CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

DISTRIBUTED CONTROL

SPECIAL SECTION: Embedded Programming

Infrared People Tracker

Programming the MC68HC705

Software State Machines

April/May, 1992 — Issue #26

$3.95 U.S.
$4.95 Canada

EDITOR'S INK

Divide and Conquer

What's the best way to approach a large control or data collection application? You can hit it with a big hammer and use lots of computing power in a central location, or you can break the problem into smaller pieces and spread the load around.

When we designed the Circuit Cellar Home Control System II, we opted for the latter approach. We put the burden of deciding when what should happen on one central controller, but we distributed the tasks of power line interfacing, infrared interfacing, and individual input and output bits onto other processors spread out around the house.

I'd like to be able to say we purposely followed our Building Automation issue with a Distributed Control issue because of the HCS II, but I can't. It was a coincidence, but a nice one. It gives us the opportunity to point out the HCS II's distributed design after having spent an issue talking about its control capabilities.

THE ISSUE AT HAND

There are many off-the-shelf solutions to distributed control applications, and Jim Butler gives us a sampling of some of them. Jim has previously written for ***Circuit Cellar INK*** about using the nine-bit multiprocessor serial format supported by so many of today's microcontroller chips

In a similar vein, when you have numerous devices that support point-to-point serial communications (usually using RS-232), you often end up with a jumble of cables and switch boxes that all must be rerouted manually. Frank Cox shows us how to build an automated switch box using a conventional chip in a somewhat unconventional way.

Going back to the HCS II, Steve adds another link to the chain with his article about the IR-Link and its optional people tracking capability. Ed covers the firmware aspects of the module. I finish up my part of the project by describing the actual language used to program the system.

In our special section this issue, we talk about the somewhat ambiguous subject of Embedded Programming. Is that physically programming memory devices for embedded applications, or is it writing code for embedded applications? We couldn't decide, so picked articles that cover both areas

Jeff finishes up his description of the emerging memory card standards by designing a simple interface that can be used for storing collected data in harsh environments. Tom gets fuzzy again as we try to decide what he's talking about. And, filling in in Practical Algorithms, John Dybowski provides some hints for making programs that deal with nonvolatile memory more bulletproof.

The next issue should be dynamite as we look at Real-Time Programming in the feature section and Embedded Sensors and Storage in the special section.

Ken

CIRCUIT CELLAR INK®

THE COMPUTER APPLICATIONS JOURNAL

FOUNDER/EDITORIAL DIRECTOR
Steve Ciarcia

MANAGING **EDITOR**
Ken Davidson

~~ASSISTANT~~ ~~ASSOCIATE~~ EDITOR Stet.
Lisa Nadile

ENGINEERING STAFF
Jeff Bachiochi & Ed Nisley

CONTRIBUTING EDITORS
Tom Cantrell 8 Chris Ciarcia

NEW PRODUCTS **EDITOR**
Hat-v Weiner

ART DIRECTOR
Lisa Ferry

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

Cover Illustration by Robert Tinney

PUBLISHER
Daniel Rodrigues

PUBLISHER'S ASSISTANT
Susan McGill

CIRCULATION COORDINATOR
Rose Mansella

CIRCULATION ASSISTANT
Barbara Maleski

CIRCULATION CONSULTANT
Gregory Spitzfaden

BUSINESS MANAGER
Jeannette Walters

ADVERTISING COORDINATOR
Dan Gorsky

CIRCUIT CELLAR INK (ISSN 0896-8985) is published bimonthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (203) 875-2751. Second-class postage paid al Vernon, CT and additional offices. One-year (6 issues) subscription rate U.S.A. and possessions $17.95. Canada/Mexico $21.95, all other countries U2.95. Al subscription orders payable in U.S. funds only, via international postal money order or check drawn on U S. bank. Direct subscription orders to Circuit Cellar INK, Subscriptions, P.O. Box 3050-C, Southeastern, PA 19398 a cdl (215) 630.1914.

POSTMASTER: Please send address changes to Circuit Cellar INK. Circulation Dept., P.O. Box 30506, Southeastern, PA 19396

HAJAR ASSOCIATES **NATIONAL** ADVERTISING **REPRESENTATIVES**

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

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

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

MIDWEST
Nanette Traetow
(708) 7893080
Fax: (708) 7893082

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

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

INSIDE ISSUE 26

14 **Embedded Controller Networking Alternatives**
by James Butler

22 **Infrared Tracking and Remote** Control/Meet the New HCS II IR-Link
by Steve Ciarcia

34 **The Frugal Networker/A** Crosspoint Switchboard for RS-232
by Frank Cox

42 **Programming the Home Control System II**
by Ken Davidson

SPECIAL SECTION Embedded Programming

52 **State Machines in** Software/A Design Technique for Single-Chip Microprocessors
by Peter Hiscocks

62 **Programming the Motorola MC68HC705C8**
by Edward Oscarson

2 Editor's INK/Ken Davidson
Divide and Conquer

4 **Reader's** INK-Letters to the Editor

8 **New Product News**
edited by Harv Weiner

70 **Firmware** Furnace/Ed Nisley
infrared Home Control Gateway

78 **From the Bench/Jeff** Bachiochi
Does it Come With a Memory...Standard?/
Part 2-The Nitty-Gritty

84 **Silicon Update**/Tom Cantrell
Twenty Years of Micros-Now What?

92 **Practical Algorithms/John Dybowski**
Writing Code to Support Nonvolatile Memory

99 **ConnecTime**—Excerpts from the Circuit Cellar BBS
conducted by Ken Davidson

112 **Steve's Own INK/Steve Ciarcia**
A Night in The Life

97 **Advertiser's index**

READER'S INK

Battery Lore Cleared Up

I was just given a copy of ***The Computer Applications Journal***—quite a magazine! In the December 1991/January 1992 issue, "ConnecTime" contained several messages addressing camcorder batteries. Boy, what a mixture of misconceptions! Working in a technical support position here at Panasonic, I have to answer these questions day after day. Here are the key points for the proper care and feeding of your camcorder batteries.

1. Batteries should be charged following each and every use! Charge each battery for about 30 minutes past the point where the charge indicator goes off. No ***24-hour charges!*** "Memory" was a condition which supposedly affected NiCd batteries. Virtually all camcorders today use lead-acid "gel cell"-type batteries. "Memory" did have an effect back in the mid 1950s; NiCd technology has changed drastically since those days. Recently, several particles have appeared which have basically written off the entire issue. Battery capacity actually increased with short cycle discharge/charge cycles.

2. Never store your battery in a partially charged state. Remove the battery when not in use, as some camcorders may draw a small amount of power and thus discharge the battery over time.

3. Exercise the batteries; treat them as you would your car battery. If you anticipate periods of inactivity, pop a tape in and let the camcorder run a tape in the play mode. Recharge the battery immediately after each use!

4. Never discharge the battery to zero. Most camcorders will shut off at about 10.8 to 11 .O volts. This is as low as you want to go. NiCd batteries should never go below 1 volt per cell; going lower will risk cell reversal and ultimately render the battery useless! Your camcorder is the best battery recycler you have-there's no need to spend extra money on add-on accessories!

Bob Kozlarek, Secaucus, NJ

How Did She Do It?

I have one question after reading the story of Mr. Ciarcia's problem with locking himself out of the house [***The Computer Applications fournal,*** issue #25). How did Mrs. Picker get over the fence?

I also thought that for systems to be successful, t hey had to be user friendly. I can assume that Mr. Ciarcia would have the system activated while he slept. Did he consider awakening and an emergency run to the bathroom?

That could be more traumatic than forgetting a key. But then again, every design has to be tested "in the field."

Joe Privitier, Burbank, CA

We're talking about a man ***who cooks souffle in dark hooded clothing. We don't want to speculate on what his nighttime habits are or, for that matter, those of his neighbors.***

By the way, if you look more closely at the story, Mrs. Picker never actually climbed the fence or entered the backyard. Steve also tells us he's already taken midnight motion by house occupants into account ***in his system. Editor***

Readability Counts, Too

This is a letter that may interest many when it comes to producing readable code in any programming language. The first tip describes a method of formatting equations in programming languages that do not reformat what you type.

Once an equation is written, it usually becomes impossible to visualize it. I would like to suggest a method that I have used to ease the burden of visually interpreting (and getting the parentheses right when initially programmed) an equation.

The algebraic form of the equation to find the roots of the familiar quadratic equation $ax^2 + bx + c = 0$ is

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The usual programming style for one root is

```
X1 = (-1*b + SQR (B^2-4*a*c))/(2*a)
```

My programming style is

```
X1 = ( -b
         +
          SQR(
               B^2- 4 * a * c
             )
       ) / ( 2 * a )
```

Agreed, it uses a lot of white space, but it doesn't require a magnifying glass to decipher and mentally visualize the equation. Note the ease in which parentheses can be matched.

Another tip is in the use of mnemonic variable names. Many programmers do not describe what their short and usually cryptic variable names were derived from and what their type is. When you are reading someone else's program, or your own years later, it is essential to know the variable type and the rationale for choosing the name of the variable. I would like to suggest a comment section in alphabetical order included in the beginning of the module that contains the name, type, mnemonic, and what that mnemonic was derived from. The alphabetical list allows easy retrieval by the reader.

Example (in FORTRAN):

```
C CURTK  real    array (CURtain THicKness)
C CURLTH real    array (CURtain LengTH)
C INIDIA real    array (INItial DIAmeter)
C STAT   integer       (STATus of motor)
C                       0 - stopped
C                       1 - running
```

A final thought on comment lines. First there is the usual style of commenting which includes brief comments, the description of what the routine does, and the compiling options. Another style I haven't seen used very often can also be useful in some instances. Describe the current state of affairs at the instant the comment is read. In other words, what you have done, instead of what you are going to do. Information presented this way can sometimes make it easier for others to modify your code.

Example:

```
C The routine has now printed
C   PROMPT  (? UNITS )
C where  ? was null or the integer VALUE
C if  ALDEF was .FALSE.
```

Ron Dozier, Wilmington, DE

Want to Hear from You

readers are encouraged to write letters of praise, lemnation, or suggestion to the editors of The Computer lications Journal. Send them to:

Computer Applications Journal
ers to the Editor
rk Street
on, CT 06066

TIMELINE INC. Blowout Sale!

Liquid Crystal Displays
16 Characters x 1 line 3 for $25.⁰⁰
40 Characters x 2 line 2 for $25.⁰⁰
40 Characters x 4 line $25.⁰⁰
16 x 2 $12.⁰⁰ .... 32 x 4 $20.⁰⁰
20 x 2 $12.⁰⁰ .... 4x2 - $5.⁰⁰
24 Characters x 2 line LCD 12.⁰⁰
5V power required • Built in C-MOS LCD driver & controller • Easy "Microprocessor" interface • 98 ASCII character generator
Certain models are backlit, call for more info.

ADAPTEC 4070A (RLL) or 4000A (MFM)
SCSI Controller, your choice $60.⁰⁰

LASER DIODE
Sharp part#: LT022MC $5.⁰⁰
5mW at 780 nm, single transverse mode

IRMA BOARD 8 bit $99.⁰⁰
Links 3270 mainframe systems to IBM PC

MAGNETIC CARD READER $25.⁰⁰
Includes: • 20 character dot matrix display with full alpha-numeric capability • keypanel with full alpha-numeric entry • separate 7.5 VDC/0 5 Amp power supply • standard telephone interface extension cord • lithium battery and flat-cone speaker.

FLOPPY DRIVES
360 K $35.⁰⁰ 1.2 Mb $55.⁰⁰
720 K $35.⁰⁰ 1.44 Mb $60.⁰⁰
5.25' bracket for 720K or 1.44Mb drives $7 extra

640 x 200 LCD $29"
The unit is the display for the Toshiba T1100+ laptop computer. Driver board to generate text and graphics not included. This is a serial interface LCD. You should be very familiar with wiring when attempting to use this unit.
640 x 400 LCD $50.⁰⁰ Seiko part #:F642F

POWER SUPPLYS
380 WATT $29.00 Non-enclosed • Output: +5V @ 50 A +12V @ 6 A, -12V @ 1.5 A, Dimensions: 12" L x 5" W x 4"H
- - - - - - - - - - - - - - - - - - - -
73 WATT SWITCHING $19.00 or 2 for $30.00, (2) 4 pin power connectors attached • 115/230 Volt, Dimensions: 8.5" L x 4.5" W x 2" H • Output • +5V @ 2-9.75 A • +12V @ 0-1.5 A • -5V @ 0-0.4 A • -12V @ 0-0.5 A
- - - - - - - - - - - - - - - - - - - -
40 & 50 WATT, NON-ENCLOSED, 2 for $25.00

HeNe Laser Tube 10Mw (max. output) $99.⁰⁰

Proteon ProNet-4 Model p1347 Token Ring Board $99.⁰⁰
16 bit • 4 Mbps • IEEE 802.2 and 802.5 compatible • twisted pair • interoperable with IBM Token Ring network

The Spy In The Sky
1728 element CCD $19.00
4096 element CCD $29.00

DC 300 XLP and DC600 tape cartridges **$12.⁰⁰**

Tandy Model 100 computer (used) $199.00

101 Key XT Keyboard $24.W
85 Key XT/AT Switchable Keybd $24.00

HP Smart wand Bar Code Reader $99.⁰⁰
Model HBCR-8300

Portable **Micro Terminal** Wow

Flip up LCD display (9-16 VDC) • Can communicate with any computer having RS 232 port • Can communicate with another Microterminal • Use by itself as electronic notebook • Onboard microprocessor, data RAM (32K) and Video RAM (64K) • Complex built in diagnostics and set up capabilities. **$99.**

AMPRO LITTLE BOARD PC $79.
256K RAM included, expandable to 768K on board (256x9 SIMMS) • Floppy Controller 360K/720K/1.2Mb/1.44Mb • 2 serial ports and 1 parallel port • Keyboard input & Speaker Output • Has SCSI controller Socket • (requires optional pal & controller) • Board is the size of a 5.25" floppy drive

Single board 286-12 and - 16 computer $99.
16 bit, plugs into passive backplane. 1Mb of memory on board

Magnavox EGA Color Monitor
$150.00
.39 dot pitch. Model #:9CM053. EGA card available.

HARD DRIVES
10Mb Drive $50.⁰⁰
(Half Height, ST506 Compatible)
20Mb Hard Drive $100.⁰⁰
40Mb Hard Drive $180.⁰⁰
CDC 94166-141 $300.⁰⁰
(115Mb, Full Height, ESDI Interface.)
20 and 40 Mb IDE drives -- call for price!

Non-Enclosed TTL Monitors
Comer with pinout. 12V at 1.4 Amp input. Hwhor frequency 15 Khz. Ability to do 40 and 80 column
5" Black & White....$35.00
7 inch (sold out) $29.95
9 inch (Amber)........$29.95

Graphic a Alphanumeric
Liquid Crystal Display
480 Dot x 128 Dot, (80 x 16 line]
15.⁰⁰ each, or 2 for 20.9
Driver board available $60.00

CONTROLLER CARD BLOWOUT!
XT hard drive controller
Controls 2 hard drives $39.00
MFM controller-1/2 card-8 bit with cables $44.00
AT Hard Drive controller 16 bit
Part #WD1003-WAH - runs up to 12 Mhz only........$24.00

16 BIT SCSI CONTROLLER BOARD $99.⁰⁰
with floppy controller built on. Comes in kit form with cables & software. Mfg.: Sumo

8 0 3 8 7
These are - 16DX Mhz co-processors. Fully tested and guaranteed to work at the desired speeds below.
16Mhz 105.⁰⁰ • 20Mhz 115.' • **25Mhz** 125.'
256K x 9 SIMM modules, 120ns $7.50, 150ns $6.50

Novell Netware
up to 60% off list. Call with version and number of users you are looking for.
3Com - 3C603 - $175.00

1490 W. ARTESIA BLVD, GARDENA, CA 90247
Order desk only: USA: (800) 872-8878 CA: (800) 223-9977
L.A. & Technical Info: (310) 217-8912 • OEM INQUIRIES WELCOME
Fax: (310) 532-6304 • 20% Restocking fee for returned orders.

Minimum Order: $20.00 Minimum shipping handling charge $5.00. We accept cashiers cheks, MC or VISA. No personal checks or COD's. CA residents add 8.25% sales tax. We are not responsible for typographical errors. All merchandis subject to prior sale. Phone orders welcome. foreign orders require special handling. Prices subject to change without notice.

#102

NEW PRODUCT NEWS

Edited by Harv Weiner

SOLID-STATE PC FOR UNATTENDED APPLICATIONS

The **Solid-State PC** from INCAA Datacom is a compact, highly reliable device designed specifically for unattended and embedded applications, such as network functions, data acquisition, and control.

The Solid-State PC is designed around an AMD 286 processor featuring a clock speed of either 12 or 16 MHz and zero wait states. It features 5 12K of DRAM with parity, expandable to 4 MB, and provides hardware support for LIM EMS 4.0. A 128K flash EEPROM, expandable to 768K replaces the disk drive.

The Solid-State PC is fully compatible with the IBM AT bus, but has no conventional keyboard, screen, or disk drives. The disk drives are simulated by memory, and a backlit LCD screen plus two general-purpose button switches are provided for operational use, but can be disabled if not required. A special BIOS manages the solid-state facilities, so programs can access them as standard devices.

A control port and associated DOS software enable a conventional PC to be attached for software installation, development, and testing. You can then transfer files between the conventional disk of the PC and the solid-state memory. The screen and keyboard of the PC can be used as if they were peripherals of the solid-state unit. This feature also allows local or remote monitoring and control of the unit by an appropriate asynchronous device or PC.

In addition to the control port, a standard serial communication port (COM1) and a parallel port (LPT1) are supplemented by two extra serial I/O ports, which can operate synchronously or asynchronously. DMA channels assigned to these ports can provide very fast throughput.

The Solid-State PC features a ruggedized, completely closed aluminum housing measuring 2.6" x 8.9" x 11.2". The absence of a ventilator reduces noise and improves reliability while permitting use in environments normally considered unsuitable for a PC. One AT expansion slot is provided for a card up to 9.4 inches in length.

Optional hardware and software items include a math coprocessor, ROM DOS, protocol drivers, emulation packages, network adapters, and a communications coprocessor. The Sold-State PC starts at $1900.

INCAA Datacom, Inc.
233 Peachtree St. N.E., Ste **404**
Atlanta, GA 30303
(404) 523-1109 • Fax: (404) 522-7116

moo

LOW-COST SINGLE-BOARD COMPUTER

A new single-board computer, designed for use as a stand-alone system or as a user interface to an existing control system, has been announced by Applied Logic Engineering.

The RS-232 SBC is an 80C32-based design that includes an on-board 1 x 16 LCD and 20-key keypad to provide you with input and output capability. The board also includes a full RS-232 serial communication port for connections to other embedded control systems or PCs.

The board accommodates a 27C256 EPROM for program storage and a 32K static RAM for memory expansion beyond the internal RAM of the 80C32. Also, the system can be battery powered by a single 9-volt battery.

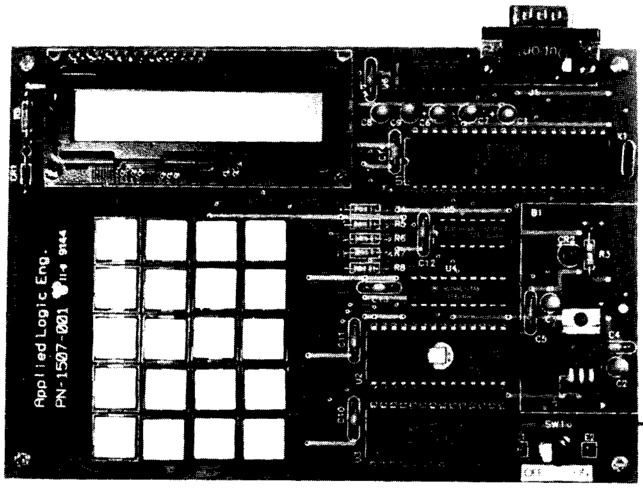
Software is provided on disk for interfacing with the keypad, LCD, and RS-232 communications. A demonstration program showing the capability of the board is included in an EPROM.

Uses for the RS-232 SBC include front ends for existing control systems, prototyping, data collection, testing, or educational purposes.

The RS-232 SBC is priced at $109.95.

Applied Logic Engineering
13008 93rd Place North
Maple Grove, **MN 55369**
(612) 494-3704
Fax: (612) 494-3704

#501



NEW PRODUCT NEWS

RS232 DATA COLLECTION DIRECTLY INTO PROGRAM

A program that collects data directly from any RS-232 serial line, modifies it, and sends it to the keyboard buffer without affecting normal keyboard functions is available from Labtronics Inc. EasyData is a TSR that "tricks" the foreground application program into thinking the data was manually entered from the keyboard. This process takes place transparently in the background without interfering with the foreground program.

Any RS-232 device with a regular data format can be used. EasyData is especially useful for electronic measuring devices, bar code readers, scales, data loggers, and portable data storage terminals. Automatically insert keyboard characters and ASCII codes (macros) before and after each data field. The effect of these macros is to press the same keys that would be pressed if the data were entered manually.

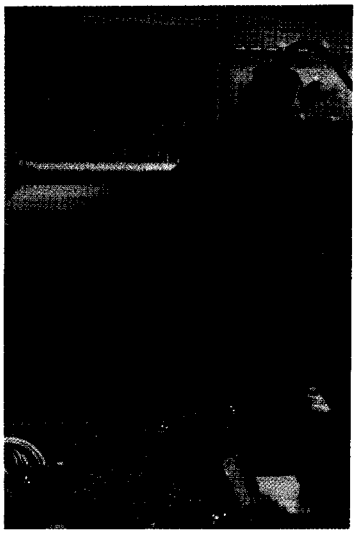
EasyData is compatible with any software that allows data entry through the keyboard. It uses standard PC serial ports as well as special serial cards. Communication is bidirectional, allowing commands to be sent to the instrument for control or to initiate data transfer. EasyData will also run under Windows.

Modify collected data before it is sent to the keyboard buffer or a file. EasyData can use multicharacter delimiters and filter out nonnumeric data. Data can also be imported directly from a file. The data will be parsed, modified, and macros can be added. The data can then be sent to either the keyboard buffer or another file.

EasyData is priced at $145.

Labtronics, Inc.
75 Rickson Ave.
Guelph, Ontario
Canada N1G 3B6
(519) 767-1061 **#502**

SPECTRUM ANALYZER IN A PROBE



The Model 107 Spectrum Probe from Smith Design converts a standard l-MHz triggered oscilloscope into a 100-MHz spectrum analyzer. The probe enables the scope to display logarithmic amplitude (vertical) versus frequency [horizontal) with a 60-dB dynamic range and a selectivity of 0.5 MHz. The scope controls can be used to provide full zoom facilities on either axis.

The Model 107 Spectrum Probe offers many of the capabilities of a spectrum analyzer, but at a fraction of the cost. Its usefulness is primarily as a quick diagnostic and general observation tool in application areas presently requiring use of bench-top spectrum analyzers and other instruments. Moreover, because the Probe is so small, you can actually "probe" and explore circuits where you might not do so with a full-fledged instrument.

In use, a wide variety of observations can be made. For example, quickly check amplifier stages for bandwidth, spot undesired resonances, adjust oscillators for fundamental or overtone operation, probe shielded enclosures or connector backshells for leakage, and test rotating machinery for internal arching.

The Model 107 features a 60-dB display range and a low-input capacity, rather than low-Z 50-ohm input. A 10-pF capacitor couples signals to its input, minimizing loading of circuits under test. Maximum input rating is 1000 VDC decreasing to 1 volt at 100 MHz. A 50-ohm coaxial input adapter is provided. A BNC connector connects the probe to a scope's vertical input.

Other specifications include a vertical logarithmic linearity within ±3 dB; a tangential sensitivity of 60 µV (±3 dB at 50 MHz), with flatness within ±2 dB from 5 to 100 MHz; and low-frequency degradation of about 8 dB at 1 MHz. Spurious responses are about -40 dB, with an IF bandwidth of 180 kHz at -3 dB. Horizontal linearity is specified to be within ±7%.

The Model 107 Spectrum Probe is priced at $249.

Smith Design
1324 Harris Rd.
Dresher, PA 19025
(215) **643-6340** • Fax: (215) 643-6340 **#503**

NEW PRODUCT NEWS

ECONOMICAL 8051 SINGLE-BOARD COMPUTER

A low-cost, 8051-based single-board computer designed for experimental use has been introduced by Suncoast Technologies. Completely assembled and tested, the 70691C computer board contains the popular 8051 microcontroller chip with its standard 128-byte internal memory. Also included is the circuitry for RS-232 serial communication between the 8051 and its host computer. The 70691 C easily connects to the host computer's serial interface using a standard four-conductor modular-type telephone line cord.

The board measures 3.875" x 4.5" and features a socket for a 2764 8K outboard EPROM. Fourteen programmable I/O ports (sixteen if RS-232 communication is not required) offer the you a low-cost computer engine. The breadboard area allows customizing the board for specific applications.

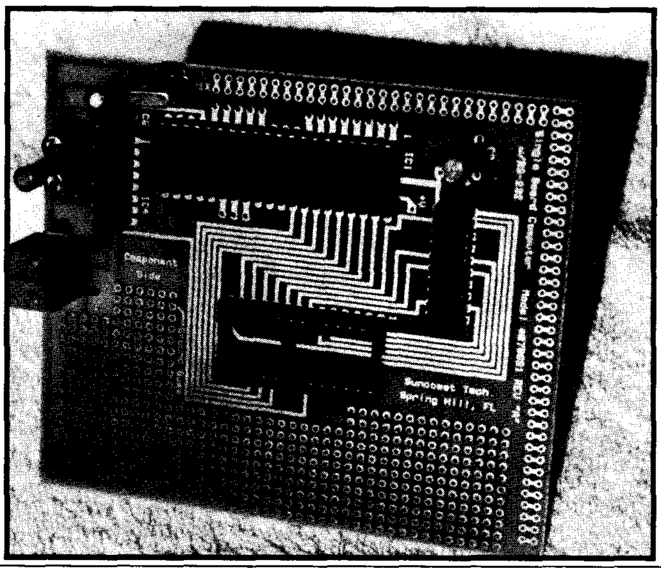
Operating from a standard 5-VDC source, the 70691C draws only 100 mA, allowing the use of the available prototyping area for the construction of a small power supply. An assembled and tested power supply is available.

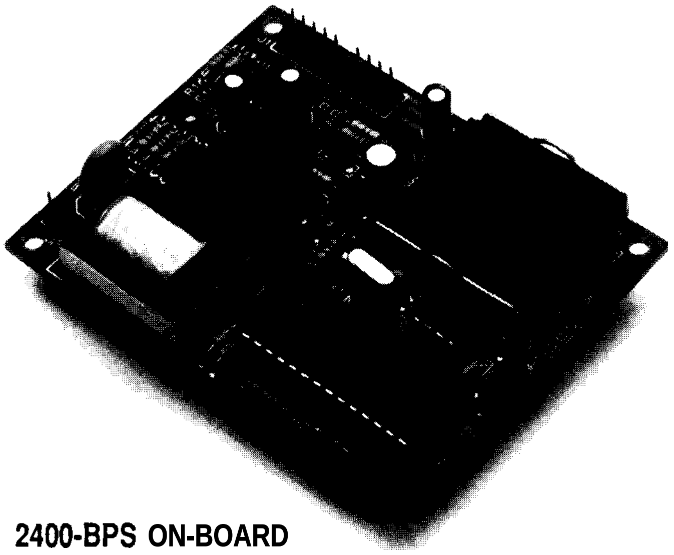
The 70691C single-board computer is priced at $38. The unit is available without the RS-232 interface circuitry for $34. Options include an interface cable ($4.95), and a 5-VDC power supply ($5.95 without 6-12-V transformer).

Software is available for writing, assembling, and converting 8051 programs, including a program editor, assembler, disassembler, and a hex-to-binary converter.

Suncoast Technologies
P.O. Box 5835
Spring Hill, FL 34606

#504





2400-BPS ON-BOARD MODEM

A high-speed modem designed for embedded applications has been introduced by Western DataCom. The 224 OEM is compatible at speeds from 300 to 2400 bps, is easy to interface to TTL-level serial ports, and is tunable to support the V.22 and V.21 international specifications. It features automatic adaptive equalization to achieve bit error rates of better than 10^{-5}, even over poor quality lines, and its speed and small size make it perfect for terminal applications.

The 224 OEM also features FCC-registered direct connect, DTMF tone or pulse dialing, compatibility with the "AT" command set, manual or automatic operation, storage of 32-digit phone number, and originate/answer operation. Control functions include Abort Timer Disconnect, Loss of Carrier Disconnect, Long Space Disconnect, and Failed Call Time-out.

Board placement and connection is simplified with its compact 3.5" x 2.75" size and power requirements of +5 volts at 180 mA (NMOS) or 30 mA [CMOS). The 224-OEM DTE interface is a standard 20-pin connector with standard TTL logic levels. A 2-pin 0.100" connector is provided for the telephone line interface. An audio output is provided if you wish to add an op-amp and speaker for aural monitoring of the call progress tones.

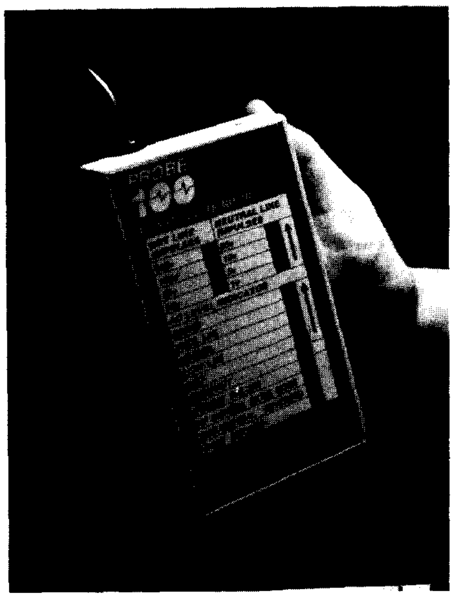
The 224-OEM Modem is priced at $179. Quantity discounts are available.

Western DataCom Co., Inc.
P.O. Box 45113
Westlake, OH 44145-0113
(216) 835-1510 • Fax: (216) 835-9146

#505

NEW PRODUCT NEWS

HAND-HELD POWER LINE MONITOR



Random computer problems such as software errors, system hang-ups, rebooting, and even hardware damage are often caused by noise on the AC power line. Eastern Time Designs has announced the availability of the Probe 100, a simple-to-use, easy-to-understand, toolbox-sized monitor that detects and displays power problems.

The Probe 100 detects a wide range of power disturbances including: spikes, sags, surges, common mode noise, dropouts, power failure, high-frequency noise, and wiring problems. It reports these disturbances on an easy-to-read LED display.

The Probe 100 is very simple to use. The unit's power cord is plugged into an outlet. It immediately identifies wiring problems such as hot and neutral wires reversed or an open ground. Then, leave the unit plugged in for 24 to 72 hours, checking the LEDs periodically. The manual gives a complete explanation of the disturbances it can detect.

The Probe 100 detects voltage impulses on the hot line from 20 to 500 V, and on the neutral line from 1 to 50 V. Impulses are measured from their location on the sine wave. The sensitivity to high-frequency noise on the hot line is 2 V peak-to-peak from 10 kHz to 10 MHZ.

The Probe 100 is priced at $149.95.

Eastern Time Designs, Inc.
2626 Brown Ave.
Manchester, NH 03103
(603) 645-6578
Fax: (603) 623-8930

#506

DC/CAD

introducing.. .

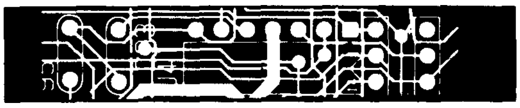
THE TERMINATOR

Super High Density Router
(Complete with Schematic & PCB EDITOR)
Features the following powerful algorithm & capability:

- **Rip - up and Retry**
- **Pre-routing** of SMT components
- Real-Time via **minimization**
- Real-Tie clean up passes
- **User defined strategies**
- **Window. 3.0 capability** as DOS Task
- 1-mil **Autoplacer** and **Autoparming**
- Two-way Gerber and DXF
- **Automatic** Ground Plane **w/** Cross-Hatching
- **Complete w/** Schematic **& Dolly** Libraries
- **Optional simulation** capability & protected mode for 386 users

Call for FREE DEMO DISK
price range: $295 - $1495

* PCB LAYOUT SERVICE AT LOW COST *
LEASE PROGRAM & SITE LICENSE AVAILABLE



Design
Computation

1771 State Highway 34
Farmingdale, NJ 07727
(908) 681 - 7700 • (908) 681 - 8733 (FAX)

" DC/CAD . . . The focal point of future CAD market "

04

IMAGE PROCESSING

Victor Library for C programmers

Image Processing: bright/contrast, sharpen, outline, resize, overlay, matrix conv, etc. TIFF/PCX/GIF/bin, use exten'd, expan'd, conv mem, images up to 4048 x 32768 grayscale, color, EGA/VGA Up to 1024x768x256, Lasejet, ScanJet+, for MSC, QuickC, Turbo C/C++, BC++. Includes free copy of ZIP & extensive examples. Source avail. No royal. $195.

ZIP Image Processing software

Bright/contrast, sharpen, outline, noise removal, embossing, matrix conv, etc. TIFF/PCX/GIF/EPS/bin. Up to 4048x4048, outstanding display and printing of grayscale images. EGA/VGA/super VGA, LaserJet, dot matrix. Versions for ImageWise/Idec/HRT/HPScanJet. Source avail.

ZIP Color-kit

Color reduction software, converts 24-bit TIFF and 16-, 24-, and 32-bit Targa images to EGA/VGA/super VGA. Lets any grayscale digitizer create color images. TIFF/PCX/GIF/TGA. $99.

Frame grabber

Capture 512x512x256 NTSC or PAL "live" video on VGA, frame averaging. With Victor and ZIP Image Processing. $499.

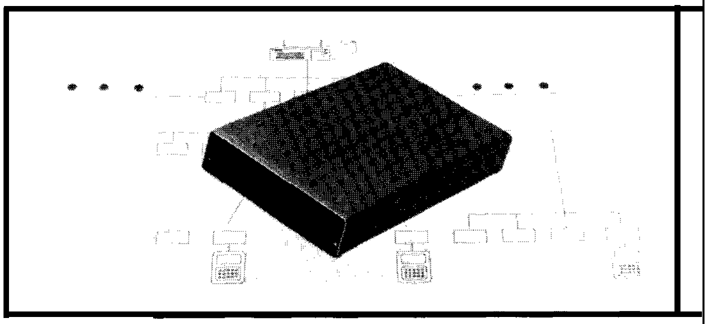
Catenary Systems
470 *Belleview St Louis MO 63119*

Call (314) 962-7833 to order
VISA/MC/COD

#105

NEW PRODUCT NEWS

REMOTE SERIAL COMMUNICATIONS CONTROLLER



The Sensor Modem 500 Series from Nota Bene Technology is designed to provide reliable data acquisition, data logging, or process point control from a remote location. The unit provides local RS-232/RS-485 and wide-area dedicated line, radio, or switched dial network access for communication with up to 800 digital or analog measurement and control points per station.

The Sensor Modem is FCC registered, 2400/1200-bps full duplex, and features autoanswering and call progress monitoring. DTMF and synthesized voice support used both locally and with telephone lines are optional.

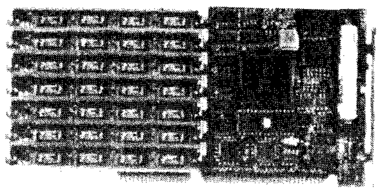
The Sensor Modem is designed around the HD64180 microprocessor and can address up to 1 MB of memory in the form of EPROM, static RAM, flash EEPROM, removable RAM cartridge with battery backup, and OTP PROM program or data modules. The 8.5" x 5.5" x 1.5" base unit requires 12 volts AC/DC at 90 mA. The Sensor Modem comes standard with ten isolated digital inputs, two relay or driver outputs, and four bipolar 12-bit analog inputs. Expansion I/O modules provide fully isolated I/O in increments of four or eight points, in any combination.

The price for the Sensor Modem base unit is $750. Expansion modules run **$150–$250**.

Nota Bene Technology, Inc.
11210 Arrowood Cir.
Dayton, MN 55327
(612) 421-9225 • Fax: (612) 421-9225

#507

ROMDISK™



High Performance Multimegabyte Disk Emulators

NEW MODELS / LOWER PRICES

- Floppy Drive and multimegabyte emulators for ISA bus computers
- 180K to 14 MB capacities
- EPROM, Flash or SRAM technologies
- Autobooting, Single or Dual disk emulation under PC or MS DOS
- List prices from $195

CURTIS, INC.
2837 No. **Fairview Ave. • St. Paul, MN 55113**
612/631-9512 FAX 612/631-9508
PC DOS is a trademark of IBM. MS DOS is a trademark of Microscoft

#106

OPTIMIZE YOUR MCU PROGRAM DEVELOPMENT !

INTEGRATE THE POWER OF
◆ **Editors** ◆ **Cross Rssemblers** ◆ **Disassemblers** ◆
◆ **Cross Compilers** ● **Data Conversion Utilities** ◆
◆ **Simulators** ◆ **Serial Communications** ◆

—— ARMADILLO +™ ——

R Unique, Universal Development / Communications Environment Supporting :
◆ All families of cross-assemblers and compilers.
◆ Communications with your target CPU.
◆ User definable utilities menu.
◆ Pull-down menus with mouse or keyboard control.
◆ IBM PC or compatible.
Now you **can EDIT. ASSEMBLE, UPLOAD, DEBUG,and MORE, all from mithin ORE, FAST,** ERSY-TO-USE **MENU DRIVEN ENVIRONMENT !**
$99.00 + $2.00 P/H
TO ORDER CALL (804) 479-3893
LIFE FORCE TECRROLOGY
5477 RUTLEDGE RD.. VR. BERCR. VR. 23464

#10

FEATURES

14 Embedded Controller Networking Alternatives

22 Infrared Tracking and Remote Control

34 The Frugal Networker

42 Programming the Home Control System II

FEATURE ARTICLE

James Butler

Embedded Controller Networking Alternatives

Distributed control systems rely on network communications to exchange data and keep processes coordinated. There are numerous networking options available; the best depends on your application.

Interest in embedded controller networking seems to be growing rapidly, perhaps because of the increasing number of ways an embedded controller network can be implemented. In this article, I describe several methods for adding networking capability to embedded controllers, including relevant hardware and software products and standards.

Embedded controller networks, like office LANs, allow embedded controllers to communicate with each other [and potentially with other types of computers). However, they differ in their application. Embedded controller networks are generally used for distributed control or data acquisition, whereas office LANs are used for such things as file sharing and electronic mail.

In a typical distributed control application, each embedded controller controls a piece of equipment. The network carries messages containing controller status information and high-level control commands. Frequently, the source of the control commands will be some central control station that constantly queries the status of the embedded controllers. This central controller may also allow a human operator to monitor and control the entire system.

Network nodes:	up to 250 (32 per cable segment)
OSI layers specified:	physical, data-link, application
Electrical specification:	RS-485
cable:	twisted-pair
length:	up to 1200 meters per cable segment
speed:	62.5 kbps, 375 kbps (up to 300 meters), 500 kbps to 2.4 Mbps (up to 30 meters)
Protocol:	BITBUS, based on IBM's SDLC protocol (open standard)
Transmission:	synchronous serial
Hierarchy:	master/slave
Media access:	controlled by master node
Error detection features:	message acknowledgement, 16-bit CRC, sequence count
Application layer features:	data-transfer, task control

Figure 1—*BITBUS is a serial control bus specification first developed by Intel in 1984.*

METHODS OF NETWORKING EMBEDDED CONTROLLERS

You can add networking capability to an embedded controller design using a variety of methods. Virtually all require additional hardware and many also require some software. These additions implement a network protocol.

The OSI seven-layer network model is often used in the description of network protocols. Most network protocols do not specify all seven OSI layers; for example, Ethernet specifies only the bottom two layers. Thus, some network protocols are composed of two or more protocols that specify different OSI layers. Simple networks may require the specification of only three layers, which are:

1. The physical layer defines the physical medium and how message bits are encoded. It is implemented in hardware.
2. The data link layer defines message construction, medium access control, and low-level error checking [among other things). It may be implemented in hardware, firmware, software, or some combination of these three.
3. The application layer specifies high-level network commands as well as the interface between the network software and application programs. It is usually implemented in software or firmware.

Methods for adding networking capability to an embedded controller can be categorized as follows:

1. Use a microcontroller with built-in networking hardware and firmware, and add physical medium interface circuitry.
2. Interface a LAN controller chip set to the embedded controller, and add software for the application layer [and perhaps other upper layers].
3. Make use of serial communication modes built-in to most microcontrollers, and add physical medium interface circuitry and network software.
4. Interface a serial communication IC to the embedded controller, and add physical medium interface circuitry and network software.

Methods 1 and 2 are hardware intensive: the network protocol is primarily or entirely implemented in hardware and firmware, simplifying software design. Methods 3 and 4 are more software intensive, allowing greater flexibility while keeping hardware costs to a minimum. Method 3 can often be used to add networking capability to existing embedded controllers. I will describe each of these methods in detail.

MICROCONTROLLERS WITH BUILT-IN NETWORKING HARDWARE AND FIRMWARE

The primary advantage of using a microcontroller with built-in networking hardware and firmware is a greatly simplified design of embedded controller hardware and software. Unfortunately, there are disadvantages. I am aware of only two microcontrollers with relatively complete built-in network protocols. Also, development tools for these microcontrollers are fairly expensive.

Intel's BITBUS solution uses the 8044 *BEM* microcontroller

BITBUS is a serial control bus specification developed by Intel (see Figure 1). It was first released in 1984, and has since gained some acceptance in industrial networking. BITBUS nodes generally use an Intel 8044 BEM microcontroller, which is essentially an 8051 integrated with a serial communication controller and firmware. Each node must also include an RS-485 transceiver because BITBUS specifies an RS-485 interface.

Intel offers a wide variety of development tools for BITBUS as well as distributed control modules. BITBUS cards for the PC are available, allowing the PC to be a BITBUS node, and you can also get board-level products from other companies.

Echelon's LON and the Neuron microcontroller

Echelon [Palo Alto, CA) introduced its Local Operating Network (LON) in 1990. Echelon suggests that its LONs (see Figure 2) could be used in a wide range of distributed control environments, such as automobiles,

Network nodes:	many
OSI layers specified:	all
Communication media:	twisted-pair (1.25 Mbps or 78 kbps). RF (4880 bps), power line (9600 bps), coax, IR, optical fiber
Protocol:	LonTalk (proprietary)
Transmission:	serial
Hierarchy:	none (peer to peer)
Media access:	predictive CSMA, optional CD, optional priority
Error detection features:	message acknowledgement (optional), 16-bit CRC, message ordering, duplicate detection
Application layer features:	network variables

Figure 2-The *Local Operating Network, or LON,* was *introduced by Echelon late in 1990.*

Network nodes:	up to **255**
OSI layers specified:	physical, data link
Communication media:	92-ohm coax, twisted pair, optical fiber (2.5 Mbps)
cable length:	up to 20,000 ft separation between farthest nodes using standard timings
Protocol:	**ARCnet** (open standard)
Transmission:	serial
Hierarchy:	none (peer to peer)
Media access:	token passing
Error detection features:	message acknowledgement, **16-bit** CRC

Figure 3—*ARCnet was originally introduced in* 1977 *by Datapoint Corp. and has since become* **a de** *facto standard*

factories, and buildings. For a more complete description of Echelon's LON, refer to Ken Davidson's article in ***Circuit Cellar INK.***

The microcontroller embodiment of the LON nodes are the Echelon-designed Neuron chips (manufactured by Toshiba and Motorola), which integrate a microcontroller and data communication hardware implementing a proprietary seven-layer network protocol (LonTalk). Transceivers handle the interface between Neurons and the communication media.

Neurons are programmed using Echelon's Neuron C. Echelon claims the features of Neuron C (including network variables), and other Lon-Builder development tools, make the development of distributed applications simpler, faster, and cheaper.

Development tools are available from Echelon, but the price may be a barrier to some; the LonBuilder starter kit costs $17,995 (lease options are available). Neuron chips cost about $10 each in large quantity, but the chip makers project the price will fall to about $5 in 1993. As of January, 1992, LonWorks Transceivers were not available. Echelon will be selling media interface modules or they will provide technical data from which you can design your own. LON interface boards for desktop and industrial PCs are also available.

LAN CONTROLLER CHIP SETS

Another way to add networking capability to an embedded controller is to include a LAN controller chip set. Chip sets are available that interface to a wide range of microprocessors and some microcontrollers.

This approach is advantageous because you can construct a moderate to high bandwidth network while using industry-standard protocols. The main disadvantage is the chip set is likely to increase the cost of your controller hardware significantly.

Another disadvantage is, regardless of the chip set you choose, you will probably have to write software that implements an application layer (high-level network commands) of your own design as well as having to write microcontroller code to talk to the chip set.

Datapoint's ARCnet

ARCnet was originally introduced in 1977 by Datapoint Corp., and has since become a de facto standard. Over 3,000,000 ARCnet nodes have been installed in everything from industrial LANs to office PC LANs.

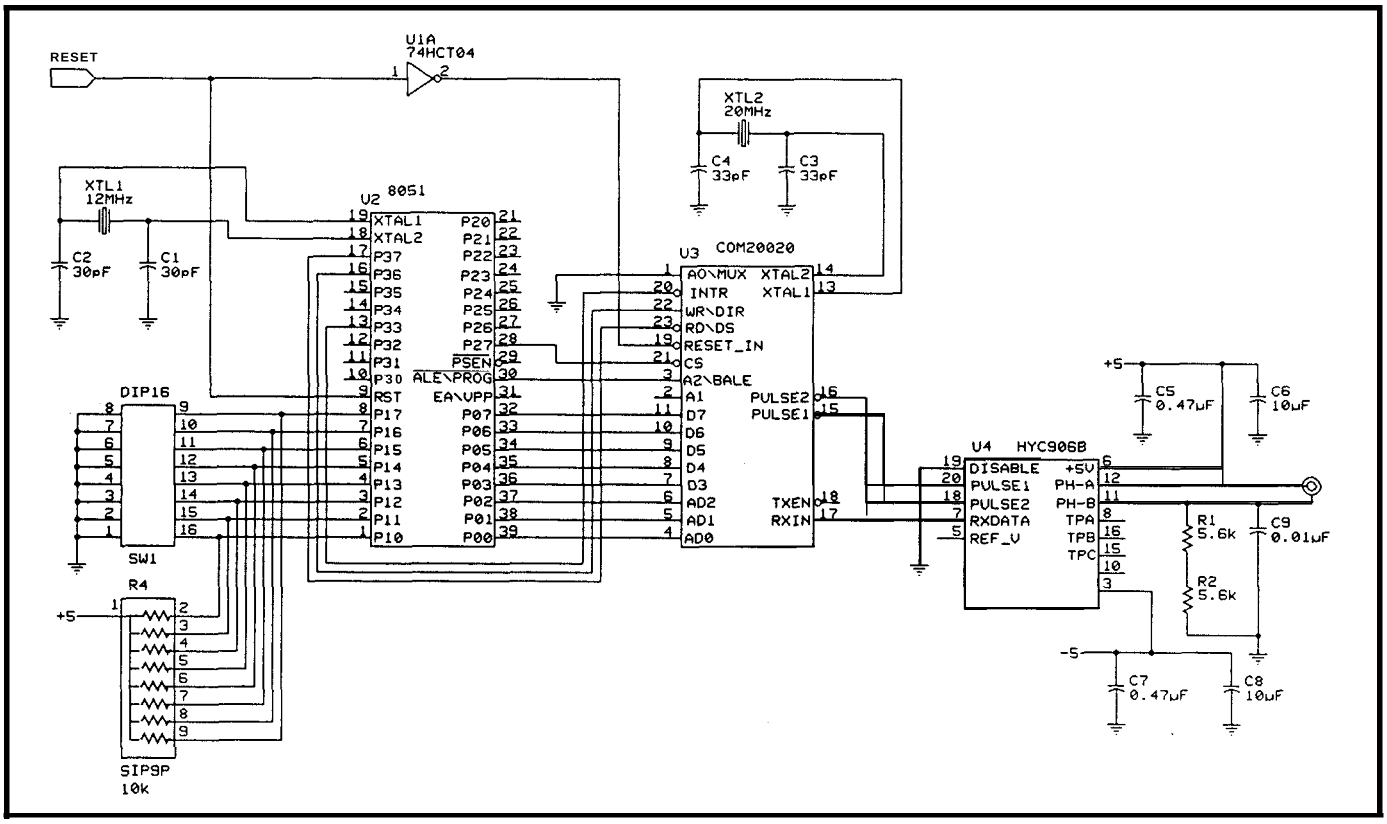

RESET
U1A 74HCT04
XTL1 12MHz
C2 30pF
C1 30pF
U2 8051
XTAL1 XTAL2 P37 P36 P35 P34 P33 P32 P31 P30 RST P17 P16 P15 P14 P13 P12 P11 P10
P20 P21 P22 P23 P24 P25 P26 P27 PSEN ALE\PROG EA\VPP P07 P06 P05 P04 P03 P02 P01 P00
DIP16
SW1
R4
+5
SIP9P 10k
XTL2 20MHz
C4 33pF
C3 33pF
U3 COM20020
A0\MUX INTR WR\DIR RD\DS RESET_IN CS A2\BALE A1 D7 D6 D5 D4 D3 AD2 AD1 AD0
XTAL2 XTAL1 PULSE2 PULSE1 TXEN RXIN
U4 HYC9068
DISABLE PULSE1 PULSE2 RXDATA REF_V
+5V PH-A PH-B TPA TPB TPC
C5 0.47µF
C6 10µF
R1 5.6k
R2 5.6k
C9 0.01µF
-5
C7 0.47µF
C8 10µF

Figure 4-The *Standard Microsystems COM20020 is an ARCnet controller chip designed to interface with the 68xx and 80xx families of microprocessors.*

Network nodes:	many
OSI layers specified:	physical, data link
Communication media:	50-ohm coax or twisted pair (10 Mbps)
cable length:	up to 1500 m separation between any two nodes on one network
Protocol:	Ethernet
Transmission:	serial
Hierarchy:	none (peer to peer)
Media access:	CSMA/CD
Error detection features:	message acknowledgement (optional), 16-bit CRC, sequence count

Figure 5—*Ethernet is perhaps the most widely used network solution at the data-link and physical layers for PCs and minicomputers.*

In the office LAN market, ARCnet competes with Ethernet. Although ARCnet has a data rate only one-fourth that of Ethernet, the performance of ARCnet can be comparable to or even better than Ethernet in certain situations. Of particular interest to embedded systems designers, each ARCnet node is guaranteed access to the network within a known length of time, and ARCnet can outperform Ethernet for short messages.

Adding ARCnet capability (see Figure 3) to an embedded controller design will typically require three packs: a controller IC, a media interface hybrid, and glue logic. The ARCnet controller will handle the transmission and reception of messages, but the software running on the microcontroller will be responsible for formatting messages to send and interpreting received messages.

These days, the most active marketer of silicon-implementing ARCnet appears to be Standard Microsystems Corporation (Hauppauge, NY). The Standard Microsystems COM20020 ($16.23 each in 1000 quantity) is a 24-pin ARCnet controller chip designed to interface with the Motorola 68xx and Intel 80xx families of processors [Figure 4 shows an example). The COM90C66 and the COM90C165 are good for Intel 80x86-based designs.

