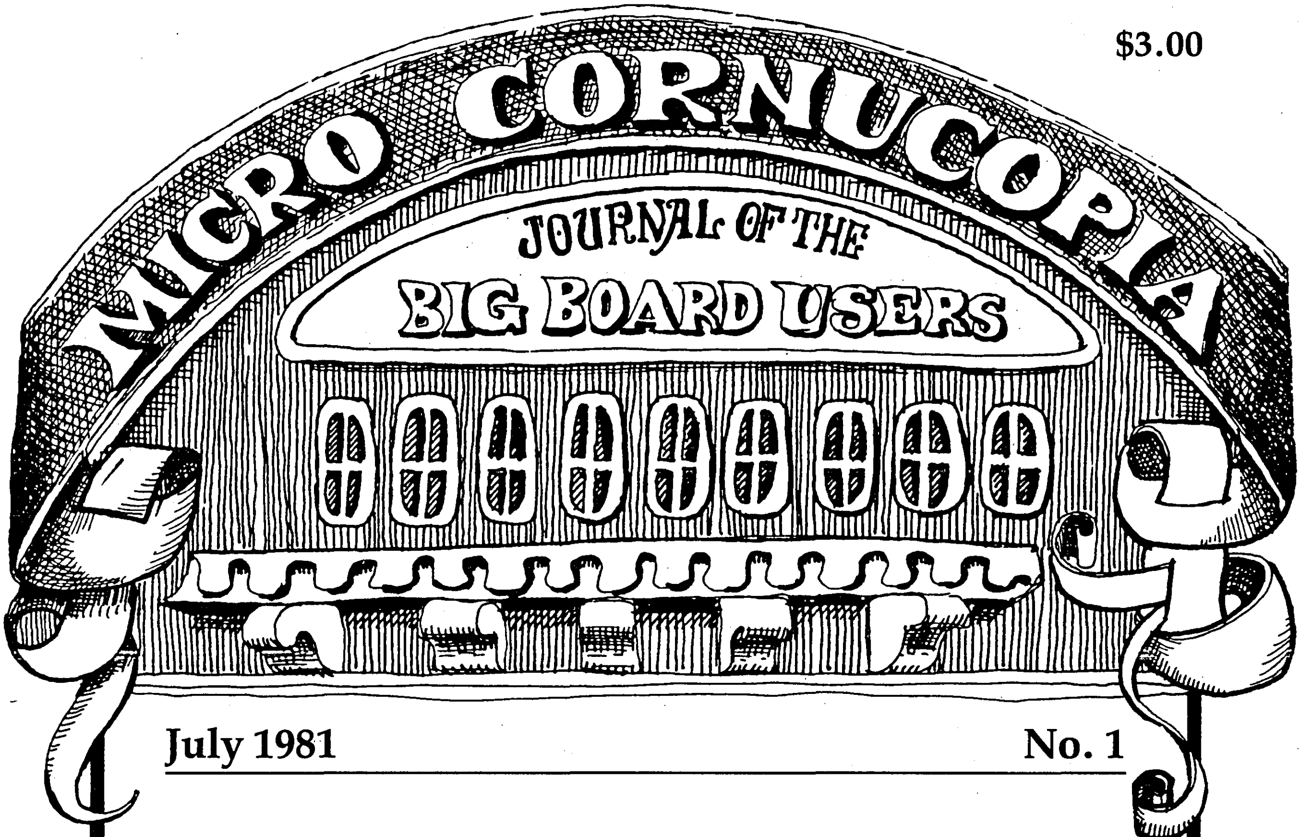
$3.00

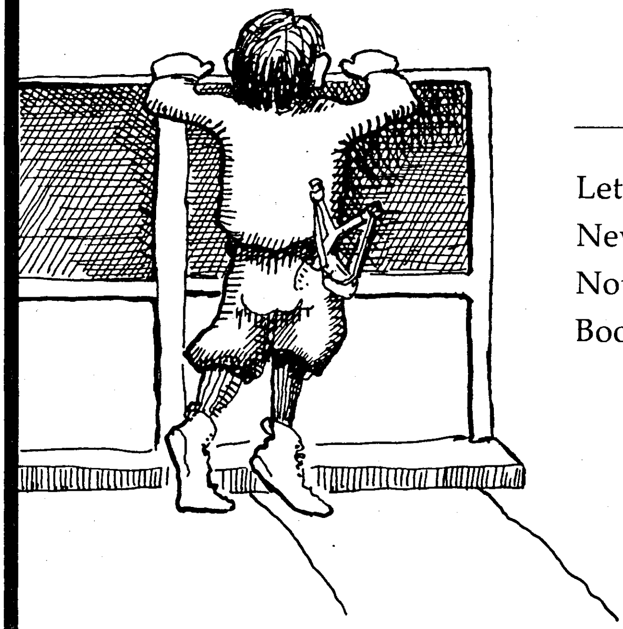


MICRO CORNUCOPIA

JOURNAL OF THE BIG BOARD USERS

July 1981 No. 1

TABLE OF CONTENTS



REGULAR FEATURES

MICRO CORNUCOPIA

July 1981 The Journal of the Big Board Users No. 1

MICRO CORNUCOPIA
11740 N.W. West Road
Portland, Oregon 97229
503-645-3253

Editor & Publisher
David J. Thompson

Technical Editor
Ruth Fredine-Burt

Graphic Design
Sandra Thompson

Typography
Patti Morris & Martin White
Irish Setter

Cover Illustration
Gerald Torrey

MICRO CORNUCOPIA is published six times a year by Micro Cornucopia of Oregon, 11740 N.W. West Road, Portland, Oregon 97229.

SUBSCRIPTION RATES:

1 yr. (6 issues)	$12.00
1 yr. (Canada)	$15.00
1 yr. (other foreign)	$20.00

All subscription orders payable in United States funds only, please.

ADVERTISING RATES: Available on request.

CHANGE OF ADDRESS: Please send old label and new address.

SOFTWARE, HARDWARE, AND BOOK VENDORS: Micro Cornucopia is establishing a group of reviewers. We would very much like to review your Big Board compatible products for Micro C. Please send material to Review Editor, Micro Cornucopia.

WRITER'S GUIDELINES: All items should be typed, double-spaced on white paper or better yet, on disk. (Your disk will be returned promptly.) Payment is in contributor's copies.

LETTERS TO THE EDITOR: Please sound off.

Welcome to the Premier Issue!

Hi, Y'all!

It was hard to imagine what this magazine would even look like on March 15th when we decided to start a publication supporting the Big Board. And now it's really exciting to see it take form.

Starting a new magazine is kind of a scary thing. You need interesting things to put in magazine so people will want to read it. You need people willing to take a chance and subscribe to a new publication, sight unseen. You need lots and lots of hours alone, staring at a video monitor, trying to generate ideas and direction. You need people who are willing to donate time and ideas to a dream. And you need a wife who is not only understanding but who does graphic design, accounting, paste-up, technical illustrating and schematic drafting. So thanks to all you folks, I get to say "Welcome."

Our typesetters, Patty Morris and Martin White are super people to work with (they are getting a Big Board to use for text editing). And Ruth, our technical editor is probably as excited as anyone about Micro C.

Then there are the people who have already submitted material for publication. I talked to Don Retzlaff while I was still deciding whether or not to jump in. His excitement about a user's group and his offer to write some very interesting things really made a publication look feasible. Don's first article appears in this issue. Thanks Don.

John Jones wrote such interesting things on his subscription form that I had to call him. He has a number of useful utilities, including the disk formatting program in this issue. More from John in future issues.

Plus, I have just received a really incredible disk from AB computers including a complete hardware and software interface for minifloppies, a reverse video cursor, and more. Stay tuned, because these super people, and you, are doing some great things with the Big Board.

David Thompson

David Thompson
Editor & Publisher

Letters

Dear Editor,

I am thinking of using one parallel port as an address bus to tell peripherals when to access the other parallel port. One bit would set the direction and then seven bits would remain to address up to 128 peripherals. These could include A/D's, D/A's, plotters, CRT vector graphics, and so on. I would like to see a standard scheme so we can trade designs within the group.

Frank Gentges
9251 Wood Glade Dr
Great Falls, VA 22066

Editor's Note:

I think Frank has an excellent idea. In fact, how does everyone feel about using port A for data and port B for address and control? Bit 7 (PB7) on port B could be the control bit. What say?

What would be super now, would be for someone to write a simple little general purpose parallel port driver that would reside up with the PFM monitor and could be called via the CP/M punch or directly. If someone did such a thing, it would run in the September issue, guaranteed.

And, if someone came up with a latch for translating 8 bits of port A into 16 bits of address and 8 bits of data why there'd be the start of a PROM burner or an S-100 bus interface etc.

Dear Folks,

I would like to locate Jim Rea, designer of PolyVue/80 or Micro Concepts the outfit that marketed Poly Vue. Has anyone done a modem interface for SIO port A? Or, has anyone configured Modem7 from the CPMug for the Big Board?

The Editor.

Dear Editor,

Why doesn't "clear to end of screen" work on the three boards I've seen?

Cole Chevalier
17862 Fitch
Irvine, CA 92714

Dear Editor,

I need: (1) modem driver for BB, (2) parallel printer driver, (3) to contact other users in my local area.

Daryl Coulhart
532 Lake Bayview Ct
Shoreview, MN 55112

New Products Received at Micro C

VEDIT—Text editor.

I have Vedit up and running on my Big Board and once you figure out a couple of idiosyncrasies it is easy to customize and install. Get the CRT version rather than the memory mapped and just follow the directions for the ADM-3A.

However: Do not enter "Carriage Return" for the "COMMON 2ND CHARACTER IN THE ESCAPE SEQUENCE." The only character I've found that works is ESC (again). After this you have to use ESC W or something rather than ESC ESC to leave visual mode, and for some reason you have to use the default for the "command iteration brackets." These brackets are ⟨ and ⟩ rather than [and] by the way.

Once you have it up and running, however, it is a small (10K), but very powerful text editor. (I am using it now to do my text editing).

SMALL C and SMALL C+

If you want to get your feet wet in C and still generate source code that will run on PDP-11s running Bell Labs' C, then these two packages are worth considering. I purchased Small C from the Code Works, Box 550 Boleta, CA 93017. I mean, $15 for a CP/M disk—how could I go wrong? It is neat, kind of like starting out using integer basic. Plus, it is public domain! Several of the fellows at Tektronix are working on it now, doing some optimizing, etc. The printed document is pretty minimal but when combined with the book, "The C Programming Language" by Kernighan and Ritchie, it is sufficient. The source for Small C, also written in Small C (it compiles itself) is also on the disk. Small C generates assembly code which can be assembled by ASM.

I picked up Small C+ at the Computer Faire from Alpha Omega Computer Systems. P.O. Box U, Corvallis, OR 97330. They say they have fixed numerous bugs in Small C and have added for-loops, do-while, and case statements, among other things. Small C+ requires M80 and L80 to compile the assembly code it generates.

