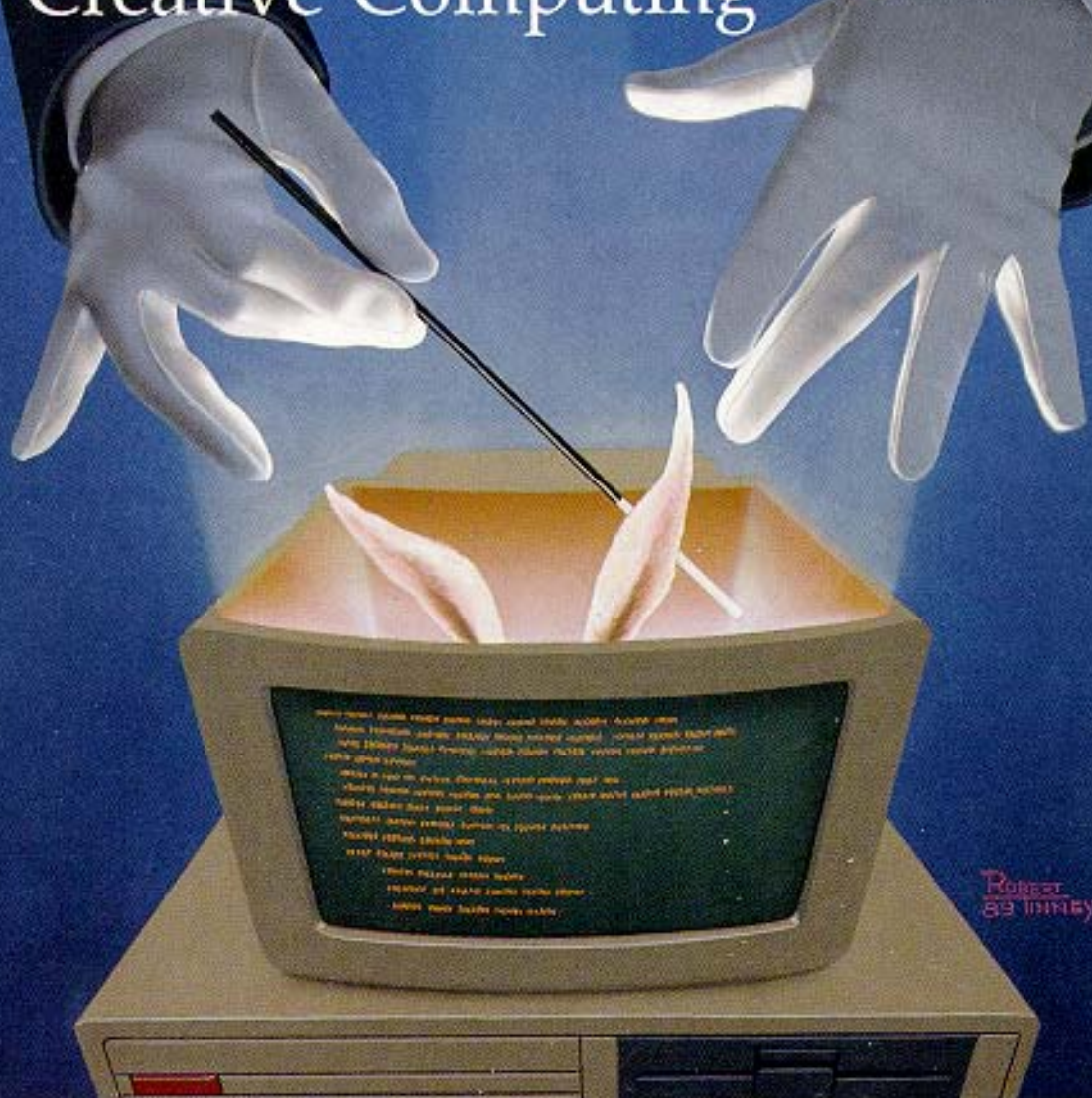


Circuit CELLARINK[®]

THE COMPUTER APPLICATIONS JOURNAL

Creative Computing



Robert
89 INKBY



April/May 1989 — Issue 8

\$3.95

EDITOR'S I N K

The Whole Story

Software is one of the major issues we wrestle with here at Circuit Cellar INK. On the one hand, our readers look to us for practical hardware solutions and innovative control techniques. There are several other high-quality magazines where readers can turn if their interests center on code rather than solder. Given that we have to be discriminating in our selection of articles, why not focus solely on hardware?

On the other hand, our subtitle reads *The Computer Applications Journal*, not *The Computer and Controller Hardware Journal*. We have promised practical, complete solutions, not black-box instructions. Since our readers are interested in functional applications, shouldn't we provide them with all the pieces required to build those applications?

When you look at our table of contents for this issue, you'll see articles on software. Yes, hardware is what makes us unique, but it's not enough. Programmable microcontrollers and complex designs have become so common that even the most diehard hardware engineers must recognize the importance of software.

It's become fashionable to talk about how huge the programs running on embedded controllers and other applications have become. This is used as a way of saying that software is the sole force driving the development of computer applications. I'm not sure that I believe all the numbers being thrown about, but I do see signs of a growing partnership between "engineers" and "programmers." Until recently, it was fashionable for hardware designers to disparage the work of programmers, and vice versa. Now, cooperation is common. What's more, hardware and software engineers are learning one another's disciplines, and finding that it makes their own work more creative and productive.

Circuit Cellar INK is dedicated to helping you become a better computer applications designer, engineer, or programmer. If we can teach you more about the discipline you know least, we're well on our way.

Speaking of Trends.. .

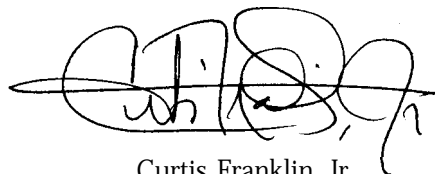
When you get as many press releases as we do here at Circuit Cellar INK, you can, if you squint just right, see trends start to develop. One of the trends that seems to be picking up steam is the use of microprocessors rather than microcontrollers in embedded and control applications. Members of the 8088 and 68000 families are showing up in places that, until recently, were reserved for 8052s and 6809s. What's up?

For starters, it's easy to see that hardware engineers aren't the ones driving this trend. After all, who wants to have to work extra glue, memory, and I/O circuitry into a design if you don't have to? It's not as though controllers aren't powerful enough. Controllers like the Intel 80960 are, arguably, more potent than anything the company offers on the microprocessor side. That argument aside, most control and embedded applications just don't need the wide data path, huge address space, and 25+ MHz clock speed of the latest microprocessors.

No, the driving reason for this trend can be found back about four paragraphs. Programmers and engineers who have to write the software for applications are getting more vocal about the need for better programming tools. In general, if you have an IBM PC on your desk, you have the basis for a high-powered 8088 software development system. If you have a 68000-based desktop machine, you're only so far from developing software for a 68000 control project. The fact is, software development time is now more valuable than mere hardware. This explains the appearance of PC-clones on just about every bus; a 4" x 6" card with 80386, 1 meg of memory, and VGA built-in; and 68000 UNIX single-board computers. I don't expect to see microcontrollers and their development tools disappear any time soon. Indications are that 8-bit microcontrollers will be the major portion of the market for years to come. But where microcontrollers were the only solutions available, now you can pick from a wide variety of platforms based on the many criteria (including financial and time) of a given project.

Time Marches On

Those with eagle eyes might have noticed that we jumped from the January/February date of Issue #7 to the April/May date of this issue. Is this a sign of corporate memory loss, or a personal vendetta against March? It's neither. It is, instead, the easiest way we could find of making the months on the cover and the real schedule match. You should notice no change in the way you've been getting Circuit Cellar INK.



Curtis Franklin, Jr.
Editor-in-Chief

FOUNDER/
EDITORIAL DIRECTOR
Steve Ciarcia

PUBLISHER
Daniel Rodrigues

EDITOR-in-CHIEF
Curtis Franklin, Jr.

ASSOCIATE
PUBLISHER
John Hayes

ENGINEERING STAFF
Ken Davidson
Jeff Bachiochi
Edward Nisley

CONTRIBUTING
EDITOR
Thomas Cantrell

CONSULTING
EDITORS
Mark Dahmke
Larry Loeb

CIRCULATION
COORDINATOR
Rose Mansella

CIRCULATION
CONSULTANT
Gregory Spitzfaden

PRODUCTION
MANAGER
Tricia Dziedzinski

BUSINESS
MANAGER
Jeannette Walters

STAFF RESEARCHERS

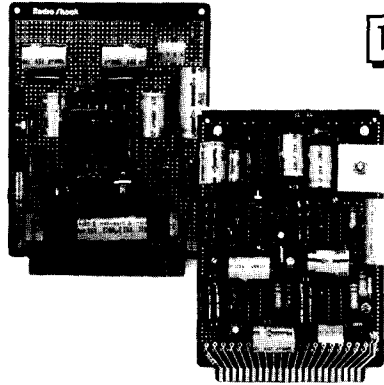
Northeast
Eric Albert
William Curie w
Richard Sawyer
Robert Stek

Midwest
Jon Elson
Tim McDonough

West Coast
Frank Kuechmann
Mark Voorhees

Cover Illustration
by Robert Tinney

FEATURES



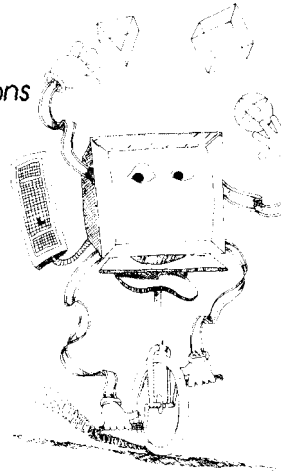
10 **Switching Power Supplies**
Efficient Power for Embedded Control Systems
by Steve Ciarcia

When you downsize the controller, the power supply has to shrink, too. Steve looks at the "black art" of designing efficient, clean power supplies in compact form factors.

22 **Product Reviews—**
The Next Generation
Circuit Cellar INK sells out and enjoys it!
Supercharged Worry Munchers
Circuit Cellar INK looks at four diverse applications solutions

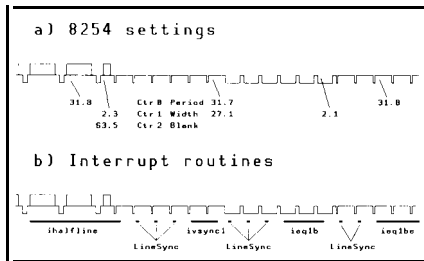
30 **Writing A Real-Time Operating System-Part 2**
Memory Management and Applications for the HD64180
by Jack Ganssle

Jack Ganssle discusses working with the HD64180's internal MMU and tips on writing applications in the conclusion of this two-part article.



DEPARTMENTS

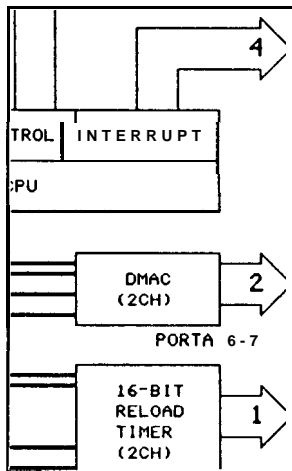
Editor's INK	
The Whole Story	1
by Curtis Franklin, Jr.	
Reader's INK—Letters to the Editor	5
Visible INK—Letters to the INK Research Staff	20
From the Bench	
Creating a Network-based Embedded Controller	46
by Jeff Bachiochi	



34 ImageWise/PC-The Digitizing Continues- Part 3

Topping it off with Software
by Ed Nisley

Although solder is the favorite programming tool of many engineers, software is the glue that holds **ImageWise/PC** together, as this three-part series concludes, Ed Nisley describes the software that makes this PC-BUS digitizer possible.



52 HD647180X—A New 8-Bit Microcontroller

Embedded Controllers Get Respect
by Tom Cantrell

While the state-of-the-art marches toward 32 bits, **8-bit** microcontrollers keep getting more powerful. The **HD647180X** is the latest integrated controller built on the foundation of the proven **Z80**.

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

The schematics provided in Circuit Cellar INK are drawn using Schema from Omotion Inc. All programs and schematics in Circuit Cellar INK have been carefully reviewed to ensure that their performance is in accordance with the specifications described, and programs are posted on the Circuit Cellar BBS for electronic transfer by subscribers.

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

CIRCUIT CELLAR INK (ISSN 0896-8985) is published bimonthly by Circuit Cellar Incorporated, 4 Park Street, Suite 20, Vernon, CT 06066 (203) 875-2751. Second-class postage paid at Vernon, CT and additional offices. One-year (6 issues) subscription rate U.S.A. and possessions \$14.95. Canada \$17.95. All other countries \$26.95. All subscription orders payable in U.S. funds only, via international postal money order or check drawn on U.S. bank. Direct subscription orders to Circuit Cellar INK, Subscriptions, P.O. Box 2099, Mahopac, NY 10541 or call (203) 875-2199.

POSTMASTER: Please send address changes to Circuit Cellar INK, Circulation Dept., P.O. Box 2099, Mahopac, NY 10541.

Entire contents copyright 1988 by Circuit Cellar Incorporated. All rights reserved. Reproduction of this publication in whole or in part without written consent from Circuit Cellar Inc. is prohibited.

Firmware Furnace

The True Secrets of Working with LCDs _____ 56
by Ed Nisley

Advertiser's Index _____ 65

ConnectTime— Excerpts from the Circuit Cellar BBS _____ 67
Conducted by Ken Davidson

Steve's Own INK

Smile When You Call Me That _____ 72
by Steve Ciarcia

READER'S INK

Letters to the Editor

I had a few comments on the article on multitasking ("Writing a Real-Time Operating System") and the discussion on control networks ("ConnectTime") which appeared in issue #7 of Circuit Cellar INK. As part of my own multitasking system (wearing many hats in a small company), I do hardware and software design, write our manuals, and do a monthly article for Radio World *Newspaper* (a technical newspaper for radio broadcast stations).

Multitasking

The multitasking system discussed in the article looks quite complete, and complicated. We've been using a simpler system on 6802-based systems. Hardware interrupts are used just for I/O buffering (rather than task switching). The interrupt-driven I/O buffering (using circular buffers) allows us to send or receive blocks of I/O from different devices or users. The system uses round-robin task switching with no priorities. Every task waits its turn. A task switch is initiated whenever the current task runs out of work. This is generally I/O related. Either the task is waiting for input and the input buffer is empty, or it has output and the output buffer is full. On exit, each task resets its "program counter" to the point where the decision was made that caused the exit. If there is a particularly processor-intensive task that we do not want to hog the system, it can be broken into parts. On exit, the task program counter is set to where the task is to continue. In 6800 assembly, the task manager looks like this:

```
TASKLOOP: JSR  TASK0  ;Go do task 0
           JSR  TASK1  ;Go do task 1
           JSR  TASK2  ;Go do task 2
           JSR  TASK3  ;Go do task 3
           BRA  TASKLOOP ;Go do it all again
```

We set up an area of RAM to hold the "program counters":

```
TASK0PC: DS  2      , -Set aside 2 bytes of
TASK1PC: DS  2      ;RAM for task 0 PC
TASK2PC: DS  2
TASK3PC: DS  2
```

Task entry looks like this:

```
TASK0:   LDX  TASK0PC;Get our program counter
         JMP  0,X    ; and go to it
```

Task exit looks like this:

```
TASK0F:  LDAB #0      , -Specify buffer 0
         JSR  BUFFULL;Go see if it's full
         BEQ  TASKOG  ;Cont routine if not full
         LDX  #TASKOF;Point where come back
         STX  TASKOPC;Store as PC for this task
         RTS                ; and exit

TASKOG:  ;Buffer not full, so output
         LDAB #0      ;Specify buffer 0
         LDAA #7      ;Get a bell
         JSR  PUTBUF  ; and put it in buffer 0
```

On entry to task 0, we pick up the program counter for this task, then start executing at that point (in the example, `TASK0F`). We redo the test that caused the task exit (checking to see if buffer 0 is full). If it's still full, we set up our task program counter for next time. If it's not full, we continue with the task (here outputting a bell character).

The key to this approach is that whenever a task needs to wait, it goes on to another task. When each other task is carried as far as it can go (without waiting), we come back around to see if this task is ready for action.

This approach uses only the standard hardware stack. There are not separate stacks for each task. Note, however, that this complicates the use of local variables. Since many high-level languages keep the local variables on the stack (throwing them out on procedure exit), the stack would quickly get confused. The simplest approach to this problem is to just use global variables that are always allocated. Each task needs its own set of variables. If each task is running the same program (thus having the same variable requirements), each variable could become an array that specifies which task the variable is used by. To save memory, an array called `stack (Task)` could be set up. Temporary variables could be stored on this "high-level-language" stack that is dedicated to this task. Since they are no longer needed, the stack pointer would be moved to delete them. This "high-level-language" stack allows reuse of memory for temporary variables.

Of course, one purpose of the hardware stack is keeping track of subroutine return addresses. If a task gets

three subroutine calls deep and then exits, all those return addresses are still on the stack. The stack will very quickly become confused again. For simplicity, all task exits would have to be from the "main-line" code. This can be accomplished **by either checking for an exit-causing condition** prior to calling the procedures (see if the buffer is full prior to calling **Buf Put**) or **having** a flag (**ProcFail**) that is set if the procedure failed and needs to be run again (and cleared if the procedure did not fail). When we get back to the main-line code, we continue if **ProcFail** is false. If **ProcFail** is true, we set up our task program counter to repeat the procedure next time, then exit this task.

Networking

In **ConnecTime**, there was quite a discussion of network protocols. I had a few comments (again, based on systems we manufacture).

Mark Lampkin gave a suggested data packet format. His simple format did not contain a return address or error checking (he's gotten no errors). His packet consisted of:

STX, Address, Command, Packet Length, data bytes, ETX

I'd suggest the following format:

Hex AA, Hex 55, ToAddress, FromAddress, PacketLength, data bytes, CheckSum

The AA55 hex is a beginning-of-message flag. We need to have a flag that will never occur in our data. Synchronous protocols do this by sending more than the maximum number of allowed 1 bits in a row in a character. To prevent a data byte from having too many 1s in a row, they use "bit stuffing" to stuff a 0 in the character after the maximum number of 1s. On the receiving end, after the maximum allowed number of 1s is received, the following "stuffed" 0 is removed, restoring the data to original.

In asynchronous communications, we cannot stuff more bits in a character (although some protocols send a break as a begin-of-message flag, which is detected at the receiver as a framing error). What we've done is "byte stuffing," quite similar to bit stuffing. If the data contains an AA hex, we stuff a 00 after it on transmission and remove it on reception. This guarantees that the flag (AA55 hex) will never occur in the data string. By the way, AA55 hex is used since it is an alternating pattern of 0s and 1s, then the reverse pattern. Both Mark Lampkin's protocol and mine include the **ToAddress**. I've included a **FromAddress** to allow the other device to respond (whether with an acknowledgement or with requested data). Mark included a "command byte." I believe this **does not** belong in the header, but should instead be in the data bytes area of the message. This allows for multibyte commands and makes even "noncommand" data packets the same. If needed, the first byte of the data bytes section could be a "packet type" byte, which determines how the remainder of the packet is to be interpreted. As Mark, I then left room

for the data. I followed this with a checksum byte, which is the sum of all bytes in the message (excluding the AA55 hex flag, but including the addresses, byte count, and data). Another approach is to use the 2's complement of the sum as the checksum. The receiver then adds up all the bytes in the message, including the checksum. If the result is 0, the message is good.

Mark had an ETX byte to mark the end of the message. Since the message already includes a packet length byte, we already know where the end of the message is, making ETX unnecessary.

Contention Avoidance

Mark's system used a fiber-optic ring, where it is not possible to get contention (multiple devices transmitting on the same medium simultaneously). Each fiber link has only one transmitter and one receiver, so a "token passing" protocol does not appear necessary. Any device wanting to send a message may transmit it at any time. The next device in the ring receives that message. If its address matches the **ToAddress** of the message, it acts on the message. If the address does not match, it passes the complete message on to the next device in the ring. Eventually, the message should find a home.

Note that this ring could be expanded to a **multibranch**ed tree or matrix where each device has several links to several other devices. Received messages that are not for this device are passed on to other devices through any one of several output ports. The decision as to which port to use **can be** based on a table held in that device. The routing information could also be included in the message header. The device originating the message would include several bytes of **ToAddress**. For the destination site to answer, it is necessary to include several bytes for the **FromAddress**.

If there really were a possibility of contention (a bus topology where more than one device can transmit on the same medium simultaneously), a contention-avoidance system is desirable (though not required). If each device transmits at random times and the system loading is low, there is a good possibility the message would get through.

The next approach to contention avoidance would be to "look before you leap." Listen for someone else transmitting before you bring up your carrier. This eliminates some contention, but there is a possibility of several sites holding off until a carrier drops, then having them all come up at once. To avoid this problem, a delay can be introduced after detecting carrier drop and before bringing up carrier. If this delay is fixed (and different¹ for each site, a priority system is established. The site with the shortest delay gets in first. If a random delay is used, all sites have equal access.

Token Passing

In token passing, a "permission-to-transmit" token is passed from device to device in the system. When a device receives the token, it transmits any messages it is holding,

then transmits the token to the next device in the system. There is danger, however, of the token getting lost, due to a transmission error. When this occurs, the system must detect it and generate a new token.

Absence-of-Data Token Passing

In one of our systems, we've used "absence-of-data token passing." Each device in the system includes a timer that is set to a "sitedelay" on reception of a valid data byte. If the timer reaches 0 before a valid byte is received, a "site counter" is incremented and the site delay timer is set back to the initial site delay. This continues until the highest site number in the system is reached. After that, the site counter is reset. When the site counter is incremented, each device in the system checks to see if the site counter now matches its site number ("it's now my turn to talk"). If so, carrier is brought up and FE hex characters are sent for site delay and act as a 'leader.'

Once the "leader" is complete, we transmit all the data packets that are held in the transmit buffer (with no more leader). When the transmit buffer is empty, we leave the carrier idle for a couple character times to allow the last byte to get through, then drop the carrier. A SiteDelay after the last valid byte, all sites increment their SiteCounters, enabling the next site in the system. If any site does not have any data to send, it leaves its carrier off, automatically passing the "permission-to-transmit" token to the

next site in the system. In addition, each time a site receives a valid message, it sets its site counter to match the FromAddress of the received packet. This resynchronizes all sites in the system. This "absence-of-data" token-passing system is really just like a standard token-passing system except for the form of the token. Further, no site must regenerate a lost token, since the token is the lack of data rather than the presence of a certain data sequence. It's hard to lose something that was never there!

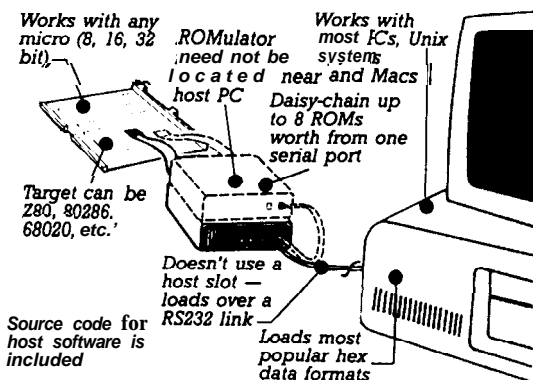
Conclusions

There are lots of different ways of handling networks. Right now, I like the absence-of-data token passing for bus systems and the routed matrix for nonbus systems. The routed matrix has some additional delay due to message retransmission, but this retransmission removes noise and timing inaccuracies from messages, resulting in better system performance. The routed matrix also has a higher throughput since the medium connecting two devices or sites is only carrying traffic that needs to go between those sites. In a bus system, all media in the system carries all the traffic. The routed matrix can be carrying several messages simultaneously over various portions of the medium. The bus can only carry one message at a time.

Harold Hallikainen
San Luis Obispo, CA

Still Blasting ROMs?

Develop and test ROM code in minutes without leaving your keyboard — with The ROMulator™.



Ask us about faster (than 150ns) ROMulators and custom cables for most non-27xxx ROM sizes.

FOR MORE INFORMATION, PLEASE CALL OR WRITE



Grammar Engine, Inc
3314 Morse Road
Columbus, Ohio 43231
614/471-1113

VISA and MasterCard Accepted
Dealer Inquiries Welcome

In Pacific and Mountain timezones:
415/595-2252

VOICE MASTER KEY™ VOICE RECOGNITION SYSTEM FOR PC/COMPATIBLES & TANDY 1000 SERIES A FULL FEATURED VOICE I/O SYSTEM

GIVE A NEW DIMENSION TO PERSONAL COMPUTING... The amazing Voice Master Key System adds recognition to just about any program or application. Voice command up to 2% keyboard macros from within CAD, desktop publishing, word processing, spread sheet, or game programs. Fully TSR and occupies less than 64K. Instant response time and high recognition accuracy. Voice recognition tool-box utilities are included. A genuine productivity enhancer!

SPEECH RECORDING SOFTWARE: Digitally record your own speech, sound, or music to put into your own software programs. Software provides sampling rate variations, graphics-based editing, and data compression utilities. Create software sound files you can add to macros for voice recognition verification response, a complete, superior speech and sound development tool.

SOFTWARE CONVERSION CODES: The Voice Master Key System operates a growing list of third party talking software titles using synthesized phonetics (text - to - speech) or digitized PCM, ADPCM, and CVSDM encoded sound files. Voice Master Key system does it all!

EVERYTHING INCLUDED: Voice Master Key System consists of a plug-in card, durable lightweight microphone headset, software, and manual. Card fits any available slot. External ports consist of mic inputs and volume controlled output sockets. High quality throughout. easy and fun to use.

ONLY \$149.95
COMPLETE

ONLY \$89.95 FOR TANDY 1000SL/TL MOODELS
SOFTWARE ONLY—REQUIRES TANDY BRAND
ELECTRET MICROPHONE.

ORDER HOTLINE: (503) 342-1271 Monday-Friday, 8AM to 5 PM Pacific Time

Visa/MasterCard, company checks, money orders, CODs (with prior approval) accepted. Personal checks subject to 3 week shipping delay. Specify computer type and disk format (3" or 5") when ordering. Add \$5 shipping charge for delivery in USA and Canada. Foreign inquiries contact Covox lot C & F quotes 30 DAY MONEY BACK GUARANTEE IF NOT COMPLETELY SATISFIED. ONE YEAR WARRANTY ON HARDWARE.

CALL OR WRITE FOR FREE PRODUCT CATALOG



COVOX INC.
675 Conger Street, Eugene, OR 97402
TEL: 503-342-1271 • FAX: 503-342-1283

Switching Power Supplies

Efficient Power for Embedded Control Systems

by Steve Ciarcia

With all the emphasis on embedded controllers and process control in the pages of Circuit Cellar INK, it was only a matter of time before I felt obligated to address the power requirements of such systems. After all, it **hardly** makes sense to build the world's most efficient micro-miniature controller only to have it powered by the world's most inefficient, grossly oversized, Neanderthal-technology power supply. At least, that's the way my logic went. Little did I realize what masochism I was inviting.

The Circuit Cellar has a variety of control systems. There is such a maze, in fact, that I'm even beginning to need a road map to figure out where everything is going. To alleviate some of the confusion and provide a substantial development base for future expansion, I will be converting much of my system to an RS-485 net with many locally intelligent data acquisition and control nodes. Rather than stringing a wire 200 feet out to the garage to monitor a switch closure, I will simply connect it to the garage controller node with the dozen or so other I/O contacts and its status will be transmitted with everything else. (This whole system will be described in Circuit Cellar INK in the coming months.)

Not only does this cut down on wiring (you *wouldn't believe how much wire there is around this place already!*) but it allows control changes to be done in a more orderly fashion. At the very least I'll have less aggravation tracing shorter wires. Most of the new system will consist of 8031- and 8052-

based dedicated controllers. [Editor's Note: For details on the RTC31 and RTC52 controllers, see "From the Bench" beginning on page 46.] So what's masochistic about this? Everything.

At the same time I decided to re-vamp the control system I thought I would take a more comprehensive approach toward power control and distribution. My present system has separate AC power supplies in each of its control areas. While I do have costly AC UPS power-backup units on critical computer control elements, the system is still susceptible to control errors when individual locally powered sensors lose power (usually false negative because they are no longer operable). Of course, there are various methods of redundancy and monitoring that I could incorporate to correct these problems, but it seemed like too much bother considering the frequency of such events. Nevertheless, I determined I'd fix this condition if I ever redesigned the system.

Instead of 115VAC for the new controllers, I intend to run everything on +12 volts. Like commercial alarm systems that run battery power even to the remote sensors, my new system could be entirely battery operated in the event of an AC power failure. One AC-line 12-volt power supply with a constantly charged 12-volt gel-cell (or car battery for that matter) could serve as an **uninterruptible** power source for the entire system. The same power supply design could also be applied to a single embedded control system to provide UPS operation.

Running things from 12 volts is

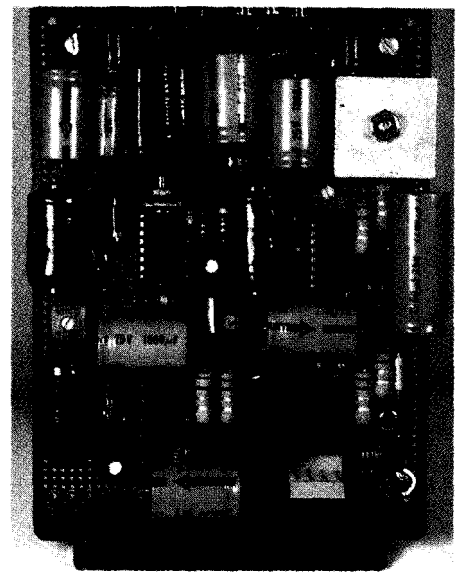


Photo 1—78S40-based design using traditional techniques.

nothing new. This application, however, presents some very special design considerations. Individual net controllers will **have** a variety of tasks. Some will be simple contact closure or temperature monitors. Others will have more elaborate configurations with event printers, modems, and displays. Even some form of data logging (hard disk) could be included.

From the outset I had to be aware of both power conversion efficiency

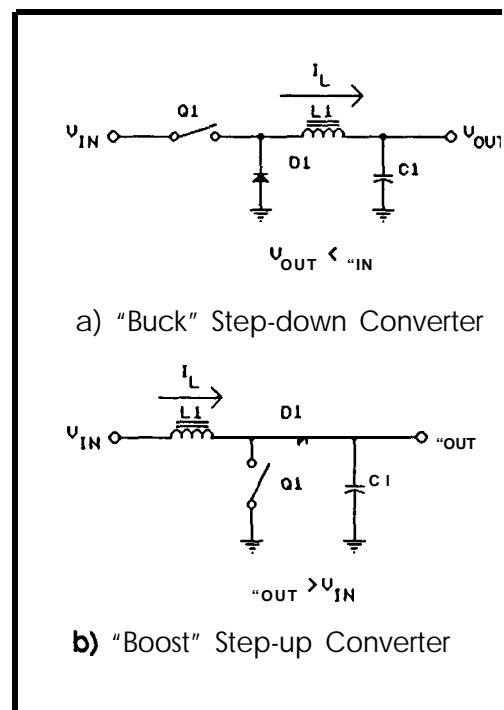


Figure 1 - Traditional DC-to-DC converter designs.

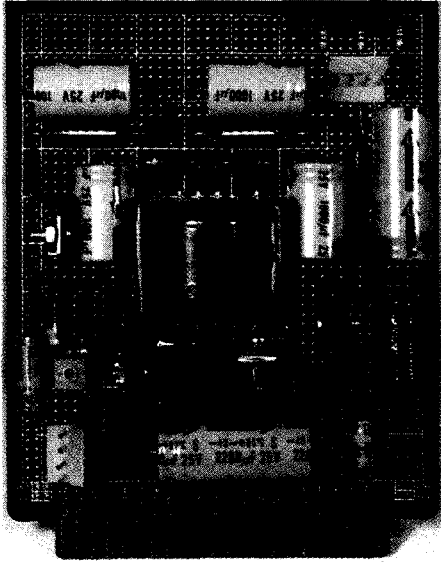


Photo 2— LT1070-based flyback converter.

and power consumption. Rather than tailor a custom supply for each node, I decided to build a general-purpose converter with three output voltages (+5V, +12V, and -12V) that could supply high currents where needed yet still have a low quiescent current when less consumption was required.

Unfortunately, getting from here to there is more easy to understand as a task than an accomplishment. Since I have built many low-current (less

than 100 mA) converters in previous projects, I perceived this as a weekend project where I merely extrapolated and expanded on basic design theory. Now, after successfully doing it, I can tell you that while the theory indeed holds true, the proof is more elusive.

General-Purpose Design Objective

My design objective was to build a DC-to-DC converter that could be used as a general-purpose 12V-powered UPS for embedded systems. Its modest specification would be: +5V at 1.5 amps, +12V at 0.5 amp, and -12V at 0.25 amp. Its efficiency should let the battery last a reasonable time. More importantly, it should be efficient enough so that the current requirements of the 12V common supply are not excessive when powering multiple converters.

This article documents the progression of events leading to a final power supply configuration. At the same time it answers basic questions on switching regulator-based DC-to-DC converter design. For the record, I don't claim to be an authority on this subject. This project is presented as a collection of tested circuits with useful observations because, in my experience, successful high-current DC-to-DC converter designs have more to do with layout technique and analog black magic than anything as tangible as component specifications or schematics. While my final converter exceeds the design objectives and is relatively easy to build, successful duplication of it will have a lot to do with your ability to hold the magic wand properly.

Since we have to start someplace, understanding the difference between plain-vanilla series-pass regulators and switching regulators provides a good introduction. I'm sure someone has already asked why we don't just regulate the 12V common down to 5V through a three-terminal regulator.

Series Pass vs. Switching Regulators

Since the advent of the three-terminal voltage regulator, it seems that

everyone has become a power supply expert. No longer are ten pages of calculations required to produce a design for even a modest power supply.

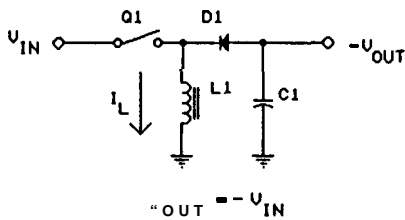
Three-terminal regulators like the LM317 are so easy to use that few experimenters stop to consider how inefficient they are. Consider for example, using an LM317 to power one of the controller nodes above. Given the maximum level of the common input voltage (V_{in}) for a 5-volt 1.5-amp output, 11 volts would be dropped across the regulator (in actuality, V_{in} will vary between 10V and 17V depending upon whether it is operating only from the battery or from an AC-powered charger and power supply). Power is dissipated by the regulator in an amount equivalent to the difference between the regulator's input and output voltages multiplied by the current through it.

The LM317 and similar linear regulators such as the 7805 and LM340 are all called series dissipative regulators. They function in a linear mode, simulating a variable resistance between the input voltage source and the load. The regulator maintains a constant output voltage by dissipating the excess power as heat. Unfortunately, as we see in this example, it consumes 16.5 watts producing the desired 7.5-watt output (31% efficiency).

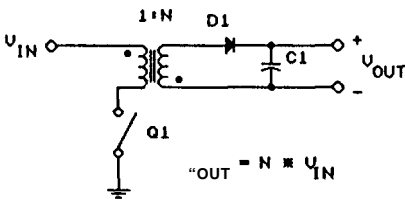
In most applications the ease of use and relative low cost of linear series regulators far outweigh the inherent lack of efficiency. The linear series regulator is well-suited for medium-current applications or applications with a small input/output voltage differential. When electricity isn't supplied by a battery and costs only ten cents per kilowatt-hour, it's hard to get concerned about losing 16.5 watts.

Why Use a Switching Regulator?

Power supply efficiency usually isn't important unless size, heat dissipation, or total power consumption is limited. Since the common power source for our network controllers is a battery (when the AC power fails), we have to be more careful about how



c) "Buck/Boost" Polarity-Inverting Converter



d) "Flyback" Transformer-Coupled Converter

much energy is converted for useful work and how much is thrown away as heat.