ARCnet interface cards are available for the PC, allowing you to put PCs and embedded controllers on the same ARCnet network. Vendors of ARCnet board-level products include Standard Microsystems and Ziatech (San Luis Obispo, CA).

Datapoint is working on ARCnet Plus, which promises higher speed (20 Mbps), longer messages (up to 4096 bytes), and more nodes (up to 2047), as well as downward compatibility with ARCnet.

Ethernet

Ethernet is perhaps the most widely used network solution at the data-link and physical layers for PCs and minicomputers (see Figure 5). Ethernet has also been promoted for factory automation by companies like DEC. The original Ethernet specification (developed by Xerox, Intel, and DEC) was used as the basis for IEEE standard 802.3, which has a slightly different message format.

Ethernet has the highest data rate of any of the solutions I describe (10 Mbps), but it may also be the most expensive solution for you. Ethernet performs best when transmitting long messages under light to moderate traffic loads. Unlike token-passing protocols, such as ARCnet and IEEE 802.4, an Ethernet node can get immediate access to the network if there are no messages currently being transmitted. However, Ethernet is not deterministic: there is no upper bound on how long it may take for a network node to gain access to the network.

Ethernet chip sets are usually designed to interface with microprocessors rather than microcontrollers. Adding Ethernet capability to an embedded controller design typically requires two or three packs plus microprocessor bus interface circuitry. The Ethernet controller will handle the transmission and reception of messages, but as with ARCnet the software running on the microprocessor will be responsible for formatting messages to send and interpreting received messages. Ethernet chip sets are manufactured by several companies including Intel, National Semiconductor, AMD, Philips-Signetics, and Standard Microsystems.

Ethernet interface cards are available for the PC, allowing you to put PCs and embedded controllers on the same Ethernet network.

MICROCONTROLLERS WITH BUILT-IN SERIAL COMMUNICATION MODES

For applications in which hardware cost must be minimized, you should consider designing a network that makes use of serial communications capability, which is built into virtually all common 8- and 16-bit microcontrollers as well as a few 32-bit devices. A popular communication medium is twisted-pair cable, to which the microcontroller is interfaced using an inexpensive RS-485 transceiver like the 75176.

This approach has some potential limitations. The network bandwidth is moderate, typically 57,000 to 375,000 bps. You have to write or purchase the software implementation of a network protocol. Finally, the communication

Microcontrollers with 9-bit asynchronous serial communication capability (partial list)

Hitachi:
- HD64180

Intel:
- 805 1 family
- 8096
- 80C186/188 EB/EC

Motorola:
- 68HC05 family
- 68HC 11 family
- 68HC16
- 68300 family

National Semiconductor:
- COP884 CG/CS
- COP888 CG/CS/FG
- HPC family

Texas Instruments:
- TMS7002/7042

Zilog:
- Z180 family
- Super8

Network nodes:	up to 32
OSI layers specified:	physical (W-485) data link, application
Electrical specification:	(RS-485)
cable:	twisted-pair
length:	up to 1200 meters
speed:	9600 bps
Protocol:	proprietary
Transmission:	asynchronous serial
Hierarchy:	master/slave
Media access:	controlled by master node
Error detection features:	message acknowledgement (optional), 8-bit checksum
Application layer features:	data transfer

Figure 6—*Micromint's MC-Net allows* **the connection of** *their 8052-BASIC and HD64180-based embedded controller boards and a PC.*

processing overhead can be significant, depending on the protocol you use and the amount of network traffic.

Asynchronous *serial* communication

Most microcontrollers with built-in serial communication capability have UARTs that can be used for networking.

Two techniques have been used to reduce asynchronous serial communication processing overhead, and most microcontrollers can make use of at least one of them. Intel's 9-bit multiprocessor mode reduces communication processing by using the ninth bit of each 9-bit character to indicate the beginning of a message, the first character of which is the node address. Motorola's technique uses an idle line to delimit messages. Of the two techniques, Intel's is found in more microcontrollers, including Motorola's more recent parts (see sidebar). If you are interested in the description of a network protocol that makes use of 9-bit multiprocessor modes, refer to my article, "A Simple RS-485 Network," in ***Circuit Cellar INK,*** issue #21.

Many people who construct asynchronous serial networks write their own network software. However, at least two companies are offering microcontroller software for asynchronous serial networking. Micromint Inc. (Vernon, CT) offers network software that allows the connection of Micromint's 8052-BASIC and HD64180-based embedded controller boards and a PC (see Figure 6). Cimetrics Technology (Ithaca, NY) offers software toolkits for embedded controller networking, which support several microcontroller families (805 1, 68HC11, Z180, and 80C186EB) and the PC. Their NSP protocol allows the master network node to read from and write to slave node memory and I/O ports (Figure 7).

Synchronous *serial communication*

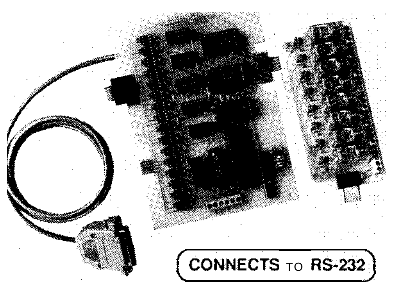
Synchronous serial communication differs from asynchronous serial communication by not using start and stop bits for synchronization, instead synchronization information is extracted from the data stream. As a result, synchronous communication

Network nodes:	up to 250 (32 per cable segment)
OSI layers specified:	physical (RS-485 recommended), data link, application
Electrical specification:	(W-485)
cable:	twisted-pair
length:	up to 1200 meters per cable segment
speed:	depends on microcontroller, 62.5 kbps typical
Protocol:	NSP, an adaptation of IEEE 1118
Transmission:	asynchronous serial
Hierarchy:	master/slave
Media access:	controlled by master node
Error detection features:	message acknowledgement (optional), 16-bit CRC, sequence count
Application layer features:	data transfer

Figure 7-The *9-bit multiprocessor* mode *supported* by many popular microcontrollers is the basis of *Cimetrics* Technology's *NSP protocol.*

RELAY
INTERFACE

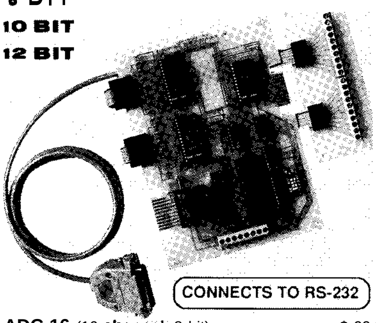
PROVIDES SOFTWARE CONTROL OF RELAYS



CONNECTS TO RS-232

AR-16 RELAY INTERFACE ....$ 89.95
Two 8 channel relay output ports are provided for control of up to 16 relays (expandable to 128 relays using EX-16 expansion cards). Each relay output port connects to a relay card or terminal block. A variety of relay cards and relays are stocked. Call for more info. RS-422 available (distances to 4,000 feet). PS-8 port selector may be used to control satellite AR-16 interfaces (up to 16,384 relays).
RD-8 REED RELAY CARD (8 relays) .... S 49.9:
RH-8 RELAY CARD (IO amp SPDT 277 VAC)....$ 69.9!
EX-16 RELAY EXPAN. CARD (16 channel)......$ 59.9!

ANALOG TO
DIGITAL
8 BIT
10 BIT
12 BIT



CONNECTS TO RS-232

ADC-16 (16 channel, 8 bit) .... $ 99.95
Input temperature, voltage, amperage, pressure,, energy usage. energy demand. light levels, joystick movement and a wide variety of other types of analo signals. Inputs may be expanded to 32 analog or 12 status inputs using the AD-16 or ST-32 expansion cards. 112 relays may be controlled using EX-16 expansion cards. Analog inputs may be configured for temperature input using the TE-8 temperature input conversion. RS-422 available. PS-8 port selector may be used to connect satellite ADC-16 interfaces (up to 4,096 analog inputs/16,384 status inputs and 14,336 relays). Call for info on 10 & 12 bit converters. [terminal block and cable sold separately)
ST-32 STATUS EXPAN. CARD (32 channel)....$ 79.95
Input on/off status of relays, switches, HVAC equipment, thermostats, security devices, smoke detectors and thousands of other devices. (opto isolators sold separately)
TE-8 TEMPERATURE INPUT CONV $ 49.95
Includes 8 linear IC temperature sensors. Temperature range is minus 78 to 146 degrees F.
TOUCH TONE DECODER and other serial interfacing products available. Call for free information packet.

- FULL TECHNICAL SUPPORT...Provided over the telephone by our staff. EACH ORDER INCLUDES A FREE DISK WITH PROGRAMMING EXAMPLES IN BASIC, C AND ASSEMBLY LANGUAGE. A detailed technical reference manual is also included.
- HIGH RELIABILITY...engineered for continuous 24 hour industrial applications. All ICs socketed.
- Use with IBM and compatibles, Tandy, Apple and most other computers with RS-232 or RS-422 ports. All standard baud rates and protocols may be used (50 to 19,200 baud).
- Use our 800 number to order FREE INFORMATION PACKET. Technical information (614) 464.4470.

24 HOUR ORDER LINE (800) 842-7714
Visa-Mastercard-Ameiican Express-COO

ELECTRONIC ENERGY CONTROL, INC.
360 South Fifth Street, Suite 604
Columbus, Ohio 43215

#110

uses network bandwidth more efficiently. Industry-standard protocols that use synchronous serial communication include HDLC (IS0 standard 4335) and SDLC.

A few microcontrollers have synchronous communication capability appropriate for networking including the Intel 8044 (similar to the 8044 BEM previously mentioned in the BITBUS section), the Zilog Z80 18 1 (a Z80 variant), and the National Semiconductor HPC16400. These chips have hardware that assists you in the implementation of the HDLC- or SDLC-like protocols. Detection of the beginning and end of messages, address recognition, O-bit insertion, and CRC computation are common features. You will have to implement upper protocol layers in software, and depending on the microprocessor you choose, you may also have to implement some of the data-link layer.

Microcontroller-peripheral networks

Another common serial communication feature on microcontrollers is a clocked synchronous serial communication subsystem for communication between microcontrollers and peripheral chips over a small area (typically within a controller or appliance). This feature can allow chips to communicate using just two or three connecting wires.

One of the most popular serial bus specifications is the Philips-Signetics I²C bus, a two-wire multimaster serial bus capable of up to 100 kbps. In addition to I²C peripherals and microcontrollers, Philips-Signetics offers I²C chips that allow some non-PC components to communicate on an I²C bus.

SERIAL COMMUNICATION ICs

A few microcontrollers and most microprocessors lack built-in serial communication capability. If you intend to use such a processor, you may be able to add a serial communication chip to your design at a cost significantly lower than adding an ARCnet or Ethernet chip set [described previously].

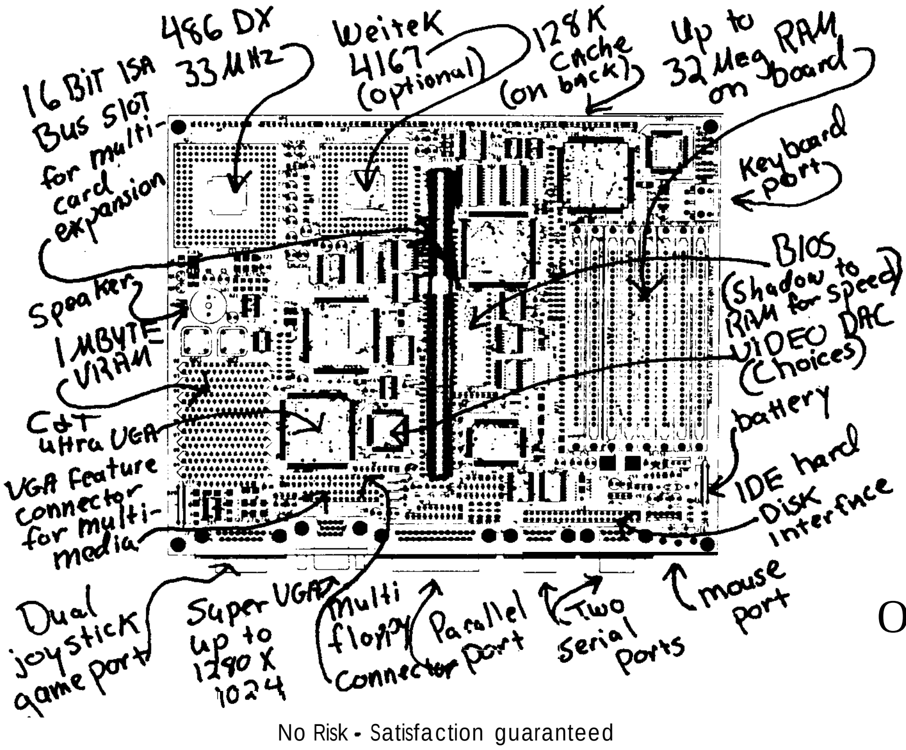
The use of a serial communication IC has the same disadvantages as the use of microcontrollers with built-in serial communication features. Namely, this solution requires a software implementation of a network protocol.

One feature to look for in serial chips is the presence of FIFOs, which can reduce the probability of lost characters and reduce communication processing time. One popular serial communications controller with both asynchronous and synchronous capability is the Zilog 28530 (the recommended chip for AppleTalk hardware). For asynchronous communication, I have had good results with the National Semiconductor NS16550AF and Intel 825 10 UARTs, both of which have FIFOs.

LOOKING AHEAD

At this point, the list of adopted or emerging standards relevant to embedded controller networking is expanding. In addition to the standards I've already described, the four men-

What will you do with this board?



No Risk • Satisfaction guaranteed

9 1/2" x 7 1/2"

It's a complete working 486 computer on a single board.
Cool. No fan required.

Use it to build:

Embedded video games. process controllers, testing instruments. production controllers, multi-racked computers. any embedded application, or tiny 486 desktop computers (tiny proto case included • holds floppy & hard disks. this board & two halt cards).

Fully functiil **with standard motherboard peripherals (DMA, clack, etc.) included.**
External 12V power supply included.

Single evaluation units: S 1799 ea.
Competitive quantity pricing.

Business Instruments
Computer Innovations Since 1978
Order **1-800-UP-I-JUMP**
MC, VISA, Company Check
Tech (214) 980-9990
FAX (2 14) 980-7030
13410 Preston Rd., Dallas, TX 75240

1111

tioned below may have significant impact on embedded controller networking. Industry-specific standards are also being developed.

CEBus is an emerging home automation standard that has been under development for the last several years. Interested readers can refer to three previous ***Circuit Cellar INK*** articles for details (see references). CEBus hardware is beginning to appear; Intellon Corporation (Ocala, FL) recently introduced a power-line modem IC capable of communication at 10,000 bps.

IEEE 1118 is a recently approved standard for a serial control bus based on the BITBUS protocol. However, I am not aware of any products other than BITBUS products that are IEEE 1118 compliant at this time.

General Motors adopted MAP (Manufacturing-Automation Protocol] in order to allow the networking of GM's numerous PLCs and robots. Subsequently, MAP was adopted by several other large corporations. At the lower protocol layers, MAP was based on IEEE 802.4 (token bus) and 802.2. MAP has the reputation of being very expensive to implement, which has certainly slowed its acceptance.

FieldBus is an emerging ISA [and IEC) serial communication standard for the networking of low-level devices in an industrial setting.

IN CONCLUSION...

International standards for embedded controller networking have been slow to develop, and some will eventually have a large impact. But as I have demonstrated, the large number of hardware and software components available make the construction of practical embedded controller networks possible today. ❑

Jim Butler is a software engineer at Cimetrics Technology. He received B.S. and M.S. degrees in engineering from M. I. T.

The BITBUS Interconnect Serial Control Bus Specification, Intel Corporation, 1988.

Ken Davidson, "Echelon's Local Operating Network," ***Circuit Cellar INK,*** **issue #21.**

"Connecting with Neurons," ***Embedded Systems Programming,*** **Sept 1991.**

"Choosing a network for local industrial control," ***EDN, Nov 24 1988.***

"ARCnet Chip Tackles Real-Time Embedded Control," ***Electronic Design, Nov 8*** **1990.**

"The deterministic character of ARCnet proves ideal for the factory floor," ***EDN,*** **Sep 15 1988.**

"ARCnet Token Bus Network: Technical Overview," ARCnet Trade Association (Arlington Heights, IL).

"The Return of ARCnet," ***BYTE,*** **Feb 1991.**

"Ethernet: Ten Years After," ***BYTE,*** **Jan 1991.**

David Flint, ***The Data Ring Main: an Introduction to Local Area Networks,*** **Wiley Heyden, pub., 1983. Good Ethernet information; also a good general reference on networks.**

Jim Butler, "A Simple RS-485 Network: Exploit the Nine-Bit Serial Communication Modes of the 805 1, 8096BH, 68HC11, 68HC05, and 2180 Microcontroller Families," ***Circuit Cellar INK,*** **issue #21.**

Ken Davidson, "CEBus Update: More Physical Details Available," ***Circuit Cellar INK,*** **issue #19.**

Ken Davidson, "CEBus: A New Standard in Home Automation," ***Circuit Cellar INK,*** **issue #10.**

Harv Weiner, ed., "New Product News: CEBus Power Line Interface Products," ***Circuit Cellar INK,*** **issue #25.**

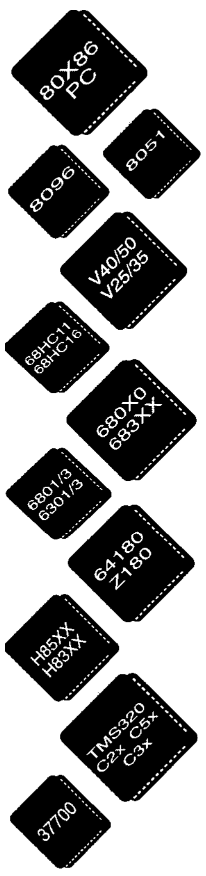
"The Best LAN May Be Found off the MAP," ***EDN, Nov 7*** **1991. Also mentions Ethernet and ARCnet.**

"FieldBus: An Emerging Communications Standard," ***Microprocessors and Microsystems,*** **Dec** ***1988.***

Ken Davidson, "Domestic Automation: CEBus Goes Coax," ***Circuit Cellar INK,*** issue **#25.**

I R S

401 Very Useful
402 Moderately Useful
403 Not Useful



Get Byte-BOS" and leave the MULTITASKING to us!

Why spend months designing, documenting, coding, and debugging a multitasking kernel when you can have Byte-BOS, a widely used multitasking operating system on your desk tomorrow?

Byte-BOS Multitasking Operating System consists of a complete set of integrated multitasking components including a robust kernel, "add on" libraries, and development tools that run on a PC.

Only $995 buys our multitasking kernel with all these features:

- ▫ preemptive task scheduling
- ▫ task management
- ▫ event management
- ▫ message management
- ▫ timer management
- ▫ resource management
- ▫ application code & user manual
- ▫ configured to your c compiler
- ▫ no royalty c source code
- ▫ 1 year of tech support & updates

Add these multitasking libraries to suit your application:

- ▫ "on chip" serial I/O manager
- ▫ external serial I/O manager
- ▫ fixed block memory manager
- ▫ multiple message buffer manager
- ▫ multiple timeout manger
- ▫ dynamic task manager

Develop **and debug Byte-BOS applications on a PC with these tools:**

- ▫ BOSPCX for target prototyping
- ▫ BOSVIEW multitasking debugger

Byte-BOS is available for a wide range of microcontrollers and **the PC. Our technical support is responsive, friendly, and knowledgeable. Call us today and see how easy multitasking can be with Byte-BOS.**

Byte-BOS™ Integrated Systems
PO Box 3067 Del Mar CA 92014

800-788-7288 or 619-755-8836

#112

FEATURE ARTICLE

Infrared Tracking and Remote Control

Meet the New HCSII IR-Link Module

Steve Ciarcia

Images of Big Brother accompany the idea of your house following your every move. Put Steve's people tracking system to good use as part of the new Circuit Cellar Home Control System.

Ever since I built my first home control system (presented in April '85), one of the improvements I've wanted to incorporate into any new HCS was an ability to extend control of the system by remote means. After all, these days I can sit on the couch with one IR remote control and command an entire audio-visual surround sound entertainment system.

By now you know that the new HCS II is a reality (see sidebar for more information). While it contains noteworthy enhancements over the original, the physical solution to producing some of these features was a far cry from the initial design technique. IR remote control was a prime example.

During the design phase I approached the task of adding remote control by looking for an off-the-shelf IR remote control chip that could easily interface to the HCS. Since IR remote control chips usually come in encoder/decoder pairs, I assumed the proper route would be to use the encoder chip to make a hand-held device with buttons (as if you really needed another IR remote) and use the decoder chip [with suitable IR recognition circuitry) wired to the HCS as the receiver. Press a coded key on the transmitter and presto, the code is received by the HCS and acted upon.

Ed, Ken, and I had already specified the basic configuration of the new HCS and its networked COMM-Links. The X-10 power line controller (designated the PL-Link) was already up and running. The IR-Link was to be the next module on the system.

My initial proposal designated the Motorola MC 145030 as a suitable remote control encoder/decoder chip [see Figure 1). An added bonus was that the MC145030 contains both an encoder and decoder, eliminating the need for two separate chips.

The MC145030 encodes and decodes 9 bits of information (512 combinations). The chip Manchester encodes the selected address input and sends the information (twice) serially out via the Encoder Out pin (pin 16). The transmission frequency of this data is determined by an RC oscillator. This frequency can be up to 500 kHz but, for reasons Ed and I will explain later, I chose 12.4 kHz. Sending a command therefore takes 5.16 ms.

I quickly threw together the circuit in Figure 2 to test the concept. This simple three-chip circuit takes the Manchester-encoded serial output and modulates it with a 38-kHz square wave (selection of the modulation frequency depends on IR receiver used). This signal then controls a pair of IR LEDs through a FET. Press transmit and everything is automatic.

The receiver circuit, shown in Figure 3, is even less complex. The 38-kHz IR signal is received and demodulated through a Sharp IS1U60 IR receiver chip. Its output is inverted and connected to an MC145030, which is configured this time as a decoder. When the specific 9-bit code selected on its address inputs is received, the MC145030 generates a "code received" signal that can be easily connected to a processor interrupt.

OK, LET'S START THROWING STUFF OUT

It took about three nanoseconds to realize that adding a serial decoder chip to a processor is like putting a saucer under a coffee cup. You only need it when you rock the boat.

In computerese, this translates to mean having the decoder chip is redundant. Given the low data rate involved, the decoding function can be completely simulated in software. All we need is to connect the IS1U60 IR receiver directly to the processor and add a little fancy "Nisley stuff." The added advantage of receiving it totally in software is that the 9 bits transmitted can now be completely used as data. Rather than designate a single address confirmation, as the hardware unit does, the 9 bits can be used to designate 5 12 codes for ID badges, optical keys, hand-held remote corn-mands, and so forth. Any way you can transmit the Manchester code, the IR-Link could now receive and process it.

Back at the transmitting end we now had a different problem. I wanted everyone to utilize the new IR-Link for remote control, but I had a lot of trouble justifying the fact that we'd have to supply a costly hardware transmitter to use it.

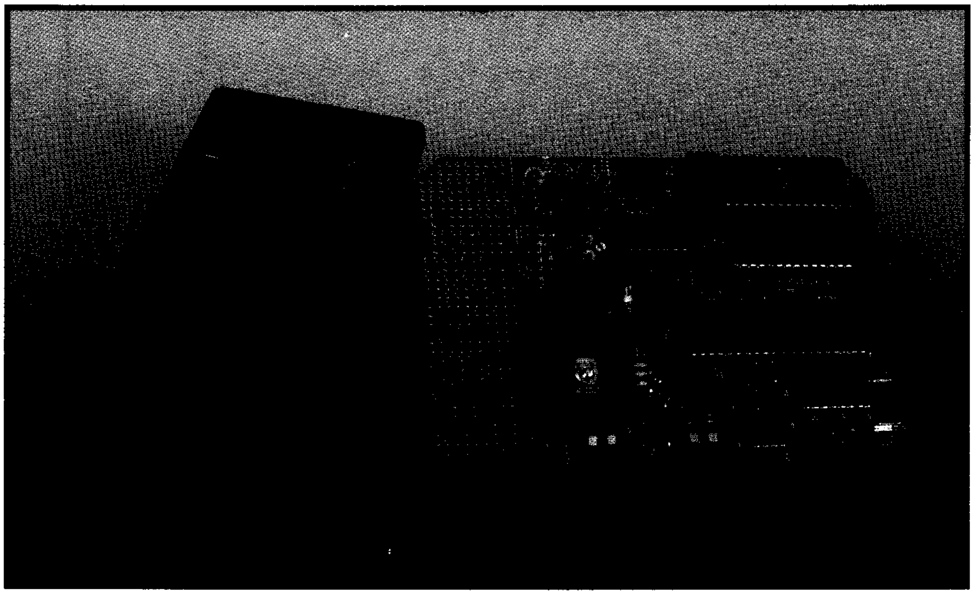


Photo l-Any *trainable* held-held remote *may be used to send commands to the HCS II. The IR-Link is a gateway responsible for directing IR codes to the Supervisory Controller.*

While I was mulling *over* this obvious production problem, testing of the basic elements continued. To eliminate changing jumpers every time I wanted a different code, I bought a trainable IR controller at Radio Shack and trained it with a dozen or so codes. Rather than change jumpers or add a keypad encoder to the prototype, I now just used the "trained" remote.

Well, it took four nanoseconds for lightning to strike this time. I used a physical circuit containing the encoder chip to create the signal source. However, if we had some way to simulate the Manchester-encoded IR signals without actually using an MC 145030, then we could use a trainable remote and eliminate the cost of making a unique hardware transmitter.

The obvious answer was that if the IR-Link processor could decode the serial data, it could also generate and encode it. In effect, the transmitter could be reduced to nothing more than an IR LED controlled by a processor output bit and a software routine.

Ed and I discussed the possibility of indeed reducing the transmitter to just an LED, but some practical limitations intervened. While a 12.4-kHz data rate was easily accommodated, the 38-kHz modulation frequency was not as easily synthesized. Because the COMM-Link uses an 11.0592-MHz crystal, the software-generated 38 kHz could not be produced exactly on frequency. In fact, it could be off by as much as 1.5%.

When you look at the specification of the IS1U60, this frequency shift does not appear to degrade performance, but given some of the physical tests I performed, I think some of their specifications are optimistic. The IS 1U60 is an extremely good IR receiver but is also a narrow-band receiver. As soon as you move off frequency, the sensitivity rapidly drops.

THE IR-LINK HARDWARE

To complicate matters, transmission frequency is only part of the equation. A trainable remote also uses a processor and internal timebase to sample and reconstruct IR waveforms. It is also susceptible to the same timebase errors described above. If we synthesize a signal with a certain error, and then sample and regenerate it with more error, we could potentially move out of the bandwidth of the IR receiver again.

To eliminate error sources from our end, I put a hardware 38-kHz oscillator on the IR-Link board. The

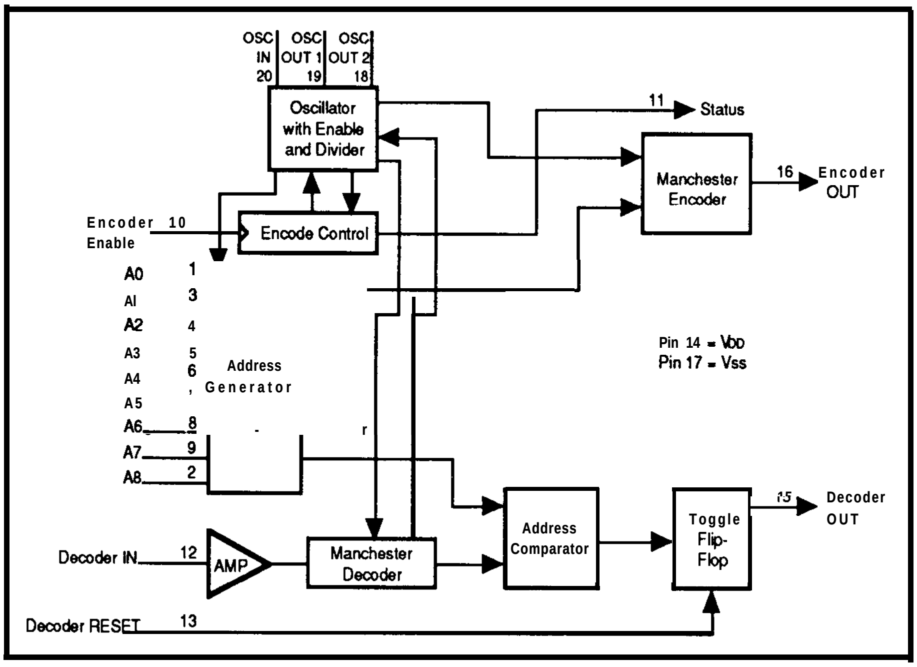


Figure l-The *Motorola MC145030 is a remote control* encoder/decoder *chip that handles up to 512 unique IR codes.*

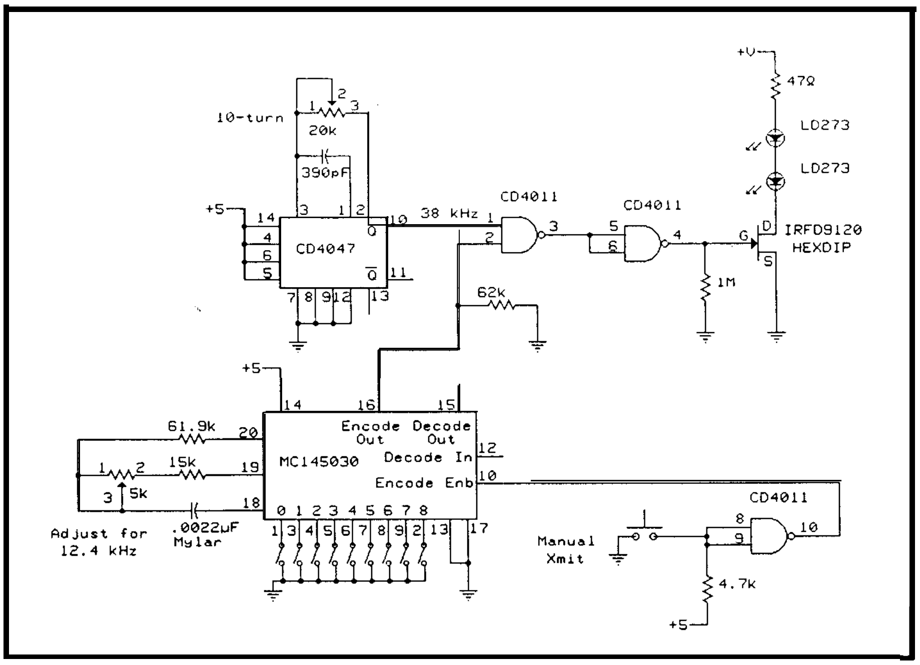


Figure 2-A basic *transmitter circuit uses the MC145030* to encode *the data and to gale a 38-kHz oscillator in response to a button press. The code sent is set with a bank of switches.*

software still encodes and creates the serialized Manchester data, but from there it is ANDed with the 38-kHz signal to produce the modulated signal. A 2N2907 transistor pulses a pair of single-element or a dual-element IR LEDs. The schematic of the 803 1 -based COMM-Link circuitry (explained in the last issue) is shown in Figure 4. Figure 5 outlines the additional circuitry specific to the IR-Link. Photo 1 shows an assembled IR-Link board.

Finally, in viewing the schematic, some of you might ask why I have a 3-volt LM317 regulator in the circuit. I have already suggested that the 38-kHz modulation frequency is critical. A shift in frequency affects the IS 1U60's receiver's response, and hence the distance over which the IR-Link could receive commands. A remote control sending a coded signal at 38 kHz will be received at a much greater distance than one sending the same code at 39 kHz, for example. After going through so much effort to reduce training errors, frequency shift in the oscillator itself had to be minimized.

A CD4047 oscillator chip with polycarbonate or mylar capacitors is a relatively stable clock source. However, the major error-causing influence is the power supply voltage. Set the oscillator when the supply is at 4.9 V, then use it at 5.1 V and there will be a frequency shift. To minimize power supply variations, the CD4047 chip has its own regulator. Since we can't know what voltage will be used to power the IR-Link (5 V, or 9-12 V, even possibly up to 35 V), the LM3 17 operates from the 5 V supplied to the other chips (3 V is still a valid TTL switching level]. As a result of the regulator, the frequency should stay the same during calibration or use.

The final IR-Link design did away with the hardware encoder/decoder chip and synthesizes the Manchester codes in software. Ed describes the command set in his article, so I won't duplicate the explanation here other than to reiterate our design approach. Like the other HCS II COMM-Links, the IR-Link is intelligent and designed for both independent or network operation. As an independent peripheral connected via RS-232, the IR-Link adds IR code-activated control and recognition to any PC.

BADGE READERS AND PEOPLE TRACKING

The idea behind the IR-Link is to use a trainable hand-held IR remote controller like the one you might already be using with your TV and have it contain 10 or 20 of these control codes. Simply aim the remote at the IR-Link and the code will be received by the HCS II's Supervisory Controller (SC). Within the SC's event repertoire, you might have

```
IF IRcode(20)=1 THEN
   Exhaust_Fan=ON
END
```

or

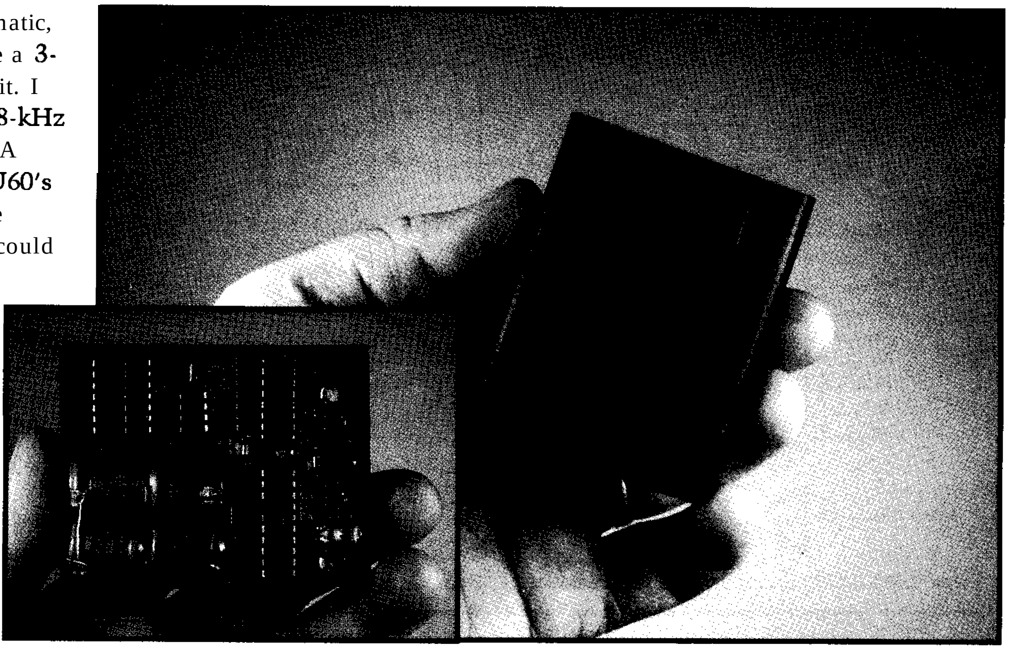


Photo 2—*The infrared ID badge, the key to the people tracking system, is no larger than a small* pager

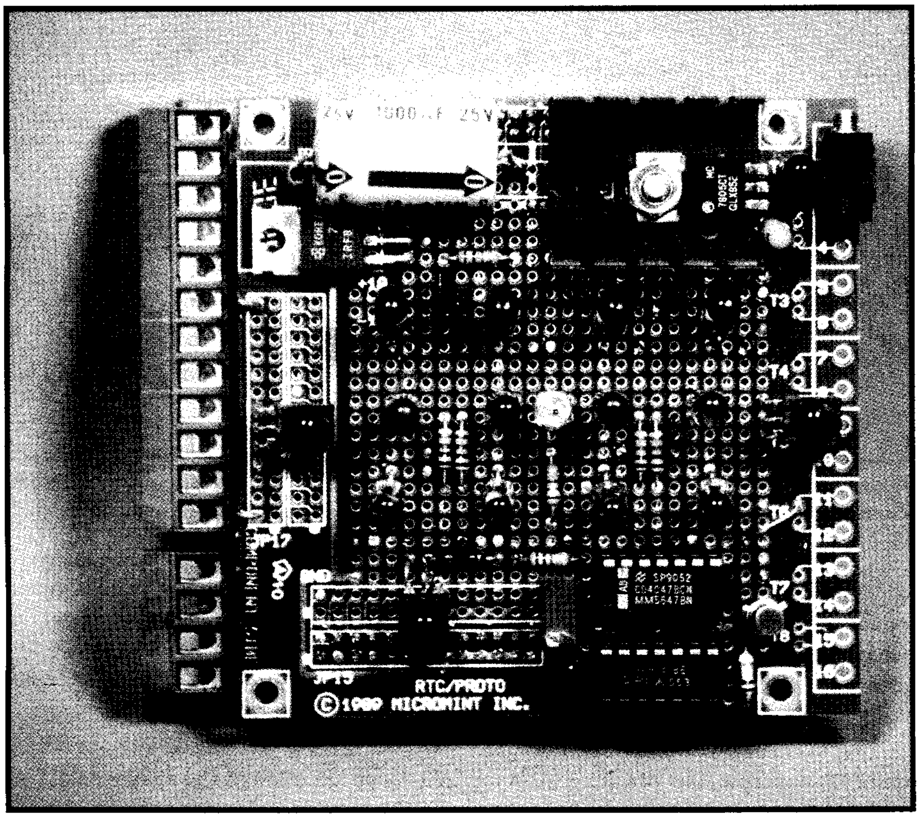


Photo 3—*The "room blaster"* "interface *for* the *IR-Link* uses *multiple IR LEDs to ensure all* badges in the mom *will* hear the *transmission regardless of their orientation.*

```
IF IRcode(32)=1 THEN
    Alarm_Bell=ON
    AllLightsOn(C)
END
```

Given 512 codes and the intelligent programming of the SC, your hand-held remote control can actually do quite a bit.

As easy as the concept might be to synthesize the IR-Link functions in software, I still found it difficult to give up soldering something. Having the ability to network together enough

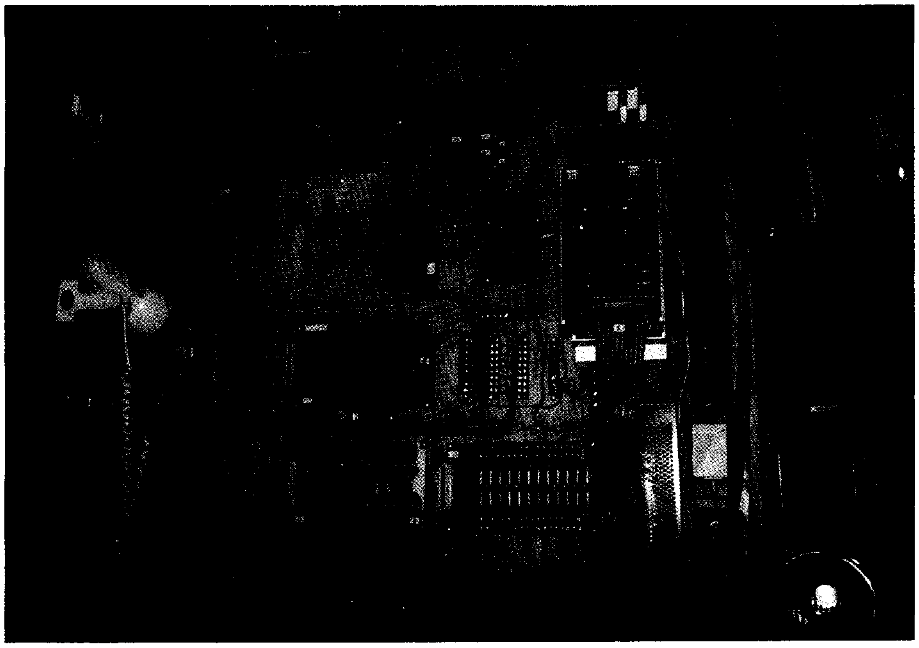


Photo 4—*Unbelievably, this is* the same *wall that* was shown *in the last* issue. The additional *features built info the HCS II and ifs expandability allow a much cleaner setup with more capability than was possible with he original HCS.*

FAST COMPLETE ACCURA TE

DRAM TEST

DIPs - SIMMs - SIPs

RAMSTAR

1ns. RESOLUTION

ACCESS SPEED VERIFICATION

80 ns. thru 180 ns. (Std.)	5249.01
45 ns. thru 110 ns. (Fast)	$349.0(
4MEG Option	Add $ 89.01

AUTO-LOOP
Continuous Test 6.25 MBits/sec.

ADAPTERS:

SIMM/SIP ADAPTER $189.01
Tests 64K,256K,1 M & 4M Devices
8 or 9 Bit versions.

NEW

NTX ADAPTER $149.00
Tests 64 Pin Dual-Edge
Laserwriter Type SIMM's.

4 X ADAPTER $ 89.010
Tests 64K & 256K By 4 Bit Devices

AC ADAPTER $ 18.00
Regulated +5V @ 1 Amp.

FREE RAMFACTS DRAM NEWSLETTER

1-800-RAMSTAR

COMPUTERDOCTORS
9204-B Baltimore Boulevard
College Park, Maryland 20740
MADE IN U.S.A. U.S. PATENT No. 4.965.79

#113

IR-Links to cover a whole house or office and only use it for IR remote control seemed especially strange.

Not that I have any great ambitions to deal with a computer as if it were another person, but home control system software can be greatly personalized if the system has a "special" program that runs when it "knows" you are there. For example, the HCS II's normal control program might be simply to turn a table lamp on in the solarium when someone enters the room (sensed by a motion detector). However, if after 5 P.M. it identifies that I am the person entering the room, rather than just turning on the one lamp, it could make my hard day at the office a faded memory as it turns on the table lamp, dims it to half, and then switches on and dims the lights over the bar as it queues the stereo and CD player. All I need now is the automatic martini maker [a serious consideration).

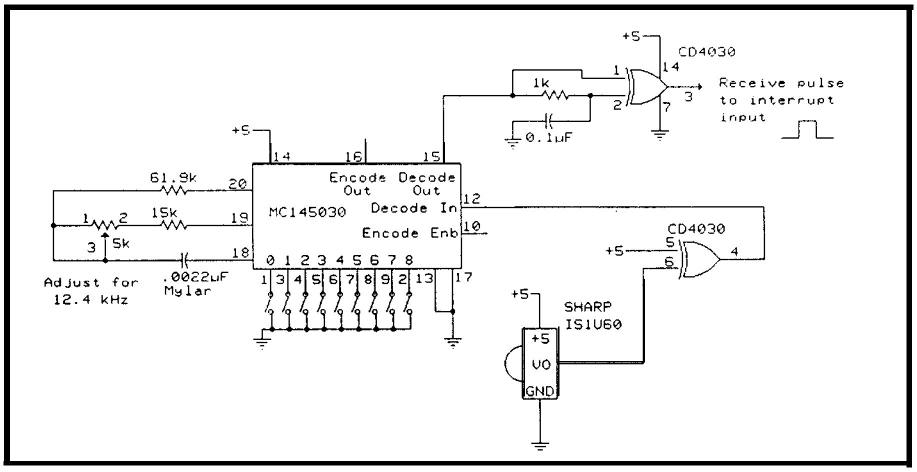


Figure 3—A *basic receiver uses a Sharp* ***IS1U60 IR*** *receiver to* ***condition the*** *incoming* ***signal*** *and the* ***MC145030*** *to decode it When the code matching* ***the*** *one set on* ***the switches*** *is* ***received, the MC145030*** *generates* ***a*** **pulse.**

This scenario can be extrapolated even further if we suggest that as long as the system could identify me and which location I am in, it could channel the music to follow me as I roamed about.

Of course, this might seem an extreme example, but the suggestion that a control program can be tailored to an individual person is valid. On the SC, the event sequence starts with:

IF Steve=Sol **ari** um

where "Steve" is defined as a specific IR response code to the **system(e.g.,**

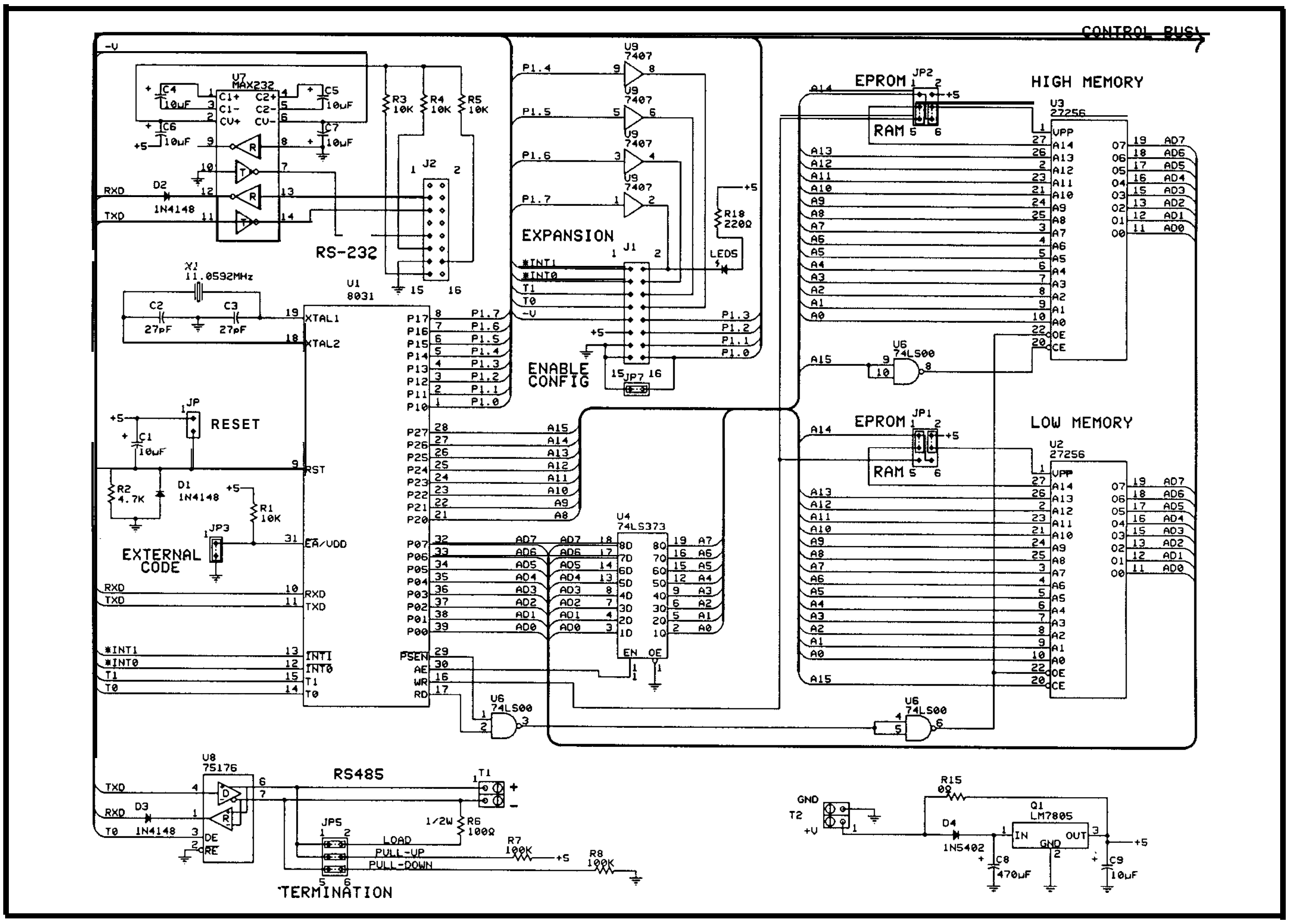


Figure 4—The COMM-Link is the basis for all the HCS II expansion *modules.* ***It*** *is based on the* ***8031*** *and has a* ***voltage*** *regulator on the board so can be run from a range of power supples.*

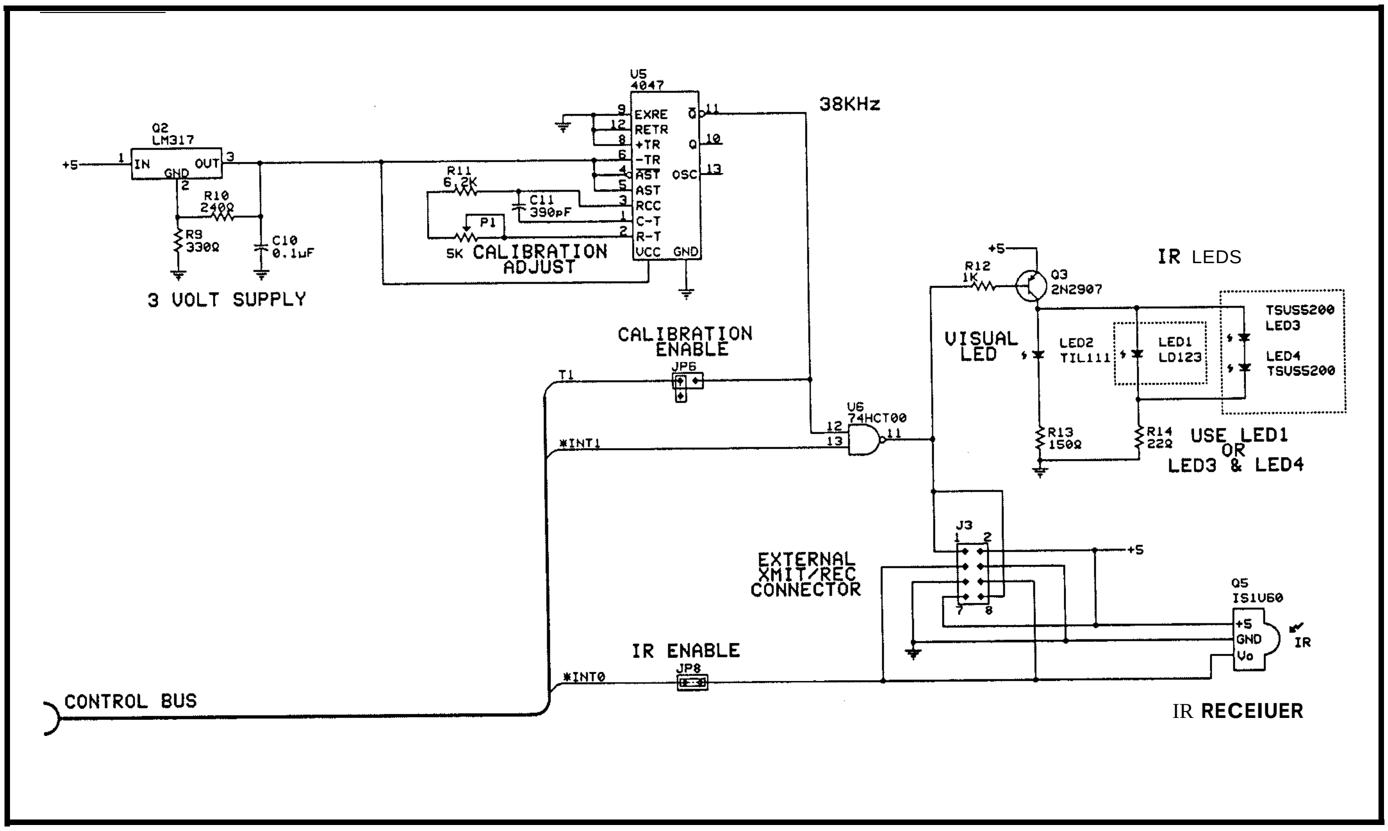


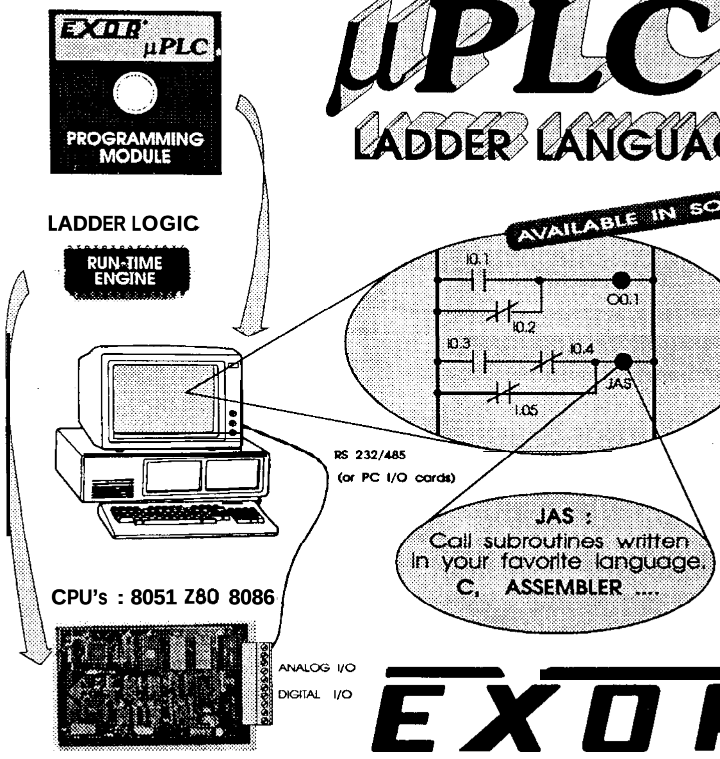
Figure 5—*To* **make** *an /R-Link bard, just add an oscillator, an IR receiver, and some LEDs m the base COMM-Link board.*

µPLC®

LADDER LANGUAGE

AVAILABLE IN SOURCE CODE

The basic building block for **your** control **system !**


EXOR µPLC
PROGRAMMING MODULE
LADDER LOGIC
RUN-TIME ENGINE
I0.1
O0.1
I0.2
I0.3
I0.4
JAS
I.05
RS 232/485 (or PC I/O cards)
JAS : Call subroutines written in your favorite language. C, ASSEMBLER ....
CPU's : 8051 Z80 8086
ANALOG I/O
DIGITAL I/O
YOUR EMBEDDED CONTROL SYSTEM

The µPLC operating system kernel trasforms your basic microprocessor into a high- performance Programmable Logic Control.

