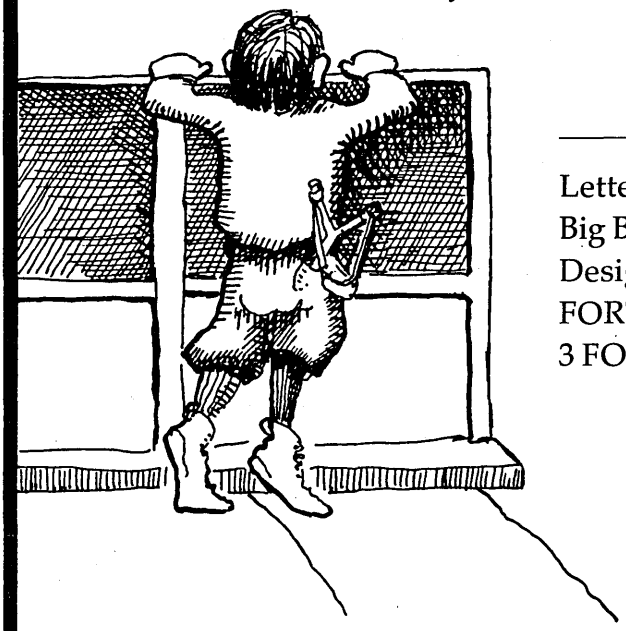
$3.00

MICRO CORNUCOPIA

JOURNAL OF THE BIG BOARD USERS

February 1982 No. 4

TABLE OF CONTENTS



REGULAR FEATURES

NEW PRICES EFFECTIVE ON THE 1ST DAY OF THE COVER MONTH OF THIS MAGAZINE

NEW LOWER PRICES

"THE BIG BOARD"

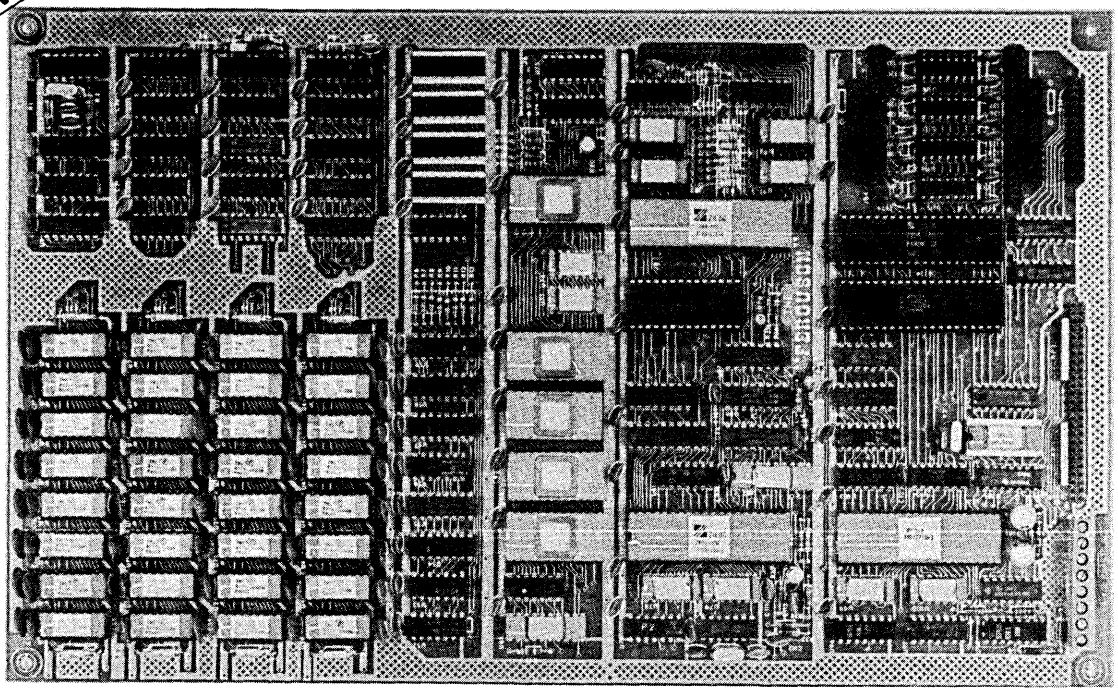
OEM - INDUSTRIAL - BUSINESS - SCIENTIFIC

SINGLE BOARD COMPUTER KIT!