Efficiency is really the name of the game. In a series dissipative regulator, conversion efficiency is directly related to the input/output voltage differential. As the difference between the two voltages increases, efficiency decreases.

It would be far better if the regulator consumed no power and if all the power were channeled to the load. Of course, perfect conversion efficiency is impossible, but the inherent fault in using series dissipative regulators is the linear operating mode of the series-pass transistor. If the transistor is used as a switch (in saturated operation) rather than as a variable resistor (in linear mode the pass transistor consumes very little power. (This is not a new discovery.)

A regulator constructed to operate in this manner is called a series switching regulator. The same series-pass transistor switches between cutoff and saturation at a high frequency, producing a square wave of amplitude V_{in} . This waveform is then filtered through a low-pass LC (inductance/capacitance) filter, producing an average DC output voltage (V_{out}) proportional to the pulse width and frequency. The efficiency of such a regulator is generally independent of the input/output voltage differential and can approach 95% in good designs.

Switching regulators come in various circuit configurations, a few of which are the flyback, buck, boost, and buck-boost types. Also, unlike the typical three-terminal dissipative regulator, the switching regulator can be directly configured to operate in any of three modes: step down, step up, or polarity inverting.

Switching-Regulator Basics

Figures 1a through 1d outline the three common modes of switching-regulator operation. Basically, the switching regulator consists of a power source which supplies a voltage V_{in} , a "switch" Q1, and an LC filter. The way the components are connected determines the output mode.

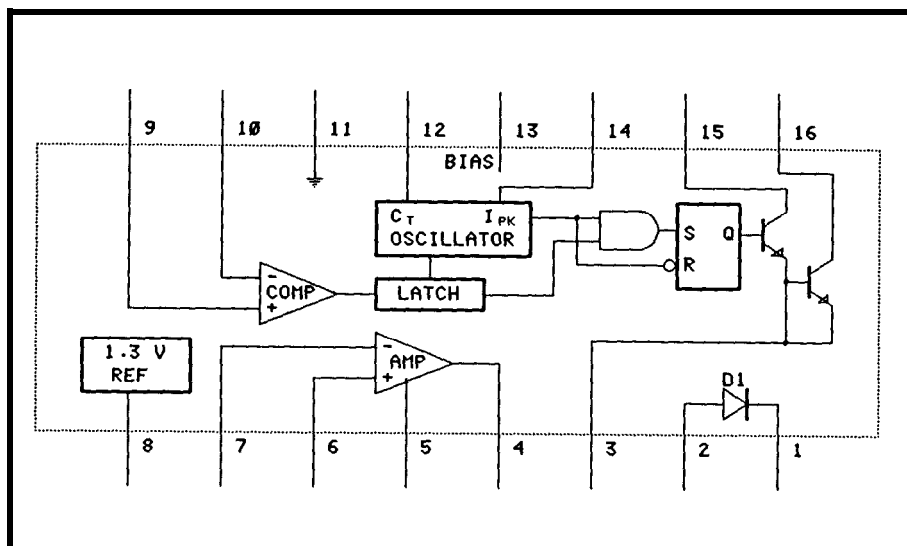


Figure Z-The 78S40 is a popular variable-frequency-type switching regulator.

Buck Regulator

In the step-down buck regulator in Figure 1a, the basic circuit operation is to close transistor switch Q1 for a time T_{on} , and then open it for time T_{off} . The total, $T_{on} + T_{off}$, is called the switching period T . Neglecting the saturation voltage of Q1 (V_{sat}) and the diode (V_{diode}), the voltage at the input to the inductor is $+V_{in}$ during the time T_{on} and zero during T_{off} . (These other voltage drops would be included in calculations that choose actual components.)

When Q1 is closed, a step increase in voltage is applied to the inductor coil, which has the value L . However, current flowing through an inductor cannot change instantaneously; instead it increases linearly according to the factor $L(di/dt)$, building a magnetic field. This reduces any instantaneous current change seen by the load. When Q1 opens, the magnetic field in the inductor decays linearly, supplying power to the load. The current path is completed through the forward-biased flyback diode D1. As you can see mathematically, the output voltage of a buck converter is always less than its input:

$$\text{Duty Cycle (DC)} = \frac{T_{on}}{T_{on} + T_{off}}$$

$$V_{out} = V_{in} \times DC$$

In this type of switching regulator, the inductor and capacitor form a low-pass filter. High-frequency pulses are applied to the input, and an averaged DC level comes out. The peak-to-peak ripple voltage is a function of the switching period T and the values of the inductance L and capacitance C . As the frequency of operation is increased, the voltage ripple is reduced, but the supply becomes less efficient.

Boost Regulator

Figure 1b illustrates the circuit configuration of the basic step-up boost regulator. In this type of regulator, closing Q1 during T_{on} charges the

CO OR IM GEWI E

ZIP Software now allows both the original serial DT01 and ImageWise/PC to digitize and display full-color pictures on an IBM PC with VGA.

See page 44 for more details,

inductor. When Q1 is opened, the inductor discharges through D1 into the load. The output voltage is:

$$V_{out} = \frac{V_{in}}{1-DC}$$

Current drawn from the input is delivered in pulses to the output load at significantly higher currents than the average load current. Both Q1 and D1 must be sized to handle these increased currents. You'll also note from the equation that in all cases, V_{out} is greater than V_{in} .

Buck-Boost Regulator

Figure 1c shows a polarity-invert-

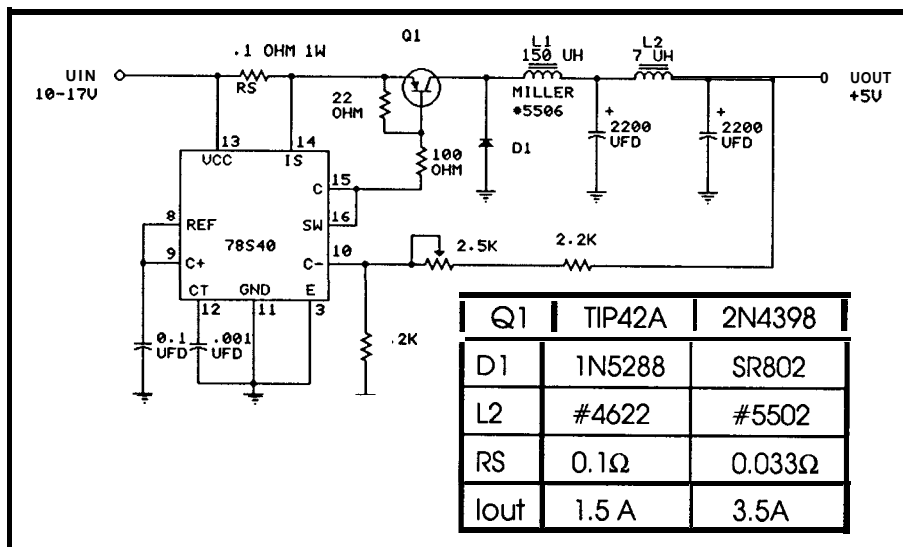


Figure 3-A 1.5-amp, 5-volt buck regulator using the 78S40.

ing buck-boost switching regulator. As in the other cases, closing Q1 charges the inductor during T_{on} . When Q1 is opened during T_{off} , there is a "kickback" voltage produced by the inductor as it discharges. This effect occurs elsewhere, too. For years, many of you have probably been putting reverse-biased diodes across relay coils, perhaps without thinking about it. The purpose of the diode is to dampen the high-voltage spike produced after a pulse is applied to the inductive relay coil. In a switching power supply, rather than short out the voltage, diode D1 directs this opposite-polarity voltage to the load. Buck-boost regulators have an output

given by:

$$V_{out} = -V_{in} \times (DC / 1-DC)$$

Note that the magnitude of V_{out} can be either greater or less than V_{in} .

Flyback Regulator

A final configuration worthy of interest is the flyback converter, shown in Figure 1d. Flyback regulators use a transformer, as opposed to a simple choke, to convert V_{in} to V_{out} . During T_{on} , energy builds up in the core due to increasing current in the primary winding. At this time diode D1 is reverse-biased. When Q1 opens during T_{off} , the total stored energy is trans-

ferred to the secondary winding and current is delivered to the load. The primary-to-secondary turns ratio (N) affects V_{out} and should be set for optimum power transfer:

$$V_{out} = V_{in} \times N \times (DC / 1-DC)$$

The greatest advantage of flyback regulators is that they can have an output voltage that is higher or lower than the input voltage and can include multiple windings on the transformer secondary to create other isolated voltages. Unfortunately, all this does not come without a price. Flyback converters have higher ripple currents due to the high energies which must

Variable- vs. Fixed-Frequency Regulation

Generally speaking, in all four cases, the output voltage V_{out} is regulated by controlling the ratio of T_{on}/T , which can be altered in a number of ways depending upon the control method. Two of the more common approaches are variable pulse width (pulse-width modulation) and variable frequency. In a pulse-width-modulated switching regulator, the switching period T is fixed and the "on" time T_{on} varied. Conversely, in a variable-frequency regulator, T_{on} is fixed and the "off" time T_{off} is varied.

The variable-frequency switching regulator is generally easier to design and build since the magnetic flux developed in the inductor coil during the fixed on-time determines the amount of power deliverable to the load. This eases the design of the inductor because the inductor's operating region within its characteristic curve is precisely defined. Operating frequency, which increases proportionally with the load, is mostly a function of the inductance L , capacitance C , and voltages V_{in} and V_{out} .

The fixed-frequency pulse-width-modulated switching regulator varies the duty cycle to change the average power delivered to the load. This method is particularly advantageous for systems employing transformer-coupled output stages and is often used in commercial switching supplies with multiple outputs. It is more complex and uses more components than variable-frequency supplies, but the advantages outweigh the extra cost in high-current applications.

Typical operating frequencies of switching regulators range from 10 to 50 kHz. However, there are some tradeoffs. High frequencies reduce the ripple voltage at a price of decreased efficiency and increased radiated electrical noise. If the frequency is lowered, greater efficiency and less electrical noise will result, but larger coils and capacitors are needed. Also,

a switching power supply operating at 10 kHz can become quite annoying to listen to after a while.

The most effective frequency range for optimizing efficiency and size with the components presently available is around 20–50 kHz. This is out of the range of human hearing yet low enough to be within the switching speeds of most inexpensive transistors and diodes. As switching speeds of newly developed high-current semiconductors increase and new ferrite components are introduced, practical operating frequencies will rise. There are discussions now about switching regulators that will operate at 1 MHz or more.

Putting Theory to the Test

My first approach to building the new switching supply was to use the 78S40 switching regulator which is readily available and inexpensive (a block diagram is shown in Figure 2). The 78S40 is a variable-frequency-type regulator which contains a current-controlled oscillator, current-limit

Characteristic	Buck	Boost	Buck-Boost	Units
I_{pk}	$2I_{OUT(Max)}$	$2I_{OUT(Max)} \cdot \frac{V_{OUT} + V_D - V_S}{V_{IN} - V_S}$	$2I_{OUT(Max)} \cdot \frac{V_{IN} + V_{OUT} + V_D - V_S}{V_{IN} - V_S}$	A
R_{SC}	$0.33/I_{pk}$	$0.33 I_{pk}$	$0.33 I_{pk}$	Ω
t_{on} t_{off}	$\frac{V_{OUT} + V_D}{V_{IN} - V_S - V_{OUT}}$	$\frac{V_{OUT} + V_D - V_{IN}}{V_{IN} - V_S}$	$\frac{ V_{OUT} + V_D}{V_{IN} - V_S}$	
L	$\frac{V_{OUT} + V_D}{I_{pk}} \cdot t_{off}$	$\frac{V_{OUT} + V_D - V_{IN}}{I_{pk}} \cdot t_{off}$	$\frac{ V_{OUT} + V_D}{I_{pk}} \cdot t_{off}$	μH
t_{off}	$\frac{I_{pk} \cdot L}{V_{OUT} + V_D}$	$\frac{I_{pk} \cdot L}{V_{OUT} + V_D - V_{IN}}$	$\frac{I_{pk} \cdot L}{ V_{OUT} + V_D}$	μs
C_T (μF)	$45 \times 10^{-5} t_{off}$ (μs)	$45 \times 10^{-5} t_{off}$ (μs)	$45 \times 10^{-5} t_{off}$ (μs)	μF
C_O	$\frac{I_{pk} \cdot (t_{on} + t_{off})}{8V_{ripple}}$	$\frac{(I_{pk} - I_{OUT})^2 \cdot t_{off}}{2I_{pk} \cdot V_{ripple}}$	$\frac{(I_{pk} - I_{OUT})^2 \cdot t_{off}}{2I_{pk} \cdot V_{ripple}}$	μF
Efficiency	$\frac{V_{IN} - V_S + V_D}{V_{IN}} \cdot \frac{V_{OUT}}{V_{OUT} + V_D}$	$\frac{V_{IN} - V_S}{V_{IN}} \cdot \frac{V_{OUT}}{V_{OUT} + V_D - V_S}$	$\frac{V_{IN} - V_S}{V_{IN}} \cdot \frac{ V_{OUT} }{ V_{OUT} + V_D}$	
I_{IN} (Avg) (Max load Condition)	$\frac{I_{pk}}{2} \cdot \frac{V_{OUT} + V_D}{V_{IN} - V_S + V_D}$	$\frac{I_{pk}}{2}$	$\frac{I_{pk}}{2} \cdot \frac{ V_{OUT} + V_D}{V_{IN} + V_{OUT} + V_D - V_S}$	A

Figure 4—Numerous formulas are used in the design of buck, boost, and buck-boost converters.

sensor, voltage reference, high-gain comparator, high-current op-amp, transistor switch, and power-switching diode. A single capacitor sets the frequency range (adjustable between

100 Hz and 100 kHz, but normally used at 20 to 30 kHz), and one external resistor provides current-limit sensing. Other than a few discrete resistors to set the output voltage, only an

Science, Engineering & Graphics Tools for Microsoft C, Turbo C and Turbo Pascal

Science and Engineering Tools are a collection of general purpose procedures and functions which solve the most common data analysis and graphics problems encountered in science and engineering applications. All procedures and functions are supplied on disk in the source code of the target language. The procedures and functions are compatible so the graphics functions can directly display the output of a regression, curvefit, etc. All of the routines can be used royalty free in compiled form. A 150 page manual describes the form, function, and parameters of each procedure and function. The Science and Engineering Tools are available for Turbo Pascal 4.0, 5.0, Turbo C 1.5, 2.0 and Microsoft C 5.x for IBM compatibles.

Ordering Information

Model#	Version	Price
IPC-TP-016	IBM Turbo Pascal	\$ 79.95
IPC-TC-006	IBM Turbo C	\$ 79.95
IPC-MC-006	IBM Microsoft C	\$ 79.95

Price includes shipping within North America. Elsewhere add \$18.00 for shipping. Mastercard, Visa, Company PO's, and personal checks accepted. MASS. residents add 5% sales tax.

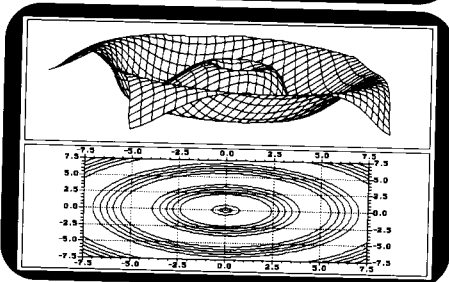
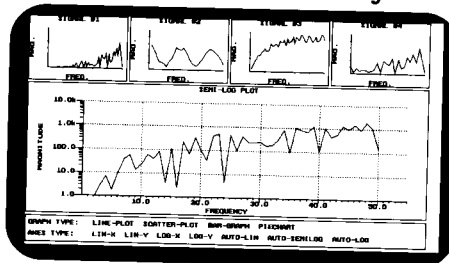
FEATURES

- 100% Royalty Free
- Turbo Pascal 4.0, 5.0 Turbo C Rev. 1.5, 2.0 or Microsoft C Rev. 5.x compilers
- CRT Graphics Adapter Support - the graphics libraries use the graphics routines supplied with the respective compiler. (CGA, EGA, Hercules, VGA)

Hardcopy support - Epson MX, FX and LQ printers, HP plotters, HP Laserjet and Thinkjet printers, Toshiba 24 pin printers and other devices

Science/Engineering charting routines - Linear, semi-log, and log graphs. Auto-scaling of axes, line, scatter, pie, and bar charts. **Charting**

Graphics Now Includes Contour Plotting.



3-D plotting - translation, scaling, rotation, and perspective routines

Statistics - mean, mode, standard deviation, standard error, etc.

Multiple Regression - With summary statistics

Curve Fitting - Polynomial and cubic splines

Simultaneous Equations - real and complex

Fourier Analysis - Forward and inverse FFT, Rectangular, Parzen, Hanning, Welch, Hamming, and Exact Blackman Windows, 2-Dimensional FFT and Power Spectrum

Matrix Math - Real and complex

Complex Number Arithmetic

Eigen values and vectors

Integration - Simpson's method

Differential Equation - Runge-Kutta-Fehlberg

Linear Programming - Simplex method

Root Solving - Bisection, Newton and Brent methods

Files Transfers - Lotus 1-2-3

Data Smoothing

Special Functions - Gamma, Beta, Bessel, error, hyperbolic trig, orthogonal polynomials

RS-232 Support - the Turbo Pascal version includes an interrupt driven RS-232 driver



1191 Chestnut St., Unit 2-5, Newton, MA 02164 USA
Tel. (617)965-5660 FAX (617)965-7117

inductor and capacitor are required to make a highly efficient switching power supply. (The internal Darling-ton-configured transistor switch and diode of the 78S40 are capable of handling 1.5 amps at 40 volts, but an external transistor and diode are

connected to a fixed 1.3V reference voltage. If the output voltage exceeds the reference, the regulator will begin to skip Ton cycles until the voltage lowers. Changing the output voltage setpoint in a buck converter is simply a matter of changing

higher or lower output current depending upon the speed and current ratings of these devices. Switching regulators use special Schottky Barrier Rectifiers specifically for their low forward-voltage drop (0.3-0.6 volts typically) and high speed. Using a TIP42A (6-amp) transistor and 1N5822 (3-amp) Schottky diode, I was able to obtain a 1.5-amp output current for a Vin range of 9-17 volts. By changing the transistor to a 2N4398 (30-amp) and the diode to an SB840 or SR802 (8-amp), the output current could be increased to 3.5 amps (the current-sensing resistance was lowered to 0.033 ohms to handle the higher currents).

Figure 4 outlines the mathematics involved in making buck, boost, and buck-boost converters with the 78S40 switching regulator. For the most part I used these calculations to form the fundamental basis of the end result with a certain sprinkling of empirical modification. Because the input voltage (Vin ranges from 10 to 17 volts) is not a constant, there is actually a range of component values which are all optimum at a specific combination of Vin and load current. The final component values are compromise selections derived by building the circuit and testing it.

An important fact about the buck converter also pertains to boost, buck-boost, and other configurations. Traditional switching regulators are electrically very noisy. Switching transients can be coupled throughout a power supply either inductively between adjacent components or directly through inadequate or misrouted grounding. **Grounding** in a switching power supply, like EM1 reduction, is one of those black magic areas.

Much of the noise generated consists of 100-200-ns spikes which occur when transistor Q1 is either turned on or off. It is not unusual to see 3-volt spikes (at about 30 kHz) on a 5-volt output line if you aren't careful! Eventually, you will discover that if you remove the 12" ground extension jumper from the scope probe and use just the short pigtail ground across the load that the noise is actually less than a volt. That's better, but not great.

In most cases, switching regula-

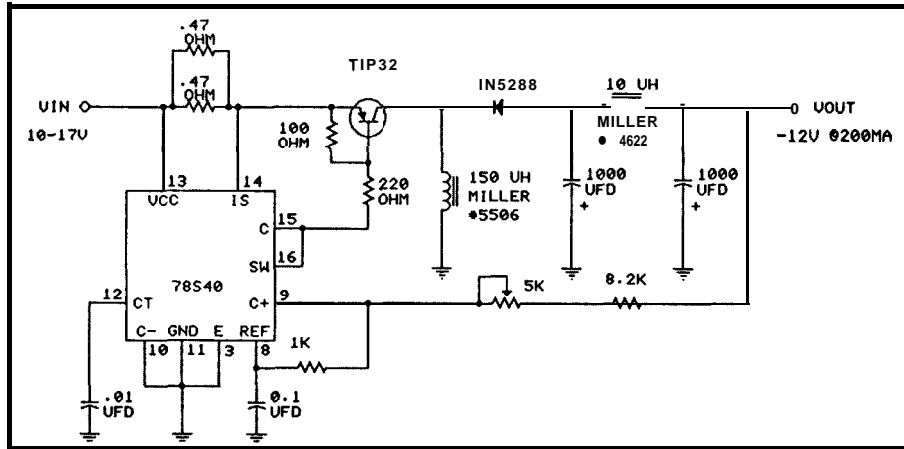


Figure 5-A buck-boost design using the 78S40 generates -72V from a 10-17V input.

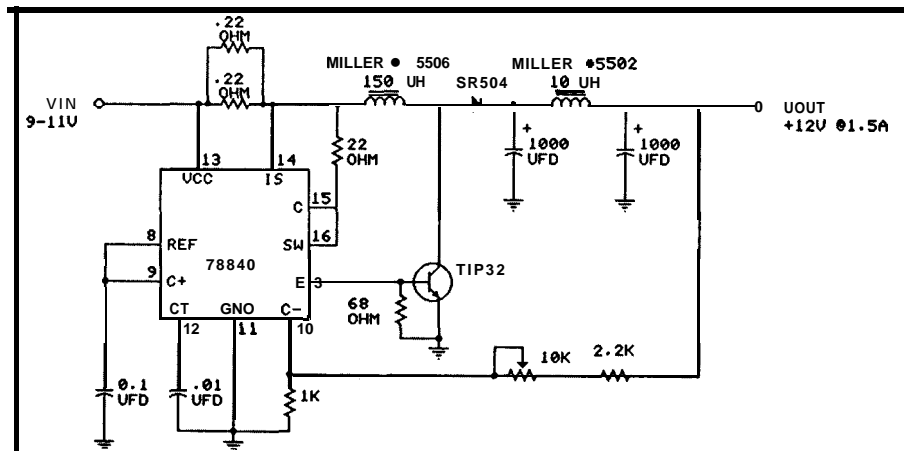


Figure 6-Again using a 78S40, a boost design is used to generate +72V from a 9-11V input.

needed here because of the increased currents involved in this design.)

5-Volt Buck Regulator

Figure 3 outlines a 1.5-amp, 5-volt buck regulator using the 78S40. Operating frequency is set by the capacitor at pin 12 (usually between 0.01 and 0.001 μF). Vin is connected through the current-sensing resistor to the transistor switch, and at Ton it charges the inductor. During Toff, the diode conducts and the energy in the inductor is transferred to the load. The voltage at the load is fed back to a comparator. The other side of the comparator is

the resistor divider between this comparator and the output.

The current-sensing resistors are intended to protect the switching transistor and diode rather than the load. The 78S40 will stop functioning when the voltage between sense points (pins 13 and 14) exceeds 0.33 volts. For a 1.5-amp buck regulator, the peak current is typically 3 amps and a 0.1-ohm resistor (or two 0.22-ohm resistors in parallel) is used.

As you might have guessed, the most important elements in switching regulators are the transistor and the diode. For the same inductor value, this regulator configuration can have

tor designs are greatly improved by post-regulator filtering. Capacitors alone, regardless of their size, usually don't eliminate these high-frequency spikes. Instead, an LC filter consisting of a 7-10- μ H choke and a 1000-4700- μ F capacitor works well in most cases. Be advised that these aren't just any old inductors. Typical low-cost molded inductors are only rated at a few hundred milliamps. Since we are talking amperes here, make sure that the filter components will handle the current. The 9-amp Miller chokes that I used reduced the electrical noise to about 200 mV peak-to-peak.

Negative 12-Volt Buck-Boost Regulator

Creating -12 volts using the 78S40 switching regulator is described schematically in Figure 5. The major difference between it and the straight buck regulator is that the inductor and diode are now in the reverse positions. The inductor is charged during T_{on} again, but this time during T_{off} the negative-polarity EMF gener-

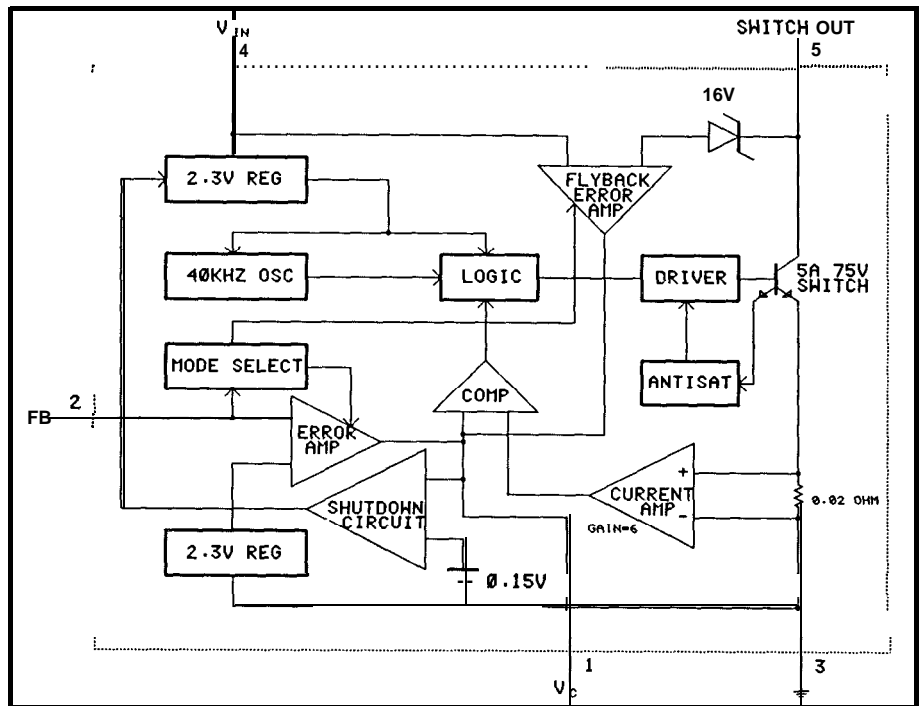


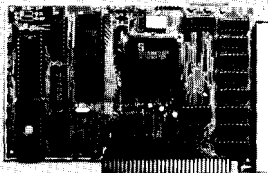
Figure 7-The LT1070 integrated switching regulator is often used in flyback converter designs

ated by the collapsing magnetic field in the inductor is directed through the Schottky diode to the output capacitor and load. Regulation set-point

feedback, current limiting, and LC output filtering are handled much as before except this circuit is designed to supply only about 200 mA.

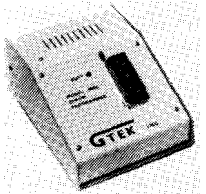
It's Your Move...

► INTELLIGENCE



- 8 Co-processor driven RS232 channels/card.
- BIOS Enhancement Software included.
- Eliminates host processor overhead.
- 32-2048KByte buffer memory.
- May be used standalone in custom appl.


► HIGH PERFORMANCE



MODEL 9000 (E)EPROM MPU PROGRAMMER

- Fastest programmer on the market.
- Quick & Intelligent programming algorithms.
- Supports megabit eproms.
- Programs largest variety of chips.


► VERSATILITY



MODEL PCSS-8X MULTIDORT SERIAL BOARD

- 8 RS232 ports par card (opt. 4 port).
- All 8 ports 100% DOS compatible.
- 32 ports may be added to a PG.
- Optional RS422 to 4000 ft.
- Interrupt driven BIOS Enhancement Software included.

► TIME SAVING




MODEL ROMX-2XL EPROM EMULATOR

- Emulates 2716-27010 eproms.
- 256K to 1024K memory available.
- Battery backed up, auto emulate on power-up.
- Low-cost, pays for itself on first project.
- Free 19.2K Serial Communications Software

If Intelligence, Versatility, High Performance and Time Saving products are what you're looking for, make that move today and let GTEK® put one, two, three or all four of these products to work for you! Remember, other development products are available, —CALL!

DEVELOPMENT HARDWARE & SOFTWARE
P.O. Box 2310; Bay St. Louis, MS 39521-2310 U.S.A.
ORDER TOLL FREE 1-800-255-GTEK (4835) FAX: 1-601-467-0935
MS & Technical Support 1-601-467-8048



12-Volt Regulator: Boost or Buck?

When I got to the 12-volt section of the power supply, the configuration was not as straightforward as the other sections. In fact, it was almost easier to contemplate going back to a low-dropout S-terminal series-pass regulator than to build what seemed to be required here. The trouble with producing 12 volts from a 10V-17V source is that it can't easily be done with a single-stage design. Let me explain.

Using a buck-converter design we can produce 12 volts from an applied voltage of approximately 13V to 45V (the upper limit of the 78S40). Similarly, using a boost-converter design we can produce 12 volts from an applied voltage of approximately 5V to 12V (a typical 12-volt output boost-converter circuit is illustrated in Figure 6). Unfortunately, since our input supply voltage can range from 10V to 17V, no one circuit adequately fits the bill.

Just for the sake of experimentation, I did build a buck-buck-boost compound regulator just to see if the results were worth the effort. Using a circuit similar to Figure 3, I made a 3.5-amp, 5-volt output buck regulator. Then, with a circuit similar to Figure 5, I configured an 8-volt-to-12-volt boost regulator. Surprisingly, the compound configuration worked, but it seemed less efficient than I would have liked and it produced more electrical noise than the other circuits; undoubtedly because of greater PC board real estate and radiating components.

The results of my empirical cutting-and-pasting are in Photo 1.

A Flyback Regulator to the Rescue

As the others around the office here can verify, I had considered forgetting this whole project idea because I hadn't arrived at a "neat," cost-effective solution to the problem. The Photo 1 prototype, while workable, was hardly something I cared to make more than one of. It was at that point that I came across the Linear Technology Corporation LT1070 integrated switching regulator and its suggested use in flyback converter designs.

The LT1070, block diagrammed in Figure 7, is a fixed-frequency current-mode switching regulator. It operates from 3V to 60V and can be used to produce the same buck, boost, and buck-boost circuits previously described as well as a flyback converter. The major advantage of the LT1070, unlike the 78S40 used at this current, is that both a 5-amp transistor switch and a 0.02-ohm current-sensing resistor are internal to the LT1070 chip. This not only reduces heatsink and board space requirements but also eliminates the radiated noise from connections between these components. The shorter the wires in a switching regulator design the less electrical noise.

The flyback converter circuit is very straightforward: The LT1070 closes the switch on the primary winding causing the transformer core to

store energy in its magnetic field. A resistor-capacitor-diode "snubber" network is inserted across the primary to reduce switching transients. No current flows in the secondary windings because the diodes are reverse-biased at this time. When the switch is turned off, the magnetic field collapses and induces a voltage into the secondary windings. Given the reverse-polarity of the collapsing field, the diodes become forward-biased and the energy is transferred to the outputs. (Note: It is important not to confuse this flyback pulse transformer with a sinusoidal AC transformer half-wave converter design. In all cases the secondary windings use Schottky diodes and the energy is transferred in square-wave pulses, much the same way as the previously described buck converters. Use of "standard" silicon rectifiers will cause excessive power dissipation and low output voltage.)

The PE-65108 pulse transformer has two primary and three secondary windings. Given the turns ratio, this circuit can work at either 6 volts with a parallel-wired primary, or at 12 volts with a series-wired primary as shown. There are three secondary windings. Two can be used for 12-volt outputs while the third is designed for 5 volts.

The system regulation is controlled by feedback from the 5-volt supply section. The 5-volt secondary output uses a resistor divider network to compare the 5-volt output to a 1.24-volt reference in the LT1070. If the output voltage starts to drop, the pulse-

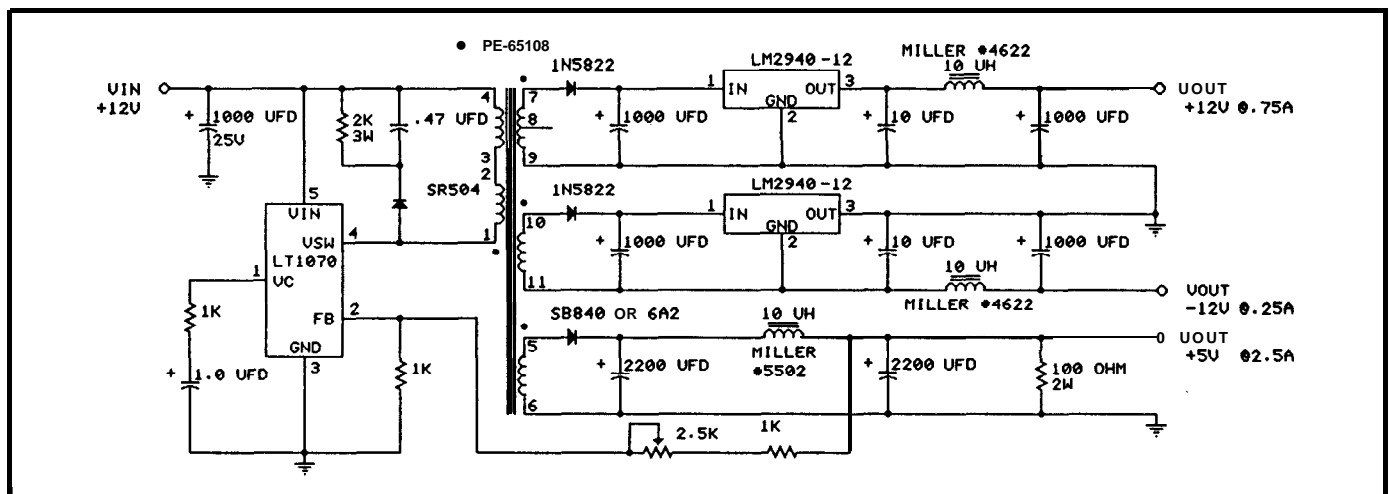


Figure 8—The final LT1070-based design uses fewer components than the equivalent 78S40-based design.

FREE CATALOG

RS-232C INTERFACE AND MONITORING EQUIPMENT CATALOG



Order direct from manufacturer TODAY and SAVE!

WRITE or CALL for YOUR FREE B&B ELECTRONICS CATALOG TODAY!

Pages and pages of photographs and illustrated, descriptive text for B&B's complete line of RS-232 converters, RS-422 converters, current loop converters, adapters, break-out boxes, data switches, data splitters, short haul modems, surge protectors, and much, much more. Most products meet FCC15J.

Your RS-232 needs for quality, service and competitive prices will be more than met by B&B ELECTRONICS. Manufacturer to you; no middleman!

Money-back guarantee! Same-day shipment! One-year warranty on products! Technical support is readily available.

Terms: Visa, MC, cash orders postpaid, P.O.'s from qualified firms accepted. IL residents add 6 1/4% sales tax.

RUSH MY NEW FREE CATALOG

Name _____
 Company _____
 Street _____
 City _____ State _____ Zip _____

B&B electronics
 MANUFACTURING COMPANY

4032C Baker Road • P.O. 1040

Ottawa, IL 61350

Phone: 81 S-434-0846

Circle No. 105 on Readers Service Card

width-modulated switching regulator just lengthens Ton a bit for each cycle to compensate and vice versa.

With the 5-volt secondary regulating properly (and not overloaded) approximately 14V-16V is induced in each 12-volt secondary winding. A pair of special low-dropout-voltage three-terminal linear regulators are used to create a regulated 12-volt output. LM2940-12 1-amp, low-drop out regulators require only 0.6 volts across them ($V_{out}-V_{in}$) instead of the 3 volts typical of 7805-type devices. The LM2940-12 is only available as a positive voltage regulator, but with isolated windings, each secondary can be configured as an isolated 12-volt output. One is simply inverted to appear as a negative 12-volt output.