Just plug the µPLC ROM into your favorite microprocessor card, load the integrated programmer/debugger onto your PC, connect a serial cable and begin taking the credit for a job well done!

Get started for only $ 399

Eval. Kit $ 99

- RS 485 multidrop distributed system
- Easy porting to custom hardware
- ROMable Run-Time Engine
- Programming tools
- On-line debugging
- Target library

Eval. Kit contains :

- *Parallel Port I/O or On Screen Simulator*
- *Programming & Debugging modules*
- *PC TSR Target*

Excellent for ladder language training

EXOR®

ELECTRONIC R & D

FAX (513) 874-3684 CALL (513) 874-4665

4850 Interstate Dr. Cincinnati, OH 45246

114

MCIR-Link, TIM-Link, and...TREK-Link?

Generally speaking, we at *Circuit Cellar INK* like to be remembered for our achievements, so we rarely speculate and try not to discuss "vapor" hardware or software. Unfortunately, the immediate success of the Circuit Cellar HCS II has caused quite a number of you to ask about other COMM-Links beyond those presently available.

Our policy has always been to design and build interfaces before announcing them. Too many unscrupulous businesses float "vaporware" before actually producing anything. However, we are often guilty of sitting on a design until all the pieces are complete before we present it.

A certain compromise may be in order. The success of the HCS II deluged our offices with questions about pending designs. Our merely answering, "No" to the potential existence of other COMM-Links could give you the impression that the HCS II is a dead-end design, rather than a continuing vehicle for high-tech experimentation among the Circuit Cellar engineering staff.

By breaking precedence and telling you our plans, I risk discussing vaporware in my effort to satisfy reader demands. However, given this caution, combined with the understanding that any projects I mention here are still just ideas and that my describing them does not mean they will be automatically published as design projects, let me outline some future plans for HCS II.

First of all, Ed and I are going back and resurrecting the Master Controller Trainable Infrared unit (published in *Best of Ciarcia's Circuit Cellar,* chapter 32) and attaching it to a COMM-Link node. After training the Master Controller [using a PC) with all your hand-held remotes, the Supervisory Controller can then command any individual or series of stored IR sequences to be transmitted. This capability makes it possible for the HCS II to, for example, cue up a specific CD or tape and set the appropriate volume in response to you walking into the room.

Second, Ken and I want to build an updated version of the Touchtone Interactive Monitor (TIM) system that I presented in issue #21. We recognize the need for outside interaction with the HCS, and a more intelligent and articulate version for TIM is our goal.

Finally, all of us have discussed a true voice interactive interface, à la the *Enterprise's* computer from "Star Trek." I still think the concept has more entertainment than control value, but the idea is intriguing nonetheless. Unfortunately, this idea is one that is still waiting for technology to advance to a point where there is a cost-effective solution. How long this module remains on the drawing board depends on such an evolution.

So I ask you to bear with us while we take some time before delivering these designs to you. After all, we'd like them to work first.

"IRcode(IO)") and "Solarium" is defined as the number of the IR-Link located in the solarium (e.g., "2"). When the condition is true, a list of actions will be performed. The only concern is finding a cost-effective method to identify an individual.

While possibly suited more for an office than the home, our solution was to use an electronic ID badge. We have already described that the IR-Link's primary function is to sit passively and monitor the ether for 38-kHz Manchester coded IR transmissions.

Take the circuit in Figure 2, add a lithium battery, replace the manual transmit switch with a slow 0.1 -Hz oscillator, and build it into a small pocket-clip container. Every 10 seconds this portable beeper transmits a coded signal just like you were pressing the button on a hand-held remote. Provided it is aimed in the vicinity of an IR-Link (more about increasing coverage later), the SC will receive that transmission as it would any other. By selecting a unique code for each badge (different from those reserved for remote control) the system "knows" both the identity and the location of an individual.

Of course, as with any asynchronous transmission method like this one, if too many badges are in the same area, there can be message collisions. The solution is to use an "intelligent" badge like that shown in Figure 6. In this circuit, each badge has a unique code (1 of 5 *12)* set by the jumpers on the MC 145030. The badge is designed to transmit only after it receives its own code.

Using the IR-Link, we "poll" a particular area by sending out specific badge codes. When the badge receives its code, the circuit waits 25 ms and then sends back the same code. When the IR-Link receives this return transmission, the system then knows the location of that ID badge. The IR-Link has built-in commands to facilitate polling an individual badge or complete ranges of badges.

To prove the concept was feasible, I built one of these intelligent badges into a package that is the size of a small pager (see Photo 2). I will have to say the entire project would have been unfeasible were it not for the unique size and capability of the tiny Sharp IR receivers.

CUSTOMIZED REALITY

I want to be careful not to confuse you at this point. The badge reader that I built is a prototype and not a production item. I built it to test both the IR-Link software and the concept of a people tracking system.

The basic HCS II IR-Link utilizes a single IR LED and IR receiver because it's supposed to be used with a trainable hand-held IR remote control. Unlike a hand-held remote that can be directly aimed at the IR-Link's IR receiver, a badge attached to a pocket or belt may not be in direct line of sight with the IR-Link. As a result, to properly implement the badge tracking system I've described, additional IR LEDs and IR receivers must be connected in parallel with those on the IR-Link. There is a 6-pin header designated for this off-board expansion.

The IS 1U60 IR receivers are open-collector devices that can be directly connected in parallel but aimed in

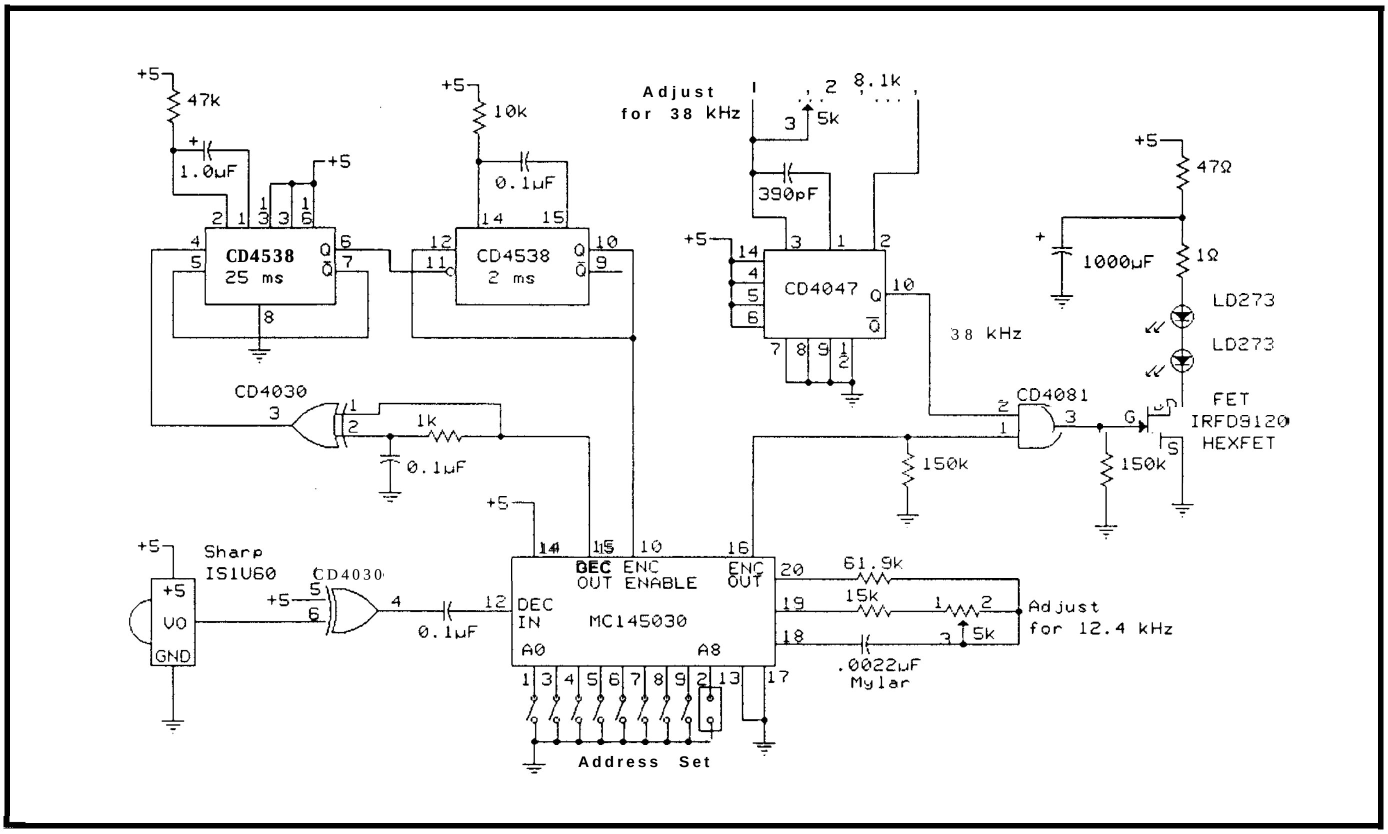


Figure 6—*An* intelligent *IR* badge *has a unique code set on* ***its switches*** *and* ***transmits its*** *code only when it hears* ***the*** *same code* ***sent*** *to it*

EXPRESS CIRCUITS

MANUFACTURERS OF PROTOTYPE PRINTED CIRCUITS FROM YOUR **CAD** DESIGNS

TURN AROUND TIMES AVAILABLE FROM 24 HRS — 2 WEEKS

Special Support For:

- TANGO.PCB
- TANGO SERIES II
- TANGO PLUS
- PROTEL AUTOTRAX
- PROTEL EASYTRAX
- smARTWORK
- HiWIRE-Plus
- HiWIRE II
- EE DESIGNER I
- EE DESIGNER III
- ALL GERBER FORMATS

- FULL TIME MODEM
- GERBER PHOTO PLOTTING

NEW!

- WE CAN NOW WORK FROM YOUR EXISTING ARTWORK BY SCANNING. CALL FOR DETAILS!

Express Circuits

314 Cothren St., P.O. Box 58
Wilkesboro, NC 28697

Quotes:
1-800-426-5396
Phone: (919) 667-2100
Fax: (919) 667-0487

#115

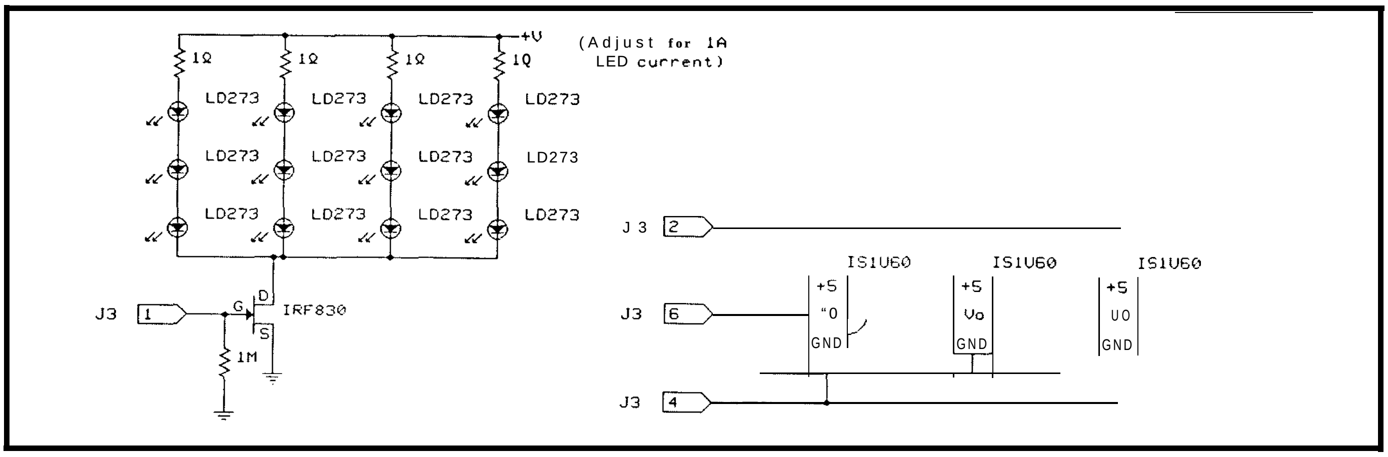


Figure 7—*In order to cover a whole room with IR transmissions and receive badge responses, a group of IR LEDs is used with several IR receivers. These extra components are connected to the basic IR-Link board to boost its transmitting and receiving power.*

different directions to cover a wider area. However, as a practical matter, additional IR LEDs should not be connected in parallel with the LD273 on the IR-Link board because there is only a single 2N2907 driver. Instead, the 6-pin header brings the raw PCM signal out. This feature should be used to drive a separate external transistor or FET.

Photo 3 shows a prototype expansion transmitter/receiver that I built. It contains three IR receivers and 12 dual-element LEDs. Using the prototype's own power supply and an IFR830 FET driver, I was able to increase the radiant power output from about 20 mW (on the single IR-Link LED) to over 4 watts! A couple transmitter/receiver "pods" like these could cover a large office or conference room quite adequately.

IN CONCLUSION

Well, this ends my part of the new Circuit Cellar HCS II description. Ed will talk a little more next month about the remaining links but I'll be busy wiring up a storm. If you remember the picture in the last issue [page 32) of my old HCS, you can compare it to the new layout here in Photo 4. Okay, wires are wires.

Showing you this picture will probably create more questions, because I took considerable engineering and poetic license in the physical assembly of my system. From the very beginning, I have claimed that HCS II

The following is available from:

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

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

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

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

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

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

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

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

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

Item 5. LCD-Link Smart 4 x 20 LCD Display Network Interface
PC board and all components including 8031 processor, RAM, IC sockets, LCD-Link firmware in EPROM, and user's manual.

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

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

Complete kit	order ADIOLNK-1K	$199.00
Assembled and tested	order ADIOLNK-1A	$299.00
4-channel, b-bit DAC upgrade		$35.00

Miscellaneous

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

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

was an industrial control system disguised and downsized as a "consumer-level" home controller (albeit a consumer with an engineering degree]. Because of my susceptibility to lightning and general interest in expanding beyond the basic limits I've described to you, I implemented my HCS II in its industrial guise so I could utilize industry-standard isolated I/O. The large green cards in the center of the wall contain optoisolators and relays. They connect by ribbon cable to a card cage on the right, which contains the SC and battery-backup electronics.

I'll keep you posted on future developments. I've already started looking at a speech I/O capability for the HCS II. I've selected a fine audio digitizer that can sound as good as anything annunciated by the ***Enterprise's*** computer but I'm having trouble finding a recognition unit. I find it hard to believe that no one has introduced a cost-effective voice recognition unit beyond the designs I did using the SP1000 chip almost eight years ago. I suppose if I have to I could resurrect an old Lis'ner 1000 and nail an Apple II to the wall. If you manufacture a voice recognition board or know of a workable system, tell me about it so I can get this project off the boards.

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

SOURCES

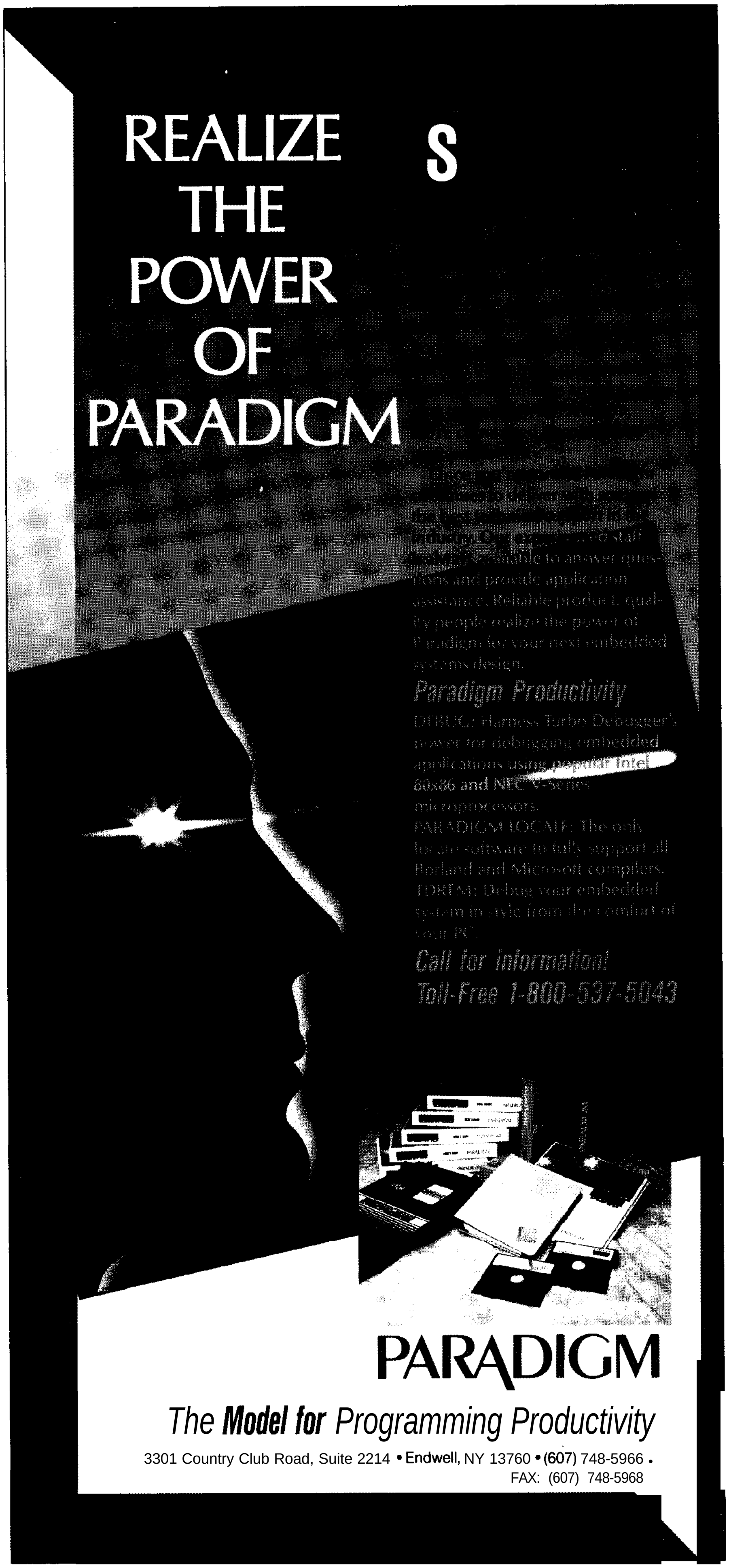
See the box at the left for the availability of HCS II modules.

Many of the components used in this article may be purchased from:

Pure Unobtainium
89 Burbank Rd.
Tolland, CT 06084-2416
Voice/fax: (203) 870-9304

I R S

404 Very Useful
405 Moderately Useful
406 Not Useful



REALIZE THE POWER OF PARADIGM

[illegible] industry. Our experienced staff [illegible] available to answer questions and provide application assistance. Reliable product quality people realize the power of Paradigm for your next embedded systems design.

Paradigm Productivity

DEBUG: Harness Turbo Debugger's power for debugging embedded applications using popular Intel 80x86 and NEC V-Series microprocessors.

PARADIGM LOCATE: The only locate software to fully support all Borland and Microsoft compilers.

TDREM: Debug your embedded system in style from the comfort of your PC.

Call for information!
Toll-Free 1-800-537-5043

PARADIGM

*The **Model for** Programming Productivity*

3301 Country Club Road, Suite 2214 • Endwell, NY 13760 • (607) 748-5966 • FAX: (607) 748-5968

The Frugal Networker

A Crosspoint Switchboard for RS-232

How many times have you wanted to be able to easily switch two or three computers between, say, a printer, two modems, and a microcontroller? Find out how to do it easily using a telephone chip.

FEATURE ARTICLE

Frank Cox

Did you ever have one of those problems that could be easily solved by buying, say, $1000 worth of new computer hardware and software? Well, it happens to me more often than to most. But instead of getting out my credit card, I usually react to these situations by heating up my soldering iron [not that I always save money). Sometimes I come up with something really useful that more than solves the problem at hand and that I'll still use when I finally do buy a modern computer. That's what happened with the Frugal Networker, a crosspoint switchboard for the RS-232.

I'm still using a 1980 vintage TRS-80 Model I (and am not ashamed to admit it!). I also have a serial terminal, modem, and a few other serial devices, all of which need to be cross-connected in different ways at different times. At first I used a bunch of toggle switches to change connections (and who hasn't played the game of swapping cables all over the place?). As I added new devices, I had to rearrange my switch setup. There had to be a better way.

I recently bought a microcontroller board from New Micros Inc. built around the F68HC11 Forth chip, which is a Motorola 68HC 11 with a complete Forth language in the chip's ROM. The 68HC11 is an interesting and powerful chip that has been well described in several recent ***Circuit Cellar INK*** articles.

Forth is a minority language. Some people love it to the point of fanaticism, or at least that's what the people who hate it say. Most don't know much about it, and I feel the latter group is missing out on a useful tool. In fact, the main reason I still use my TRS-80 is because it has a good Forth system. But back to my problem.

The controller board also has an RS-232 port on it. I started to add some more toggle switches to my original network to handle the new serial device, but the number was getting to be a bit much. My new board had just presented me with my next project.

THE GREAT CHIP SWITCH

Chips designed for one purpose often find homes in other kinds of applications. I chose the Mitel Semiconductor MT8809 for my project. It's an 8 x 8 analog crosspoint switch designed for use in telephone switching equipment. I'll describe using it with my F68HC11, but there should be little trouble adapting it to any system.

The switch contains an array of 64 CMOS transmission gates arranged as a matrix of eight columns by eight rows. The columns are called Y I/Os and the rows are called X I/Os (see Figure 1). A corresponding set of 64 latches acts as a control memory for this array. These latches in turn are controlled by a 6-to-64-line decoder. When interfacing it to a controller, all you need to do is treat it much like a 64-byte by l-bit memory device.

Data on the DATA input is asynchronously written to the latch selected by the address on AXO-AY2 whenever • CS and *STROBE are held low, and it's latched on the rising edge of *STROBE. A "1" written to an address turns on the corresponding switch and a "0" turns it off. Only the crosspoint switch being addressed at the time is changed by a given write, and any combination of X and Y I/Os can be interconnected by giving the right sequence of write commands. Bringing the • RESET pin low resets all latches to "0" and turns off all the crosspoint switches.

One thing to note is that the control memory is a WOM (for Write Only Memory). The control latches can be written but not read. If software needs to know the state of the switches, some method of "remembering" all the writes since the last reset is needed. I'll talk about some approaches to this problem later.

Figure 2 lists some of the MT89809's specs. Notice the switches have good frequency response and low distortion. They should have no problem handling any speed RS-232 signal. They would also make a good sound switcher. In fact, other members of this chip's family have a third power supply pm so they can be used with bipolar analog signals. Designing an automated sound-mixing board using this type of chip might be fun.

Another interesting application shown in the Mitel handbook is a "Test Equipment Switching System," which connects a rack of electronic test equipment to a number of points in a circuit under test. You could use these chips to make an automated test setup. See Mitel's ***Analog Communications Handbook*** for more information on this and other telephone-oriented chips.

THE DESIGN

My first thought was to run the RS-232 signals right through the crosspoint switches. This arrangement would make a neat little one-chip-plus-"glue" design and would work fine as long as the signals didn't exceed the power supply voltage to the transmission gates. Unfortunately, the Mitel devices only handle a maximum of 12 volts peak-to-peak and an RS-232 can be as much as 30 volts. Also, I wanted to plug my circuit into the expansion jack of a microcontroller board with only 5 volts available.

I decided to use a pair of MAX238s so I could run the transmission gates at 5-volt logic level. These are quad 5-volt-only RS-232 line drivers and receivers from the wizards at Maxim. This choice gave me eight receivers that can withstand a full 30 volts peak-to-peak and eight line drivers that can put out a "legal" RS-232 signal using on-chip "charge pump" DC-DC converters.

A MAX238 needs four external capacitors for its power supply, so I used a total of eight in my prototype, but two chips can share larger "V+" and "V-" caps as shown in the schematic, according to a Maxim application note. Maxim has since come out with the MAX235, which has these capacitors inside the package. They had to use a 0.6-inch-wide package though, so it wouldn't save any board space but it would simplify board layout and should also increase reliability. I suggest you try them if you want to reproduce this design.

The 68HC11 doesn't separate memory and I/O spaces like on Z80s and Intel chips, so I needed to find a good place in the memory map for the MT8809's 64 bytes. The New Micros board has a ● MEMDIS pin on the expansion jack that makes this search easier. Pulling 'MEMDIS low disables RAM, allowing a section of memory to be "notched out" by an external device. A neat feature. The New Micros chip moves the F68HC1 l's 64 configuration registers to B000 and its 5 12 bytes of EEPROM to B600. The 1536-byte "island" between them seemed like a good place to put my switcher. I used a 74HC688 plus a 3-input NAND gate from a 74HC10 to build my address decoder, which gives a lot of flexibility to move the address around later if I want to.

With a decoder giving a 'CS and ● MEMDIS signal each time my device is addressed, all I needed was the proper *STROBE whenever a write took place. Using the remaining two gates from the 74HC10 to combine the 68HC1 l's R/*W and E signals took care of this requirement.

Figure 3 shows the schematic of the circuit. U2 and one gate from U1 form the address decoder that responds to a 64-byte block of memory to generate ● CS for U3 and 'MEMDIS for the controller. Changing the jumpers on the B inputs of U2 will move this block to the upper or lower 64 bytes of any memory segment that is a multiple of 256. This block of memory starts at hex address B5C0 and ends at B5FF, the last byte before the F68HC 1 l's EEPROM. The remaining two gates in U1 combine the F68HC1 l's R/*W and E signals to create a data *STROBE for U3. A0 through A5 go to U3 and select one of 64 addresses, and DATA comes from the HC 1 l's DO line.

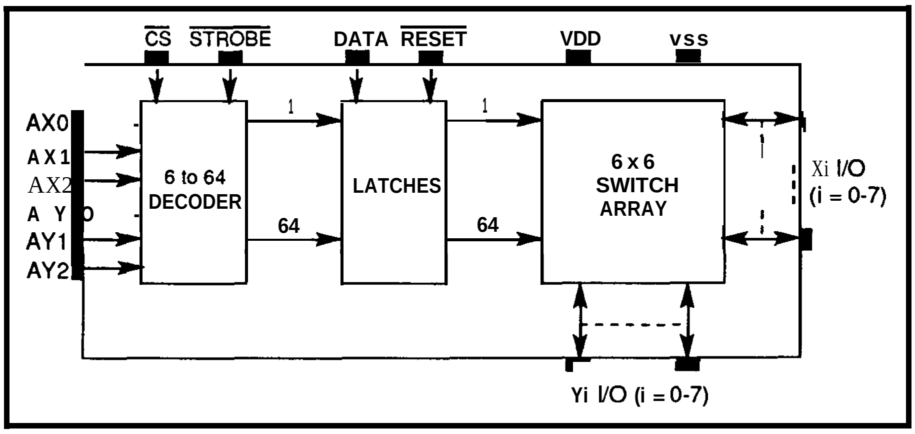


Figure 1—*The Mitel MT8809 has eight inputs and eight outputs that may be cross-connected in any combination.*

The *RESET pin of the MT8809 is tied to the F68HC11 board's *RST line, so the whole system is reset at once. U4 and U5 are the Maxim quad 5-volt-only receivers and drivers. The receivers are connected to the X I/Os of the crosspoint switches and the line drivers are connected to the Y I/Os.

THE PHYSICAL DESIGN

The New Micros board uses a vertical stacking-type connector for the expansion jack, so it was natural for me to piggyback my board on top of theirs. I used modular phone-type jacks for my RS-232 I/O and found that eight of these jacks side by side are exactly the same width as the controller board! These connectors are taller than the distance between the boards, so I just hung them over the edge.

Now when I say "RS-232," what I really am describing is the two serial lines and signal ground. I've been blissfully living my life without ever using the other signals that are part of a real RS-232 connection, and I've avoided most of the confusion associated with this messy standard. However, I realize not everyone can avoid doing so. I used 4-pin jacks for this project and just made up the way they are wired because there is no standard for wiring RJ- 11 jacks in RS-232 applications. In retrospect, I should have used 6-pin RJ- 11 **s** just in case I need the other pins some day. See **"ConnecTime"** in ***Circuit Cellar INK,*** issue **#24,** for a good discussion on the use of RJ-11 jacks in RS-232 connections.

I built everything up on **0.** 1"-grid prototyping board with a ground plane on one side, gold-plated through-holes, and pads on the other side. It probably would work without the ground plane, but I feel safer using it. I wired the circuit point-to-point.

THE SOFTWARE

As I mentioned previously, the **F68HC11** has a complete language in 8K of on-chip ROM. This New Micros version of Forth is called Max-Forth. Like most Forths, it's an integrated compiler, interpreter, assembler, and operating system. I won't go into its many features here, but I would encourage anyone who hasn't checked out Forth to do so. Forth is an excellent microcontroller language because of its power, small size, and ease of development.

I'll just present a simple application program to show you how the Frugal Networker works. Although I suspect most readers don't know Forth, I think these examples will be understandable to most programmers. If you try this circuit with another language, I hope the algorithms I provide will help you.

To begin, I have eight jacks, each with an incoming and an outgoing signal line. Normally, I would want to connect two devices together by connecting the transmitter of one to the receiver of the other and vice versa. But doing so would mean turning on two switches, and I'll want to connect the attached devices together in different ways to do different jobs. I don't want to turn on and off different configurations of switches. Being able to type something such as **"PC modem CONNECT"** and **"terminal PC DISCONNECT"** would be nice, as would being able to group commands. For example, **"WELL"** could mean "connect the terminal to the modem and also let the computer listen, then dial the WELL and ignore all traffic on this line until I tell you to turn everything off again." Doing this sort of thing is easy in Forth, and having an on-line language system means you can build commands like this example as you need them.

I first need to build a few "primitives," though I'm not sure this description is the best one for a Forth word. To turn a switch on and off, I need to write a "1" or a "0" to the respective corresponding address. So in Forth, I could type

```
HEX  B5C0 C! <ENTER>
```

H EX will turn on hexadecimal mode and C **!** (pronounced "C Store") will write an 8-bit value of 1 to hex address B5C0 and connect the input of jack **J1** to its output making a loop-back for anything plugged in there.

Now that the hex arithmetic is set I can type

```
0 B5C0 C!
```

to turn it off again.

I've already mentioned the **write**-only nature of the **MT8809.** One way around the problem of being unable to read the state of the switches is to just turn them all off before each new setup. I'll define a Forth word to do

	PARAMETER		SYMBOL	MIN	MAX	UNITS
1	Supply Voltage		V_{DD}	-0.3	15.0	V
			V_{SS}	-0.3	$V_{DD} + 0.3$	V
2	Analog Input Voltage		V_{INA}	-0.3	$V_{DD} + 0.3$	V
6	Package Power Dissipation	PLASTIC DIP	P_D		0.6	W
		CERDIP	P_D		1.0	W

	CHARACTERISTICS	SYM	MIN	TYP	MAX	UNITS	TEST CONDITIONS
1	Quiescent Supply Current			120	400	mA	All digital inputs at $V_{IN} = V_{SS}$ or V_{DD} except $\overline{RESET} = V_{SS}$

	CHARACTERISTICS		SYM	25°C		70°C		85°C		UNITS	TEST CONDITIONS
				TYP	MAX	TYP	MAX	TYP	MAX		
1	On-state Resistance	$V_{DD} = 12V$	R_{ON}	45	65		75		80	Ω	$V_{SS} = 0V$, $V_{DC} = V_{DD}/2$, $\lvert V_{Xi} - V_{Yj} \rvert = 0.4V$
		$V_{DD} = 10V$		55	75		85		90	Ω	
		$V_{DD} = 5V$		120	185		215		225	Ω	
2	Difference in on-state resistance between two switches		DR_{ON}	5	10		10		10	Ω	$V_{DD} = 12V$, $V_{SS} = 0$, $V_{DC} = V_{DD}/2$, $\lvert V_{Xi} - V_{Yj} \rvert = 0.4V$

	CHARACTERISTICS	SYM	MIN	TYP	MAX	UNITS	TEST CONDITIONS
3	Frequency Response Channel "ON" 20LOG (V_{OUT}/V_{Xi}) = -3dB	F_{3dB}		45		MHZ	Switch is "ON"; $V_{INA} = 2Vpp$ sinewave; $R_L = 1k\Omega$
	Total Harmonic Distortion	THD		0.01		%	Switch is "ON"; $V_{INA} = 2Vpp$ sine wave f = 1 kHz; $R_L = 1k\Omega$
	Feedthrough Channel "OFF Feed. = 20LOG (V_{OUT}/V_{Xi})	'DT		-95		dB	All Switches 'OFF"; W_{NA} = 2Vpp sine wave f = 1 kHz; $R_L = 1k\Omega$.
	Crosstalk between any two channels for switches Xi – Yi and Xj – Yj. Xtalk = 20LOG (V_{Yj}/V_{Xi})	(tall		-45		dB	$V_{INA} = 2Vpp$ sine wave f = 10MHz; $R_L = 75\Omega$.
				-90		dB	$V_{INA} = 2Vpp$ sine wave f = 10kHz; $R_L = 600\Omega$.
				-65		dB	$V_{INA} = 2Vpp$ sine wave f = 10kHz; $R_L = 1k\Omega$.
				-60		dB	$V_{INA} = 2Vpp$ sine wave f = 10kHz; $R_L = 10k\Omega$.

Figure 2—*Specifications* **for he** *MT8809* **show** *that,* while *the chip* was designed for voice switching, *it* a/so *works* well *for switching serial lines.*

this action, but first I'll define a constant named XSWITCH. Thus,

```
B5BF CONSTANT XSWITCH
```

Recall from the above discussion that B5BF is 1 byte before the first address of the Frugal Networker. That way XSWITCH+1 is the first address of my switcher. Actually, it would be XSWITCH 1 + because Forth, like a Hewlett-Packard calculator, uses stack-oriented "postfix" arithmetic. Now to define the word ALL.OFF:

```
: ALL.OFF  XSWITCH
     40 0 DO
          1+ DUP 0 SWAP C!
          LOOP  DROP
```

For non-Forthers, I'll explain that the : and ; delineate a Forth definition. Typing in the above code or loading it from mass storage will compile it into memory as a word that will execute whenever called. Invoking the word ALL.OFF will zero all 64 crosspoint switches and turn them "all off."

The 40 and 0 are the loop indices in hex. The word 1+ increments the address XSWITCH, which is on the stack. Then, 0 is put on the stack and SWAPped into proper position for the word C!. After performing these steps 64 times, the leftover address, now XSWITCH + 64, is DROPped. I know this process is a little messy, and you Forth programmers are thinking I'm nuts for not using the much faster and shorter FILL or ERASE commands. Well, I would have used them but the MT8809, or at least the one I have, doesn't seem to want to take the data that fast.

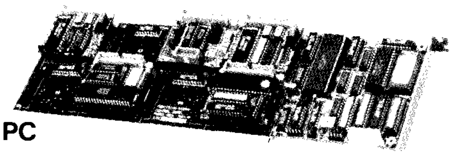
Now I'll define some words such as 1 2 ON, which will let me connect input 1 to output 2, or something similar like 1 2 OFF. But first I should do a reality check; I can only allow the numbers 1 through 8. I'll make this limit with two words.

```
: RANGE?   DUP 1< SWAP 8 > OR
  ABORT' <need 2 numbers 1 to 8>";
```

RANGE? makes a copy of the top of the stack and if it's not 1 through 8,

Position and/or Velocity

Motion Control



DCX-PC

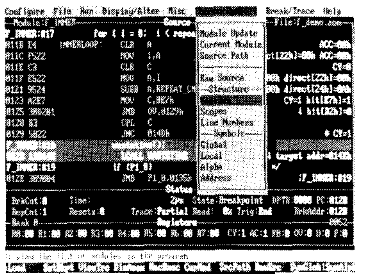
8-Axis Programmable Motion Controller

- **State-of-the-art Digital Muhifunction Controller can be configured in minutes using "off -the-shelf" components**
- **DCX motherboard contains intelligence/memory/firmware and 8 sockets for "plug-in" Modules**
- **1 to 8 "plug-in" DC Servo, Stepper or Special Purpose Control Modules can be mixed/matched on same DCX board**
- **Install in any PC/XT/AT compatible, or use "stand-alone" with RS232 and/or IEEE-488 interface Modules**
- **High level interface libraries in "C" and "BASIC," with examples and source code, included**

Precision Micro Control
CORPORATION
8122 Engineer Road, San Diego, CA 92111
(619) 5654500   FAX (619) 565-1511

#116

8051 68HC11 COP8



iceMASTER™

Your Window To Emulation Productivity

Rental And 10-Day Trials Available

- **iceMASTER delivers productivity:** easy to learn, easy to use and fast!
- **Hyperlinked On-line help** guides you through the emulotion process.
- **iceMASTER is FAST! The** 115.2K baud serial link keeps typical downlood times to under 3 seconds using a stondord COMM port!
- **iceMASTER is Versatile:** iceMASTER- 51, iceMASTER-68HC11 and iceMASTER-COP8 support most family derivatives.
- **Flexible user interface: you** ran completely configure the windows for size, content, location and color.
- **iceMASTER is convenient!** It connects easily to your PC, requires no disassembly, nor does it take up any expansion slots. It works on ony PC (DOS or OS/2), Micro Channel or EISA. Even laptops!
- **Supports source level debug** (C and PL/M) and source level trace. 4K trace buffer with advanced searching and filtering capabilities.
- New **Products!** 68HC11 A, D, E, F; 8XC552; 8XC528; 8XC515A and 8XC517A

HOT NEW PRODUCTS!

Call today for FREE DEMO DISK!
Call today to ask about FREE 8051 Macro Assembler!

(800) METAICE (800) 638-2423

MetaLink Corporation

MetaLink Corporotion P.O. Box 1329 Chandler, Az 85244-1329 Phone: (602) 9260797 FAX: (602) 926-1198 TELEX: 4998050MTLNK

#117

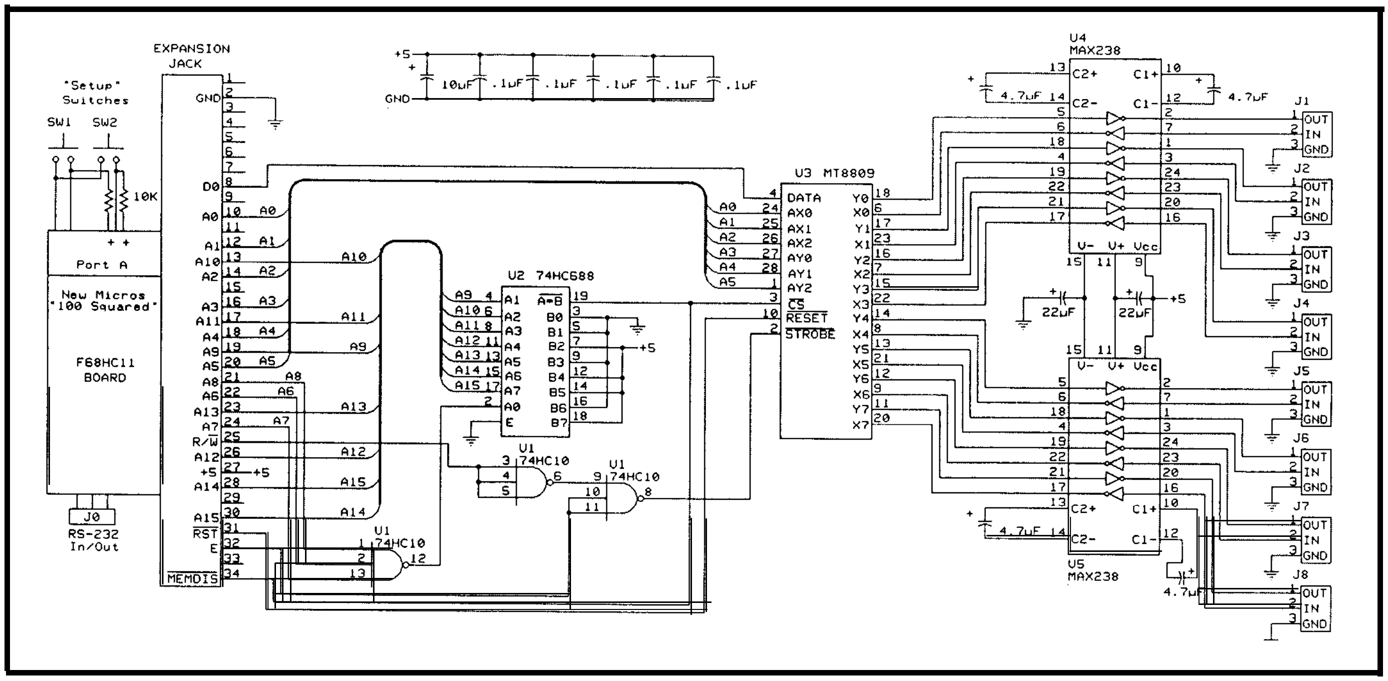


Figure 3—*The Frugal Networker uses an off-the-shelf micro, telephone cross-point switch, and some level converters to make a smart RS-232 switch.*

ABORT" will send the message to the terminal, empty the stack, and return to the prompt averting the danger.

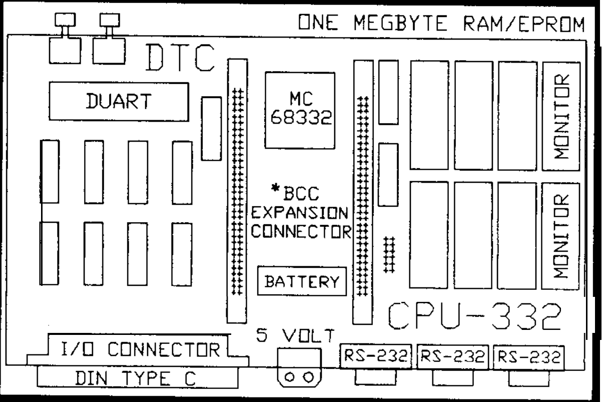
```
: CHECK DUP RANGE? OVER RANGE? ;
```

CHECK will RANGE? the top two numbers on the sack. If either is out of range or if there is only one of them, then it's ABORT" time. If all is well, then they are left as they were found.

I need to take these two numbers and make one number that I can add to the base address of our crosspoint switcher to turn on the desired switch. The matrix of numbers I need to add to

68332 68HC16

HIGH NOISE IMMUNITY



$595 QUANTITY ONE

Our FOUR layered CPU and I/O boards are designed for tough factory floor environments. Custom design and board mfg. available. Please call for specifics.

*Motorola *Business Card Computer*

TEL (510) 475-8147 • FAX (510) 489-5356

33476 Alvarado-Niles Blvd.. Suite 6
Union City, CA 94587

#118

The $595 Solution

to* 8051 *System Development

PDK51

The PDK51 is a fully integrated hardware, firmware, and software system designed to help you develop your products quickly and cost effectively.

All you need to use the PDK51 is an IBM-PC/XT/AT or compatible. We supply the rest.

- *SIBEC-II microcontroller board (8052AH-BASIC CPU) with 16K RAM*
- *BASIC interpreter source on disk*
- *SIBEC-II hardware manual with schematic*
- *Monitor/debugger ROM plus manual*
- *Monitor/debugger hardcopy source listing*
- *UL listed 5V power supply*
- *Programming power supply adjustable from 4V to 26V*
- *KERMIT communications program*
- *SXA51 8051/8052 cross assembler*
- *BTK52 BASIC programmers toolkit with tutorial*
- *160 page BASIC Programming Manual by Systronics*
- *RS232 cable*
- *PDK51 tutorial*
- *Optional: aluminum case -$45, monitor/debugger source—$25, battery-backed real-time clock- -$39.*
- *All components are manufactured to exacting standards and warranteed for one year.*

PDK51 PLUS includes everything in the PDK51 plus Vers. 3 of out popular BXC51 8051/8052 BASIC compiler-$800.

Call Now! 603-469-3232 or FAX 603-469-3530

Binary Technology, Inc.

Main Street . PO Box 67 . Meriden, NH 03770

#119

XSWITCH, or B5BF, is, in decimal,

```
                  in
1   2   3   4   5   6   7   8
----------------------------:
1   2   3   4   5   6   7   8 : 1
9  10  11  12  13  14  15  16: 2
17  18  19  20  21  22  23  24: 3  o
25  26  27  28  29  30  31  32: 4   u
26  34  35  36  37  38  39  40: 5    t
27  42  43  44  45  46  47  48: 6
28  50  51  52  53  54  55  56: 7
57  58  59  60  61  62  63  64: 8
```

and I can use

```
    : CALC    8 *     8ROT     ;
                               15
```

Looks unfathomable? Nothing to it really. Remember, I'll have two numbers on the stack, 1 through 8, "in" and "out." **CALC** multiplies the out value by 8, leaving 8, 16, 24, 32, 40, 48, 56, or 64. These numbers make up the last column on the table. I put an 8 on top of the stack and ROT the in value up from third to the top of the stack and subtract it from 8. Then I subtract this result from the result of multiplying the out value by 8 and I'm done. This math works the same way in hex, but the table looks stranger.

Let me illustrate what I've just explained.

```
3 5 CALC--> 8 * 5 is 40
            8 ROT  - gives 5
            40 - 5 = 35
```

From the table you can see that 3 in, 5 out is in fact 35. Bingo. Now you're ready for

```
:ON CHECK CALC XSWITCH + 1
                        SWAP C! :
:OFF CHECK CALC XSWITCH + 0
                        SWAP C! :
```

The SWAP is needed to put the address and value in the proper order for finicky C !.

I am now able to type in things like 1 2 ON and make new words by

```
: SETUP1 ALL.OFF 1 8 ON 8 1 ON ;
: SETUP2 ALL.OFF 1 5 ON 5 1 ON
                 1 7 ON  7 1 ON :
```

I'm sure you get the idea. I'll also make two other words:

```
: CONNECT 2DUP ON SWAP ON ;
: DISCONNECT 2DUP OFF SWAP OFF ;
```

1 2 CONNECT **is** thesameas 1 2 ON 2 1 ON, andsimilarlyfor DISCON-N **ECT.** At this point, I can add a few other features to this little application. For example, I'll declare the jacks that my various devices are plugged into as constants. Thus,

```
5 CONSTANT modem 6 CONSTANT PC
7 CONSTANT terminal
8 CONSTANT 100s
```

That **last one,** by the **way,** is the **New** Micros "100 Squared" board. The controller's RS-232 port can be plugged into one of the jacks so it can switch itself around the network. Even though these are called constants, their values can be changed later with the help of the word ' or TICK.

TIRED OF WAITING FOR THE PROMPT ?

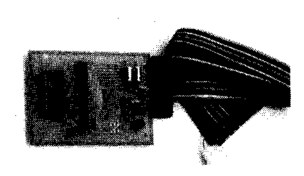
DOS IN ROM!

Speed up with a ROM DRIVE! Boots DOS and programs instantly. Also used to replace mechanical drive completely in controllers or diskless workstations. The only perfect protection from viruses. Easy to install half-size card.

MVDISK1 64k.....$95
MVDISK2 360k....200
MVDISK3 1.44m...300

$95

Quantity discounts!



$95 EPROM PROGRAMMER

-PLUGS INTO PRINTER PORT
-DO ROMS FOR SBC OR MVDISK
-PROGRAM 2764/27010 EPROMS

WORLDS SMALLEST PC !!!

ROBOTS ALARMS RECORDERS DOS

8088 SINGLE BOARD COMPUTER

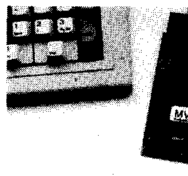
THREE EASY STEPS:
1. Develop on PC
2. Download to SBC
3. Burn into EPROM

$95

-2 PARALLEL -LCD INTERFACE
-3 SERIAL -KEYBOARD INPUT
-PC TYPE BUS -REAL TIME CLK
-BIOS OPTION -BATTERY OR 5V

FREE SHIPPING IN U.S.