Since small C+ is also public domain, I plan to make it available as part of a group exchange disk. Small C+ also compiles itself and can be compiled by the original Small C. The source and the documentation are on the disk. Two programmers at Alpha Omega did the extension pretty much as a personal project and I hope to talk to them about Small C+ in the near future.

PASCAL/MT+

I learned Pascal on a big system, I mean a BIG system (60 bits/word), and after using some of the small subset languages commonly available for micros (Small C, ALGOL/M, . . .) I didn't really expect much more than a usable subset of Pascal. I was wrong. Pascal/MT+ is playing with a full deck.

I have tried it on some small "gee I wonder if it will" type programs, and it did. Hopefully I will have a chance to look at it more thoroughly in the near future. Manual and all, it is an impressive package. MT Microsystems has also put out an editor and debugger package to use with Pascal/MT+ (I've heard). If it is anything like the language package, the combination should be hard to beat for someone doing serious application programming. Contact MT Microsystems, 1562 Kings Cross Dr., Cardiff-by-the-sea, CA 92007.

Crowe Z80 Assembler

Byte's Nybbles made available a Z80 assembler by Patrick Crowe. The assembler uses standard Zilog Z80 mnemonics as defined in the "Zilog Z-80 Assembly Language Programming Manual." Byte originally made this program available for $4.00 as a printed listing. I'm checking now to see if it is still available or if we can make it available, this time on disk instead of as a 60-page listing.

What makes this piece of software particularly interesting is that John Jones did the I/O linking for the Big Board and has supplied the source of that. And it works very well. More about all this as I get information from Byte. (All kinds of exciting things! Thanks, John.) ■ ■ ■

Notes From Garland, Texas

Now for the news you have all been waiting for, the latest, greatest from Digital Research Computers.

New ROMs for old.
Jim Tanner is now shipping the Big Board with character ROMs created by yours truly. And, he will reburn (for free) any of the old style upper case and smaller upper case ROMs you send him. If you can't part with your old character ROM for a few days then send him $10.00 and he will send you a new ROM.

New video rocks for free.
For those of you who haven't appreciated the wiggle you get on the video display, here's relief. (No, you don't have to give up drinking.) Any registered owner who sends in his serial number and date of purchase to Jim will receive, free, a 13.9776 MHZ crystal. Take out the old 14.318 video crystal and replace it with the new one and the wiggle will be gone. Not even a genie could do better than that.

4 MHz the easiest way of all.

- Step 1. Remove U96
- Step 2. Jumper what was pin 4 of U96 to pin 4 of U97.
- Step 3. DON'T replace U96.

That's it, no crystals to buy and no board runs to cut. However, it won't work on all boards because of the precharge requirements on the RAM.

First of all, you probably need 200ns RAM chips. Big Boards have been shipped with 300ns, 250ns, and 200ns chips. About 40% were 300ns, 40% 250ns, and the other 20% were 200ns. This mod generates a clock that is more like 60/40 rather than 50/50 High/Low so even the 200ns RAM is just barely making it.

Out of three boards that they have modified at Digital Research two worked and one didn't, though they all had 200ns RAM. On most of the boards it is pretty easy to tell how fast the RAM is. The number on the chip will be 4116-X where X is probably 20, 25 or 30. 20 stands for 200ns, 25 stands for 250ns and 30 stands for 300ns. The National chips have a -4

(continued next column)

Notes from Garland continued

for 250ns and a -3 for 200ns. Any others you should look up in a parts book.

If you are among the folks who have done a successful mod to speed up the Big Board, please send it in and I'll publish it (for those of us who don't have 200ns RAM or can't get this mod to work). In fact, if I get 20 different mods for speeding up the Big Board, I'll publish them all. Why not?

Double double density density.

Jim has someone working on a three-chip board which will plug into the 1771 socket. It will do single and double density on 8 inch and mini floppies (according to Tanner). I would guess that they are aiming for availability sometime late summer or early fall but no one's making any promises.

The chips will be Western Digital and the main controller will be the 1795. (Hooray, it's NOT the 1791.) Perhaps those of you struggling with the idiosyncrasies of the 1791 should write to Western Digital for a new data book.

■ ■ ■

Contributing to Micro Cornucopia

How do you contribute to Micro C? What are we interested in? What should you send, disk, printer output, post card, papaya leaf? What if you can't write? What if the thing you are doing is pretty basic or maybe too advanced? Well, here is the information.

Form: Send articles on paper, (double-spaced) or, even better, on disk. If you send a disk, we will copy the contents of the latest Big Board user's group disk onto your disk before we return it.

It's easier on us if you don't include any formatting characters in the text. These characters may help your text formatter but they have to be removed before Patti and Martin can typeset the article.

Programs: Here a disk is a super way to go. Please include at least a few paragraphs of introduction. If the program requires compiling or assembling please include a COM file along with the source. And if the compiler or assembler is public domain please include it and anything else needed to do the compilation. Most of the software contributed will be placed in a group disk and made available to everyone in the group.

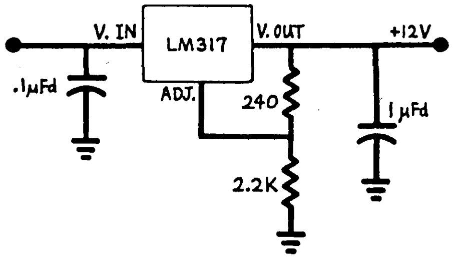
Personal information: Please include some information about yourself (like raising bees and running your big board off wind power) and about how you are using the Big Board.

What to write about: We're looking for anything on the following list, along with just about anything not on the following list.

- **Hardware interfacing,** complete with schematics (we can redraw them if it's needed) and comments about what the circuit does and how it does it.
- **Software drivers** or other mods to the operating system. This time include a listing, etc. (See "Programs" above.)
- **Reviews of software** take a critical look at how easy it is to learn, how powerful it is, and how easy it is to use once you've learned it. Note: part of the user interface is determined by the quality of the documentation and part by the structure of the software.
- **Reviews of languages** take a critical look at the language for particular applications, systems, etc. What are its weaknesses (size, speed) and it's strengths (floating point, string manipulation, documentation, for instance). The primary languages I'm looking for are, C, Pascal, assembly, Fortran, Forth, Lisp, APL, ADA.
- **Inside scoops** on the latest, greatest rumors from the industry. It sometimes takes a little yellow journalism to keep the industry on its toes. If you would like to use a pen name like ZOSO does, let me know and presto, the Micro Cornucopia shadow can strike fear into the hearts of those wearing their three-piece-vested-interests.
- **And anything else** (which covers a lot of things). ■ ■ ■

Power to the Big Board

By David Thompson



Schematic of +12V Regulator

Picking a power supply these days can be a problem. Everyone and his kid brother are building them in variations that read like the marquee at an ice cream parlor. So the following may be a little help, both in the selection of a supply and in understanding the consequences of a poor choice.

A group of us in Portland are using the Power One model CP 384. This is a simple linear supply with three outputs, +5V at 9 amp, -12V at about an amp, and +24V at .7 amp average or 5 amp peak. The price for this unit is about $120 in single unit quantities. It includes over-voltage and over-current protection.

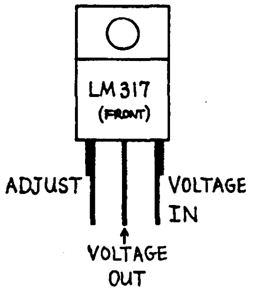
This supply is made to power 8-inch disk drives but if you add a simple 3-terminal regulator for +12V, it will also supply all the power for the Big Board.

To add +12V, tie the input of the regulator to the + lead of either of the two 60V electrolytics. The connecting post marked 24V return is ground (in fact, I just tied all the return posts together and ran them to the aluminum frame on the supply). The + lead on the electrolytics is at about 38V above ground which is higher than a standard 12V regulator (7812) is rated for. One member of our group is using a 7812 anyway and it is working fine. The LM317, however, is supposed to handle 38 volts just fine and it has a variable output to boot. Its output is designed to be 1.2V above the adj. lead, so by having approximately 1/10 of the drop between the output and the reference and 9/10 between the reference and ground you should get 12V. It comes out pretty close.

Mount the regulator against the frame with a mica insulator. Be sure to use silicon grease because it has to dissipate up to 13 watts.

Double check yourself.

It's a good idea to put a resistor load on the supply and then use a digital voltmeter to double check the outputs before connecting it up to your system. I have heard some pretty gruesome stories about folks accidentally putting outrageous voltages on their systems. Sometimes the systems have gone down permanently, other times they have gone temporarily insane, while a few have miraculously survived. It's best, obviously, to check the supply thoroughly.



LM 317 Regulator

Also, check to see that the supply will deliver 24V at 5 amps. The Power One's current limit is set at 1 amp at the factory. It will work in the circuit that way until you try to write something on the disk. The drive can then get very strange, generating random CRC errors and in some cases rendering a disk unusable.

If you a having drive problems, check the 24V line during a write operation. It shouldn't drop below 22V. (If the 24V line drops below 15V, you will probably get a buzz as the relay tries to load the head.)

To adjust the 24V current limit on the Power One Supply, locate the small screwdriver pot marked "24V I.LIMIT" and turn the control fully clockwise. It should now give you 5 amps at a rock solid 24V.

If you have had experience with other power supplies, let me know and I'll pass the word along here in Micro C. ■ ■ ■

Notes on Book Reviews

A good book or manual is a conversation with the author. At first it is a story, the reader sharing experiences with the author through the transparency of the written word. Later when the reader has questions about the material covered, the conversation turns to question and answer and the book becomes a reference volume.

Conversation: The tone of the conversation is very important. No one would freely choose to sit through hour upon hour of impersonal lecture if there were any easier way to get the same information. And yet some authors get mired in pages of third person passive.

Transparency: When the words move you smoothly and easily from idea to idea, then what you see are the ideas, not the words. The words have become transparent. If the sentences are too long and confusing or are short. Choppy. Broken up. Or if the ideas don't fit well together, then the conversation is reduced to one word at a time.

Asking questions: Technical books are generally used for two primary purposes. First, they are learning tools (the original conversation) and second, they are references as questions arise. Many technical books are arranged as training manuals only or as reference manuals only (sometimes for very good reason).

For instance, Microsoft's Basic 80 manual is primarily an alphabetical list of commands, which is fine if you know what commands you need to use and just need syntax examples. Kernighan and Ritchie's C book, on the other hand, is a well written introduction to the language, but if you want to look up a command you will have to start at the index and then refer to three or more places scattered through the book. At least they did an index.

And finally all the things you normally notice when reading a book:

- Content. Is the information appropriate to this group. Is the book a bargain in terms of information content.

(continued next page)

Three Books on CP/M

David Thompson Reviews

Using CP/M,
A Self-Teaching Guide
by Fernandez and Ashley
John Wiley & Sons
ISBN 0 471 08011-X

"Using CP/M" is the book that introduced me to CP/M. I purchased this text immediately after ordering the Big Board and by the time I had my system running I was pretty comfortable with the simpler portions of its operating system. But then I had already read the book cover to cover at least three times in anticipation.

The authors use an informal, conversational, writing style that's clear and easy to read. The text comes in short chunks. Each half-page or so, is followed by approximately a half-page of questions about the material just covered. I just skipped the questions, which meant that I skipped about half the total book. If you're really into questions you can use mine.