The Proof is in the Pudding

One final concern is radiated EMI. To minimize this electrical noise and inductive pickup, the divider resistors and the components between pin 1 and pin 3 should be mounted very close to the LT1070. A single-point grounding system should be employed with the wires routed for least radiation (black magic again). All three secondary outputs have post-regulator LC filters to eliminate spikes. You'll note that the divider feedback network is connected on the "quiet" side of the LC filter so that output noise does not unduly influence regulator stability.

This flyback regulator seemed to work quite well and noise was generally less than 200 mV peak-to-peak on any output on my prototype. I would expect it to improve on a production PC board. With a 12-volt input I was able to obtain 2.5 amps at +5 volts, 0.75 amp at +12 volts, and about 0.25 amp at -12 volts. If I raised the input voltage to 17 volts, more output current was available from all three outputs proportionally. Conversely, at a given input voltage, the 5-volt output current could be raised if the 12-volt output currents were lowered and vice versa. There seemed to be a minimum 5-volt current necessary for regulation, however, and a 100-ohm resistor was added to make sure it was always

there (and should be removed in a fixed-use application).

As the input voltage was lowered to 10 volts, the available output current was also reduced. Conservatively, it was still 1.5 amps at +5V and a 100 mA at ±12V. I tested this to the extreme and found that the board still regulated down to about a 6.5-volt input (of course if I ever had to depend on that, I'd probably have worse problems elsewhere). If you use this design for currents lower than I specified, then you may want to use an alternative regulator. Both the LT1071 and LT1072 have lower current ratings and costs.

The completed power supply circuit, shown in Figure 8 and Photo 2, used considerably fewer components and less board space than the previous three designs to produce the same relative power output. □

Sources

J.W. Miller

Division Bell Industries
 19070 Reyes Ave.

Rancho Dominguez, CA 90224
 (213) 537-5200

linear Technology
 1630 McCarthy Blvd.
 Milpitas, CA 95035
 (408) 432-1900

Pulse Engineering, Inc.
 P.O. Box 12235
 7250 Convoy Court
 San Diego, CA 92112
 (619) 268-2400

IRS

201 Very Useful

202 Moderately Useful

203 Not Useful

Circuit Cellar Books

Circuit Cellar INK writers often refer to previous Ciarcia's Circuit Cellar articles. These past articles are available in book form from Circuit Cellar Inc., 4 Park St., Suite 12, Vernon, CT 06066.

Ciarcia's Circuit Cellar Volume I covers articles in BYTE from September 1977 through November 1978. Volume II covers December 1978 through June 1980. Volume III covers July 1980 through December 1981. Volume IV covers January 1982 through June 1983. Volume V covers July 1983 through December 1984. Volume VI covers January 1985 through June 1986.

VISIBLE INK

Letters to the INK Research Staff Answers; Clear and Simple

Tell me Why

I want to thank you again for the fine work you are doing to disseminate electronic leading-edge technology and encouraging us to look deeply into this field and its mushrooming future.

I have a few questions for you:

1. Concerning HAL-4, why are the signals to IC8 (74HC373) scrambled on input and then descrambled on output?

2. The original schematics show the IC4 (8255) CS hooked over into IC7 (74LS138), which supposedly is not there until the full system is defined. The same situation exists for IC5. Why is this? How should it be hooked up?

3. At a previous job, I worked on a parallel processing system which had three CPUs running in parallel, and used a memory scheme which included "shared" RAM, which could be accessed as read/write by any or all of the CPUs, as well as "private" RAM which could only be accessed by its associated CPU. I was wondering how one would go about setting up a circuit layout so as to be able to do this with microprocessors.

4. I am having a bit of difficulty finding information as to what to do to establish a CMOS/TTL interface. Can you give me some hints here?

5. Where can one get information as to what sort of current demands different devices will incur, so as to be able to total up a system's power usage?

6. What is a good way of interrogating a battery-supplied power setup so as to detect a "battery low" condition?

Allan R. Summers
Pasadena, TX

Dear Allan,

Thanks for the kind words on *OUT* projects! We try to strike a balance between "leading edge" and "practical" projects; letters like yours tell us that we are on the right track.

The scrambled lines on Hal's IC8 make sense if you look at the printed circuit board. Assigning the bits in *that* order made the traces fit on a double-sided board! Since this depends on the particular board, you'll find different layouts on all the projects; we often leave the choice of bits up to the guy who does the PC board layout.

That schematic showing the combinations of 8031 systems was essentially correct, but you have to take the colored backgrounds with a grain of salt. Basically, some of the gates needed for various "subsystems" wound up in the wrong colors simply because gerrymandering in all the peninsulas and tributaries would have complicated things a little too much. What it boils down to is that, if a circuit looks like it needs a connection, it does!

There were a few minor errors in the schematics; if you're particularly interested in the details you should sign onto the Circuit Cellar BBS and get the latest information.

The trick to having both local and shared memory is that you must have buffers and gates to provide separate data paths for each CPU, with separate address logic on both sides. If you take a look at the *ImageWise* schematics, you'll see how the video data bus (going to the ADC and DAC chips) is separated from the 8031 data bus (going to the EPROM and address latch). Trace down the addresses through the multiplexers to see how the 8031 can be doing something while the RAMs are handling video data.

There are several ways to convert from CMOS to TTL, ranging from ICs dedicated to the job all the way down to balancing wire circuits you build on the fly. It depends on how many lines you need to convert and how many units you're building; a one-off project can use circuitry that makes no sense in a real production unit.

Rather than give you the answer, we'll describe the process. There are two conditions you have to cover: a CMOS low level must draw enough current from the TTL gate to ensure that it "sees" a zero, while a CMOS high level must supply enough current for the TTL back leakage to shut off the gate. Therequisite currents and voltages differ for the various flavors of TTL and CMOS logic, as well as for individual gates within each family, so there are no hard-and-fast rules.

One rule of thumb is that you should not have a single gate driving both CMOS and TTL loads; stick with one or the other. This simplifies building the converters because you don't have to worry about the effects of drawing nonstandard currents from the gate on the rest of the system.

The data books that give you enough information to build level converters also tell you about the maximum and average current requirements for the complete chip. Mail order parts sources usually advertise the National and TZ data books; take a look at the *Digi-Key* and *Jameco* catalogs for the specifics. Make sure you get the Application Notes books, too, because they have lots of recommendations and hints for handy circuits.

As you might expect, defecting a low-battery condition requires an analog-to-digital converter. Fortunately, you only need one bit, so it isn't too hard to build, since something like an

LM311 comparator that compares a reference against the battery voltage fills the bill. When the battery falls below the reference voltage, the LM311 output changes state (either high to low or low to high, depending on your circuitry) and your CPU starts saving things.

The trick is that you've got to be able to correlate the battery voltage with remaining charge. The voltage varies with ambient temperature, as does the available charge, so this isn't quite as simple as it sounds. We don't recall the part number offhand, but we think National Semiconductor makes a battery-voltage monitor IC that handles some of the grisly details. Once you get the data books you can take a walk through the "building blocks" section and find something useful.

You can get rapid answers to questions like these by signing onto the Circuit Cellar BBS. There are a bunch of really competent folks hanging out there who can probably answer any question you can dream up. Best of all, the whole group can suggest approaches you haven't thought of and describe all the details based on real experience.

A Problem of Power

I have a car restoration shop. My problem is that my CAD system (AT clone running at 10 MHz) sends out beeps and stops when we use our plasma cutter, which has no HF to start its arc.

Our telephone system, a genuine Bell setup, lights up all six buttons and won't ring when our HF piggyback arc stabilizer runs when we're using the heliarc welder.

Is the fix as simple as adding ground wires? Or should I build a screen room around the welder/cutter?

How about this as a project for your magazine—"Clean Power in an Ugly World"?

Robert J. Schumann
Kansas City, MO

Dear Robert,

Every now and again we get a letter that reminds us of just how odd things get out in the real world.. .

The hash in your AT and phone system is probably coming through the power lines. If could be radiated (we bet you don't have any background music playing while you're welding, do you?), but we think the place to start is with the line cords.

Any electric arc will generate energy across the entire electromagnetic spectrum, quite literally from DC to daylight in the case of a welder. Naturally enough, the more power in the arc, the more power shows up as interference. What you need is a filter to remove the objectionable frequencies while letting the arc burn normally. From our experience with EMI (electromagnetic interference) generated by computer systems, a ferrite filter is the way to go.

Ferrite filters are made up of a finely divided iron compound with a high resistance to electrical current. A current-carrying wire passing through a ferrite slug induces a current in the

ferrite, but the ferrite's resistance dissipates the energy very well. The resistance increases with frequency, so the slug forms a quite effective low-pass filter.

One catch is that the ferrite slug must be able to handle the induced current without saturating. In the case of your welders and cutters, that's going to take a pretty big chunk! Rather than paying real money for this project, fake a trip to the local junk yard and scavenge the yokes from a couple of TV sets—the older, the better. If you've never rummaged around in a TV before, what you're looking for is the deflection hardware around the neck of the picture tube. Along with all the coils is a big hunk of black ferrite, which is just what you want. You may have to break the tube to get the yoke off, but that's why you do this trick in the junk yard instead of Sears.

Do be careful, though, because the implosion resulting from shattering the picture tube can blow glass all over the neighborhood. Wear a face shield and gloves. The safest method is to put the neck of the tube in a plastic bag and rap it with a screwdriver. This presumes, of course, that the junk yard has no further interest in the tube!

With a few yokes in hand, simply string the power lines through them. If the power lines go directly to a junction box on the wall, run the cables on the secondary side through the yokes. The key point is that the ferrite slug should form a continuous path around the current carrying wire, with one wire per yoke. Don't bother wrapping the wire around the yoke, because one "turn" is enough; more turns simply builds a step-down transformer with a single-turn shorted secondary.

You should see an immediate improvement, but if not, try moving the yokes closer to the workpiece. Because the arcs are the source of the hash, the less wire carrying the current, the less interference will be induced elsewhere in your building.

As an alternative, try puffing a yoke or two on the power lines leading to the AT and phone system. After all, it doesn't matter where you filter the hash out as long as it doesn't get to the circuitry. You will have to filter all of the power lines leading to the AT; don't forget the printer and modem!

We've been thinking of doing an article on RFI control and your letter has pushed us over the edge. It'll probably show up around the end of 1989, simply because we've got so much other stuff to do between now and then.

IRS

204 Very Useful
205 Moderately Useful
206 Not Useful

In Visible INK, the Circuit Cellar Research Staff answers micro-computing questions from the readership. The representative questions are published each month as space permits. Send your inquiries to:

INK Research Staff
c/o Circuit Cellar INK
Box 772
Vernon, CT 06066

All letters and photos become the property of CCINK and cannot be returned.

PRODUCT REVIEWS:

The Next Generation

Circuit Cellar INK sells out and enjoys it!

It's not often that **one** is witness to a major event in human history, but if you're reading this, you're not only a witness, but an eyewitness! Yes, you see before you an occurrence of epoch-changing proportions. Mere human inventions pale in significance before this. Polio vaccines, atomic energy production, and those little plastic things that slowly slither down your walls all fade away when compared to what you hold before you.

What am I talking about? Why the Circuit Cellar INK Benchmarking, Product Evaluation, and Junk-Food Consumption Testing Facility.

Now, I know we said that you wouldn't see any "me-too" reviews or evaluations in Circuit Cellar INK, but hey, times change. First, we saw what was being passed off as product reviews by other magazines and realized that no publication in this end of the universe was better qualified to write technical, hard-hitting reviews than we were. Second, we were getting miffed that companies weren't sending neat toys (free of charge) to our offices. Third, it seemed like a good way to get paid for breaking stuff.

No Dweebs Here

At Circuit Cellar INK, when we decide to do something, we don't go in for half-measures. Before we opened the CCINK **BPE&JFCTF**, we performed an exhaustive study of what "the other guys" were doing. What did we find? Wimpy software benchmarks and effete instrumentation, that's what! Where were the calisthenics for individual instructions, the particle accelerators, the blast craters? They were not to be found, at least not until now.

We have assembled the only testing facility of its type in the free world. On the hardware side, we put together a state-of-the-art laboratory filled with expensive gadgets, heavy industrial machinery, and scary electronic stuff. For wringing out the software, we contracted with a little-known hacker/genius to write assembly code that's so compactly written, so immune to optimization, and so thorough in its exercising of undocumented features, that we're not sure what the hell it really does. Finally, and most importantly, we've assembled the strangest group of gonzo testing weenies ever to don lab coats. These guys live to trash computers, and subsist on Chinese take-out, **Chee•tos**, and Jolt soda. Is our crew qualified, or what?

let the Games Begin

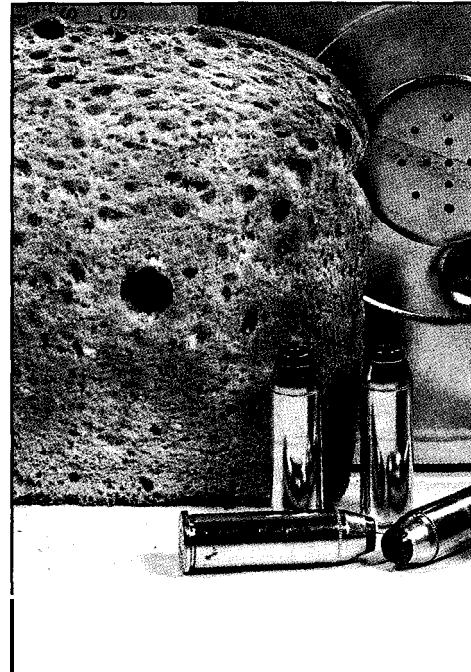
As we began looking through the literature on evaluating systems, we worked our way through the Whetstones, Dhrystones, Savages, Livermore Loops, and many more. We found that none of them really told volume purchase influencers what they needed to know about the system at hand. We thought about presenting all of the tests in an **easy-to-read** chart format so that readers could form their own opinions based on the information presented.

Then we came to our senses and realized that what readers really want is to be led by the nose, bludgeoned by a single sledgehammer factoid that they can point to in self-defense when things fall apart. If it's a flattering number that companies can use in advertising to justify their high prices, all the better.

With all of the above in mind, we formed our "Magic Number Search Committee." We knew that the most important piece of the puzzle was the name. After hours of research, pages of dictionary work, and several intense rounds of Balderdash, we found The Word. Henceforth, when systems of any type are compared, they will be compared in our devastatingly elegant unit of measure, the Mopoke. The range of Mopoke is from -53 to 177, using a logarithmic scale. Using the logarithmic scale makes for snappier graphics and often makes many mediocre products look far better than they truly are.

Inside the Mopoke

Where most benchmarking programs content themselves with simple performance measurements, you must



understand that the Mopoke is a unit for measuring far more than mere performance. In deciding to use the Mopoke, with all its subtle complexities, we acknowledge that most users are interested in more than how fast the processor can idle, or how many bits can be blasted to the disk per second. No, most users really want, and Circuit Cellar INK alone is now able to offer, an objective evaluation of the true gestalt of the system.

We start, of course, with performance. The first stage of Mopoke evaluation is to individually exercise every single instruction possible in a given system. Each instruction is performed, with no setup, overhead, environment, or operator intervention, for one million iterations. The

time for each individual iteration (timed and recorded by a special nearly noninterventionist external CCINK BPE&JFCTF clock that is linked by special optical links to the Atomic Clock at the National Bureau of Standards) is logged, and the results are averaged using a geometric mean. Next, polar Fourier transforms are performed on the total data set at S-degree intervals. These results, along with the geometric mean from the first evaluation, become fodder for the Mopoke cannon.

Next, we fearlessly dive into the human factors morass. Our dedicated staff painstakingly measures keyboard feel (key travel, key cap size, depth at roll-over, angle of attack, height of the little bumps on the home keys), display ergonomics (dot size, dot pitch, glare, aspect ratio, jitter, swim, bugaloo, accessibility of controls, versatility of inputs, and how hard it is to sneak the thing home to use **with your VCR**), and main unit construction (footprint, depth, height, hat size, presence of nifty LED displays, decibel level of the fan, pitch of the fan, aesthetic considerations). We then go **where no** review has gone before and quantify what can only be called "Ego Appeal Factor" (EAF). The EAF takes into account important issues such as: Will this system make your boss jealous? Does the color of the system complement your power ties? Do the noises of start-up guarantee that everyone in the office knows **when** you fire that sucker up? We run the results from all of these scientific tests through a weighted averaging system, where weights are determined using a complex system based on the researcher's biorhythm, the hourly exchange rate between the U.S. dollar and the Greek drachma, and the **Solunar** Table published in *Field and Stream*.

We don't overlook reliability in the Mopoke, either. We are the only publication that runs each and every test unit through both a steel annealing oven and a cryogenic life-extension chamber to check for continued operation at temperature extremes. We also operate each system in booth #7 at the "Tans for the Memories" tanningsalon and at the bottom of a washtub filled with Evian water to test for susceptibility to damage from humidity extremes. Our line-noise isolation test includes operating the system on the same line as a heliarc welder, and injecting a signal taken from side B of Def Leppard's most recent single into the test AC line.

As thorough as are these tests, we recognized that most failures involve hard disks. To stress-check hard disks of portable computers, we bolt the little monsters to the main oscillating mixer down at "Merle's Paint'N'Plaster World." Desktop units are subjected to a **patent-pending** procedure we like to call the "Extreme Prejudice Test." Suffice it to say that this rigorous examination of Winchester technology involves expensive disk drives, high-velocity ammunition, and a very low "Pass" ratio.

Now, most magazines would stop right there, but we're committed to press on. We know that most users are far more afraid of their system's manufacturer failing than of the system itself failing. So, in an action unprecedented in computer journalism, we rate the reliability of the manufacturer. First, we use the standard ratings based on information from Standard & Poor, Dun & Bradstreet, and the men's room attendant at the New York Stock Exchange. Next, we test based on "look and feel" issues such as color (or presence) of the CEO's hair, number of company executives wearing brown shoes with blue suits, glossiness of the annual report, and quantity of shrimp served at the company's COMDEX party. Finally, we take a hard look at marketing expertise including public relations budget (did they bribe us?), advertising budget (do they advertise with us?), and use of inappropriate celebrities, water fowl, or hors d'oeuvres in their ad campaign.

As with the other categories of data, company reliability information is not left to peacefully ferment. No, we run the raw numbers through a data colander unmatched in subtlety and precision. The results are factored with numbers taken from deep between the lines of the *Wall Street Journal*, *New York Times*, and *Daily Racing Form*. When we're through, you're presented with the one number that can indicate beyond a shadow of a doubt whether you should place your trust in a company by purchasing its product. Several Wall Street heavies have come to us begging for our numbers, but we have steadfastly refused. You see, we value the welfare of our readers far more than we value the paltry few million dollars offered for our secrets. Knowing that you will be able to make major purchasing decisions based upon what you read here is worth more to us than miles of yachts or buckets of caviar.

So now you know the why and the wherefore of our review process. As you turn the page, do so with the proper reverence, for you are truly taking part in a new era: The Age of MOPOKE. □

With our powerful new benchmarking tools in place, we decided that the first review should be comprehensive, a detailed look at a broad cross-section of powerful computational tools. For years, **limp-wristed** computer journals have been telling you that there's no real way to compare different system types on an even and fair basis. Poppycock! With the Mopoke at our disposal, we can readily compare Apples and oranges, **DECs** and **Wangs**, and **IBMs** and **Banana Juniors**.

Knowing that our readers are nothing if not diverse, we chose four very different computing solutions to compare in our first review. For the volume purchase influencer, we picked the GeneriComp 286, a **plain-vanilla PC/ AT** clone. The more traditionally minded among you should be cheered by the inclusion of the **Googolplexx 100**, an **S-100 mainframe** system. For the aficionado of portable computing we picked the most powerful computer, in the smallest package we could find—the **Hewlett-Packard HP-41CX**. Finally, just to prove that there's no place the Mopoke can't go, we included a simple, dedicated system, known and beloved by many as an essential tool for modern living. We're talking about the **Proctor-Silex TPB-5342 Dual-slot toaster**.

There you have it, the lineup for the first iteration of the most **comprehensive and impressive product evaluation** system in the history of mankind. Now let's move on to the tests.. .

The Hardware

The GeneriComp 286: If you're looking for the most risk-averse choice in office productivity solutions, you'll be **hard-pressed** to find a more conservative computer. The safety-conscious **6-MHz** top speed, the haven of an **Industry Standard Architecture (ISA)** bus, and the soothing beige

Supercharged Worry Munchers

Circuit Cellar INK looks at four diverse applications solutions

color unite to make this computer a "safe" option in many environments. We got this baby with 1 MB RAM, EGA, one 1.2-MB floppy drive, a **40-MB** hard disk, **101-key** keyboard, one parallel port, and one serial port. A copy of GeneriComp MS-DOS version 4.0 came with the system, along with a handsomely photocopied Users's Manual. If **CYA** is the order of the day at your office, the **Generi-Comp** deserves strong consideration.

The Googolplexx 100: In days of yore, when men were men and computers could heat the south wing of your house, Googolplexx became the computer marquee of choice among those truly "in the know." The famed Googolplexx One has been updated through **the years**, with the latest batch of revisions bringing us to **the Googolplexx 100**. As with all computers of this ilk, the 100 comes out of the box pretty much useless. We stood around and gaped at the 43-slot backplane, the **382-watt** power supply, and the variable-pitch **1.5-horsepower cooling**

fan. When we came back to earth and realized that testing would be hard with no components, we ordered a **6-MHz Z80 CPU**, 1 MB of RAM (**bank-switchable** in **64-KB** chunks), a quad serial port board, a parallel port board with cache, a **DEC VT220** terminal, two **8-inch** quad-density floppy disk drives, and a **40-MB** hard disk. We also decided to get a five-foot-high equipment rack in laboratory blue to keep everything corralled. For software, we stuck with **CP/M** version 2.2. Documentation included 47 separate User's Guides, Technical Guides, Application Guides, and Schematic Supplements. No overall index or bookshelf was included.

The HP-41CX: If **cranking away** at numbers while you're bouncing around in the back of a Jeep is part of your job description, then a portable computer may well be high on your wish list. We chose the **HP-41CX** as an example of a portable computer that makes sacrifices in user interface in order to concentrate on the portability side of the portable computer equation. The **41CX** packs 8 KB of memory into a package less than **1/20** the size of the GeneriComp. The HP comes with a one-line display, 39-key keyboard, and business-like dark brown case. It has a built-in clock/calendar and four proprietary expansion slots. We chose to get the handy leatherette carrying case (belt loop included) for better portability.

The 41CX comes with an operating system and fully functional software. Documentation is in two useful volumes.

Proctor-Silex TPB-5342: Who hasn't experienced the warm feeling of waking up to hot chocolate and a couple of pieces of crunchy buttered toast? When we were trying to decide which system to pick to **round** out our review, this sturdy chrome beauty was the nostalgic unanimous choice. The hardware in-



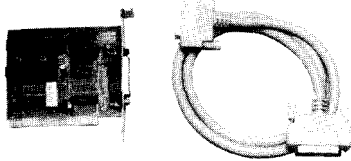
JDR Microdevices

- 30 day money back guarantee
- 1 year warranty on all products
- Toll-free technical support

New! Modular Programming System

FROM MODULAR CIRCUIT TECHNOLOGY

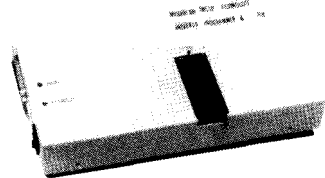
This integrated system is ideal for developers—it easily expands as your needs grow! All the modules use a common host adaptor card so you need just one slot to program EPROMS, PROMS, PALS and more!



Host Adaptor Card \$29.95

- A universal interface for all the programming modules
- User-selectable programmable addresses prevents addressing conflicts
- Includes a high quality molded cable

MCT-MAC



Universal Module \$499.99

- Programs EPROMS, EEPROMS, PALS, bi-polar PROMS, 8748 & 8751 series devices.
- Programs 16V8 & 20V8 GALS (gallium arsenide) from LATTICE, NS, SGS
- Tests TTL, CMOS, Dynamic & Static RAMS
- Load disk, save disk, edit, blank check, program, auto read master, verify and compare
- Textool socket accepts .3" to .6" wide IC's from 8-40 pins

MCT-MUP

EPROM Module \$7 19.95

- Programs 24-32 pin EPROMS, CMOS EPROMS and EEPROMS from 16K to 1024K
- HEX to OBJ converter
- Auto, blank check/program/verify
- VPP selectable 5, 12.5, 12.75, 13, 21 & 25 volts
- Normal, intelligent, interactive, and quick pulse programming algorithm

MCT-MEP

MCT-MEP-4-EPROM Programmer	\$169.95
MCT-MEP-8 R-EPROM Programmer	\$259.95
MCT-MEP-16-EPROM Programmer	\$499.95

PAL Module \$249.95

- Programs MMI, NS, TI20 & TI24 pm devices
- Blank check, program, auto, read master, verify and security fuse blow

MCT-MPL

IPAL Programming development software

MCT-MPL-SOFT	\$99.95
--------------	---------

Order toll free 800-538-5000



JDR MICRODEVICES

2233 BRANHAM LANE, SAN JOSE, CA 95124

LOCAL: (408) 866-6200 FAX (408) 559-0250 TELEX 171-110

RETAIL STORE: 1256 S. BASCOM AVE., SAN JOSE, CA

HOURS: MON.-FRI. 9-7, SAT. 9-5, SUN. 12-4 (408) 947-8881

Terms: Minimum order \$10.00. For shipping and handling include \$2.50 for ground and \$3.50 for air. Orders over 1 lb and foreign orders may require additional shipping charges—please contact our sales department for the amount. CA residents must include applicable sales tax. Prices are subject to change without notice. We are not responsible for typographical errors. We reserve the right to limit quantities and to substitute manufacturer. All merchandise subject to prior sales. A full copy of our terms is available upon request. Items pictured may only be representative.

COPYRIGHT 1989 JDR MICRODEVICES CONTINENTAL US AND CANADA

cludes a smart rotary toasting duration knob, two separate cook sequence initialization switches, two full-width slots, a complete set of state-of-the-art radiant energy coils, a pair of time-actuated target product egress elevators, and a handsome chrome case.

The TPB-5342 has achieved an admirable level of user friendliness. The software was completely transparent to all testers. Given the dedicated attempt the company made at providing a complete system solution which requires remarkably little user training, the saddle-stitched documentation was completely adequate.

The Tests Begin.. .

The first part of our testing procedure is an examination of the execution times for instructions. As we worked our way through this section, we made some startling discoveries.

The GeneriComp computer performed pretty much as you would expect. I mean, everyone in North America has seen at least 93 reviews of similar machines, right? Anyway, since detailed information on each niggling detail is not what the Mopoke's all about, suffice it to say that after yawning through every one of the "Friday the 13th," "Nightmare on Elm Street," and "Halloween" movies while the 80286 trotted through its paces, we were finally able to dump the raw data into the Mopoke's gaping maw. We let it quietly ferment while we went on to the Googolplexx.

When we started on the Googolplexx, some of our engineers got all misty-eyed thinking about the "old days" and wanted to install toggle switches on the front panel so they could enter all of the instructions by hand. Then we ran into our first dilemma with this system. When exercising the instructions, should we test the mainframe and the terminal, or just the mainframe? Boy, did we ever scratch our heads over this one. We finally decided that the terminal couldn't really hurt, and might actually help the results, so in it went. By now, we were looking forward to the stark simplicity of the Hewlett-Packard. Little did we know.. .

Now, if you look back at our explanation of the Mopoke, you'll see something about a million iterations of each instruction. That's no big problem with a big counter, and a nice keyboard to help you out. The HP-41CX has a fairly good counter, but have you ever really looked at the keyboard? We'll say more about it in the human factors section, but you've got to know that those keys are tiny—and hard. We ended up taking more time than we had planned because we had to keep sending out for Dr. Scholl's pads (we found out they work just as well on fingers as on toes) and adhesive tape. Grown engineers were reduced to whimpering hulks as their index fingers took the brunt of the brutal testing procedures. It took dedication, but our chests swelled with pride when we were finally able to shove the HP results through the Mopoke's statistical wood chipper. Since we finished with the 41CX at about 4:30, we were all looking forward to the Proctor-Silex.

The basic procedure for testing the toaster was simple: Insert two pieces of bread, set brown level, start toasting, and time until finished. It wasn't until we sat down to order the testing material that we realized there was a problem. Let's say that the average loaf of bread contains 25 slices. Four loaves take up about one cubic foot. If you're following, you see that we have 100 slices per cubic foot. Each iteration of the test requires two slices (one for each slot), so we need 2,000,000 pieces of bread, occupying approximately 20,000 cubic feet. But wait! There are eight different brown-level settings on the toaster. Our chief lab technician decreed that each level was in fact a different instruction, meaning that we really need 16,000,000 slices of Wonder Bread. That many slices will take up approximately 160,000 cubic feet. We're talking about filling up a block of average houses with toast, here. Most other labs would have given up at this point, but not us. The order was placed, the trucks rolled in, and the toasting commenced. We all became jam and jelly experts, and our publisher has a great idea for what to give away with new subscriptions.

Circle No. 115 on Reader Service Card
Dealers Circle No. 116

The Touchie-Feelie Stuff

When we started looking at human factors for the GeneriComp computer, we all started feeling underdressed. This is, after all, a machine that pretty much demands a pin-striped suit and wing-tip shoes. The box itself is rather austere, with sharp comers and lines, a staid tandem ear, and no little LED panels telling you how fast it's going. The keyboard is a standard 101-key affair with a soft, mushy feel, as if they had used Kraft Miniature Marshmallows as return springs. Most of the staff rated it over the 41CX, but well below the VT220.

The display is a standard VGA monitor of obscure heritage. It's bright, sits on top of the computer, and can show you a DOS prompt in 256 colors. Wow. Then there's the all-important Ego Appeal Factor. The GeneriComp gets middle-of-the-road marks on this score, because there are just so many like it out there. Sure to peg you as a mid-level manager in most firms, the 286 screams for a spreadsheet to be locked on its display. It is the K-car of computers: It's solid and reliable, but won't get you tons of style points.

The Googolplex 100 is not the taut, trim box of the GeneriComp. The rack would crush most desks, and Sasquatch has a smaller footprint. Actually, even if the rack would fit on your desk, you probably wouldn't want it there since the fan sounds like an A-10 'Warhog' attack jet and has enough force to blow your briefcase across the room. The Googolplex will use lots of power, but your utility bills shouldn't go up much because the heat generated by the unit will let you lower your furnace thermostat.

The VT220 impressed most of our lab staff. Good keyboard, nice "hi-tech" look, graphics that are fine for assembly language programming.. it took top marks from our expert crew.

We gave the Googolplex high marks in Ego Appeal Factor. There are several neat lights among the several front panels, everyone on the block will know when you turn it on, and you'll need a bigger office to store it. If you wear the right accessories (horn-rim glasses, pocket protector, white

socks), the company will find a way to work "Engineer," "Programmer," or "Technical Wizard" into your job title after you arrive with this baby.

The HP-41CX was a bit of a problem. The keyboard is lousy for touch-typing, and the screen makes any flight simulator program a real challenge. Fortunately the bad stuff ends there.

The 41CX has the smallest footprint of any system in this group. The sleek dark case will look right at home on the most thoroughly Italian-designer-inspired desktop.

Of course, the 41CX makes its case on portability. To be even-handed, we tried the other systems for similar portability. The HP-41CX arrived in a genuine leatherette case with a handy belt loop. It does, indeed, fit on a standard belt, so became our baseline. The next system to go onto the belt was the GeneriComp. After the test subject pulled his pants back up, we gave this one low portability marks. The Googolplex was even less portable. As a matter of fact, we couldn't even move it by attaching it to everyone's belts. The Proctor-Silex came closest to the HP in portability. It (and the belt) stayed up, but the cord tended to limit traveling range somewhat.

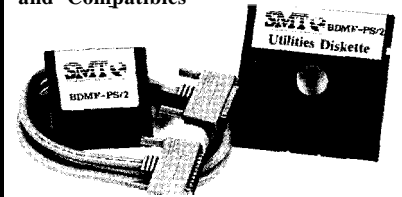
Apart from its lack of true portability, the Proctor-Silex had soon worked its way to the top of our human factors lists. Take the keyboard far away. Why? Because you don't need it! What genius! The display, too, is a masterpiece of understated elegance. No other system allows you to simply look in the top of the case for an instant update on the progress of processing.

The controls are ergonomically correct and easy to use. When you talk about intuitive, user-friendly computing, you have to start with the Proctor-Silex. Plus, despite its small footprint, the high-quality chrome finish helps the TPB-5342 make a big statement on any executive desk.

The big statement is why the Proctor-Silex scored highly in the EAF. There's certainly no mistaking this little gem for a Mac or PS/2. As a matter of fact, there's no mistaking the kind of statement you'll make by putting this system on your desk next to your Day-Timer.

THE INTERCHANGE™

Bi-directional Data Migration Facility for IBM PS/2, AT, PC, PORTABLE and Compatibles

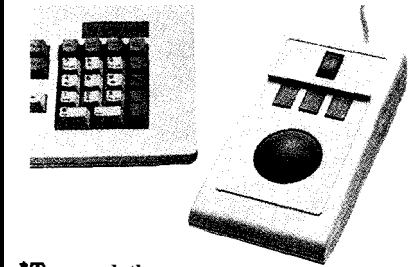


Features:

- *Parallel port to parallel port.
 - *Economical method of file transfer.
 - *Bi-Directional file transfer easily achieved.
 - *Supports all PS/2 systems (Models 30, 50, 60, and 80).
 - *Supports IBM PC, XT, AT, Portable and 100% compatibles.
 - *Supports 3 1/2 inch and 5 1/4 inch disk transfers.
 - *Supports hard disk transfers.
 - *Supports RAMdisk file transfers.
 - *The SMT 3 Year Warranty.
- ONLY \$39.95

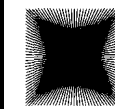
FastTrap™

The pointing device of the future is here!



- *Two and three axis pointing capability.
- *High resolution trackball for X and Y axis input.
- *High resolution fingerwheel for Z axis input.
- *Use with IBM® PC's, XT's, AT's and compatibles.
- *Three input buttons.
- *Full hardware emulation of Microsoft® Mouse.
- *Standard RS-232 serial interface.
- *Includes graphics drivers and menu generator.
- *Easy installation.
- *1 year warranty.
- *Made in U.S.A.

ONLY \$149.00



LTS/C Corp.

167 North Limestone Street
Lexington, Kentucky 40507
Tel: (606) 233-4156

Orders (800) 872 - 7279
Data (606) 252-8968 [3/12/2400 8-N-11
VISA, Mastercard, Discover Card,
TeleCheck

Circle No. 128 on Reader Service Card

More Mopoke

Well, there's plenty more detail to give you, but that's not what the Mopoke is all about, now is it? Sure, we could go on about our search through the financial section for data on company reliability, or even make your hair stand on end with tales about the adventures we had with the "Extreme Prejudice" hard disk test, with a sidebar on the ballistic properties of .44 Magnum hollow-point ammo. But we won't muddy the water with details. No, we'll boldly go where no computer magazine has gone before and cut to the..

Surprise Ending

When we gathered close around the Multiprocessor Mopoke Engine to see what the results would be, the anticipation was palpable. When the results finally came out, you could have knocked us over with a feather. There, in stark black and white, was the winner of our first group review: The

Proctor-Silex TPB-5342 Dual-Slot Toaster.

The TPB-5342 scored an amazing 102.6984 Mopokes. In looking back over the raw data, we're not sure why it scored so highly, but we suspect the high ratings in Human Factors, Product Reliability, and Company Reliability overshadowed the toaster's rather mundane performance rating.

The HP-41CX came in second in our tests, with a 96.4205 Mopoke score. The HP far surpassed the other systems in portability, making up for a keyboard that was suboptimal for most touch-typing applications. A strong second-place showing in Human Factors helped, too.