Z-80 CPU! 64K RAM!

NEW!

PARTIALLY ASSEMBLED KITS
For All Sockets Installed
And Soldered Add $50.



WANT MORE INFO?
Full Documentation and
Schematics — $5.

THE BIG BOARD PROJECT: Three years in the works, and maybe too good to be true. A tribute to hard headed, no compromise, high performance, American engineering! The Big Board gives you all the most needed computing features on one board at a very reasonable cost. The Big Board was designed from scratch to run the latest version of CP/M*. Just imagine all the off-the-shelf software that can be run on the Big Board without any modifications needed! Take a Big Board, add a couple of 8 inch disc drives, power supply, an enclosure, C.R.T., and you have a total Business System for about 1/3 the cost you might expect to pay.

$399.00 ** (64K KIT BASIC I/O)

SIZE: 8½ x 13¾ IN.
SAME AS AN 8 IN. DRIVE.
REQUIRES: +5V @ 3 AMPS
+ - 12V @ .5 AMPS.

FULLY SOCKETED!

FEATURES: (Remember, all this on one board!)

64K RAM

Uses industry standard 4116 RAM'S. All 64K is available to the user, our VIDEO and EPROM sections do not make holes in system RAM. Also, very special care was taken in the RAM array PC layout to eliminate potential noise and glitches.

Z-80 CPU

Running at 2.5 MHZ. Handles all 4116 RAM refresh and supports Mode 2 INTERUPTS. Fully buffered and runs 8080 software.

SERIAL I/O (OPTIONAL)

Full 2 channels using the Z80 SIO and the SMC 8116 Baud Rate Generator. FULL RS232! For synchronous or asynchronous communication. In synchronous mode, the clocks can be transmitted or received by a modem. Both channels can be set up for either data-communication or data-terminals. Supports mode 2 Int. Price for all parts and connectors: $49

BASIC I/O

Consists of a separate parallel port (Z80 PIO) for use with an ASCII encoded keyboard for input. Output would be on the 80 x 24 Video Display.

24 x 80 CHARACTER VIDEO

With a crisp, flicker-free display that looks extremely sharp even on small monitors. Hardware scroll and full cursor control. Composite video or split video and sync. Character set is supplied on a 2716 style ROM, making customized fonts easy. Sync pulses can be any desired length or polarity. Video may be inverted or true. 5 x 7 Matrix - Upper & Lower Case

FLOPPY DISC CONTROLLER

Uses WD1771 controller chip with a TTL Data Separator for enhanced reliability. IBM 3740 compatible. Supports up to four 8 inch disc drives. Directly compatible with standard Shugart drives such as the SA800 or SA801. Drives can be configured for remote AC off-on. Runs CP/M* 2.2.

TWO PORT PARALLEL I/O (OPTIONAL)

Uses Z-80 PIO. Full 16 bits, fully buffered, bi-directional. User selectable hand shake polarity. Set of all parts and connectors for parallel I/O: $19.95

REAL TIME CLOCK (OPTIONAL)

Uses Z-80 CTC. Can be configured as a Counter on Real Time Clock. Set of all parts: $9.95

BLANK PC BOARD — $149

The blank Big Board PC Board comes complete with full documentation (including schematics), the character ROM, the PFM 3.3 MONITOR ROM, and a diskette with the source of our BIOS, BOOT, and PFM 3.3 MONITOR.

CP/M* 2.2 FOR BIG BOARD

The popular CP/M* D.O.S. to run on Big Board is available for $159.00.