The book starts at a beginning level and stays there. It goes over and over the basics; spending 9 pages, for instance, on how to enter generalized filenames (*.*). And then it covers DDT in 10 lines.

Graphically speaking, "Using CP/M" doesn't make it. The writers organized the material pretty well but that organization disappears into a forest of sameness. Even the question sections are not visually separated well from the text, so it is sometimes hard for your eye to skip to the next piece of text. And skimming through the text to find a particular command is nearly impossible.

The only prayer this book has as a reference is the index. But if something didn't make the index you're in real trouble. Try to find the CP/M line editing commands (not ED). I gave up trying.

All in all, this text is reasonable for someone who is just starting out and and wants to do a lot of light reading.

■ ■ ■

The CP/M Handbook with MP/M
by Rodnay Zaks
Sybex
ISBN 0 89588 048 2

I got "The CP/M Handbook" after trying to use "Using CP/M" for a reference, so most of my experience with this text is for reference work. It's a real improvement. This book is full of tables, charts, reference guides and appendices. The chapters are organized in logical manner. The design and many illustrations (and index) help the reader locate specific information.

All of Zak's books that I've seen have been easy to read. The book starts at a beginning level and then progresses to to such things as reconfiguring CP/M for different system sizes. Advanced topics such as DDT and ASM, however, are covered just enough for the reader to access the programs. DDT gets about 2½ pages and ASM gets about 3. The reader is then referred to the user's guide from Digital Research.

This is a good text for someone using CP/M for running applications programs. PIP is pretty thoroughly covered in its own chapter and ED gets the detailed look it needs to keep the reader from losing his cursor entirely. So, for those not digging heavily into CP/M itself, this book is a definite option.

■ ■ ■

Osborne CP/M User Guide
By Thom Hogan
Osborne McGraw-Hill
ISBN 0 931988-44-6

The "Osborne CP/M User Guide" is the latest book to jump on the CP/M bandwagon and is the most technical of the three books. The introduction for beginners is relatively brief; and PIP, for instance, is presented in 21 pages of formatted text rather than a chapter in standard paragraph form.

This book contains a complete chapter on assembly language utilities, a subject skimmed over by the other texts. In fact, DDT and ASM each get 12 pages of remarkably thorough coverage. Like the Sybex book, Hogan makes extensive use of appendices for command summaries, etc. but he also adds some extra goodies like an annotated bibliography and addresses of companies supplying CP/M based products. (Hooray!)

Hogan's writing style is variable. Generally it is friendly but there are places where it is more formal than Zaks or Fernandez/Ashley. And he uses very few illustrations. However, the graphic layout of the material is very well done. In fact, you probably won't notice the dearth of illustrations because of the excellent use of type and layout to make the organization obvious. The combination of graphic design and index make this a first class reference work for CP/M.

This book is definitely the best book I've seen for someone using CP/M on a day-to-day basis. A beginner, however, might seriously consider starting with Zaks' book and then moving up to this one as he gains experience.

■ ■ ■

Notes on Book Reviews continued

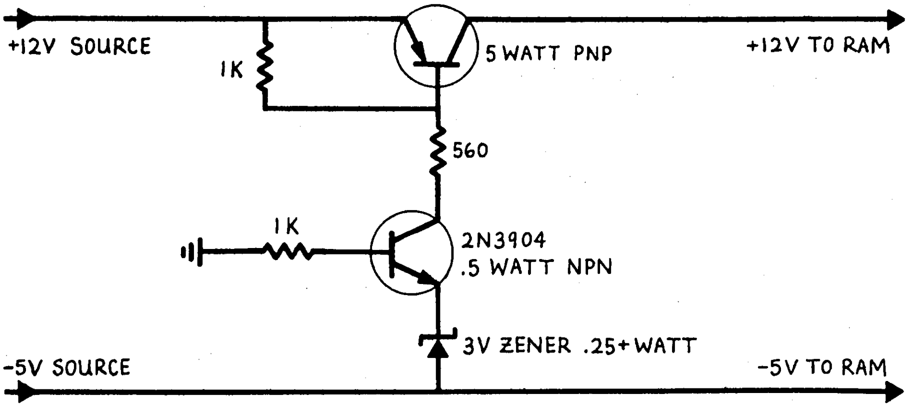
- Organization. Is the way the author progresses into the subject obvious? Is it easy to go back and find the information you need?
- Graphic design. Is the book visually appealing? Can you skim through glancing at the headlines and the illustrations and follow the book's progression through the subject?
- Illustrations. Are the illustrations well thought out and technically accurate or just afterthoughts to pretty up the page?
- Author's command of the subject. It's fun to catch a mistake in print. It's sort of like Moses messed up when chipping the rock, but too many errors cast doubt on the validity of the whole book.

So if you have books that are interesting to you and might be interesting to others in the group then by all means put the information down on a disk or paper or post card or whatever and let us know.

■ ■ ■

RAM Protection Circuit

By David Thompson



Schematic of RAM Protection Circuit

The RAM chips used on the Big Board (4116s) require three voltages for operation, +5V, +12V and -5V.

The +5V and +12V are used for device operation while the -5V provides an internal protective bias to keep the +12V from breaking down the chip. Isolation between some regions is provided by reverse biased diode junctions and the -5V provides the reverse bias.

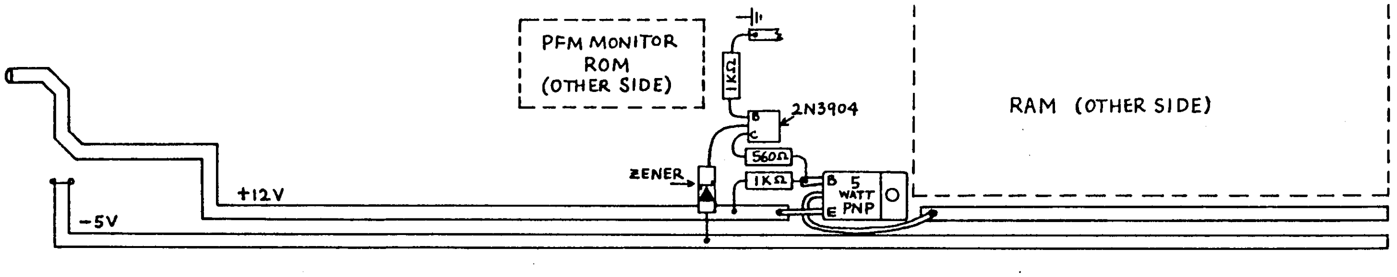
So, the device manufacturers strongly recommend that the -5V be available before the +12V. And they recommend that the -5V be available after the 12V goes away.

Most personal computers (TRS-80 etc.) have gotten around the problem by providing a slightly longer time constant for the +12V on power-up and a shorter time constant on power-down. But if the -5V supply ever shuts down momentarily or doesn't come up for some reason then the owner gets to buy new RAM. The Big Board, on the other hand is at the mercy of the supply.

The documentation recommends that you use a quality supply but there are many other reasons why -5V might not be available.

The following circuit takes care of the problem and has already saved our group a couple of sets of 4116s. The parts are mounted on the underside of the board and only one run (the +12V) has to be cut. Nothing is critical. The NPN is just a small, plastic, half-watt transistor with a DC gain of about 100. The PNP is a larger tab-style package and has a DC gain of 10 or more. Since the PNP is either saturated or off, it doesn't dissipate enough to require heat-sinking.

It is easy enough to check the whole thing out on the bench before installing it on the Big Board. When the -5V line drops down to about -3.5V the NPN should stop pulling current out of the base circuit of the PNP. As the PNP base rises, the PNP shuts off, removing the +12V from the RAM. ■ ■ ■



Example Installation of RAM Protection Circuit

Video Wiggle

The Cause and Cure

Quite a number of folks have noted on their subscription forms that they are bothered by wiggle on their video displays.

Well, the wiggle is caused by a frequency difference between your power line and the vertical output in the video generator. The video generator is 1 Hertz off (It's 61 Hz) and when it beats against power supply ripple in a Leedex monitor (for instance) you get wiggle. Many monitors also have trouble maintaining vertical sync because the frequency is outside their normal operating range.

To completely cure the problem, change the frequency of the CRT display generator crystal. Jim Tanner now has new crystals available free for Big Board owners. See "Notes From Garland, Texas" for more information.

A partial cure requires adding additional power supply filtering to the monitor. One additional 6000 ufd capacitor on the 12V DC line makes quite an improvement.

On the other hand, if your monitor accepts separate vertical, horizontal, and sync signals then you probably won't have any trouble. I've tried it both ways and my ancient Tektronix monitor with its separate inputs is as solid as a rock (it's also about that heavy).

■ ■ ■

Disk Formatter

By John P. Jones
5826 Southwest Ave.
St. Louis, MO 63139

Like most of the routines I use, this is nothing fancy but it gets the job done.

Since memory is not a problem on the Big Board, an entire track's data image is set up in memory. The WD-1771's write track command can then be used for formatting the disk. The listing is reasonably well commented so the only additional point I'll make is that the same basic method used in PFM-80 to eliminate the need for DMA is used in this routine.

The routine was tested with a deliberately "trashed" disk (totally wiped out with a magnet). In fact, the disk I sent with this article was re-formatted after deliberately being destroyed. The routine does no prompting or error reporting. To use it, place the disk to be formatted in drive B and enter FORMAT.

Editor's note: This program really works! If you don't have something like M80 to assemble this with then hang on. The COM version will be on the group disk plus I'm trying to make the Crowe Z80 assembler available.