Third place in our lineup goes to the Googolplex 100. It seems that Ego Appeal Factor was the major consideration in helping the Googolplex attain 73.9917 Mopokes. It is certainly a system that anyone here would be proud to own. In addition to EAF, the Googolplex came out on top in general performance. As any WordStar fan will tell you, a 6-MHz

Z80 is one cranking little processor.

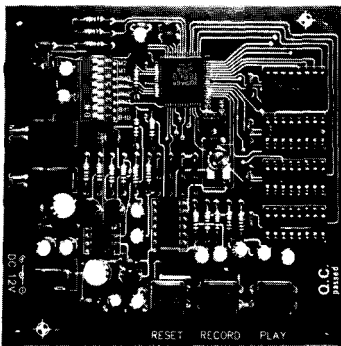
Every shootout has to have a loser, and in this case it's the GeneriComp 286. The buttoned-down demeanor of the AT clone was good for only 34.9628 Mopokes. The Genericomp's best finish was in performance, where it squeaked out a second-place standing over the 41CX. After that, an unbroken string of thirds and fourths contributed to the last-place finish.

While we were surprised by the Mopoke rankings, it's obvious that good ergonomics, a strong design, and serious use of RISC principles are the system design wave of the future. Volume purchase influencers looking for a safe, forward-looking choice should heed the message in this solution-based comparison.

Now, we don't want to make decisions for you, but the lesson is clear: If an appliance is what you want, a toaster is hard to beat. □

IRS

207 Very Useful
208 Moderately Useful
209 Not Useful



— DIGITAL VOICE MODULE —

- Low cost
- Selectable banks
- 4 Sampling rate
- Super quality
- IWamp
- DRAM operation

DVM-1 — \$49 (without RAM)

— PASSIVE INFRARED DETECTOR —

- Used In alarm system, moving detection
- Super sensitive
- Very reliable
- Exchangeable lens
- Analog Pulse Count

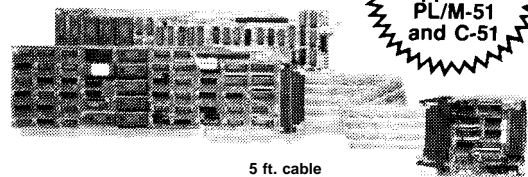
RK4000PCA — \$59

MING Engineering, Inc.

515 S. Palm Ave., #5
Alhambra, CA 91803
(818) 570-0058



"The Best 8051 Emulator"



5 ft. cable

NEW
Source Level
support for
PL/M-51
and C-51

8051

See EEM 88/89
Page D-1304

PC based emulators for the 8051 family

(8051/51FA/52/31/32/44/152/451/452/535/552 + CMOS + more

- PC plug in boards or RS-232 box
- Pull-down menus combined with Command-Driven User Interface
- Context sensitive help and On-Screen Editing of data
- 20 MHz real time emulation
- 128K emulation memory
- 48 bit wide, 16K deep trace buffer with loop counter
- Program Performance analyzer
- Powerful Macros with IF-ELSE, REPEAT, WHILE structures
- Source Level debug for PUM-51 and C-51
- Symbolic debugging with m-line assembler and disassembler
- Execution time counter
- Trace can be viewed during emulation!

PRICES: 32K Emulator for 8031 \$1790: 4K Trace \$1495'

CALL OR WRITE FOR FREE DEMO DISK!

'US only

Ask about our demo VIDEO!

NOHAU
CORPORATION

51 E. Campbell Avenue
Campbell, CA 95008
FAX (408) 378-7869
(408) 866-1820

Writing a Real-Time Operating System

by Jack Ganssle

Part 2

Memory Management and Applications for the HD64180

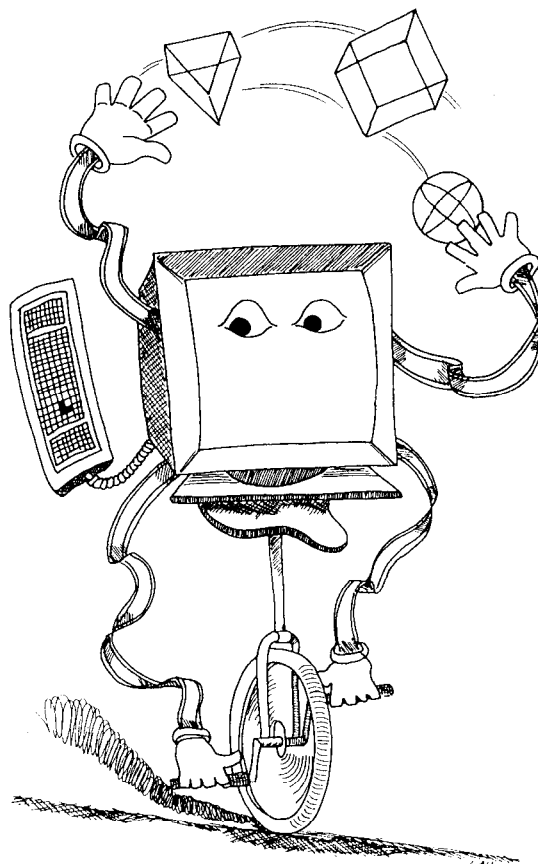
Last issue's installment about BCC180 RTOS discussed how the task structure works. This month we'll examine the memory management aspects of multitasking, and an example application.

Memory Management

The RTOS structure described so far is more than adequate for normal operations on a 280. However, the HD64180's most powerful enhancement is its internal MMU. A method of managing it is essential.

The MMU is both a blessing and a curse. It gives programs access to 1 MB of memory, 16 times the space available to the Z80. Programs previously running up against memory limitations can use the MMU to gain as much elbow room as needed.

The HD64180's hardware can handle a large address space, but how can a program with 16-bit addresses get to it? Unlike the 8088, there are no segments; you can't split data and code references (Zilog's new Z280 does give split addressing; a Z80 program can go to 128K with essentially no change in the code). The software must dynamically remap the MMU every time an access is needed outside of the current 64K logical map. This makes it virtually impossible to write a conventional linear program longer than 64K. A way of breaking the code into smaller logically complete sections is needed, along with a method to command the MMU to bring the proper sections of code into view as needed.



Fortunately a multitasking RTOS can automatically remap the MMU to give access to very large programs. Tasks are logically complete sections of code; since only one task can be active at a time, only the active task needs to be mapped into the current space at any time.

The BCC180 RTOS takes advantage of this by assigning each task to a separate map. The operating system automatically resets the MMU after every context switch, bringing the proper code into view.

BCC180 RTOS's memory map is configured as three separate banks. In keeping with standard HD64180 programming practice, Common 0 starts at location 0000H and runs to logical address 3FFFH. The Bank area starts

at 4000H and goes to 7FFFH. Common 1 runs from 8000H to the end of memory. This configuration gives the operating system and other utilities 16K of memory starting at 0000H. All tasks typically start at logical 4000H and can be as long as 16K each. Common 1 is the RAM data area.

Common 0 (the operating system and other goodies) and the RAM area are always visible in the logical map; they are never mapped out. The Bank area, where all tasks are stored, is the only section affected by remapping. It is usually remapped for each task to bring new ones into view.

With this memory layout, 16K is reserved for the RTOS and other important global routines; 32K is available for variables. The rest of memory, almost 1 MB, can be used for tasks. Sixteen kilobytes per task is almost always enough, since tasks by their nature should each be simple; it's better to have a lot of simple tasks than a few big complex ones.

In some applications, other memory configurations may be required. More RAM may be needed, or larger or smaller task sizes accommodated. The only thing sacred about the map outlined is that the RTOS start in low memory and never be mapped out, and that RAM be available somewhere and never remapped. The value `CBAR_DATA` in the code can be changed at will, within these constraints, to suit other requirements.

Since the HD64180 MMU has a resolution of 12 bits, this approach implies that each task will use at least

4K of memory space. Does this mean that a program with four short tasks will need 16K (possibly wasting most of it)?

Fortunately not! The TCB contains two values that affect mapping. A BBR (Bank Base Register) is stored for each task, so the MMU can be remapped to this address. However, every task's start address is also saved. If all tasks started at 4000H logical (for example), a start address would not be needed.

By saving a start address, many tasks can be located within one map. In the extreme case, the entire program can use only 64K—set all BBR values the same, store each task at a different address, and set the proper start addresses within the TCB (using **OS-DEFINE**). Thus, BCC180 RTOS gives the user all possibilities; mapping can be defined in virtually any fashion, or it can be completely ignored simply by setting all BBR values to 0.

A word of caution: Since tasks can be assigned to maps, no task should ever call or access an address in another task. That address may not exist when the task is running! If you choose to disregard this advice, be sure you're a multitasking expert, and be sure the tasks are reentrant.

Performance

By its nature, a real-time operating system must be fast! It is charged with making sure external events are monitored and serviced on time; presumably those events won't wait for a slow computer. For this reason, the RTOS must be coded in assembly language and optimized for speed. Portability is the rage today, so some operating systems are written in C or even Pascal, a practice that I find offensive.

Context-switch times are difficult to measure since they're a function of the number of tasks in the TCB (the RTOS must search the TCB on every task), priority assignments, and typical task states. However, it is possible to find average times for average situations.

BCC180 RTOS, when handling 16

tasks requiring some sort of service, typically performs a context switch in about 800 microseconds. This means about 8% of the processor time is used to support the multitasking (since ticks occur every 10 ms).

The HD64180 is a reasonably fast processor while handling interrupts and for context switching. New architectures are now becoming available with built-in features enabling blindingly fast tasking. For example, the H16, another Hitachi product, includes 16 identical register banks. Much of the work of a context switch can be done in a single instruction!

Another important criterion for determining the performance of any RTOS is to examine the code's size. Intel's iRMX can be huge, precluding its use in low-end applications.

Admittedly, it is very powerful, but that power is useless if it won't fit in the system's ROM. In these days of cheap memory, designers often elect to throw ROM at a problem, but that is not always possible. Hitachi's new **HD647180X**, a single-chip computer derivative of the HD64180, is limited to 16K of ROM-period. Especially in single-chip systems, memory must be conserved! BCC180 RTOS is quite small, requiring only a few hundred bytes.

Sample Application

We've described the operating system in detail. Now I'll present a simple multitasking program to show the use of the RTOS and the various system calls that request task service.

```

start: ld      sp,t0sp
      call    inituart      ; initialize the UART
      ld      hl,0
      ld      (counter),hl  ; zero counter
      rt_init                    ; initialize the os

      rt_define task1,t1sp,0,40,1; define task 1
      rt_define task2,t2sp,1,32,2; define task 2
      rt_define task3,t3sp,2,32,3; define task 3
      rt_define task4,t4sp,3,32,4; define task 4

      rt_run  1,5              ; start task 1
      rt_run  2,300           ; start task 2
      rt_run  3,1000          ; start task 3
      rt_run  4,50            ; start task 4

root:  jr      root
;
; inituart--init the UART to 9600 bps
inituart:
      ld      a,64h
      out0   uartinit        ; 8 data, 1 stop, no parity
      ld      a,baudrate
      out0   baudport        ; set data rate
      ret

; task 1--inc a counter

      aseg
      org    4000h
task1: ld      sp,t1sp
      ld      hl,(counter)
      inc    hl
      ld      (counter),hl
      rt_exit                    ; end task

; task 2--reset the counter to 0

```

Listing 1 -Example multitasking application (continued on page 32)

```

        org      5000h
        .phase   4000h
task2:  ld      sp,t2sp
        ld      hl,0
        ld      (counter),hl      ; zero counter
        rt_exit      ; end task
        .dephase
;
; task 3--cancel task 2 after a while
;
        org      6000h
        .phase   4000h
task3:  ld      sp,t3sp
        rt_cancel 2      ; cancel task 2
        rt_cancel 3      ; cancel ourselves
        rt_exit
        .dephase
;
; task 4--display the counter
;
        org      7000h
        .phase   4000h
task4:  ld      sp,t4sp
        ld      hl,(counter)      ; number to display
        call    hex4      ; dump it
        ld      c,0dh
        call    cout      ; send a cr
        ld      c,0ah
        call    cout      ; send an lf
        rt_exit
        .dephase

```

Listing 1 -(continued from page 31)

ing.

Listing 1 is a five-task program (remember, task 0, at `root`, exists by default). Task 0 does nothing; it runs an infinite loop. Task 1 increments a memory-based counter once every five tics. It runs at a high priority, established when the task was defined by the macro `RT DEFINE`.

[Editor's Note: Complete software from this article is available for downloading from the Circuit Cellar BBS or on Software On Disk #8. For downloading and ordering information see Page 70.1

Task 2 runs infrequently (once every 300 tics). It resets the counter to zero.

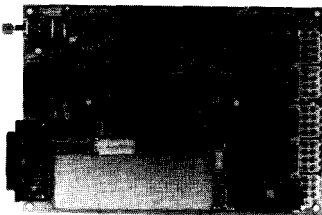
Task 3 cancels task 2 after 1000 tics, effectively killing off the resetting of the counter to zero. It also cancels itself, since once task 2 is killed, task 3 is done forever.

Task 4 prints the value of `COUNTER` twice a second (50 tics). (`HEX4` prints the number in HL in hex.1 This lets you see the effect of task interaction. The counter value will be generally increasing since task 1 runs often. Once every three seconds it is reset to zero

New, Improved!

SIBEC-II

The ideal solution for embedded control applications and stand-alone development.



- Intel 8052AH BASIC CPU
- Serial printer output and 5, 8 bit I/O ports
- 5 in.² prototyping area
- Memory: 8K RAM, expandable to 128K
- Power requirements: 5V.DC @ 300 ma. only
- PROM programmer; ZIF socket for 2764 or 27128 EPROM
- Interrupt handling capability
- Built to exacting standards and warranted
- Still only \$228.00 including documentation (quantity 1)

Inquire about our **PDK51** 8051-8052 product development kit for the IBM-PC/XT/AT: **\$595**.
Our **BXC51** 8051/8052 BASIC compiler: **\$295**.

Call now! 603-469-3232



Binary Technology, Inc.

Main Street • PO Box 67 • Meriden, NH 03770



Development Tools

PseudoSam Cross-assemblers \$50.00

PseudoMax Cross-simulators \$100.00

PseudoSid Cross-disassemblers \$100.00

PseudoPack Developer's Package \$200.00 (\$50.00 Savings)

POWERFUL

PseudoCode is pleased to announce the release of an extensive line of professional **cross-development** tools. Tools that speed development of microprocessor based products. Fast, sophisticated macro assemblers to generate your program code. **Versatile** simulators that allow testing and debugging of the program even before the **hardware** exists. **Easy** to use disassemblers to help recover lost source programs.

AFFORDABLE

Until now, powerful tools like these have been priced from 5 to 10 times Pseudocode's price. Putting these time saving tools out of reach of all but large corporate engineering departments.

BROAD RANGE OF SUPPORT

- PseudoCode currently has products for the following **microprocesso** families (with more in development):

Intel 8048	FCA 1802.05	Intel 8051	Intel 8096
Motorola 6800	Motorola 6801	Motorola 6811	Motorola 6805
Hitachi 6301	Motorola 6809	MOS Technology 6802	WDC 65C02
Rockwell 65C02	Intel 8080,85	Zilog Z80, NSC 800	Hitachi HD64180
Motorola 68000,8	Motorola 68010		

- To place an order call one of our dealers:

Programmer's Connection USA (800) 336-1166 INTL (216) 494-3781
KORE Inc. (616) 791-9333

PseudoCode

P.O. Box 1423

Newport News, VA 23601-0423

(804) 595-3703

by task 2 and the count resumes its incrementing. After 10 seconds, task3 cancels task 2, so the count is never again reset to zero.

Each of the tasks runs in its own memory map. They're small, so they do waste most of the address space allocated to them. However, this illustrates the use of the RTOS with memory management and the loading of tasks in different maps.

Every task is assigned to logical address 4000H, and occupies exactly 1000H bytes (the mapping resolution of the MMU). The BBR values increment from 0, so the physical addresses of the tasks are 4000H (task 1), 5000H (task 2), 6000H (task 3), and 7000H (task 4).

The tasks may *not* be defined at their physical addresses since internal jumps will then go to the indicated address and not the logical address, which is where the program really runs. Tasks cannot be defined at their logical addresses because then all of the tasks would be at the same address, causing assembly errors.

The solution is to use an ORG di-

rective to set the physical address of the task, and .PHASE and .DEPHASE to force the assembler to assume the code is really somewhere else in this case at the execution address (4000H). The code is loaded at the correct physical address but it is assembled at the **proper** logical spot.

Possible Enhancements

BCC180 RTOS is **more** a starting point than an end product. Many applications **may demand** more performance, more features, and greater versatility. One logical extension is a dispatcher. Instead of compiling operating system calls (via macros) into the application's code, it would be simple to issue a RST instruction with an argument to request RTOS service.

BCC180 RTOS makes all of RAM global to all of the tasks. RAM could instead be allocated to tasks and messages passed via mailboxes and semaphores. A more dramatic extension would let tasks idle while waiting for a message.

At the extreme, a command-line interpreter and disk system could be added, giving a multitasking form of CP/M. There seems little reason, though; most CP/M application programs won't run well in a multitasking environment.

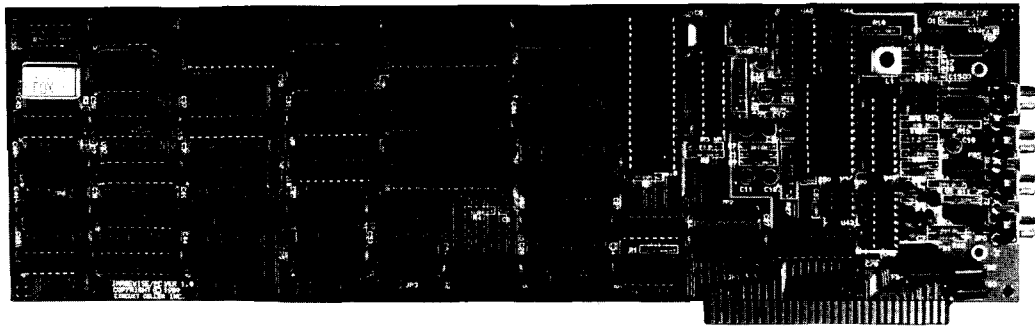
A Complete Environment

BCC180 RTOS provides a **complete environment for handling multitasking problems**. While it does not provide many of the functions interactive computer users are accustomed to, it is a solid base on which to build real-time multitasking applications. All of the multitasking and memory management is handled by the operating system and is mostly invisible to the user, making BCC180 RTOS a good place to begin exploring real-time multitasking. □

IRS

- 210 Very Useful
- 211 Moderately Useful
- 212 Not Useful

Micromint now makes affordable Video Digitizing even better with ImageWise/PC™



The company **that** made video digitizing **affordable** now makes affordable digitizing even better with **ImageWise/PC™**. Bring your reports, graphics, security system, or video application up to the new standard in cost-effective gray scale video digitizing with the new ImageWise/PC™.

- Digitize any NTSC, PAL, or SECAM video source!
- Up to 256x255 resolution with 256 level gray scale!
- True frame grabber - digitizes in 1/60 second!
- Digitizes 30 frames per second!
- Composite video output!
- Can display digitized pictures on EGA or VGA!
- Advanced overlay and split screen capabilities!
- Digitized images compatible with paint and desktop publishing programs!
- Modify, enhance, display, and print images using sophisticated ZIP Software Included with every ImageWise/PC!

ImageWise/PC™
an affordable
\$795.00

Order by: TEL: (203) 871-6170
FAX: (203) 872-2204
TELEX: 643331



MICROMINT, INC. - 4 Park St., Vernon, CT 06066

ImageWise/PC-The Digitizing Continues

Part 3

by Ed Nisley

Topping it off with Software

Although solder is the favorite programming language around here, the ImageWise/PC is held together with a large chunk of firmware. Now that you are acquainted with the hardware design, we can investigate the code making it all possible.

I'll start by describing the 8031 firmware controlling the ImageWise/PC card, then the routines that pass information and commands between the card and the PC, and finally the PC utility programs needed to run the card.

Firm Control

The fundamental reason we used an 8031 microcontroller on the ImageWise/PC card is that it simplifies the design without restricting flexibility. Using firmware instead of hardware means a change to the control logic is accomplished with a program rather than a soldering iron.

For example, the 8254 Programmable Interval Timer (U21) generates timing for the rest of the circuitry from a 10-MHz clock. A video sync pulse starts when the OUT0 pin of Counter 0 triggers Counter 1 and the duration of OUT1 sets the pulse duration. The firmware controls those two counters on the fly to create all the different horizontal and vertical pulses needed for each video frame.

Although a dedicated sync generator chip could produce those pulses, the firmware squeezes more tricks from the 8254. For example, the NTSC, PAL, and SECAM standards are just a bunch of different pulse timings. Genlock is largely a matter of repro-

gramming Counter 0's mode and routing the incoming sync pulses to its trigger input.

Best of all, adding features we didn't anticipate can be a simple matter of firmware! There are some changes that can't be handled in the code, but many require just a few (hundred) more lines of assembler.

Making a Big Sync

The 8031 firmware has only two main tasks: create video sync pulses and communicate with the PC. The code is structured to ensure that sync generation has absolute, unquestioned priority over anything else that may be going on. Failing to put a sync pulse out at the right moment will cause painfully obvious jitter on the monitor...and there's nothing more annoying than an obvious glitch.

Fortunately, although the sync code must have high priority, the 8254

provides some relief from critical timing. A change made to an 8254 register affects the next output pulse rather than the current one, so the firmware can load a new value any time before the current interval expires. This requires some careful thought to make sure that the right values go into the right registers at the right time, but overall the process is easy to understand.

Figure 1a shows the sync pulses occurring at the end of the first video field, with times in microseconds. The firmware must load the 8254 registers at the indicated points to produce the vertical sync signal. Because the 8254 uses a 10-MHz clock, the firmware will load 318 (decimal) to produce the interval shown as 31.8 microseconds.

Even in this short section of video signal you can see that most intervals have no timing changes. During the 244 visible scan lines there are no timing changes at all. Although the

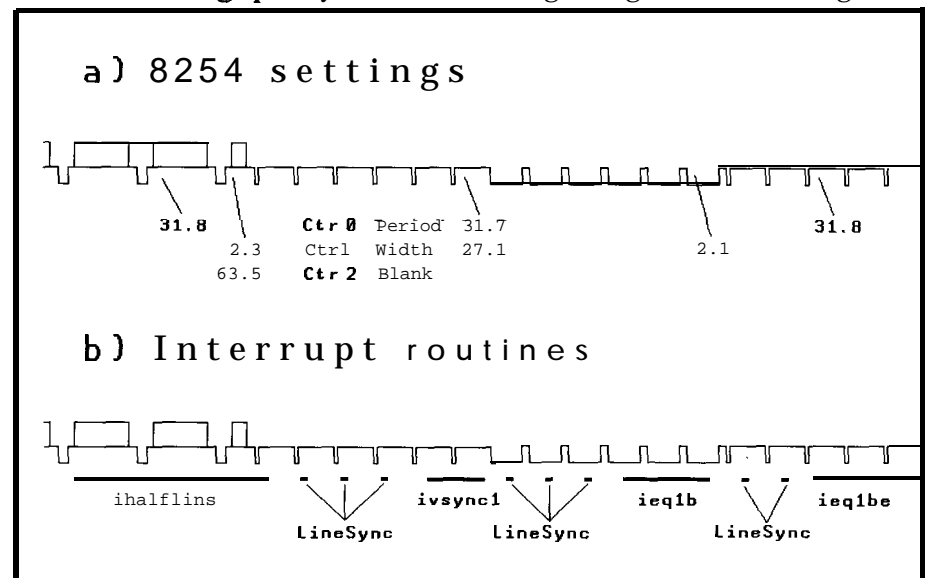


Figure 1 — The 8031's primary responsibility is sync generation.

sync generation is critical, **much** of the time there is nothing to do!

Each sync pulse causes an INTO interrupt which vectors into the main interrupt handler shown in Listing 1. The variable REPCOUNT counts down to zero, at which point the STATE variable selects the appropriate sync routine to handle the next timing change. The overhead during the active video lines is around 12 μ s, which is under 20% of the total time available.

When REPCOUNT becomes zero,

LINESYNC saves the processor registers, swaps to Register Bank 1, and uses STATE to select an entry in the jump table pointed to by R6 /R7. All sync routines end with a jump to ENDSYNC to restore the registers and return to the interrupted routine.

The labels in Figure 1b correspond to the sync routines shown in Listing 2. For example, IHALFLINE gets control on the next-to-last full active video line to set up for the half line at the end of the first field. LINESYNC

handles the interrupt without further branching when REPCOUNT is nonzero.

Because saving and restoring registers takes about 30 μ s in addition to the 12 μ s needed to decrement and test REPCOUNT, most sync routines get control one sync before the 8254 registers must be loaded. This ensures that they have ample time for any instructions required to get ready for the 8254 updates. For example, the first two lines in IHALFLINE simply wait for the next sync pulse on the INTO input (which is EQUated to SYNCINT, which is much easier to remember when you're writing code).

Sync Choices

Although it may sound as though the differences between various sync standards are just values loaded in 8254 registers, the situation is a little more complicated than that. It turns out that a different set of firmware routines is needed for each sync standard, as well as for internal and genlocked sync in each standard.

The reason is simple: the firmware must guarantee that the 8254 registers will be loaded at the correct time. As you see in Listing 2 and Figure 1b, some of the interrupt routines span several INTO pulses because the timing does not allow saving and restoring registers. Because the precise timing during the vertical sync pulses depends on the sync standard, the code **must** be optimized to suit.

The firmware starts up in internal NTSC mode. Selecting a different sync standard, changing the timing, or genlocking must be handled by a PC program similar to SETIMWPC.

Because control from the PC is so critical to actually using the ImageWise/PC, it's time to describe how that process works.

Registered Data

In the previous installment I mentioned that the PC communicates with the ImageWise/PC firmware using a set of registers holding timing values, counts, and voltage settings. This idea is similar to the 6845 controller used in IBM PC CGA video boards, but, un-

```

;-----
; Video sync interrupt handler
; Basic timing is set by 100-ns clock for 8254 timer
; At least 5 microseconds have elapsed up to this point
; Interrupt source is controlled by "genlock" output bit,
; but the state table base address in R6/R7 determines the
; routine that's in control...

LineSync      PROC                                ; entry from vector code
               PUBLIC   LineSync
;- tick state rep count and decide what to do
               DEC      repcount                ; tick counter
               XCH     A,repcount              ; is it zero yet?
               JNZ     goback                  ; not zero, continue
;- need to handle a state change
               XCH     A,repcount              ; restore A
               PUSH    PSW                    ; save bystanders
               PUSH    ACC
               PUSH    B
               PUSH    DPH
               PUSH    DPL
               SETB    RSO                    ; select reg band 1
               MOV     A,state                 ; get state offset
               RL      A                      ; get two-byte offset
               MOV     DPL,R6                 ; set up state code base
               MOV     DPH,R7                 ; set up state code base
               JMP     @A+DPTR
;- if no state change, return to normal code
goback        EQU     $
               XCH     A,repcount              ; restore A and count
               RETI

;- common return for state routines
EndSync       EQU     $
               PUBLIC   EndSync
               POP     DPL                    ; restore caller's regs
               POP     DPH
               POP     B
               POP     ACC
               POP     PSW
               RETI
LineSync      ENDP

```

listing 1 — Video sync interrupt handler

like that chip, the **ImageWise/PC** uses firmware to simulate the registers.

Listing 3 shows the firmware's main loop. If I hadn't already described how sync generation works, you might think there was something tragically wrong: there are only three instructions! Because sync is entirely interrupt-driven, the main loop needs no calls to that code.

The 8031 input bit `INDEXLOADED` is set to zero whenever the PC writes a value to the `INDEX` port. When the firmware isn't executing the sync generation code it checks that bit every four microseconds. While that may seem excessive, remember that the code has literally nothing else to do.

When `INDEXLOADED` becomes zero, the firmware branches to the code shown in Listing 4. The first steps read the `INDEX` and `DATAREG` values (from addresses `PCINDEX` and `PCDATAIN`) into 8031 registers.

Bits 0 through 6 of the `INDEX` value specify which firmware register is of interest, while bit 7 is set to write the register and cleared to read it. because the firmware uses only registers 0 through 2F hex, a range check on the `INDEX` value rejects incorrect register numbers. There is no way to flag an error, but because the interface routines use constant register addresses this does not pose a problem.

Most of the registers are mapped into values in the 8031 Internal RAM, so reading or writing them is just a matter of copying data between `DATAREG` and RAM. Most of the Internal RAM values are used by the sync interrupt routines, which move the values into the hardware at the correct times.

Some registers collect several bits from different I/O ports or status flags. The Control Register, addressed by `INDEX = 0`, is an example of such a register. It controls functions ranging from enabling the video output to grabbing a new image, each selected by a single bit value.

The code at `PC_wctrls` executes when the PC writes a byte into the Control Register. It writes a copy of the entire byte into Internal RAM at `PC_controls` and then moves the individual bits into their proper places.

The Control Register video en-

```

;- field 1 half line of video
; gets control at next-to-last full video line
; merges with the equalizing routine to ensure correct timing
, captures half-line sync to ensure correct timing
; captures first equalizing sync to ensure correct blanking
; Half line of video is +1 count

ihalfline PROC

half0 JNB syncint,half0; wait for line sync
      CLR syncint ; suppress interrupt
      M W DPTR,#I82540
      MOV A,IntLineH ; figure half line
      RRC A
      M W B,A
      MOV A,IntLineL
      RRC A
      ADD A,#1 ; add one count to half line
      MOVX @DPTR,A
      MOV A,B
      ADDC A,#0
      MOVX @DPTR,A
      INC frames ; tick frame counter
half1 JNB syncint,half1; wait for half-line sync
      CLR syncint ; suppress interrupt

;- set up for equalizing pulses
; 6 equalizing pulses are +1 count

      M W DPTR,#I82541
      M W A,IntEqualL ; equalizing sync duration
      MOVX @DPTR,A
      M W A,IntEqualH
      MWX @DPTR,A
      M W DPTR,#I82542
      M W A,IntLineL ; blank whole line
      MOVX @DPTR,A
      M W A,IntLineH
      MOVX @DPTR,A
half2 JNB syncint,half2; wait for equalizing sync
      SETH disablevideo ; force blanking
      SETS vretace ; indicate vert retrace time
      CLR syncint ; suppress interrupt
      CLR frameflag ; indicate field 1 is done
      M W reccount,IntEqualN ; two syncs captured
      DEC reccount
      DEC reccount
      INC state
      LJMP EndSync

ihalfline ENDPROC

;- field 1 vertical sync
; gets control after 5th equalizing sync
; captures 6th sync to ensure correct timing
; vertical sync pulses are -1 count

ivsync1 PROC

vs1 JNB syncint,vs1 ; wait for 6th eq sync
     CLR syncint ; suppress interrupt
     MW DPTR,#I82540
     M W A,IntLineH ; figure half line

```

listing 2-NTSC sync interrupt routines (continued on page 37)

```

RRC      A
MOV      B,A
MOV      A,IntLineL
RRC      A
MOVX     @DPTR,A
MOV      A,B
MOVX     @DPTR,A
MOV      DPTR,#I82541
MOV      A,IntVSyncL ; vert sync pulse duration
MOVX     @DPTR,A
MOV      A,IntVSyncH
MOVX     @DPTR,A
MOV      reccount,IntVSyncN ; one sync caught
DEC      reccount
DEC      reccount
INC      state
LJMP     EndSync

ivsync1  ENDPROC

;- field 1 second equalizing pulse
; gets control after 4th vertical sync
; captures 5th sync to ensure that we get control correctly
; captures 6th sync to ensure correct timing
; The first 3 equalizing pulses are -1 count, remaining 4
; are +1 count

ieqlb    PROC

eqlb1    JNB      syncint,eqlb1; wait for 5th vert sync
          CLR      syncint ; suppress interrupt
eqlb2    JNB      syncint,eqlb2; wait for 6th vertical sync
          CLR      syncint ; suppress interrupt
          MOV      DPTR,#I82541
          MOV      A,IntEqualL ; equalizing sync duration
          MOVX     @DPTR,A
          MOV      A,IntEqualH
          MOVX     @DPTR,A
          MOV      reccount,IntEqualN ; two syncs caught
          DEC      reccount
          DEC      reccount
          DEC      reccount
          INC      state
          LJMP     EndSync

ieqlb    ENDPROC

;- transition to full horizontal line timing
; gets control after 3rd eq sync, sets period to +1 count
; captures 4th sync to ensure good timing
; captures 5th sync, changes to normal horizontal period
; captures 6th sync, merge with vertical blanking setup

ieqlbe   PROC

          MOV      DPTR,#I82540
          MOV      A,IntLineH ; figure half line
          RRC      A
          MOV      B,A
          MOV      A,IntLineL
          RRC      A
          ADD      A,#1 ; add one count to half line
          MOVX     @DPTR,A
          MOV      A,B

```

listing 2—(continued)

able bit provides an interesting example of the complexities that creep into something that “ought to be simple.” In principle, `PC_video` turns the video output on or off, so you might expect it to control `DISABLEVIDEO` directly.

In practice, the sync interrupt routines set `DISABLEVIDEO` whenever the output video must be turned off. That code checks `PC-video`, which is controlled by the PC, so that the video remains off when the PC wants to blank the screen. Unfortunately, if the video is enabled or disabled in the middle of the active video lines, many monitors develop heartburn because their sync detectors mistake that voltage transition for a vertical sync.

So the firmware sets or clears `PC_video` as commanded by the PC, then waits for the interrupt code to update `DISABLEVIDEO` during the vertical retrace. Although the routine could return control to the PC before `DISABLEVIDEO` matches `PC-video`, that might conflict with a subsequent attempt to capture an image. Rather than forcing the PC code to figure out when to capture an image, the firmware just spins until everything is OK.

The sync interrupt will turn `DISABLEVIDEO` off (thus enabling the signal) only when it finds `PC_video` on. This allows the PC interface code to wait for the first match between `PC_video` and `DISABLEVIDEO` with confidence that the interrupt code won't change the setting later.

The genlock control bit is another complex example. The action required depends on which sync standard is in use, whether the camera produces clean sync, and whether the video is already locked. In this case, the firmware can compare the `GENLOCK hardware output bit` (read as an input) with the desired state of `PC_genlock` to decide what to do. The `Capture-Lock` and `UnLock` routines handle all of the complexities, returning only when the deed is done.

The program on the PC must verify that the genlocking status bit in the firmware status register (`INDEX = 1`) matches the intended state in the control register. If genlocking failed, perhaps because there was no camera at-

How to get more from your IMAGEWISE

ZIP Utility Pack

New! \$39

CONVERT gray scale images for use with other programs. Choose among these popular file formats: PIW-PIF-PCX-ASC-WKS-TIF

ANALYZE images for quality control and research. Load image data into I-2-3, etc. and analyze according to your specific needs. Generate histograms, descriptive statistics, and pixel count at each gray level.

CREATE images with your spreadsheet -- use data, calculations, or equations to generate images for use in ZIP and view x-y-z interactions. Be creative and make shading patterns, frames, or surreal landscapes.

EDIT images and individual pixels with any VGA paint program. Turn the palette into 64 shades of gray for realistic viewing.

DISPLAY custom slide shows of video images, rotate, and create mirror images for transfer applications.

ZIP Image Processing

Improved! ZIP \$79

Controls the serial **ImageWise** Transmitter and Receiver. Holds 3 pictures for image processing and combining images. Sophisticated image processing functions for enhancing images and creating special effects. Advanced display techniques for **EGA/VGA**. Saves in PIW-MAC-PCX-TIF file formats. Supports dot matrix, inkjet, color, and laser printers. Now shipping version 2.7, updates available to registered owners.

New! ZIP8 \$79

Controls the new **ImageWise/PC**. Reads/writes PIF-PCX-TIF gray scale files. 256 gray levels. Autocalibration feature adjusts **IW/PC** for the best picture using available light. Contains all ZIP features.