PRICE CUT!

PFM 3.3 2K SYSTEM MONITOR

The real power of the Big Board lies in its PFM 3.3 on board monitor. PFM commands include: Dump Memory, Boot CP/M*, Copy, Examine, Fill Memory, Test Memory, Go To, Read and Write I/O Ports, Disc Read (Drive, Track, Sector), and Search. PFM occupies one of the four 2716 EPROM locations provided.
Z-80 is a Trademark of Zilog.

Digital Research Computers
(OF TEXAS)
P.O. BOX 401565 • GARLAND, TEXAS 75040 • (214) 271-3538

TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order, VISA, MC, cash accepted. We will accept COD's (for the Big Board only) with a $75 deposit. Balance UPS COD. Add $4.00 shipping.
USA AND CANADA ONLY

U.K. AND EUROPE: CONTACT VINCELORD LTD. 47 GOLDHURST TERRACE LONDON NW6

*TRADEMARK OF DIGITAL RESEARCH. NOT ASSOCIATED WITH DIGITAL RESEARCH OF CALIFORNIA, THE ORIGINATORS OF CPM SOFTWARE
**1 TO 4 PIECE DOMESTIC USA PRICE.

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)	$16.00
1 yr. (Canada & Mexico)	$20.00
1 yr. (other foreign)	$26.00

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.

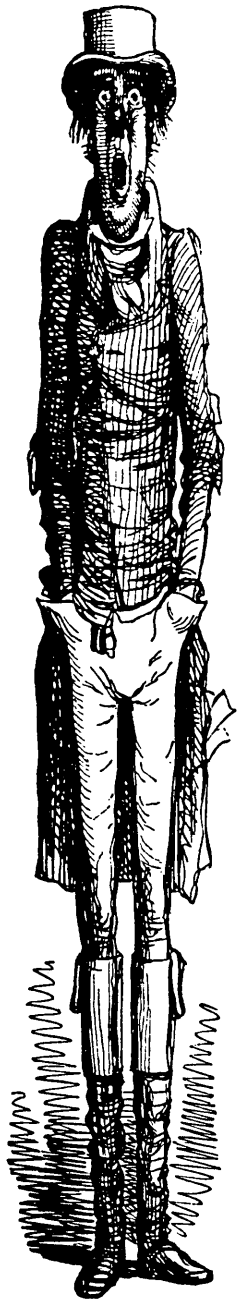
SOFTWARE, HARDWARE, AND BOOK VENDORS: Micro Cornucopia is establishing a group of reviewers. We would very much like to review your Big Board compatible products for Micro C. Please send material to Review Editor, Micro Cornucopia.

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.

MICRO CORNUCOPIA

Feb. 1982 **The Journal of the Big Board Users** **No.4**



Extra! Extra!
Read All About It!

Can't Leave Well Enough Alone

Two Exclusives!

Just when things seem to be settling down to a dull roar someone decides things are just too quiet and they light off a firecracker. Actually, the way news has been breaking around here it sounds like someone is celebrating an early July 4th.

First, I found out that Russell Smith and Jim Ferguson are coming out with a whole new Z80-based single board computer with such goodies as graphics, DMA, STD bus interface, double density, up to 5 MHz, and 64 K RAM chips. Because of the STD bus interface, the RAM is expandable. They are planning to supply it in kit form, as a finished board, and as a complete system. More information on this as it's available.

Second, Jim Tanner is reducing the price of the basic Big Board package from $650 to $499 on February 1, the publication date of this issue. You can say you saw it first right here in the pages of Micro C.

FORTH in ROM

Quite a number of folks have written asking when FORTH in ROM would be available. Well, it's available now. It is FIG FORTH complete with disk I/O customized for the Big Board. Now you can run FORTH even if you don't yet have CP/M, or even a disk drive. The price is $65.00 for the three chip set and documentation. We also have the disk editor available (on disk) for $15.00. Again, no CP/M is required.