■■■

Disk Formatter Listing

```
        TITLE   'SSSD DISKETTE FORMATTER'
;
;       THIS PROGRAM IS DESIGNED TO FORMAT A SINGLE DENSITY
;       SINGLE SIDED SOFT SECTORED 8" DISKETTE INTO STANDARD
;       128 BYTE SECTORS. IT IS DESIGNED TO RUN ON
;       THE FERGUSON BIG BOARD Z-80 COMPUTER. IT TAKES
;       ADVANTAGE OF THE WD-1771 FLOPPY DISK CONTROLLER'S
;       CAPABILITIES FOR SEMI-AUTOMATIC FORMATTING.
;
;
;
;       WRITTEN: J.P. JONES  4/20/81
;
;       MODIFIED: J.P.J. 5/14/81
;
;
        ORG     100H            ;STD CP/M COM PROGRAM
;
BOOT    EQU     0               ;CP/M BOOT
MONITR  EQU     0F000H          ;WILL USE SOME OF PFM-80
WDSTAT  EQU     10H             ;1771 STATUS ADDR
WDCTL   EQU     WDSTAT          ;CONTROL = STATUS WRITE
WDATA   EQU     13H             ;1771 DATA I/O
HOME    EQU     MONITR+1EH      ;DISK HOME ROUTINE
SEEK    EQU     MONITR+21H      ;SEEK TRACK ROUTINE
SELECT  EQU     MONITR+1BH      ;SELECT DRIVE ROUTINE
;
;       FIRST, SET UP ONE TRACK'S DATA IMAGE
;
        LD HL, DATA             ;POINT TO DATA AREA OF DISK IMAGE
        LD DE, DATA+1
        LD (HL), 0E5H           ;E5 = BLANK VALUE
        LD BC, 127
        LDIR                    ;FILL DATA AREA
;
;       COPY ONE SECTOR 25 TIMES
;
        LD HL, SECT1            ;START OF SECTOR 1 DATA
        LD DE, SECT2            ;ADD ONTO END
        LD BC, 186*25           ;186 BYTES PER SECTOR
        LDIR                    ;FILL THAT MEMORY
;
;       NOW SET UP SECTOR NUMBER IN PROPER POSITIONS
;
        LD HL, SECTNO           ;POINT TO SLOT IN FIRST IMAGE
        LD DE, 186              ;OFFSET TO NEXT SECTOR
        LD BC, 26*256+1         ;B=COUNTER, C=SECTOR #
;
SECTID  LD (HL),C               ;STORE SECTOR #
        INC C                   ;INCREMENT SECTOR #
        ADD HL,DE               ;POINT TO NEXT SECTOR DATA
        DJNZ    SECTID          ;DO ALL 26
;
;       NOW PUT TRAILER OF FF'S AFTER WHOLE TRACK
;
        LD HL, SECT1+4836       ;POINT AFTER DATA
        LD A, -1                ;NEED 247 BYTES OF FF SO
        LD B,0                  ;DO 256 BYTES FOR INSURANCE
ENDMRK  LD (HL), A
        INC HL
        DJNZ    ENDMRK
;
```

(continued next column)

```
;       SET UP TO FORMAT DISK IN DRIVE 'B'
;
        LD C, 1                 ;B IS DRIVE #1
        CALL    SELECT
;
;       NOW DO SETUP FOR FORMATTING
;
        LD A,(66H)              ;GET BYTE AT NMI VECTOR
        PUSH    AF
        LD A, 0C9H              ;RET INSTRUCTION
        LD (66H), A             ;STORE RETURN
;
;       DO THE FORMAT
;
        DI                      ;CANNOT INTERRUPT
        LD C,0                  ;START WITH TRACK 0
NXTTRK  LD B, 26                ;26 SECTORS PER TRACK
        PUSH    BC              ;NEED THEM LATER
;
;       SEEK NEXT TRACK IN SEQUENCE
;
        CALL    SEEK
;
        POP     BC              ;TRACK # AND SECTOR CTR BACK
        PUSH    BC              ;WILL NEED AGAIN
;
;       PUT TRACK ID'S IN PROPER PLACE IN TRACK IMAGE
;
        LD HL, TRKNO            ;POINT TO POSITION IN IMAGE
        LD DE, 186              ;OFFSET FOR EACH SECTOR
TRAKID  LD (HL), C              ;STORE CURRENT TRACK NO.
        ADD HL,DE               ;POINT TO NEXT SECTOR
        DJNZ    TRAKID
;
;       DO THE TRACK WRITE
;
        LD HL, LEADER           ;POINT TO DATA
        LD D, 20                ;20 * 256 + 36 = TOTAL BYTES
        LD B, 36
        LD C, WDATA             ;C POINTS TO 1771 DATA PORT
        LD A, 0F4H              ;WRITE TRACK COMMAND
        OUT (WDCTL), A          ;SEND COMMAND
;
```

(continued next page)

Disk Formatter Listing (continued)

```
NXTBYT  HALT                    ;WHEN 1771 READY, WILL NMI
        OUTI                    ;SEND BYTE
        JP NZ   NXTBYT
        DEC D                   ;OUTER BYTE COUNTER
        JP NZ   NXTBYT
;
;       DO SETUP FOR NEXT TRACK
;
        POP     BC              ;GET TRACK COUNTER BACK
        INC C                   ;UPDATE IT
        LD A,C
        CP 77                   ;IF 77, DONE
        JP NZ   NXTTRK
        POP AF                  ;GET BYTE BACK FOR NMIVEC
        LD (66H), A
        EI                      ;REENABLE NORMAL OPERATION
        JP      BOOT            ;BACK TO CPM
;
;       TRACK DATA FOLLOWS
;
LEADER  DEFW    -1              ;40 BYTES OF FF
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
;
        DEFW    0               ;6 BYTES OF 0
        DEFW    0
        DEFW    0
;
        DEFB    0FCH            ;WRITE INDEX MARK COMMAND
;
        DEFW    -1              ;26 BYTES OF FF
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
```

(continued next column)

```
;
SECT1   DEFW    0               ;SECTOR DATA STARTS
        DEFW    0               ;6 BYTES OF 0
        DEFW    0
        DEFB    0FEH            ;WRITE ID ADDRESS MARK CMD
TRKNO   DEFW    0               ;FIRST BYTE OF WORD = TRACK #
SECTNO  DEFW    0               ;FIRST BYTE OF WORD = SECTOR #
        DEFB    0F7H            ;WRITE CRC COMMAND
;
        DEFB    -1              ;11 BYTES OF FF
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
;
        DEFW    0               ;6 BYTES OF 0
        DEFW    0
        DEFW    0
        DEFB    0FBH            ;WRITE DATA ADDRESS CMD
;
DATA    DEFS    128             ;ACTUAL DATA AREA
;
        DEFB    0F7H            ;WRITE CRC COMMAND
;
        DEFB    -1              ;27 BYTES OF FF
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1
        DEFW    -1              ;END OF ONE SECTOR
;
;       ON EXECUTION THE PRECEDING SECTOR DATA WILL BE
;       DUPLICATED 25 TIMES TO GIVE A FULL 26 SECTORS OF
;       DATA.  THE 256 BYTES FOLLOWING THE FULL TRACK'S
;       DATA WILL BE FILLED WITH FF'S.  THIS FILLS THE AREA
;       BETWEEN THE LAST SECTOR AND THE NEXT INDEX. THE WD-
;       1771 EXITS THE 'WRITE TRACK' MODE ON RECEIPT OF THE
;       INDEX SIGNAL.
;
SECT2   END
```

end

New Character ROM

Sometime after the first of this year, Jim Tanner began shipping the Big Board with a new character ROM. The ROM has true lower case characters rather than the smaller upper case/larger upper case ROM shipped in the early boards.

- The ROM uses a 5 by 8 dot matrix so it has one-dot descenders.
- It contains the standard character set for 00(hex) through 7F(hex). (Even though the Big Board only displays 20—7F.)
- And I like it because I designed it and gave it to Jim.
- However, It isn't perfect.

So, for a week or so I worked on the g, y, t, f, and q characters until . . . well, if it isn't perfect now, I give up because I'm absolutely tickled.

If your board has true upper/lower case but you would like to have the absolute latest greatest, then send me a ROM and $5.00.

If you have one of the old upper case/smaller upper case ROMs you have a choice. Send a ROM to Jim Tanner at Digital Research Computers of Texas and he will burn a copy of my first character ROM (the one he's using in the new boards) for you, free. Or you can send me the ROM and $5.00 and get the deluxe version.

Price

- $5.00 if you send a 350ns 2716 and a self-addressed, stamped package I can ship it back in.
- Or instead of $5.00 you can submit something to the magazine, a program, a book or software review, a schematic and comments, a page or two about what you are doing with the Big Board, etc., along with your ROM and SASE and presto, you get fame AND a new character set, free! (And those who contributed to this issue also qualify for a free burn.)

Make checks payable to Micro Cornucopia. If you don't agree that it's a $5.00 improvement, I'll send you the $5.00 back.

PFM-80 Monitor

By Don Retzlaff

6435 Northwood
Dallas, Texas 75225

The PFM-80 Monitor is the primary control program for your Big Board computer. It was burned into the EPROM that is installed in the first ROM socket (U67).

PFM and CBIOS were written by Russell Smith, who is an exceptional young programmer who operates his own software house in Denton, Texas. He has helped me immeasurably in understanding PFM and implementing my programs on the Big Board. As time goes on I will pass along some of this expertise to you, through this column.

If your curiosity is like mine you want to know what PFM stands for. I was informed that PFM is the abbreviation of the profound literal description of what the monitor is: "PRETTY F——KIN' MAGIC."

When the computer is turned on or the reset button is pressed, the Big Board automatically starts executing the COLD START BOOT program in the monitor ROM. The first five instructions in the ROM (starting at location 0000H) copy the PFM monitor program from the ROM into upper memory starting at location F000H and continuing through F7E6H. The RAM locations starting at location FF00H through FFA8H are used as monitor data storage locations.

After PFM has been booted into RAM the monitor starts executing and goes through the cold start initalization routine that does the following:

1. Initalizes data storage pointers.
2. Clears the scratch RAM with zeros.
3. Fills CRT storage with blanks.
4. Initalizes values in memory.
5. Initalizes programmable I/O devices.
6. Waits for input from keyboard or terminal.
7. Sets baud rate for SIO input if input from there.
8. Displays sign-on message on the apporpriate device.
9. Displays monitor prompt *
10. Waits for input.

At this point PFM is up and operating.

I think that it is important to note that whenever an RS-232 serial terminal is connected to SIO PORT B, PFM automatically determines the BAUD rate of the terminal by analyzing the input from the single carriage return. It then sets up the baud rate generator to the correct frequency.

In future articles we will get deeper into the monitor.

Now let's discuss the monitor entry point table. Starting at location F000H you will find a series of jump instructions. These provide a fixed address that can be used as entry points to the various monitor routines. These will be useful in software routines that you write. This table will provide a constant jump location for these routines even if updates are made to the monitor. Thus, changes in addresses of the internal routines will not affect your software.

I plan to cover the various features of PFM and CBIOS which work together to control your Big Board. In succeeding articles I will lead you through the assembly language listings of both PFM and CBIOS, pointing out the features of each and how you can make the most from each.

In the next issue we will discuss the mechanics of modifying the monitor.

■ ■ ■

Editor's Note: The first installment of the PFM monitor listing begins on the following page. We will continue the listing in the September issue.