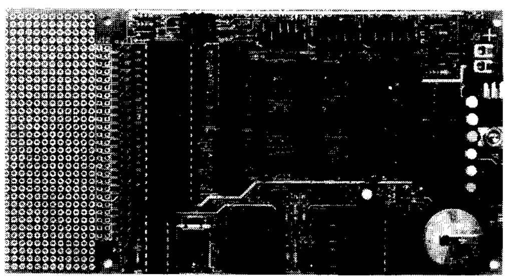
5 YEAR LIMITED WARRANTY



MVS Box 994 Merrimack, NH (508) 792 9507

#120

EMBEDDED CONTROLLER
for Quick Development



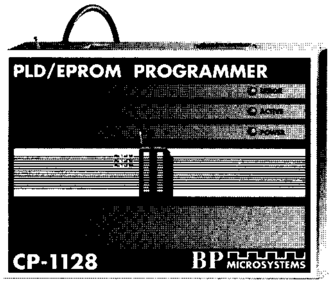
The EC-32™ is a versatile 80C32 microcontroller board. It is ideal for quickly developing products, prototypes or test fixtures.

- 80C32 microcontroller (805 1 compatible)
- BASIC-52 or MONITOR-52 available
- Program in **C,** BASIC or assembly language
- 8 to 92K RAM, EPROM or EEPROM
- Breadboard area and expansion bus
- RS-232 port and 12 digital I/O lines
- $100 for 11 MHz, $145 for 20 MHz

Iota Systems, Inc.
(702) 831-6302
POB 8987
Incline Village, NV 89450

20 MHz Board Available

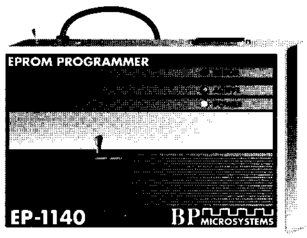
#121



CP-1128
Combination Programmer
$1295.00

- ✔ Supports AMD's MACH110/210/120/130/230 EPLDs, Altera's 900-, 1800-series and MAX EPLDs, Cypress' CY7C361, Lattice's isp LSI1032 & pLSI1032, National Semiconductor's -5D & -7D devices ond MAPL devices.
- ✔ Qualified and recommended by AMD. Lattice. National Semiconductor, Signeticsond others.
- ✔ Utilizes only manufacturer approved programming algorithms.
- ✔ Supports upto 28-pin E/EPROMs and bipolar PROMs including the microwire I^{2}C devices. /Supports Dallas Semiconductor NVRAMs and TI DSP320, Microchip PIC microcontrollers.
- ✔ Lifetime FREE software updates available via BBS and US Mail.
- ✔ Call for a DEMO disk and literature pack.
- ✔ Made in the USA

● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●



EP-1140
E/EPROM & µcontroller
Programmer
$895.00

- ✔ The EP-1 140 supports NEC's 27C8001, 8-Mbit EPROM, all 27C240 4-Mbit 16 bit EPROMs, FLASH EPROMs, NVRAMs and all microwire EPROMs.
- ✔ All Intel, AMD, ond Signetics 40-pin µcontrollers supported directly.
- ✔ Qualified and recommended by Intel, Signetics, Notional Semiconductor, and others
- ✔ Lifetime FREE software up dotes available via BBS and US Mail.
- ✔ Risk-free thirtyday money-bock guarantee.
- ✔ Made in the USA

● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●

With lifetime FREE software updates, you can't go wrong and BP Microsystems offers o thirty-day money-back guarantee to ensure product satisfaction. Remember, BP Microsystems is...

The Engineer's **Programmer**™

BP MICROSYSTEMS

10681 Haddington • Houston, TX 77043.3239
(713) 461.9430 • FAX (713) 461.7413

© 1991 BP Microsystems, Inc.

#122

I need one more word before I can define the word WELL, which I mentioned previously. **STAND . BY** essentially says "do nothing until I tell you."

```
: STAND.BY
   BEGIN
     ?TERMINAL IF KEY 18 =
                 ELSE 0
              THEN
     UNTIL
```

This word loops or stands by until it sees an **18 hex or CONTROL-X.**

So if I want a command that dials the WELL and sets me up for an online session, I could use

```
: well
    100s    modem    ON
    ."ATDT 415 332 7398" CR CR
    100s modem      OFF

    terminal modem CONNECT
    pc modem        CONNECT
  STAND.BY    (for CTRL-X)
    terminal modem DISCONNECT
    pc modem        DISCONNECT
    terminal  100s  CONNECT
```

You probably recognize that **ATDT** as a command to the modem to dial a number.

Notice these few lines of code that I've shown have already built up a useful program. You can embellish it to best fit your application. One improvement would be a **STAND. BY** word that does more than just waste time looping. Max-Forth allows multitasking as one fancy way of improving it. Another alternative is to use some other interrupt-driven scheme. I'll leave that as an exercise for you.

Finally, I'd like to cover some other solutions to the WOM problem I mentioned earlier. There will be times when you won't want to disturb a connection by using **A L L . 0 F F.** What you can do is declare a 64-byte variable to store an image of the crosspoint matrix. Then you modify 0 N, 0 **F F,** and **ALL . 0 FF** to write to both places. You could write a neat little routine to display the results in, say, an 8 x 8 matrix. Another way would be to work out a compression and decompression scheme and store an image in only 8 bytes because there is really only 64 bits of data involved. This method is good if you want to store a lot of complicated "prerecorded" setups.

I hope you find the Frugal Networker interesting and that it leads you to other ideas using the MT8809 family. Also, I hope I've convinced you to use Forth as a small systems language. Just so I don't leave you with the impression that I'm a monomaniac concerning this subject, let me explain that I intend to use Pascal and C in upcoming projects, but as long as I have the Frugal Networker, I won't forget my Forth. ■

Frank Cox is back as a full-time student studying science and engineering with an emphasis on electronics after spending many years in industry.

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.

SOURCE

Many of the components used in this article may be purchased from:

Pure Unobtainium
89 Burbank Rd.
Tolland, CT 06084-2416
Voice/fax: (203) 870-9304

CONTACT

Mitel Semiconductor
4320 Stevens Creek Blvd., Suite 225
San Jose, CA 95129
(408) 249-2111
Fax: (408) 249-1635

I R S

407 Very Useful
408 Moderately Useful
409 Not Useful

FEATURE ARTICLE

Ken Davidson

Programming the Home Control System II

In the last issue of *Circuit Cellar INK*, members of the engineering staff introduced the Circuit Cellar HCS II. Find out now how to program the system to run your home.

Have you taken a good look at what is on the market these days for home automation systems? At the low end is the X-10 CP290 controller. You plug this box into a PC, set on and off times for a bunch of X- 10 modules, and disconnect the PC. While inexpensive, it is nothing more than an easy-to-program timer.

Next is the Enerlogic ES-1400. It uses a language similar to what I'm about to describe, but, again, it can only talk to X-IO modules. Its advantage over the CP290 is its capability to receive from the power line and base decisions on what it hears, but it has no direct inputs or outputs. How do you connect a motion detector, light-level sensor, or temperature sensor to it? There are X-10 units that send commands onto the power line based on motion and light level, but given the reliability of X- 10 transmissions, do you really want to trust them?

At the high end of the scale are the units that start in the $3000 range and can only be obtained through "authorized dealers" who must do all the installation and programming. What good are these devices to experimenters who know what they want to do, and just need some specialized hardware to do it!

There are a few low-cost systems available that allow the experimenter to add hardware piecemeal and do the programming, but they are based on the premise that you don't mind leaving your 200-W IBM PC running all the time to do the controlling. I wouldn't want to see that electric bill. And what happens when your kids want to play Tetris?

HCS II REVISITED

In the last issue of ***Circuit Cellar INK,*** Steve introduced the new Circuit Cellar Home Control System II and I discussed the brains of the operation: the supervisory controller. The HCS II is based on a number of separate building blocks that may be connected using a network made up of a single twisted-pair wire and is designed for people who know what they want to do and how to do it, but need some low-cost hardware and software to use as tools.

The Supervisory Controller *(SC)* is responsible for determining how the system operates and when things happen. It has 24 bits of TTL I/O (optionally up to 48 bits of buffered I/O in addition) and eight channels of 8-bit analog-to-digital conversion. The PL-Link module handles all X- 10 power line communication, including both transmitting and receiving. The IR-Link [which Steve and Ed discuss in this issue) allows you to issue commands with a hand-held infrared controller. The LCD-Link contains an LCD display that responds to a subset of standard ANSI terminal control sequences and also has four bits of I/O. The DIO-Link adds eight bits of TTL I/O, and its cousin, the ADIO-Link, includes 24 bits of I/O, eight channels of 8-bit analog-to-digital conversion, and four channels of 8-bit digital-to-analog conversion. Up to 32 of these modules may be connected to a single network, although cost will probably limit system size to fewer than that.

A complete HCS II system can be as simple as a lone SC, or as complicated as you want to make it.

COMMUNICATE THIS

While the HCS II is made up of several autonomous building blocks, you need some way of interacting with the system as a whole in order to program it. In any properly configured home control system, you should be able to set it and forget it, so we use an IBM PC compatible for doing the “setting,” then disconnect it when finished to do the “forgetting.”

The program used to communicate with the SC is called **HOST.** When run, **HOST** displays a number of windows containing such information as current time and date, current state of X-10 modules (you decide which housecodes), current state of local inputs and outputs, and what network modules are in use. **HOST** allows you to set a new time and date (it actually reads it from the host PC, so be sure you’ve set it correctly before running **HOST)** and allows you to load a new program into the SC. I’ll cover where that program comes from next.

THE LANGUAGE OF HOMEOWNERS

The original HCS that Steve presented about seven years ago had a simple menu-driven interface and control scheme. It gave the user several programming options including turning an X-10 module or direct output on and off at specific times, in response to an input, or after a specific time period. While that scheme is easy to use and allows a good degree of control, it falls well short of meeting the needs of a more sophisticated control system that may be applied to both industrial and home environments.

Suppose I have an area with two lights, two motion sensors, and a door sensor. I want the lights to come on if either of the motion sensors is tripped, then go off 15 minutes after no motion is detected. I also want the lights to come on if the door is opened and to stay on as long as the door is open, regardless of motion. When you have a system like the old HCS, where you could only key actions on a single input, such a scenario is impossible to realize without additional circuitry to combine the sensors external to the HCS. Remember this situation for later and you’ll see how easy it is to do with the new system.

The HCS II programming language is called XPRESS (expandable Programmable Real-time Event Supervisory System) and is based on what I call an “event equation.” The equation consists of an “ **I** F” section, followed by a **"THEN"** section. If the **I F** section is true, the **TH EN** section is executed. If not, the action is skipped. It’s as simple as that. Much of the power comes from being able to combine any number of conditions in the I F portion, and have any number of actions initiated in the **TH EN** portion.

a high state and an edge. A falling edge is tested by checking for both a low state and an edge. Inputs are broken into two categories: local inputs and network inputs. Local inputs [tested with the **I N P UT** keyword) are those connected directly to the SC and are the fastest to test. Any device that needs quick action (like a motion detector at the top of a flight of stairs) should be connected to one of the local inputs. Network inputs (tested with the N **ETB** I T keyword) are those found on the LCD-Link, DIO-Link, and

ADC(n) <op> c	n = analog-to-digital channel
	<op> = “=”,“>”,“<”,“>=”,“<=”
	c = constant g-255
Input(n)=ON/OFF/EDGE	n = input number O-239
IRcode(c) <op> n	c = received IR cede O-255
	<op> = “=”,“>”,“<”,“>=”,“<=”
	n = IR-Link module number O-7
Module(m)=ON/OFF	m = module A1–P16
NetBit(n)=ON/OFF/EDGE	n = network I/O bit number O-239
Output(n)=ON/OFF	n = output number C-239
Reset	True after reset
Time <op> hh:mm:dd	hh = hour O-23; mm = min O-59; dd = day
	SU,MO,TU,WE,TH,FR,SA,DY (daily)
	<op> = “=”,“>”,“<”,“>=”,“<=”
Timer(n)=ON/OFF	n = timer number O-63
Timer(n) <op> s	n = timer number 031
	<op> = “=”,“>”,“<”,“>=”,“<=”
	s = O-65535 seconds
Timer(n) <op> s	n = timer number 32-63
	<op> = “=”,“>”,“<”,“>=”,“<=”
	s = O-65535 minutes
Variable(n)=TRUE/FALSE	n = variable number O-15
Variable(n) <op> c	n = variable number O-15
	<op> = “=”,“>”,“<”,“>=”,“<=”
	c = constant g-255

Figure **1—*XPRESS*** condition keywords a/low the *testing of a* wide range *of sensor inputs and system variables*

CONDITIONAL CONTROL

Figure 1 shows a summary of valid **I F** statement conditions. I’ve done a lot of work on the system since the last article was written, and have expanded on what was presented there. I’ll quickly explain how each of the conditions works. [I know a laundry list of mostly self-explanatory commands can be dry, so I’ll try to be brief.) Note as you go through the list that many systems on the market allow you to base actions on time or on inputs, but rarely allow you to combine the two as we do here.

Inputs may be tested for a high state [on), a low state (off), or an edge (either rising or falling). To test for a rising edge, you simply check for both

ADIO-Link modules. Since these modules must be polled over the network, system response to a network input is somewhat slower than to a direct input.

Similarly, the current state of any output may be tested for either on or off. As with inputs, there are local outputs (tested with **OUTPUT)** and network outputs (tested, as with network inputs, with N **ETB I T). The** same issue of response time applies to outputs as to inputs.

Analog inputs may also be tested using the **ADC** keyword. A single keyword is used for both local and network ADCs because there are far fewer potential analog inputs than either digital inputs or outputs.

AllLightsOn(h)	h = housecode A-P
AllUnitsOff(h)	h = housecode A-P

DAC(n)=c	n = digital-to-analog channel
	c = constant O-255

LCD(n)="string"	n = LCD-Link number O-7
	string = any ASCII text string

Module(m)=ON/OFF	m = module A1–P16

Module(m)=DIM(n)	m = module A1–P16
	n = dim level l-31

Module(m)=BRIGHT(n)	m = module A1–P16
	n = bright level l-31

NetBit(n)=ON/OFF	n = network output bit O-239
Output(n)=ON/OFF	n = output number O-239
Refresh=r	r = refresh interval 0–99 minutes (0 = off)
Timer(n)=ON/OFF	n = timer number O-63
Variable(n)=TRUE/FALSE	n = variable number O-15

Figure 2—*XPRESS action keywords* give you *plenty of control* **over** your *living environment*

The current state of any X-10 module may be tested for either on or off using the MODULE keyword. There is no easy way to keep track of a lamp module's dimmed intensity, so we don't even try. A dimmed lamp simply shows up as being on.

The current time of day may be tested in a number of ways using the TIM **E** keyword. The typical =,>,>=,<, and <= operators may be used to compare the time with a constant made up of hour, minute, and day of week. There isn't any way to make comparisons with month or day of month, but we found little use for such comparisons and left them out.

There are many situations where elapsed time must be measured. A common example is turning a light on for a certain number of minutes when motion is detected, then turning it off. Sixty-four timers are defined for the HCS II: 32 that time in seconds and 32 that time in minutes. Each timer may be tested for whether it is on or off, and may be checked for whether it is less than, equal to, or greater than a constant value using the **TIMER** keyword. Before you think I'm crazy for providing 64 timers, keep in mind who the primary user of this new system is [Steve) and what his control requirements must be.

For those cases where a counter must be maintained or simple true and false states must be stored, there are 16 S-bit variables available.

When multiple IR-Links are connected to the system, knowing which IR-Link received a particular code or command from the IR remote is sometimes necessary. Using the **I RCODE** keyword, you can test whether a particular code was received by either a particular IR-Link or a range of IR-Links. For example, writing "I**F** **I** Rcode(**13**)=0" tests whether code **13 was** received by IR-Link number 0. You can *use* the information to produce different responses based on which room the transmission came from. When the IR-Links are used as part of a people locator system, knowing which IR-Link received the code tells you where a certain person is located. A statement like "I F IRcode(13)>=0" can be used to see if the code was received by any IR-Link. Steve's article in this issue covers the IR-Link in more detail.

Finally, setting up default states when the system is reset and continuing from a known condition is often useful. The **RESET** keyword tests true just once on the first pass through the event equations after a reset, then tests false from then on.

Now that you have all the conditions under your belt, it's time to combine them into useful tests. Any number of conditions may be combined in a single I F statement using **NOT, AND, OR,** and parentheses. A **NOT** preceding any of the above conditions complements the result. **AN Ds** and 0 **Rs** do pretty much as you'd expect. Some examples of valid conditions might include

```
IF Input(20)=ON AND
     Modul e(L11)=OFF
```

or

```
 IF (Input(10)=ON AND
      Output(21)=OFF) OR
    Timer(2)>30
```

JUST DO IT

On the other side of the coin is the action list following the **THEN** statement. Figure 2 lists all the valid actions. Any number of actions may be listed, either on separate lines or separated by semicolons. The list is always terminated with an **END** statement. As before, I'll quickly describe each action, trying not to be too arid.

Any X- 10 module may be turned on or off, dimmed, or brightened. Similarly, an "All Lights On" or "All Units Off" command may be sent out to any given housecode.

Outputs may be turned on or off. The same distinction is made between local and network outputs as for the I F tests. Analog outputs may also be set to any 8-bit value.

Timers may be turned on or off. When a timer is turned on, it is cleared to zero and starts counting. If you want to clear a timer that is already on, simply turning it on again will clear the count. Turning a timer off forces all tests on that timer to return a false response.

Variables may be set to any 8-bit value or to true or false. Like many

!	*Start* of comment
Begin	Start of program
Config <link> = n	<link> = "PL-Link","IR-Link","LCD-Link", "DIO-Link","ADIO-Link" n = 0–7
Define <label> = <expr>	<label> = alphanumeric string up to 32 characters <expr> = any valid program statement
Display Modules=<list>	<list> = X-10 module housecodes
GIF/GEND	Start and end of global IF

Figure 3—*Additional keywords allow* **you** *to configure your system on he fly and make programs more readable.*

Listing 1—*System actions may be based on any number of input conditions, making complex control scenarios realizable.*

```
!
! Example HCS II Program
!
CONFIG PL-Link = 1

DISPLAY Modules = A

DEFINE Lamp1 = Module(A1)
DEFINE Lamp2 = Module(A5)
DEFINE MotionA = Input(2)
DEFINE Motion6 = Input(6)
DEFINE Door-Sensor = Input(10)
DEFINE Basement-Timer = Timer(38)

BEGIN
  IF MotionA=EDGE OR MotionB=EDGE THEN
    Lamp1 = ON: Lamp2 = ON
    Basement-Timer = ON
  END

  IF Door_Sensor=ON THEN
    Lamp1 = ON: Lamp2 = ON
  END

  IF Basement_Timer>=5 AND Door_Sensor=OFF THEN
    Lamp1 = OFF: Lamp2 = OFF
    Basement-Timer = OFF
  END
```

Listing 2—*As Steve and Ed describe elsewhere in his issue, IR badges may be used with IR-Links to track people throughout a house and base decisions on where people are located.*

```
!
! Demonstrate use of IR badges for tracking
!
! Be nice to Steve. but razz Ed if he walks
!  in the room without Steve
!
CONFIG PL-Link = 1
CONFIG IR-Link = 3

DISPLAY Modules = A.B.C

DEFINE Steve = IRcode(0)    ! Badge number 0
DEFINE Ed = IRcode(2)       ! Badge number 2
DEFINE LivingRoom = 1       ! IR-Link number 1

DEFINE Lamp1 = Module(A1)
DEFINE Lamp2 = Module(A5)
DEFINE Stereo =Module(A8)
DEFINE Air-Horn =Module(A16)

BEGIN
  IF Steve=Li vi ngRoom THEN
    Lamp1 = ON; Lamp2 = ON
    Stereo = ON
  END

  IF Ed=LivingRoom AND NOT Steve=LivingRoom THEN
    Air-Horn = ON
    Lamp1 = OFF; Lamp2 = OFF
  END
```

8051 Family Tools

In Circuit Emulators

The DryICE Plus is a modular emulator designed so you can get maximum flexibility from your emulator purchase. The base unit contains all the hardware necessary to support pods containing many of the most popular members of the 8051 family of embedded control microprocessors. Buy one base unit, and select one or all of the pods you need to do the job at a much reduced cost. You get the same great functionality found in our popular DryICE 8031 emulator plus real-time Execute-to-Breakpoint, Line-by-Line Assembler, and much more. And the price is (almost) unbelievable! (Yes, it works with the Mac, too!)

Base Unit (w/RS-232 **IF) -- $299**
Available Pods: $149 each
8031/32, 80C31/32, 80C154, 80C451, 80C535, 80C552/562, 80C652, 80C51FA, 8751/52, 87C51/52.
Call about 87C751/752 support
16K Trace Buffer option: Avail. 2nd Qtr '92
Standard 8031 DryICE -- Still only **$199**
Enhanced 8031 DryICE -- **$269**

8051 Simulation

The 8051SIM software package speeds the development of 8051 family programs by allowing execution and debug without a target system. The 8051 SIMulator is a screen oriented, menu command driven program doubling as a great learning tool. **$99**.

Single Board Computers

8031SBC -A fast and inexpensive way to implement an embedded controller. 8031/32 processor, 8+ parallel I/O, up to 2 RS232 serial ports, +5 volt operation. The development board option allows simple debugging of 8031/51 family programs. **$99**ea

80C552SBC -10 bit 8 ch. A/D, 2 PWM, 1 RS232 & 2 RS232/422/485 serial ports, sockets for 64k ROM, 64k RAM, +5 volt operation; optional RT Clock w/ battery, 2k EEPROM. Development board version available. **Call for pricing!**

Call for your custom product needs. Free Quote - Quick Response

VISA MasterCard AMERICAN EXPRESS Card

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

(619) 566-1892

#124

other programming languages, false is simply zero while true is any nonzero value. Variables may also be incremented or decremented for use as counters. When a variable reaches zero, further decrementing has no effect. Likewise, when one reaches 255, incrementing does nothing. No other math besides incrementing and decrementing is supported.

Text messages may be sent to any LCD-Link in the system. A very useful subset of the ANSI cursor control commands has been implemented on the LCD-Link, and the SC allows you to send any of these commands to the display module.

Finally, the PL-Link's refresh period may be set. If you'll recall Ed's article in the last issue, the PL-Link may be set to send out on or off commands periodically to all modules it's referenced since its last reset. That way, any module accidentally triggered by garbage on the power line will be set back to its proper state. Setting the period equal to zero turns refresh off.

BELLS AND WHISTLES

Besides the basic language building blocks discussed above, there are additional commands and features that are used for configuration or to make the programmer's job easier. Figure 3 lists these extra keywords.

The SC must know what COMM-Link modules are connected to the network so it doesn't waste its time polling modules that aren't there. The CONFIG keyword is used to define how many of each of the COMM-Link modules are out there. If the program contains no CONFIG statements at all, the SC simply doesn't access the network.

HOST displays the current state of any X-10 module, local input, or local output you ask. The **DISPLAY** keyword is used to define just what housecodes you want displayed.

DEFINE is the key to making programs more readable. Instead of using keywords like "Input(2)" and "Module(L3)" when writing a program, DEFINE lets you give such keywords descriptive labels like "Kitchen_Motion" and "Bathroom_Light."

After all the definitions, a single BEGIN is used to denote the start of the program.

One last keyword used in programming is the global IF, or GIF. "Global" probably isn't quite the right word, but it seems to fit the best. GIF allows a single level of nesting of IF statements. The GIF keyword is followed by any valid combination of conditions. If it evaluates true, then any number of IF/THEN combinations following it up to the GEND are executed. If it evaluates false, the whole block is skipped. GIF allows you to selectively execute or skip

Listing 3—*In a more realistic example, you might control a set of lights based not only on motion but on time-of-day as well.*

```
! Practical example of different system behavior
! depending on the time of day
!
CONFIG PL-Link = 1

DISPLAY Modules = C,L

DEFINE Dark = 16:30:DY      ! Sunset
DEFINE Bedtime = 23:00:DY   ! Time for bed
DEFINE Morning = 6:30:DY    ! Starting to get light out

DEFINE BedroomMotion = Input(12)   ! Bedroom motion detector
DEFINE BedLight = Module(L4)
DEFINE CeilingLight = Module(L2)
DEFINE BedTimer = Timer(40)

DEFINE HallMotion = Input(10)      ! Hallway motion detector
DEFINE HallLight = Module(L8)
DEFINE HallTimer = Timer(41)

BEGIN
  GIF Time>=Dark AND Time<BedTime THEN
    IF BedroomMotion=EDGE THEN
      BedLight = ON; CeilingLight = ON
      BedTimer = ON
    END

    IF BedTimer>20 THEN
      BedLight = OFF: Ceilinglight = OFF
      BedTimer = OFF
    END

    IF HallMotion=EDGE THEN
      HallLight = ON
      HallTimer = ON
    END

    IF HallTimer>10 THEN
      HallLight = OFF: HallTimer = OFF
    END
  GEND
!----------------------------------------------
  GIF Time<Morning THEN
    IF HallMotion=EDGE AND HallLight=OFF THEN
      HallLight = DIM(12)
    END

    IF HallMotion=EDGE THEN
      HallTimer = ON
    END

    IF HallTimer>3 THEN
      Halllight = OFF: HallTimer = OFF
    END
  GEND
```

whole portions of a program depending on some condition. Check out some of the example programs I present later if you're still not clear about G I F.

Finally, comments are preceded by an exclamation mark (!). All text following the "!" up to the end of the line is skipped.

COMPILE TIME

The program actually run by the SC is made up of binary representations of the keywords I described above. I presented the basic format of that binary in the last issue. In order to translate from the English-like program developed by the user to the binary used by the SC, a compiler is necessary. Simply named C OM P I L E, the compiler runs on an IBM PC compatible and takes as input straight ASCII text entered with any text editor, does a syntax check, and generates a file called **EVENTS.** BIN containing the raw binary code used by the SC.

When the user running **HOST** presses the "L" key, **HOST** looks for **EVENTS. BIN,** loads it into memory, and sends it to the SC. The SC immediately starts running the new program and the **HOST** screen reconfigures itself assuming the transfer went all right..

PROGRAMMING CONSIDERATIONS

A key idea to keep in mind is that a program is made up of a series of event equations. The SC continually runs through the list over and over again. Any action that takes place near the top of the program could affect the evaluation of equations later in the list, so a certain precedence can be achieved by where in the list a particular equation falls.

Input levels and edges stay static throughout a pass through the list, changing only after a pass is complete. An input edge is always cleared at the end of a pass, but may be tested multiple times in a single pass.

One concept I'm sure is going to bite some people is once an equation evaluates true and its companion action has been carried out, that action won't be executed again until the equation evaluates false at least once, then true again. For example, suppose I have the statement:

```
IF Input(3)=ON THEN
  Module(J2) = ON
END
```

If I were to send out an on command to module J2 every time the condition Input (3)=ON evaluated true, I'd have a very busy power line until input 3 went off. Executing the on command just once, then skipping it until input 3 goes off, then on again is one way to get around the problem.

Another way I get around the above problem is to be a little smart about X- 10 commands. Transmitting an on command to a module if the module is already on is kind of silly. Therefore, I check the status table first, and if the module is on, I don't send another command. This check doesn't work for bright or dim commands, because the light will likely be already on when you want to dim or brighten it to a new level, so you have to make your program a bit smarter when using those commands to avoid repeat transmissions.

LEARNING BY EXAMPLE

I find the easiest way to learn something new is by practical application of the concepts. I've listed all the keywords and rules involved in writing a program, but a few simple examples should help clear up some of the haze in the air.

Let me go back to the scenario I cited earlier. I have two motion detectors, two lights, and a door sensor. Listing **1 shows one way a** program can be written to solve the control problem. The first equation looks for an edge on either motion detector and turns on the lights if it finds one. It also starts a timer. The next equation checks the door sensor and, if the lights haven't already been turned on by motion, it turns them on. The last statement checks the timer to find out if five minutes have elapsed. If so, and the door is closed, the lights are turned off and the timer is stopped. If the door is open, the lights stay on

A
Speed Development
MULTITASKING
Simplify Coding
CORE

The TAPIR Multitasking Package offers a complete kernel for developing and debugging multitasking code. Written for speed, this 100% assembly language program delivers:

- **True pre-emptive multitasking**
- **A debug module to examine register status of all tasks during runtime**
- **The ability to run with or without DOS**
- **No Royalties**

The TAPIR multitasking package is available for the NEC V and Intel 80x86 Series microprocessors for $595. (The complete source code is available for an additional $295)

For more information or to place an order, contact:

TAPIR
PO Box 1553
Seguin, TX 78155
(512) 372-2587

#125

until it is closed. Also note that if motion is detected again before the timer times out, the timer is restarted from zero. The result is the lights stay on for five minutes from the last detected motion, and not the first.

Since we're dealing with the IR-Link elsewhere in the issue, how about an example of performing one set of tasks based on the presence of a particular person, and another if that person is absent? Listing 2 shows such code. When Steve walks into the living room wearing his IR badge, the system will be nice and turn on the lights and stereo. If Ed happens to walk into the same room, but Steve isn't there, we'll razz him with an air horn and turn the lights off. If Steve is with him, though, we'll be nice again because we don't want to subject Steve to all that noise.

For something more practical, suppose you have a motion detector in the bedroom and you want the lights to come on after dark. You really don't want the lights coming on during the night every time you turn over in bed, and you also don't want them to come on when it's light out. Similarly, you have a motion detector in the hall just outside the bedroom and you want the hall light to come on when it's dark. However, if you happen to take a trip to the bathroom during the night, you'd like the hall light to come on and dim down to a tolerable level.

Listing 3 shows how such a setup might be done. It also points out a number of programming "gotchas" that will likely bite novice HCS II programmers. The first global if is only executed between the time it gets dark out and when you go to bed. The statements are very similar to those in the other examples, with lights going on in response to motion, and off in response to a timeout.

In the second global if, take a look at the condition. The HCS II time of day is based on a midnight-to-midnight cycle, so the condition responds as if it were written

```
IF Time>=0:00:DY AND
     Time<Morning THEN
```

When counting from 0 to 23, it's not necessary to test if the value is greater than or equal to 0; it always will be. The same applies here. Obviously, the hall lights won't come on between bedtime and midnight, but how often do you get up within the first hour after you've gone to bed? If it's a problem for you, just add a bit more intelligence to the program to fill the gap.

Next, notice the extra condition in the first IF statement and the extra IF statement following it. Remember I said that if an X-10 module is already on, I don't send another on command. That is why the previous statements can be written without checks to see if the light is on; it's done automatically. If motion is detected after the light is turned on and before the timer times out, the timer will be cleared but the on command is skipped. When you're dealing with dim (and bright) commands, though, you have to be smarter. You may want to intentionally dim a light that is already on, so I can't block the dim command like I can the on command. You have to add an extra condition to check for whether the light is off, and send the dim command only if it is. That way the command is only executed once and you don't end up with a light dimmed all the way to black. The second IF statement is necessary to retrigger the timer should more motion be detected before the timer times out.

When the conditions in neither GIF are met, which is true any time it's light out, then nothing happens in response to any of the motion detectors, which is just what we want.

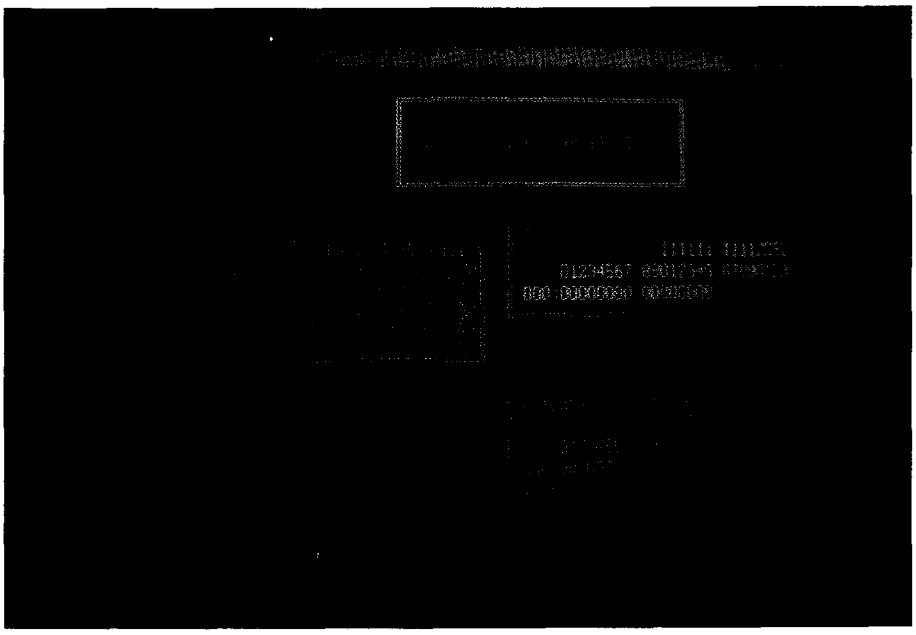


Photo 1-A typical status screen displayed by HOST shows two X-10 housecodes, 16 inputs and 8 outputs on the SC, network activity, and current time and date. The size of the windows and the amount of information displayed change dynamically depending on the equipment you have in your system.

CONTROLLING THE FUTURE

That about does it for version 1.0 of the HCS II. As we get more feedback from those of you living with the system, we'll be refining it to more closely meet your needs. Let us know what you think. ❑

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

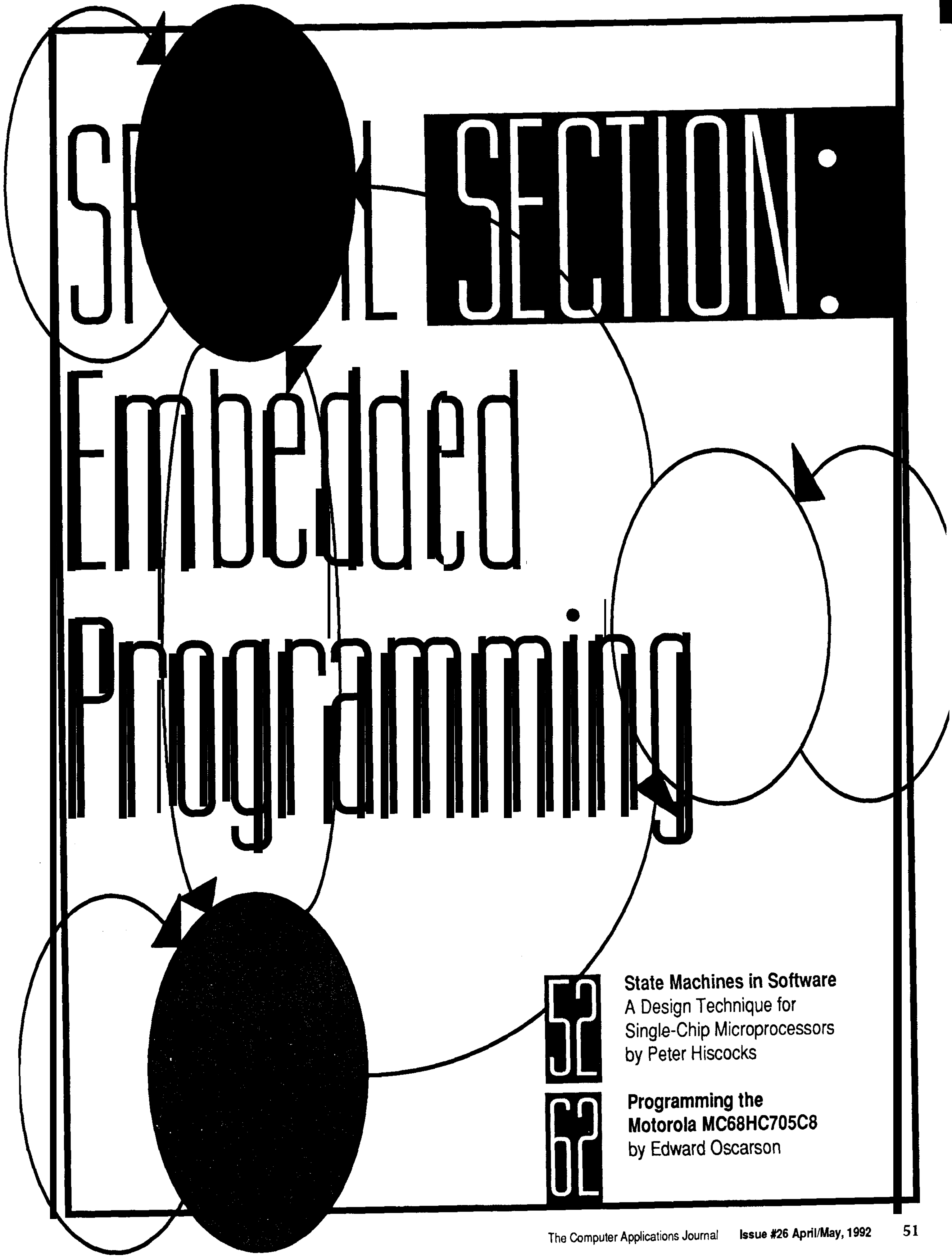
SOURCE

Please see page 31 for more information about the availability of HCS II components.

IRS

410 Very Useful
411 Moderately Useful
412 Not Useful

SPECIAL SECTION: Embedded Programming



52 **State Machines in Software**
A Design Technique for Single-Chip Microprocessors
by Peter Hiscocks

62 **Programming the Motorola MC68HC705C8**
by Edward Oscarson

State Machines in Software

A Design Technique for Single-Chip Microprocessors

State machines are commonly built in hardware using discrete logic. Explore the use of similar techniques for embedded software.

FEATURE ARTICLE

Peter Hiscocks

When you design a digital control system, you may find viewing it as a machine with a number of states useful. The classic example is a traffic light: it has a red state, a green state, and a yellow state. It may also have other states: left turn, advanced green, flashing, pedestrians-only crossing, all stop, and so forth. The state machine is particularly effective at detecting, preventing, and signaling errors in input from a human operator. For that reason, you should always seriously consider a state machine as the basis for a user interface design.

State machines are also very effective at decoding messages. This attribute makes them useful in communication systems like network protocols, and in parsing the text of a command language or a computer program.

THE BUBBLE DIAGRAM

If each one of the machine states is represented by a bubble (Figure 1), then the transition from one state to another is represented by an arrow. Associated with the arrow is *input condition a, b, c,* or *d,* which allows the transition to occur. For example, *a* might represent the transition condition *stop timer time-out* and cause a transition from the red state to the green state.

Generally, most assume that the default condition is for the state machine to stay in a particular state. However, showing a transition, with a condition, looping back into the current state is sometimes useful. This representation indicates that the condition causes the machine to stay in that particular state.

The outputs from the state machine are not shown on Figure 1, but of course they must occur for the machine to be useful. Outputs are produced from a state machine during either a particular state or the transition between states. For example, consider the transition from red to green. I may indicate that the red state actuates the red light: an output conditional on a particular state. Alternatively, I could indicate next to

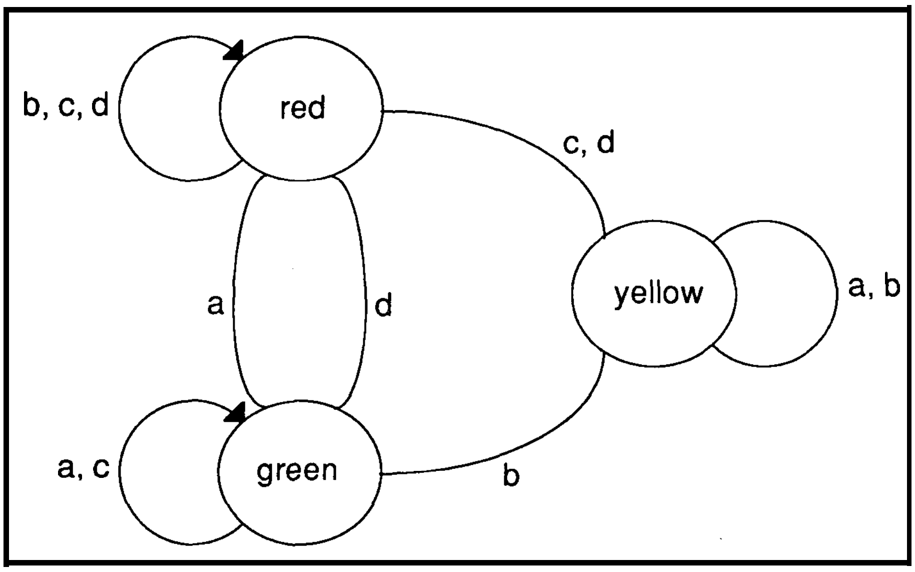


Figure 1—*Bubble diagrams are traditionally used for small state diagrams, but rapidly become unwieldy for /age ones. Each bubble represents a discrete state, while the lines and arrows show transitions between states based on input conditions.*

Current State	Input Condition a	b	c	d
red	green	red	red	red
green	green	yellow	green	red
yellow	yellow	yellow	red	red

Figure 2—*In p/ace* **of the** *bubble diagram, a state &b/e* **is** *sometimes used. The mws represent the current* **state** *of the machine while the* **columns represent** *the input* **conditions.** *For a* **large** *number* **of states** *and* **input conditions,** *he state* **table also** *becomes* **unwieldy.**

the green-red transition ***turn on red light, turn off green light.***

THE STATE TABLE

Bubble diagrams like Figure 1 are useful for small state diagrams, but rapidly get unwieldy for large ones. The number of possible transitions grows as an exponential function of the number of states, so a state diagram quickly outgrows small sheets of paper.

A better approach is to use a *state table,* an array that usually has the rows representing the ***current stnte*** of the machine and the columns representing the ***input conditions. Thus,*** each entry in the state table represents one possible combination of a current state and an input condition. This organization is one of the advantages of the state table: it forces the consideration of all possible states and all possible inputs. The state machine represented by the bubble diagram in Figure 1 is shown as a state table in Figure 2.

Again, with a large number of states and inputs the state table eventually becomes unwieldy, and you must eventually resort to a different approach. One possibility is to break the single large table into several smaller tables.

The state table technique is an example of a ***table driven*** program: its behavior is defined by a data structure rather than by a block of procedural code. Once the procedure that ***interprets*** the table is fully debugged, the system will operate according to the contents of the state table. This aspect makes it very easy to debug and to modify.

Each entry of the table has two parts:

- the *next state* of the machine
- the ***output action*** to occur with this transition

If there is no transition for a given state, then the next state is simply the current state. If a particular input does not occur during a particular state, then you can choose to have the state machine ignore the input or generate an error message. The error message can be quite informative because a particular input has occurred in a particular state (e.g., "Can't push that button while the light is red"].

The interpretation of the state machine, ignoring outputs for a moment, can be reduced to a five-line program similar to what I've shown in Listing la. This code says the next state of the machine is determined by the current-state and the **input_ condi ti on. The** program reads the s t a t e_t a b 1 e matrix to determine the next state, and then sets current_ **s t a t e** equal to that value.

Now, how do I handle outputs? In BASIC and assembly language, I use two arrays, each two-dimensional, each addressed by current state and input condition: a ***next stote array*** and an ***output action array.*** The output action array contains numbers that are codes for various output actions. The processor looks up the action code corresponding to the current state and input condition, and then uses it to determine what action routine it should call.

The pseudocode I provided in Listing la now becomes that in Listing lb. Of course, in assembly language you don't usually have instructions to access entries of an array, and a great deal of singing and dancing is required to simulate the effect of a two-dimensional storage array with sequential lists.

If I use C to design the state table data structure, each entry has a two-part structure consisting of a next state and an output action. The program enters the table with current state and input condition, and reads the next

Listing 1-a) *Interpretation of the state machine, ignoring outputs, can be reduced to a five-line program. b) Adding output* **support** *increases the size of he* **pseudocode** *by* **two** *lines.*

```
a)
  do
    read input-condition
    next_state:=state_table{current_state,input_condition}
    current_state:=next_state
  loop


b)
  do
    read input-condition

    output_action_code=action_table{current_state,input_condition}
    procedure(output_action_code)

    next_state:=state_table{current_state,input_condition}
    current_state:=next_state

  loop
```

state and output action out of the structure entry in the array.

A STATE MACHINE IN PSEUDOCODE

The code in Listing 2 is the stoplight state machine, coded in a sort of high-level pseudolanguage. This code seems like an awful lot of program to define the behavior of the stoplight, but most of it is in definitions. This listing will be eliminated in the machine code for a compiled language or assembly language. In return, you have a very error-resistant program, which may be easily modified with the addition of states. Changing the behavior of the state machine is a matter of changing the entries in the two tables.

A STATE MACHINE IN ASSEMBLY LANGUAGE

Now I will discuss how a state machine approach can be used in a 6805 microprocessor. Implementing a state machine in assembly language on this type of processor presents some

Listing 2—*When the stoplight state machine is coded in a high-level pseudolanguage, the bulk of the program consists of definitions.*

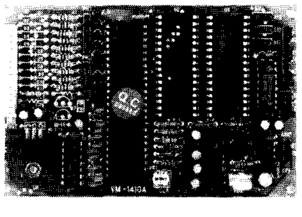
```
/* define the size of the next-state array */
    number_of_states:=3
    number_of_inputs:=4
    dimension next_state(number_of_states,number_of_inputs)
    dimension output_action(number_of_states,number_of_inputs)

/* define the array indices */
/* Next States: */
    red:=0
    green:=1
    yellow:=2

/* Input Conditions */
    a:=0
    b:=1
    c:=2
    d:=3

/* Define the 'next state' array */
    next_state(red,a):=green
    next_state(red,b):=red
    next_state(red,c):=red
    next_state(red,d):=red

    next_state(green,a):=green
    next_state(green,b):=yellow
    next_state(green,c):=green
    next_state(green,d):=red
```

(continued)

digital voice R us!

VOICE MODULES

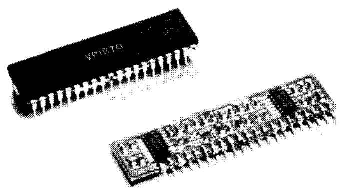
VM-1410A $75 (w/o EPROM)



* Plays natural-sounding voice/sound
* Holds up to 10 messages
* Max. length: 512 seconds @ 32Kbps
* Built-in 1W power amplifier
* Single 6 - 12 VDC operation
* Voice digitization: VP-880 ($495)
* Demostration EPROM available
* Other models available
* For security alarms, P.O.S. promotions, **amusement** parks, elevators.. .

VOICE IC CHIPS

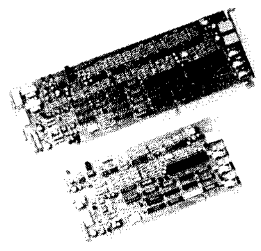
VCS-1870 voice chip set **$29.50**



* Includes a CVSD codec chip (VP-1870) and a filter module (SF-700)
* Sampling rate is adjustable from 9.6 to 128 Kbps
* Universal 8-bit uP/uC interface
* Advanced CMOS technology
* Single 3 - 6 VDC operation
* Other models available
* For digital recorders, digital answering machines, PC voice boards.. .

PC VOICE BOARDS

V-Link/1 voice board $145



* Single-line, IBM PC compatible
* Connects to telephone line or PBX
* Records and plays messages
* Automatic answering and dialing
* Touch tone/dial tone detection
* Requires additional programming
* Software developer's kit ($150) for Microsoft C, Turbo C and Clipper
* Multi-line models available
* For voice mail, voice response.. .

Eletech Electronics, 1262 East Katella Ave, Anaheim, CA 92805, (714) 385-1707 Fax: (714) 385-1708

#129

Listing **2—*continued***

```
     next_state(yellow,a):=yellow
     next_state(yellow,b):=yellow
     next_state(yellow,c):=red
     next_state(yellow,d):=red

/* Define the 'output-action' codes */
     no_output:=0
     print_illegal:=1
     print_emerg:=2

/* Define the 'output-action' array */
     output_action(red,a):=no_action
     output_action(red,b):=print_illegal
     output_action red,c):=print_illegal
     output_action(red,d):=print_emerg

     output_action(green,a):=print_illegal
     output_action green,b):=no_action
     output-action green,c):=print_illegal
     output_action green,d):=print_emerg

     output_action(yellow,a):=print_illegal
     output_action(yellow,b):=print_illegal
     output_action(yellow,c):=no_action
     output_action(yellow,d):=print_emerg

/* Now. finally, the program:      */
current_state:=red                        /* start in state 0 */
```

(continued)

challenges. The Motorola 68000 has a number of instructions that support this kind of program, but single-chip micros like the 6805 do not, and you are forced to use some "ingenious hacks."

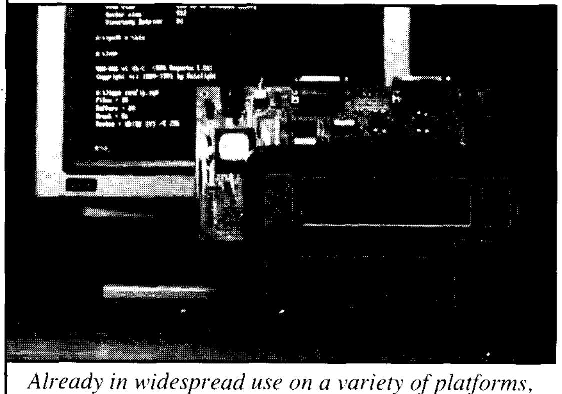
IMPLEMENTING THE NEXT STATE

To recap, I wish to store next state addresses in a two-dimensional array and access the next state address by two indices.

Let me discuss how the next state section of the program would appear in the 6805 [refer to Listing 3a). To begin, I will consider how the machinery works to determine the next state. [In fact, I actually do the output action before changing the state, but the latter is easier to understand, so I'll do it first.)

I assume the location input_condition is properly set by external events. To retrieve the next state from the next state table, I need some kind of indexed LDA instruction. In 6805 assembler, it is of the form

DOS Where You Want It -Runs *on Disk, ROM, FLASH*


*Already in widespread use on a variety of platforms, **ROM-DOS** has proven its dependability.*

Develop your programs on a standard PC and run them in ROM with ROM-DOS, the flexible, affordable operating system that was designed for embedded applications. Compatible with MS-DOS 3.3 1, ROM-DOS boots and runs from ROM, disk, or FLASH memory. Features include: small size (low as 36K); runs executable files from ROM; FLASH disk support; remote disk and multimedia support; and a powerful utility suite. Confirmed compatible by VeriTest, an independent evaluator, ROM-DOS is already in widespread use on multiple system types. Developers Kit-which contains all the tools and utilities needed to configure ROM-DOS to your system, and includes license for 20 copies-costs only $495. ROM-DOS royalty fees depend on quantity purchased, and range from $3 to $25 per copy.

