and keyboards. The guys who would spend countless weekends wire wrapping that 4K memory board.

Yep, that's the connection. I see the big robot companies selling really big robots to other really big companies. And at the same time there is a group of people putting together their own creations out of salvaged or homemade mechanical components, experimenting with motors and moving parts, and putting some sort of programmable system in it to run the whole thing. All I can say is, keep the faith. You all have seen what happened to the computer market. Almost as if by magic, but we really know it was all "right time, right place." Still in retrospect, I don't know if any of us knew when the key moment arrived that made the industry transform into what it is today. Who knows? The same transformation could happen to personal robotics any day now.