PFM Monitor Listing

```
                  ;*******************************************************
                  ;*                                                     *
                  ;*      BIGBOARD MONITOR ROM, NON-RELOCATABLE VERSION  *
                  ;*      Russell Smith         2-August-1980            *
                  ;*                                                     *
                  ;*******************************************************
                  ;
                  ;
                          PSECT   ABS
>F000             ROM     EQU     0F000H     ;START OF 2K ROM
>FF00             RAM     EQU     0FF00H     ;START OF 256 BYTE RAM
>3000             CRTMEM  EQU     3000H      ;BASE OF 4K CRT MEMORY
                  ;
                  ;
>F000                     ORG     ROM
                          INCLUDE INIT.ASM
                  ;*******************************************************
                  ;*                                                     *
                  ;*      COLD START INITIALIZATION ROUTINE FOR          *
                  ;*      CONFIGURING THE SYSTEM AFTER A POWER-ON        *
                  ;*      OR PUSHBUTTON RESET.                           *
                  ;*                                  18-Oct-80          *
                  ;*                                                     *
                  ;*******************************************************
                  ;
                  ;
                  ;-- MONITOR ENTRY POINT TABLE --
                  ;
F000    C32AF0    COLD:   JP      INIT       ;MONITOR COLD ENTRY POINT
F003    C32BF1    WARM:   JP      PROMPT     ;MONITOR WARM ENTRY POINT
F006    C331F4    CONST:  JP      KBDST      ;CONSOLE STATUS VECTOR
F009    C339F4    CONIN:  JP      KBDIN      ;CONSOLE INPUT VECTOR
F00C    C320F5    CONOUT: JP      CRTOUT     ;CONSOLE OUTPUT VECTOR
F00F    C320F5            JP      CRTOUT     ;CRT OUTPUT VECTOR
F012    C3E8F4            JP      SIOST      ;SIO CHANEL B STATUS VECTOR
F015    C3F0F4            JP      SIOIN      ;SIO CHANEL B INPUT VECTOR
F018    C3FEF4            JP      SIOOUT     ;SIO CHANEL B OUTPUT VECTOR
F01B    C3B1F6            JP      SELECT     ;DISK DRIVE SELECT
F01E    C3E9F6            JP      HOME       ;HOME R/W HEAD
F021    C3FBF6            JP      SEEK       ;SEEK TO TRACK
F024    C32AF7            JP      READ       ;READ SECTOR
F027    C31FF7            JP      WRITE      ;WRITE SECTOR
                  ;
                  ;
                  ;
                  ;DO A SHORT POST-RESET TIME DELAY. ALSO INITIALIZES THE
                  ;STACK POINTER AND FILLS THE MONITOR SCRATCH RAM WITH ZEROS
                  ;
F02A    F3        INIT:   DI
F02B    2100FF            LD      HL,RAM     ;POINT TO START OF MONITOR RAM
F02E    3600      INIT1:  LD      (HL),0     ;FILL 256 BYTE SPACE WITH ZEROS
F030    F9                LD      SP,HL      ;SOMETHING USEFUL TO ADD DELAY
F031    2C                INC     L
F032    20FA              JR      NZ,INIT1-$ ;LOOP TAKES 4 MILLISECONDS
                  ;
                  ;INITIALIZE THE Z-80 FOR INTERRUPT MODE #2

F034    7C                LD      A,H
```

```
F08A    21CAF0            LD      HL,RATES-1
F08D    23        BAUD3:  INC     HL         ;INDEX INTO BAUD RATE TABLE
F08E    17                RLA                ; USING COUNT DERRIVED IN A
F08F    30FC              JR      NC,BAUD3-$
F091    7E                LD      A,(HL)     ;GET BAUD RATE CONTROL BYTE FROM
F092    D30C              OUT     (BAUDB),A  ;TBL & OUTPUT TO COM-8116 TIMER
F094    CDF0F4            CALL    SIOIN      ;DISCARD 1ST SERIAL INPUT CHAR
F097    3E01              LD      A,1
F099    D307              OUT     (SIOCPB),A ;RE-PROGRAM SIO B TO GENERATE
F09B    3E1C              LD      A,00011100B;INTERRUPTS ON RECEIVED DATA,
F09D    D307              OUT     (SIOCPB),A ;PARITY DOES NOT AFFECT VECTOR
F09F    21FEF4            LD      HL,SIOOUT
F0A2    220DF0            LD      (CONOUT+1),HL ;RE-DIRECT CONSOLE OUTPUT
;                                                TO SIO
                  ;
                  ;PRINT SIGNON MESSAGE
                  ;
F0A5    FB        SIGNON: EI
F0A6    CDECF3            CALL    PNEXT
F0A9    0D0A              DEFB    CR,LF
F0AB    2E2E2E20          DEFM    '... system monitor 3.3 ...'
        73797374
        656D206D
        6F6E6974
        6F722033
        2E33202E
        2E2E
F0C5    0D0A              DEFB    CR,LF
F0C7    04                DEFB    EOT
F0C8    C303F0            JP      WARM       ;GO ENTER MONITOR
                  ;
                  ;
                  ;
                  ;BAUD RATE CONSTANTS FOR COM 8116 BAUD RATE GENERATOR
                  ;
F0CB    05        RATES:  DEFB    0101B      ;    300 BAUD
F0CC    06                DEFB    0110B      ;    600 BAUD
F0CD    07                DEFB    0111B      ;   1200 BAUD
F0CE    0A                DEFB    1010B      ;   2400 BAUD
F0CF    0C                DEFB    1100B      ;   4800 BAUD
F0D0    0E                DEFB    1110B      ;   9600 BAUD
F0D1    0F                DEFB    1111B      ;  19200 BAUD
F0D2    0F                DEFB    1111B      ;  19200 BAUD
                  ;
                  ;
>F0D3             INTAB   EQU     $          ;INITIALIZATION DATA TABLES
                  ;
                  ;
                  ;INITIALIZE THE Z-80 'I' REGISTER INTERRUPT VECTOR TABLE
                  ;
F0D3    02                DEFB    2
F0D4    1AFF              DEFW    SYSVEC+2
F0D6    8CF4              DEFW    KEYSRV     ;PARALLEL KBD INTERRUPT VECTOR

F0D8    02                DEFB    2
F0D9    16FF              DEFW    CTCVEC+6
F0DB    9FF4              DEFW    TIMER      ;1 SEC TIMER INTERRUPT VECTOR

F0DD    04                DEFB    4
F0DE    04FF              DEFW    SIOVEC+4
F0E0    AFF4              DEFW    SIOINT     ;SIO RECEIVE INTERRUPT VECTOR
F0E2    CFF4              DEFW    SIOERR     ;SIO PARITY, OVERRUN & FRAMING
;                                             ERROR
                  ;
                  ;INITIALIZE DISK I/O DRIVER VARIABLES
                  ;
F0E4    08                DEFB    8
```

```
F035  ED47      0059           LD      I,A          ;LOAD I REG WITH MSB OF VECTOR
;                                                    TABLE
F037  ED5E      0060           IM      2            ;AND SELECT INTERRUPT MODE 2
                0061 ;
F039  CDECF5    0062           CALL    CLRSCN       ;FILL THE CRT MEMORY WTH BLANKS
                0063 ;
                0064 ;STORE ANY NON-ZERO VALUES FOR VARIABLES IN MEMORY
                0065 ;
F03C  21D3FO    0066           LD      HL,INTAB     ;POINT TO DEFAULT VAR TABLE
F03F  0600      0067 INIT2:    LD      B,O
F041  4E        0068           LD      C,(HL)       ;BC=DATA BLOCK BYTECOUNT
F042  23        0069           INC     HL
F043  5E        0070           LD      E,(HL)       ;DE=DESTINATION FOR DATA
F044  23        0071           INC     HL
F045  56        0072           LD      D,(HL)
F046  23        0073           INC     HL
F047  EDBO      0074           LDIR                 ;COPY DATA @ HL TO VAR @ DE
F049  CB7E      0075           BIT     7,(HL)
F04B  28F2      0076           JR      Z,INIT2-$    ;LOOP AGAIN IF NOT END OF TABLE
                0077 ;
                0078 ;INITIALIZE THE PROGRAMMABLE I/O DEVICES
                0079 ;
F04D  23        0080           INC     HL           ;POINT TO I/O INIT DATA TABLE
F04E  46        0081 INIT3:    LD      B,(HL)       ;B=INIT LOOP BYTECOUNT
F04F  23        0082           INC     HL
F050  4E        0083           LD      C,(HL)       ;C=DEVICE CONTROL PORT#
F051  23        0084           INC     HL
F052  EDB3      0085           OTIR                 ;SEND DATA @ HL TO PORT @ C
F054  CB7E      0086           BIT     7,(HL)       ;TEST FOR TABLE END MARKER
F056  28F6      0087           JR      Z,INIT3-$    ;LOOP AGAIN IF NOT AT END
                0088 ;
                0089 ;DETERMINE IF CONSOLE I/O CONFIGURATION WILL BE FOR THE
                0090 ;ON-BOARD CRT AND KEYBOARD OR AN EXTERNAL SERIAL TERMINAL.
                0091 ;
F058  ED78      0092           IN      A,(C)        ;TEST SIO READ REG 2 TO CHECK
F05A  FE06      0093           CP      00000110B    ;IF THE SIO IS INSTALLED
F05C  2012      0094           JR      NZ,PARALL-$;SKIP CONFIG TEST IF NO SIO
F05E  DB1E      0095           IN      A,(KBDDAT)   ;MAKE SURE KBD PIO 'READY'RESET
F060  0610      0096           LD      B,00010000B;B=RESET SIO EXT STATUS COMMAND
F062  ED41      0097 DECIDE:   OUT     (C),B        ;TEST FOR ARRIVAL OF A SERIAL
F064  ED78      0098           IN      A,(C)        ; INPUT CHAR START BIT
F066  CB67      0099           BIT     4,A
F068  200E      0100           JR      NZ,BAUD-$    ;EXIT LOOP IF START BIT DETECTED
F06A  DB1C      0101           IN      A,(BITDAT)
F06C  CB5F      0102           BIT     3,A          ;TEST FOR DATA RDY STROBE FROM
F06E  20F2      0103           JR      NZ,DECIDE-$;PARALLEL KBD, LOOP IF INACTIVE
F070  DB1E      0104 PARALL:   IN      A,(KBDDAT)   ;DISCARD FIRST KEYBOARD CHAR
F072  3E83      0105           LD      A,10000011B
F074  D31F      0106           OUT     (KBDCTL),A   ;ENABLE INTERRUPTS FROM KBD PIO
F076  182D      0107           JR      SIGNON-$     ;SIGNON WTH BUILT-IN CONSOLE I/O
                0108 ;
                0109 ;
                0110 ;AUTOMATIC BAUD RATE SETTING ROUTINE FOR SIO
                0111 ;
F078  AF        0112 BAUD:     XOR     A
F079  ED41      0113 BAUD1:    OUT     (C),B
F07B  ED50      0114           IN      D,(C)        ;READ SIO STATUS REGISTER
F07D  CB62      0115           BIT     4,D          ;TEST THE SYNC/HUNT BIT
F07F  28F8      0116           JR      Z,BAUD1-$    ;LOOP UNTIL IT CHANGES STATE
F081  3C        0117 BAUD2:    INC     A
F082  ED41      0118           OUT     (C),B        ;RESET REGISTER #0 FLAGS AGAIN
F084  ED50      0119           IN      D,(C)        ;&LOOP TIMING THE SYNC/HUNT BIT
F086  CB62      0120           BIT     4,D
F088  20F7      0121           JR      NZ,BAUD2-$ ;REPEAT UNTIL BIT CHANGES AGAIN
```