Free Demo Disk! Call Today Toll-Free 1-800-221-6630

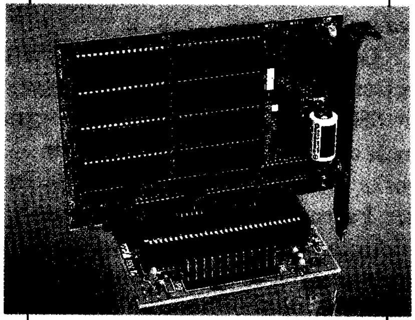
Datalight

17455 68th Ave NE, Suite 304, Bothell, WA 98011, USA • 206-486-8086 • fax 206-486-0253

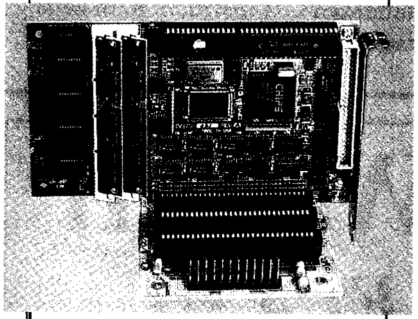
#130

VMAX®
TOTAL PACKAGE PERFORMANCE
QUALITY PRODUCTS
RESPONSIVE SERVICE
RELIABLE DELIVERY

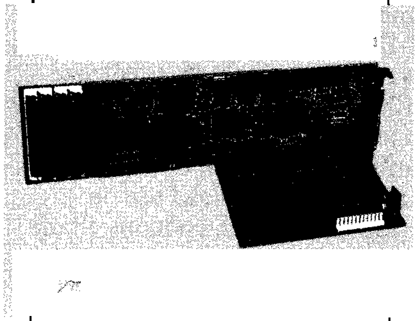
SOLID STATE DISK – $124*
½ Card 2 Disk Emulator
EPROM, Flash and/or SRAM
1 Meg Total, Bootable



TURBO XT WITH FLASH DISK – $266*
2 Serial, 1 Parallel Port
Up to 2 Flash Drives, 1 Meg Total
Software Included, 2 Meg DRAM



20 MHz AT CPU CARD – $536*
4 Meg DRAM, 0 to 60 C
2 Serial, 1 Parallel Port
CMOS, Low Power, Surface Mount



All Tempustech VMAX® products are PC Bus Compatible. Made in U.S.A.,30 Day Money Back Guarantee *Qty. 10, 0k, Call for Quantity Pricing

TEMPUSTECH, INC.
TEL: (800) 634-0701
FAX: (813) 643-4981
Fax for fast response!
295 Airport Road
Naples, FL 33942

Listing 2—continued

```
do                                              /* the main loop      */
  get character                                 /* accept input char  */
  action:=output_action(state,character) /* get action code    */
  call action_subroutine(action)              /* do output action   */
  new_state:=next_state(current_state,character) /* next state  */
  current_state:=new_state                      /* move to new state */
loop                                            /* end of main loop  */

/* These are the action routines   */

procedure action_subroutine(action):
  if action:=0     /* no output     */
  endif

  if action:=1     /* illegal input */
      print "Illegal input'
  endif

  if action:=2     /* D key  input  */
      print "Emergency Red"
  endif
return
```

```
LDA WORD, X
```

where WORD is some 2-byte quantity and X is the content of the X index register. The effective address is the sum of **WORD** and the contents of the X index register. For example,

```
LDX #5
LDA 200. X
```

would have the effect of loading the accumulator with the contents of location 205.

So recognizing that I have four input conditions in my table, the offset into the table of a next state will be given by

```
(current-state  x  4)+
           input-condition
```

For example, if the current state is GREEN (i.e., 1) and the input condition is C (i.e., 2), then the table offset of the desired next state entry is 6. Using assembly language, I would retrieve the next state as shown in Listing 3b.

All well and good. But notice, I avoid having to use a multiply by selecting a number of input conditions that is a power of two. The "multiply" is accomplished by using left shifts.

Usually, resolving such an issue isn't this simple. Note the X register can only accommodate 8 bits, which limits my total table offset to 255 bits maximum.

THE OUTPUT ACTION MACHINERY

First, I define the various actions by numerical codes and construct the output action table shown in Listing 4a. This time, I have to retrieve an action code from the table and act on it. The offsets into the two tables will be identical because the next state and action code tables are the same size and structure. Once the offset into the next state table is calculated, I can also use it to retrieve an action code.

Now, I wish to jump to an action routine based on the value of the action code retrieved from the table. The 6805 ***indexed JSR*** instruction, together with a jump table, will accomplish the task.

I assume that the action code routines exist somewhere in memory. (The preface AR_ indicates an action routine.) See Listing 4b. Why, you may ask, bother with the ***no action*** routine when all it contains is a **NOP?** Because if you ever want to add something to the system, you simply replace the **NOP** in the action routine with the

#131

desired code. The routine makes the system much easier to maintain.

Now, I need to make a jump table out of the addresses of these routines, putting them in the order of their action codes, as shown in Listing 4c.

You access the action routine with a four-step process:

1) Determine the index into the action code table the same way as I determined the index into the next state table.

2) Retrieve the action code from the table using this index.

3) Multiply the action code by 3 to get the offset into the jump table because each entry of the action code jump table is a 3-byte **JMP** instruction.

4) Set up the offset as an index and do an indexed **JSR** through the table, which will send the processor to the appropriate action routine. The action routine terminates with an **RTS** instruction, sending the processor back to the main interpreter loop. The code for this setup is shown in Listing 4d.

The **JSR JMP_TABLE,X** instruction sends the program to the correct entry in the jump table, the jump table entry sends the program to the action routine, then the action routine does what's required and does an **RTS** back to the main routine.

THE PUSH AND RTS HACK

Some microprocessors do not have an indexed **JSR** instruction. The 6502 is one of these, but it does have a **PUSH** instruction. In that case, I've included a useful trick below:

1) Set up a table containing the addresses of the action routines

Listing 3-a) *The 6805 implementation of the next state section consists completely of definitions.* b) *Retrieving he next state involves calculating an offset into the table.*

```
a)
    ;
    INPUT CONDITION DS 1          ,Input variable
    CURRENT-STATE    DS 1         ; Current. state variable (in RAM)

    :Next state definitions
    RED           EQU 0
    GREEN         EQU 1
    YELLOW        EQU 2

    : Next state table:
    NEXT-STATE    EQU *            ; Name the start of the table

    STATE-RED    DB      GREEN     ; Input A
                 DB      RED       ; Input. B
                 DB      RED       ; Input C
                 DB      RED       ; Input. D

    STATE-GREEN  DB      GREEN     ; Input A
                 DB      YELLOW    ; Input B
                 DB      GREEN     ; Input C
                 DB      RED       ; Input D

    STATE_YEL    DB      YELLOW    ; Input A
                 DB      YELLOW    ; Input B
                 DB      RED       ; Input C
                 DB      RED       ; Input. D


b)
        LDA  CURRENT-STATE        Get the current state
        ASL                       Multiply by number of input
        ASL                        conditions. in this case 4
        ADC  INPUT_CONDITION      Add in the input condition
        TAX                       Move table offset. to X register
        LDA  NEXT_STATE,X         Next state is now in A
        STA  CURRENT-STATE        Jump to the next state
```

REALIZE THE POWER OF...

[illegible]

Call for Information!
Toll-Free 1-800-537-5043

PARADIGM

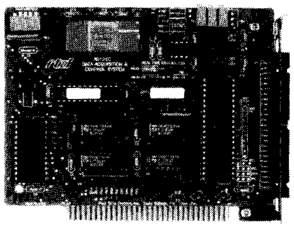
The Model for Programming Productivity

3301 Country Club Road
Suite 2214
Endwell, NY 13760
(607) 748-5966
FAX: (607) 748-5968

All trademarks are property of their respective holders.

Real Time Devices
"Accessing the Analog World"

Exciting New Products!



AD1200 - $359

- **12-bit 125 kHz A/D Conversion**
- **16 Analog input Channels**
- **3 Timers and 16 DIO Lines**
- **Supports DMA & Pacer Clock**
- **AT Bus Version Available!**

for AMPRO CPUs

DM200 12-bit 40 kHz 8 channel analog input board; 8254 timer and 16 digital I/O lines $295

DM406 12-bit 100 kHz 16 channel analog I/O board; 2 D/A outputs; 8254 timer; 16 digital I/O lines; Supports DMA and pacer clock. $449

DM806 High current digital I/O board; 8254 timer; opto-22 compatible $195

PC/XT/AT Boards

AD2700 16 channel 12-bit 150 kHz analog input board; Supports DMA, pacer clock and programmable gain; 8254 timer; 16 digital I/O lines; 16-bit AT bus operation. $525

TC48 Dual Am951 3 Timer/Counters; 24 digital I/O lines; 16-bit AT bus operation with expanded interrupts. $398

DIO24 High current digital I/O board; 24 buffered DIO lines; 8254 timer; Compatible with opto-22 equipment $195

AD3110 Super fast 16 channel A/D board! 12-bit 200 kHz A/D rate; Burst mode operation; On board FIFO memory; Programmable gain; Timers; DIO lines; Supports DMA and pacer clock; Optional D/A outputs $665

AD3710 Low cost version of AD3110 with on board FIFO memory and 200 kHz burst mode operation $525

DA810 8 channel 12-bit D/A; Voltage or current loop output; 8254 timer; 24 digital I/O lines $589

MR16 110 VAC mechanical relay expansion board; Computer control of 16 AC loads........................ **$215**

0P16 Optoisolated 16 channel digital input expansion board. $225

TS16 16 channel thermocouple expansion board; Supports J & K types. $298

Unparalleled choices for real world **control.** *Over* **50** *hardware and* **soft-***ware* **products** *for single and* **OEM** **users.**

FREE 80 PAGE CATALOG!

Real Time Devices, Inc.
State College, PA USA

Tel.:814/234-8087
FAX:814/234-5218

#132

Listing 4-a) *The action codes are defined first, followed by the output action table. b) Next, the wde to carry out each action is defined. c) In order to get to each action routine, a jump table is defined. d) Last comes the code to retrieve he address of a given action routine and jump to it.*

```
a)
   NO-ACTION     EOU 0             : no action
   PRT_ILLEGAL EQU 1               ; print 'illegal '
   PRT_EMERG     EOU 2             ; print 'emergency'

   ; Output action table:
   OUTPUT_ACTION EOU *               ; Name start of table

   ACTION-RED DB       NO_ACTION; Input A
              DB       ILLEGAL   ; Input B
              DB       ILLEGAL   ; Input C
              DB       EMERG     ; Input D

   ACTION-GREEN DB     ILLEGAL   ; Input A
                DB     NO-ACTION; Input B
                DB     ILLEGAL   ; Input C
                DB     EMERG     ; Input D
   ;
   ACTION_YEL  DB      ILLEGAL ; Input A
               DB      ILLEGAL   ; Input B
               DB      NO_ACTION; Input C
               DB      EMERG     ; Input D
b)
   AR_NO_ACTION NOP          ; 'no action' action routine
                RTS

   AR-ILLEGAL   .....        ; print 'illegal
                .....
                RTS

   AR_EMERG     .....        ; print 'emergency'
                .....
                RTS
c)
   JMP_TABLE EOU *                ; action routines jump table
             JMP  AR_NO_ACTION    ; action code 0
             JMP AR-ILLEGAL       ; action code 1
             JMP AR_EMERG         ; action code 2
d)

   ; We assume the offset of the action code is in the X register

        LDA OUTPUT_ACTION,X    ; retrieve action code from table
        STA TEMP               ; multiply it by 3
        ASL                        (first multiply by 2
        ADC TEMP                    and then add it in again)
                               ; to get the jump table offset
        TAX                    ; set up for indexed jsr
        JSR JMP_TABLE.X        : jump to the vector in the table
        .....                  ; action routine RTS returns here
```

somewhere in ROM. Each table entry, assuming a 16-bit address bus, will consist of a low and high address byte.

2) Index into this table and retrieve the 2-byte address of the action routine using the action code.

3) Push the 2 bytes of the action routine onto the stack in the same order that a subroutine call would do it.

4) Execute an **RTS** instruction.

The effect of an **RTS is to** pop the address off the stack and jump to it. In effect, this hack gives me the capability of a computed jump.

Listing 5-a) *To utilize self-modifying wde, a table of the addresses of the action routines is necessary. b) RAM locations 40, 41, and 42 are used to store the synthetic jump instruction.*

```
a)
   ADDRESS-TABLE EQU *
                 DW  ACTION_ROUTINE_1   ; Address of routine 1
                 DW  ACTION_ROUTINE_2   ; Address of routine 2
                   ....etc......
                 DW  ACTION_ROUTINE_n   ; Address of routine n

b)

   ; Assume X contains the action opcode

      LDA  #CC                 ; Store the op code for JMP
      STA  $40                 ; at location 40 in RAM

      LDX  ACTION_CODE         ; Get the action opcode
      ASLX                     ; Double it
      LDA  ADDRESS_TABLE.X     ; Get high byte of action routine
      STA  $41                 ; and store it in RAM
      INCX
      LDA  ADDRESS_TABLE.X     ; Get low byte of action routine
      STA  642                 ; and store it in RAM

      JSR  $0040               ; Do a JSR through constructed JMP
                               : instruction. to action routine
```

SELF-MODIFYING CODE

The final method, an alternative for the 6805, is similar to the **PUSH** and **RTS** hack. Rather than do an indexed J SR though a jump table, I have the program construct a J M P instruction with an address that points at the desired action routine. The address associated with the J M P will change, so the instruction must be constructed in the RAM area of the processor.

The addresses of the action routines are set up in a table much like the jump table. The processor uses the action code to index into the action routine address table, retrieves the action routine address (2 bytes), and then stores the action code address [prefixed by the opcode for J MP) in three locations in RAM. The processor then does a J SR to the first of these three RAM locations.

The 6805 code that uses the self-modifying code method for vectoring to an action routine is shown below. Here, I assume the addresses of the action routines are stored in a table as in Listing 5a (DW is the assembler operative for *define word).*

In the code fragment in Listing 5b, locations 40, 41, and 42 are used to store the *synthetic jump* instruction.

COMPARING VECTORING METHODS

I have examined three different methods of vectoring to action routines based on an ***action opcode.*** Which is best for your particular application?

The advantages of the jump table method are that it is entirely contained in ROM [does not require RAM registers) and can be done without **PUSH** and **POP** instructions. The disadvantage is it requires a jump table containing 3 bytes for every action routine.

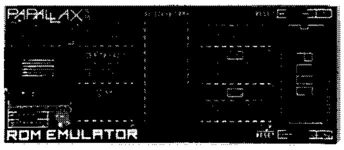
The **PUSH** and **RTS** method has the advantage of using a shorter action code routines table and 2 bytes per action routine. However, the action code table does require that the assembler be able to extract the low and high bytes of an address for each entry.

The self-modifying code method has the advantageous **PUSH** and **RTS** method, but it does not require that the microprocessor have a PUSH instruction. However, it does require 3 precious bytes of RAM and induces a slightly queasy feeling in those who adhere to the tenets of structured programming.

EPROM EMULATORS

An EPROM emulator appears as an EPROM to a target system. Instead of programming EPROMs, you simply download your code to the emulator. In seconds, you see results.

27256 EPROM EMULATOR



Emulates 2764, 27128, & 27256 EPROMs.

Plugs into target EPROM socket and connects to PC parallel port via telephone cable.

Loads Intel. Motorola, hex, and binary files.

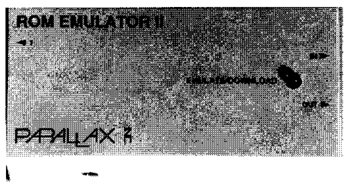
Reset outputs restart target after downloading.

Downloads 32K in 2 sec (12 MHz PC AT)

$199

27010 EPROM EMULATOR

Up to 4 units can be daisy-chained to emulate consecutive EPROMs and to support 16 and 32-bit systems.



Emulates 2764, 27128, 27256, 27512, and 27010 EPROMs.

Plugs into target EPROM socket and connects to PC parallel port via telephone cable.

Reset outputs restart target sytem.

$349

VISA PARALLAX MasterCard

(916) 721-8217
Fax (916) 726-1905

Parallax, Inc. • 6200 Desimone Lane, #69A
Citrus Heights, CA 95621

Dealer/OEM Pricing Available

Prices subject to change without notice.
California residents add appropriate sales tax.
Add $4.00 for UPS ground. $7.00 for UPS 2nd day.
$15.00 for UPS next day.

A CASE HISTORY

Comparing the state machine technique with a more conventional approach to building a microprocessor program is interesting. The case history I describe here is a special-purpose terminal, which uses the 68705P3 microprocessor, containing a 16 x 1 LCD character display and a 4 x 3 keypad. Pressing keys causes the processor to select or edit displays, or to send messages out a serial RS-232 port.

I originally wrote the code for this project using the brute force get-an-input-compare-and-branch approach. However, the result contained a great number of conditional and unconditional branch statements. Alarmed at what looked like a massive debugging effort, I then rewrote the code using the state machine approach.

Both programs required about the same amount of RAM: 24 versus 25 bytes. The compare-and-branch method required 1540 bytes of EPROM for the entire program: tests, branches, and subroutines. The state machine, which contained eight states and nine possible inputs, was 121 bytes smaller. This decrement is not a great saving, but it does indicate that a problem of this magnitude will be smaller if the state machine approach is used. The benefit from the state machine method would probably increase with larger programs.

The big advantage of the state machine approach shows in the number of decision and jump statements. Unlike the load and store instructions, branches tend to be a source of error. For example, conditional branch instructions are usually associated with loops, making it easy to introduce errors such as executing a loop too many or too few times, or inadvertently branching past code. Thus, the number of branch statements can be taken as a very rough indicator of ***code complexity.***

In this example, the control logic of the compare-and-branch method contained 140 branch statements, but the control logic of the state machine method contained 18 branch statements: three in the state machine logic and sixteen in the action routines (there are an additional 21 **JMP** instructions in the state machine jump table).

Clearly, the control logic of the state machine is much less complex and therefore simpler to debug. Furthermore, and this point is very important, ***the state machine approach assumes no defaults.*** Every input condition for every possible state has an entry in the state and action tables, and has therefore been accounted for. On the other hand, the compare-and-branch logic assumes the default occurs if none of the comparison branches occur. Overlooking an erroneous state and input combination would be very easy.

Finally, for applications where speed is critical, the state machine approach is attractive because it minimizes the number of tests and branches required to make a decision and to produce output. ❑

Peter Hiscocks teaches electronics at Ryerson Polytechnical Institute in Toronto and free-lances in electronic design construction.

REFERENCES

Jack V. Landau, "State Description Techniques Applied to Industrial Machine Control," *IEEE **Computer*** (February 1979): 32-40.

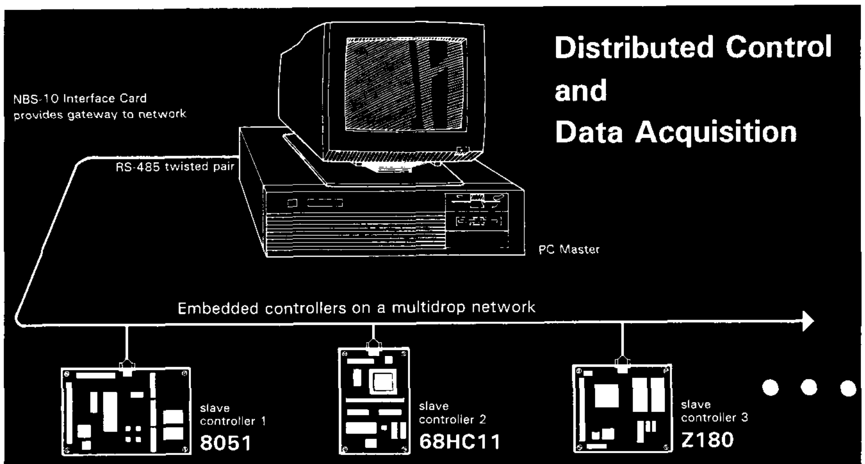
G.A.Van den Bout, "Designing A Command Language," ***BYTE*** Magazine (June 1979): 176-l 87.

David E. Cortesi, "Using Finite State Machines," ***BYTE*** Magazine (October 1979): 70-72.

William E. Hamilton, "State Machine Models Simplify Software Development," *EDN (4* August 1982): 129-134.

413 Very Useful
414 Moderately Useful
415 Not Useful

Distributed Control and Data Acquisition



Connectivity is implemented at low cost with a network designed by CIMETRICS TECHNOLOGY.

Our hardware/software solution includes:
Microcontroller code, PC/XT/AT interface card, NSP protocol, and application development software toolkit.

CIMETRICS TECHNOLOGY
the network solution for embedded controllers

120 West State Street, Ithaca, New York, 14850
Phone: **607-273-5715** FAX: **607-273-5712**

#133

SPECIAL SECTION

Edward **Oscarson**

Programming the Motorola MC68HC705C8

Most low-cost, stand-alone EPROM programmers can't handle microcontrollers with on-chip EPROMs. What's a developer to do? Use the chip to program itself; Edward shows us how.

The latest entry in Motorola's stable of "05" series 8-bit microcontrollers is the MC68HC705C8. The part is becoming more and more popular because of its high level of integration, low power consumption, and reasonabe price ($16 for the windowed version). What has been lacking is an inexpensive programmer that supports the part's on-chip EPROM. The only inexpensive programmer I've been able to find so far is a board Motorola offered as an incentive to try out the MC68HC705 controller.

Motorola's board is a stand-alone programmer with an RS-232 port and a socket for the chip to be programmed. You must provide a power supply for +5 V, ± 12 V, and Vpp voltages, and have to manually apply the power and Vpp and twiddle the reset switch at just the right times. The board is further complicated by the number of programming options supported, such as copying the contents of an external EPROM into the chip or downloading and executing programs from the on-chip RAM. Programming the PLCC version of the processor is supported if you supply a PLCC socket.

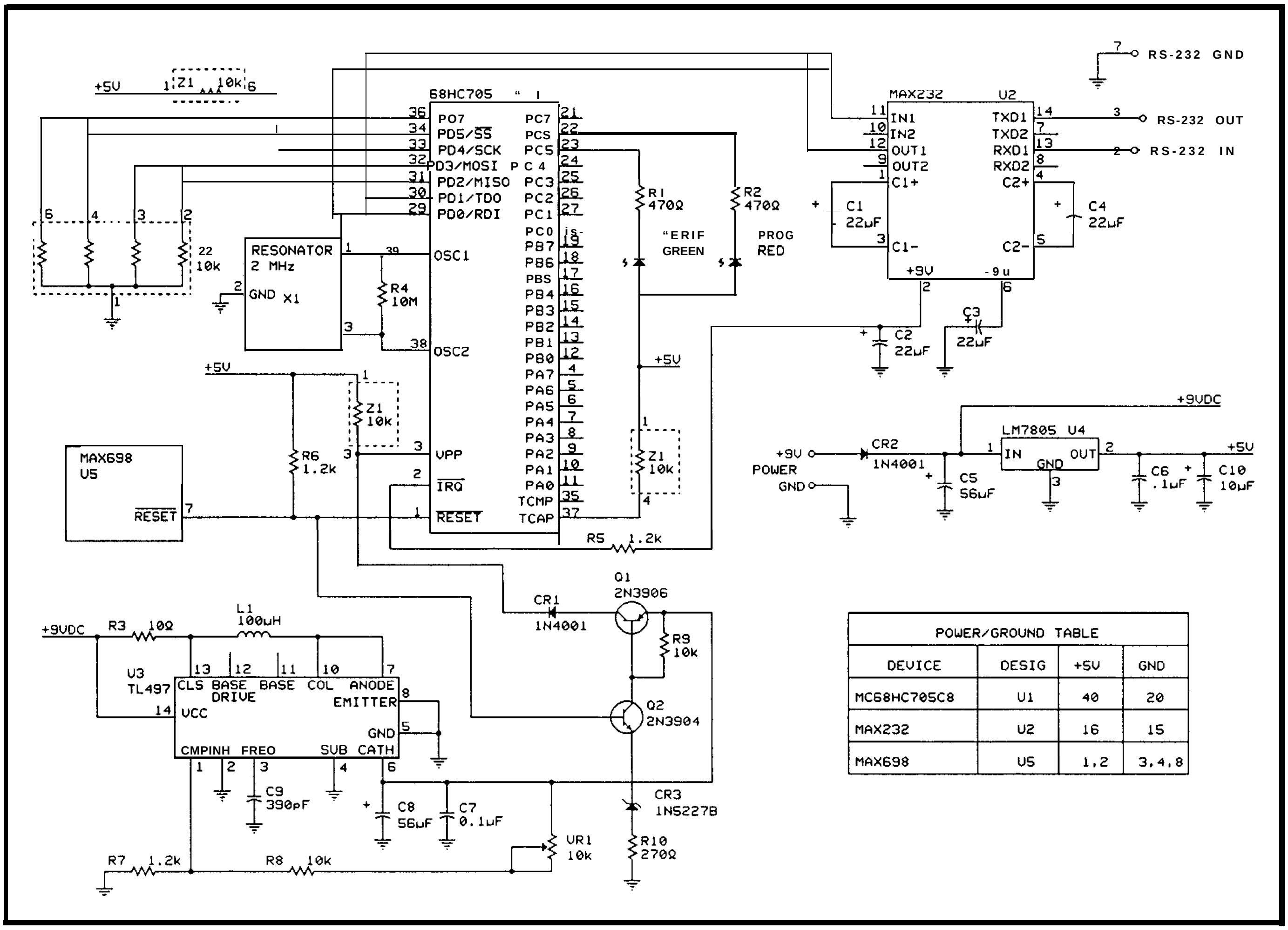
I decided to fill the gap by designing and building the PGMHC05. The PGMHC05 is a low-cost programmer designed for the microcontroller experimenter. To keep the cost down, it only supports device programming of the DIP version of the part. Most experimenters will want to use the DIP version anyway because it is available in both erasable and one-time programmable (plastic) versions. The PLCC version is not erasable and therefore not useful for development (although it does take much less space when surface mounted on a board). The PGMHCOS is completely self-contained and powered by a 9-VDC wall transformer. An RS-232 connection to the host PC is the only other requirement. Vpp is generated on the board and is automatically switched on by the programmer after the +5-volt supply is stable. The RS-232 interface is provided by a MAX232 transceiver, which generates the necessary voltages for communication with the PC.

Motorola has made two software packages available on their "freeware" bulletin board system that were designed for their programmer. One is used to send files to the programmer and other for programming the parts. Rather than reinvent the wheel, I simply made the PGMHCOS compatible with both programs. You'll also find the programs on the Circuit Cellar BBS. I'll discuss the software more in a bit, but first I'll cover the hardware.

THE HARDWARE

Many of Motorola's microcontrollers feature a data loader and programming logic built into ROM on the chip. Therefore, the "brain" of the PGMHCOS is the MC68HC705 itself. The rest of the board is broken up into five major sections: the MC68HC705 support circuitry, reset circuit, RS-232 interface, Vpp supply, and +5-volt supply. Figure 1 shows a complete schematic for the programmer board.

The MC68HC705 support circuitry consists of the clock oscillator and miscellaneous configuration resistors. The microcontroller has an internal oscillator that is designed to operate correctly with either a ceramic resonator or a crystal. Again to keep costs down, I use a 2-MHz ceramic resonator that is about half the cost of a crystal and two capacitors. The



POWER/GROUND TABLE			
DEVICE	DESIG	+5V	GND
MC68HC705C8	U1	40	20
MAX232	U2	16	15
MAX698	U5	1,2	3,4,8

Figure 1--The "brain" of the PGMHC05 is the MC68HC705 being programmed. A TL497 switching regulator is used to generate Vpp, while a MAX232 is used to generate proper RS-232 levels.

accuracy of the resonator is more than adequate for generating the programming timing pulse and the clock for the 9600-bps serial interface.

The configuration resistors are used to select one of the MC68HC-705's many programming modes (see Table 1). I permanently configure the programmer for to mode 2 using the resistor networks connected to pins 3134. Mode 2 allows the PC to load a program into RAM and execute it. This function enables the PC to set up the microcontroller to receive data and copy it to the internal EPROM.

The MC68HC705's reset line is controlled by a Maxim MAX698 POR (power-on reset) chip. This chip senses the level of the regulated +5-volt supply and provides a controlled reset (active low] output. The reset output is delayed by an internal 150-ms timer during power up and is immediately switched to reset during power down. The timer provides enough time for the programmer's Vpp voltage to stabilize before the MC68HC705 is brought out of the reset state.

The RS-232 interface used for downloading data uses an inexpensive MAX232 transceiver chip and four capacitors. The MAX232 eliminates two power supplies and regulators from the design at the expense of the four small capacitors required for the internal switching supply. The chip also provides ±9-volt outputs for external circuitry. The +9-volt output is connected to the microcontroller's ● IRQ pin to enable the programming circuitry inside the chip.

To generate Vpp, I use a TL497 switching regulator. While this part may not be the latest and greatest, it is inexpensive, easy to find, and requires only an inductor and some resistors and capacitors. The TL497 is capable of supplying up to 500 mA and can easily supply the less than 10-mA programming current required by the MC68HC705C8.

Designing with a switching regulator isn't always obvious, so I thought I'd go over how I came up with the component values I used. The output voltage is set by the resistor divider formed by R7, R8, and VR1. The output voltage is determined by

$$V_{out} = \frac{R7 + R8 + VR1}{1000}$$

VR1 provides a limited adjustment of the voltage, allowing Vpp to be set to +14.75 volts for the MC68HC705C8 or + 19 volts for the MC68HC805C4. If only one type of part is to be programmed, the value of R8 and VR1 may be changed to 13k and 2k, respectively, for the HC705C8 or to 16k and 5k for the HC805C4. The current limiting in the supply is provided by R6 and serves to limit the inductor current as well as the output current. Determine the peak inductor current using

$$I_{PK} = 2 I_{Omax} \frac{V_O}{V_I}$$
$$= 2 \times 10\ mA \times \frac{15\ V}{9\ V}$$
$$= 33\ mA$$

In the equation above, I_O is the maximum output current and is set to 10 mA, I_{PK} is the peak inductor current, and V_O and V_I are the output and input voltage, respectively. The value of I_{PK} will be used to determine the current-limiting resistor. Substituting 33 mA for I_{PK} in the equation that determines the value of R6 yields

$$R6 = \frac{0.5\ V}{I_{PK}} = \frac{0.5\ v}{0.033\ A} = 15\Omega$$

The calculated value of R6 is 15 Ω. I then decreased the value of R6 to 10 Ω for a little more margin.

The inductor used in the supply is a 100-µH part from Toko and is available from Digi-Key. Due to the light output loading, almost any value inductor in the range of SO-500 µH will work. The current handling capability of the inductor should be at least 100 mA to prevent saturating the inductor during current peaks.

A 390-pF timing capacitor was chosen to provide an on time of about 32 µs. This standard value is within the operating range of the part. At the low current the programmer requires, almost any value of capacitor in the range of 200-2000 pF should work, but the value of C8 will change based on the resulting on time.

I chose a value for C8 to provide an output ripple of less than 50 millivolts at a typical load of 10 mA. To determine the value of C8, use the following:

$$C_F(\mu F) \approx T_{ON}(\mu s)\frac{\frac{V_I}{V_O} I_{PK} + I_O}{V_{RIPPLE}}$$

Assigning a typical load of 10 mA to I_O and substituting the calculated value of I_{PK} in the equation for the output capacitor yields

Mode	Pin 31	Pin 32	Pin 33	Pin 34	Operating Mode
0	GND	GND	GND	GND	Program and Verify from an external EPROM
1	GND	GND	GND	VCC	Reserved
2	GND	GND	VCC	GND	Load Program to RAM and run program
3	GND	GND	VCC	VCC	Reserved
4	GND	VCC	GND	GND	Verify EPROM against external EPROM
5	GND	VCC	GND	VCC	Secure EPROM & Verify against external EPROM
6	GND	VCC	VCC	GND	Dump Contents of EPROM via RS-232 interface
7	GND	VCC	VCC	VCC	Secure EPROM and Dump contents via RS-232
8	VCC	N/A	N/A	N/A	Execute program in RAM at location $0051

Tabb I--The *MC68HC705 has numerous programming modes, selected ty pulling* pins on *the chip high or low at reset.*

MAGNUM OPUS
CUSTOM ELECTRONIC ENGINEERING
XT™ IBM SINGLE BOARD COMPUTER
** CLOSE OUT SPECIAL **
1/$65.00 10/$63.00 25/$60.00 100/$55.00

- includes 256K RAM -Supports 640K
- BIOS ROM
- 4.77 MHz, 8 MHz Operation
- Parallel Port
- Clock
- 8087 Socket
- Keyboard Port
- Model #XT-640R
- Speaker Port
- Includes 8 Slot Passive Backplane

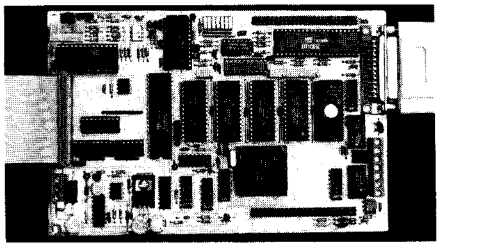
EP-1 EPROM PROGRAMMER
- Plugs into PC, XT, AT
- 2716-27010 EPROMS
- 16K-1Mega Bit EEPROMS
- NMOS, CMOS
- Includes Software
- Read, Write, Verify, Copy
- $195.00 quantity 1

BASIC SBC
- BASIC Interpreter
- 8K-32K RAM, 8K-24K EPROM
- On Board EPROM Programmer
- Serial Port—RS232C
- 24 Parallel I/O Lines
- Timers, Interrupts
- $197.00 quantity 1

Engineering ● Manufacturing
Analog ● Digital ● Microprocessor
Standard ● Custom ● Semi-Custom

284 Kennedy St • Iselin, NJ 08830
(908) 283-4925

INDUSTRIAL STRENGTH



SINGLE BOARD COMPUTER
HAS OPTIMUM FEATURES FOR DATA AQUISITION AND CONTROL. PROFESSIONALLY DESIGNED AND BUILT FOR TOUGH APPLICATIONS WITH A 2 YEAR WARRANTEE AND A 30 DAY MONEY BACK GUARANTEE. PROGRAMMING IS A SNAP IN EITHER MULTI-TASKING BASIC, FORTH OR ANSI C. FEATURES INCLUDE:
- 8 OPTICALLY COUPLED DIGITAL INPUTS
- 8 HIGH DRIVE BUFFERED DIGITAL OUPUTS
- 16 PROGRAMMABLE DIGITAL I/O LINES
- 16, THIRTEEN BIT ANALOG INPUTS
- 2 OPTIONAL TWELVE BIT ANALOG OUTPUTS
- 4 SERIAL RS232/485 SERIAL PORTS
- 6 TIMER/COUNTERS
- UP TO 512K MEMORY PLUS EEPROM
- E-PAC 3000 G2 COST $389.00 QTY. 1

EMAC, inc.
618-529-4525 Fax 618-457-0110
P.O. BOX 2042. CARBONDALE, IL 62902

#137

$$C_F\,(\mu F) \approx 32\ \mu s\ \frac{\frac{9\ V}{15\ V}\,0.033\ A + 0.010\ A}{0.05\ v}$$

$$\approx 17\ \mu F$$

From the equations above, the calculated value for C8 is 17 µF. I used a 56-µF electrolytic capacitor to ensure a low ripple under all conditions. In addition to the electrolytic, I also used a 0.1 -µF ceramic capacitor on the Vpp output to provide additional filtering of any high-frequency switching transients from the supply.

The Vpp switching is controlled via the microcontroller's *RESET line and a two-transistor switch. Q2 is used to turn Q 1 on when the MC68HC705 is brought out of reset, which switches the programming voltage to the microcontroller. A 3.6-volt zener diode (CR3) on the emitter of Q2 ensures Ql and Q2 are off when the 5-volt supply drops below 4 volts. I want to be sure the programming voltage is off when the 5-volt supply is not in regulation and when programmed devices are removed from the programmer. R10 limits the current through Q2 to about 4 mA, which is enough to force Ql into saturation when it is on. Figures 2 and 3 show the timing of the programming voltage with respect to Vcc during power on and power off sequencing.

Power for the board may come from any wall transformer or power supply capable of providing 9 VDC at 150 mA. I put a series diode on the board to protect the circuitry against reverse voltage on the supply line. Do

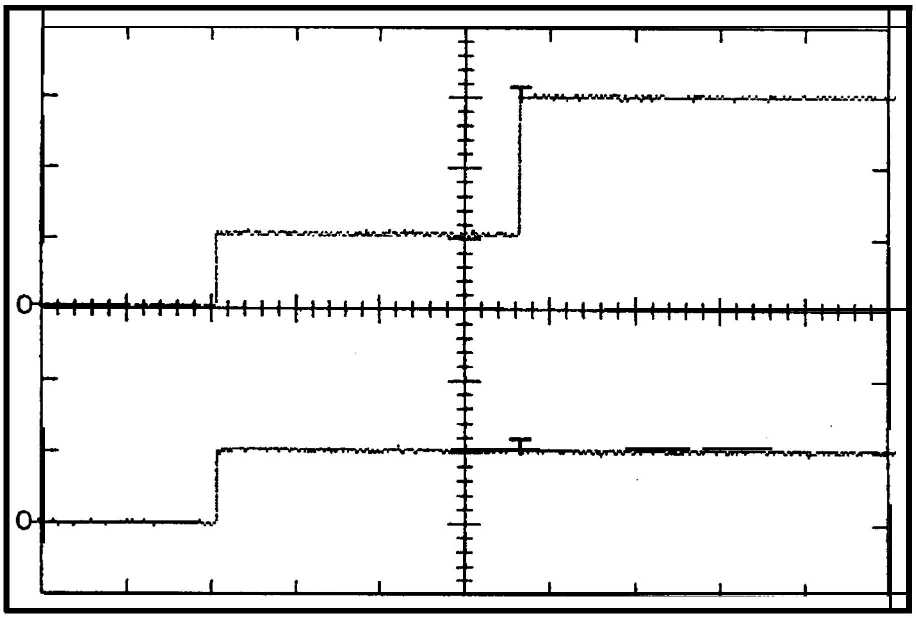
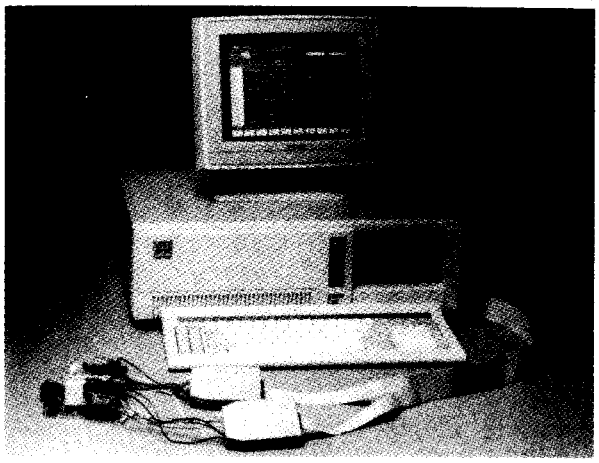


Figure 2—*Programming voltages must be applied in sequence during power up. The fop trace shows Vpp while the bottom shows Vcc. In both cases, the vertical scale is 5 volts per division and the horizontal scale is 50 ms par division.*

PC-Based Logic Analyzers



New Prices for 1992

ID160 (50MHz) $595

ID161 (100 MHz) $695

*High Speed ● 8K Trace Buffer • 16 timing channels expandable to 32 state channels *Multi-Level Triggering *State Pass Counting *Event Timer/Counter ● Performance Histograms *Hardcopy Output *Disassembles S-bit micros *Supports VGA /EGA/HGA *Demo diskette available

30 Day Money Back Guarantee

INNOTEC DESIGN, INC.
6910 Oslo Circle, Suite 207
Buena Park, CA 90621
Tel: 714-522-1469 FAX:714-527-1812

#1

X-10 TW523
DEVELOPERS KIT!

AT LAST !! Now you can TRANSMIT & RECEIVE X-10 COMMANDS under control of YOUR application program. We supply you with the I-O SOURCE CODE. Be up and running in MINUTES!! Write your own CLIPPER/BASIC program. We also have comple application packages. Shipped within 24 hours worldwide. Phone our BBS for complete pricing and details.

$75

Each Kit contains:

- **X-10 TW523 Transceiver**
- **Serial OR Parallel RJ-1 1/DB25 Interface**
- **Microsoft/Turbo C I-O Drivers(source)**
- **6 foot cable**
- **Technical Reference Manual**

DEALER INQUIRIES WELCOME

BARAN-HARPER GROUP
LOS ANGELES*TORONTO

VOICE 416-294-6473 BBS 416-471-677

#139

not use a supply over + 12 VDC because you won't be able to properly adjust the Vpp voltage. The +5-volt supply is regulated with an LM78L05 regulator.

I intended the programmer's circuit design to be as simple as possible. Low-cost and available parts are used throughout. Most parts, including the zero insertion force socket, are available from Digi-Key.

I designed the PGMHC05 mechanical layout to simplify the construction of an enclosure. All active parts are mounted on what would be considered the component side of a two-sided PC board. After they are soldered in place, the programming socket is mounted on the solder side [I usually mount the programming socket in a standard IC socket to get a little extra height above the component leads protruding from the board]. This arrangement allows the board to be mounted in a plastic enclosure that has a square cutout for the programming socket and four mounting holes for the board. A couple of additional holes for the power switch, power jack, and LEDs, and the enclosure is complete.

Connections for two status LEDs are provided on the board. These LEDs provide a visual indication of programmer operation when the **PRO** G7 software package is used. They are provided for compatibility with the Motorola board and may be left out if you'd like because the **BURN 0** 5 program provides programming status information via the PC display.

THE SOFTWARE

B U R NO 5 is a basic, no-frills program that will only do device programming using a PC as the host. It accepts a standard S-record file as its input and bums the program into the microcontroller's on-board EPROM. Typical programming time is about 30 seconds with larger files taking proportionally longer. **BURN 0 5** executes from the DOS command line and requires no switch manipulation once the board is turned on and Vpp is applied to the chip (which is done automatically by the PGMHC05). When the programming is complete,

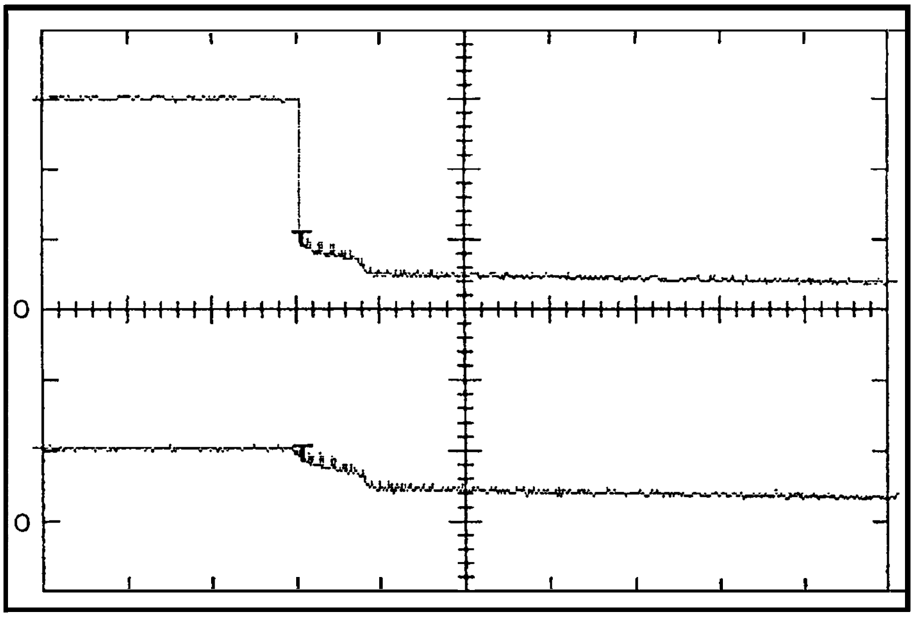
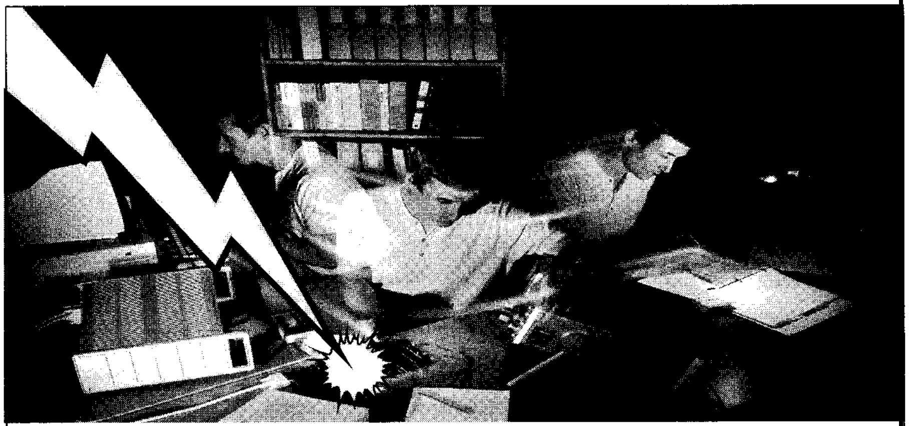


Figure 3—*As with power up, the power supplies must be sequenced during power down. The top trace shorn Vpp and the bottom shows Vcc. The scales are 5 volts per division vertically and 50 ms per division horizontally.*

the verification status is displayed on the PC. A typical **BURN05** command would be **"BURN05 TEST.** S19" where **TEST.** S19 is the S-record file.

PRO G7 is a fancier, menu-driven program used to program, load, and execute programs under control of a host PC. The program is cumbersome to use when programming parts, and programming times are significantly longer than when using **BURN05**, but **PROG7** has some extra features not



MultiTask!™ Execs and GOFAST™ Math Speed Time-to-Market

Zap your application with peak performance using tested code and expert support. Control real-time scheduling with MultiTask! source code executives, or **ROM-able, re-entrant** GOFAST floating point:

*Replace 80x87 MATH COPROCESSORS;
*Drop-in IEEE SOURCE LIBRARIES;
*Link-and-Go with C compilers: Intel®, Microsoft®, Borland®, WATCOM®, Zortech, Metaware®, and more..

Solutions for **80386/486 PROTECTED-MODE**, 80x86 & V-Series, Z80/180/64 180, 8085, 68xxx, **68HC16**, 68HC11, 6801, 6809, 8051, 80196, i960, R3000, SPARC® and more.

Call for free information diskettes today. PHONE 503-641-8446; FAX 503-644-24 13; USA TOLL FREE **800-356-7097**.

14215 NW Science Park Drive
Portland. OR 97229

U S SOFTWARE®

© 1992 US Software Corporation. GOFAST and MultiTask! are trademarks of US Software Corporation. All other trademarks belong to their repective owners.

#140

found in its little brother. Among them, **PRO** G7 can transfer the microcontroller's EPROM contents to the PC, which can be useful for comparing the contents to a data file or for making copies of the part.

PROGRAM SOME PARTS

Setting up the programmer and programming parts is easy. First, connect the RS-232 port to COM1 on the PC. Make sure the board power is off and insert the MC68HC705 in the ZIF socket. Next, turn on the programmer's power (the two LEDs may flash momentarily). Type **"BURN05 f i l ename. ext" on** the PC, where **f i l ename. ext is** the name of the S-record file to be programmed into the part. The PC should display the status on the screen as each operation completes (blank check, program, and verify). When programming is complete, the final status is displayed on the screen. Board power must be off prior to removing the programmed part from the socket.

FINALLY...

The PGMHC05 is a valuable and inexpensive tool for the microcontroller designer. It has proven very useful when developing microcontroller programs for my projects as well as for small production runs of programmed parts. Now many more system architects can add a new dimension to their designs. ❑

Edward Oscarson is a senior design engineer with Hamilton Standard and holds a B.S.E.E. from the University of Connecticut.

Complete kits of parts and assembled PC boards are available from:

Single Chip Solutions
P.O. Box 680
New Hartford, CT 06057
(203) 496-7794

Kit-includes all parts (less optional resistors and LEDs), wall transformer and mating connector, DB-25 connector, and PC board [no enclosure is available) for **$100.**

Asy-Assembled PC board with wall transformer, mating connector, and DB-25 connector [no enclosure is available) for $130.

public domain cross-assembler for the PC and **BURN05** (copy charge) for $5.

Add $5.00 for shipping and handling on Kit or Asy. Connecticut residents add sales tax.

Motorola Freeware **BBS : (512) 891-3733 (BURN05** and **PROG7** software)

Digi-Key Corporation
701 Brooks Ave. South
P.O. Box 677
Thief River Falls, MN 56701-0677
(800) 344-4539
Fax: (218) 681-3380

I R S

416 Very Useful
417 Moderately Useful
418 Not Useful

Cross-Assemblers from $50.00
Simulators from $100.00
Cross-Disassemblers from $100.00
Developer Packages
from $200.00 (a $50.00 Savings)

Make Programming Easy
Our Macro **Cross-assemblers are easy to use. With powerful conditional assembly and unlimited include files.**

Get It Debugged -- FAST
Don't **wait until the hardware is finished. Debug your software with our Simulators.**

Recover Lost Source!
Our line of disassemblers can help you recreate the original assembly language source.

Thousands Of Satisfied Customers Worldwide
PseudoCorp has been providing quality solutions for microprocessor problems since 1995.

Processors

Intel 8048	RCA 1802, 05	Intel 8051	Motorola 6805
Motorola 6800	Motorola 6801	Motorola 68HC11	WDC 65C02
Hitachi 6301	Motorola 6809	MOS Tech 6502	NSC 600
Rockwell 65C02	Intel 8080, 85	Zilog 280	
Hitachi HD64180	Mot. 68k, 8, 10	Intel 8096, 196kc	

New
Zilog Z8
Zilog Super8
. All products require an IBM PC or compatible.

For Information Or To Order Call:
PseudoCorp
716 Thimble Shoals Blvd., Suite E
Newport News, VA 23606
(804) 873-1947
FAX: (804) 873-2154

#14

EPROM PROGRAMMERS

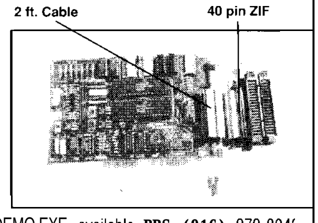
Stand-Alone Gang Programmer **$750.00**

Completely stand-alone or PC-driven
Programs E(E)PROMs
1 Megabit of DRAM
User upgradable to 32 Megabit
.3/.6" ZIF Sockets .RS-23.7.
Parallel In and Out
32K internal Flash FFPROM for easy filmware upgrades
• **Quick Pulse Algorithm 127256 in 5 sec. 1 Megabit in** 17 sec.)
• 2 year warranty
. Made in the US A
• Technical support by phone
- Complete manual and schematic
- Single **Socket Programmer also available. $550.00**
. Split and Shuffle 16 & 32bit
. 100 User Definable Macros 10 User Definable Configurations
• Intelligent Identifier
. Binary, Intel Hex. and Motorola S
• 2716 to 4 Megabit

Internal Programmer for PC **$139.95**

New Intelligent Averaging Algorithm Programs 64A in 10 sec. 256 in 1 min 1 Meg (27010, 011) in 2 min 45 sec. 2 Meg (27C2001) in 5 min Internal card with external 40 pin ZIF

• Reads. Verifies, and programs 2716, 32, 32A. 64, 64A, 128, 128A 256, 512, 513 010, 011, 301 27C2001, MCM 68764, 2532, 4 Megabits
• Automatically sets **programming voltage**
• Load and save buffer to disk
• Binary, Intel Hex, and Motorola S formats
• **No personality** modules rewired
• 1 Year warranty
• 10 days money back guarantee
. Adapters available for 8748. 49 51, 751, 52. 55, TMS 7742, 27210, 57C1024, and memory cards
. Made in U S A



EMPDEMO.EXE available **BBS (916)** 972-804

NEEDHAM'S ELECTRONICS
4539 Orange Grave Ave -Sacramento. CA 95841
(Monday-Friday, 8 am-5 pm PST)
C.O.D.