HOGWARE COMPANY
470 BELLEVIEW
ST LOUIS MO 83119

To order ZIP products call:
(314) 982-7833

VISA/MC

ZIP requires 384K RAM, MSDOS 2.0 or higher. Display requires EGA or VGA. ZIP trademark of HOGWARE CO. IMAGEWISE trademark of Circuit Cellar Inc. 1-2-3 trademark of Lotus Development.

Circle No. 124 on Reader Service Card

```

                                ADDC      A, #0
                                MOVX     @DPTR, A
eqlbel      JNB      syncint, eqlbel      ; wait for 4th sync
                                CLR      syncint      ; suppress interrupt

eqlbe2     JNB      syncint, eqlbe2      ; wait for 5th sync
                                CLR      syncint      ; suppress interrupt
                                MOV      DPTR, #I82540
                                MOV      A, IntLineL      ; set full line
                                MOVX    @DPTR, A
                                MOV      A, IntLineH
eqlbe3     MOVX    @DPTR, A
                                JNB      syncint, eqlbe3      ; wait for 6th sync
                                CLR      syncint      ; suppress interrupt

;- field 1 vertical blanking

                                MOV      DPTR, #I82541
                                MOV      A, IntSyncL      ; normal sync duration
                                MOVX    @DPTR, A
                                MOV      A, IntSyncH
                                MOVX    @DPTR, A
                                MOV      reccount, IntVBlankN ; no syncs pending
                                INC      state
                                LJM     EndSync

ieqlbe     ENDPRCC

```

Listing 2--(continued)

```

; -----
; The Main Loop!

mainloop   PROC

                                JB      indexloaded, L?nop      ; 1 = index notloaded
                                CALL    DecodeCmd
L?nop      EQU      $
                                LJM     mainloop

```

Listing 3--The Main loop

```

; -----
; Command decoder
; Gets control when PC loads the index register
; Flips state of status bit when operation is finished
; (PCs may not be able to detect a few microsecond pulse every
; time...)

DecodeCmd  PROC
                                PUBLIC  DecodeCmd

;- extract the data from the PC interface hardware

                                MOV      DPTR, #PCindex      ; get index reg value

```

Listing 4--Image Wise/PC code for the PC interface (continued on page 39)

```

MOVX    A,@DPTR
MOV     PCregnum,A
MOV     DPTR,#PCdatain    ; get data
MOVX    A,@DPTR
MOV     PCregdata,A

;-----
; Handle registers than need special code

    >>>> bulky code omitted <<<<

;-----
; Handle register reads

    JB     ACC.7,regwrites
    ADD    A,#-30H        ; must be 0..2F
    JC     PC_badop
    MOV    A,PCregnum
    ADD    A,#PCregbase ; convert to int RAM addr
    MOV    R0,A
    MOV    PCregdata,@R0; set up return value
    JMP    PC-complete

;-----
; Handle register writes

regwrites    EQU    $
    CLR    ACC.7        ; turn off write flag
    MOV    PCregnum,A
    MOV    A,PCregnum
    ADD    A,#-30H        ; must be 0..2F
    JC     PC_badop
    MOV    A,PCregnum
    ADD    A,#PCregbase ; convert to int RAM addr
    MOV    R0,A
    MOV    @R0,PCregdata
    JMP    PC_complete

;-----
; Write PC Control Register

PC_wrctrls    EQU    $

;- video on/off
; The video actually switches at vertical sync times to
; give a prettier transition. We wait here until
; disablevideo is in the right state, which saves some
; ugly polling in the PC program. Waits only occur if
; the bit has changed!

    MOV    A,PC_controls; get old bits
    XRL    A,PCregdata ; identify changes
    CLR    EA          , hold off interrupts!
    MOV    PC_controls,A; and set changes for tests
    MOV    C,PC_video ; save changed flag
    MOV    PC_controls,PCregdata ; set new bits
    SETB   EA
    JNC    L?enabledone ; skip if no change
    JB     PC_video,L?waiton ; decide what to
                                ; wait for
L?waitoff    JNB    disablevideo,L?waitoff ; ... off
    SJMP   L?enabledone

```

listing 4-(continued)

tached, PC_genlock will still be 1 but PC_locked will be 0. The PC code **must** take appropriate action, which may involve screen messages, retries, or whatever.

Regardless of whether the PC wrote or read a firmware register, the firmware copies the current register contents into PCdataout so the PC can see what happened. Writing that port also deactivates INDEXLOADED in preparation for the next exchange.

The INDEXLOADED bit appears in the PC status register port, so the PC could test it to tell when the firmware has completed the request. However, the firmware is fast enough that it may be able to flip the bit off before the PC is ready to test it, so the PC can't tell whether the request was completed or the machinery is broken.

A second bit, called OPCOMPLETE, provides an unambiguous indication marking the end of each operation. The firmware flips that bit when it is finished so the PC test becomes a matter of detecting a change in the bit's state. If the bit does not change within a reasonable time, the PC may conclude that the firmware is out to lunch and start whatever recovery procedures are needed.

Having covered the firmware activity for a PC transaction, it's time to look on the PC side of the fence.

PC Communications

Listing 5 presents the two lowest-level routines to read and write firmware registers. The routines are similar so I'll walk through ReadReg to illustrate the main points. Because the PC code is in Microsoft C 5.10 and MASM while the firmware is written in Avocet AVMAC assembler, you need to keep several names for the same bit in mind at one time.

One bit of C trickery that may not be obvious is my use of a union to combine a bit field with an integer variable. I prefer to test status bits by name, rather than perform obscure arithmetic on integer variables, but the C library I/O routines handle only integers. The solution is to define HWSBITS as a bit field to get the individual bit names, then form a union

with an integer value to keep the library happy.

ReadReg starts off by reading the hardware status port to get the OP-COMplete bit. The port value is stored into the integer member of SBITS1 using the union notation SBITS1.I, which also sets the bit field members found in SBITS1.B because they occupy the same storage byte.

The next step is to write the register number into the index port. Because the high-order bit is off (compare this with WriteReg), the firmware will go through the gyrations described in the previous section, resulting in the current register value appearing in the data port.

The do-loop simply reads the hardware status port into the integer member of SBITS2. The loop test compares the OP-COMplete bit using the bit field members of SBITS1 and SBITS2. The result of the timeout test is saved in the global variable TimeOut, which the calling routine can test to determine if something untoward has occurred.

Regardless of how the loop terminates, ReadReg's return value is read from the data port. Because there are no bit fields involved, the value is a simple integer.

All of the ImageWise/PC support routines eventually call ReadReg and WriteReg to handle data transfers between the PC and the firmware. Two additional routines (called ReadAllRegs and WriteAllRegs) read and write all of the firmware registers in one shot to simplify saving or restoring a complete configuration.

Twiddling the Knobs

Back in the good old days, electronic equipment came with a real front panel. Changing an operating condition was a simple matter of twisting a knob or flipping a switch. There was a satisfying physical connection between the knobs and the functions with a lot of tactile feedback.

Sad to say, the times have changed. Apart from four RCA jacks on the back panel, there is little to reveal an ImageWise/PC in your system. Certainly there are no additional knobs

```

L?waiton      JB      disablevideo,L?waiton      ; ... on
L?enabledone  EQU      $
;-- simple bits
              MOV      C,PC_overlay
              MOV      enableoverlay,C
              MOV      C,PC_digital
              MOV      digital,C
;-- genlock
              JNB      PC_genlock,nolock
              JB       genlock,lset ; if already locked, skip
              CALL     CaptureLock
              SJMP     lset
nolock        EQU      $
              JNB      genlock,lset ; if already unlocked, skip
              CALL     UnLock
lset          EQU      $
;-- if continuous mode is off, force acquire off immediately
              JB       PC_auto,isauto
              CLR      acquire
isauto        EQU      $
;image acquisition
              JNB      PC_grab,noget
              CALL     GetVideo ; snag a new image
              CLR      PC_grab ; and simplify PC code...
              MOV      PCregdata,PC_controls; update grab bit
noget         EQU      $
;-- return
              JMP      PC_complete
;-----
; General collection point for errors
; We return an unreasonable value for most things...
PC_badop      EQU      $
              MOV      PCregdata,#OFFH
              JMP      PC_complete
;-----
; Standard completion
; PCregdata has the return value for reads and writes
; Writing the data out register resets the "loaded" input
PC_complete   EQW      $
              MOV      DPTR,#PCdataout
              MOV      A,PCregdata
              MWX      @DPTR,A ; clear "index loaded" flag
              CPL      opcomplete ; done = status bit flip
              RET
Decode&d      ENDP RCC

```

listing 4—(continued from page 39)

```

typedef struct {
    unsigned int  vertsync:1; /* 1 during vertical retraces */
    unsigned int  syncpulse:1; /* 0 during any sync */
    unsigned int  genlocked:1; /* 1 when hardware genlock active*/
    unsigned int  field1:1; /* 1 during fld 1, 0 during fld 2*/
    unsigned int  misc:2; /* unused bits */
    unsigned int  opcomplete:1; /* flips 0->1 and 1->0 after ops */
    unsigned int  indexloaded:1; /* wr index->1, 8031 data rd -> 0*/
} HWSBITS;

typedef union {
    unsigned int i;
    HWSBITS b;
} HWSTAT;

/*-----*/
/* Read a program register from the 8031*/

unsigned int ReadReg(unsigned int regnum) {

    HWSTAT sbits1,sbits2;
    time_t start,now;
    int retval;

    sbits1.i = inp(IOBase+CTL_STAT); /* get current status*/
    time(&start); /* mark starting time*/

    outp(IOBase+INDEX,regnum); /* set register number */

    do { /* stall until done */
    sbits2.i = inp(IOBase+CTL_STAT);
    } while ((sbits1.b.opcomplete == sbits2.b.opcomplete)
    && (!(TimedOut = (TIMEOUT < difftime(time(&now), start))));

    retval = inp(IOBase+DATAREG); /* snag current ctl bits */
    return retval;
}

/*-----*/
/* Write a program register into the 8031 */

void WriteReg(unsigned int regnum,unsigned int regvalue){

    KWSTAT sbits1,sbits2;
    time_t start,now;

    outp(IOBase+DATAREG,regvalue); /* set output value */
    time(&start); /* mark starting time*/

    sbits1.i = inp(IOBase+CTL_STAT); /* get current status*/
    outp(IOBase+INDEX,regnum+REG_WRITE); /* set index with write
    flag*/

    do { /* wait for cmd end */
    sbits2.i = inp(IOBase+CTL_STAT);
    } while ((sbits1.b.opcomplete == sbits2.b.opcomplete)
    && (!(TimedOut = (TIMEOUT < difftime(time(&now), start))));
    }
}

```

listing 5-PC code for the Image Wise/PC interface

INTROL CROSS DEVELOPMENT SYSTEMS

SAVE Development
and Debugging Time
of Embedded
Microprocessor Systems!

- INTROL-C Cross-Compilers
- INTROL-Modula-2 Cross-Compilers
- INTROL Relocating Macro Cross-Assemblers

COMPILER PACKAGES INCLUDE:
Compiler . Assembler . Linker
. Runtime library, including
a multi-tasking executive .
Support utilities . Full year's
maintenance

TARGETS SUPPORTED:
6301/03 • 6801/03 • 6809 •
68HC11 . 68000/08/10/12 •
68020/030/881/851 . 32000/
32/81/82

AVAILABLE FOR FOLLOWING
HOSTS: VAX and MicroVAX;
Apollo; SUN; Hewlett-Packard;
Macintosh; Gould Power-
Node; IBM-PC. XT, AT, and
compatibles

INTROL CROSS-DMLOPMENT
SYSTEMS are proven, accepted
and will save you time, money,
and effort with your develop-
ment. All INTROL products are
backed by full,
meaningful,
technical support.
CALL or WRITE
for facts NOW!



INTROL
CORPORATION

647 W. Virginia St.
Milwaukee, WI 53204
414/276-2937 FAX: 414/276-7026
Quality Software Since 1979

hanging off your PC and nothing in the way of tactile feedback. Fortunately, the PC does provide a full keyboard, a CRT display, and lots of computing power, so there is little excuse for an awkward user interface despite the lack of knobs.

The SETIMWPC program takes advantage of the PC's features to provide a convenient and legible front panel for the hidden ImageWise/PC card. I designed it to show all of the firmware interface registers, as well as control and status information, in one screen because I can never remember arcane command-line sequences. Figure 2 shows the screen layout, which differs slightly from the one you saw two issues ago. [Editor's Note: Complete executable software from this article is available for downloading from the Circuit Cellar BBS or on Software On Disk #8. For more information see page 70.1

SETIMWPC stores all the configuration information in a file called IMWPC.CFG. Whenever you start SETIMWPC, it automatically reads this file and sets the ImageWise/PC registers.

Getting Started

Because the ImageWise/PC has so many options, you may have some trouble figuring out where to begin. The firmware and software can cope with a wide variety of video signals but you have to know what to adjust for the right results.

When you install the ImageWise/PC board, you should make sure that all the terminating jumpers are installed. Although you can change these to adapt the board for different terminating systems, start out with known conditions. If you plan to use the video overlay circuitry, you must use the on-board terminators to prevent overshoot on the signals.

Connect a monitor to the digital video output on J3. If you have a second monitor available, you can connect it to J4 to watch the overlay output. While you don't need both, it does make setup a little easier. Of course, you must connect a camera to J1 if you plan to capture video images.

The monitors on J3 and J4 should

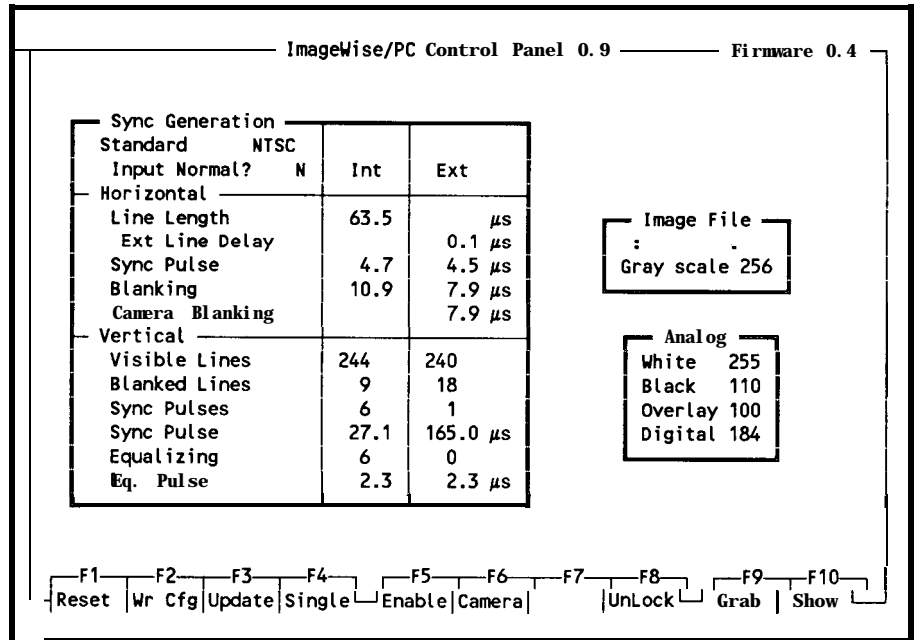


Figure 2— The Image Wise/PC's control panel allows the user complete control over most timing parameters.

be terminated to match the default voltage settings in IMWPC.CFG. Both TV monitors remain blank when you fire up the PC because the ImageWise/PC is reset until you explicitly start it. Run LOADIMW, which will read IMWPC.CFG and load the program from IMWPC.HEX if you have an ImageWise/PC with RAM program memory. When it releases the reset you will see a test pattern on J3 and camera video on J4. The board is now running in internal sync mode with overlays disabled.

There are two groups of timing information you may need to change: internal and external. Internal sync values (on the left side of the Sync Generation block in Figure 2) control the signals when the ImageWise/PC is producing internally generated sync. External sync values (on the right side) control the signals when the ImageWise/PC is genlocked to an external video source.

If your camera produces standard NTSC sync, you can use the default IMWPC.CFG because it specifies NTSC sync timings for the external source as well. Most modem cameras produce standard sync signals because all of the timings are derived using an NTSC sync chip; that's as perfect as you can get in the world of

consumer electronics.

I supply several CFG files to cover some of the more common cases. Take a look to see what's available; we may have covered your situation already. After you copy the appropriate file to IMWPC.CFG, you are on the air.

With the correct images on the monitors and the camera in focus, run SETIMPWC. It will read IMWPC.CFG, reload the registers with the defaults (again), and present you with a display similar to Figure 2.

The legend under F8 should read UnLock, indicating that the output sync on J3 is not locked to the incoming sync on J1. Press F8 to start genlocking; if the timings are correct and all is successful, the legend will change to Locked. If the ImageWise/PC firmware can't lock to the signal it will return an error which will appear as a message on the bottom line.

If your camera timings are merely close to the values in one of the IMWPC.CFG files, the firmware may think it is locked up but actually be off by a few lines. You will see the camera's vertical sync pulse roll through the image coming from J3. The cure is to adjust the number of visible lines in the External column until the rolling stops. Remember to press F3 after each change to write the new values

into the registers; press F8 to lock and unlock with the new values.

As a last resort you may need to get an oscilloscope and actually count the pulses during the vertical sync intervals. If the camera sync is particularly nonstandard, you should also measure the pulse widths. There may be some cameras that we can't handle, but so far we've been doing pretty well.

When you have everything Just Right, press F2 to create a new IMWPC.CFG. This will overwrite the old file, so make sure you have a copy stashed somewhere safe.

A Good Image

Press F9 to grab an image; make sure there is no file name in the Image File block so that the program doesn't create a disk file yet. The image on J3 should match the live video shown on J4 when you pressed the key. If you were not in genlock mode (F8 said UnLock) before grabbing the image, the signal on J3 probably jumped a bit as the firmware locked to the incom-

ing video. You can avoid that twitch by tapping F8 until its legend reads Locked before F9.

Now press F4 to start continuous acquisition. The firmware captures the first field of each frame; you will see the results as they are digitized. Remember that the PC code is not transferring the data out of the video buffer so there is no problem digitizing in real time even on slow PCs. This mode is handy for focusing your camera if you have only one monitor on J4.

If the White and Black settings in the Analog block are not matched to your camera, you will see washed-out highlights or flat black areas in the picture. With the firmware in continuous acquisition mode (F4 says Cont) you can change these values to match your camera. Black settings below a certain point (about 50-80) will have no effect because they are below the black level of the picture. Similarly, some cameras produce white levels above the 255 setting; those cameras are well out of spec!

Once those levels are set, press F2

again to write a new configuration file with the new values.

Now you're ready for overlays. Press F9 to grab an image (this will also turn off continuous mode), then press F6 to overlay the analog image atop the digital image. At any point where the analog voltage level exceeds the Overlay voltage, the digital image will show through. You may have to adjust the Overlay value in the Analog block to get a suitable level. Try waving your hand in front of the camera and watch what happens; with Overlay set correctly the digital image will show through your hand.

TV stations use a similar method to produce weather maps, although they use a particular blue color rather than just brightness. With the Image-Wise/PC, you can use a whiteboard as a window into a digital image and display your own weather maps. Just like the real TV stations, too, the background will peck through any other areas in the image with the right intensities!

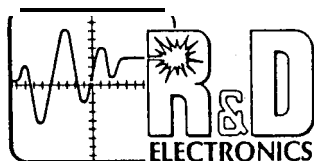
Press F6 again to overlay digital atop the analog. Now the digital image

R&D ELECTRONIC SURPLUS, INC.

Has been in business since 1946 selling NEW surplus electronics and electromechanical parts.

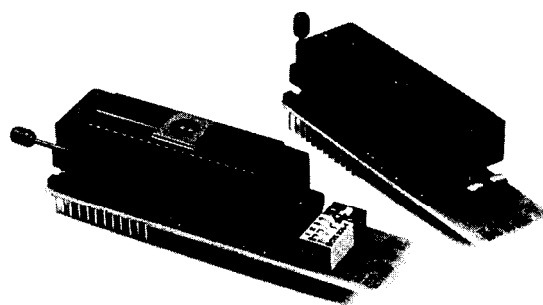
Call for our FREE 40 page catalogue detailing:

Batteries	LEDs
Cables	Lugs
Capacitors	MOVs
Clocks	Ni-Cads
Connectors	Power Devices
Digital Timer/Controllers	many Power Supplies
Diodes	Relays
Displays	SemiConductors galore
Enclosures	Stepper Motors & Driver ICs
Fans	Speakers
Filters	Many Switches
Heat-Shrinkable tubing	Telephones and components
Heatsinks	Transformers
Integrated Circuits ,	Zeners
Lamps & Lights	etc.



1224 Prospect Avenue
Playhouse Square
Cleveland, OH 44115

Telephone: (216) 621-1121 • Fax: (216) 621-8628



87C51 PROGRAMMER \$125.

Logical Systems brings you support for the Intel 87C51. The UPA87C51 programs this popular microcontroller on general purpose programmers that support the 2732A. With the UPA87C51 you can program the 8751 and 87C51 security bits and the 87C51 encryption array. Logical Systems, helping you get the most out of your programming equipment with our growing line of adapters. OEM inquiries welcome.

ADAPTER	PROGRAMS	PRICE
JPA8751	C8751, 8751H, AMD8753H, 8744	\$95.00
JPA87C51	C8751, 8751H, AMD8753H, 8744 87C51, 87C51FA	125.00
JPA63701V	Hitachi HD63701V0	65.00
JPA451N	Signetics SC87C451 (64 pin DIP)	125.00
JPA63701X	Hitachi HD63701X0 (64 pin shrink dip)	
	-L Low insertion force socket	95.00
	-Z Textool ZIF socket	149.00
JPA63701Y	Hitachi HD63701Y0 (64 pin shrink dip)	
	-L Low insertion force socket	95.00
	-Z Textool ZIF socket	149.00
JPA63705V	Hitachi HD63705V0	65.00

CALL (315) 478-0722 or FAX (315) 475-8460

LOGICAL SYSTEMS CORPORATION
P.O. Box 6184, Syracuse NY 13217-6184 USA, TLX 6715617 LOGS

MORE GOOD CODE... FAST!

Softaid's In-Circuit Emulators give you all the power and speed you need to develop microprocessor based products in realtime, increasing your productivity and saving you time and money.



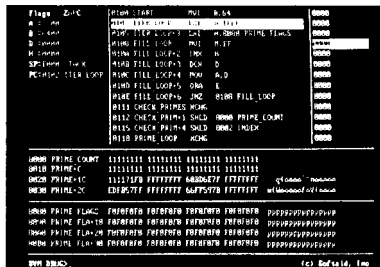
Emulators available for:

64180, Z80, Z180, 808818086,
80188/80186, 8085, V40/V50

Priced from \$595 to \$2995

FULL SCREEN DEBUGGING!

With the optional source level debugger, you get a real time, full screen debugging environment with pop-up windows and symbolic displays. Your source code and comments are displayed in a window that is automatically linked to the debugging session. This makes embedded system debugging FAST and EASY!



TIMELY TECHNICAL SUPPORT!

Our technical staff is ready to answer your questions. Give us a call to discuss your microprocessor development needs!

Complete information is also available on our BBS from 5 p.m. to 8 a.m. EST -- 301-964-8456.



8930 ROUTE 108
COLUMBIA, MD 21045
(301) 964-8455
(800) 433-8812

will show through wherever it exceeds the Overlay setting, so as you wave your hand it will disappear behind the brighter parts of the digital image. You will probably have to adjust Overlay until you find a suitable level. This mode is useful for digital titles and graphic overlays, although you must keep the ImageWise resolution in mind when you design your layouts.

Press F6 until the label reads Camera again; this disables the overlays and restores the camera image to J4.

Picture Files

Type a file name into the Image File block. If you have a big RAM disk you can save the files there; otherwise use a hard disk. Each image file uses from 30K to 80K bytes of disk space, depending on the file format and picture content.

The code can handle four different file formats, determined by the file extension and number of gray levels. PCX or PCC files (as in PICTURE.PCX) are stored in the 256- or 1 g-gray-scale format used by ZSoft Publisher's Paintbrush and PC Paintbrush. IMW files (as in PICTURE.IMW) will be stored in the serial ImageWise format if you select 64 gray levels. All other files are stored in 256-gray-level binary.

Pressing F9 or F10 when you have a file name filled in will save or show that file. Saving the same scene in a variety of different formats will show you the effect of 16, 64, and 256 gray levels on the image quality when you show the files in quick succession. The new 256-level files should win hands down!

On Your Own

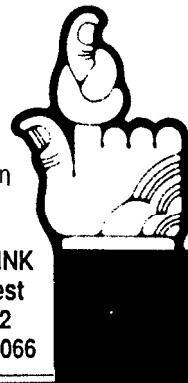
After our journey through the ImageWise/PC from design principles to software you should have a good idea how to put the thing to use. Now, what can you do with an inexpensive gray-scale digitizer?

IRS

2 13 Very Useful
214 Moderately Useful
215 Not Useful

DON'T FORGET

The deadline for Circuit Cellar INK Design Contest entries is May 1, 1989. Please get your submissions in to us SOON.



Circuit Cellar INK
Design Contest
P.O. Box 772
Vernon, CT 06066

COLOR IMAGEWISE

ZIP Software now allows both the original serial DT01 and ImageWise/PC to digitize and display full-color pictures on an IBM PC with VGA. Using three colored filters (red, green, & blue), ZIP digitizes three sequential pictures and combines them into one full-color composite PCX-format picture which can be displayed on VGA or used in a variety of paint and publishing programs.

ZIP Image Processing Software
\$79.00

The following is available from:

CCI
4 Park Street
Vernon, CT 06066

For information and orders:

Tel: (203) 875-2751
Fax: (203) 872-2204

Item 1: ImageWise/PC PC board experimenter's kit. Comes with 4-layer PC board, and assorted key components including 27C256 EPROM; TML 1852 D/A; AD7226 D/A; CA331 8A/D; RCA jacks; ferrite beads; user's manual; and ImageWise/PC utilities on IBM PC format diskette.

Order IWPC-EXP \$399.00

Item 2: ZIP image processing software for ImageWise/PC from Hogware.

Order IWPC-ZIP \$79.00

A procurement list for all ImageWise/PC board components is posted on the Circuit Cellar BBS. Full kits are not currently available. ImageWise/PC is available assembled and tested.

All payments should be made in U.S. dollars by check, money order, or Mastercard and Visa. Shipping and handling: surface delivery add \$6.50 for U.S. Call for Canada and air freight delivery elsewhere.

FROM THE BENCH

Creating a Network-based Embedded Controller

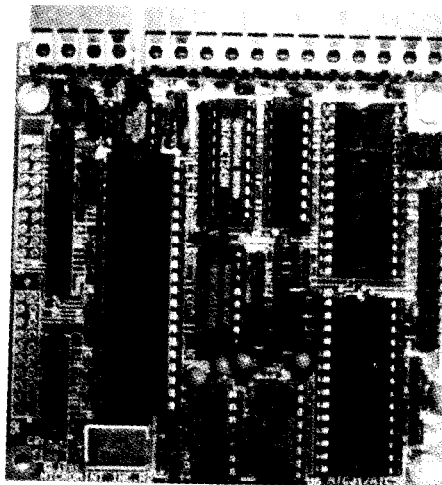
by Jeff Bachiochi

The activity level of readers responding to the DDT-51 article overwhelmed many of us involved with that project. For many, the DDT-51 would be an even more interesting project if they had an 8031-based system to use it on.

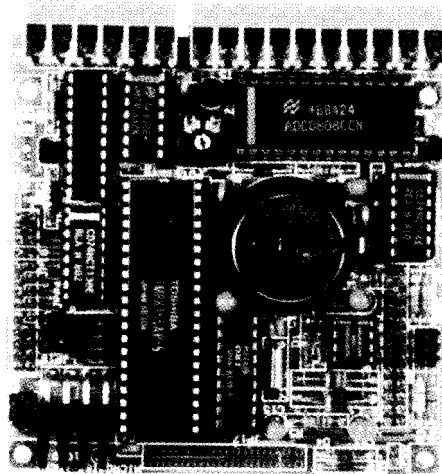
The primary reason for the success of DDT-51 is that it puts kilobuck development-tool potential into the hands of experimenters and consultants for less than \$100. Unfortunately, once you've whetted a person's appetite with low-cost development, the next immediate need is cost-effective end-use implementation.

The majority of 8031 and 8052 controller boards are designed as general-purpose controllers with external-board expandable I/O. This is not a new revelation, just an observation. If you want an ADC, you add an ADC board. Want more parallel I/O? Then add a parallel I/O board.

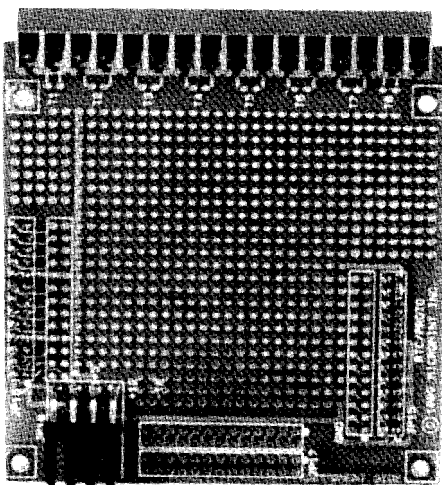
The benefit of this architecture is that such a system can be expanded in many directions to incorporate massive amounts of I/O. Where the system fails, however, is that the features which offer easy expandability and on-board software development in large configurations are usually unnecessary in minimum-configuration, minimum-control applications. Often, we don't need the 48 I/O lines afforded through an expansion I/O card but only two or perhaps three inputs. Similarly, we may not need microsecond-speed, infinite-accuracy analog conversions but only a simple 8-bit ADC for temperature measurements. What's needed is a plain-brown-wrapper 8031/8052 controller that is optimized for these mini-



RTC 1



RTCIO



RTC-Proto

mal-configuration applications.

This month I am presenting a new, lower-cost 803x/805x system which meets these criteria. I've included features that will let you use this design for one-shot prototyping or for serious applications. This new Real-Time Controller, designated the RTC31 (8031 processor) or RTC52 (80C52-BASIC processor), is designed to be small and cost effective.

The RTC system measures only 3.5 inches square and uses vertical-stacking connectors for I/O expansion. The RTC processor board contains the processor, EPROM and RAM memory, address decoding and buffering, parallel I/O with screw terminals, and an RS-232/RS-485 serial port.

So far, we have designed two stackable expansion boards. First, the RTCIO board (which I consider a packaging achievement), contains three parallel TTL I/O ports; an 8-channel, 8-bit A/D converter; a 4-channel, 8-bit D/A converter; a battery-backed real-time clock; and a DC-to-DC converter which allows complete 5-volt-only system operation. The second, the RTC-Proto board, is just that—a prototyping board. It permits the user to add his or her own special I/O circuits.

A stacked-board arrangement has certain benefits besides eliminating costly gold-contact backplanes (motherboards). It allows configuration of either a basic system for experimentation or an expanded system for black-box applications, yet still retains its low profile. Each vertically stacked board only increases the height by 5/8 inch.

Additional cost can be saved by populating only the I/O necessary for the application.

The RTC Processor Board

The RTC31 processor board, outlined in Figures 1a and 1b, has a no-frills architecture designed to meet basic system requirements. Like any 8031-based system, it needs at least a processor, a low-address latch, and memory*

The 8031 microcontroller has the ability to address two independent 64K regions: data space (read/write memory) and code space (read-only memory). These spaces can be addressed separately or overlaid for a combined "anything goes" space. To accommodate any programming eventuality, the entire 64K address space of the 8031 is addressable on the RTC31 and two memory sockets are provided to handle either 8K or 32K RAMs or EPROMs (6264/62256 RAMs or 2764/27256 EPROMs).

A chip select (*CS) decoder would normally require two ICs, one for decoding minimum address blocks (8K; 2764/6264) and another for the "ORing" of 8K blocks when using larger devices (32K; 27256/62256). A small bit of flexibility is traded here to reduce this to one chip (Figure 1a). An open-collector decoder allows 8K blocks to

be "ORed" by wiring them directly together. These "wire-ORed" chip selects save space but require that memories sharing any address block, as in a separated data/code space, be of the same memory size.

Serial communication is also provided on the RTC31 processorboard. Full-duplex RS-232 communication with +5-volt-only operation is provided by a MAX232 level converter. In addition, half-duplex multidrop RS-485 communication, which opens exciting new application possibilities by network-connecting embedded controllers, is also included. Using a multicontroller communication network, the user can move task processing from a central area to the source. Adding this potential is as simple as the addition of a single B-pin RS-485 bus-driver chip (plus a little networking software).

The rest of the board logic consists of a bus driver for the multiplexed address/data bus and a "glue" chip for "ORing" some control lines. The bus driver would not be necessary if we limited the system to this single processor board with no further expansion (the ADO-7 port on the processor can drive up to four TTL loads; only three are used). Instead, the buffered address/data bus is directed through two vertical-stacking connectors which facilitate both electronic expansion and mechanical support (more about this in a bit). Two additional connectors are in-

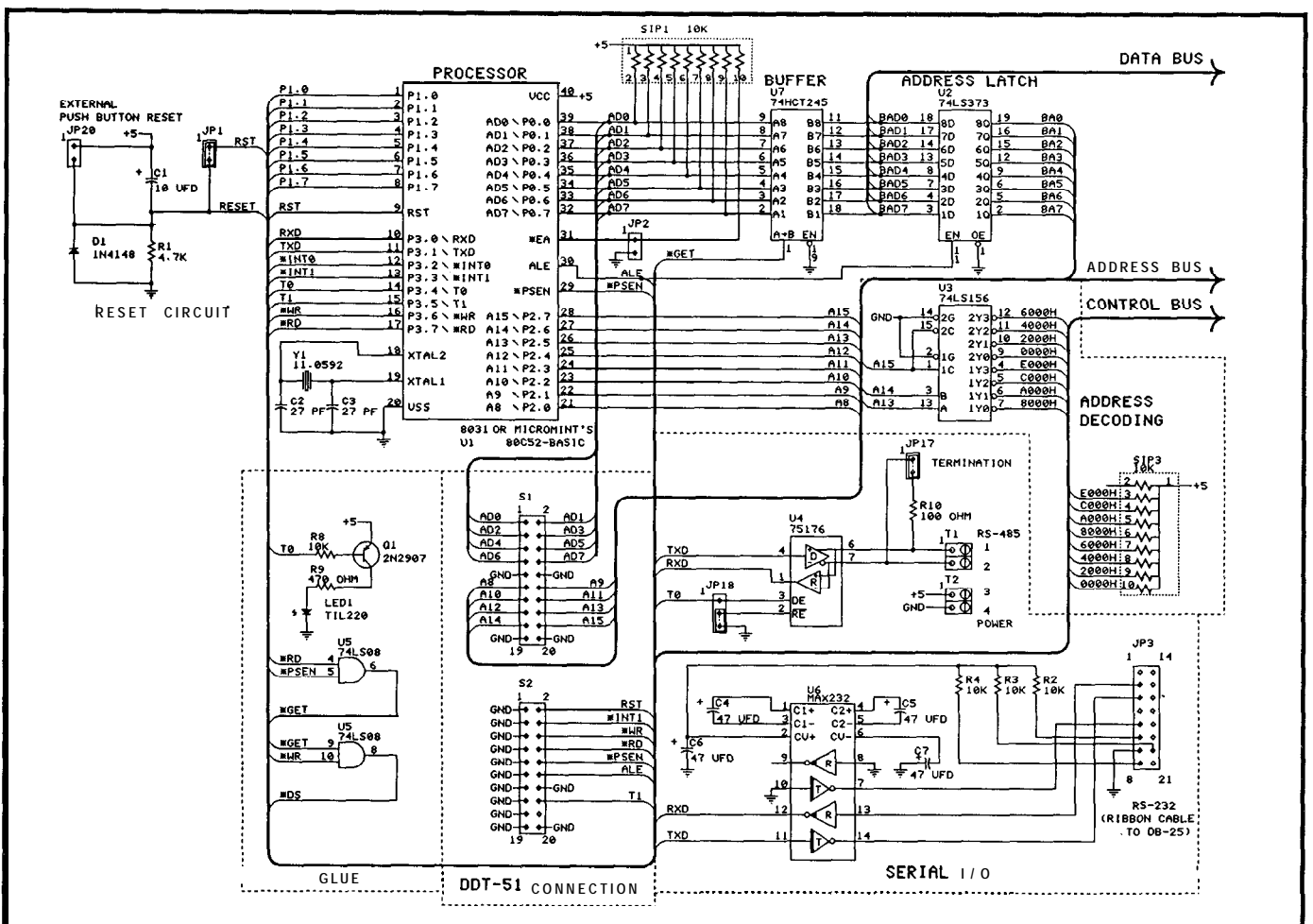


Figure 1a - The RTC31/52 provides flexible address space decoding plus RS-232/RS-485 serial communication and direct connection to the DDT-51 debugging tool.

cluded to directly connect the RTC31 to a DDT-51. Together, they serve to create the ideal controller development environment.