Other ROMs are also available. If you need a new character ROM but don't want to part with your present one, or if you need a faster monitor ROM to run 4 MHz, I will burn it for you. (See the Micro C ad.) If you want a copy of the monitor ROM, please either send me the old one to copy or send the serial number of your board. I wouldn't feel right making copies of the monitor for folks who don't have Big Boards.

Other Products

We are contacting the manufacturers of the products we review (products that look particularly good) about making the items available through Micro C at a discount. Check the Micro C ad each issue for information on what's available. If you are using something fantastic that we haven't mentioned in the magazine, holler!

(continued on page 19)

Letters

Dear Editor,

Please send user's disk #1, and advertising rates. I have 5, 10, and 20 M hard disks available for the Big Board. The price is $2825 for 5 Meg, $3945 for 10 Meg, and 4715 for 20 Meg. Expect delivery in February.

Jim Thompson
41 King St.
Redwood City, CA 94062

Dear Editor,

I want to set up my Big Board as a lightweight terminal for accessing my employer's scientific computer. Dragging a TI Silent 700 through airports has made my right arm 2" longer than the left.

I have a Sanyo 9" monitor that would fit into my briefcase along with the Big Board and power supply. So the Big Board seems like a good way to go but since I wouldn't have disk drives I would need the terminal software in ROM.

Basically the software I need would do minimal word processing and dumb terminal emulation so I could create text and move it to and from the main computer.

Is anyone interested in doing it? I have very little loose money but I'll bet we could work out a very profitable trade.

Thomas Mason
Professional Engineer
2402 Audubon Rd
Akron, OH 44320

Dear Editor,

Four MHz has been a problem. After making the mods the board runs until I hit reset. Then the processor goes into the weeds and I can't recover unless I power down. Why? What's the fix? I use 150 ns RAM and Z80A parts.

Also, my keyboard entries randomly change from upper case to lower. I suspect that the stash subroutine at F333H is changing the lock byte at FF33H or the flag byte at FF34H. How about it?

Finally, how do you load a CBIOS and CP/M without making calls into the monitor. Then how do you recover when reset time comes. Oops, lost the CBIOS and back into the PFM monitor.

Keep up the good work on the magazine.

Larry Blazek
3210 Echo Rd
Sapulpa, Ok 74066

Editor's note:

The monitor ROM is probably just barely making it when it is cool and as it warms up, it slows down. Try using a 2716-1 for the monitor. Fast monitor ROMs are available from Micro C. As for the other questions, what say anyone?

Dear Editor,

I thoroughly enjoy your magazine. The Direct Input Routine on page 7 of issue #2 can be done more easily in Microsoft Basic version 5.1 or higher.

```
10 L$=INKEY$
20 IF L$ <> " "
   THEN [DO SOMETHING]
   ELSE [DO SOMETHING ELSE]
```

INKEY$ does not hang up the program if there is no input.

Dr. Sandy Warshaw
Laurence Livermore Lab
PO Box 808 l-461
Livermore, CA 94550

Editor's note:

Another tip for Microsoft users (this one undocumented). To use random access records larger than 128 bytes each, you have to call the Basic program as follows:

MBASIC FILENAME /S:(BYTES/RECORD)

Dear Editor,

A thousand thanks for your prompt reply with issue #3 of Micro C. I was quite overwhelmed with excitement (not a lot happens on a Thursday in Nunawading, Australia). Imagine finding out that there are others doing the same things I am and having the same hassles.

From your brief description of the first 2 issues, it appears that they cover most of the problems I've just fought my way through and one I've just started to work on. My MX-100 printer arrived on Wednesday (you can imagine how I was on Wednesday) but I am already tired of just looking at it in admiration and running the self test. So I have decided to actually connect it to the Big Board.

Congratulations on a first class publication both technically and philosophically.

Do you have many Australian members? I know that at least 300 to 600 Big Boards have been sold here. If you are interested in further exposure here I would be happy to distribute any sample issues and promotional material to the local CP/M and microcomputer user groups.