Call for **more** information
(916) 924-8037
FAX (916) 972-9960

#14

DEPARTMENTS

70 Firmware Furnace

78 From the Bench

84 Silicon Update

92 Practical Algorithms

99 ConnecTime

Ed Nisley

Infrared Home Control Gateway

You can already control your TV, VCR, and CD player from the comfort of your chair using a hand-held infrared remote. Why not control your home as well? Ed shows us the firmware involved in doing just that.

Some folks have problems in places where most of us don't even have places. Consider the fate of the poor chap who fixed Steve's furnace last week: he didn't move around enough to keep the motion-sensing lights turned on. The house was on manual lighting control while the repairman was at work, if you can imagine that.

Now, suppose everybody in Steve's house wore a badge that broadcast a unique ID. The HCS II could then track each person and adjust the room lighting, sound system volume, security defenses, and so forth, depending on who was where. Finally he'd have a truly personalized, automated house that would work without manual adjustments.

Of course, the system can only track people (or, I suppose, dogs) wearing IR badges. But consider what the HCS II's response might be should the basement door open without a valid badge ID on either side.. . .

The topic this month is the firmware needed for an IR gateway to the new Circuit Cellar Home Control System II. It receives infrared signals from remote units and passes them to the Supervisory Controller (SC) for action, so you can control the HCS II without leaving your chair, as well as transmitting IR signals that can poll

B		Set badge response delay (default 40 ticks)
CR n		Calibrate remote **MC1 45030 bit clock** (12.4 **kHz)**
		n=0–3 sets report detail
CT		Calibrate transmitter oscillator
		connect 38 **kHz** output to TI input
D		Dump program status (debugging use)
E		Show and clear error flags (debugging use)
In		Set badge polling interval (default 190 ticks)
Ln		Set logging mode (bit mapped)
	L	report current mode
	LO	disable (default)
	L1	show received **IR** messages
	L2	show transmitted **IR** messages
	L4	show generated polls
Nn		Set network/interactive mode
	N	report current mode
	NO	set interactive **mode**
	N1	network mode (no error messages) (default)
	N2	network mode with command echo
0 x=n		Set output bit x to n
	OA=O	set bit A
	OB=1	set bit B
	OC=O	set bit C
	OD=1	set bit D
Pn		Set number of badges to poll (default 0, no polling)
Q		Query and reset received **IDs**
	a	dump all 512 **IDs** in hex (130 chars in line)
		ID 0 is bit 0 of first byte
	Qn	report ID n status in format **003=1**
	On-m	report **IDs** n through m in hex bytes
		ID n is bit 0 of first byte
RESET		perform power-on reset, must be completely spelled out
Sn		Send **IR** ID n

Figure 1—*The IRGATE* commands ***include functions to send and receive IR signals as well as control the firmware's operation. All times are in 5.12-ms* units. *and numeric values are decimal unless otherwise noted.***

specific "people tracker" badges. An HCS II system can have up to eight of these gateways on its RS-485 network, so you can put a different unit in each room to ensure adequate coverage.

GATEWAY FUNCTIONS

The overall structure of the IR Gateway (or IR-Link) firmware resembles that of the Smart X10 controller (or PL-Link) I described in the last issue. It receives and transmits serial information over an RS-485 network, includes a simple decoder to handle ASCII commands from the HCS II SC, and is written in Micro-C.

I've taken a bit of heat for abandoning 80.5 1 assembly language in this column. For the record, about half of the "Micro-C" source lines for the Smart X10 and IR Gateway firmware are actually assembler code, written using Micro-C's in-line assembler. As I continue to point out, C is good for the overall program logic, but not at all suitable for high-speed bit banging and interrupt handlers.

Figure 1 presents IRGATE's command set. As with the Smart X10 controller, there are commands to send and receive IR signals as well as control how the firmware operates.

IRGATE also includes calibration routines.

Steve's description of his IR hardware explains how the various IR remote units and badges work. He has the advantage of using the MC145030 chip directly; on the firmware end of the signal, there is essentially no hardware at all! As a result, I must discuss the MC145030 data format in some detail before explaining how the firmware transmits and receives it.

IR I/O

Figure 2 shows the MC145030 data format, adapted from the data sheet. The chip uses Manchester encoding, which defines two complementary signals for each data bit; Motorola calls the pair of signals a "bit frame" to indicate a single data bit. A complete message requires 39.5 bit frame times: 24 frames hold data or framing bits, while the remainder provide silent periods before, during, and after the "real" frames.

Motorola calls the contents of the message "address" bits because they think of the MC145030 as a widget to control a device at a specific address. I refer to them as "data" bits because our messages convey information to the HCS rather than selecting an address. Fortunately, the bits don't care what they are called and do the same thing regardless of their labels.

Steve chose the bit frame time to work correctly with the Sharp IS1U60 receiver, which specifies a 600-µs minimum On and Off time. Because a Manchester-encoded bit frame uses both states, the minimum frame time is 1200 µs. Steve picked 1290 us, which means the MC145030 uses a 12.4-kHz oscillator.

Converting an integer (between 0 and 511, as there are only nine data

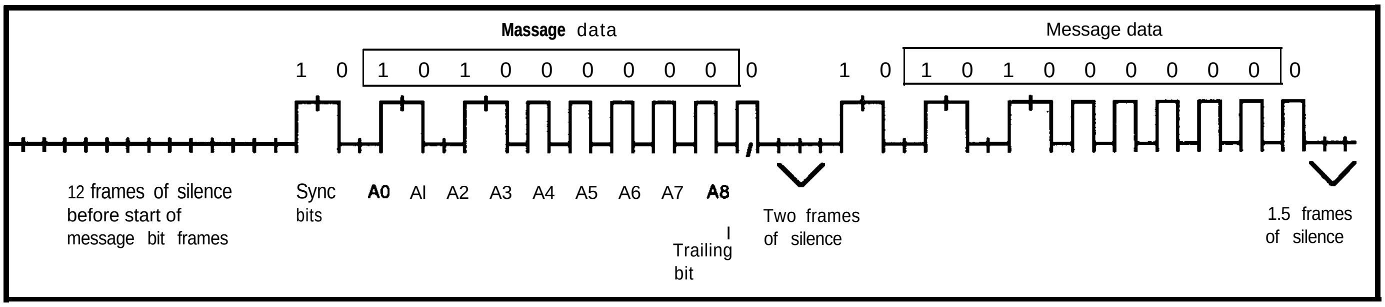


Figure 2-The *MC145030* chip *sends a single 9-bit number. A complete transmission includes two copies of the* number *amid various pauses and synchronizing Ms.*

bits available) into a message is straightforward, as shown by the three routines in Listing 1. Rather than pack the output bits into bytes, I squandered 79 bytes of the 8K External RAM on a C array called ITrnBuf. Each byte holds one bit of the encoded message, so each array element represents half of a Manchester bit frame.

The final line of IRFormatMsg() sets the FlagIRPending bit, which is monitored by a timer interrupt routine that ticks every 5.16 ms. When it sees FlagIRPending set on, it cranks the timer interrupt to 645 µs and shovels bits from the array to the output pin on each interrupt. After finishing the message, the interrupt handler resets the timer interrupt period to 5.16 ms.

The interrupt handler updates several software timers that count off intervals of a few seconds. The motivation for the peculiar 5.16-ms "normal" rate is that it is exactly eight times the 645 µs dictated by a 1.290-ms Manchester bit frame. After sending all 79 bits, the code simply adds 10 counts to the software timers. This addition keeps the timers reasonably accurate even if the firmware sends many IR messages.

Because the interrupt handler knows nothing about the message format, the higher-level code can send any bits it wants. In this application, I used the MC145030 data format, but similar code could mimic nearly any other chip. Such mimicking is especially valuable when the chip may change at any time, or when the requirement becomes "either of these two remote control chips" after the first one becomes obsolete.

Sending a message is as simple as issuing the "S" command to IRGATE over the serial link. The number after the "S" must be in the range 0 to 511. The firmware holds outgoing messages in a ring buffer, so it can accommodate bursts of commands from the SC or your program.

Listing 1—*These three functions translate an integer between 0 and 511 into the format required by the MC145030 chip. Each entry in the output array represents the state of the output pin for half of a 1290-µs bit frame. The entries are transferred to the output by an interrupt handler routine driven by the on-chip timer.*

```
/*-----------------------------------------------------------*/
/* Convert argument into Manchester data frame (two         */
/*   outputs per call)                                       */
/* Uses only low bit of argument to simplify the calls       */
/* Output is 1 for "IR On" and 0 for 'IR Off'                */

IRFormatFrame(MsgBit,pMsgFrame)
unsigned int MsgBit:
BYTE *pMsgFrame:
{
   if (0 == (MsgBit & 0x0001)){
      *pMsgFrame = 1:
      *(pMsgFrame+1) = 0;
   }
   else {
      *pMsgFrame = 0:
      *(pMsgFrame+1) = 1:
   }
}
/*-----------------------------------------------------------*/
/* Convert argument into Manchester-encoded burst at ptr */

IRFormatBurst(MsgNum,pMsgBase)
unsigned int MsgNum;
BYTE *pMsgBase:
{
BYTE Index:

   IRFormatFrame(1,pMsgBase+0);           /* start bits */
   IRFormatFrame(0,pMsgBase+2);
   pMsgBase += 4:

   for (Index=0; Index<NUMDATABITS; ++Index) { /* encode msg */
      IRFormatFrame(MsgNum,pMsgBase);
      MsgNum >>= 1:
      pMsgBase += 2:
   }
   IRFormatFrame(0,pMsgBase);          /* trailing bit */
}
/*-----------------------------------------------------------*/
/* Convert integer into bit pattern in the transmitter buffer */

IRFormatMsg(MsgNum)
WORD MsgNum:
{
   memset(ITrnBuff,0,sizeof(ITrnBuff)); /* flush previous bits */

   IRFormatBurst(MsgNum,ITrnBuff+24);   /* first transmission */
   IRFormatBurst(MsgNum,ITrnBuff+52);   /* repeat one time... */

   INumSend = 79:             /* how many half-frames in buffer */
   SETBIT(FlagIRPending);     /* indicate ready to go           */
}
```

BITS FROM THE ETHER

Receiving bits from a remote MC145030 transmitter is somewhat more difficult. A Sharp IS1U60 IR receiver translates the IR signal into a TTL voltage for the 8031's INT0 input pin. The firmware sets up INT0 to produce an interrupt for each downgoing edge, but a little study of Figure 2 will show that the Manchester data format does not produce an interrupt for each bit frame.

The firmware can take advantage of the bit frame timing to anticipate when each bit should arrive. The first sync bit produces an interrupt in the middle of the bit frame that defines when all of the other bits should occur. The INT0 intdrrupt handler sets the hardware timer to interrupt one quarter of a bit frame later, in the middle of the last half of the first bit frame, and sets a variable to indicate that reception is in progress.

The timer interrupt handler then begins sampling the input and verifying that the data matches the MC145030 data format. Each sample must be followed by its complement (01 or 10) to form a valid bit frame, and the silent periods (00) must occur at the right times. During the two groups of frames holding data bits, the interrupt handler shifts the second sample of each frame into a pair of 16-bit variables.

After the final silent period, the firmware compares the two variables to ensure that both data values are equal. If so, and if all the silent periods were truly silent, the firmware adds the data value to the receiver's ring buffer for later analysis by the higher-level C code.

The INTO input continues to generate interrupts whenever the IR signal goes on, so the firmware puts that information to good use. The timer should be midway in its count to the next sample at each negative transition, so the INTO handler reloads it with exactly one-quarter of the bit frame time [think about it), which allows the firmware to track remote units with slightly off-frequency oscillators. Timing the interrupts this way ensures the data samples occur in the "middle" of each half of the bit frame, where they are least likely to be corrupted by noise or timing errors.

Because the receiver and transmitter use the same hardware timer in different ways, IRGATE cannot receive while it is transmitting, so, unlike Smart X10, IRGATB cannot monitor its own output. The INTO handler checks the timer handler's state variable before clobbering the timer; the valid state transitions are Idle→Receive and Idle→Transmit, but not Receive→Transmit or vice versa.

IRGATE records all received codes in a table that it displays when you issue the "Q" [query) command over the serial link. There are several different formats so you can get the values you need in the fewest characters. After sending the table's contents, IRGATB clears it so each "Q" reports only new codes.

BADGE TRACKING

Although IRGATE's main purpose is receiving inputs from push-button remotes to control Steve's new HCS, I included all the functions you need to build a system that can track people wearing IR badges. Ken's SC has code built into it to support active badges (more in a moment), and allows you to initiate actions based on what room a particular badge is in, so the groundwork is done.

As far as IRGATE is concerned, there are two types of badges: active and passive. Active badges are essentially identical to the remote units described above, except they transmit their ID code (0 through 5 11) periodically. Passive badges transmit their ID only when they "hear" that ID from another source.

Active badges are quite simple, because they do not need any receiver hardware. However, because each badge transmits without regard for the others, a room full of badges might produce no recognizable messages due to all the collisions. You must pick the repeat rate to minimize the number of collisions while not introducing too much delay in determining when a new badge enters or leaves the room.

Passive badges, on the other hand, must include a receiver that is active continuously, so they will dram their batteries faster. In addition, IRGATE must know which badge IDs to send, because a passive badge responds only to its own ID code. While there are no collisions, IRGATE must poll all possible badge IDs to find out which ones are within range.

I'll just point out that designing the badge hardware is not easy and leave it at that. The receiver and transmitter hardware used by IRGATB need the changes Steve describes in his article to ensure it can cover an entire room.

If your system uses active badges, IRGATB needs no changes because it cannot tell the difference between a remote control unit and a badge. After all, there are only 5 12 possible codes and the bits mean whatever you want them to. Your controller must poll

Want low power?

Is 10 microwatts low enough? (That's not for a CPU chip, but for a complete *anoLINK* Controller *board* in a typical application. The automatic power-cyclin technology that makes that possible is standard and built-in.)

Fast development?

What if we write the device drivers for you? And the timing-critical, low-level code? And throw in a complete BIOS? And a full monitor with source code? And what if the hardware includes all the real-world interfaces you'll probably need, like A/Ds, buffered digital I/O, and high power switching?

And low cost?

What if the standard board package *includes* a macroassembler? An integrate editor? A communications interface? Complete, illustrated technical references What if the hardware includes a switching supply and even the serial cable? What if you don't even need an EPROM programmer?

There <u>is</u> a better way.

For information, call:
1-800-GET-DATA (4383282)
or
602-996-0255 (FAX)

nano | Power Devices

A division of AirDigital Corporatio

146

IMAGENATION...

VIDEO FRAME GRABBERS

- ◆ Simplicity
- ◆ Functionality
- ◆ Affordability
- ◆ Accuracy

Cortex-I $495.00

- ◆ Real-Time Capture
- ◆ Half Slot XT/AT
- ◆ 512 x 484 x 8 Bit
- ◆ RS-170/CCIR
- ◆ External Trigger

New

Cortex-STD $595.00

- ◆ Dual Video input
- ◆ Opt. XMS Mapped
- ● Low Power Options
- ◆ STD-80 or 32 Bus
- ◆ External Trigger

New

Vidmux-4 $99.00

- ◆ 4 to 1 MUX
- ◆ Half Slot XT/AT

Includes Software...

- ◆ C Library & Source
- ◆ Image Capture Utility
- ◆ Tiff Utilities
- ◆ "Image" Drive Ram Disk Emulation

STARTER PAK...

Cortex-Pak $1395.00

- ◆ NEC/TI-23EX Camera With Lens
- ◆ 9" Video Monitor
- ◆ Frame Grabber
- ◆ Software & Cables

OEM PRICING AVAILABLE

IMAGENATION CORP.
P.O. Box 84568
Vancouver, WA 98684

PH/FX (206) 944-9 13 1

#147

Listing 2—*Measuring the remote MC145030's oscillator frequency requires determining the bit frame time based on the received data. This firmware routine analyzes an array holding the time of each negative transition in the IR signal. The first element of the array is the length of the pulse formed by the first two sync bit frames. All the debugging output statements controlled by ShowDetail are removed to save space in this listing.*

```
IRCalRemote(ShowDetail)
int ShowDetail;
{

BYTE FrameNum,FrameCount,Index;
unsigned int FrameTime,FrameTime1,FrameTime2,AvgFrTime;

  do {
    memset(IRTimes,0,sizeof(IRTimes));
    SerWaitOutDone();

    IRSample();                /* get the IR transition times */

/*-- get differential time between each sample                  */
/*   [0] is start bit frame time; estimates true frame time     */
/*   [1] is time since the start.  so it is already a delta time */

    for (FrameNum = MAXIRFRAMES-1; FrameNum > 1: --FrameNum) {
      IRTimes[FrameNum] = IRTimes[FrameNum]- IRTimes[FrameNum-1];
    }

/*-- convert times from timer ticks to microseconds              */
/*   magic number is 1.085 microsecs per timer tick. but we      */
/*   must do it in two tiny chunks to avoid overflowing 16 bits */

    for (FrameNum = 0; FrameNum < MAXIRFRAMES; ++FrameNum){
      FrameTime1 = IRTimes[FrameNum] * 2:              /* 0.08  */
      FrameTime1 = (FrameTime1 + 12) / 25;
      FrameTime  = (IRTimes[FrameNum] + 100)/200:      /* 0.005 */
      IRTimes[FrameNum] += FrameTime1 + FrameTime2:    /* 1.085 */
    }

/*-- set up initial frame sorting thresholds                     */
/*   this is done in chunks to avoid overflows. too              */
/*   first frame is special-cased sample. needs 80% derating to */
/*    real time because the trailing edge is delayed by IR      */
/*    receiver                                                   */

    FrameTime1 = IRTimes[0] * 8;
    FrameTime1 = FrameTime1 / 10;
    IRTimes[0] = FrameTime1;
    for (Index = 0: Index < NUMTHRESH; ++Index){
      IRThresh[Index] = IRTimes[0] * (2*(Index+1) + 1);
      IRThresh[Index] >>= 2:
    }

/*-- now find the average frame time                             */
/*   determine each interrupt duration                           */
/*    (1.0. 1.5. 2.0... bit frame times)                         */

    AvgFrTime = 0:
    FrameCount = 0;
    for (FrameNum = 1;
         (FrameNum < MAXIRFRAMES) && (IRTimes[FrameNum]);
         ++FrameNum) {
      if (IRTimes[FrameNum] <= IRThresh[0]) {
        continue;
      }
      else {
        if (IRTimes[FrameNum] > IRThresh[NUMTHRESH-1]) {
          continue;
        }
      }
```

(continued)

Listing P-continued

```
      for (Index = 1;  Index  < (NUMTHRESH-1); ++Index){
        if (IRTimes[FrameNum] <= IRThresh[Index]) {
          FrameTime = IRTimes[FrameNum] * 2;
          FrameTime /= Index + 1;
          AvgFrTime += FrameTime;
          ++FrameCount;
          break;
        }
      }
    }
/*--- figure average frame time and display the result         */

    AvgFrTime /= FrameCount;
    sprintf(CmdRespBuff," Bit frame time is %u us\n",AvgFrTime);
    Nputstr(CmdRespBuff);

  } while (!chkch());                          /* assume interrupts */

  SerFlushIn();                                /*  discard  inputs  */
}
```

IRGATE to determine which ID codes have been "seen" since the last poll, and it must also determine how to handle new and missing IDs.

The "P" command tells IRGATE how many passive badges require polling. The default is "PO," or no polling. Polling starts with ID 0 and goes up to (# badges) – 1, so "P10" will poll badges 0 through 9 inclusive. Your badges must respond to those ID codes for this feature to work!

The "I" command sets the polling interval, with the default being two seconds when polling is enabled. You should adjust this value to suit your purposes, taking into consideration battery life and response time. The interval is the time between successive polls, so polling badges 0 through 9 with a 2-second interval would take 20 seconds. The minimum value is 20 ticks, which is about 100 ms.

The "B" command sets the amount of time IRGATE waits for a badge response after sending an ID code. The default is 40 clock ticks [about 200 ms). When polling is not active, this delay determines the minimum time between IRGATE transmissions.

You can use remote control units as well as badges in the same system if you choose the ID codes carefully. I suggest you put all your control codes in the range 256-511 so the high bit distinguishes badges from remote controls and the low byte holds the entire ID code. If you are using passive badges, you must allow enough "dead air" between polls to accommodate remote control inputs; set the polling

Affordable 8031 Development

Single Board Computers, Assemblers, Compilers, Simulators, and EPROM Emulators

Control-R Series, Single Board Computers

Two models of Control-R series computers make prototyping, one of a kind products, or small production runs easy and economical. Both feature RS232 compatible serial ports, single 5 volt supply operation, and direct access to Ports 1 and 3 of the 8031. Additional features are as follows:

Control-R Model 1 $49.95

Fully populated board with I/O header for Ports 1 and 3, serial port, and 8K EPROM socket. 3.0" x 4.0"

Control-R Model 2 $79.95

Same features as the Control-R 1 plus 8K of SRAM and expansion bus with data, address, RST, INT1, WR, RD, PSEN, ALE and T1. 3.5" x 4.5"

Software and Hardware Development Tools

Control-C 8031 Cross-Compiler $200.00

The Control-C 8031 cross-compiler is a full featured K&R style C development system available at an affordable price. Optimized for embedded system use, it will produce ROMable code for any 8051 based system including designs using only the 128 bytes of internal RAM. Package includes compiler, pre-processor, assembler, simulator, printed documentation and complete library source code. Requires IBM PC or compatible. 5.25", 360K disk.

PROMulator 256 $189.95

An EPROM emulator lets you avoid "Bum and Test" development cycles. In circuit emulation of 2K-32K 27xx series EPROMs. ABS Plastic case. Assembled or compiled code is downloaded directly to the target hardware.

Cottage Resources Corporation
Suite 151, 10271 South 1300 East
Sandy, Utah 84094
VISA/MC, COD. Call to Order: (801) 268 - 2875

#148

interval to 5 or 10 seconds to keep from clobbering yourself.

CALIBRATION CONSTANTS

Although I describe the IR signal as being either "on" or "off" when transmitting or receiving a message, that is not quite accurate. The IS1U60 receiver expects the IR signal to have a 38-kHz modulation, so an "on" signal is actually a burst of 38-kHz IR pulses.

Producing the modulating frequency with firmware isn't feasible (at least not while monitoring a serial network connection!), so Steve added some hardware to chop the output signal. But he insisted on a way to calibrate the oscillator against the 803 1's 11.0592-MHz crystal rather than use a frequency counter or scope.

So IRGATE includes a transmitter calibration routine that displays the frequency of that modulating signal. To use it, just connect the oscillator to the 8031 T1 input pin, enter the "CT" command, and tune the frequency. You even get a simple tuning graph.. . try it and see!

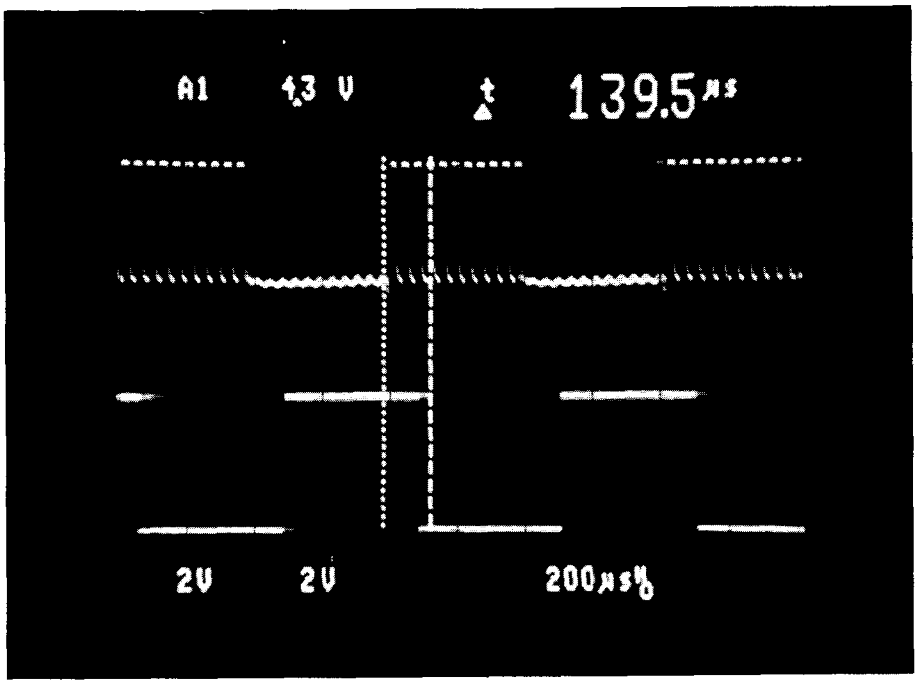
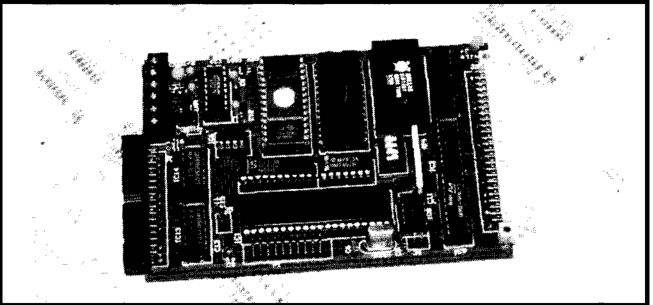


Photo 1-The *IS1U60 needs* some time *to decide that it is seeing en IR signal end more time to decide that if is not The top trace shows /he raw IR signal, while the bottom trace shows the IS1U60's output.*

A second calibration routine measures and displays the remote unit's bit frame time. The nominal value is 1290 us, but if you study Figure 2 again, you'll see how difficult it is to determine this value; remember you don't get an interrupt for each frame and you don't know what the data is!

ProControl



Introducing the Most Expandable 3 1/2 Board Available

MCU-31/2 from $149.95

On Board Options Include:
- ✖ 16 Channels 10 bit A/D -14µs w/S/H
- ✖ Dallas 1287 RTC
- ✖ Flexible mem config's: RAM - 8/32KB ROM - 8/16/32/64 KB
- ✖ RS-232 or 485 -jumper selectable
- ✖ Watchdog timer w/ jumper sel. reset source
- ✖ Pre decoded external bus for very easy user interface - directly compatible with other ADS boards (see back issues of Circuit Cellar INK)

More I/O modules are availble. Call for our FREE catalogue today!

(404) 352-4788

ADS Advanced Design Solutions
1920 Moores Mill Road
Atlanta, GA 303 18

#14

FAST CODE RELIEF!

Safe and effective for the whole 8051 family.

BCI51 Basic Compiler.
- •Soothes assembly and C irritation
- *Long-lasting relief of time and budget headaches
- *Maximum strength error handling
- •The affordable cure for long code winters!

Fast integer math-supplement
BASIC-52 or stand-alone
outstanding technical support.

Only $299 *(Assembler & utilities included)*

Dallas DS5000 extensions $149
Assembly Language Toolkit $149

Call 801-487-7412
FAX: **801-487-3130**

SYSTRONIX Inc.
754 East Roosevelt Ave. Salt Lake City, Ut. 84105

#150

My first inclination was to measure the pulse produced by the first two sync bits, which looks like it should be exactly one bit frame long. Unfortunately, the IS1U60 cannot switch instantly, because it requires some time to decide that it is seeing an IR signal and more time to decide that it is not. Photo 1 is a magnified view of a few IR bursts to show the IS1U60 response time.

I finally realized that the time between successive negative transitions of the IR signal must be 1.O, 1.5, or 2.0 bit frame times, so the code could extract the frame time without knowing the data values. The trick is to use the length of the initial sync pulse to estimate the frame time, then classify each transition time and compute the average frame time. The code in Listing 2 does just that.

Because Micro-C does not yet implement long integer (32-bit) variables, some of the computations are more laborious than you might expect. As an exercise, compute the frequency (near 12400 Hz) from the average frame time (near 1290 us) without either overflowing 16 bits or discarding most of the significant bits.

RELEASE NOTES

The BBS files this time include IRGATE.HEX, which is the EPROM data for the full-function IR gateway firmware. The source code for IRGATE.HEX is available for licensing from Circuit Cellar Inc.

Also included are the Micro-C source files needed to create IRMON.HEX, which is a receive-only, nonnetworked version of IRGATE. It will report all the MC145030 codes it "sees" and includes the calibration function needed to adjust the remote unit's oscillator to 12.4 kHz. You can use this code as the basis for receiving and decoding other remote control codes, if you are inclined to use a different chip.

Next time, a networked Home Control System LCD panel and keypad, so the HCS II Supervisory Controller can show you what it's up to and you can give it suggestions. ❑

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

SOURCE

Please see page 3 1 for more information about the availability of HCS II components.

SOFTWARE

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

I R S

419 Very Useful
420 Moderately Useful
421 Not Useful

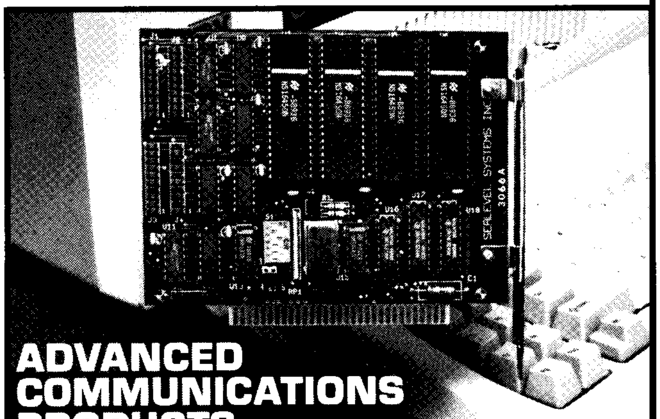
The *only* 8051/52 BASIC compiler that is 100 % BASIC 52 Compatible *and* has full floating point, integer, byte & bit variables.

- *Memory mapped variables*
- *In-line assembly language option*
- *Compile time switch to select 805 1/803 1 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 SXA5 1 cross-assembler & hex file manip. util.*
- *Extensive documentation*
- *Tutorial included*
- *Runs on IBM-PC/XT or compatibile*
- *Compatible with all 8051 variants*
- ***BXC51 $ 295.***

603-469-3232
FAX: 603-469-3530

Binary Technology, Inc.
Main Street . P.O Box 67 • Meriden, NH 03770

#151



ADVANCED COMMUNICATIONS PRODUCTS

SEALEVEL SYSTEMS PROVIDES THE EXACT COMMUNICATION CARDS YOU NEED.
PRODUCTS:
- 1, 2 OR 4 PORT RS-232 AND RS-422/485 BOARDS
- CURRENT LOOP SERIAL INTERFACES
- HIGH SPEED SYNC (HDLC, SDLC) AND ASYNC WITH DMA
- RS-530 AND V.35 INTERFACE BOARDS
- DIGITAL AND RELAY I/O BOARDS.
- DISKLESS EPROM BOARD WITH PROMKIT SOFTWARE BY ANNABOOKS.
- NEW LAP-TOP ADD ONS! • MADE IN USA
- DELIVERY FROM STOCK • SATISFACTION GUARANTEED
- EXCELLENT TECHNICAL SUPPORT

SEALEVEL
COMMUNICATIONS & I/O

SEALEVEL SYSTEMS INC.
PO BOX 830
LIBERTY, SC 29657
(803) 843-4343

#152

Does It Come With a Memory... Standard?

Part 2: The Nitty-Gritty

In the last issue, Jeff introduced us to the PCMCIA memory card standard. This time, he shows us how to build an interface that should stand the test of time (assuming the standard does, too).

FROM THE BENCH

Jeff Bachiochi

Problem: logging data over any period of time, whether it be temperature patterns, device status, or caloric intake, often requires more storage space than an inexpensive microcontroller has to offer. I needed a simple way to extend the amount of data I could gather and provide a semipermanent record of that data.

Solution: Although disk storage is not rugged enough for many environments, it does have the advantage of a removable, semipermanent medium. Nonvolatile memory devices overcome the environmental problems associated with disk storage, yet still provide "shirtpocket" transportation and archiving of data.

In my last column, I revealed the arrival of a new interface standard for solid-state memory cards. Although it had its start back in the '80s, this standard never took off [like it should have) due to a lack of interface specifications. After a bit of refinement and functional diversification, the hardware specifications include I/O devices as well as the original memory interface. Software that recognizes and takes full advantage of attached devices is still under development at the upper layers, but using the interface at its lowest levels with assorted memory cards requires little effort. I believe that designing with the PCMCIA interface will extend the life of any project by ensuring compatible memory cards do not become obsolete.

A DROP IN THE BUCKET

The term *bit bucket* has come to mean a place where unused data is tossed, gone forever, like a write-only device. However, this project brings new meaning to the term, allowing you to retrieve all the data in a FIFO-like fashion. The design includes a micro (8031) with five additional latches to increase the normal 64K data space associated with most 8-bit micros to a possible 64 megabytes. See Figure 1 for the schematic.

Data I/O is handled in both of the traditional methods available to most host systems: serial and parallel. An RS-232 serial port is supported using a MAX232 for level conversion. (This dual transmitter/receiver 5-volt converter replaces a 1488, a 1489, and a bipolar supply.) Data is transmitted or received at a predetermined data rate of 9600 bps [or any other rate of your choice) with no handshaking necessary. This rate results in a storage routine that must be completed within one character reception time or about 1 ms (at 9600 bps) or data will be lost. Also, the host must be able to digest the data being played back at this rate.

Parallel data transfers have two advantages. The first is the number of data paths used. While serial has a single path or bit stream, parallel has eight paths or a byte stream. This difference can mean higher throughput. The second advantage is handshaking. The *STroBe and ● Acknowledge play a "please and thank you" game to ensure good data transfers using two additional control lines.

Supporting SRAM modules up to 64K is a simple and direct task. Going beyond the 64K limit of an 8-bit micro's address range requires bank switching. In most bank-switched applications, only part of the total 64K data space is switched. An unswitched portion of RAM is used for variable storage, while the remaining part is switched; a nice boundary of, say, 32K might be used. Because this microcontroller has nothing to do but store and retrieve data, and the variables and stack all fit within the 803 1, I can bank switch the total 64K of (data) space. In fact; an 8751 could be used, which would eliminate the need for an external EPROM. The total code is a few hundred bytes.

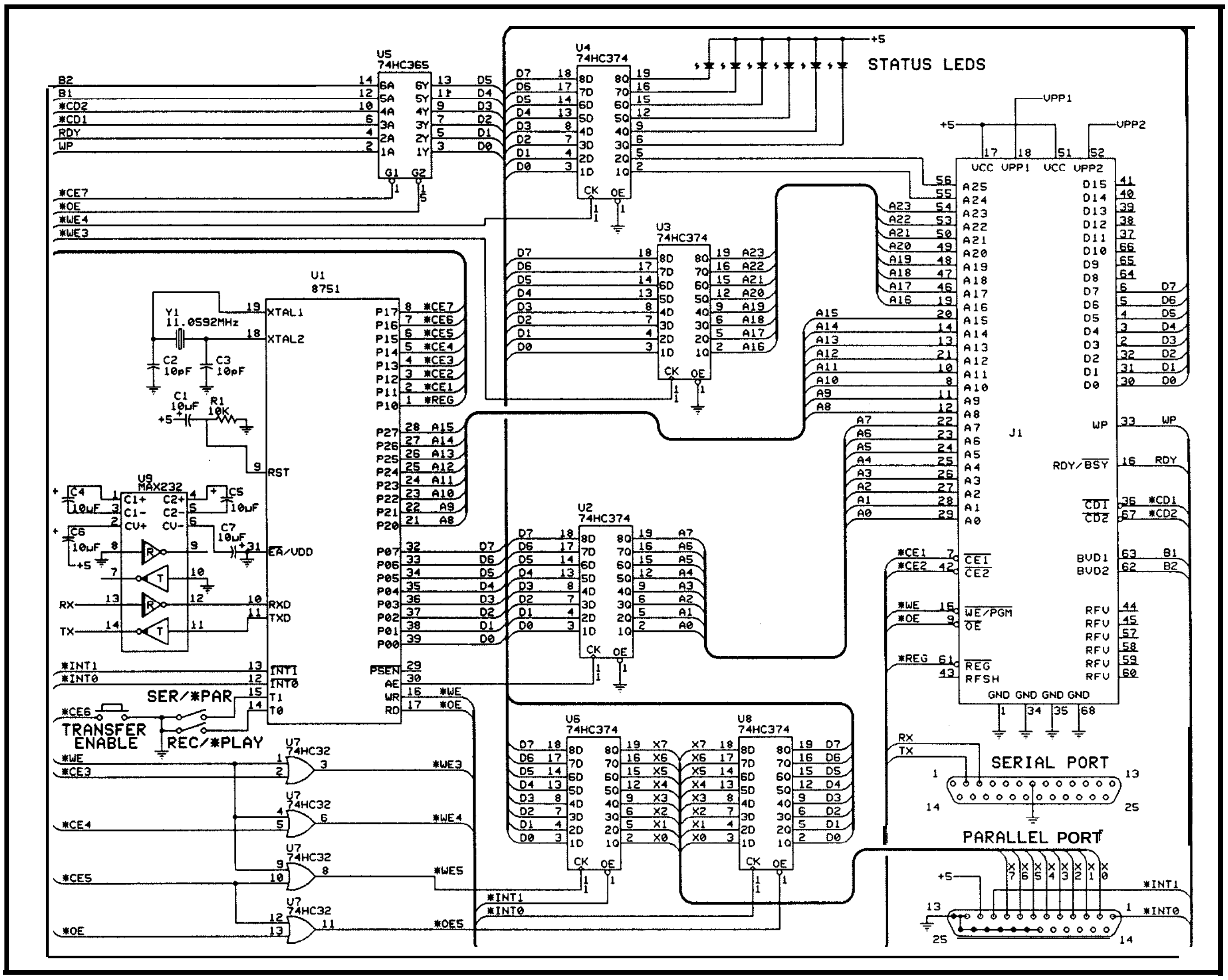


Figure I--The *Bit Bucket is based on an* **8751** *(with built-in* **EPROM)** *and several* ***latches that*** *make **it possible to** address more **than 64K** of **card memory**. Data may be exchanged **with the unit through either** a serial or a **parallel interface**.*

SECRET INGREDIENT

There are no secrets here. The PC Memory Card International Association has the hardware well documented. The interface to the PCMCIA card socket consists of 26 address lines, 16 data lines [I've chosen to only use an 8-bit data path), 5 control lines into the socket, and 6x status lines out of the socket.

The five control inputs are as follows: ● CEl and ● CE2 are chip enables (one for the lower 8 bits of data and the other for the upper 8 bits when using a 16-bit path]. ● OE is the output enable and WE is the write enable (*PGM for programmable devices). ● REG is the alternate memory area normally set aside for device recognition.

The six status lines are configured as follows: *CDl and *CD2 are grounded lines (within a memory card) used to indicate when the card is fully inserted. [Last time, I mentioned the unique arrangement of multilength contacts used in the interface socket. The longest pins apply power to the memory card first, the medium pins connect all signals second, and the shortest pins, *CDl and ● CD2, are used to signal the card is fully inserted.) BVDl and BVD2 indicate the state of the battery and whether the data held within the memory card could be corrupt. WP reflects the status of the write protect switch on the card. RDY/*BSY is used by programmable memory devices to relate the status of the programming cycle.

Two 74LS374 8-bit registers are used as address latches for the address lines above the stock AO-A15 (64K space]. Address lines A16–A23 are held in the first latch and A24–A25 in the second latch. The remaining 6 bits of the second latch are used for LED status indicators. The LEDs reflect the status of the battery in the SRAM memory module, the write protect switch, the memory card insertion detection inputs, the address pointer (end of RAM), and storage or retrieval activity. See Figure 2 for an explanation of each LED.

A 74LS365 6-bit input buffer reflects the output logic levels from the PCMCIA's interface adapter. The status outputs include battery status, write-protect switch position, memory card insertion detection, and ready or busy for programmable devices.

The two remaining latches set up a Centronics-compatible parallel I/O port. Two external interrupt lines are used as handshaking for the port. (Note: most PC parallel ports are not 8-bit bidirectional. Some discrete ports can be altered to tristate the output latch with an unused chip select. See *Circuit Cellar INK,* issue #2 [March/April 1988] for the printer port modification used in the all too famous "Circuit Cellar Neighborhood Strategic Defense Initiative." This "cut and jumper" fix has been used in a number of projects discussed here in *The Computer Applications Journal.*) *If you* have a bidirectional parallel port, you can handshake data both into and out of the bit bucket. If you have a unidirectional port, parallel data can be saved, but you'll have to use a serial connection to "suck" the bits back out one at a time.

Front-Panel LEDs

LED7 On	End of RAM, data transfer halted
LED6 On	Cannot record, card is write protected
LED5 Off	Low battery, system halted
LED4 Off	Data corrupt, system halted
LED3 Off	No memory card installed, system halted
LED2	Toggles with each data byte transferred

Rear-Panel Switches

Push Button1	Transfer enable
Switch1	Record/Playback mode
Switch2	Serial/Parallel mode

Figure 2-M *Bucket LED indicators and configuration switches.*

Just about any micro can be used in this project. I chose to base it on an 803 1 because of its built-in UART and the availability of a 44-pin PLCC package. (If you really need to combine functions for an even smaller design, there is an 875 1 PLCC version.) The latches can be replaced with a couple of 8255s (PLCCs available here, too), but I think you'll find the latches easier to work with, cheaper, and able to drive LEDs directly.

THE FIRMWARE

If you've been following this column for a while, you know my language of choice is BASIC whenever possible. It's quick to write, easy to follow (even without comments), and its execution speed usually isn't a concern. This time speed is top priority. The serial reception won't wait for any slow code. (Multiple characters will be received in the time it takes BASIC to execute one line of code.)

There are only three routines written for this project. The first is initialization code to get the 803 1 ready for the task at hand. Here I give myself some breathing room by moving the stack pointer up to 21H. I'm not using all the registers below the stack, but I do want to use the bit-

smx simple multitasking executive v2.1

FULL FEATURED KERNEL

- ❑ task manager
- ❑ intertask comm.
- ❑ i/o, events, & timing
- ❑ preemptive
- ❑ memory manager
- ❑ error manager
- ❑ resource manager
- ❑ ROM'able

EASY TO USE

- ❑ libraries for Microsoft C, Borland C/C++, Zortech C/C++ and assembler
- ❑ Quick Start, User's Guide, and Reference manuals
- ❑ standalone, PC, and DOS platforms
- ❑ *smxProbe* task debugger
- ❑ 6 months free support and updates

FAST & SMALL

- ❑ 15 usec max. interrupt latency
- ❑ 150 usec typical task switch
- ❑ 8 to 25 KB code size

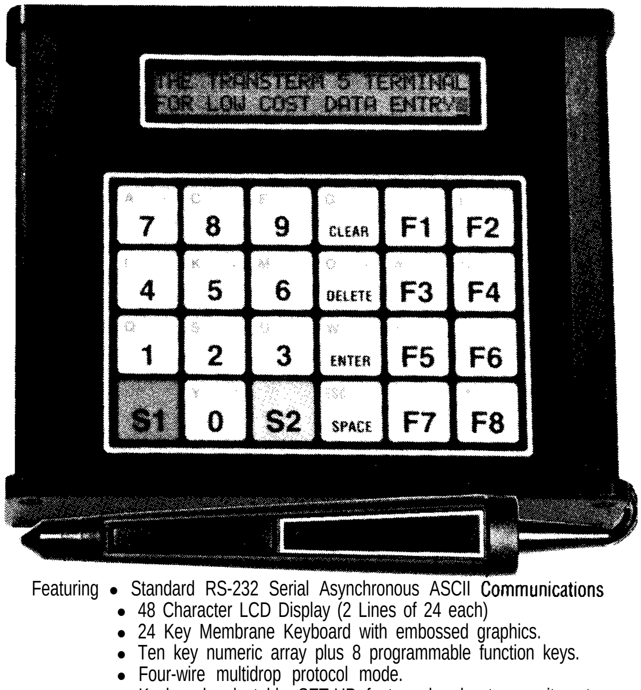
Ask about our $95 evaluation kit.

MICRO pd DIGITAL
6402 Tulagi St.
Cypress CA 90630-5630

Call for information:
1-800-366-2491
FAX 714-891-2363

#1

$249.TERMINAL



Featuring
- Standard RS-232 Serial Asynchronous ASCII Communications
- 48 Character LCD Display (2 Lines of 24 each)
- 24 Key Membrane Keyboard with embossed graphics.
- Ten key numeric array plus 8 programmable function keys.
- Four-wire multidrop protocol mode.
- Keyboard selectable SET-UP features-baud rates, parity, etc.
- Size (5.625" W × 6.9" D × 1.75" H), Weight 1.25 lbs.
- 5 × 7 Dot Matrix font with underline cursor
- Displays 96 Character ASCII Set (upper and lower case)

Options—backlighting for display, RS-422 I/O, 20 Ma current loop I/O,

COMPUTERWISE, INC.
302 N. Winchester . Olathe, KS 66062 . 913-829-0600 . 800-255-3739

#154

addressable ones starting at byte 20H for the status information.

The serial port is set up to use an 8-bit data word using Timer 1 in the 8-bit autoreload mode at 9600 bps. The internal registers used to reflect the present state of address lines Al 6–A25 (bank select) are cleared and the memory card size registers used for the end of RAM detection are set up.

Now the first external address latch is updated from the internal register with A16–A23. The front-panel status LEDs are on the upper 6 bits of the second external address latch, which holds A24 and A25. A check is made on the status of the memory card. The appropriate bits are masked with A24 and A25, and then latched into the second external address latch. If the status is not OK, a loop is taken back to recheck the status from the memory card. If all is well, the ***transfer enable*** push button is checked.

Once the button is pushed, a check is made on the ***transfer mode.*** If the ***parallel mode*** has been selected, a jump is made to the P-START routine. If the transfer mode is serial, serial interrupts are enabled and a check is made on the ***function mode.*** If the function is record, then jump to S-START; other wise, the function is playback, so set the transmit interrupt (which indicates the transmitter is empty] and fall into S-START. S-START is an endless loop where code hangs out while not in the serial interrupt routine. Setting the TI bit will cause the serial interrupt to be entered in the ***serial playback mode*** as will a received character in the ***serial record mode.***

S-START **(Serial Routine)**

A check must be made to determine the source of the serial interrupt because there is only one serial interrupt vector. If it's from a received character, then clear the interrupt, save the character in a temporary register, and check for the end of RAM. If the end is flagged, then exit the interrupt without saving the data; otherwise, save it to the memory card, set up the next address, and do a RETI.

If the serial interrupt source is from a transmitted character, clear the interrupt and check for the end of RAM. If the end is flagged, then exit the interrupt without sending any data; otherwise, retrieve a byte of data from the memory card, place it into SBU F (which will start the serial transmission), set up the next address, and do a RETI.

P-START **(Parallel Routine)**

Once thrown into the P-START routine, a check of the function mode

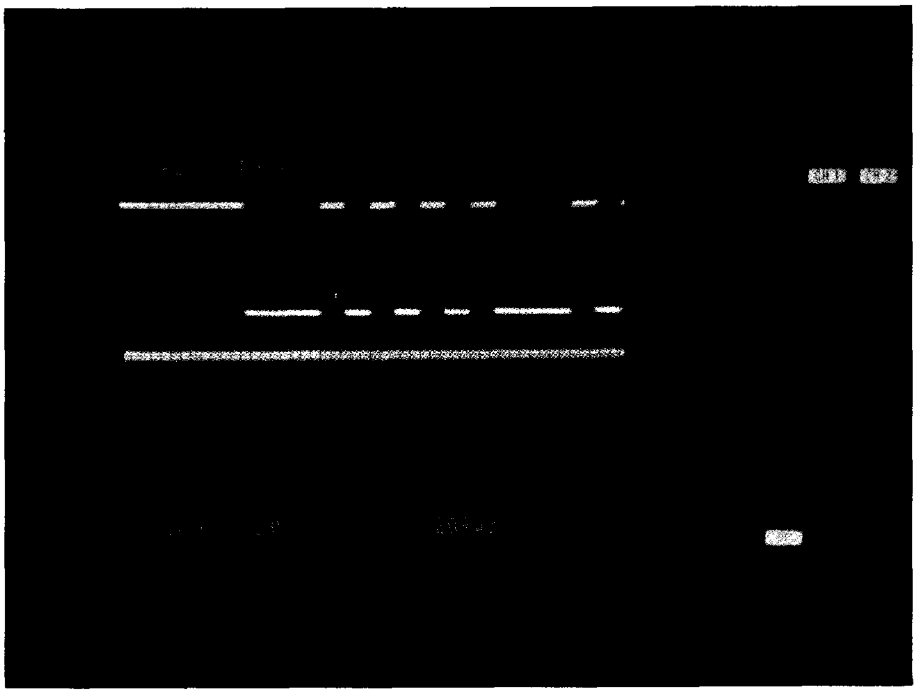


Photo 1-A *9600-bps character transmission* **(top trace) is** *shown* **in relation to the chip** *select* **used to** *enable he SRAM memory card (*CE, bottom trace)* **in he** *serial record mode.*

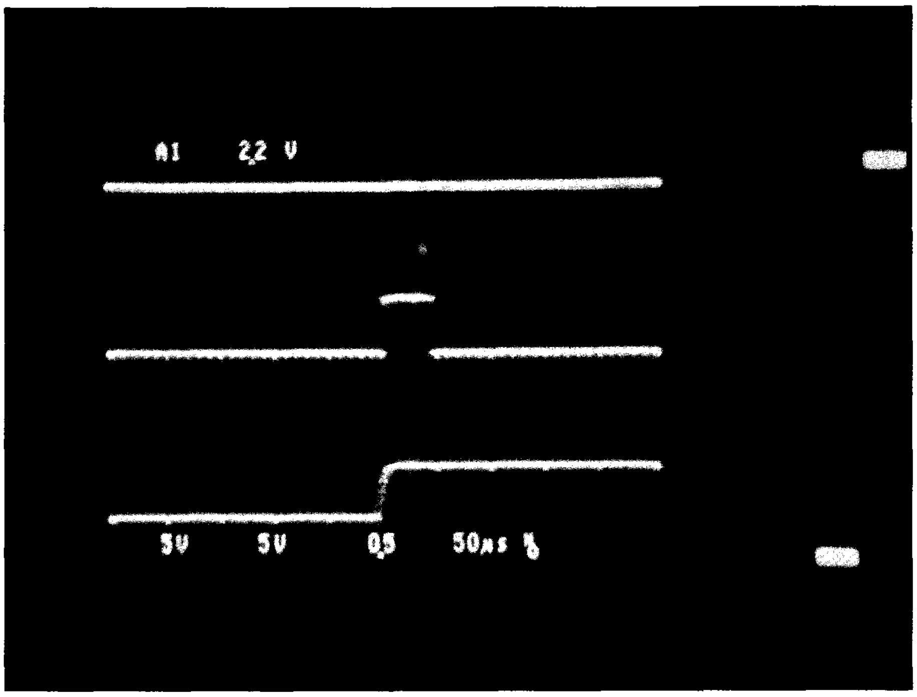


Photo 2-The *parallel* playback *mode requires hardware handshaking between the* ***Bit Bucket*** **and a** ***PC's parallel*** *interface.* ***Characters*** *are* ***grabbed (*CE1,*** *top trace) from* ***the SRAM*** *memory card and* ***placed*** *info* ***the output*** *register.* ****INT1*** *(middle trace) is lowered, signaling We PC that a character is ready. The PC reads* ***its* parallel *port*** *and* ***acknowledges*** *by lowering and* ***raising*** *ifs* ***strobe*** *line ('INTO,* ***lower trace).*** *The* ***Bit*** *Bucket* ***raises *INT1*** *and the* ***cycle is*** *repeated for each additional character.*

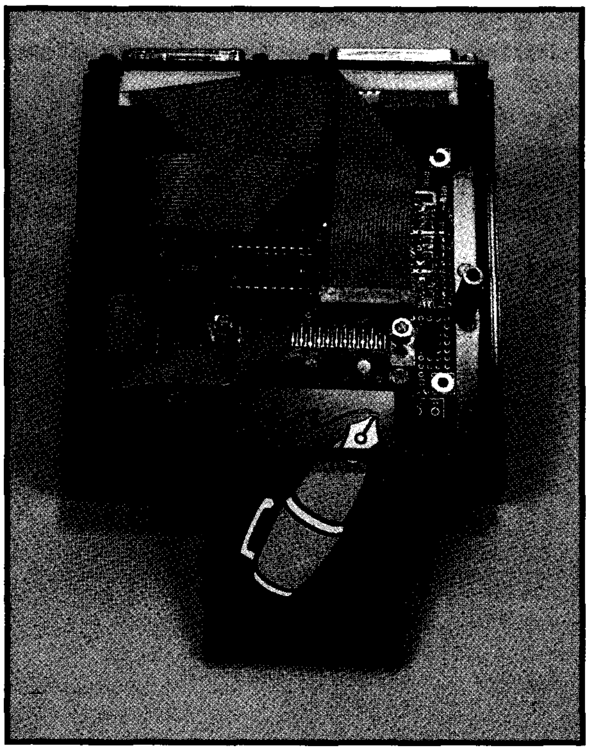


Photo 3—*All the circuitry for the Bit Bucket (minus he "wall wart" power supply) fits into a small 5-inch-square Pactec enclosure. Operating current is less than 200 mA.*

directs execution into the ***parallel record mode*** or the ***parallel playback mode.*** Like the serial record mode, an endless loop is entered at **P-RECORD** while waiting for a character. In the parallel record mode, a character is ready when the external parallel port activates a strobe. The strobe is tied to the ***INTO** line on the microcontroller, which triggers an external interrupt. The interrupt occurs on the falling edge and a loop is entered that waits for the rising edge (an indication of "data ready"]. If an end-of-RAM check says "that's it," then exit without saving the data; otherwise, get and save the data to the memory card and set up the next memory address.