(continued on top of page 10)

```
FOE5  65FF      0181           DEFW    UNIT
FOE7  FF        0182           DEFB    255          ;FLAG ALL DRIVES AS DE-SELECTED
FOE8  FFFFFFFF  0183           DEFB    255,255,255,255 ;CLEAR HEAD POSITION TABLE
FOEC  00        0184           DEFB    00000000B    ;SELECT FASTEST SEEK SPEED
FOED  80        0185           DEFB    128          ;SELECT 128 BYTE SECTOR LENGTH
FOEE  1E        0186           DEFB    30           ;SET MOTOR TURN-OFF TIMER
                0187 ;
                0188 ;INITIALIZE THE CRT DISPLAY CURSOR
                0189 ;
FOEF  02        0190           DEFB    2
FOFO  75FF      0191           DEFW    CHRSAV
FOF2  20        0192           DEFB    ' '
FOF3  5F        0193           DEFB    '_'          ;USE NON-BLINKING UNDERSCORE
                0194 ;
                0195 ;SET DEFAULT 'SOFTWARE' INTERRUPT VECTORS
                0196 ;
FOF4  06        0197           DEFB    6
FOF5  57FF      0198           DEFW    TIKVEC
FOF7  80F4      0199           DEFW    DSKTMR       ;POINT'TIKVEC'TO DISK MTR TIMER
FOF9  44F4      0200           DEFW    STASH        ;POINT'PINVEC'TO FIFO INPUT
;                                                    ROUTINE
FOFB  44F4      0201           DEFW    STASH        ;POINT'SINVEC'TO FIFO INPUT
;                                                    ROUTINE
                0202 ;
                0203 ;SET FREE MEMORY POINTER
                0204 ;
FOFD  02        0205           DEFB    2
FOFE  7AFF      0206           DEFW    FREPTR
F100  E6F7      0207           DEFW    ROMEND       ;POINT TO 1ST LOC AFTER MONITOR
                0208 ;
                0209 ;
F102  FF        0210           DEFB    -1           ;END OF VARIABLE INIT TABLE
                0211 ;
                0212 ;
                0213 ;
>0000           0214 BAUDA     EQU     00H          ;CHANEL A BAUD RATE GENETATOR
>0004           0215 SIO       EQU     04H          ;DUAL SERIAL I/O
>0008           0216 GENPIO    EQU     08H          ;GENERAL PURPOSE PARALLEL I/O
>000C           0217 BAUDB     EQU     0CH          ;CHANEL B BAUD RATE GENERATOR
>0010           0218 WD1771    EQU     10H          ;WEST DIGITAL DISK CONTROLLER
>0014           0219 SCROLL    EQU     14H          ;CRT SCROLL MEM SCROLL REGISTER
>0018           0220 CTC       EQU     18H          ;QUAD COUNTER/TIMER CIRCUIT
>001C           0221 SYSPIO    EQU     1CH          ;SYSTEM PARALLEL I/O
                0222 ;
                0223 ;INITIALIZE SYSTEM PIO FOR USE AS BANK-SWITCH,
                0224 ;DISK DRIVE SELECT AND PARALLEL KEYBOARD INPUT
                0225 ;
>001C           0226 BITDAT    EQU     SYSPIO+0
>001D           0227 BITCTL    EQU     SYSPIO+1
>001E           0228 KBDDAT    EQU     SYSPIO+2
>001F           0229 KBDCTL    EQU     SYSPIO+3
                0230
F103  031D      0231           DEFB    3,BITCTL
F105  CF        0232           DEFB    11001111B    ;PUT SYSTEM PIO IN BIT MODE
F106  18        0233           DEFB    00011000B    ;MAKE BITS 4 AND 3 BE INPUTS
F107  40        0234           DEFB    01000000B    ;DISABLE INTERRUPTS
                0235 ;
F108  011C      0236           DEFB    1,BITDAT
F10A  00        0237           DEFB    00000000B    ;DE-SELECT ROMS, ENABLE DRIVE 0
                0238 ;
F10B  021F      0239           DEFB    2,KBDCTL
F10D  4F        0240           DEFB    01001111B    ;PUT KBD PORT IN INPUT MODE
F10E  1A        0241           DEFB    SYSVEC+2     ;LOAD KEYBOARD INTERRUPT VECTOR
                0242 ;
                0243 ;
```

(continued next page)

PFM Monitor Listing (continued)

```
                  ;INITIALIZE CHANELS 2 AND 3 OF THE CTC
                  ;TO GENERATE ONE SECOND INTERRUPTS FROM CTC3
                  ;
>0018             CTC0      EQU     CTC+0         ;CTC CHANEL 0 PORT#
>0019             CTC1      EQU     CTC+1         ;CTC CHANEL 1
>001A             CTC2      EQU     CTC+2         ;CTC CHANEL 2
>001B             CTC3      EQU     CTC+3         ;CTC CHANEL 3

F10F  0118                  DEFB    1,CTC0
F111  10                    DEFB    CTCVEC        ;BASE INTERRUPT VECTOR FOR CTC
                  ;
F112  021A                  DEFB    2,CTC2
F114  27                    DEFB    00100111B     ;PUT CTC2 IN TIMER MODE
F115  69                    DEFB    105           ;CTC2 PERIOD=105*256*400 NS
                  ;
F116  021B                  DEFB    2,CTC3
F118  C7                    DEFB    11000111B     ;PUT CTC3 IN COUNTER MODE
F119  5D                    DEFB    93            ;CTC3 PERIOD=999936 uS
                  ;
                  ;
                  ;INITIALIZE SIO CHANEL B FOR ASYNCHRONOUS SERIAL
                  ;INTERFACE TO PRINTER OR TERMINAL
                  ;
>0004             SIODPA    EQU     SIO+0         ;SIO DATA PORT A
>0005             SIODPB    EQU     SIO+1         ;SIO DATA PORT B
>0006             SIOCPA    EQU     SIO+2         ;SIO CONTROL/STATUS PORT A
>0007             SIOCPB    EQU     SIO+3         ;SIO CONTROL/STATUS PORT B

F11A  010C                  DEFB    1,BAUDB
F11C  05                    DEFB    0101B         ;SET COM 8116 TO 300 BD DEFAULT

F11D  0B07                  DEFB    11,SIOCPB
F11F  04                    DEFB    4             ;SELECT REGISTER #4
F120  45                    DEFB    01000101B     ;16X CLK,1 STOP BIT,ODD PARITY
F121  01                    DEFB    1             ;SELECT REGISTER #1
F122  04                    DEFB    00000100B     ;STATUS AFFECTS VECTOR,
;                                                  NO INTERRUPTS
F123  03                    DEFB    3             ;SELECT REGISTER #3
F124  41                    DEFB    01000001B     ;7 BITS/RX CHAR
F125  05                    DEFB    5             ;SELECT REGISTER #5
F126  AA                    DEFB    10101010B     ;7 BITS/TX CHAR, ASSERT DTR
F127  02                    DEFB    2             ;SELECT REGISTER #2
F128  00                    DEFB    SIOVEC        ;LOAD INTERRUPT VECTOR BASE
F129  02                    DEFB    2             ;SELECT READ REG#2 FOR SIO TEST

F12A  FF                    DEFB    -1            ;END-OF-TABLE
                  ;
                  ;INIT DONE
                  ;
                  ;
                            INCLUDE MONITOR.ASM
                  ;*****************************************************
                  ;*                                                   *
                  ;*         BASIC HEX MONITOR FOR Z-80 PROCESSORS     *
                  ;*                                     3-Aug-80      *
                  ;*                                                   *
                  ;*****************************************************
                  ;
                  ;
                  ;
                  ;
```

```
F19F  20F3                  DEFW    OUTCMD        ;WRITE TO OUTPUT PORT
F1A1  BDF1                  DEFW    DSKCMD        ;DISPLAY DISK SECTOR DATA

>0021             CMDSIZ    EQU     $-CMDTAB
                  ;
                  ;
                  ;*****************************************************
                  ;*                                                   *
                  ;*      MONITOR COMMAND ACTION ROUTINES PACKAGE      *
                  ;*                                                   *
                  ;*****************************************************
                  ;
                  ;
                  ;
                  ;
                  ;
                  ;-- DISK BOOT LOADER COMMAND --
                  ;
F1A3  0E00        BOOT:     LD      C,0           ;SELECT DRIVE 0 FOR BOOT LOAD
F1A5  CDB1F6                CALL    SELECT
F1A8  203D                  JR      NZ,DSKERR-$
F1AA  CDE9F6                CALL    HOME          ;HOME HEAD TO TRACK 0
F1AD  2038                  JR      NZ,DSKERR-$;ERROR IF NOT READY OR AT TR0
F1AF  218000                LD      HL,0080H      ;POINT TO CP/M READ BUFFER
F1B2  0E01                  LD      C,1           ;SELECT SECTOR 1
F1B4  CD2AF7                CALL    READ          ;READ TRACK 0/ SECTOR 1
F1B7  202E                  JR      NZ,DSKERR-$
F1B9  F1                    POP     AF            ;CLEAN UP STACK
F1BA  C38000                JP      0080H         ;GO EXECUTE LOADER
                  ;
                  ;
                  ;-- DISK SECTOR READ COMMAND --
                  ;
F1BD  FE03        DSKCMD:   CP      3             ;CHECK PARAMETER COUNT
F1BF  37                    SCF
F1C0  C0                    RET     NZ
F1C1  4D                    LD      C,L           ;USE FIRST ARG AS UNIT#
F1C2  CDB1F6                CALL    SELECT
F1C5  2020                  JR      NZ,DSKERR-$
F1C7  217EFF                LD      HL,PARAM2
F1CA  4E                    LD      C,(HL)        ;USE SECOND ARG AS TRACK#
F1CB  CDFBF6                CALL    SEEK
F1CE  2017                  JR      NZ,DSKERR-$
F1D0  2180FF                LD      HL,PARAM3
F1D3  4E                    LD      C,(HL)        ;USE THIRD ARG AS SECTOR#
F1D4  218000                LD      HL,0080H
F1D7  CD2AF7                CALL    READ
F1DA  CBC7                  SET     0,A           ;MARK ERROR BYTE AS DUE TO READ
F1DC  2009                  JR      NZ,DSKERR-$
F1DE  218000                LD      HL,0080H
F1E1  110800                LD      DE,8
F1E4  C327F2                JP      DUMP          ;DUMP DISK READ BUFFER & RETURN
                  ;
                  ;
F1E7  4F          DSKERR:   LD      C,A           ;SAVE 1771 STATUS
F1E8  CDECF3                CALL    PNEXT
F1EB  6469736B              DEFM    'disk error '
      20657272
      6F7220
F1F6  04                    DEFB    EOT
F1F7  0608                  LD      B,8           ;PRINT 1771 ERROR BYTE IN BIN
F1F9  AF          DSKR2:    XOR     A
F1FA  CB11                  RL      C
F1FC  CE30                  ADC     A,'0'         ;TRANSFORM A INTO ASCII'1'OR'0'
F1FE  CD15F4                CALL    OUTPUT
F201  10F6                  DJNZ    DSKR2-$       ;REPEAT FOR 8 BITS
F203  B7                    OR      A
F204  C9                    RET
```

```
F12B  CDECF3   0304 PROMPT: CALL    PNEXT
F12E  0D0A     0305         DEFB    CR,LF
F130  2A20     0306         DEFM    '* '
F132  04       0307         DEFB    EOT
F133  2188FF   0308         LD      HL,LINBUF
F136  0E20     0309         LD      C,32
F138  CD3BF3   0310         CALL    GETLIN      ;INPUT A BUFERED CONSOLE LINE
F13B  3835     0311         JR      C,WHAT-$    ;PRINT 'WHAT ?' IF INPUT ERROR
               0312
F13D  AF       0313         XOR     A
F13E  3284FF   0314         LD      (ESCFLG),A
F141  CDFCF3   0315         CALL    CRLFS
F144  3A88FF   0316         LD      A,(LINBUF) ;GET FIRST CHAR IN LINE
F147  FE0D     0317         CP      CR
F149  28E0     0318         JR      Z,PROMPT-$ ;JUMP IF A NULL LINE
F14B  2182F1   0319         LD      HL,CMDTAB  ;SEARCH FOR A MATCHING CHAR
F14E  010B00   0320         LD      BC,CMDSIZ/3      ; IN COMMAND SEARCH TABLE
F151  CD60F3   0321         CALL    SEARCH
F154  201C     0322         JR      NZ,WHAT-$  ;TRY AGAIN IF SEACRH FAILS
F156  C5       0323         PUSH    BC
F157  FD2189FF 0324         LD      IY,LINBUF+1
F15B  CD6AF3   0325         CALL    PARAMS      ;INPUT NUMERIC PARAMETERS FROM
F15E  DDE1     0326         POP     IX          ; LINE BUFFER AND TEST IF ERROR
F160  3810     0327         JR      C,WHAT-$
F162  2A7CFF   0328         LD      HL,(PARAM1)
F165  ED5B7EFF 0329         LD      DE,(PARAM2)
F169  ED4B80FF 0330         LD      BC,(PARAM3)
F16D  CD80F1   0331         CALL    CALLX       ;CALL SUBROUTINE @ IX
F170  30B9     0332         JR      NC,PROMPT-$;GO BACK TO PROMPT IF NO ERRORS
               0333
F172  CDECF3   0334 WHAT:   CALL    PNEXT
F175  20776861 0335         DEFM    ' what ?'
      74203F
F17C  07       0336         DEFB    'G'-64      ;SAY 'what ?' AND BEEP THE BELL
F17D  04       0337         DEFB    EOT
F17E  18AB     0338         JR      PROMPT-$
               0339 ;
               0340 ;
F180  DDE9     0341 CALLX:  JP      (IX)        ;CALL SUBROUTINE @ IX
               0342 ;
               0343 ;
               0344 ;
F182  52       0345 CMDTAB: DEFB    'R'
F183  4F       0346         DEFB    'O'
F184  49       0347         DEFB    'I'
F185  47       0348         DEFB    'G'
F186  54       0349         DEFB    'T'
F187  46       0350         DEFB    'F'
F188  4D       0351         DEFB    'M'
F189  43       0352         DEFB    'C'
F18A  42       0353         DEFB    'B'
F18B  44       0354         DEFB    'D'
F18C  53       0355         DEFB    'S'
               0356
F18D  29F3     0357         DEFW    SWITCH      ;SWITCH CONSOLE OUTPUT VECTOR
F18F  05F2     0358         DEFW    MEMDMP      ;DUMP MEMORY IN HEX/ASCII
F191  A3F1     0359         DEFW    BOOT        ;BOOT UP CP/M
F193  E6F2     0360         DEFW    BLOCK       ;MEMORY BLOCK MOVE
F195  57F2     0361         DEFW    VIEW        ;MEMORY EXAMINE/CHANGE
F197  D8F2     0362         DEFW    FILL        ;FILL MEMORY
F199  8CF2     0363         DEFW    TEST        ;RAM DIAGNOSTIC
F19B  81F2     0364         DEFW    GOTO        ;JUMP TO MEMORY LOCATION
F19D  FEF2     0365         DEFW    INCMD       ;READ FROM INPUT PORT
```