Finally, no basic controller configuration would be complete without some parallel I/O. Eight bits from port 1 and four bits from port 4 of the processor are brought out to screw terminal blocks for user I/O. These 12 TTL I/O bits are sufficient for most simple controller applications, making additional hardware unnecessary.

The total chip count for the RTC31/RTC52 design is nine chips. Cramming these, plus the associated connectors and **discretes**, onto a **3.5-inch-square** board was no easy **task**; we had to use fine-line (**8-mil**) PC-board technology to fit everything. The end result, which draws only **80 mA** at 5 volts (without the RS485 driver) is destined to be used in many upcoming embedded controller projects around here. Stay tuned.

System Expansion with the RTCIO

For those who need a little bit of everything, the stacking RTCIO board, shown schematically in Figure 2, provides a powerful assortment of I/O including parallel TTL, ADC, DAC, and real-time clock.

An 8K block of data memory space (at **E000H**) is allocated for all the I/O functions in an RTC system. It is decoded into two jumper-selectable groups of four addresses, called "upper" and "lower." Within each group,

the addresses are designated as CS0, CS1, CS2, and CS3. CS0 is used for parallel I/O; CS1 is used for the ADC; CS2 is used for the DAC; and CS3 is used by the real-time clock. Since each RTCIO board requires only four of these addresses for its functions, two RTCIO boards can be stacked on the RTC processor board if required.

The 24 bits of parallel I/O are provided by an 8255 PPI. The chip is connected to external devices through a **26-position square-pin header**. **The PPI** has four port registers which are memory-mapped to addresses E000H-E003H (if using the low address group). The PPI is initialized by writing to register 4, the mode port. The value written to the port configures the first three registers as input or output. Register 1 (port A) and register 2 (port B) are **8-bit bidirectional ports**, each configured as either 8 bits in or 8 bits out. Register 3 (port C) can be similarly used or split into control-line support. Once the configuration **has been** set in the mode register, these ports can be written to or read from directly.

Eight channels of **8-bit A/D conversion** are provided through an ADC0808 ADC chip. The eight external inputs, which can be in a range from 0 to 5 volts, are brought to the **A/D converter** through screw-terminal blocks on the edge of the board. The conversion speed of the ADC0808 is about 9000 conversions per second (about 110 microseconds per conversion).

The ADC0808 is memory-mapped at the default addresses **E100H-E107H** (channels 0-7). A dummy write to

8031 Forth

Complete Development Environment

- Bryte's development environment uses **BRYTE-FORTH** on the actual production hardware during product development. No emulators, no changes, no surprises.
- Optional PC-based cross-development tools use DOS files as microcontroller mass storage. These files can be used to generate compact EPROM images, detailed listings, and cross-references.

BRYTE-FORTH 8031 EPROM	\$100.00
(includes 130 page User's Manual)	
Utility disk(s)	\$65.00 *
Cross-compiler/Cross-assembler	\$235.00 *
8031 unlimited quan. license	\$1000.00 *

* Includes complete source code

**bryte
computers
inc.**

P.O. Box 46 • Augusta, ME 04330-0046

Call Today
207-547-3218

Circle No. 108 on Reader Service Card

Create Professional Quality Circuit Diagrams with

MacSchematic

from
**Thinking
Tools**

MacSchematic is a library of over 800 Electrical and Electronics symbols for professional quality circuit diagrams on the Mac.

Symbols are in both PICT format for MacDraw, MacDraft, or other CAD/Draw programs and in MacDraw II libraries.

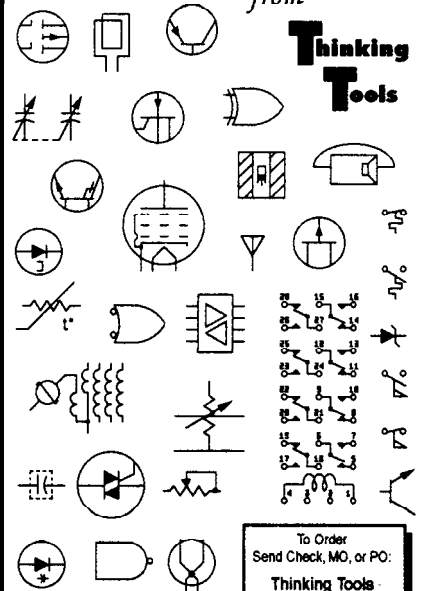
MacSchematic symbols are object oriented for superior printer, plotter and laser output. They can be rescaled without losing their high resolution.

MacSchematic symbols are designed to snap to grid so their external connections fall on grid points and lines connect to symbols "on the dot".

Includes symbols from ANSI Standard Y32.2:

- Analog Components
- Gak/Digital/Devil
- Industrial Wiring
- Ladder Diagrams

Not Copy Protected
All Macs



Symbols shown at 50% Reduction.

MacSchematic: \$80
Demo + Manual: \$10
plus \$5 shipping/handling

To Order
Send Check, MO, or PO:
Thinking Tools
2411-M Linden Ave
Baltimore, MD 21217

For Information
or Visa/MC Order
Call (301)-383-6490

Circle No. 141 on Reader Service Card

E100H will start a conversion of channel 0 (IN0). The EOC (end of conversion) signal is inverted by Q2 and available on JP25. Either INTO or INT1 may be selected to provide a hardware EOC as opposed to doing a software delay loop to predict EOC. Reading the same address after EOC occurs will grab the 8-bit conversion value from the ADC0808. The resolution of the converter is about 20 millivolts with a V_{ref} of 5 volts. If you're using 80C52-BASIC with this ADC, a line of BASIC takes longer to execute than a single conversion time, so waiting for EOC is not necessary!

An AD7226 furnishes four channels of D/A conversion, with each output ranging from 0 to 5 volts (set by the voltage reference). Screw terminals along the edge of the PC board provide easy connection to the outside world. Full-scale-change settling time is 4 microseconds maximum. The memory-mapped default addresses of the D/A converter are E200H-E203H. Writing an 8-bit value to E200H will set the output for channel 0 (V_{outA}). The 8-bit resolution of the AD7226 is about 20 millivolts using a V_{ref} of 5 volts. Reference voltages for both the ADC and the DAC are supplied by an LM336Z-5, a 5.0-volt precision reference diode. All the I/O on this board runs on 5 volts except V_{ref} and the DAC. The DAC requires +11.4 to +16.5 volts for V_{dd} and a negative bias on V_{ss} . These voltages can

come from an external power supply or can be generated on-board by the MAX633 DC-to-DC converter. **Editor's Note:** For more information on how these converters function, see "Switching Power Supplies" beginning on page 10.1

An OKI M6242B provides a real-time clock/calendar (on-board battery backup). Control of the clock/calendar is performed through sixteen registers at the default addresses E300H-E30FH. Each digit of the seconds, minutes, hour, month, day, year, and day of week can be individually read from or written to the chip as a BCD nibble.

In addition, three registers indicate various status conditions. Reading and writing the registers may be done with or without regard for whether the chip is busy. If busy is disregarded, double reads should be done and the values compared to ensure correct information.

If system power is lost, falling lines may cause a random *CS on the clock/calendar chip since it is battery-backed. To prevent this from occurring, the programmable V_{cc-low} level detection output line from the MAX633 is connected as a clock chip enable. An out-of-tolerance V_{cc} (during power up or power down) deselected the clock, preventing erroneous data from being written into the clock registers.

The eight chips on the RTCIO board combine to make a powerful I/O combination. Partial population of the

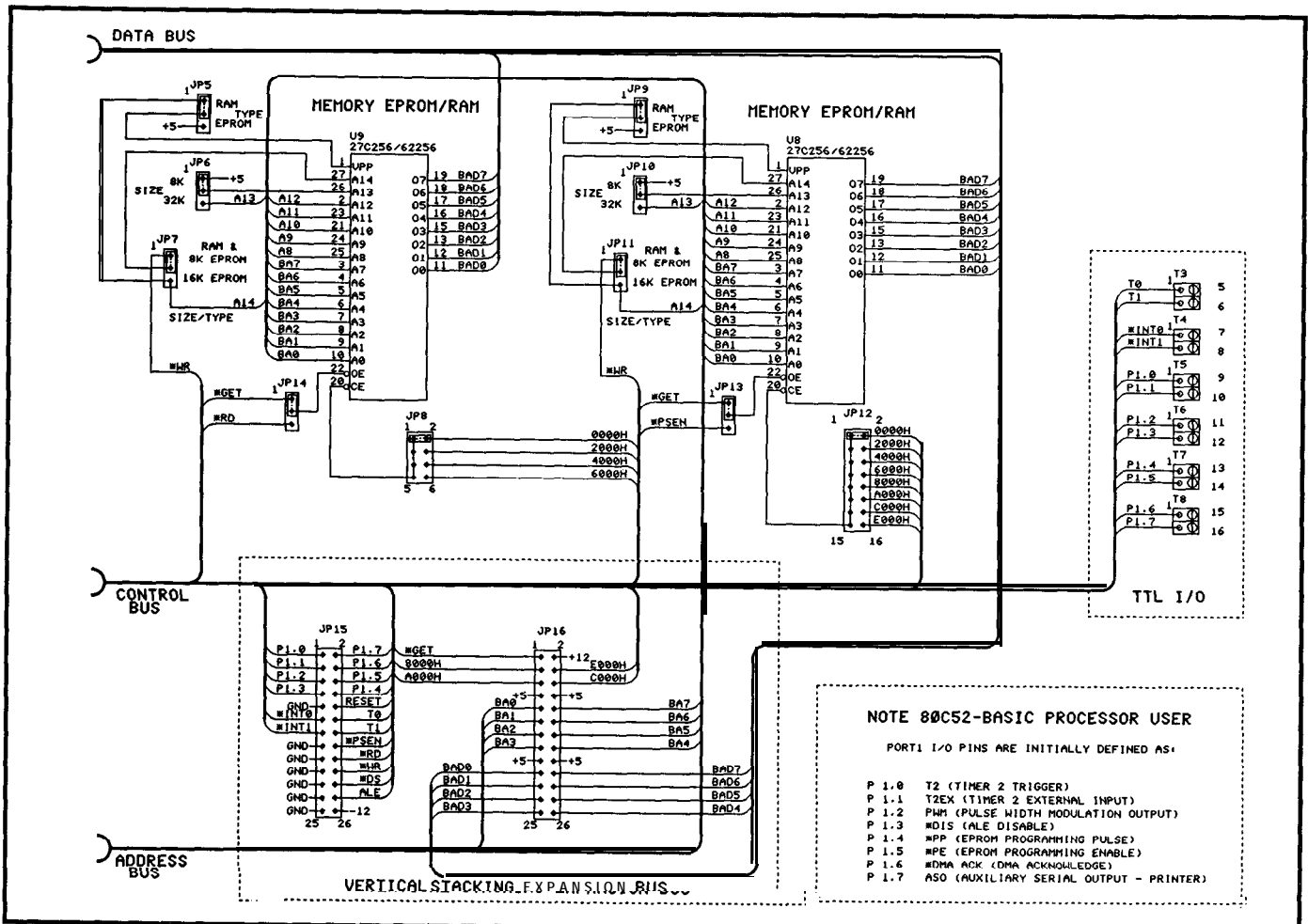


Figure 1 b- Either 8K or 32K RAM and EPROM may be used on the RTC3 1/52. Screw terminals and vertical-stacking connectors make expansion easy and compact.

I/O board makes it a cost-effective approach to system expansion. Need more?

For those of you who ask, "What about the...," a prototyping board, with an array of holes on 0.100-inch centers, makes it a snap to add a custom circuit to the microcontroller engine. In addition to a bank of 12 screw terminal blocks, a 26-pin header layout is provided for bringing control signals onto or off of the proto board.

Minimum System + Flexible Design = Minimum Cost

Summing it all up, the RTC31 & RTC52 are indeed a powerful combination. The stacking design allows a minimum system to be flexible yet inexpensive for both the developer and the end user. The degree of expansion is controllable, yielding a lower total cost, yet system expansion adds increments of only 5/8 inch to the total height. Five-volt-only operation is maintained by on-board DC-to-DC converters, and RS-485 provides multidrop communication over a twisted-pair cable. (The ability to network microcontrollers might not be a big advantage to you now, but I think you will see it develop briskly in the near future, especially here in the pages of Circuit Cellar INK.) Finally, the RTC system connects directly to the DDT-51 and provides a low-cost platform for using it. □

IRS

- 2 16 Very Useful
- 2 17 Moderately Useful
- 2 18 Not Useful

SOURCES

MAX633

Maxim Integrated Products, Inc.
120 San Gabriel Drive
Sunnyvale, CA 94086
(408) 737-7600

M6242B

OKI Semiconductor
650 North Mary Avenue
Sunnyvale, CA 94086
(408) 720-1900

AD7226

Analog Devices
One Technology Way
P.O. Box 9106
Norwood, MA 02062-9106
(6 17) 329-4700

ADC0808

National Semiconductor Corp.
2900 Semiconductor Drive
Santa Clara, CA 95051
(408) 72 1-5000

MAX232

Maxim Integrated Products, Inc.
or
Dallas Semiconductor
4350 Beltwood Pkwy South
Dallas, TX 75244-3219
(2 14) 450-0400

8031 /51 80C31/51

Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051
or

Matra Harris Semiconductor
2840-100 San Tomas Expwy.
Santa Clara, CA 95051
(408) 986-9000
or

Signetics Corporation
811 E. Arques Avenue
P.O. Box 3409
Sunnyvale, CA 94088-3409
(408) 99 1-2000

80C52-BASIC

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

Miller Coil

J.W. Miller Division
Bell Industries
19070 Reyes Avenue
Rancho Dominguez, CA
90221
(2 13) 537-5200

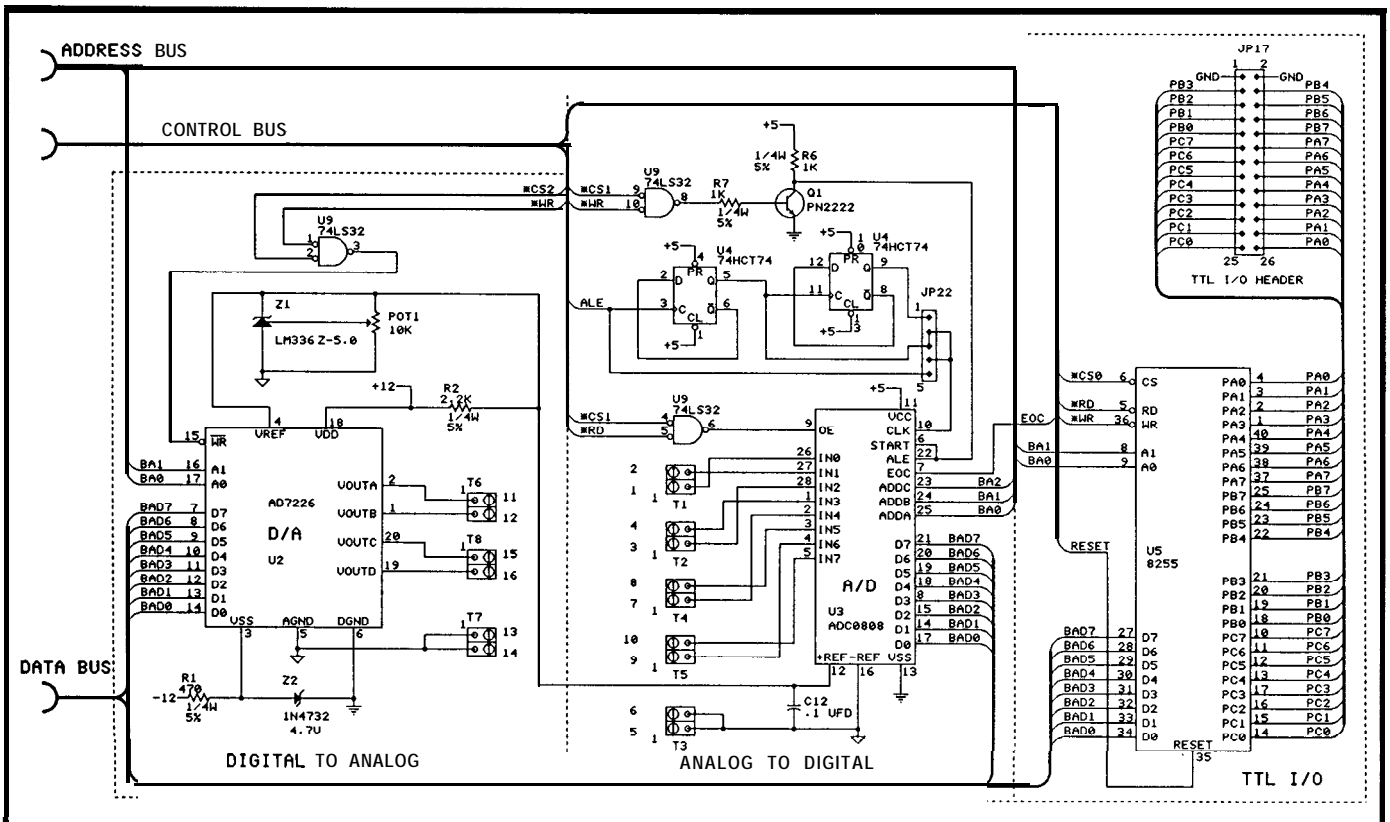


Figure 2a—The RTCIO I/O board provides such useful functions as 24-bit TTL parallel I/O; 8-channel, 8-bit analog-to-digital conversion; and 4-channel, 8-bit digital-to-analog conversion.

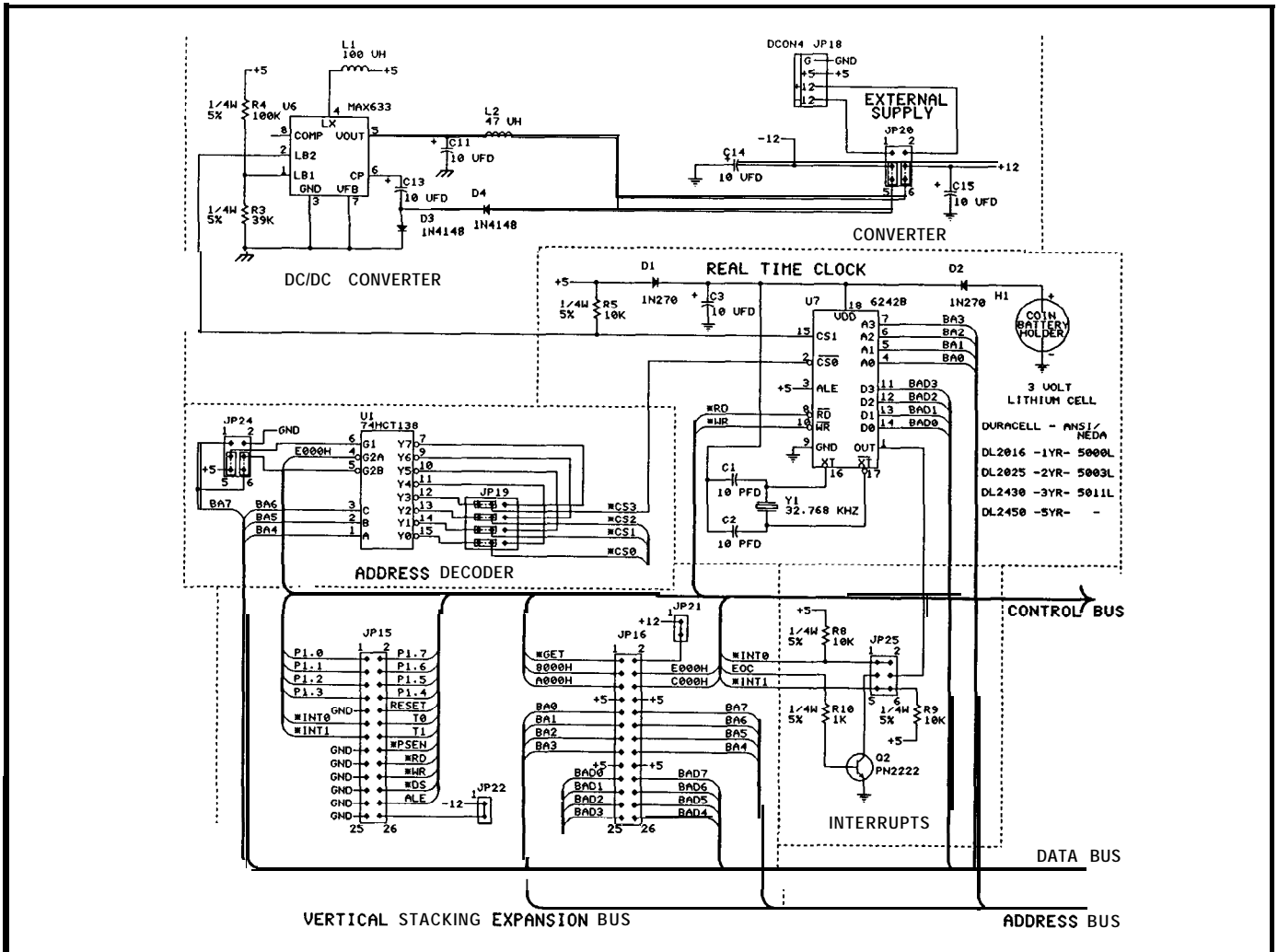


Figure 2b—Additional features of the RTCIO board include a battery-backed real-time clock and a DC-to-DC converter.

The following is available from: CCI • 4 Park Street, Suite 12 • Vernon, CT 06066

For information and orders: Tel: (203) 875-2751 • Fax: (203) 872-2204

RTC31 board

RTC31 8031 microcontroller board experimenter's kit. Includes PC board, 8031 microprocessor, MAX232, sockets, all ICs except RAM and EPROM, piggyback expansion board headers, manual, and detailed parts list. Screw terminals optional.

Order **RTC31 K-1** \$89.00
RTC-term (screw terminals) Add \$8.00

RTC52 board

RTC52 8052 microcontroller board experimenter's kit. Same as RTC31 except a Micromint CMOS 80C52-BASIC microprocessor is supplied instead of the 8031. 80C52-BASIC chip can be jumper-selected to also function as 80C31/80C32.

Order **RTC52K-1** \$89.00
RTC-term (screw terminals) Add \$8.00
Intel BASIC-52 Programmer's manual \$15.00

RTC-PROTO Board

RTC-system prototyping board. Includes PC board, piggyback expansion board sockets, and eight 2-position screw terminal blocks.

Order **RTC-Proto** \$39.00

RTCIO board

RTCIO Expansion board parallel I/O and A/D kit. Includes PC board; parallel port PPI; & channel. 8-bit A/D chip; related board-mounted components; and piggyback expansion board sockets. Screw terminals optional.

Order **RTCIOK-1** \$79.00
RTC-term (screw terminals) Add \$8.00

RTCIO Board Options:

OPTION #1: Hardware Real-Time Clock components for RTCIO board. Includes I.46242 clock chip, crystal, Lithium battery, battery holder, and related components.

Order **RTCIO clock upgrade #1** Add \$30.00

OPTION #2: D/A components for RTCIO board. Includes AD7226 4-channel D/A chip and related components.

Order **RTCIO D/A upgrade #2** Add \$35.00

OPTION #3: 5-volt-only operation DC-to-DC converter components for RTCIO board. Includes MAX633, inductors, and related components.

Order **RTCIO DC-to-DC upgrade #3** Add \$19.00

The RTC31, RTC52, and RTCIO boards are also available assembled and tested and in volume OEM quantities. Call for price and delivery.

All payments should be made in U.S. dollars by check, money order, MasterCard, or Visa. Shipping and handling: surface delivery add \$3 for U.S.; 2nd-day delivery add \$7 for U.S. Call for Canada and air freight delivery elsewhere.

HD647180X-A New 8-Bit Microcontroller

Embedded Controllers Get Respect

by Tom Cantrell

Maybe it's wishful thinking, but it seems as though the infatuation with 32-bit micros and "workstations" is fading. Sure, there is plenty of sizzle, but where's the beef? Clearly, the hype level far exceeds that justified by the market size.

Enter embedded controllers. The exact definition of an "embedded controller" is a little hard to pin down. It's one of those things you can't easily define, but you know one when you see it. INK readers, being heavily involved in real-world applications, know what I mean.

Whatever an embedded controller is, the chip manufacturers are starting to realize (it's suprising they ever forgot) that a design win in a car, phone, or consumer product is much more lucrative than one in the latest "Personal-SuperMicro-Mainframe," glamorous as the latter may be.

Four-bit controller chips continue to hang in there, and 16-bit controllers are being announced. But for now, and the foreseeable future, 8-bit controller chips lead the pack (Figure 1).

The traditional 8-bit workhorses are the Intel 8048 and 8051, Motorola 6801 and 68HC11, and Zilog Z8. Let's take a look at a new 8-bit contender, the Hitachi HD647180X.

CPU Roots

The immediate predecessor of the HD647180X, the HD64180 (also offered by Zilog as the Z64180 and Z180), is a highly integrated Z80-compatible CPU. Included on the chip are the CPU core, system support logic, and a number of I/O devices.

Though the HD64180 CPU core design is quite different from the Z80—the HD64180 is microcoded—it faithfully executes the Z80 instruction set. From the programmer's point of view, the only difference is that the HD64180 has a few extra instructions (notably including MuLTipl) and many of the instructions execute in fewer clock cycles. Another byproduct of microcoding is that the HD64180 can trap "bad" opcodes typically caused by

what really sets the HD64180 apart from other 8-bit CPUs is the addition of a bank-select-type MMU (Memory Management Unit) which expands memory addressing beyond the 64-kilobyte barrier to 1 megabyte.

The HD64180 also integrates those system "glue" functions required in most designs (wait state generation, DRAM refresh, and interrupt control) to reduce chip count and improve performance.

Finally, the HD64180 includes a complement of I/O functions: two channels each of UART, timer, and DMA as well as a single-channel clocked serial I/O port. Compared to older 8-bit controllers, the number and functionality of the HD64180's I/O devices are much improved.

The HD64180 is built using a CMOS (Complementary Metal-Oxide Semiconductor) process and includes a number of low-power modes which allow programmable tradeoffs of functionality and power consumption. Low

power is not just an issue for battery-powered designs. It affects cost (small power supply, no fan, plastic IC package) and reliability (especially in harsh environments).

The CMOS process cuts power consumption dramatically. Consider that a 6-MHz HD64180 consumes just

Sales forecasts for microcontrollers (worldwide)

(Millions of dollars)

	1988	1989	1990	1991	1992	1993
4-bit	\$944.7	\$991.4	\$979.0	\$1,081.3	\$1,174.6	\$1,181.8
8-bit	\$1,576.3	\$1,654.4	\$1,626.0	\$1,911.0	\$2,219.6	\$2,267.8
16-bit	\$15.0	\$17.9	\$30.0	\$58.1	\$125.6	\$219.1
Total	\$2,536.0	\$2,663.7	\$2,635.0	\$3,050.3	\$3,519.8	\$3,668.6

Source: Dataquest, Inc.

Figure 1 -Sales forecasts show that the 8-bit microcontroller market will be strong for years to come.

software errors or an electrical glitch. Needless to say, critical control applications such as those involving heavy equipment or life-support products cannot be allowed to run wild! But

30 mA maximum (15 mA typical) under full-speed operation. By contrast, the NMOS ZSO, without any of the HD64180 on-chip I/O, consumes more than six times as much power (200 mA max).

At one time, CMOS was a dirty word implying slow performance and nonstandard I/O levels. Today, any serious contender in the controller market must offer a high-performance, low-power CMOS process.

'X' Marks The Spot

The new "X" version moves the original HD64180 design squarely into the 8-bit controller fray. Three key additions are on-chip EPROM and RAM, parallel I/O lines, and analog input capability (Figure 2).

On-chip Memory

Memory technology marches on and to paraphrase political wisdom, "A K here and a K there—pretty soon you're talking real memory." The HD647180X includes 16K bytes of EPROM and 512 bytes of RAM—twice as much memory as older chips and more than enough for typical 8-bit applications.

A unique feature of the HD647180X is the use of EPROM; a masked-ROM version isn't even offered. The concept, known as "ZTAT" (Zero Turn Around Time) or "OTP" (One-Time Programming) combines the field programmability of EPROM with the low cost of a nonwindow plastic package. This makes sense when you realize that EPROM-based devices allow production floor firmware fixes/upgrades and eliminate ROM mask charges and inventory exposure. For lab and development work, which actually requires erasability, a windowed version of the HD647180X is offered.

Programming the HD647180X EPROM is simple. Timing and voltage levels are the same as those of a standard 27256. All you need is a socket adapter which connects the pins of the 27256 programmer socket to the HD647180X (Figure 3). Those wishing to secure their intellectual prop-

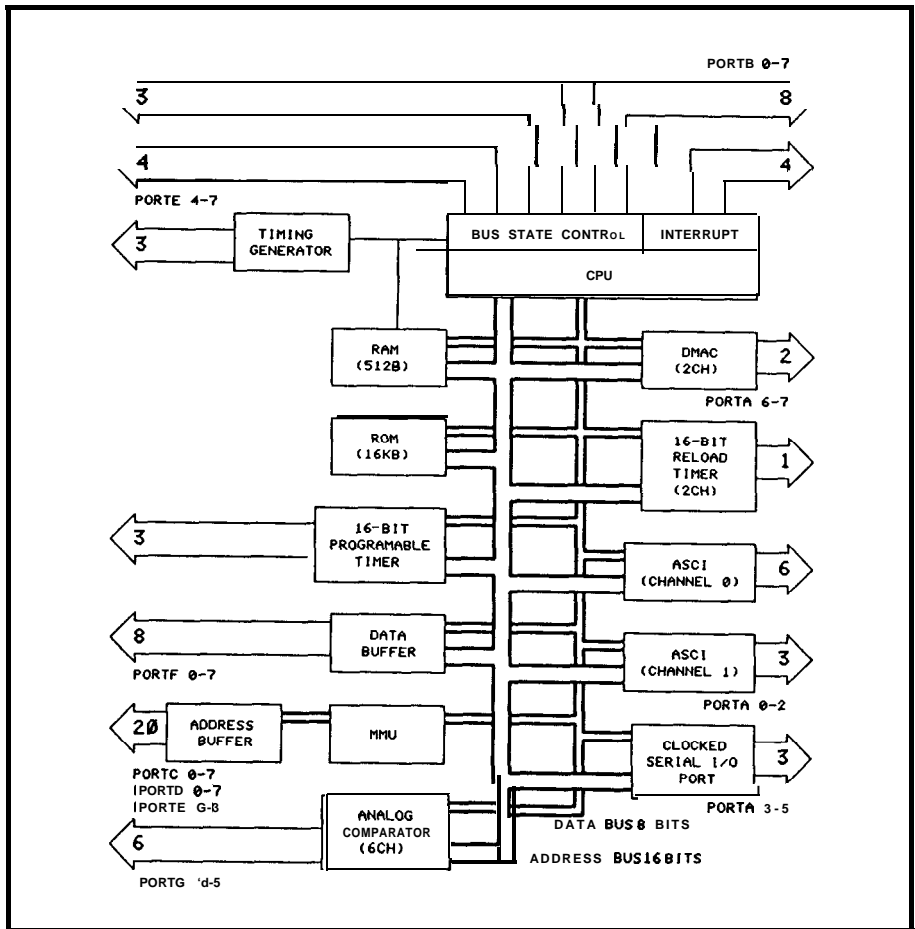


Figure 2— The HD647180X is similar to the HD64180 with the addition of on-board RAM, EPROM, and analog comparator

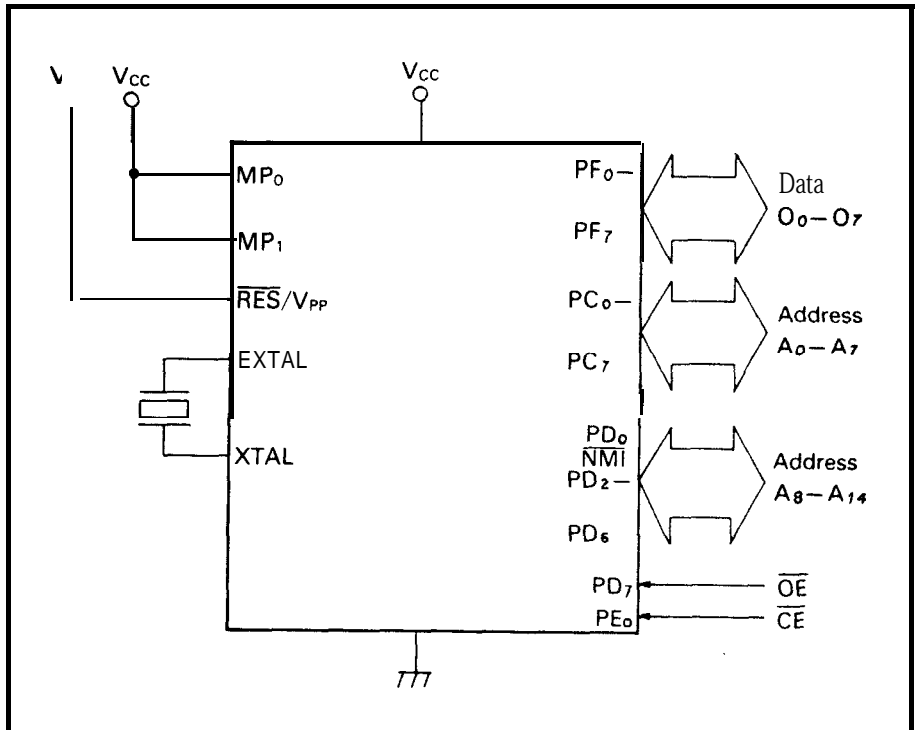
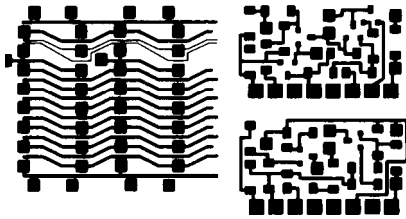


Figure 3—A simple adapter plus an EPROM programmer capable of programming 27256s are all that are needed to program the HD647180X.

DROE GE

DESIGN ROBOT FOR
THE ORIGINATION OF
EXACTING GRAPHIC
ENGINEERING

A MANUAL PRINTED
CIRCUIT CAD/CAM
SYSTEM



MULTI-LEVEL SYMBOL CONCEPT. MEMORY LAYOUT ABOVE IS A SYMBOL MADE FROM TWO CHIP SYMBOLS WITH ADDED BUS WIRE. CHIP SYMBOLS ARE MADE FROM MULTIPLE PAD SYMBOLS.