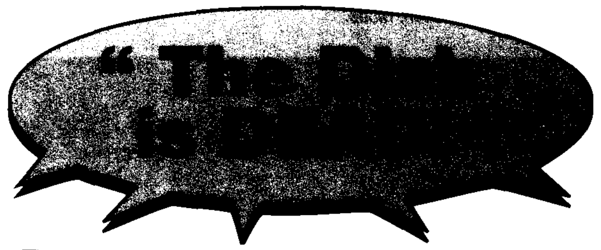
The ● INTl line is connected to the ● ACK pin on the external parallel port. An acknowledgement signal of at least 5 ms is used to indicate to the PC that the data has been read prior to **RET I .**

P_P LAY (Parallel Play Mode)

P_P **LAY** is the only routine that does not use interrupts. This routine simply checks for the end of RAM and exits if done. Otherwise, data is read from the memory card, written to the parallel output port, and the ● INTl line (● ACKJ is used as the strobe to signal the external parallel port that data is available. This time the external parallel port's ● STB line is the one that acknowledges the data. Once the *INTO line is seen strobing low, the ● INTl line is set [removing the data available signal). The next address is set up and the P_PLAY routine begins again.

FILLING AND EMPTYING THE BUCKET

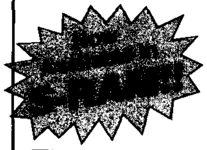
Transferring serial data at 9600 bps requires a fair amount of overhead.



The disk reliability problem is widespread.

There IS an Alternative!

Dust, moisture and vibration kill rotating disks every day. **Instant Access**™ **No Motion**™ solid state memories offer extremely high reliability, light speed access and hermetically-sealed data integrity. **Instant Access** is available in **FLASH EEPROM** and NVRAM and is programmable in DOS or WINDOWS. Rely on **Instant** Access-the demonstrably superior No **Motion Memory.**



APPLICATIONS

Mission Critical	**Mobile**	VISA
High Security	**Diskless PC**	MasterCharge
Banking	**Hazardous Environments**	AMEX
Military Communications		

PRODUCTIVITY ENHANCEMENT PRODUCTS, INC.

SEE US AT COMDEX!

26072 Merit Circle, Ste. 110 • Laguna Hills, CA 92653
1(800) 451-DISK • FAX (714) 348-1310

#155

---

4 Configurations Available

The Heart of the Matter...

RTXC™

Real-Time Multitasking Executive

• HITACHI 6303 • INTEL 80x88/x86, 80x96, 80x51
• MOTOROLA 680x0, 683xx, 68HC11, 68HC16
• INMOS T400, T800 • ZILOG Z80/Z180

- Preemtive Scheduling
- Fixed or Dynamic Priorities
- Timeout on some services
- Configurable and ROMable
- Intertask Communications
  - -Messages
  - -Queues
  - -Semaphores
- Memory Management
- Resource Manager
- Over 50 Executive Services Available
- System Level Debugging Utility
- System Generation Utility
- 450 Page User Manual

- Written in C
- Source Code Included
- No Royalties
- Technical Support
- Broad C Compiler Support
- Sensible License Agreement
- Most Popular C Compilers Supported

Ask about ASSIST™ Our new PC Development Package Combination

One Time License Fee From $995
Discounts for Multiple Licenses/Ports
The only real-time kernel you'll ever need™

A.T. BARRETT & ASSOCIATES, INC.
11501 Chimney Rock, Houston, TX 77035
FAX 713/728-1049
Phone 800/525-4302 or 713/728-9688

FREE DEMO AVAILABLE

#156

Two additional bits are sent in addition to the 8 bits of data: a start bit and a stop bit. Forgetting that this transmission is in reality 10 bits for each 8 bits of data is easy to do. At 9600 bps, that's 960 characters of data per second or 57,600 characters per minute. The bit bucket meets these ideal transfer rates by taking about 68 seconds to transfer a 64K file using a variety of methods. The simplest is using a DOS copy command, **COPY BITBUCK.TXT/B COM1/A**, to send serial data to the bit bucket. Procomm, or any similar communication package, can capture the bit bucket's serial output to a file.

A parallel data transfer will be much faster because data is moving 8 bits at a time. Although handshaking is employed, and timing depends on the length of the strobes used, a transfer rate of 4800 characters per second can be achieved with the standard DOS copy command. This time the file is sent via PRN instead of COM1.

Transferring the data from the bit bucket to the parallel ports requires two things. First, a bidirectional port (as discussed earlier) and second, a routine to make use of the port's ability to read back 8-bit data. I did not write an interrupt routine for my PC to make use of the bidirectional port; however, I did dedicate 5 minutes to writing a BASIC program to test the interface. Transfer was an excruciatingly slow 140 characters per second or about 8 minutes for the 64K bytes of data. That's **13** minutes total to get a program written and data transferred. I don't care how much you may hate BASIC, if that's the extent of the task, it was cost effective. Isn't that the bottom line?

FLEXING THE FIRMWARE

Now that you can see the simplicity of the hardware and firmware, you may wish to take this project to new heights. A routine to nondestructively test the amount of SRAM a card contained could be substituted for the predefined size selection. Maybe a two-digit hexadecimal display indicating the data presently being transferred. How about the ability to use other types of memory cards, such as EPROM, OTP, Flash, or EEPROM? Each has its own timing and power supply requirements.

Perhaps you'd like to randomly place and retrieve data. This aspect is a tough one for binary transfers unless your format uses a control word to define the block of data that follows. This method could cause problems, though, if sync is lost between the control word and the data block. I would prefer a format in which the lower nybble holds data and the upper nybble holds control information. This way, each byte would pass commands and data at the same time. Commands might indicate that the data in the lower nybble is part of a data byte or is part of an address to read or write.

The path you decide to take will depend on your specific application. Keep in mind, you will end up with a less universal device if you design a more complex control format. Universal is the key word here, no special transmission codes to interpret, no special file formats to handle, just dump your data out and let the bit bucket do the rest. Once recorded, the memory card can be write protected and stored or transported easily from site to site. I'm betting on the PCMCIA standard. As for its future—well, it's in the cards.

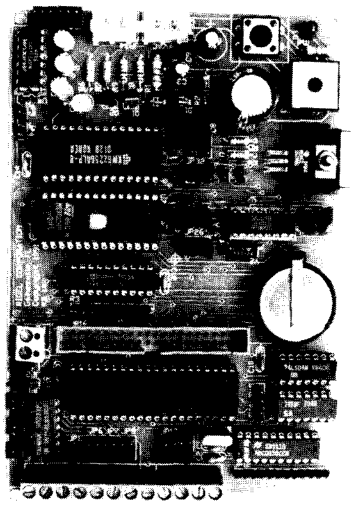
Jeff Bachiochi (pronounced BAH-key-AH-key) is an electrical engineer on the Computer Applications Journal's engineering staff. His background includes product design and manufacturing.

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.

422 Very Useful
423 Moderately Useful
424 Not Useful

8031 Embedded Controller / Development System



A Multipurpose Board for Embedded Systems or Software Development

The R-31 board may be used with the 8031/32 microcontroller or with Intel's 8052-Basic microcontroller. The R-31 becomes a complet development system when used with R-ware. R-ware consists of an on-board monitor program and integrated menu-driven PC-based host software for edit, assembly, PC-to-board communications, and debug functions.

Board includes power supply and regulator, up to 64K of memory using static RAM with batter back up or EPROM, an EPROM burner (used with the 8052 basic microcontroller), serial port up to 12 digital input/output ports, an 8 channe 8-bit analogue-to-digital converter, a battery-backed clock/calendar, 3 RS-485/422 line driver/receivers, and a comprehensive manual with circuit diagrams and example software

The R-31 is available with various options, with or without R-ware, unpopulated, as a kit, o assembled and tested. **Kit prices start at $195.**

Rigel Corporation,
P.O. Box 90040, Gainesville, FL, 32607 (904) 373-4629

#157

Reader Service #187

Twenty Years of Micros—Now What?

SILICON UPDATE

Tom Cantrell

The computer industry has come a long way in the last twenty years. One of the more esoteric technologies to be developed is fuzzy logic. Tom takes another look at this somewhat ticklish subject.

It was twenty years ago today, the gurus taught the chips to play, to paraphrase the Beatles.

Despite my tendency towards cynicism, I truly give humble thanks for having been associated with the micro business. The reason for my sentiment: the simple bottom-line. What other business continually offers more for less?

Consider some analogies comparing various products with the roughly 1000-fold improvement in microchip price and performance. How about a car that goes from 0 to 60 MPH in 1 second, a "four-pounder" burger for 2 cents, or a 20,000-square foot home for $1500?

The cost of "services" is even more of a joke. For example, today's "nonservice" gas station would be replaced with the real thing-a car wash, engine rebuild, and shoeshine while you wait! Need a lawyer or doctor? Just "buy" one for $100 a year!

I feel sorry for those people who chose other businesses. Year after year, they feed their customers apologetic mumbo-jumbo "justifying" the latest price increase (the best approach—whine that your customer is raising their price, too).

Imagine if I had to approach a customer with a pitch like, "Here's the new 8008x with 0.01 MIPS for only $10,000." Ouch!

No, the future looks bright for me and the micro.

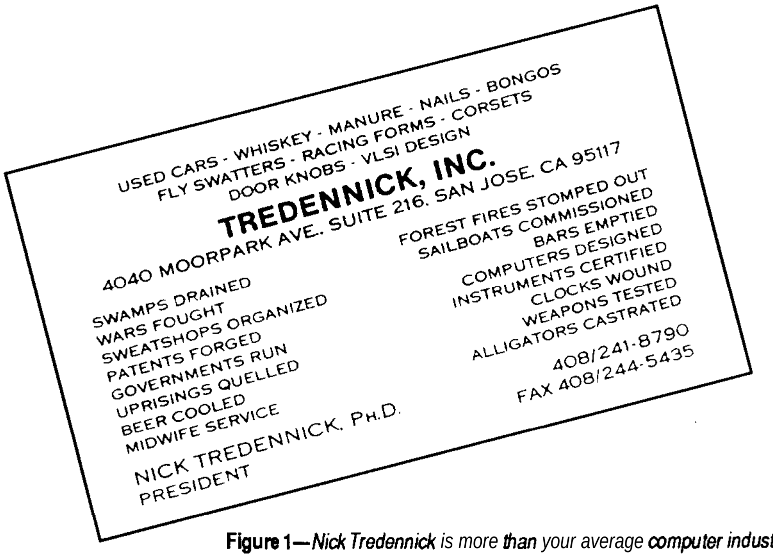
WHO'S ON FIRST?

Currently, the micro is caught in a custody battle. Who are its rightful parents?

At the recent Microprocessor Forum, the ever-entertaining Nick Tredennick hosted a 20th birthday "awards" ceremony. Nick is a well-known Silicon Valley gadfly, respected for his blunt (as you'll see) assessments of the local high-tech thrills and spills.

If you ever deal with Nick, you'll find he is quite an interesting fellow. First, check out his business card (Figure 1). Also, as his signature he signs the mirror image of his name. All in all, a real California kind of guy. Why is Nick this way? I don't know and, actually, I'm afraid to ask.

Anyway, the first part of Nick's presentation postulates that a "scientific" approach-much like the NFL quarterback rating system-should be used to rank the contenders for "Inventor Of The Microprocessor." The foil in Figure 2 pretty much summarizes his scheme.



USED CARS - WHISKEY - MANURE - NAILS - BONGOS
FLY SWATTERS - RACING FORMS - CORSETS
DOOR KNOBS - VLSI DESIGN

TREDENNICK, INC.

4040 MOORPARK AVE. SUITE 216. SAN JOSE. CA 95117

SWAMPS DRAINED
WARS FOUGHT
SWEATSHOPS ORGANIZED
PATENTS FORGED
GOVERNMENTS RUN
UPRISINGS QUELLED
BEER COOLED
MIDWIFE SERVICE

FOREST FIRES STOMPED OUT
SAILBOATS COMMISSIONED
BARS EMPTIED
COMPUTERS DESIGNED
INSTRUMENTS CERTIFIED
CLOCKS WOUND
WEAPONS TESTED
ALLIGATORS CASTRATED

408/241-8790
FAX 408/244-5435

NICK TREDENNICK, PH.D.
PRESIDENT

Figure 1—*Nick Tredennick is more than your average computer industry consultant.*

Going further (naturally), Nick highlighted some of the year's best headlines:

"U.S. SEEKS WORKSTATION LIMIT" *(San Jose Mercury)* "Pentagon officials, afraid the computers will be used to design weapons..."

Because the United States is the leading exporter of weapons, either this situation is a case of the government's right hand not knowing what its left hand is doing or a stealth-like version of industrial policy. Go figure.

"PLAN FOR *MIPS* CONFUSES EXPERTS" ***(New York Times).***

Is ACE a Joker? I guess it means things will work out for MIPS as long as they don't confuse beginners (or themselves).

"WINDOWS 3, USERS 0" (John Wharton, ***Microprocessor Report)***

Notable quotes of the year were remembered fondly:

*"We do lots of things with Apple. We spend billions of dollars on lawsuits, for example." (Bill Gates, speaking at Bay Area Mac Users Group [BMUG])

*"Their last original idea was to copy Intel." (Andy Grove, speaking about the lawsuit between Intel and AMD.)

*In the '80s, IBM progressed from the "A:" prompt to the "C:" prompt. (Jim Canavino at the Agenda '92 conference, paraphrased)

Next, the terrified audience (all fearful they might be singled out for an "award") was treated with announcements of:

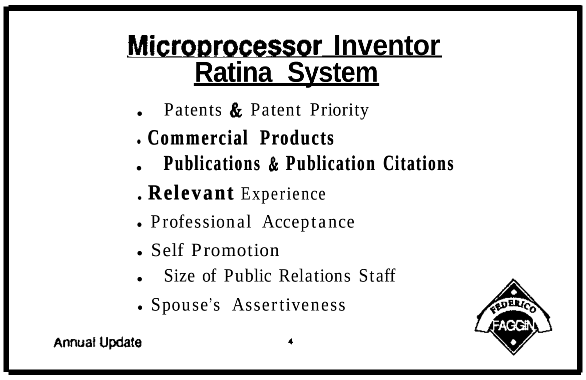
*Worst-kept Development Secret: Chips&Tech '386

*Time From First Rumor To Delivery: NexGen

*Best Self-induced Setback: Software Patents (Hey, come on Nick, it's not too late for law school)

Finally, he closed with the perennial favorite:

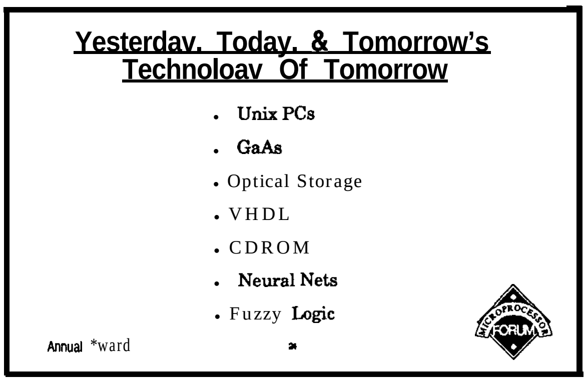
Figure 2—Nick Tredennick postulates that a "scientific" approach should be used to rank the contenders for 'inventor Of The Microprocessor."



Microprocessor Inventor Rating System

- Patents & Patent Priority
- **Commercial Products**
- **Publications & Publication Citations**
- **Relevant** Experience
- Professional Acceptance
- Self Promotion
- Size of Public Relations Staff
- Spouse's Assertiveness

Annual Update



Yesterday, Today, & Tomorrow's Technology Of Tomorrow

- **Unix PCs**
- **GaAs**
- Optical Storage
- VHDL
- CDROM
- **Neural Nets**
- Fuzzy **Logic**

Annual Award

Figure 3—The "Technology of Tomorrow" is a moving &get not often hit with much accuracy.

*Yesterday, Today, and Tomorrow's Technology of Tomorrow (Figure 3)

Thanks a lot Nick. Now I've got all my topics for '92 and don't have to write about 4-bit processors.

I'm glad there is a Nick Tredennick because he has single-handedly established a gigantic "irreverence umbrella" under which minor-league blasphemers can take shelter from risk-averse editors. I can safely "push the envelope" knowing that Nick has already shredded it, burned it, and spit out the fire.

I'M O.K., YOU'RE FUZZY

"Fuzzy logic" certainly exhibits some symptoms of a permanent Technology Of Tomorrow. Two years ago, I wrote about a fuzzy chip ***(Circuit Cellar INK,*** issue #15). Since then, little has happened, except that the company I wrote about apparently couldn't wait for the Revenues Of Tomorrow.

Nevertheless, in the Phoenix-like way of many high-tech outfits, a new company-Neuralogix-has risen from the ashes of the old data sheets. Is this persistence just another case of (as Nick might say) "you can't keep a bad idea down," or is there really something more to this fuzzy stuff after all?

Let me take a few paragraphs to explain fuzzy before I delve into NeuraLogix's latest offering; otherwise you could possibly dismiss it as just another variant of Artificial Intelligence [AI), which is widely known as "that which cannot be done by computer."

The simplest way to grasp fuzzy logic is to realize it is quite like the Boolean logic we know and love, except that the two values allowed in that scheme, True and False, are extended to encompass "multivalues," such as Could Be or You're Ridding.

That sounds good, but the next levels of detail, "membership functions," "singletons," "alpha cut," and so forth, are usually what causes eyes to glaze over. I'm going to skip all the theoretical stuff (which is well covered in the NeuraLogix literature and academia) and proceed directly to a real-world example.

Consider the ubiquitous PID controller. PID, which stands for Proportional, Integral, Derivative, is a closed-loop control scheme commonly used to set a "control output," depending on the state of an "error input." For example, in a disk drive with an analog head positioner [i.e., voice coil), the power applied to the positioner (control output) is determined by the error input (i.e., the distance between the current track and the track being sought).

PID refers to the three aspects of the "error" that should factor into the "output." They are as follows:

The *P* term says the output is simply a function of the current error. For instance, when using the disk example, if the track sought is far or close, apply more or less power to the head positioner.

The *I* term deals with the "accumulated error" (i.e., area under the error curve) over time. Imagine your Brand X drive has some contaminants fouling the positioner. When it gets stuck along the way from track A to B, the *I* term will continually increase the power until something gives.

The *D* term accommodates "differing rate of change" in the error. If the head is flying towards track 0 [perhaps the result of getting unstuck thanks to the *I* term), the *D* term will attempt to put on the brakes before "terminal overshoot" (i.e., hard drive seppuku) occurs.

PID algorithms are commonly implemented on micros. Indeed, the core of a PID loop usually looks something like

Output = P x (Error) + I x (Error Integral) + D x (Error Derivative)

where *P, I,* and *D* are the "gain" factors applied to the relevant Error measure.

Though straightforward, the software PID approach does have drawbacks. First, the calculation can be slow to execute, especially if floating point is called for. Fortunately, integer approximation tricks can often be used, but even then a typical micro may be limited to a few hundred loops per second. Second, determining the optimal gain factors is often nontrivial. Ideally, the controlled process is completely modeled to calculate the required gains; however, this operation is only feasible for those of you who

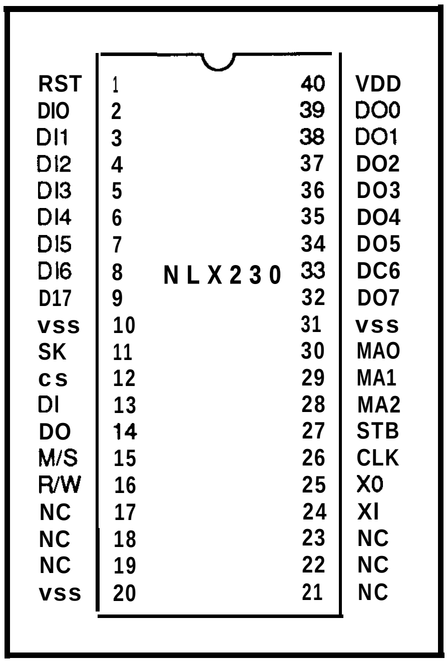


Figure 4—The *NeuraLogix NLX230 Fuzzy Microcontroller can be used in applications where traditional microprocessors lack the speed or memory to do the job.*

know "Laplace" isn't some kind of yuppy bar. We whose heads are less egg-like usually resort to various PID tuning procedures. Often, in the haste to get it working, intricate PID tuning schemes quickly devolve into "play around with the dials and see what happens."

To start understanding the fuzzy approach to a microapplication like PID. first consider the ultimate brute-force approach. Instead of using an algorithm, control is made using a "look-up table" (i.e., a predefined memory).

Let me see. Even using only S-bit integers, I need a memory with 24 bits of address (8 bits each for the *P, I,* and D terms) and an 8-bit output. The bad news is 16 megabytes of memory is over $500, even with falling memory prices. The good news about the look-up table approach is it is fast-the loop time is the same as the memory cycle time (i.e., millions of loops per second).

The procedure to initialize the memory with the proper output data for all **16** million possible input states is left as an exercise for the reader.

The promise and hope of fuzzy is to combine the concise representation of an algorithm with the speed and simplicity of a look-up table. To see how it works, take a look the NLX230 Fuzzy Microcontroller.

FISC

Imagine you had a computer with an instruction set, including gems like Branch Maybe and Output Approximately (should make for exciting debug sessions!). Fortunately, the NLX230 isn't such a beast, even though it is named a Microcontroller. The chip is much simpler than that. Indeed, it doesn't have an instruction set at all. Instead, the NLX230 samples inputs (eight channels of 8 bits each)

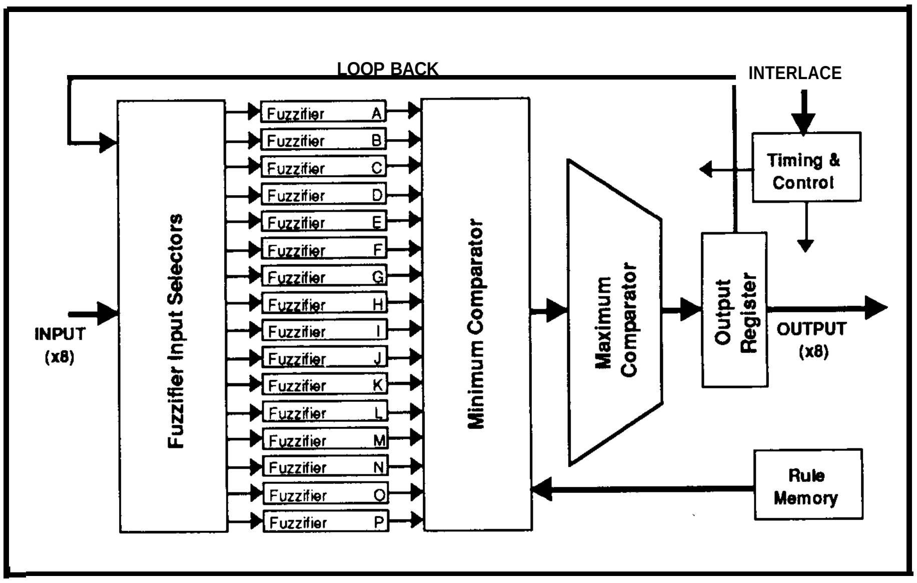


Figure 5—*Each of the 16 fuzzifiers on the NLX230 may be assigned to an input: either one of he eight input channels of a loopback from the output.*

and generates outputs (ditto) according to "rules" and "terms."

Interfacing the NLX230 is simple (Figure 4). The DIx and DOx pins are the process inputs and outputs, respectively. The ability to multiplex eight channels worth of data onto each of the DIx and DOx buses is provided with the three MA pins, which encode which of the eight channels are using the pins. The STB pin times the DIx and DOx transfers.

X1 and X2 are crystal connections while OSC is a TTL-compatible square-wave output of the on-chip clock generator. The NLX230 has been designed to run at up to 30 MHz, which translates to 30 million rules per second (RIPS?) in FuzzySpeak.

Before the chip can start fuzzing, the definition of the terms and rules used in decision making have to be loaded into on-chip registers. The DI, DO, SK, and CS implement a clocked serial interface compatible with commodity serial EEPROMs. At RST, the NLX230 will automatically access the external EEPROM, load the initialization info, and start processing, assuming the M/S pin is high (selecting master mode).

If M/S is low (selecting slave mode), after RST the NLX230 will wait for another device to clock the initialization data in using the aforementioned signals. In slave mode, a R/W input is also available, allowing two-way communication with the host.

Moving on-chip, Figure 5 shows the guts of the NLX230. Let me trace its operation, starting with an input and tracking it through to the output.

Each of the 16 "fuzzifiers" can be assigned to an input, either one of the eight input channels or a loopback from the output.

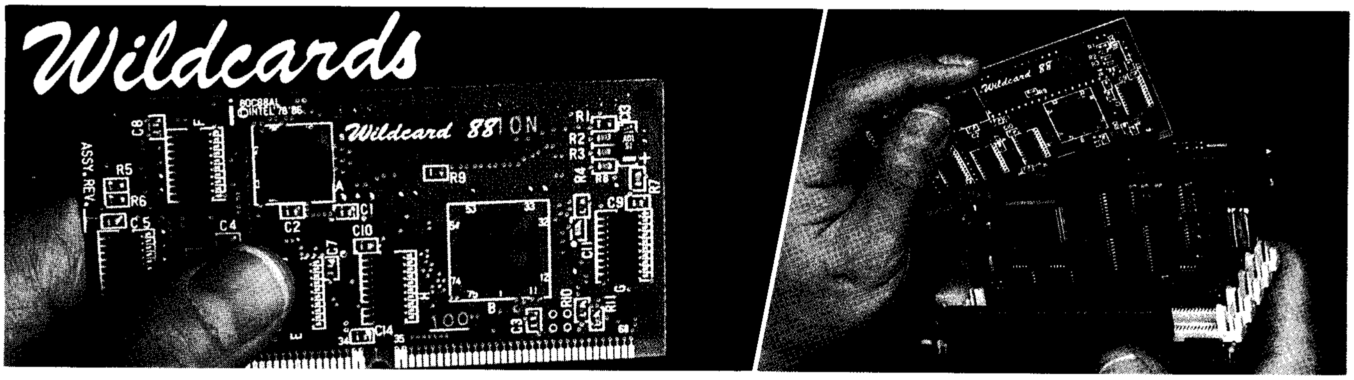
The purpose of a fuzzifier (Figure 6) is to classify the degree to which an input is a member of a fuzzy set. To this end, each fuzzifier has specified its center location and width. The center location (8 bits) corresponds to the input value, which is fully a member. The width value (5 bits) determines the range outside of which an input is no longer considered a member of the fuzzy set. So, the input and center location are passed through a subtracter that determines the "distance" between them (i.e., ABS[CENTER–INPUT]).

The Alpha Cut Calculator simply outputs the complement of the distance between the input and center location. Thus, if the input is a perfect match with the center location, 31 is output (remember, width is 5 bits), while if the input isn't in the fuzzy set at all (i.e., outside the width), a 0 is output. A type bit specifies whether the defined fuzzy set is considered inclusive or exclusive as shown in Figure 7 (width is 13 in this example).

The combination of a fuzzifier (including specified center and width) and its input is known as a term. Thus, with 16 fuzzifiers, the NLX230 handles 16 terms.

At this point, each fuzzifier associated with a given input will compute the membership value (0–31, nonmember to ideal member) in parallel.

2" x 4" EMBEDDED PC



Microcontroller. Microcomputer.

"Megatel Wildcards provide PC functionality in a flexible, small format."

Wildcard 88™
- CPU clock to 10 MHz
- Replaces full PC motherboard
- Co-processor and BIOS socket
- DMA, Bus, DRAM, Keyboard controllers

Multi/10
- On-board SCSI Host Adapter (supports up to 7 devices)
- Floppy Controller (1.44M, 1.2M)
- 2 RS-232, 1 Parallel, 1 RS-485 multi-protocol serial port

Vid/Mem:
- 640Kb User memory
- Video/Colour LCD controls CGA, Hercules®, IBM® Mono; (runs LCD Panels)

All Wildcards are low power single +5 volt operation.

125 Wendell Ave., Weston, Ont. M9N 3K9 Fax: (416) 245-6505

For information on our representatives please contact our head office at the number below.

(416) 245-3324

Wildcard 88 and Megatel are trademarks of Megatel Computer Corp. Hercules is a trademark of Hercules Corp. IBM is a trademark of IBM Corp.

megatel®

#158

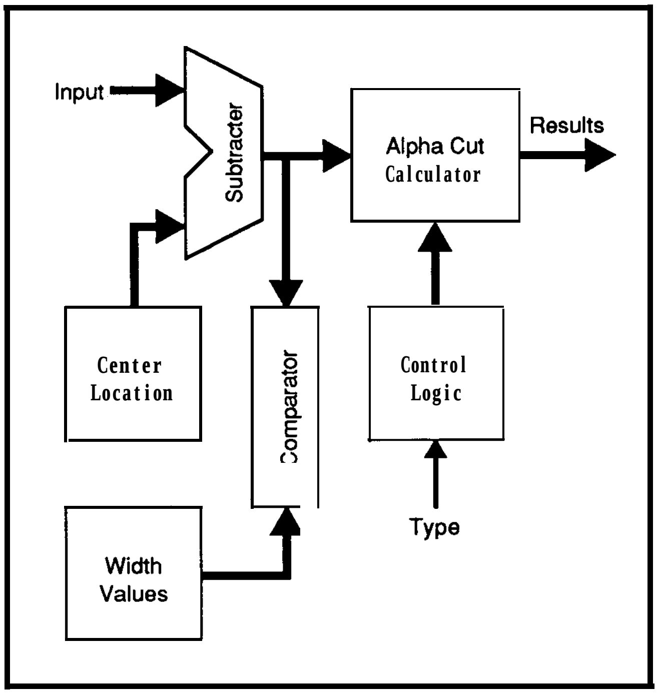


Figure 6—*The purpose of a fuzzifier is to classify the degree to which an input is a member of a fuzzy set.*

Next comes the minimum comparator that combines the membership info from the fuzzifiers with the rules stored in on-chip rule memory; the NLX230 can hold up to **64** rules. Fuzzifiers are assigned to inputs in the same way as rules are assigned to outputs (i.e., each output has between 1 and 64 rules associated with it exclusively. One rule can't apply to more than one output at a time).

A rule consists of 24 bits. Sixteen bits specify which of the fuzzifiers are applied to the rule (each bit specifies a fuzzifier). The remaining 8 bits comprise the "action value," the value that is output from the chip. Actually, besides direct output, an "accumulator mode" allows the output to be the sum of the action value and the previous output. So, each output's rules are applied against the the rule-specified fuzzifiers and their associated inputs.

Finally, the minimum and maximum comparators serve to select the rule whose action value drives the output. Effectively, this "maximize the minimum" operation performs a fuzzy logic sum-of-products (so maybe they should really call it a "Fuzzy PAL"?).

Intuitively, what I'm looking for are the inputs that "least fail" to be members of a set. For example, say I have a rule

```
IF tall AND wide AND heavy
  THEN go_on_a_diet
```

Now, assume there are two candidates:

1) kind-of-tall, kind-of-wide, kind-of-heavy
2) tall, wide, not-so-heavy

Under the NLX230 max/min scheme, candidate #1 is the one that should get the message. Of course, #2 may get it as well, unless there is another rule like

```
IF tall AND wide AND not-heavy
  THEN eat_before_you_die_fool
```

SUCK IT UP

Fortunately, instead of endless air computing, you can get your hands on some real hardware and try it yourself. NeuraLogix sells a very nice development system (Photo 1) that includes an NLX230-based PC plug-in board, download and simulation software, and excellent documentation. The whole package is only $395—rather a bargain in my opinion.

So far, I've discussed the NLX230 at a machine-language level [i.e., input selectors, fuzzifier numbers, membership center and width, etc.). A key portion of the software provided is a much easier to use "HLL" (maybe they should call it "C~~") for the NLX230.

An example of an NLX230 "program" is one to drive a fuzzy vacuum cleaner. The goal is for the vacuum to automatically adjust to varying operating conditions.

In particular, the vacuum design monitors three inputs: Pressure, Dirt, and Texture, and drives five outputs: Vacuum, Height, Speed, Clean, and Bag.

The program (Listing 1) starts by defining 15 of the 16 terms or fuzzifiers. The format specifies the input [e.g., **Pressure),** followed by a classification name [e.g., Pressure Very Low or PVLow) associated with a center and a width value (e.g., 0, 30), and finally, whether an INclusive or an Exclusive membership should be used. Notice how the latter feature allows easy definition of mutually exclusive sets. For example, **PVLow is** defined as **Pressure** with a center of 0 and a width of 30 inclusive, while NotPVLow is defined as **Pressure** with a center of 0 and a width of 30 exclusive.

Next, each of the rules associated with the five outputs is defined. Note

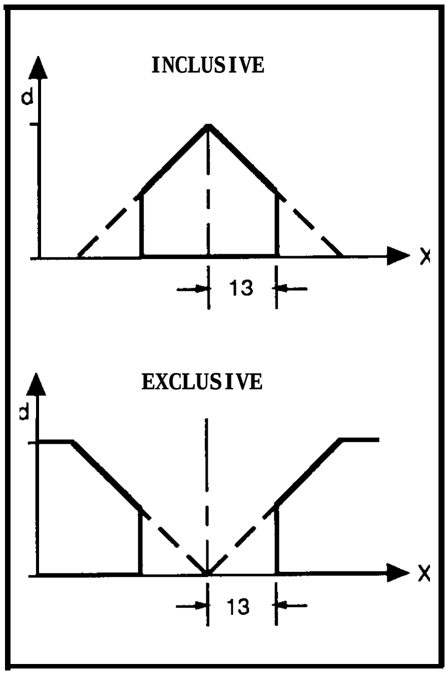


Figure 7-A type *bit* going into *the* Alpha *Cut* Calculator *specifies whether he defined fuzzy set is* considered inclusive or exclusive.

that each output can have a different number of rules; in this example a total of 25 of the 64 rule memories are used. The format of a rule is

IF **input** IS class [AND **input** IS class] **THEN** **output**

The first line of each output definition names the output (e.g., **Vacuurn)**, specifies an initial value (e.g., 50), and specifies either Accumulate or I Mmediate output mode for the action value.

Notice how terms and outputs can freely allocate fuzzifiers and rules within the total limits of the chip (sixteen terms, sixty-four rules]. The allocation corresponds to the "resolution" of the input/output. In this example, Pressure and Dirt inputs are classified with twice the resolution of texture (six vs. three fuzzifiers). On the output side, beater height can be set to eight levels while the bag-full light is only on or off (eight vs. two rules).

The NLX230 kit allows you to simulate operation, download to the chip for actual execution, or both. As for I/O, 8-channel 8-bit analog-to-digital and digital-to-analog conversion, in addition to digital I/O, are provided.

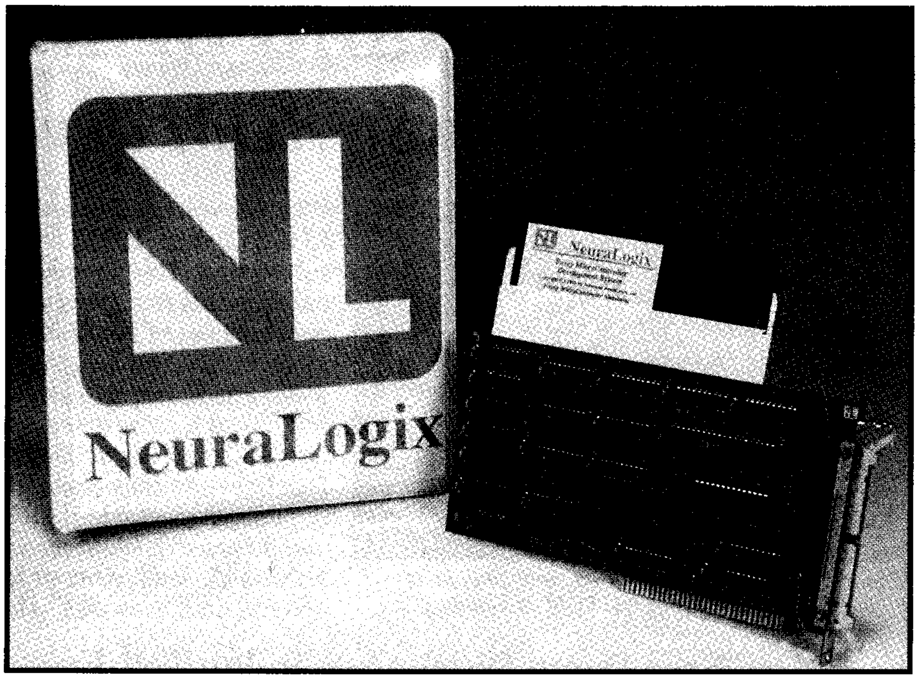


Photo 1—*NeuraLogix makes a complete PC-based development system for the NLX230.*

LET'S GET FRIENDLY

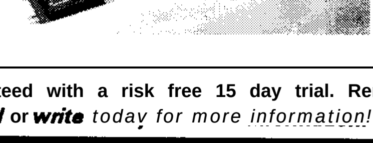
Before fuzzy logic can take off, engineers have to design fuzzy products. At least in Japan, from what I understand, fuzzy vacuum cleaners,

Z80, Z180, HD64180, HD647180
In-Circuit Emulators

Affordable **state-of-the-art in-circuit emulation for Z80 family processors. Increase your productivity with features like:**

- **Symbolic and C Source Level Debugging**
- **Pull-Down Menus and Command Line Interface**
- **Complete Context-Sensitive Help**
- **Real Time Emulation up to 12 MHz**
- **Serial Interface works on any PC, even on laptops.**

Z80
2180
Z84C13
Z84C15
HD64180
HD647180
TMP84C015

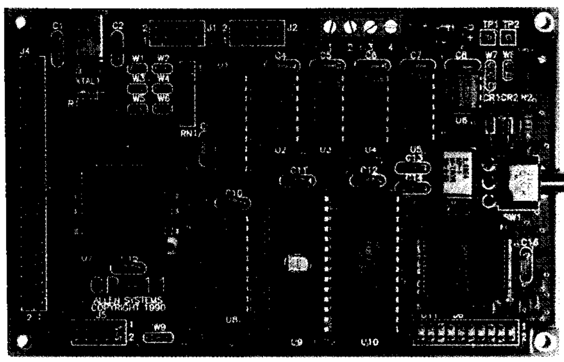


Satisfaction guaranteed with a risk free 15 day trial. Rentals also available. *Call* or *write* *today for more information!*

Control Resources
19042 San Jose Unit S, P.O. Box 8694
Rowland Heights, CA, 91748
818/912-5722 (Voice) 818/854-1439 (Fax)

#159

8051, 8096, 68HC11, 68332
SINGLE BOARD COMPUTERS



We **feature a series of** single board computers for process control applications. Each is available as a bare printed circuit board, or fully assembled and tested. Optional development software is also available. Please contact us to discuss your requirements and receive a literature package covering technical specs and pricing.

ALLEN SYSTEMS
2346 Brandon Road • Columbus, OH 43221
(614) 488-7122

Listing 1—*Putting a microprocessor in a vacuum* **cleaner** *may seem extreme, but is a good example of how fuzzy logic can be used to simplify programming.*

```
NLX230          /* Vacuum Cleaner Example */
TERMS
Pressure    IS  PVLow   0   30  IN /*  Vacuum Pressure     */
Pressure    IS  PLOW    40  20  IN /*  Vacuum Pressure     */
Pressure    IS  PMed    70  20  IN /*  Vacuum Pressure     */
Pressure    IS  PHigh   90  20  IN /*  Vacuum Pressure     */
Pressure    IS  PVHigh 135  30  IN /*  Vacuum Pressure     */
Pressure    IS  NotPVLow 0  30  EX /*  Vacuum Pressure     */
Dirt        IS  DLow    0   30  IN /*  Quantity of Dirt    */
Dirt        IS  DMLow   40  20  IN /*  Quantity of Dirt    */
Dirt        IS  DMed    70  20  IN /*  Quantity of Dirt    */
Dirt        IS  DMHigh 100  20  IN /*  Quantity of Dirt    */
Dirt        IS  DHigh  127  20  IN /*  Ouantity of Dirt    */
Dirt        IS  NotDLow 0   30  EX /*  Ouantity of Dirt    */
Texture     IS  TSmooth 0   25  IN /*  Texture of Floor    */
Texture     IS  TMed    30  20  IN /*  Texture of Floor    */
Texture     IS  TRough  60  30  IN /*  Texture of Floor    */

OUTPUTS
Vacuum 50 AC /* Vacuum Control        */
{
 IF Pressure IS PLow AND Dirt IS DHigh AND Texture IS TRough
     THEN 50
 IF Pressure IS PLow AND Dirt IS DMHigh AND Texture IS TRough
     THEN 30
 IF Pressure IS PHigh AND Dirt IS DMLow AND Texture IS TSmooth
     THEN 10
 IF Pressure IS PHigh AND Dirt IS DLow AND Texture IS TSmooth
     THEN 40
 IF Pressure IS PVHigh AND Dirt IS DLow THEN -50
}

Height  50  AC  /*  Beater Brush Height  */
{
 IF Dirt IS DHigh AND Texture IS TSmooth THEN 40
 IF Dirt IS DMHigh AND Texture IS TSmooth THEN 30
 IF Dirt IS DMHigh AND Texture IS TMed THEN 10
 IF Dirt IS DMLow AND Texture IS TMed THEN -30
 IF Dirt IS DLow AND Texture IS TSmooth THEN -50
 IF Dirt IS DMHigh AND Texture IS TRough THEN 20
 IF Dirt IS DHigh AND Texture IS TRough THEN 50
 IF Dirt IS DMLow AND Texture IS TSmooth THEN -10
}

Speed   50  AC  /*  Beater Brush Speed    */
{
 IF Dirt IS DLow AND Texture IS TSmooth THEN -50
 IF Dirt IS DLow AND Texture IS TMed THEN -10
 IF Dirt IS DMHigh AND Texture IS TMed THEN 10
 IF Dirt IS DHigh AND Texture IS TMed THEN 20
 IF Dirt IS DMHigh AND Texture IS TRough THEN 30
 IF Dirt IS DHigh AND Texture IS TRough THEN 50
}

Clean   50  AC  /*  Cleanness Indicator  */
{
 IF Dirt IS DLow THEN -50
 IF Dirt IS DMLow THEN -10
 IF Dirt IS DMHigh THEN 10
 IF Dirt IS DHigh THEN 50
}
Bag     0   IM  /*  Bag Change Indicator  */
{
 IF Pressure IS PVLow AND Dirt IS NotDLow THEN 100
 IF Pressure IS NotPVLow THEN 0
}
```

fuzzy rice cookers, and so on are emerging.

I think the slow acceptance of fuzzy logic is inherent in the oxymoron-like nature of the term. LOGICal people (i.e., engineers) often don't design warm and FUZZY (i.e., friendly) products.

My coffee maker can automatically turn on at any time with crystal-controlled accuracy. But heaven forbid if its pot was left in the dishwasher overnight. No, it doesn't take a fuzzy chip to make a coffee maker that won't merrily drench the counter. It does take a fuzzy mindset on the part of the designer, though.

Same for the VCRs that take a Ph.D. to program. The only one I've ever been able to conquer is about 10 years old and features blessedly few buttons. I think "few buttons" is a key characteristic of a fuzzy design.

Hopefully, fuzzy chips like the NLX230 can speed the availability of friendly products. Until then, I suggest you keep a close eye on your appliances-or else!

CONTACT

American NeuraLogix, Inc.
411 Central Park Drive
Sanford, FL 32771
(407) 322-5608
Fax: (407) 322-5609

Microprocessor Report
& Microprocessor Forum
MicroDesign Resources, Inc.
874 Gravenstein Hwy. So., Suite 14
Sebastopol, CA 95472
(707) 823-4004
Fax: (707) 823-0504

Tom Cantrell holds a B.S. and an M.B.A. from UCLA. He owns and operates Microfuture Inc., and has been in Silicon Valley for ten years working on chip, board, and system design and marketing.

IRS

425 Very Useful
426 Moderately Useful
427 Not Useful

Writing Code to Support Nonvolatile Memory

PRACTICAL ALGORITHMS

John Dybowski

Designing a system to survive an unexpected power loss may seem as easy as adding a battery backup to the RAM. However, making a truly bullet-proof system often requires more safeguards.

Your system has been running flawlessly for weeks. Now you've made the final code tweaks and you know you're going to blow the competition out of the water. At your presentation, everyone waits expectantly as you plug in the new production PROM and power up the system. Quickly, you realize that something is wrong. The system is now making funny noises and scrolling strange characters across the display. Sheepishly, you reach over and reset the system. Same result. What could be worse? A call comes in from the computer room, the entire network is down, choked by a rash of gibberish from your demo unit. Your finely tuned, well-crafted creation resembles some cheap, dime store novelty. Beads of sweat form on your forehead as you search your mind for an explanation.

NONVOLATILE RAM

Used to be, that embedded systems were simple-minded machines. They powered up and performed their tasks until power was removed. Then, reminiscent of some government officials.. . they didn't remember. We quickly found that for many applications, maintaining the contents of RAM memory in the absence of power could add features and open up new applications. Using the various backup power supply options available, many people cobbled up backup protection schemes that worked reasonably well. More recently, commercial products have been released that specifically address the issues of RAM nonvolatility. Close examination of these RAM nonvolatizing capabilities indicates many offer features beyond what can be attained reasonably using a discreet approach. Reasonably, that is, in regard to component count, board space, and the need for manual tweaking. The arrival of these reliable, integrated RAM protection circuits may lull many people into the belief that the valuable contents of their RAMs are secure. Well folks, I hate to rain on the parade, but it ain't necessarily so!

Unfortunately, a whole new set of software considerations exists in a system that relies on nonvolatile RAM for its continued successful operation. Knowledge of the varying levels of security attainable, and their pitfalls, are important in the design stages of a project. All too often, designers discover the weaknesses in their systems late in the design cycle when there is no recourse other than applying patches to correct the deficiencies. This solution is usually difficult because at times problems lay deep within the bowels of our code.

RAM integrity is compromised in different ways. An operation may not run to completion, perhaps only partially updating some vital control block. The system can suffer a static hit or power surge, possibly modifying some RAM bits. A hit of sufficient magnitude may even disrupt the normal sequential operation of the processor itself, resulting in its attempt to execute data or to resume at some arbitrary point of the program. Should this irregularity happen to be in the RAM_CLEAR routine, all bets are off. Obviously, the most desirable response to such system anomalies is for some recovery mechanism to put things straight and for normal operation to resume. At the very least, you want the system to detect the error and either continue to run in a degraded mode or to remove itself from operation and possibly issue a distress call to the host computer.

THE WARM AND COLD OF IT

The first thing you want to determine early in the power up sequence is whether the system is performing a cold or a warm start. On

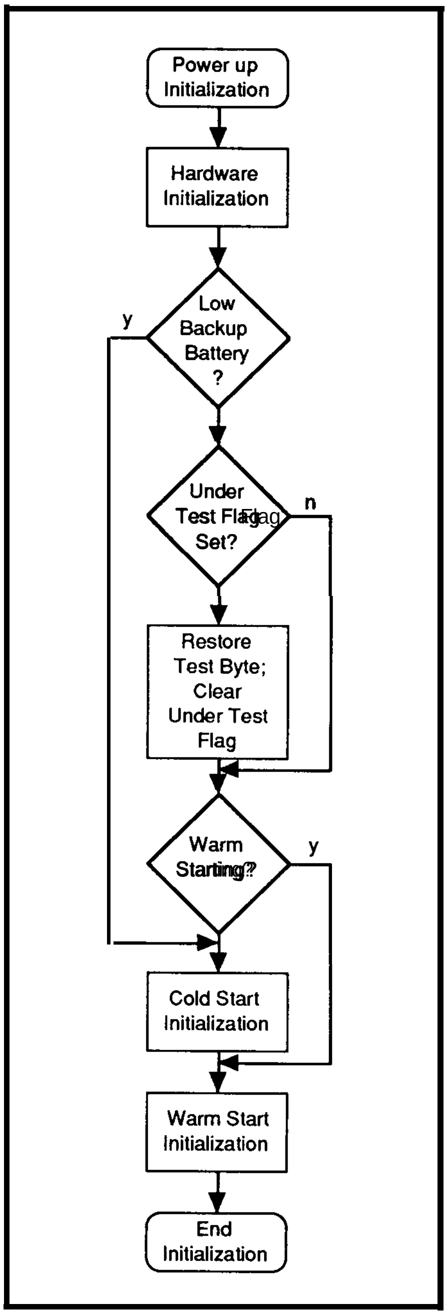


Figure 1-A *typical power-up sequence includes* **checking** *the* **status of the backup** *battery and* different *initialization* ***blocks for*** *cold and* ***warm starts.***

a cold start anything goes. You can perform your favorite system diagnostics, go through the rigors of a thorough destructive RAM test, and initialize the system variables to a default state before indicating the system is warm. This denotation may be accomplished by writing a unique string to RAM. In some cases, it may not be performed until receiving the required configuration information via download from a host computer or from some other source.