(continued on top of page 12)

```
               0432 ;
               0433 ;
               0434 ;
               0435 ;-- MEMORY DUMP COMMAND --
               0436 ;
F205  3D       0437 MEMDMP: DEC     A           ;CHECK PARAMETER COUNT
F206  2806     0438         JR      Z,MDMP2-$
F208  3D       0439         DEC     A
F209  2808     0440         JR      Z,MDMP3-$
F20B  2A86FF   0441 MDMP1:  LD      HL,(LAST)
F20E  111000   0442 MDMP2:  LD      DE,16
F211  180D     0443         JR      MDMP3B-$
               0444
F213  EB       0445 MDMP3:  EX      DE,HL
F214  ED52     0446         SBC     HL,DE       ;DERRIVE BYTECNT FOR DUMP RANGE
F216  0604     0447         LD      B,4
F218  CB3C     0448 MDMP3A: SRL     H           ;DIVIDE BYTECOUNT BY 16
F21A  CB1D     0449         RR      L
F21C  10FA     0450         DJNZ    MDMP3A-$
F21E  23       0451         INC     HL
F21F  EB       0452         EX      DE,HL
F220  CD27F2   0453 MDMP3B: CALL    DUMP        ;DUMP DE*16 BYTES STRTING AT HL
F223  2286FF   0454         LD      (LAST),HL
F226  C9       0455         RET
               0456 ;
               0457 ;
F227  E5       0458 DUMP:   PUSH    HL          ;SAVE STARTING ADDRESS
F228  CDCDF3   0459         CALL    PUT4HS      ;PRINT STARTING ADDRESS IN HEX
F22B  CD02F4   0460         CALL    SPACE
F22E  0610     0461         LD      B,16
F230  7E       0462 DUMP2:  LD      A,(HL)      ;GET A DATA BYTE @ HL
F231  23       0463         INC     HL
F232  CDD2F3   0464         CALL    PUT2HS      ;PRINT THE DATA IN HEX
F235  10F9     0465         DJNZ    DUMP2-$     ;REPEAT 16 TIMES
F237  E1       0466         POP     HL          ;RESTORE STARTING ADDRESS
F238  0610     0467         LD      B,16
F23A  7E       0468 DUMP3:  LD      A,(HL)      ;GET BACK DATA BYTE @ HL
F23B  23       0469         INC     HL
F23C  CBBF     0470         RES     7,A
F23E  FE20     0471         CP      20H
F240  3804     0472         JR      C,DUMP4-$
F242  FE7F     0473         CP      7FH
F244  3802     0474         JR      C,DUMP5-$
F246  3E2E     0475 DUMP4:  LD      A,'.'       ;PRINT DOT IF DATA < 20 OR > 7F
F248  CD15F4   0476 DUMP5:  CALL    OUTPUT      ;PRINT ASCII CHARACTER IN A
F24B  10ED     0477         DJNZ    DUMP3-$
F24D  CDFCF3   0478         CALL    CRLFS
F250  C0       0479         RET     NZ          ;EXIT IF ESCAPE REQ INDICATED
F251  1B       0480         DEC     DE
F252  7A       0481         LD      A,D
F253  B3       0482         OR      E
F254  20D1     0483         JR      NZ,DUMP-$
F256  C9       0484         RET
               0485 ;
               0486 ;
               0487 ;
               0488 ;
               0489 ;-- MEMORY EXAMINE COMMAND --
               0490 ;
F257  CDCEF2   0491 VIEW:   CALL    MDATA
F25A  CD07F4   0492         CALL    ECHO
F25D  FE0D     0493         CP      CR
F25F  281B     0494         JR      Z,VIEW4-$
F261  FE2D     0495         CP      '-'
F263  2819     0496         JR      Z,VIEW5-$
F265  CDBDF3   0497 VIEW2:  CALL    ASCHEX
```

(continued next page)