Articles I'd like to see include a terminal program in ROM and an Epson plotting package.

Michael Staindl
63 Mt Pleasant Rd
Nunawading, Australia 3131

Editor's note:

Thank you for the comments Mike. We will be sending you some material about Micro C that you can pass around. We have only a couple of subscribers in Australia and we would really like to reach those who don't know about us.

We'd also like to reach all those other folks in the US, Canada, etc. who either don't know about us or just haven't bothered to get a copy of the magazine to see what the group is doing. If any of you know of such a person or group, let us know and we will make sure they at least hear about the Big Board User's group. (In other words, rat on you friends, it's for their own good.)

■ ■ ■

ANNOUNCING
THE BIG BOARD ADD-ON

****** FEATURES ******

Program 2708, 2716, 2732, and 2764 type EPROMS. With four programming sockets you can program lots of memory at once. Programs EPROMS sequentially or in parallel for small production runs.

Second bank (64K) of memory will allow fast screen swaps, larger EPROM program storage, etc. Memory is fully-static 6116 CMOS type RAMS which will allow RAM/EPROM intermixing. Battery back-up for CMOS RAM.

More goodies. Sixteen channels of both 8-bit A/D and D/A conversion. Connect any of the FCC approved modems by NOVATION directly into the board. Plus, there's an S-100 connector which will allow you to connect an S-100 card directly or interface with an S-100 motherboard. And speaking of features, voice output with the on-board VOTRAX phoneme generator chip.

All this plus: four serial channels, four parallel ports, everything socketed, and it runs at 4 MHz. (In fact, with all this, you may just forget about the Big Board altogether!)

This is the board you have been waiting for. This board is intended for the serious builder and the novice alike. It will be available in bare board, full kit, partial kits, and assembled and tested. The board has the same dimensions as the BIG BOARD so it piggy-backs into the same space. Available 3/20/82.

Bare board .....$ 99.00
Complete package ..... CALL

****** OTHER BIG BOARD ACCESSORIES ******

Big Board power supply kit (BB + 2 DRIVES +) .....$ 85.00
Big Board power supply A&T 135.00
C.ITOH 8510 9xN matrix, graphics, 5 char sets printer (ser.) 645.00
C.ITOH 8510 same as above but par. interface..... 595.00
4Mhz mod that WORKS.... $7.50 BIG BOARD PARTS..... CALL

E.C.R.L.,INC.
P.O. BOX 387
CANBY, OREGON 97013-0387
503-266-4982 *24 HRS* or 503-656-3382

Big Board Notes

By David Thompson

Video Jitter

Numerous folks have contacted me about problems they are having with video jitter. I had the same problem so I put an oscilloscope (Tektronix) on the video crystal and noticed a harmonic in the waveform. I tried changing the feedback capacitor from 33 pF to 80 pF but that didn't help much. Finally I talked to Tom Brandt, a local Big Board guru, and he mentioned that the 74LS04 (U11) oscillator sometimes had to be hand selected or replaced with a 74S04 to cure the problem. There are several 74LS04s on board so it is easy to swap them around.

System Reset

I can finally sympathize with all you folks who got alternating garbage on the screen the first time you fired up your Big Board, and then spent hours and hours trying to figure out why. It happened to me when I fired up my second board.

One thing I noticed immediately was that nothing happened when I reset the system. So I probed around with my trusty scope and found that the reset line to the Z80 (U80-pin 26) was not going low when I hit the reset button. Well, one thing led to another. The reset button input is clocked through the reset circuit by the M1B line. The M1B (buffered) came from the Z80 M1, and M1 wasn't moving. (Nor was the refresh line, etc.) I replaced all the chips in the kernel but it didn't help.

So I borrowed a small logic analyzer. It showed me that the power-up reset went away before the 20 MHZ clock got started. Thus, the system never got initialized and there was no M1 to clock through another reset. Since M1 obviously wasn't going to be there when I needed it, I decided to do without it and put the reset button across C141 (located in the lower left-hand side of schematic 1.)