BASIC CIO.00 POSTPFIIO:

CGA 3 COLORS 12 LAYERS
64 BY 64 INCH WORKSPACE
ANY GRID TO 0.001 INCH
RUNS ON ANY PC COMPATIBLE
15 LINE WIDTHS
DOT MATRIX OUTPUT
200KB DOCUMENTATION ON
TWO DISKS
16 WORK AREAS SAVEABLE
STITCH BETWEEN LAYERS
ZOOM AND PAN TO ANY SCALE

ADVANCED 66100.00 POSTPFIIO:

ABOVE FEATURES
EPA RESOLUTION 15 COLORS
LARGE R JOBS
MOUSE
ADVANCED EDITING
PRINT 0 MANUAL

ENVIRONMENTAL OPTICS CORP.
P.O. BOX 296 BATAVIA, IL 60510
SEND BASIC ADVANCED INFORMATION
CHARGE VISA MC PAYMENT. ENCLOSED
CHARGE NUMBER _____ EXP _____

NAME _____
ADDRESS _____
CITY _____
STATE _____ ZIP _____
TELEPHONE ORDERS EVENINGS (312) 879-2949
THIS ADD WAS COMPOSED WITH DROE GE AND
PLOTTED ON OUR PHOTO PLOTTER. WE CAN MAKE
PHOTO PLOTS FOR YOU FROM PROGRAM OUTPUT.

Circle No. 117 on Reader Service Card

	MODE 0 (Single Chip)	MODE 2 (Expanded)
PORT A (8 bits)	Parallel or on-chip I/O lines	<same>
PORT B ""	Parallel I/O	CPU control lines
PORT C ""	Parallel I/O	A0-A7
PORT D ""	Parallel I/O	A8-A15 or parallel I/O
PORTE ""	Parallel I/O	CPU control lines and A16-A19 or parallel I/O
PORTF ""	Parallel I/O	DO-D7
PORT G (6 bits)	Analog or parallel input	<same>

Figure 4-The HD647180X's many modes allow the designer to match the features of the chip to the application.

erty will be pleased to note that the HD647180X has protection modes to prevent the contents of the EPROM from being read. To get your code, potential pirates will have to get down and dirty and probe the die itself.

Parallel I/O

The HD647180X operates in one of four modes, selected by the levels on two external pins at reset. EPROM programming mode (mode 3) is self-explanatory. Mode 0 is single-chip mode while modes 1 and 2 are expanded modes (Figure 4).

In mode 0, most of the chip's pins (and there are a lot—the HD647180X is packaged in 80-pin flat pack or 84-pin PLCC) are configured as I/O lines. A total of 54 I/O lines are provided, organized as six 8-bit ports and a six-bit port. Generally, the ports are programmable at the bit level as parallel input or output, though there are some restrictions. Port G is input only while port A is programmable (at the bit level) as either parallel I/O or UART and DMAC lines. If the 16K EPROM, 512 bytes RAM, and 54 I/O lines aren't enough for your application, you can use one of the expanded modes.

Mode 1 replaces most of the I/O lines with a complete address (20-bit) and data (8-bit) bus, as well as a number of bus status and control lines.

Furthermore, the on-chip EPROM is disabled. In essence, a mode 1 HD647180X is quite similar to a standard HD64180 with the addition of the on-chip RAM and a few I/O lines.

Mode 2 is more likely to find use in actual applications. It is kind of a cross between mode 1 (fully expanded) and mode 0 (single-chip) in that the on-chip EPROM remains enabled but an external bus (for connection to additional memory and peripheral ICs) is provided as well. A neat feature is that 12 of the I/O pins can be programmed individually to serve as either upper address lines (A8-A19) or parallel I/O. Many applications that only need a small external address space (to add an I/O chip or two) can take advantage of this.

Analog Comparator

Real-world applications often need to process analog signals. To this end, the HD647180X includes a six-channel analog comparator which works as follows:

Six lines (Port G) act as inputs to the comparator. Any of the six lines can be programmed to serve as either a reference voltage (V_{ref}) or a compared voltage (V_{in}) (naturally, the same line shouldn't be selected for both functions). The comparator function simply sets a result bit indicating

which voltage (V_{ref} or V_{in}) is higher.

Though not intended to replace a high-resolution A/D chip, the ability to switch reference and input channels leads to some interesting possibilities. For instance, five of the inputs could be "hardwired" to different reference voltages with which an input voltage could be compared. Each comparison takes only thirteen clocks, or about 2 μ s with the CPU running at 6 MHz, so even a brute force approach of five sequential comparisons could approach a 100k/second sampling rate. Similarly, a DAC (with output voltage selected by HD647180X output lines) could serve as a programmable reference for comparison with five different voltage inputs.

Bottom line

Whether the HD647180X fills your bill depends on how well its feature set maps to your requirements. It is ideal for midrange performance applications that need single-chip form factor (with low-cost field programmability), lots of on-chip memory and I/O, low power consumption, and Z80 software compatibility. On the other hand, some of the HD647180X "large system" features—such as the MMU and DRAM refresh controller—may be misplaced in a chip that is best used in a "single-chip" mode.

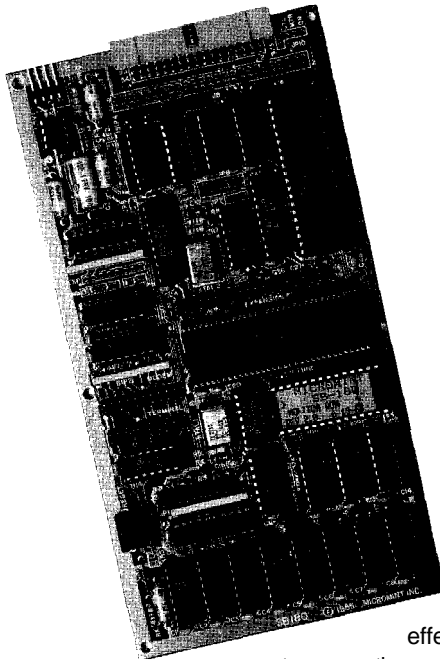
Price and Availability

The 100-piece price for the HD647180X is \$23.30 for the ZTAT/One-Time Program version in PLCC package and \$94.00 for the windowed LCC package version. For literature or parts, contact your local Hitachi representative or distributor or:

Hitachi America, Ltd.
Semiconductor & IC Division
2000 Sierra Point Parkway
Brisbane, CA 94005-1819
(415) 589-8300

IRS

2 19 Very Useful
220 Moderately Useful
221 Not Useful



Finally!!!

The Circuit Cellar SB180 Single-Board Computer Kit

Since the time it was introduced on the cover of the September '85 issue of BYTE, the SB180 has established itself as one of the most reliable and cost-

effective single-board 8-bit computer systems on the market. Incorporating up to 256K RAM, an

8K EPROM monitor, a floppy disk controller, two serial ports, and a parallel printer port, the SB180 has a remarkable list of features for its small 4" x 7.5" size. An optional SCSI interface adapter easily expands the SB180 to include a hard disk.

Using the Z80-code-compatible Z180/HD64180 super chip, the SB180 runs the thousands of Z80/8080/8085 programs faster and more efficiently than ever before. Up to three times as fast as a 4-MHz Z80, the 9-MHz SB180 can be used as a stand-alone controller, or as a complete development system running CP/M 2.2, CP/M Plus, Z-System, MP/M II, TurboDOS, or Oasis operating systems.

The SB180 comes with a plug-and-go 24-command high-performance ROM monitor which exercises and tests all its basic functions. For real computing performance, we have an extensive software collection including the Z-System enhanced disk operating system. Considerably more advanced than CP/M, Z-System offers users utility programs and DOS features that have only recently become common to 16-bit PC users.

Through a special licensing arrangement, Circuit Cellar is now able to offer a complete 9-MHz SB180 kit (less DRAM) for the remarkable price of \$195. Just add a bank of 64K or 256K DRAMs and you are instantly on the air. The optional Z-System O/S software has also been redesigned with computer experimenters in mind. The operating system checks available memory, apportioning it between TPA and RAM disk, and looks for the SCSI hard disk as well. Adding a 32-Mbyte hard disk is as easy as plugging on the SCSI adapter to what you already have running—no merging or recompiling software. Complete source code for the BIOS and ROM monitor are also included. Plug and go!

Features:

- 9.216-MHz Z180/HD64180 CPU
- 64K or 256K bytes RAM supported
- 8K monitor and boot ROM
- Measures 4" x 7.5" with mounting holes
- Floppy controller (1-4 drives, 3.5" or 5.25", single/double density, single/double sided, 40/80 tracks)
- One Centronics parallel printer port
- Two RS-232C serial ports (75-19200 bps)

SB180K-1: 9.216-MHz SB180 single-board computer kit. Includes all components except DRAM \$195.00

SB180K-1-20: Same as above with Z-System hard/floppy disk operating system; BIOS and ROM sources) \$295.00

COMM180K-S: SCSI hard disk adapter board for SB180 \$89.00

Available from: CCI • 4 Park St. • Vernon, CT 06066

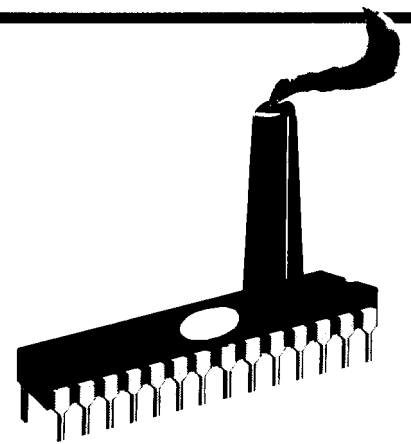
For information and orders: (203) 875-2751 • Fax: (203) 872-2204

All payments should be made in U.S. dollars by check or money order. MasterCard, or Visa. Shipping and handling: surface delivery add \$3 for U.S.; 2nd-day delivery add \$7 for U.S. Call for Canada and air freight delivery elsewhere.

FIRMWARE FURNACE

The True Secrets of Working with LCDs

by Ed Nisley



Many controller applications need a better display than you can get with LEDs, but don't have nearly enough information to justify a CRT. For example, while your new temperature controller should show both the **setpoint** and the actual temperature, it doesn't need full color graphics to get the message across.

Even when you have a CRT, an auxiliary display can come in handy. Suppose your PC application monitors a set of switches. Would a small display near the switches make them easier to use? Or could it provide status or debugging information that doesn't fit on the CRT?

Photos 1 through 3 show three liquid crystal displays. The first is a 16-character display that costs all of \$10 from TimeLine. I bought the 2x20 display (along with a bunch of others) several years ago from a commercial supplier; you can buy a similar unit from Digi-Key for about \$32 (single quantity).

This month's column covers the care and feeding of small alphanumeric liquid crystal displays. I'll illustrate the important points by putting together the hardware and software required to run an LCD from the parallel port of an IBM PC. To make it easy for you to evaluate new units, the code this time around is in Microsoft C for the IBM PC; all you need is a printer port, some wire, and a power supply to get useful results. Of course, this being the real world, my code won't work with every LCD made, but it will work with those that use one of the most common methods of interfacing to the outside world.

Lots of Dots

You maybe familiar with the work required to drive a liquid crystal display panel. There are all manner of timing restrictions, problems with DC bias, connections to a glass sheet, and so on. All in all, it's not a trivial exercise.

Fortunately for us, technology has advanced to the point where all of the ugly details required to drive an LCD panel are hidden inside an LSI chip. The display shown in the photos (and many others on the market) use the Hitachi HD44780 LCD controller, with a few supporting driver chips in larger displays. That means the units all share a common interface to the outside world.

The HD44780 is adaptable to many different display layouts. Unfortunately, the data sheets are not particularly helpful when it comes to explaining the details. Before learning how to

use these displays you must know how they work!

There are two basic types of liquid crystal displays: segmented displays and dot matrix displays. The LCD watch on your wrist uses shaped segments, which are easily visible as the separate bars making up each digit and the glyphs marking AM/PM, ALARM, and so forth. Your new laser printer (or your friend's new laser printer) uses a dot matrix LCD to provide status information.

Shaped segment LCDs have a separate connection between each segment and the controller IC. This is feasible for a few digits, but a watch displaying the date, day of week, and time requires 75 active segments. Squeezing all those connections onto the periphery of the display poses a considerable challenge, as does getting all the wires into the controller IC, so larger displays are correspondingly

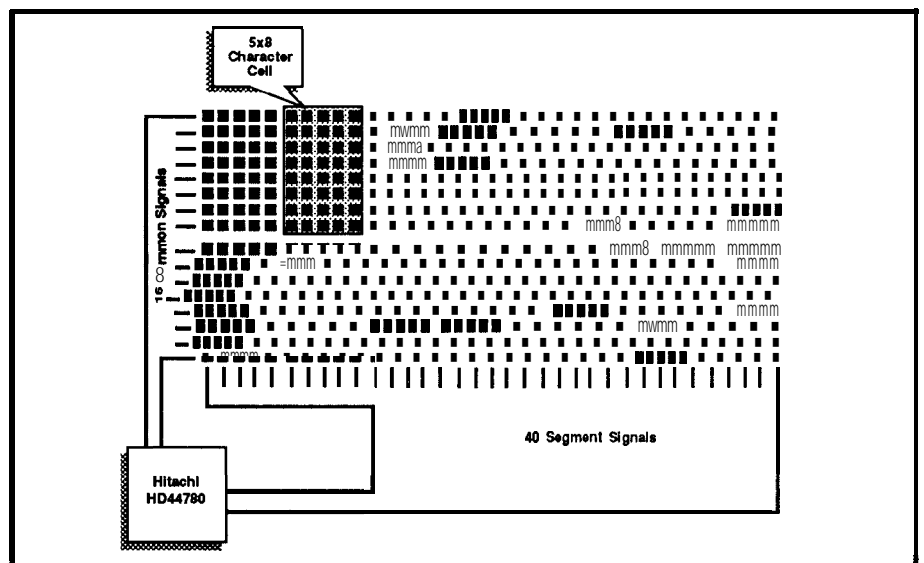


Figure 1a-The HD44780 requires just 56 connections to the LCD to control 280 dots.

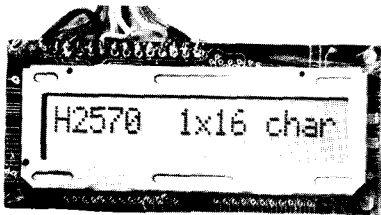


Photo 1

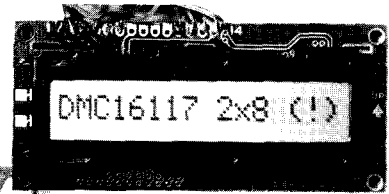


Photo 2

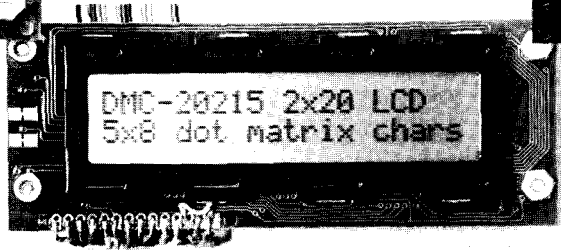


Photo 3

more expensive.

Dot matrix LCDs divide the visible area into horizontal rows and vertical columns. The intersection of each row and column forms a dot which can be turned on or off by sending the proper waveform along each conductor. The controlling IC activates one row at a time and turns on selected dots as needed to construct the image. The displays can be made quite large because the connections to each dot are shared among the whole array.

A single HD44780 controller can handle a display with 16 rows and 80 columns. Figure 1a shows the connections to such a display; a little figuring indicates that 1280 dots require only 56

connections. The regular array pattern also simplifies the task of getting the conductors off the display and into the IC.

The HD44780 is designed to produce characters, so it divides the dot array into groups of cells. A single character is five columns wide and may be either seven or ten rows tall, with an additional row used for an underscore cursor. The 1280 dots can thus hold eight 5x11 cells or sixteen

5x8 cells. A pair of control bits determines the cell size and the number of cell lines.

The 5x11 character cells occupy the upper rows of the array and the lower five rows are disabled. There are two choices for 5x8 cells: a single line across the upper eight rows or two lines using all sixteen rows. The hardware doesn't allow you to select two lines of

on fire, but the HD44780 cannot drive more than 1280 dots by itself. Figure 2 shows how an auxiliary HD44100 driver chip provides 80 additional dot columns with either one or two character lines. The HD44780 passes the column data to the HD44100 over a single wire using a common clock signal, so the connection is almost trivial.

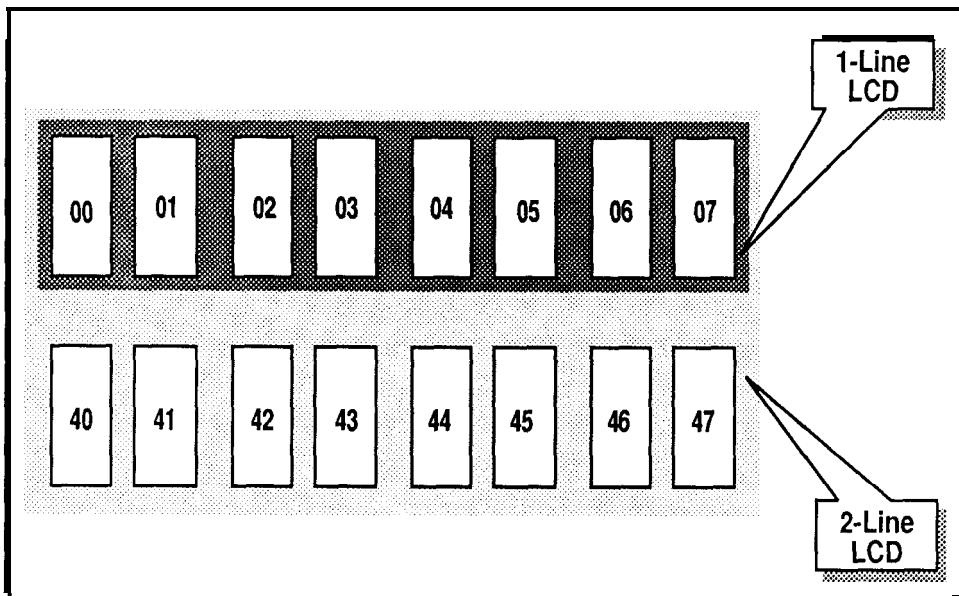


Figure 1 b-Portions shaded in dark gray show the 1-line display layout. The addition of another line results in a 2-line display, shown in light gray.

5x11 character cells; that choice is forced to two rows of 5x8 cells.

The HD44780 has an 80-byte Display RAM to hold the character codes. Figure 1b shows how the addresses in that RAM match up with the LCD array character cells. The second line of a two-line LCD starts at address 40 hex, a fact which causes a great deal of confusion for new users!

A display with two lines of eight characters doesn't exactly set the world

Four HD44100s strung together after an HD44780 display up to two lines of 40 characters. The 80 bytes of Display RAM sets an absolute limit on the display size, so if you see a larger LCD module, it won't have an HD44780 in charge.

Figure 3 shows the Display RAM addresses for some common LCD units. Figure 3a is particularly interesting because it maps the two lines produced by an unassisted HD44780 into what looks like a one-line, 16-character display. Much to the novice's surprise, the eighth character on the display isn't adjacent to the seventh character in Display RAM! The Hitachi manual has a special section explaining this, urn, feature in the Troubleshooting section.

In a similar fashion, Figure 3d shows a 4x20 LCD. Writing data sequentially into Display RAM starting at address 0 fills the first, third, second, and fourth rows in that order. Take careful note of the separation between the last address in the third row and the first address in the second: those Display RAM addresses are not implemented.

If the display has fewer than 80 characters, you can store information in the "off-screen" addresses. Small 8031 applications, with only 128 bytes of on-chip RAM, may be able to make good use of that additional memory. Do, however, refer to Figure 3 to make sure you know which addresses are used in the display. In the interest of simplicity, the interface in this article doesn't allow reads from the chip.

The HD44780 includes a character generator ROM that translates character codes into the appropriate dot patterns for the display. The codes between 20 and 7F hex become the usual ASCII glyphs (with a few oddities thrown in), while the codes above 80 hex are a mixture of Greek, Japanese, English, and mathematical symbols.

The character generator has a RAM section for the eight characters between 00 and 0F hex; space precludes discussing this feature here, but the test program will show you how to create your own characters.

For the rest of this article, I will concentrate on LCDs with one or two rows of 5x8 characters because those are the sizes most suitable for small controller projects. The task of modifying the code supplied with this article is left as a project for the interested reader with an oddball display in hand.

Interfaces Between.. .

The HD44780 reduces the task of using a liquid crystal display into that of building a component into your system. As with any component, you must consider four different interfaces: mechanical, electronic, timing, and programming. Because we are considering the entire display as a single component, each aspect deserves some

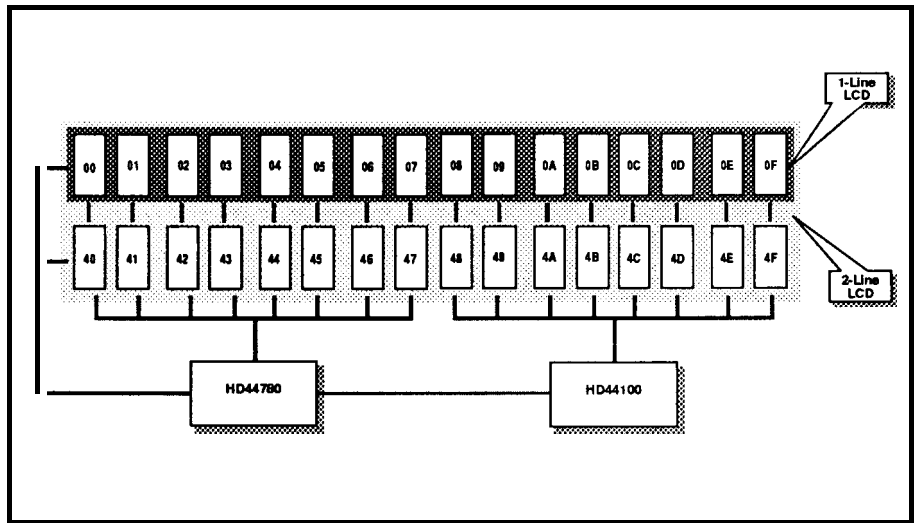


Figure 2-A single-wire connection to an HD44100 is all that's required to add another 80 dot columns to the HD44780's control realm.

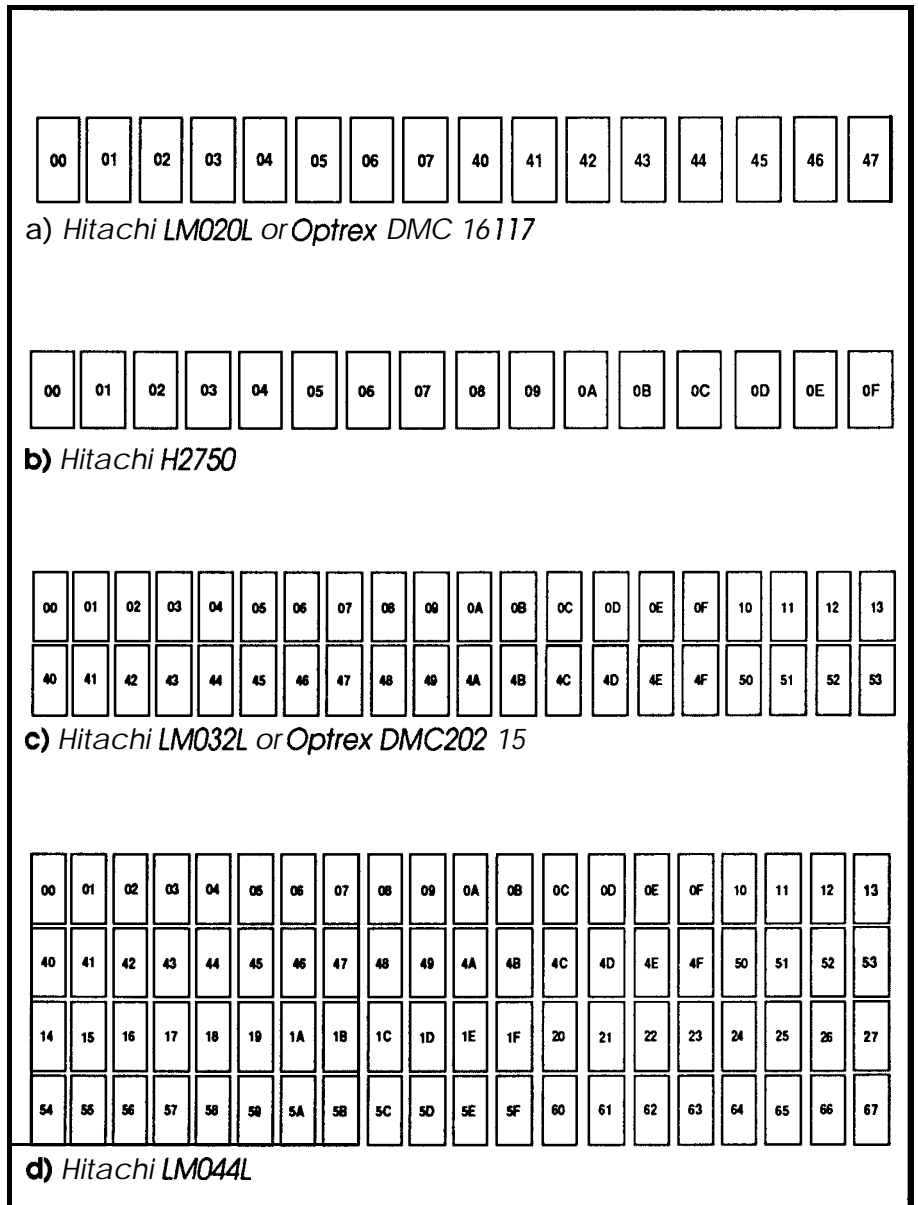


Figure 3-LCD displays are available in countless shapes and sizes.

comment.

The mechanical interface consists of four mounting holes in locations that vary across manufacturers. The dimensions tend to be "hard metric" rather than English, so you may need a metric ruler... or perhaps just bigger holes in the front panel. You must avoid flexing the LCD, but 1/4" stand-offs solve that problem. The 14 edge pads are on 2.5-mm (well, actually 0.1-inch) centers, but a ten-conductor ribbon cable covers all the absolutely essential connections with flex to spare.

The electrical interface is equally simple. The boards require +5 volts at barely a few milliamps. Although the HD44780 is a CMOS chip, all signal lines can use TTL levels. You should avoid very long wires, although I have driven the displays through a 15-foot printer cable with no troubles. If you are trying for a world record distance you should use cable with interleaved grounds to prevent glitches on the control lines.

The voltage on the Vee terminal sets the display contrast. The correct

setting depends on the number of active display rows (8, 11, or 16), ambient temperature, viewing angle, and the phase of the moon. For room temperature applications, connecting the terminal to ground works just fine. Figure 4 shows how to connect a contrast trimpot if you really want one. Note that the data sheets specify the voltage as the difference between Vcc and Vee, so +4.5 volts is actually 0.5 volts above ground with a +5 volt supply.

Wiring an LCD board to a processor is more interesting. Because the HD44880 uses 8-bit characters, you might expect that it needs eight data lines. Many embedded controller systems use low-end micros with a 4-bit data path, so the Hitachi designers provided the ability to transfer eight bits of data in two nibbles of four bits each. The first part of the initialization sequence determines the interface size, which cannot be changed after that point.

Although the Hitachi manual describes how to connect the display di-

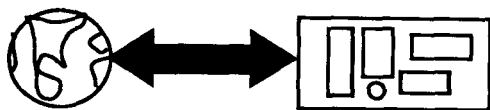
rectly to the processor bus, the HD44780 can't handle the blazing speeds used in contemporary CPUs (or even the lowly 8031). In 4-bit mode, it needs only six lines, so you can drive it from a single parallel port. That port can be an LS374, one byte of an 8255, a few bits from an 8031, or whatever you happen to have lying around.

The HD44780 cannot accept information in a steady torrent. Each data or command transfer causes the chip to become busy for a short time; a transfer made while it is busy may result in aberrant behavior. Although there is a way to read the busy status flag from the chip, a programmed delay works quite well. The C software in this column has enough delay that there is no need for specific delay loops.

I've taken the easy way out with the interface in this column; if you have a parallel printer port on your PC you can experiment with these displays. If whatever you've got isn't an IBM PC, but it does have a parallel port, you can probably shuffle the code

Hobbyists, experimenters, R & D groups!

8031 μ Controller Module



Our 8031 μ Controller Module forms the core of a complete prototype and saves hours of tedious hand-wiring.

We include a prototype quality printed circuit board populated with an 8031 microprocessor, 11.059 MHz crystal, 74LS373 Address Latch, capacitors, communications and I/O headers and a 2764 EPROM with diagnostic software.

The module also contains a socket for a MAX232 level converter to provide an industry accepted serial communications port if your project requires it.

The Developer's Manual provided, describes the module and provides an introduction to programming the 8031.

Send \$39.95 plus \$3.00 S&H per module
Illinois Residents Add 6.25% Sales Tax

Cottage Resources

Suite 3-672, 1405 Stevenson Drive
Springfield, Illinois 62703
(217) 529-7679

STOCKS.....
.....OPTIONS I.....
.....FUTURES

Turn your PC into a MARKET QUOTATION MONITOR

New book covers complete information on financial news and market quotes for your PC from satellite and FM radio.

Topics include:

- Data Encryption
- Password Methods
- Receiver Unit Design

Covers quotation processing and data broadcasting from the trading floor to the desktop, \$19 plus \$2 S/H (includes demo diskette).

Send for FREE catalog of

- DATA RECEIVER KITS
- QUOTE DISPLAY SOFTWARE
- DESCRAMBLING UTILITIES

CALL — (303) 223-2120 (anytime)

DATArx

111 E. Drake Rd., Suite 7041
Fort Collins, CO 80525

around to work with your system. Fair enough?

Figure 4 shows the Centronics and IBM PC printer port connections. All of the strobes and controls are generated in software, so there are no critical bus timings. Getting the timings correct is, of course, a simple matter of firmware.

If you use a female Centronics connector for this project you can plug the display directly into the printer cable. This is handy if you don't want to grub around in the back of your PC, because Centronics connectors use snap fasteners that don't have to be unscrewed. Try to avoid the temptation to print something to the display, though, because it just won't work.

It is worth mentioning that there are no standards for IBM PC printer cable ground connections. The Centronics schematic uses both pins 16 and 19 because two of my printer cables route pin 19 on the PC end to either of those pins. You must verify that your cable and port cooperate to get at least one ground wire to the LCD board. The display will not work with a missing ground!

The photos show what's involved in this project. The power supply is one of those bricks you've seen advertised in the surplus catalogs ever since the Coleco Adam computer went down the tube. It provides -5 and +12 volts as well as +5 volts, but those aren't needed here. I put some tape on the back of the LCD to keep the usual conductive litter on my bench from shorting out the LSI chips. You might want to attach a backplate like the one shown on the 2x20 display in Photo 3.

Passing the Word

Figure 5 presents the relevant sections of the HD44780 instruction set. The instructions to read the busy status bit, Display RAM address, and Display RAM data are omitted because the interface used in this article doesn't permit reads. If you are designing a project requiring data readback, you should consult the Hitachi manuals for details on the exact timing requirements.

Notice how the instructions are dis-

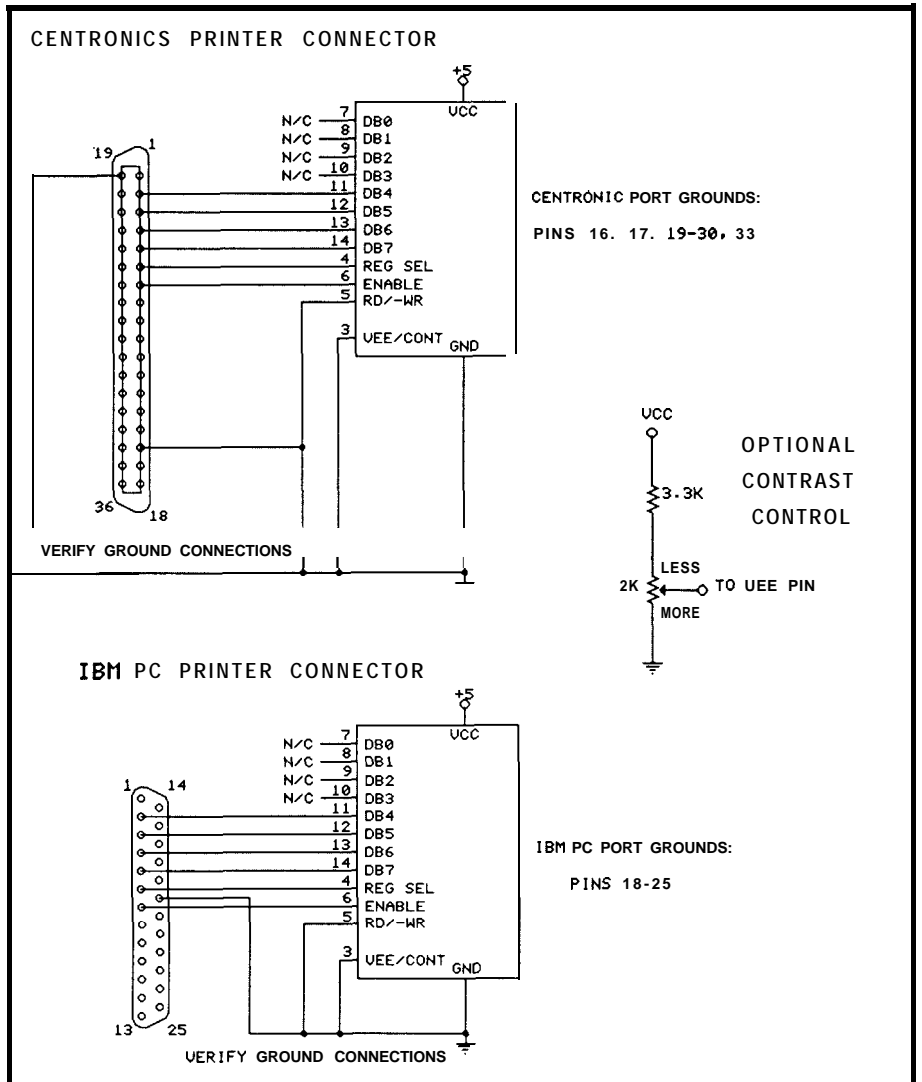


Figure 4—A Centronics printer port makes a convenient interface to the LCD module for experimentation purposes.