PFM Monitor Listing (continued)

```
F268  3F                CCF
F269  D0                RET     NC
F26A  07                RLCA
F26B  07                RLCA
F26C  07                RLCA
F26D  07                RLCA
F26E  4F                LD      C,A
F26F  CD07F4            CALL    ECHO
F272  CDBDF3            CALL    ASCHEX
F275  3F                CCF
F276  D0                RET     NC
F277  B1                OR      C
F278  77        VIEW3:  LD      (HL),A
F279  CDB9F2            CALL    CHECK
F27C  23        VIEW4:  INC     HL
F27D  23                INC     HL
F27E  2B        VIEW5:  DEC     HL
F27F  18D6              JR      VIEW-$
                ;
                ;
                ;
                ;-- JUMP TO MEMORY LOCATION COMMAND --
                ;
F281  3D        GOTO:   DEC     A               ;CHECK PARAMETER COUNT
F282  37                SCF
F283  C0                RET     NZ
F284  E5                PUSH    HL
F285  DDE1              POP     IX
F287  CD80F1            CALL    CALLX           ;CALL ADDRESS PASSED IN HL
F28A  B7                OR      A
F28B  C9                RET                     ;RETURN IF WE GET BACK AGAIN
                ;
                ;
                ;
                ;-- MEMORY READ/WRITE DIAGNOSTIC COMMAND --
                ;
F28C  FE02      TEST:   CP      2               ;CHECK PARAMETER COUNT
F28E  37                SCF
F28F  C0                RET     NZ
F290  13                INC     DE
F291  5A                LD      E,D             ;GET ENDING PAGE ADDRESS INTO E
F292  54                LD      D,H             ;GET STARTING PAGE ADDR INTO D
F293  0600              LD      B,0             ;INITIALIZE PASS COUNTER
F295  62        TEST1:  LD      H,D             ;POINT HL TO START OF BLOCK
F296  2E00              LD      L,0
F298  7D        TEST2:  LD      A,L
F299  AC                XOR     H               ;GENERATE TEST BYTE
F29A  A8                XOR     B
F29B  77                LD      (HL),A          ;STORE BYTE IN RAM
F29C  23                INC     HL
F29D  7C                LD      A,H
F29E  BB                CP      E               ;CHECK FOR END OF TEST BLOCK
F29F  20F7              JR      NZ,TEST2-$
                ;NOW READ BACK EACH BYTE & COMPARE
F2A1  62                LD      H,D
F2A2  2E00              LD      L,0             ;POINT HL BACK TO START
F2A4  7D        TEST3:  LD      A,L
F2A5  AC                XOR     H               ;RE-GENERATE TEST BYTE DATA
F2A6  A8                XOR     B
F2A7  CDB9F2            CALL    CHECK           ;VERIFY MEMORY DATA STILL GOOD
F2AA  C0                RET     NZ              ;EXIT IF ESC REQ IS INDICATED
```

```
F2F7  EB                EX      DE,HL           ;HL & DE FOR BYTECOUNT
F2F8  D5                PUSH    DE
F2F9  C5                PUSH    BC
F2FA  D1                POP     DE              ;GET OLD BC INTO DE
F2FB  C1                POP     BC
F2FC  03                INC     BC              ;GET COUNT+1 INTO BC
F2FD  C9                RET
                ;
                ;
                ;
                ;
                ;-- READ FROM INPUT PORT COMMAND --
                ;
F2FE  3D        INCMD:  DEC     A               ;CHECK IF PARAMETER COUNT=1
F2FF  37                SCF
F300  C0                RET     NZ
F301  4D                LD      C,L             ;POINT C TO INPUT PORT
F302  CDFCF3    IN1:    CALL    CRLFS
F305  79                LD      A,C
F306  CDD2F3            CALL    PUT2HS
F309  ED78              IN      A,(C)
F30B  CDD2F3            CALL    PUT2HS
F30E  CD07F4            CALL    ECHO
F311  FE0D              CP      CR
F313  2806              JR      Z,IN2-$
F315  FE2D              CP      '-'
F317  2804              JR      Z,IN3-$
F319  B7                OR      A
F31A  C9                RET

F31B  0C        IN2:    INC     C
F31C  0C                INC     C
F31D  0D        IN3:    DEC     C
F31E  18E2              JR      IN1-$
                ;
                ;
                ;
                ;-- WRITE TO OUTPUT PORT COMMAND --
                ;
F320  FE02      OUTCMD: CP      2               ;CHECK IF PARAMETER COUNT=2
F322  37                SCF
F323  C0                RET     NZ
F324  4D                LD      C,L             ;POINT C TO OUTPUT PORT
F325  ED59              OUT     (C),E           ;OUTPUT DATA PASSED IN E
F327  B7                OR      A
F328  C9                RET
                ;
                ;
                ;-- SWITCH CONSOLE OUTPUT DEVICE COMMAND --
                ;
F329  2185FF    SWITCH: LD      HL,COFLAG
F32C  34                INC     (HL)            ;TOGGLE CONSOLE OUT TYPE FLAG
F32D  CB46.             BIT     0,(HL)
F32F  21FEF4            LD      HL,SIOOUT
F332  2803              JR      Z,SWIT2-$       ;JUMP IF ZERO TO ONE TRANSITION
F334  2120F5            LD      HL,CRTOUT
F337  220DF0    SWIT2:  LD      (CONOUT+1),HL   ;STORE NEW CNSL OUT ADDR
F33A  C9                RET
                ;
                ;
                ;**************************************************
                ;*                                                *
                ;*     CONSOLE I/O PACKAGE AND UTILITY ROUTINES   *
                ;*                                                *
                ;**************************************************
                ;
                ;
```

```
F2AB  23        0559         INC     HL          ;ELSE GO ON TO NEXT BYTE
F2AC  7C        0560         LD      A,H
F2AD  BB        0561         CP      E           ;CHECK FOR END OF BLOCK
F2AE  20F4      0562         JR      NZ,TEST3-$
F2B0  04        0563         INC     B           ;BUMP PASS COUNT
F2B1  3E2B      0564         LD      A,'+'
F2B3  CD15F4    0565         CALL    OUTPUT      ;PRINT '+' AND ALLOW FOR EXIT
F2B6  28DD      0566         JR      Z,TEST1-$   ;DO ANOTHER PASS IF NO ESCAPE
F2B8  C9        0567         RET
                0568 ;
                0569 ;
                0570 ;
F2B9  BE        0571 CHECK:  CP      (HL)
F2BA  C8        0572         RET     Z           ;RETURN IF (HL)=A
F2BB  F5        0573         PUSH    AF
F2BC  CDCEF2    0574         CALL    MDATA       ;PRINT WHAT WAS ACTUALLY READ
F2BF  CDECF3    0575         CALL    PNEXT
F2C2  73686F75  0576         DEFM    'should='
      6C643D
F2C9  04        0577         DEFB    EOT
F2CA  F1        0578         POP     AF
F2CB  C3D2F3    0579         JP      PUT2HS      ;PRINT WHAT SHOULD HAVE READ
                0580 ;
                0581 ;
F2CE  CDFCF3    0582 MDATA:  CALL    CRLFS
F2D1  CDCDF3    0583         CALL    PUT4HS
F2D4  7E        0584         LD      A,(HL)
F2D5  C3D2F3    0585         JP      PUT2HS
                0586 ;
                0587 ;
                0588 ;
                0589 ;-- FILL MEMORY WITH CONSTANT COMMAND --
                0590 ;
F2D8  FE03      0591 FILL:   CP      3           ;CHECK IF PARAMETER COUNT=3
F2DA  37        0592         SCF
F2DB  C0        0593         RET     NZ
F2DC  71        0594 FILL1:  LD      (HL),C
F2DD  E5        0595         PUSH    HL
F2DE  B7        0596         OR      A
F2DF  ED52      0597         SBC     HL,DE       ;COMP HL TO END ADDRESS IN DE
F2E1  E1        0598         POP     HL
F2E2  23        0599         INC     HL          ;ADVANCE POINTER AFTER COMPARE
F2E3  38F7      0600         JR      C,FILL1-$
F2E5  C9        0601         RET
                0602 ;
                0603 ;
                0604 ;
                0605 ;
                0606 ;-- MEMORY BLOCK MOVE COMMAND --
                0607 ;
F2E6  FE03      0608 BLOCK:  CP      3           ;CHECK IF PARAMETER COUNT=3
F2E8  37        0609         SCF
F2E9  C0        0610         RET     NZ
F2EA  CDF3F2    0611         CALL    BLOCAD
F2ED  79        0612         LD      A,C
F2EE  B0        0613         OR      B
F2EF  C8        0614         RET     Z           ;EXIT NOW IF BC=0
F2F0  EDB0      0615         LDIR
F2F2  C9        0616         RET
                0617 ;
                0618 ;
                0619 ;
F2F3  EB        0620 BLOCAD: EX      DE,HL
F2F4  B7        0621         OR      A           ;CLEAR CARRY
F2F5  ED52      0622         SBC     HL,DE       ;GET DIFFRENCE BETWEEN
```

(continued on top of page 14)

```
                0690 ;
F33B  41        0691 GETLIN: LD      B,C         ;SAVE MAX LINE LNGTH PARAM IN B
F33C  CD07F4    0692 GLIN1:  CALL    ECHO        ;GET A CHAR FROM THE CONSOLE
F33F  FE0D      0693         CP      CR          ;CHECK FOR CARRIAGE RETURN
F341  280E      0694         JR      Z,GLIN2-$
F343  FE08      0695         CP      'H'-64      ;CHECK FOR CTL-H BACKSPACE
F345  280C      0696         JR      Z,GLIN4-$
F347  FE20      0697         CP      ' '
F349  D8        0698         RET     C           ;OTHER CONT CHARACTERS ILLEGAL
F34A  77        0699         LD      (HL),A
F34B  23        0700         INC     HL          ;STORE CHARACTER IN BUFFER
F34C  0D        0701         DEC     C
F34D  20ED      0702         JR      NZ,GLIN1-$  ;GET ANOTHER IF MORE ROOM
F34F  37        0703         SCF
F350  C9        0704         RET                 ;RETURN WITH CARRY=1 IF TOO
                0705                             ;MANY CHARACTERS ARE ENTERED
F351  77        0706 GLIN2:  LD      (HL),A      ;PUT <CR> ON END OF LINE
F352  C9        0707         RET                 ;RETURN WITH CARRY BIT=0
                0708
F353  2B        0709 GLIN4:  DEC     HL          ;DELETE LAST CHAR FROM BUFFER
F354  CDECF3    0710         CALL    PNEXT
F357  2008      0711         DEFB    ' ','H'-64  ;PRINT A SPACE TO OVERWRITE THE
F359  04        0712         DEFB    EOT         ;LAST CHAR,THEN DO A BACKSPACE
F35A  0C        0713         INC     C
F35B  78        0714         LD      A,B         ;MAKE SURE YOU'RE NOT TRYING TO
F35C  91        0715         SUB     C           ;<BS> PAST START OF THE LINE
F35D  30DD      0716         JR      NC,GLIN1-$
F35F  C9        0717         RET
                0718 ;
                0719 ;
                0720 ;
F360  EDB1      0721 SEARCH: CPIR                ;SRCH TBL @HL FOR MATCH WITH A
F362  C0        0722         RET     NZ          ;EXIT NOW IF SEARCH FAILS
F363  09        0723         ADD     HL,BC
F364  09        0724         ADD     HL,BC       ;+ RESIDUE FROM CPIR BYTECOUNT
F365  09        0725         ADD     HL,BC       ;TO HL 3 TIMES TO GET POINTER
F366  4E        0726         LD      C,(HL)      ;TO ADDRESS PART OF TABLE ENTRY
F367  23        0727         INC     HL
F368  46        0728         LD      B,(HL)
F369  C9        0729         RET                 ;EXIT WITH Z=1 TO SHOW MATCH
                0730 ;
                0731 ;
                0732 ;
                0733 ;
F36A  010000    0734 PARAMS: LD      BC,0
F36D  FD7E00    0735         LD      A,(IY+0)
F370  FE0D      0736         CP      CR          ;CHECK IF LINE TERMINATES
F372  2008      0737         JR      NZ,PARA2-$  ; IMMEDIATELY WITH A RETURN
F374  AF        0738         XOR     A
F375  C9        0739         RET                 ;RET WITH PARAM COUNT=0 IF SO
                0740
F376  0C        0741 PARA1:  INC     C
F377  0C        0742         INC     C
F378  CB59      0743         BIT     3,C
F37A  37        0744         SCF
```

(continued next issue)

SUBSCRIPTION FORM

(It's OK to brag!)

□ I own a big board (Hooray!)
□ I don't own a Big Board but am very interested (There's hope)

	EXPERTISE Guru=5 Novice=0	INTEREST Fanatic=5 None=0
Software Systems	□	□
Software Applications	□	□
Languages 1. ______________	□	□
2. ______________	□	□
3. ______________	□	□
Hardware	□	□

Are you willing to be a resource in the areas where your expertise is 4 or 5?

love to □
probably □
maybe □
no □

How are you using the Big Board?

Home System □
Business System □
Software Development □
OEM □
Education □
Other ________ □

What kinds of information do you need right now?

____
____
____
____
____
____

What are your hardware/software needs now?

____
____

In the near future? ______________________
____
____

What kind of exciting adventure (misadventure) are you working on? ______________________
____
____
____
____

If you get the idea that this document is as interested in enlisting your aid and ideas as it is in getting a subscription, you're right. Lots of people are willing to subscribe, lots of people have ideas - and we'd like to encourage lots of people (especially you) to take an hour or two and put ideas and needs and accomplishments down on paper or disk. Then we can pass them along to others and that's what this journal is all about.

Send me six issues (1 yr.) of **MICRO CORNUCOPIA**. I understand that I can cancel at any time and receive a refund for the balance of the subscription. (Issue #1 was published in August 1981.)

U.S.
□ $16.00
□ $20.00 (1st class mail)
□ Back issues, Specify #s ________
$3.00 each

Canada & Mexico
□ $20.00 (U.S. funds)
□ Back issues, Specify #s ________
$3.00 each (U.S. funds)

Other Foreign
□ $26.00 (U.S. funds)
□ Back issues, Specify #s ________
$5.00 each (U.S. funds)

NAME ______________________________ PHONE (?) ______________

ADDRESS ______________________________________

CITY ______________________________ STATE ________ ZIP ________

□ Renewal

MICRO CORNUCOPIA • 11740 N.W. West Rd • Portland, Oregon • 97229

MICRO CORNUCOPIA
Journal of the Big Board Users Group

11740 NW WEST ROAD
PORTLAND, OREGON 97229