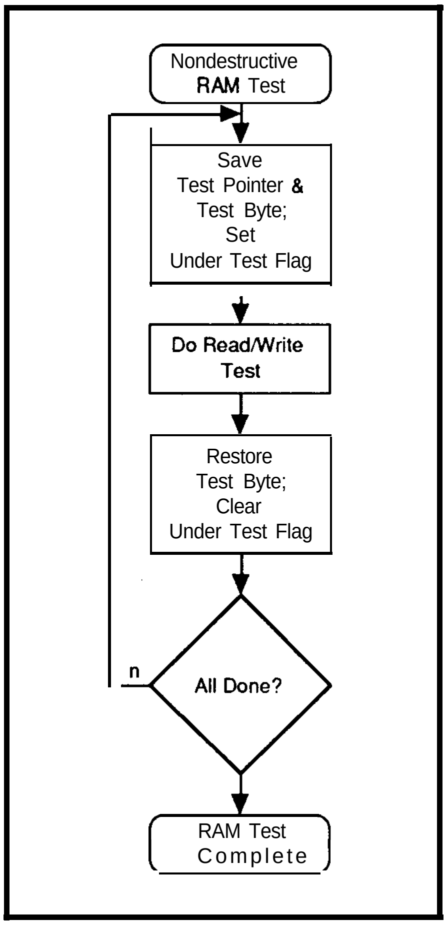
A warm start is more restrictive. Most likely you will only want to do a cursory RAM test and verify the contents of some critical control blocks. Of course, it's a good idea to check the state of the backup power at this time. Many nonvolatile RAM controllers provide a mechanism that allows easy checking of the backup battery's state. The popular DS1210 indicates a low battery condition by suppressing the second access to the RAM following power up. What you actually do when you determine the backup battery is low depends to a great extent on the type of backup power source you're using. If it's a rechargeable unit, then making the low battery test a part of the cold / warm determination logic is reasonable. Of course, if the backup source is not rechargeable, then you're dead in the water and your options are limited. Figure 1 summarizes what's involved in a typical power-up sequence.

Make sure you don't overlook the recovery time parameter when working with the integrated nonvolatile controllers. The recovery time, implemented in most controller circuits, is the "dead time" during which the controller inhibits RAM access following power up. This pause allows the system to come under control before access to the RAM chip is permitted. For example, the DS1210's average recovery time is specified as 80 ms and can span from a minimum of 2 ms to a maximum of 125 ms, which is quite a range, and believe me, you don't want to be marginal on this one.

THE NONDESTRUCTIVE, NONDESTRUCTIVE RAM TEST

In many cases, once a system is put into service and initialized, it may run warm for the remainder of its operating life. Periodically performing system diagnostics is still highly desirable, but under such circumstances it may not be possible to bring the system down to do so. Therefore, diagnostic routines must be devised to operate unobtrusively and nondestructively. The nondestructive RAM test, having the right to affect every byte in RAM, is especially critical. The general premise behind this test is to read the contents of a RAM location, save the contents in a register, perform some write and read verifications on the RAM location, and restore the original data back to its rightful place. The actual bit patterns used in the RAM test usually have some special significance to the programmer. (Sometimes this preference borders close to a religious fervor, so I won't get in the middle of this one, thanks.)

Of course, the big mistake is to believe the RAM test will run to completion! Usually included as part of the power up sequence, the RAM test is particularly vulnerable to power fluctuation problems. Consider a system dropping in and out of brownout. In this case, the RAM test usually provides a pretty good window on doing some damage. Thrash a table of jump vectors or a pointer block and things rapidly go down hill from there. For a RAM test to be truly nondes-



Nondestructive RAM Test
Save Test Pointer & Test Byte; Set Under Test Flag
Do Read/Write Test
Restore Test Byte; Clear Under Test Flag
n
All Done?
RAM Test Complete

Figure *2-A* ***truly nondestructive*** *RAM test must account for he* ***possibility*** **of** *a power loss in the middle of* ***the*** *test.*

tructive, enough information must be maintained to allow restoring the right data to the right location in the event the test is interrupted before completion (see Figure 2). This information may reside in a fundamental RAM control block and consists of the address of the location under test, the value stored at that location, and an under test flag indicating the location is indeterminate. For obvious reasons, you will want to exclude this block from the area subject to the RAM test.

The RAM recovery code should be exercised early in the power-up sequence by interrogating the under test flag. Should this flag be set, pick up the pointer, get the data, and put it back! You want to do this reset right after the obligatory hardware initialization, before the cold or warm determination. After all, you may have whacked your warm RAM indicator! If the start happens to be a cold one after all, this extra step obviously causes no problems.

THE DIVISIBLE DATA STRUCTURE

Another problem could occur when updating fundamental data structures. Should the processor die in the midst of updating a double precision count or, say, a 16-bit pointer, the result could be meaningless. The problem is most aggravated when using B-bit processors because many of the elements, such as counters and pointers, require multiple instruction steps in their manipulation. Admittedly, the window for such an error is small. This possibility becomes a real concern over a large installed base, especially in the hostile environment embedded systems seem to find themselves in and particularly when doing a lot of data manipulations. A problem call from the field indicating a controller once again exhibited some strange phenomenon has an unsettling effect on the current "hot project." Of course, they never mention the guy doing the installation lashed the thing to a 440-volt compressor!

Say you have to keep a multiprecision running tally that is critical in nature. One way to deal with this issue is implementing dual counters that are updated in tandem. You can provide an effective way of trapping an errant count by maintaining checksums for each counter. The addition of a flag to indicate the current primary counter completes the scheme (see Figure 3).

This arrangement uses the following process to update a count. Calculate the checksum over the counter denoted as primary (in accordance with the current counter flag). If the checksum verifies, the count is OK, so read the counter. (If the checksum verification fails, you can fall back to the secondary counter and try the same procedure.) Do as you will to the count and calculate a fresh checksum. Then store the count and checksum to the secondary counter and mark the secondary counter as primary. For good measure, copy the new count and

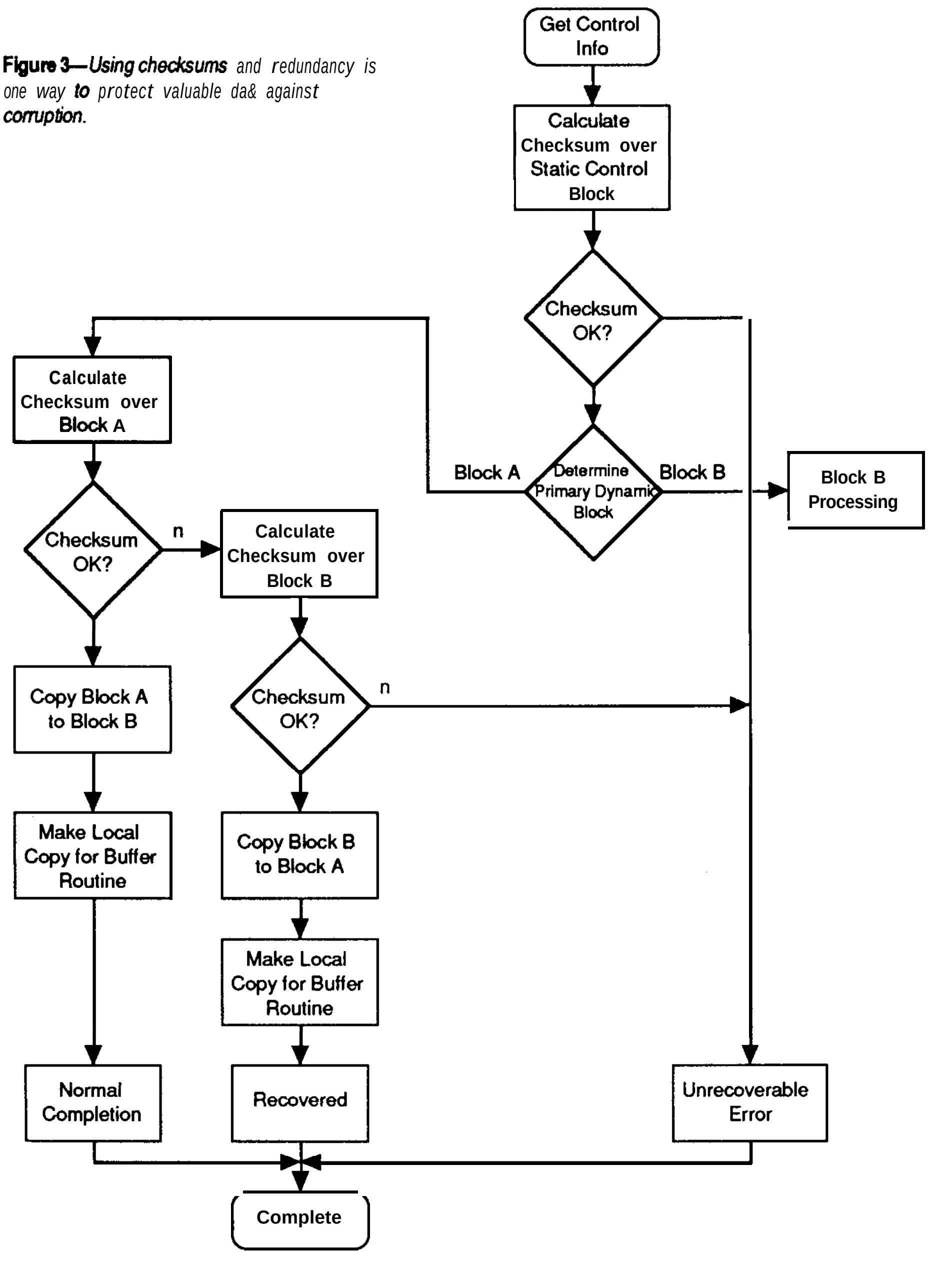

Get Control Info
Calculate Checksum over Static Control Block
Checksum OK?
Determine Primary Dynamic Block
Block A
Block B
Block B Processing
Calculate Checksum over Block A
Checksum OK?
n
Calculate Checksum over Block B
Checksum OK?
n
Copy Block A to Block B
Make Local Copy for Buffer Routine
Copy Block B to Block A
Make Local Copy for Buffer Routine
Normal Completion
Recovered
Unrecoverable Error
Complete

Figure 3—*Using checksums and redundancy is one way to protect valuable da& against corruption.*

checksum into what is now the secondary counter in order to have a valid backup copy.

SECURE RING BUFFERS

Ring buffers are frequently employed in embedded designs. Often these are quite large and may hold collected data that is uploaded to the host computer either on-line or on demand. As such, errant operation may not be restricted to the faulty system itself, and ultimately may adversely affect the network communications and host computer. In a multidropped environment, the problem could affect all the attached controllers as well by disrupting normal communications. At this point, I will describe the potential problem areas and how to safeguard them.

A ring buffer is implemented by allocating a linear region of RAM as a storage area. The bounds of the storage area are defined by the start and end pointers. Each time a write or read pointer is incremented, it is checked against the end pointer. If it has just incremented past the end pointer, it is set equal to the start pointer. The buffer functions in a ring or circular fashion using these means.

The ring buffer is considered empty when both the read and write pointers are equal. For all intents and purposes, the buffer is full when the area between the write pointer and the read pointer is insufficient to hold the data you wish to store. The success or failure of the requested operation is communicated back to the calling routine via a return code so the caller can take the appropriate action. The fundamental operations the ring buffer must perform are write record, read record, and remove record. Additionally, an initialization function allows the buffer to be allocated dynamically at run time. A check function is useful to return the empty or full [error) status of the buffer without actually doing a read or write operation.

The ring buffer as described requires two control blocks in order to operate. First, a static block, written once at initialization time, that defines that start point and end point of the buffer area, the total amount of bytes available for storage, and the largest record allowed on a read or write operation. Second, a dynamic block that contains the read and write pointers and the bytes-used counter, which is modified on each operation that writes to or removes data from the buffer.

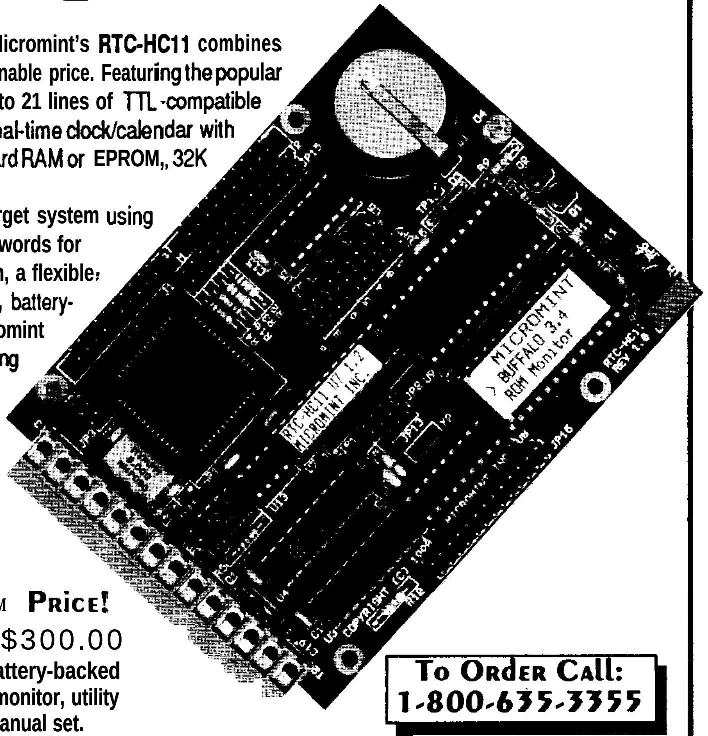
You can obtain a certain degree of security simply through the order of actions you perform when affecting the ring buffer. For the write operation, proceed by making a copy of the write pointer and free space counter, check if the record will fit, and then move and delimit the record. The new values are written into the dynamic control block once the new value of the write pointer has been determined and the free space counter has been recalculated.

Monkey with the pointers and other values in the actual control blocks as little as possible in order to minimize the amount of time the block is in an unknown state. Therefore, all intermediate operations should be performed on copies of these

RTC-HC11 **Processor Board**

Offering exceptional value in a single-board embedded controller, Micromint's RTC-HC11 combines all of the most-asked-for features into a small 3.5' x 4.5' package at a reasonable price. Featuring the popular Motorola MC68HC11 8-bit microcontroller, the RTC-HC11 gives you up to 21 lines of TTL-compatible I/O; an &bit, B-channel analog-to-digital converter; two serii ports; a real-time clock/calendar with battery backup; 512 bytes of nonvolatile EEPROM; and up to 64K of on-board RAM or EPROM,, 32K of which can be battery backed.

Software development can be done directly on the RTC-HC11 target system using BASIC-1 1, an extremely fast integer BASIC interpreter with dedicated keywords for I/O port, A/D converter, timer, interrupt, and EEPROM support. In addition, a flexible configuration system allows a BASIC program to be saved in the on-board, battery-backed static RAM, and then automatically executed on power-up. Micromint afsooffersseveral hardware and software options for the RTC-HC11 including the full line of RTC-series expansion boards as well as an assembler, ROM monitor, and C language cross-compiler.



Additional features include:
- **Asynchronous serial port with full-duplex RS-232 and halfduplex RS-435 drivers**
- **1 -MHz synchronous serial port**
- **CPU watchdog security**
- **Low-power 'sleep" mode**
- **5-volt-only operation**
- **RTC stacking bus**

Special Development System Price!

RTC-HC1 1 -DEV $300.00

Board w/ B-bit ADC, EEPROM, 32K battery-backed RAM, Clock/Calendar, BASIC-1 1, ROM monitor, utility diskette with assembler and complete manual set. Other configurations also available starting at $239.

To Order Call: 1-800-635-3355

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

#162

elements. Also, setting a limit on the maximum size for records that can be written to and read from the ring buffer is helpful in case the storage region somehow gets corrupted and the end of record marks get scrambled. You don't want to move a huge block of data on a read that could overflow the destination buffer, destroying adjacent areas. This method of handling the ring buffer uses some common sense techniques to reduce the chance for errors, but further steps can be taken to maximize the ring buffer's integrity.

As in the simpler data structures, you can use dual redundant blocks with checksum verification and a primary/secondary flag to determine whether a block is intact and contains valid information. (The implementation details are basically the same as for the counter example.) Now you can incorporate some error traps whereby problem status can be communicated to the calling routines by adding more return codes to the basic buffer functions.

DATA PREPPING AND ROUTING

At this point, you have reduced the likelihood of undetected errors and provided some redundancy, for fault tolerance, in the basic structure of the ring buffer. Now I will show you how to structure the code that will actually deposit and extract the data records. Consider for my example a hypothetical system that performs some measurements, writes the measurement data to the ring buffer, and transmits data from the ring buffer to the host computer on a continual basis.

The question now is when the data record actually becomes nonvolatile. Consider that the controller has performed its measurement and now has critical data it wishes to dispatch to the ring buffer. At this time, the data is in a work area and if power to the system is interrupted the data will be lost. Worse, the possibility of insufficient space remaining in the ring buffer to accept this data record exists, and I am left with the dilemma of how to proceed.

One way to handle this situation is to store the data record in an intermediate nonvolatile buffer as soon as it becomes available. A pending buffer write is then signalled by the setting of a flag (write pending), which also resides in a nonvolatile area in RAM. The write pending flag can also be interpreted as indicating a busy condition by the data collection code, which would inhibit taking more measurements until the condition clears. Now the ring buffer management code can be centralized in the mainline loop (or perhaps run as a task) that polls the write pending flag and, if set, attempts the write operation.

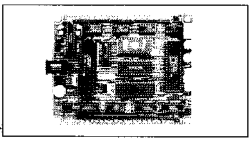
After completing a successful write, the flag is cleared and normal operation resumes. The simplicity of this approach handles all write operations in a clean and consistent manner. No decision making is required as to whether I am recovering from a power interruption, in a buffer full condition, or just doing normal a write operation. Actually, a small

803 1 PROGRAMMABLE CONTROLLER BOARDS

Common features: • 8031AH CPU, • 11.0592 MHz clock speed, User interface is via a terminal emulator program running on a host computer and linked through the RS232 Serial Port on the controller board.

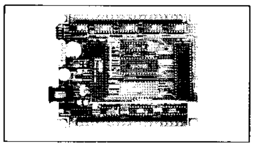
- 32 K FLASH MEMORY {Eliminates the need for Programmer and UV Eraser]
- 16KEPROMand8KRAM
- 32 Buffered I/O lines

Assembled and Tested board $155


DG32F Board (4.80"x6.00")

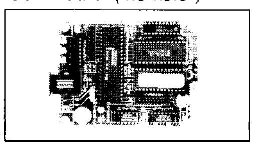
- 16 K EPROM and 8 K RAM
- 32 Buffered I/O lines

Assembled and Tested board $129


DG31 Board (4.5"x5.5")

- On board 8 or 16 K EPROM, 8 or 32 K RAM capacity
- Ports 1 and 3 processor lines on 20 pin headers, I/O expansion lines on 20 p. header

Assembled and Tested board $80

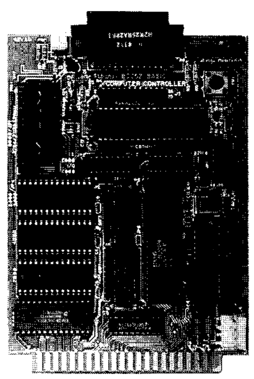

EMC32 Board (3.00"x4.00")

KITS, Power supplies, Support **and Application Software available.**

L.S. ELECTRONIC SYSTEMS DESIGN
2280 Camilla Rd. Mississauga, Ontario, Canada
L5A 2J8 Phone/Fax: (416) 277-4893

Terms: Shipping US/Canada $6 Check or Money Order, Please

BCC52
BASIC -52
Computer/Controller



The **BCC52 Computer/Controller** is Micromint's hottest selling standalone single-board microcomputer. Its cost-effective architecture needs only a power supply and terminal to become a complete development or end-use system, programmable in BASIC or machine language. The **BCC52** uses Micromint's 80C52-BASIC CMOS microprocessor which contains a ROM-resident 8K- byte floating-point BASIC-52 interpreter.

The **BCC52** contains sockets for up to 48K bytes of RAM/EPROM, an "intelligent" 2764/128 EPROM programmer, three parallel ports, a serial terminal port with auto baud rate selection, a serial printer port, and is bus-compatible with the full line of BCC-bus expansion boards. **BASIC-52's** full fbating-point BASK: is fast ard efficient enough for the most complicated tasks, while its cost-effective design allows it to be considered for many new areas of implementation. It can be used both for development and end-use applications.

Processor
- 80C52-BASIC, 8-bit CMOS microcomputer
- jumper-selectable conversion to 80C31/80C32 functionality
- 8K bytes ROM (MI BASIC interpreter)
- 256 bytes RAM
- three 16-bit counter/timers
- 32 I/O lines
- 11 MHz system dock
- 6 interrupts

Input/Output
- console I/O RS-232 serial pat
- line pints RS-232 serial pat
- three 8-bit programmable TTL-compatible parallel I/O ports using a 8255 PPI
- alternate console RS-422/RS-485

Memory
- expandable to 62K bytes
- five on-board sockets
- up to four 6264 (8Kx8) static RAM
- either an 8K 2764 or 16K 27128 EPROM

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

		Single Qty.	100 Qty.
BCC52	BASIC-52 Controller Board with 8K RAM	$189.00	5149.00
BCC52C	Lower-power all-CMOS version of the BCC52	5199.00	$159.00
BCC52I	Full industrial temperature range	5294.00	5220.00
BCC52CX	CMOS, Expanded BCC52 w/32K RAM	5259.00	$159.00

MICROMINT, INC. 4 Park Street, Vernon, CT 06066 window does exist between the time the write successfully completes and the write pending flag is cleared. The failure mode in this case would not be a lost record but a duplicate record.

The extraction of data from the ring buffer is a two-stepped process in which the read function operates in a nondestructive fashion and the remove function must be used to delete a record from the buffer. Also locating the outgoing services here is natural because you have already assigned a block of buffer management code to service the incoming data. The process consists of continually performing reads on the buffer. If data is returned, it is posted for transmission to the host computer and a pending removal is indicated by setting a flag (remove pending), which inhibits further read operations. The remove function is invoked after the host computer acknowledges receipt of this data, and on its completion the remove pending condition is cleared. Note that for this process to work properly, the remove pending flag must always be set to the cleared state on power up. In other words this flag must be volatile.

RAM VERSIONS VERSUS PROGRAM VERSIONS

The situation I described in my opening remark was intended to be amusing. However, I must admit it's not entirely contrived. While there weren't any witnesses, it did happen to me. Let me explain. The tweaks I made to the code inadvertently allocated an extra byte in the RAM area. This byte was added after the warm RAM string right in the middle of system RAM. The RAM, being initialized, allowed the system to warm start. Of course, once control was transferred to the main program, most RAM references were shifted a byte and the system got mighty confused. The only way to get things back to normal was to force a cold start in order *to* reinitialize all the RAM variables to where they were supposed to be.

Actually the solution to the problem is fairly simple: make the warm RAM string the program number and revision level. Program revisions will ensure an orderly start-up. I would also suggest providing an external means to force the program to cold start the system in the event of a catastrophe. This function could be invoked via a jumper, switch, or by keyboard from a system configuration mode. Having a clean way to restart the system is nice should your defensive coding and elaborate recovery mechanisms fail. Believe me, you don't want to talk your customer through the procedure of shorting out RAM power with a paper clip. ■

John Dybowski has been involved in the design and manufacture of hardware and software for industrial data collection and communications equipment.

IRS

428 Very Useful
429 Moderately Useful
430 Not Useful

MAXIMUM CONTROL MINIMUM PRICE

The Micro-440 is a versatile, full-featured controller with an on-board BASIC interpreter. Available with economical development packages (including C51) and a wide range of accessories.

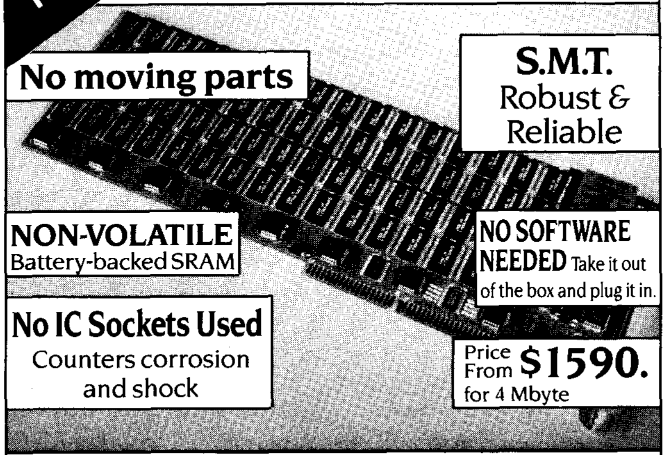
• AUTOMATION
• DATA LOGGING
• HOBBY/HOME PROJECTS
• OEM APPLICATIONS

FREE CIRCUIT BOARD OFFER

BLUE EARTH™ RESEARCH
310 Belle Ave • Mankato, MN 56001
507-387-4001 • FAX 507-387-4008
VISA/MASTERCARD ACCEPTED

#164

NEW SRAM Solid State DISK



No moving parts

S.M.T. Robust & Reliable

NON-VOLATILE Battery-backed SRAM

NO SOFTWARE NEEDED Take it out of the box and plug it in.

No IC Sockets Used Counters corrosion and shock

Price From $1590. for 4 Mbyte

IDEAL FOR HARSH ENVIRONMENTS
Humidity, dust, vibration. shock.

• 4 to 64 Mbytes
• Standard AT ISA bus
• Fully DOS Compatible
• Seek time. less than 0.2 millisecs
• Data retention at least 2 years
• Device driver on-board

Business Machine Interfaces Inc.
2 13-05 39th Ave., Bayside NY 11361
Canada (4 16) 756-9477
FAX (416) 756-4118
U.S. 1 ● 800*663+4605

#16

CONNECTIME

conducted by Ken Davidson

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

In this installment of ConnecTime, we're going to look at sensing current in an AC line, the ins and outs of halogen lamps, and some issues surrounding CRCs.

You might think sensing current in an AC power line to determine whether a motor is running is a simple matter. Think again.. .

Msg#:50778
From: RICHARD PFEIFFER To: ALL USERS

I am using an RTC52 to control a train horn. I need to monitor the status of a 1-HP AC compressor motor. The AC is always on, but the motor turns off based on the pressure in the tank. The control switch is a sealed unit and I cannot get in between it and the motor. What I need is a way of sensing current in the incoming AC line. I thought that wrapping a thin wire around one lead in the AC cord I could sense the current status, but it is too low.

Help. I don't do op-amps and such. If I were rich I would just whip out a PC and a GPIB voltmeter and all this would be simple, but I just need a simple TTL-level on/off signal. If anyone can help I would appreciate it.

Msg#:50788
From: DALE SINCLAIR To: RICHARD PFEIFFER

There are a few companies that sell items called "current monitors" for AC power control. I don't have any names at my fingertips, but there are plenty of ads in the trade journals. They work on the same principle you mentioned, and look like an epoxy donut with wires or pins coming out of it. They are available in a variety of current ranges, and some actually have a TTL output to indicate current on. Prices are more than you'd think, tens of dollars, but it's a good price for what you get.

Msg#:50789
From: JEFF BACHIOCHI To: RICHARD PFEIFFER

Are there any moving parts on this compressor you can get at? Use an opto-interrupter or somesuch to sense movement of the motor shaft, a pulley, or whatever is available.

Msg#:50805
From: RICHARD PFEIFFER To: JEFF BACHIOCHI

No, no accessible moving parts, but a neat idea. Someone at work suggested I monitor the dam pressure change, but I'd need a O-200 psi sensor and RTC-IO board.

Msg#:50796
From: MARC WARREN To: RICHARD PFEIFFER

You might also consider a Hall-effect sensor. These are solid-state devices that can sense a small magnetic field and switch on or off depending on the field strength. You could use it to sense current in the line cord or directly on the motor if enough field "leaks" out. Sprague and Microswitch manufacture these things—TTL output is available for most devices.

Msg#:50815
From: MICHAEL MILLARD To: RICHARD PFEIFFER

What you need is Catalog #20 from Microswitch (Honeywell). Their Atlanta number is (404) 248-2565. In this catalog, you will find exactly what you are looking for. What is the current range you are looking to measure? (I may have an extra sample laying around?) Also, do you just need a comparator output when a preset threshold is exceeded or do you want a binary output scaled to the current? Either way, it's in #20. Also included are the Hall-Effect devices that others are suggesting.

Msg#:51089
From: PELLERVO KASKINEN To: RICHARD PFEIFFER

A word of warning: you may not really benefit from plain AC current sensing! The current in a motor with an idle compressor changes very little when the compressor is loaded. What changes is the phase angle (power factor]. In other words, there is plenty of current sloshing through the motor windings even when the compressor is idle, but it is practically 90 degrees out of phase with the supply voltage. When the compressor turns on, the phase angle gets much more in line with the voltage, when real work is being performed.

CONNECTIME

You might need just an old fashioned CT (current transformer), available from your local electric contractors or their supply houses for a few bucks. The problem with them is that they generally produce a 5-ampere secondary, while you would benefit from a 1-ampere or even better 0.1-ampere secondary. You can make one yourself. You actually indicated you have tried, but the missing part was the magnetic path-a wire alone will not do. And definitely, it cannot be wound around the motor lead.

What goes around the motor lead is the magnetic core. Mostly often a toroidal (doughnut) shape, but any closed path is OK. Then you have a winding of several turns around that core. In fact, the motor lead can pass the hole in the core several times as well. What you get is a transformer. The ratio depends on the turns just like in any transformer, but normally you consider the voltages and then the ratio is direct. In CTs, the ratio is inverse. Let's say your primary current is 10 amperes max. Make 10 turns through the core and you have 100 ampere-turns. Now make 1000 turns as the secondary. The ampere-turns must match, so you get 0.1 amperes times 1000 turns to produce the 100 ampere-turns. Simple?

After the current transformer, YOU MUST HAVE a load resistor!!! Consider the primary being 230 V. Your turns ratio, if there is no load, will produce 23,000 V. Hardly something you want to happen.

For a 0.1 -A current and normal electronic signal levels of 5 to 10 VAC, you need a 50- to 100-ohm power resistor, preferably two in parallel, secured directly to the secondary winding ends, not into some terminal strip far away. You don't want the load to ever be opened accidentally.

As to the phase angle detection, you need another transformer that takes your voltage down to the same 5- or 10-V level. Then you make a zero voltage detector. Finally you sample the current signal at these zero crossings. It is a basic principle of it, although an original signal phasing needs to be provided, but I would need too much space and some graphics capabilities for all of that.

The popularity of halogen lamps is increasing every day. Their benefits over ordinary incandescent bulbs, including higher efficiency, whiter light, and smaller size, add to the attraction. But how do they work, and what are some practical issues to keep in mind when using them?

Msg#:51464
From: ANDREAS MEYER To: ALL USERS

Here's a question for you home-control wizards.. .

I'm using a commercially available photoswitch to control an incandescent lamp in my living room (so when it gets dark outside, the lamp or whatever is plugged into the switch, is turned on). The lamp I'm using is the type that takes a 3-way bulb (which may or may not be relevant), but it seems that at a fairly regular interval, one of the bulb's "stages" has burned out and I need to replace the bulb. (This interval seems markedly shorter than the average life of the same sort of bulb, so I suspect there's something about the way the photoswitch works that is shortening the life of the bulb.) Can someone explain why this is happening, and what I can do to prevent it?

Also, I'm thinking of replacing the entire incandescent lamp with a halogen torchiere, and would like to control it with the same photoswitch. Are there any reasons why I shouldn't use the halogen lamp in this way? And, can I expect to go through as many halogen bulbs?

Msg#:51471
From: PAUL PETERSEN To: ANDREAS MEYER

If your photoswitch is one of the cheap K Mart varieties you might be stressing the bulb's filament. I have one in my kitchen and have noticed in the morning, the light "flickers" a bit just as the sun is coming up and morning twilight is fading. It's that very critical crossover point where the photodetector doesn't know if it's dark or light. It only lasts a couple of seconds but while it's turning on and off very rapidly, I can hear the filament in the bulb twanging away. I've wondered if this was good for the filament, but the bulb is a 20-watt night light and only burns out after 6 months of usage...hm...

Msg#:51502
From: KENNETH SCHARF To: PAUL PETERSEN

I have one of those photoelectric units on an outdoor porch lamp. It turns the bulb on slowly because the SCR goes into partial conduction during the period that it isn't quite dark yet. Instead of being "digital," the unit seems to be "linear," the photoresistor being part of a divider network that changes the conduction angle of the SCR. As a result, the bulb never goes abruptly on or off, and while going on is "preheated" slowly. I think this extends the life of the bulb, rather than limits it. (I got over a year out of the last bulb: a 40-W "bug lite.") So it works both ways...

Msg#:51495
From: ED NISLEY To: ANDREAS MEYER

The "high" filament has a shorter life expectancy than the "low" filament, on the assumption that you only turn up the steam when you need more light. That assumption is completely wrong for those of us with lights on timers, as we never touch the lamp switch.. .

I finally gave up on one of my lights and installed a standard bulb of the right wattage; works like a champ, gives off more light, and lasts longer, too. If you've got a real problem, try a rough-duty bulb that will cost more than the photoswitch...but will continue to work while you pound it on the floor.

Msg#:51567
From: KEN DAVIDSON To: ANDREAS MEYER

Something to be careful of is whether you use a 120-V or a 12-V halogen lamp. If the photoswitch uses a triac to do the switching, chances are the waveform getting to the lamp is somewhat chopped. That is fine for a 120-V lamp, but the low-voltage lamps use a transformer that could be damaged by the chopped power.

We've been shopping for halogen lights for our new house and I want to control them with X-10 lamp modules. Just the other day it dawned on me that we can't consider the low-voltage variety if I want to use the X- 10 module.

Msg#:51669
From: STEVE LANGER To: ANDREAS MEYER

In message #5 1464, Andreas Meyer describes an unusually high incidence of filament failure when using bulbs controlled by a "photo switch." (I assume that's an SCR which, in turn, uses a photocell to generate its trigger signal).

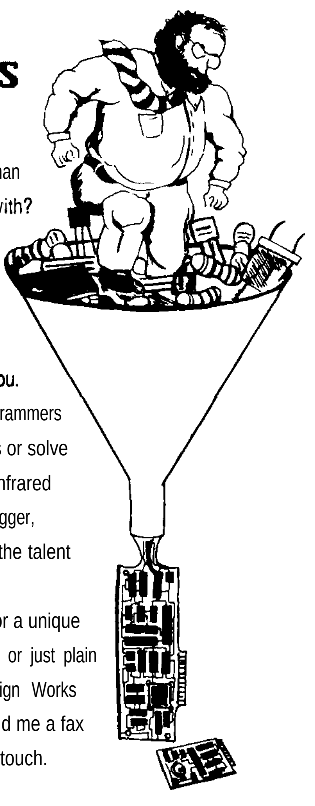
When cold, a tungsten filament has a much lower resistance than when it is incandescent:

300	K-5.6	micro-ohms/cm
1000	K-24.9	" " "
2000	K-56.7	" " "
3000	K-92.0	" " "
3600	K-115.0	" " "

(A ten-fold increase in temperature results in a twenty-fold increase of resistivity).

This means that at turn-on, the SCR is initially seeing a load that more nearly approaches a short circuit. The first conduction cycle(s) after turn-on may, therefore, allow a very high peak current, which then quickly normalizes in

The Ciarcia Design Works



Does your big-company marketing department come up with more ideas than the engineering department can cope with?
Are you a small company that can't afford a full-time engineering staff for once-in-a-while designs?
Steve Ciarcia and the Ciarcia Design Works staff may have the solution for you.
We have a team of accomplished programmers and engineers ready to design products or solve tricky engineering problems. Need an infrared remote controller, multi-channel data logger, or 7-axis drill controller? The team has the talent to design and manufacture it!
Whether you need an on-line solution for a unique problem, a product for a startup venture, or just plain experienced consulting, the Ciarcia Design Works stands ready to work with you. Just send me a fax discussing your problem and we'll be in touch.

Remember...*a Ciarcia design works!*
Call (203) 875-2199 • Fax (203) 872-2204

8051 MULTITASKING EXECUTIVE

Develop Real Time Applications FAST!

Our DCE51 ™ Executive provides:

- Preemptive Scheduling
- 4 Priority Levels
- Up to 8 Tasks
- Memory Management
- Message Passing
- Interrupt Management
- Dynamic Task Creation
- Small, Efficient Kernal (4K)
- Monitor Task Available

Iota Systems, Inc.
PH: (702) 831-6302 FAX: (702) 831-4629
POB 8987
Incline Village, Nevada 89452

#166

subsequent cycles as the filament heats up. In an "unlucky" circuit, the filament can-and does--fail instantly during that first surge, without ever getting a chance to start working as intended.

In a three-way lamp, one of the filaments has a lower resistance to start with; it's the one that probably would tend to burn out first during the initial high-current peak. Is this the effect that's been observed?

In the case of low-voltage, high-intensity lamps with compact filaments, the phenomenon is exacerbated. A 6- or 12-V high-intensity lamp can easily burn out instantly when turned on via a simple SCR controller, whereas it would not do so if activated via a mechanical rheostat or variable-transformer arrangement. The failure is caused by the very high initial current peak allowed by the SCR.

So called "halogen lamps" may or may not be be more resistant to this kind of failure. On one hand, their filaments are capable of withstanding higher temperatures without burnout-that is the most important characteristic of the halogen lamp design, and is also the one that allows production of higher light intensities and higher color temperature. On the other hand, most (not all] halogen lamps are designed as low-voltage, high-current bulbs. This might militate against long life under SCR control because it's precisely this type of low-resistance filament that's most vulnerable to the initial current peak.

In Message #51502, Kenneth Scharf correctly stated that a "linear" (i.e., gradual) turn-on is likely to prolong filament life, whereas a "digital" [i.e., sudden) turn-on is likely to shorten it. That is precisely the case.

In a "halogen bulb" (more properly, a quartz-tungsten-halogen bulb), iodine is usually a component, but not necessarily the only halogen used. Bromine is often also added. The function of the halogen is not to condense and evaporate to/from the filament-that's quite impossible, because the filament operates at temperatures that are immensely higher than the boiling point of any halogen, including iodine.

What is actually happening is that the tungsten filament, the halogen vapor, AND the quartz envelope all form a closed, multiphase, dynamic chemical system.

In a regular bulb, tungsten atoms evaporate from the hot filament and condense on the coolish 'walls* of the envelope. When enough of this action has taken place, the filament becomes thin at some point, its resistance at that point increases, as does its temperature there, causing a still more rapid evaporation. This initiates a vicious circle which accelerates until severance (burnout) occurs at that point.

In a quartz-tungsten-halogen bulb, the halogen vapor combines with the tungsten atoms deposited on the envelope wall, forming a volatile tungsten halide and becomes a "reverse transporter" of tungsten FROM the (hot) walls of the quartz envelope BACK to the (still hotter) filament, where the tungsten-halogen compound is decomposed by heat with redeposition of tungsten and liberation of halogen, which is then recycled. This retards the thinning of the tungsten filament, thus giving increased filament life. The recycling process doesn't work if the envelope walls are cold(ish) or if they are not made of quartz (silicon dioxide).

Increased filament life enables operation at higher temperatures BY DESIGN, and halogen bulbs indeed provide a higher color temperature [as well as one hell of a lot more infrared] than conventional bulbs.

Actually, the higher filament temperature is not only a possibility, but also a necessity, as is the small envelope size. Both factors cause the walls to be very hot, which is needed for the halogen(s) to combine with the tungsten atoms deposited on the envelope walls); it helps keep the tungsten redeposition process going at a good rate.

The may have seen lamps with dimmers that seem to require a higher setting of the variable-intensity control to turn ON than to turn OFF; a sort of hysteresis. This, too, is explainable in terms of changes of filament resistance with temperature. It takes more current to make a cold filament start producing light than to keep it producing light once it reaches operating temperature.

Msg#:51691
From: KEN DAVIDSON To: STEVE LANGER

That is why I always thought halogen lamps shouldn't be dimmed. We learned in physics that in order for the processes in the halogen lamp to work properly, the lamp must get very hot. Dimming the lamp, it would seem to me, would not allow the lamp to achieve the proper temperature and would greatly shorten its life.

Msg#:51996
From: STEVE LANGER To: KEN DAVIDSON

Yes, but if the lamp is dimmed only for short periods, and is allowed to operate at its normal high temperature most of the time, it should be all right, don't you think?

I just saw some desk-type lamps with a halogen bulb in them, and they also had what looked like an SCR dimmer. Of the three samples on display at the store, two had nonfunctional dimmers [they would function like a ON/OFF switches but wouldn't do any dimming), the third one was working as intended.. . wonder why?

CONNECTIME

Msg#:52042
From: KEN DAVIDSON To: STEVE LANGER

I picked up a book on interior lighting and it explained almost what you just did. Running the lamp dimmed will indeed allow the inside of the quartz to begin to blacken, but running the bulb at full intensity every so often will allow the complete reaction to take place and will "restore" the lamp.

Msg#:51997
From: STEVE LANGER To: KEN DAVIDSON

By the way, my source for info on halogen bulbs was the horse's mouth itself...some few (hrrrmmpphhh!) years ago, we were looking to replace a conventional bulb in an instrumentation light source with a halogen bulb and were dealing with GE directly. I got to spend a couple of very interesting hours with two of the fellows on the development team that worked on halogen bulb technology there.

CRCs are very often used in checking communication data streams for errors. Might there be more to them than meets the eye?

Msg#:52670
From: KENNETH SCHARF To: ALL USERS

We use the CRC- ***16*** polynomial equations to checksum data on a transmission scheme on several of the products I work on. From what I have read, this polynomial equation will produce a unique number for a string of bytes of up to 4096 in length. The idea behind this is there are 32K bits in 4K bytes, and each bit may be a one or a zero. That adds up to 64K, which is represented in a 16-bit number. The polynomial is both data and position sensitive, interchanging bytes will produce a different result. Now if this is true....

If you know the original seed for the polynomial accumulator, and the length of the byte string [and the length is less than 4K), can you reconstruct the string from the CRC result [i.e., work the polynomial equation backwards]?

What a hell of a data compression scheme! Can someone tell me why this won't work?

Msg#:52890
From: ERIC BOHLMAN To: KENNETH SCHARF

The contention that a 4-byte CRC will be unique for all possible 4096-byte data streams can't be true. There are 232 possible 4-byte CRCs, but 2^{32768} possible 4K data streams. By Dirichlet's "pigeonhole" principle, that means at least one CRC will be associated with more than one data stream (though it does NOT mean that each CRC will be associated with EXACTLY 1024 data streams).

The real contention probably was there would be an almost negligible probability that two 4K data streams that were almost identical [most transmission errors affect a few bits, leading to a received data stream that's almost identical to, rather than totally different from, the transmitted one] would share the same CRC. However, by the pigeonhole principle above, different data streams will map into identical CRCs; the error-detecting power comes from the fact that it takes major, rather than minor, transformations to change one data stream into another one that shares the same CRC.

Msg#:52892
From: KENNETH SCHARF To: ERIC BOHLMAN

Well I knew I was missing something. The standard called for a maximum data length of 4096 bytes for use with

CONNECTIME

a 16-bit CRC sum. TANSTAAFL. (There ain't no such thing as a free lunch).

Msg#:52904
From: ED NISLEY To: KENNETH SCHARF

Uh, subject to one of the mathematical heavyweights giving you the true poop, I'll venture an opinion.. .

The CRC can be implemented by a tapped shift register, into which you feed the data as a serial bit stream and out of which you get the final CRC value, effectively "in parallel" at the end of the calculation.

If you started with the final CRC value in the shift register and shifted it "backwards" you'd also have to stuff the original data into the register to tell the "inverse" XORs which bit to "unset" in the right way...or something like that.

But I like the notion; it's sort of like the classic trick of exchanging the contents of two registers without using a temporary register. I've always wanted something like that for laundry so you could swap a basket full of clothes with a washer load without either losing socks behind the dryer or soaking the front of your shirt.

It's been a long day.. .

Msg#:52962
From: MICHAEL MILLARD To: KENNETH SCHARF

In short...no. (BUT IF YOU CAN CODE IT, WE NEED TO TALK!!!] Because CRCs aren't perfect (99.998x%), how would you know that you, as a receiver, were reconstructing a correct message? Somebody else had a more elegant way to express this thought, but the gist is there's a possibility of reconstructing a bogus message.

But I have another thought on this: It would seem to me that just reconstructing the original bit stream would be very rigorous for the receiver to do, perhaps greatly exceeding the receiver's resources.

I'm going out on a limb here discussing the more common CRC implementation, but the thought should apply to most other applications: Remember that CRC- 16 (at least as normally defined] is derived from the one's complement of the remainder that you get when you modulo-2 divide the original message (data) by the generating polynomial. (The equation for which is $x^{16} + x^{12} + x^5 + 1$.) This looks complicated to do in software (and I wouldn't want to code it without a serious math library) but it turns out to be a very simple thing to do in hardware by bit-fiddling with a couple shift registers and XOR gates. Like everything else, it always helps if you know the answer ahead of time, so knowing the hardware solution would greatly accelerate the programming approach time.

Anyway, the idea here is that the reverse bit-fiddling is not an option. Therefore, the receiver processing overhead would likely be overwhelming. In the time it might take the receiver to reconstruct the message [assuming you could make an accumulator wide enough), you could probably have just sent the entire message conventionally. Probably more than once.

But your idea does have some practical uses.. .

There are some other schemes whereby the receiver actually does do a little math to (well not quite reconstruct a message but...) correct a few bits of the transmission. A good example would be the Motorola Bravo radio pager. This device uses a coding scheme called 23/12 Golay which is a signaling format designed to correct a few bits in the pager address field and a couple bits in the data message field. This is all done without sending the message a second time. Pretty slick! This format uses a 23rd ordered polynomial as the generating polynomial. POCSAG, another paging format, uses the same order and operates in a similar fashion but has a higher radio transmission rate. (And we just think they're a nuisance!)

As a final note, it is interesting that not all orders of generating polynomials allow for easy bit-fiddling as indicated above. You'll need to be a math major (as I am told) to really understand why, though. And since I've been cited before for not providing references, you'll find a good description of the XOR/shift register solution to the CRC-16 problem in the Sept. '86 BYTE article on the subject. You might also check the Sept. '90 C Users *Journal.*

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

[illegible]

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) 87.52199. Be sure to specify the issue number of each disk you order. Please add $3 for shipping outside the U.S.

IRS

431 Very Useful *432* Moderately Useful 433 Not Useful

STEVE'S OWN INK

A Night In The Life

Now that there is considerable water under the bridge, I don't mind telling you I had certain trepidations about converting from the old home control system to the new HCS II. Granted, I had long ago exceeded the design capabilities of the original unit and had resorted to a number of outboard "patch" systems to compensate for its shortcomings. But the prospect of replacing an entire wall of electronics, even if it would end up considerably smarter, was a genuinely frightening prospect.

Because we have a rule about testing what we design, however, the only appropriate way to observe the new HCS II was to install it and use it for real. Quite understandably, finding volunteers around here willing to wire their homes for "science" was like asking a Texan to eat sushi for the next six months. The net result was that Ken and I got the "privilege" of being guinea pigs.

Of course, one doesn't just swap an EPROM or switch an I/O board when making a great leap in technology. In reality, converting from the old HCS to the new one meant walking up to the control board, pulling the master power switch, and forcefully applying a crowbar to the majority of electronics on the board.

After six years of blissful living, I found myself standing in total darkness with no idea how to turn the lights back on. I had remembered to bring a manual X-10 controller and a list of codes, but never having been left in the dark before, I forgot the obvious need for a flashlight to find a plug. Going for a wall switch was even more trying.

When you don't need wall switches and outlets, they are quite easy to misplace. I know an outlet was somewhere along this wall six years ago. What? Who put this bookcase and file cabinet in front of it. I can't even reach it anymore! Argh!

Eventually I found an outlet, inserted the manual controller, and pressed "All Lights On" for the three house codes I use. The whole house was lit up like a Christmas tree. At least I wasn't in the dark anymore.

After a few hours of this charade, I remembered why I hated manual X-10 stuff and built an automated controller in the first place. Guinea pig or not, I was committed to installing HCS II posthaste.

Given the ease of writing the control sequences in XPRESS, I had the entire lighting system running in an evening. After all, how difficult is it to write a control equation that actually makes sense (as opposed to some languages)?

It would be more than an oversight for me to say that first night of testing went completely trouble free. Al about 2 A.M., the driveway chime sounded, indicating that someone had just driven in. About 20 seconds later it sounded again. Another car? Next, the high-power blue strobe lights in the driveway started flashing, and the chime sounded again. What, a motorcycle convention?

I grabbed an X-10 controller and pressed "off" codes for the strobes. The strobes went on again four minutes later! Hey, Ken?

A sleepy and somewhat bemused Ken look my call and listened to my plight. In between yawns he asked if I had disabled automatic refresh. Disable automatic refresh? I didn't even remember enabling it. OK, no sleep tonight until this gets fixed.

Down in the Circuit Cellar I fired up the PC and booted HOST to talk to the HCS. (How do I say, "Cease and desist"?) The X-10 status table on the screen showed a bunch of modules on that shouldn't be on. I fired up another PC laptop and a stand-alone PL-link to monitor the power line. Once in logging mode, it started vigorously listing X-10 transmissions that were occurring about every 5 seconds. Say what?

After looking at my XPRESS listing I had indeed not specified anything about refresh rate, so I added REFRESH=0 (no refresh) to my program. I quickly recompiled and reloaded the HCS.

I went out to the control board to detach the laptop when I noticed that X-10 commands were still being sent once every 8–10 seconds or so. Come on, Ken, I don't need bugs like this at 3 A.M.

Considering that this was starting to look like a runaway controller, cutting off the output seemed like a logical alternative. In exasperation, I reached up to the PL-Link and unplugged the TW523 X-10 transceiver. Disconnecting the HCS from the power line would stop the transmissions until I could work on it in the morning.

One glance at the laptop monitoring the power line told a completely different story. X-10 codes were still being transmitted every 8–10 seconds!

After I pulled the power to the whole PL-Link module and the transmissions still persisted, my thoughts wandered lo a nearby Browning 12 Gauge Over 8 Under which could be used as a last resort if logical alternatives failed. At 3 A.M. I wasn't going to call Ken and tell him we had invented an Immaculate PL-Link.

This was so absurd it was funny. Fortunately the logic of what was happening hit me soon enough to get some sleep after all. It turns out that smart testing takes smarter human testers. When I put the second PL-Link on the power line to print what it heard, I neglected to check its other default conditions. While it was "listening," it was updating its own module status table. The fact that some of these transmissions were logically wrong (later confirmed as a wiring error) didn't mean anything because they were still valid X-10 codes. Because I had neglected lo shut off refresh in the second PL-Link (the default was 5-minute refresh) it was resending the "bad" codes. Since it also listens to itself when transmitting, the second PL-Link dutifully reported "hearing" all the same codes continuing to be transmitted.

As one night in the life of automated home control comes to an end we can only speculate what the next night will bring. I can't wait for the system to have a voice and telephone privileges, too.

Steve