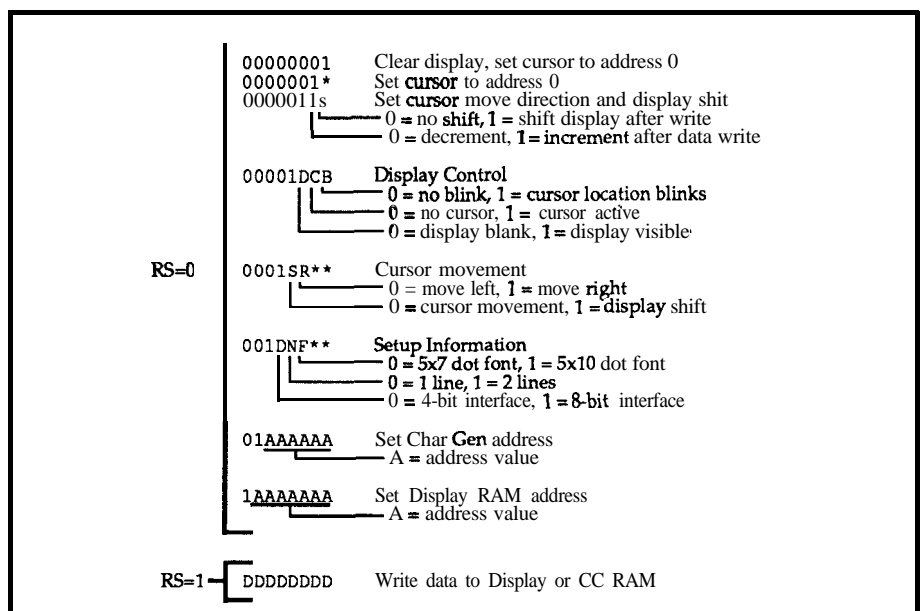


Figure 5—The HD44780's instruction set includes everything necessary to initialize the display, move the cursor, and display characters.

```

/*-----*/
/* Constants, structures, and the like */
typedef struct { /* bit layout in printer port */
    unsigned int data:4; /* data nibble in low bits */
    unsigned int RS:1; /* register select */
    unsigned int E:1; /* enable */
} PORTBITS;
typedef union {
    unsigned int i;
    PORTBITS b;
} PORTDATA;
/*-----*/
/* Send a single nibble to the board */
/* The RS bit gets set up before calling this routine */
/* Interrupts shut off to prevent timing burps during E pulse */
void LCDout(PORTDATA portval) {

    portval.b.E = 0; /* put data in port with strobe off*/
    _disable(); /* ensure no interrupts during this*/
    outp(LCDport, portval.i);
    portval.b.E = 1; /* turn strobe on */
    outp(LCDport, portval.i);
    portval.b.E = 0; /* and back off again */
    outp(LCDport, portval.i);
    _enable(); /* interrupts are OK now */
    return;
}
/*-----*/
/* Send a byte to the display RAM */
/* RS must be set active */
/* High order nibble goes first */
/* There is no minimum time between the two nibbles */
/* Caller must enforce minimum delay between characters */
void LCDchar(unsigned char dbyte) {

    PORTDATA portval;
    portval.b.RS = 1; /* select data */
    portval.b.data = dbyte >> 4; /* do high byte */
    LCDout(portval);
    portval.b.data = dbyte & 0x0f; /* do low byte */
    LCDout(portval);
    return;
}

```

listing 1 — Code to send a single nibble and a full byte to the LCD display.

tinguished by the first nonzero bit. Although I don't know the internal details of the HD44780's architecture, I suspect it is built around a shift register and state machine that can decode bit positions with no extra effort.

The RS bit distinguishes between instructions that control the HD44780 and data destined for the Display or Character Generator RAM. All instructions must be written into the chip with RS=0, while the data uses RS=1.

Because the interface in Figure 4 uses only four bits, the instructions must be transferred in two nibbles of four bits each. The most-significant nibble is written first, followed by the least-significant four bits.

Each nibble is written into the HD44780 by transitions on the ENABLE line. To meet the data setup and hold timing requirements, the data is presented with Enable=0, then ENABLE is brought high and low. In our case, these transitions are made in

software, so the actual timings are much slower than the specifications, but (fortunately!) there are no maximum timing restrictions.

The LCDout routine in Listing 1 shows the steps needed to write a single nibble. Because all of the data and control lines wind up in the same printer port byte, the PORTDATA union allows reference to either the individual bits or the whole byte. Pascal programmers will be familiar with unions, although the standard texts would have you believe that they are unreservedly A Bad Thing.

LCDchar accepts a byte (known as an unsigned char in C jargon), cracks it in half, and calls LCDout to send the two nibbles. There obviously isn't a lot of computation involved in this process, so the pulse lengths and separations are largely determined by the C calling conventions.

A single call to LCDchar produces two ENABLE pulses. On a 20-MHz PS/2 Model 80 with moderate C compiler optimizations, each pulse is 7 microseconds wide and the leading edges are 30 microseconds apart.

Even with these relaxed timings there is no problem updating the display. Listing 2 shows LCDstring, which copies a string into the Display RAM using calls to LCDchar. It turns out that LCDstring writes bytes every 190 microseconds, so 8 milliseconds suffice for a 40-character display.

Your actual mileage may vary depending on processor speed and code optimizations. On a stock 4.77-MHz PC, for example, LCDchar's two pulses are 53 microseconds long and separated by about 260 microseconds. LCDstring can write a byte every 1.6 milliseconds, so it takes fully 64 ms for 40 characters.

That may sound like an awfully long time, but the optical response time of the liquid crystal material is over 100 ms at ordinary room temperature. Even a stock PC can update the display before the syrup knows what hit it! The HD44780 changes the display without interference so there is no "snow" during rapid updates.

Incidentally, those timings are one

reason all the code is in C. Had I cast it in assembler it would need all manner of checks enforcing the minimum time between characters. As it is, you can tinker with the code to improve things quite a bit; start by removing all the `WaitSec(0.0)` calls. When you're done you won't be able to see the difference!

If you use these LCDs in 8031 projects you must pay particular attention to the minimum delays. Make sure you take data sheet in hand before writing the code.

Getting Started

Al though the HD44780 resets itself after the power comes on, the default values are usually not what you want. It selects an 8-bit interface and disables the display, exactly what we don't need for this application. The Hitachi manual details what you must do to reset the chip. What they don't emphasize is that you cannot play fast and loose with the initialization sequence.

The `LCDinit` routine in Listing shows the required setup procedure. The timings listed are minimums, so the large C code delays are quite acceptable. In fact, one way to pin down initialization problems is to single step through the code until you find the instruction that didn't get enough time.

The first three steps may seem silly for our application because they repeatedly set an 8-bit interface with four wires. Remember that the chip may be in an unknown state, with an instruction already in progress when the first byte is written. After three writes separated by the appropriate delays, the chip is in a known state and ready to receive the remaining setup codes.

The choices concerning the interface length, number of display lines, and character font size cannot be changed without going through the whole reset sequence again. The remaining steps may not be quite what you want but they must occur in the specified order. Follow the instructions in the listing and your display should start up successfully.

If all else fails, you will need a copy

```

/*-----*/
/* Send a string to the LCD */
/* Does NOT worry about where the string starts or whether */
/* it's displayable. */
/* Presumes that the string ends in the usual \0 character... */
/* If using the character generator, remember that this */
/* routine won't send out a binary zero because that's the */
/* end-of-string flag. Use characters \x08 through \x0f to */
/* get to the character generator. Minimum time between */
/* characters is 120 microseconds..no problem! */
void LCDstring(char *string) {
    while (0 != *string) {
        LCDchar(*string++);
        WaitSec(0.0); /* minimum delay */
    }
    return;
}

```

listing 2-Code to send a string to the LCD display.

The last column indicates the default values.

```

Firmware Furnace LCD Test Driver

Syntax is LCDTEST <switches>
where the switches are:
-c:x    cursor selection (0 - 3)    0
-f:x    font selection (7 or 10)    7
-l:xx   characters in logical row 64
-p:x    printer port ID            1
-r:x    number of logical rows     1

```

To use LCDTEST with a two-line display connected to LPT2, type in:

```
LCDTEST -p:2 -r:2
```

Figure 6—LCDTEST supports several command-line parameters that cover the more common LCD displays.

of the Hitachi documentation and a scope to make sure you are not violating the timing specs.

Random Displays

If you are building commercial projects, you can obtain LCD units from your distributor. For those of you who want to try out this stuff without going through minimum order hassles, start paying attention to your surplus catalogs, and advertisements from distributors like **TimeLine**, **Digi-Key**, **Jameco**, and **JDR**. Small LCD units are showing up in laser printers

and the like, so the production volumes ensure lots of interesting surplus.

The 16-character display in Photo 1 was advertised by **TimeLine** late last year. They had a few hundred at \$5 each in lots of five units; I bought ten on the off chance they might come in handy. Good planning, huh?

I have seen liquid crystal displays advertised in other catalogs over the years, but their availability is subject to all the usual vagaries of the surplus business. If you are interested in this sort of thing, drop by the Circuit Cellular BBS for up-to-the-minute informa-

ADVERTISER'S INDEX

Reader Service Number	Advertiser	Page Number
101	2500 AD Software	63
102	AISI Research	C4
103	Alpha Products	7
*	Associated Comp.	71
104	AVOCET	C2
106	Baby Computers	19
*	Best Associates	71
107	Binary Technologies	32
108	Bryte Computers	48
109	Cabbage Cases	24
110	Circuit Cellar	55
111	Circuit Cellar	29
*	Collins Associates	71
112	Cottage Resources	59
113	Covox, Inc.	8
114	DATArx	59
*	David Baker Assoc.	71
117	Environmental Optics	54
	Express Circuits	24
118	Galacticomm	C3
120	Grammar Engine	8
121	GTEK, Inc.	16
122	Hazelwood Computer	63
123	High Resolution Tech	45
124	Hogware	38
125	Innotec	66
126	Introl Corp.	41
115/116	JDR Microsystems	26
*	Lear Labs	71
127	Logical Systems	43
128	LTS/C Corp.	27
		63
129	Micromint Resources	61
131	Micromint	33
132	Micromint	69
133	Ming Eng.	28
134	NOHAU Corp.	28
135	PseudoCode	32
136	Quinn-Curtis	14
		43
138	Red Electron Systems	69
		24
140	Sofaid Creek	44
141	Thinking Tools	48
142	Timeline	4
*	Tinney	29
143	x-10	9

IRS INK Rating Service

How useful is this article?

At the end of each article and some features there are three 3-digit numbers by which you can rate the article or feature.

Please take the time to let us, at Circuit Cellar INK, know how you feel our material rates with you. Just circle the numbers on the attached card.

tion on what's hot. [Editor's Note: For information on connecting with the Circuit Cellar BBS, see page 70. For information on how to order Circuit Cellar BBS On Disk, see page 70.1

Because you may not have much choice as to number of rows or characters, the LCDTEST program is adaptable to nearly any display based on the HD44780. Often you won't know which controller is on the back of the board until you get it, but the Hitachi unit is fairly common.

Display Exercises

LCDTEST demonstrates various aspects of the HD44780. While it's not a complete diagnostic tool, if your new LCD works correctly for all of the tests, you can be pretty sure that you've got a live HD44780 (or a clone!) driving it. If not, well, you did get the data sheets, right?

LCDTEST has several command-line parameters that cover the more common LCDs. You can't blow out a display by incorrect programming, so some judicious experimentation will help you understand how the HD44780 does its magic. Wiring the power connections backwards is another matter, of course..

LCDTEST initializes the LCD circuitry, then writes all character codes from 0 through 255 into the Display RAM. It will continue to do this until you tap the space bar, so you can watch the display fill up and determine the starting addresses for each row. The first few characters will be gibberish because the character generator RAM hasn't been loaded yet, so don't get worried right away.

The next step tests the ability to set the Display RAM load address. Several character strings appear at various places on the display; you should check to make sure that they are correct.

Although I glossed over the HD44780's character generator, LCDTEST loads eight graphics characters and illustrates how to display them. It also changes the CG RAM on the fly, a trick that can produce a limited form of animation.

The final demonstration displays

the real time (according to your PC's clock) and a loop counter. The update rate is limited by your PC's processing speed, but you will see that it exceeds the LCD's ability to keep up. Even on a stock PC, the lower two digits are illegible!

Off The Bottom

These little LCD displays are somewhat like peanuts: once you've used one, you just can't stop. As long as you have a microcontroller in your project already, it makes a lot of sense to use a classy display instead of (or in addition to) those chintzy LEDs. □

IRS

- 222 Very Useful
- 223 Moderately Useful
- 224 Not Useful

CCINK's 1st Year Reprints

Our fast rise in circulation has resulted in a virtual sellout of the first year of INK. So as not to disappoint any of our readers, we are offering a B&W offset reprint of CCINK's first year (Issues 1-6). Available for \$20.00 in the U.S. and \$24.00 to Canada and Europe (shipping & handling included).

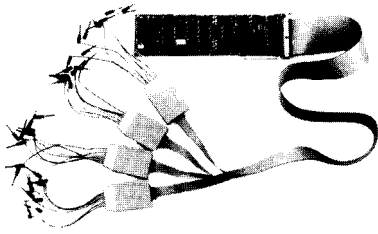
Send check or money order to:
Circuit Cellar INK
1st Year Reprint
P.O. Box 772
Vernon, CT 06066

Visa or MasterCard accepted, call (203) 875-2199

AVAILABLE BACK ISSUES

Issue #6—Remote Video Surveillance • ROVER: Remotely Operated Video-Based Electronic Reconnaissance • Home Satellite Weather Center
Issue #5—ImageWise/PC-The Digitizing Continues • DDT-51 Revealed
Issue #7—Computing in Real Time • ImageWise/PC: The Hardware • Build a Remote Analog Data Logger • Home Satellite Weather Center -Finishing the Firmware for the Peripheral Processor • Writing a Real Time Operating System

Send \$4.00 per issue (includes S&H) in check or money order to: Circuit Cellar INK, P.O. Box 772, Vernon, CT 06066. Visa and Mastercard accepted, call (263) 875.2199.



Introducing

the ID320

Logic Analyzer Card
for the IBM PC/XT/AT

State of the art design brings
you high-end performance at
low-end price

Features:

- 32 channels 2K deep
- 25 MHz state analysis
50 MHz timing analysis
- Variable Threshold
Input Pods
- Single Slot Design
- Multi-level triggering
- Selective Data Capture
- Software Performance
Analysis
- Disassemblers for
popular 8/16 bit
microprocessors
- Test Development
Language supports
ATE applications
- Friendly User Interface
- And much more

from \$1395

Satisfaction Guaranteed or
your money back

**INNOTECH
DESIGN INC.**

4640 Firestone Blvd., Ste. C
La Mirada, CA 90638
Tel: 714-521-5454

```

/*-----*/
/* Initialize the LCD board */
/* The timings are minimums, which we exceed by a gross margin*/
/* if you are porting this to assembly language, you MUST */
/* delay correctly. If not mentioned, the minimum timing is */
/* 120  $\mu$ s between bytes. This code sets up a I-bit interface.*/

void LCDinit(void) {

    PORTDATA pbits;
    unsigned char cmddata;

/* No matter how silly, the following values are REQUIRED. Do*/
/* NOT change the sequence of bytes going to the display. */

    pbits.i = 0;
    pbits.b.data = 0x03; /* 8-bit interface (honest!)*
    LCDout(pbits); /* set 8-bit intf */
    WaitSec(0.1); /* min 4.1 ms delay */
    LCDout(pbits); /* set 8-bit intf again */
    WaitSec(0.1); /* min 100 us delay */
    LCDout(pbits); /* set 8-bit intf again */
    WaitSec(0.1); /* min 1.6 ms delay */

    pbits.b.data = 0x02; /* now set 4-bit interface */
    LCDout(pbits); /* minimum delay */
    WaitSec(0.0); /*

/*- after this point you have a little flexibility */
/*Use ONLY the alternatives described in the comments */
/*Do NOT omit any bytes from the sequence going to the display*/

    cmddata = 0x20; /* 4 bit interface again */
    if (NumRows == 2) {
        cmddata |= 0x08; /* set two lines */
    }
    if (FontSel == 10) {
        cmddata |= 0x04; /* set 5x10 font */

    LCDcmd(cmddata); /* tell the display */
    WaitSec(0.0); /* minimum delay */

    LCDcmd(0x08); /*display off,no cursor,no blink*/
    WaitSec(0.0); /* minimum delay */

    LCDcmd(0x01); /* clear display */
    WaitSec(0.1); /* min 4.9 ms delay

    LcDcmd(0x06); /* 06=inc cursor, no shift */
                /* 04=dec cursor, no shift */
                /* 07=inc cursor, shift */
                /* 05=dec cursor, shift */

    WaitSec(0.0); /* minimum delay */

/*- You may now send whatever you want to the display. You */
/* CANNOT change the interface width or the char font size. */

    LCDdisplay(1,CursorSel); /* turn on disp & select cursor */
    return:

```

CONNECTIME

Excerpts from the Circuit Cellar BBS

THE CIRCUIT CELLAR BBS
300/1200/2400 bps
24 hours/7 days a week
(203) 87 1-1988 — 4 incoming lines
Vernon, Connecticut

Conducted by Ken Davidson

The message base of the Circuit Cellar BBS is now available on disk. See page 70 for details.

Most bulletin board systems rely heavily on local callers to make up a large portion of their user base. And that's fine for an amateur BBS that caters to having the latest video game or utility available in the files area, or to discussions of whether the local high school football team will beat their arch-rival. But when a BBS such as the Circuit Cellar BBS specializes in one particular area, it tends to attract a much wider audience extending beyond state lines and, in some cases, country borders.

Right from the start, the Circuit Cellar BBS has attracted a geographically widespread audience. The majority of our callers are located outside our local calling area and, for that matter, outside the state (and in a state the size of Connecticut, that's not hard to do!).

I've been keeping track of where some of our international callers are calling from, and the list is fascinating. Over the past year or two, we've had callers from Mexico City, Mexico; Verona, Italy; Vienna, Austria; Kingston, Jamaica; Moelln, West Germany; Helsinki, Finland; Bangkok, Thailand; Hong Kong; Sao Paulo, Brazil; Haifa, Israel; Florida, Argentina; Riyadh, Saudi Arabia; and the list goes on. We have quite a few callers who routinely call from Australia. In fact, in just one 24-hour period recently, we had calls from Stockholm, Sweden; Sydney, Australia; Malmoe, Sweden; Paris, France; and Maebashi, Japan.

So the next time you get your phone bill and cry over how long you spent on-line, think of some of your fellow callers ringing from outside the U.S. and imagine what some of their phone bills must be like!

We've all seen large computer-generated displays in sporting arenas and outside banks. This first discussion centers around building one such large display.

Msg#: 9648

From: TIM EMERSON To: ALL USERS

I have a requirement for a large-screen display (4 ft x 6 ft) to display status information in a factory. The ideal solution would be an LCD unit, but I have had no luck in finding one larger than a laptop PC screen. Projection displays are too bulky and provide a lot more resolution than I need to display simple text. Any ideas where I can look?

Msg#: 9649

From: RICHARD ANDREWS To: TIM EMERSON

If you can use a dot-matrix-type display, I may have an idea for you. The Staver Co. Inc. makes displays that are electromechanical and static. They have a disc which is black on one side and yellow, or whatever, on the other side. The unit has two coils; energizing the coils one way turns the black side out; energizing them another way turns the other side out. One nice feature is that the displays are static, so once set, power can be removed from the coils and the disc will stay put. I have seen this type of display used for time/temperature displays on banks, etc. Their number is 516/666-8000.

Msg#: 9651

From: JIM NELSON To: TIM EMERSON

The 1988/89 EEM carries some catalog pages for the Staver Company's Signalex displays. See volume B, pages 1093-1097. They have 7x5 dot-matrix alphanumeric modules which vary in size from 4" wide by 6" high through 13" wide by 18" high. The dot display elements are also available individually. They are static electromechanical affairs, and are controlled by a pair of 12V to 18VDC pulses. One pulse sets a dot to the fluorescent color, and the other pulse resets the dot to the background color. The pulse widths necessary vary from 1 ms at 18 VDC to 2 ms at 12 VDC. The required current for 12VDC pulses is about 700 mA.

No controller is provided; characters or graphics must be built by pulsing the individual dots, which are addressed in an X-Y matrix, on and off. This is a nasty problem, but is straightforward to handle with the right mix of experience and microcontrol and interface tools.

AEG makes some large composite LCD displays. Why do you think these are better for your application than the equally monochromatic electromechanical system I mentioned earlier?

Msg#: 9802

From: TIM EMERSON To: JIM NELSON

Actually, I'm not sure that the LCD display is going to be the best way to go. I really need to get all the data before I decide. Who is AEG and how can I get hold of them? Thanks for your reply. I have found out more through this BBS in a few days than I have through normal channels in weeks.

Msg#: 9868

From: JIM NELSON To: TIM EMERSON

Telefunken Electronics, AEG Corporation, P.O. Box 3800, Route 22, Orr Drive, Somerville, NJ 08876-1269, 201/231-8300.

By the way, here's an oft-used rule of thumb for estimating clear viewing distance as a function of character size: Under the best conditions, and with normal eyesight, viewing distance = character height (in inches) * 50 feet; usually, character height is calculated as 1 inch per 25 feet.

The most-flexible LCD technology is a new product called NCAP film. Great-looking displays may be combined with membrane switches. Thin flexible plastic is first coated with conductive indium tin oxide, then coated with a thin uniform layer of an emulsion of tiny liquid-crystal capsules. This is laminated to a second indium-tin-oxide-coated plastic sheet which has been photochemically etched to leave conductive areas corresponding to the pattern elements to be displayed. No polarizers are used, so its contrast ratio is very high, and there are not the severe limitations on viewing angle of older glass **LCDs**. Both **transmissive** mode backlighted NCAP and reflective mode NCAP displays with colored reflectors may be combined with graphic overlays. Power consumption is down around **9 mA** per square foot.

Someday I'll have some wallpaper made out of this stuff.

I am currently involved with two projects incorporating flat (LCD) displays. Utilization of the technology has a lot of parallels in video, which seems a thread in one way or another through 90% of what I've been doing for the past six years.

Msg#:10208

From: PELLERVO KASKINEN To: TIM EMERSON

You might also consider one of the electroluminescent displays that you see in most European airports. They are made by **Finlux**. Contact them at 408/725-1972.

As another point: AEG was in financial problems a few years ago and was saved by being partly bought by Daimler Benz. Now they are in an upswing, considering a purchase of some **multibillion** dollar enterprises. If that goes through, they may be the largest industrial enterprise in the whole world. Other than that, they make anything you might want from hair dryers and power drills to nuclear power plants. Their name pretty much says the same to the Germans that a GE says to the U.S. population and incidentally, translates to "General Electric Inc." or something like that!

If you redly want to start a religious war, fry telling someone his favorite programming language is for the birds and some other language is the best. At least the following discussion was kept civil while the merits of various languages used for microcontroller programming were debated.

Msg#: 9954

From: DAVE FILICICCHIA To: ED NISLEY

I have been using the 8051 for some time now and started with learning **PL/M51**. I use it at work and have enjoyed its flexibility. I want to do some home projects using a high-level language with the 8051, but all the software packages are so expensive (**\$750-\$1000**). Do you know of any high-level language package for the 8051 that can be purchased for a more reasonable price (like around \$200 or so)? It would be great to stick with **PL/M**, but C or maybe even BASIC would be OK. Thanks.

Msg#: 9991

From: ED NISLEY To: DAVE FILICICCHIA

Beats me...

Avocet advertises both C and Pascal, but they're both in the **high-price** end of the market.

I'm not entirely convinced that high-level languages are a Good Thing for single-chip micros. After all, they depend on architectural features that microcontrollers (at least 8051s) don't have: big stacks, lots of registers, orthogonal instruction sets, lots of address space.

The tack we've been taking on Circuit Cellar projects is to put as much function in the PC code and as little in the 8051 code as possible. The principle is to make the 8051 code simple, because writing and debugging it is such a pain-as you're well aware!

Msg#:10068

From: DAVE FILICICCHIA To: ED NISLEY

Actually, I have found **PL/M** to be a pretty reasonable language. It seems to have a lot of parallels to C without being too cryptic. Table look-up becomes **easy** and even pointers aren't too difficult to use. I've done a few designs that use the 8051 as a machine controller that didn't require any external RAM.

After just learning Turbo C, I still prefer **PL/M** for now because it has a bit more structure and forces you to declare variables in a more obvious manner than C. Also, I find using keywords to be helpful even though it requires more typing. Constructs in C that use the **"?"** in the special case of IF are confusing to me as a beginner C programmer, and the IF-THEN-ELSE of **PL/M** is rather straightforward. I guess we all tend to use what is most comfortable. Thanks.

Msg#:10078

From: ED NISLEY To: DAVE FILICICCHIA

Yeah, even if it wasn't comfortable when you got started, you can put up with almost anything else rather than learn something new. I've had this slap me a couple of times so far..

Msg#: 9968

From: JIM NELSON To: DAVE FILICICCHIA

Bryte Computers sells a Forth for the 8031 for \$100. Their Maine phone number is 207/547-3218. This Forth will be 10 times faster than BASIC on the 8031. Of course, if you can't respect such an inexpensive product, you can buy 8051/31 ChipForth from FORTH Inc. for \$3750.00 or from MPE for 225 pounds sterling.

Bryte Forth is supplied in an 8K EPROM. The Forth assembler is included along with a 130-page manual. Minimum RAM required is 2K bytes; program and data memory spaces must be mapped to the same physical memory devices.

Forth has been called a programming amplifier-it multiplies the power of good programmers. It has the interactivity that BASIC programmers embrace. This is combined with the speed and Lisp-like extensibility necessary to allow you unequaled access to and control of microcontroller hardware.

Msg#: 9984

From: GARY LEAR To: JIM NELSON

Jim's message sounds like that of a true-blue Forth man (we have one at work). There are times, however, when the microcontroller is already running with pins to the wall and ***only*** assembly will do (or else add some more hardware). Forth has its place as a tight, self-contained operating environment, but it

takes a completely different mindset to program in it at all effectively.

Msg#:10031

From: JIM NELSON To: GARY LEAR

Ray Duncan wrote an article on Forth for the November/December issue of "Programmers Journal." He begins with these quotations:

"I have yet to see a decent piece of software written in Forth. Let's face it, Forth stinks." John Dvorak

"Forth is the first language which has been honed against the rock of experience before being cast into bronze." Charles Moore

"Only brain-damaged programmers use Forth." Allen Holub

"Forth is like the Tao. It is a Way, and is realized when followed. Its fragility is its strength, its simplicity is its direction." Michael Ham

Rather than finding myself bound by a mindset, I have found a great freedom in using Forth. What isn't widely understood is that a good Forth programmer can command, in a single application, the entire range of machine level to fourth-generation software engineering technology. I used Forth tentatively for over a year before I became aware of the mindsets consequent to

Powerful, Low-Cost Data Acquisition and Control with Commodore C64 & C128



80-line Simplified Digital I/O Board with ROM cartridge socket
Model SS100 Plus \$129. Additional \$119.



Original Ultimate Interface
Universally applicable dual 6522 versatile interface adapter board.
Model 64IF22 \$169. Additional \$149.

16-Channel, 8-bit analog-to-digital conversion module.
Requires model 64IF22. Model 64IF/ADC0816 \$69.

Interface boards include extensive documentation and program disk. Manuals available separately for examination. Call or write for detailed brochure.

Resources for Serious Programmers.

- Symbol Master Multi-Pass Symbolic Disassembler. C64 & C128. \$49.95
- PTD6510 super-powerful Symbolic Debugger. C64. \$49.95
- MAE64 6502/65C02 Macro Editor/Assembler. \$29.95
- C64 Source Code Book. Kernel and Basic ROMs. \$29.95

SCHNEDLER SYSTEMS

Dept. C, 25 Eastwood Road, P.O. Box 5964
Asheville, North Carolina 28813 Telephone: (704) 274-4646

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 new BCC52-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' 27841 128 EPROM programmer, 3 parallel ports, a serial terminal port with auto baud rate selection, a serial printer port, and it is bus compatible with the full line of BCC-bus expansion boards. The BCC52 bridges the gap between expensive programmable controllers and hard-to-justify price-sensitive control applications.

BASIC-52's full floating-point BASIC is fast and 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.

Since the BASIC-52 is bus oriented, it supports the following Micromint expansion boards in any of Micromint's card cages with optional power supplies:

- | | |
|---|--|
| BCC22 Smart terminal board | BCC53 Memory and 6-port I/O exp. board |
| ADP500 User vocabulary, digitized speech board | BCC13 8-Channel 8-bit A/D converter |
| BCC25 LCD display board | BCC30 16-Channel 12-bit A/D converter |
| BCC33 3-port I/O expansion board | BCC18 Dual channel serial I/O board |
| BCC400 8-Channel optoisolated I/O expansion board | BCC55 Prototyping board |
| BCC40R 8-Channel relay output board | BCC45 Stepper Motor board |

BCC52 BASIC-52 Controller board \$189.00

BCC-SYST.5 '52 PAK' Starter System \$449.00

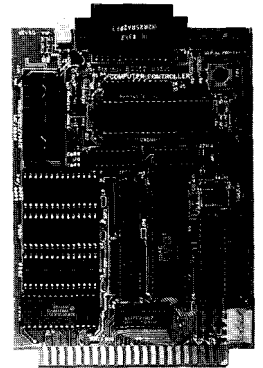
Includes: BCC52, ROM A&B UTIL., CC01, MB08, UPS10

BCC52 OEM 100 Quantity Price \$149.00

BCC52C Lower power all-CMOS version \$199.00

Note: The BCC52 series is available in Industrial Temperature Range, fully tested. Prices start at \$294.00 single qty. Call for OEM pricing.

Micromint, Inc - 4 Park Street, Vernon, CT 06066



To Order Call
1-800-635-3355
Tel: (203) 871-6170
FAX: (203) 872-2204
TELEX: 643331

my experience in traditional languages.

Msg#:10121

From: GARY LEAR To: JIM NELSON

I have written a few small programs in Forth in an attempt to grasp the differences in overall philosophy. So far I have had only minimal success (I assume that you are familiar with Leo Brodie's two books, "Starting Forth" and "Thinking Forth," at least one of which was written at Forth Inc.). So far I have the same complaint that I have with C: In an attempt to be concise, the language is excessively terse. This may be contrasted with Pascal, which is a heavily "typed" language (which means two entirely different things to beginning **and** experienced Pascal programmers!). I still favor assembly for fast control applications, but this may be attributable to my hardware origins.

We often get questions on the BBS about where to find parts for projects. The following is a list of suppliers that we and others have had experience with in the past. Do note the cautions concerning the quality of components from surplus suppliers, however.

Msg#:11176

From: RICHMOND ARMSTRONG To: ALL USERS

Is there anyone who can supply ICs, sockets, and other discrete components in small quantities for experimenters? Any suggestions for places to look are greatly appreciated. Thanks.

Msg#:11202

From: KEN DAVIDSON To: RICHMOND ARMSTRONG

This is just a partial list of suppliers of both electronics components and surplus parts and equipment. Most have shipping charges which depend on the size of the order, and some have minimum order amounts, but all sell parts in single quantities. If anyone has a favorite supplier not on this list, let me know and I'll add it.

ICs, Discretes, Sockets, Switches, etc.

Digi-Key Corporation
701 Brooks Ave. South
P.O. Box 677
Thief River Falls, MN 56701-0677
(800) 344-4539
(218) 681-6674
Easylink: 62827914
Telex II: 9103508982 "DIG1-KEY CORP"
Fax: (218) 681-3380

Mouser Electronics
2401 Hwy. 287 North
Mansfield, TX 76063
(800) 346-6873
(800) 992-9943 (to order a catalog)
(201) 328-3322
Fax: (817) 483-0931

Jameco Electronics
1355 Shoreway Rd.
Belmont, CA 94002
(415) 5928097
Telex: 176043
Fax: (415) 592-2503

JDR Microdevices
110 Knowles Dr.
Los Gatos, CA 95030
(800) 538-5000
(408) 866-6200
Telex: 171-110
Fax: (408) 13788927

The following list of surplus suppliers comes from the pile of catalogs we had on hand. Quality and availability of parts from surplus houses can vary considerably, and we make no claims for or endorsements of any of the suppliers listed here. We welcome comments, both good and bad, from any users who may have experience with any of these companies.

Surplus Parts and Equipment

Marlin P. Jones & Assoc.
P.O. Box 12685
Lake Park, FL 33403-0685
(407) 8488236
Fax: (407) 844-8764

H&R Corporation
401 E. Erie Ave.
Philadelphia, PA 19134-1187
(215) 426-1708
Fax: (215) 4258870

Jerryco, Inc.
601 Linden Place
Evanston, IL 60202
(312) 475-8440

Edlie Electronics
2700 Hempstead Tpke.
Levittown, NY 11756-1443
(800) 645-4722
(516) 735-3330

John J. Meshna, Inc.
19 Allerton St.
Lynn, MA 01904
(800) 637-4627
(617) 595-2275

Sintec Company
28 8th St., Box 410
Frenchtown, NJ 08825
(800) 526-5960
(201) 996-4093

R&D Electronics
1202H Pine Island Rd.
Cape Coral, FL 33909
(216) 621-1052

The Circuit Cellar BBS runs on a 10-MHz Micromint OEM-286 IBM PC/AT-compatible computer using the multiline version of The Bread Board System (TBBS 2.1M) and currently has four modems connected. We invite you to call and exchange ideas with other Circuit Cellar readers. It is available 24 hours a day and can be reached at (203) 871-1988. Set your modem for 8 data bits, 1 stop bit, and either 300, 1200, or 2400 bps.

IRS

- 225 Very Useful
- 226 Moderately Useful
- 227 Not Useful

SOFTWARE and BBS AVAILABLE on DISK

Software on Disk

Software for the articles in this issue of Circuit Cellar INK may be downloaded free of charge from the Circuit Cellar BBS. For those unable to download files, they are also available on one 360K, 5.25" IBM PC-format disk for only \$12.

Circuit Cellar BBS on Disk

Every month, hundreds of information-filled messages are posted on the Circuit Cellar BBS by people from all walks of life. For those who can't log on as often as they'd like, the text of the public message areas is available on disk in two-month installments. Each installment comes on three 360K, 5.25" IBM PC-format disks and costs just \$15. The installment for this issue of INK (April/May 1989) includes all public messages posted during January and February, 1989.

To order either Software on Disk or Circuit Cellar BBS on Disk, send check or money order to:

Circuit Cellar INK — Software (or BBS) on Disk

P.O. Box 772, Vernon, CT 06066

or use your MasterCard or Visa and call (203) 8752199. Be sure to specify the issue number of each disk you order.



STEVE'S OWN INK

Smile When You Call Me That

Like most engineers, I enjoy the opportunity to investigate new technology and utilize it where it seems appropriate. Having to produce 12 monthly design-oriented projects for the last 12 years was more than ample incentive to build some truly wild things. My house now sports a complete electronics lab, a computer-controlled wood stove, electronic lighting control, and a surveillance system that rivals the National Security Agency.

Unfortunately, the next step in career evolution for me, again like most engineers, is to move from pure engineering to management. According to Circuit Cellar INK's reader demographics, a lot of you are in the same boat with me. Fully a third of the readers define their training as engineers, yet further designate that their current function is management. I remember a long time ago when I worked for one of the industrial giants, the joke down in the pits was that the best way to destroy a good engineer was to make him a manager.

So, what are my observations a year after this career modification?

As with any job, the proper solution to a management assignment is part intuitive ability and part logically defined physical exertion. Simply stated, management success is usually a combination of proper task definition and delegation.

This is not a consulting course on management training, and I don't wish to bore you with the obvious. Virtually all engineers have little problem defining tasks. In fact, they are better suited to it than most professions. Where they have greatest difficulty, however, is delegation. Engineers generally think in terms of tasks for themselves, and solutions that they intend to accomplish. Yes, it was a joke that putting an engineer in management produces both a bad manager and a useless engineer, but it is not without some basis in fact. We train engineers to be independent fountains of knowledge with little emphasis on communication and people management skills. When an engineer tries to apply traditional engineering techniques and methods in a pure management situation, he soon finds that logic and opportunity are often in conflict.

There are two ways to learn things in life: trial and error, or someone tells you. You might think that after spending a solitary existence in a cellar for 12 years that I'd be at a disadvantage. Be advised, however, that my alleged status as a "hermit" is more myth than reality. I surely don't have to fail first hand to get the message that different tactics are required when you direct a magazine rather than just write for it.

I discovered that the proper place for me is managing the direction of Circuit Cellar INK, not managing the everyday tasks of publishing it. Fortunately, I had no problem attracting the right people. The staff of Circuit Cellar INK reads like a who's-who of publishing. I'd like to think that it was the result of my personal charisma, but in reality these people are here because they have a common interest in producing the best computer applications magazine around. With people this motivated, managing objectives is easy.

So, in actuality there really hasn't been any metamorphosis for me, just a modification in time allocation. I don't spend as much time in the cellar as I once used to, and I don't just design projects to be published. My responsibilities now include communicating and promoting our message more. I still warm up the soldering iron to work on my own projects but I am more inclined now to direct the Circuit Cellar INK engineering staff to work on one of my ideas rather than doing it personally.

Sigh... I guess I have graduated into being a manager. Regardless of that fact, my first love is engineering and the contents of Circuit Cellar INK will always reflect that reality. Yes, I've ascended to management. But you'd better smile when you call me that!

SteveCiarcia