When I hit the reset button, there was a short pause and then the screen went blank (hooray). When I hit the carriage return, there it was, 'PFM-80 . . . ' (And at two o'clock in the morning, that one little message was Pretty F . . . . . Magic!)

Serial Port A Problem

I decided to use port A to drive my Diablo printer. After pages and pages of manuals about configuring the SIO and configuring the Diablo so they would speak to each other (it was worse that our two little girls), I got them on speaking terms. Occasionally, however, the Diablo heard strange things.

After additional hours of head scratching (causes wear in strange places), I went for help. Lynn Cochran and a trusty 834 Data Comm Tester determined that the data I was sending out was just fine, but the Diablo was still having indigestion. It turned out that the DSR A line was the problem.

The schematic shows both port A and port B DSR lines tied to +12 V through 4.7K resistors. However, the port A resistor (R45) is really tied to +5 V. (Obviously a board layout error.) So cross talk between the lines in the RS-232 cable caused the DSR line to occasionally go negative, shutting down the receiver in the Diablo. There are two fixes: change R45 to 1K ohms, or pull its 5 V lead out of the hole and wire it to +12 V.

Bad Parts

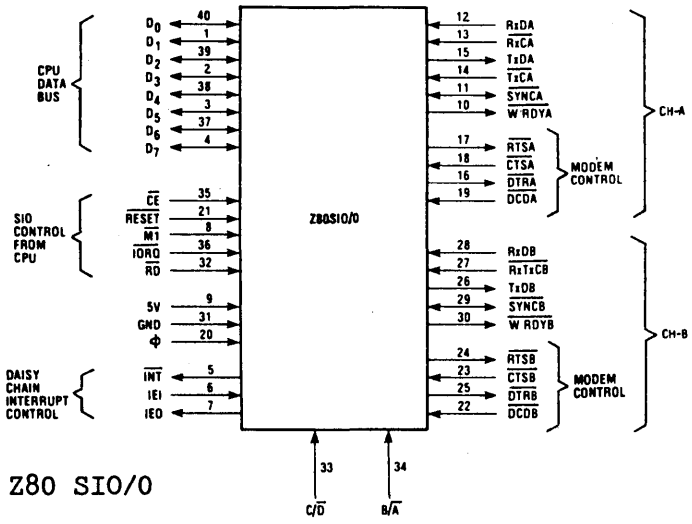
In the last two boards that I put together I found two bad parts, a 74LS123 flip/flop and a 74LS242 bi-directional buffer. Several other folks have found defective 74LS123s (this includes folks who got complete parts sets and folks who gathered their own).

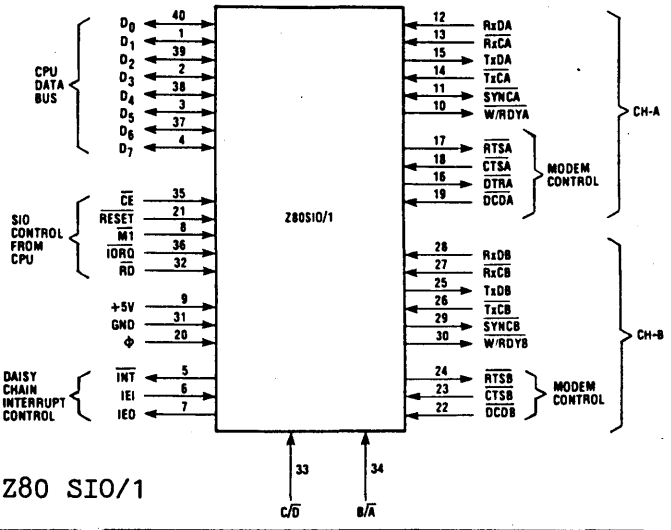
Using an SIO-1

I'm using an SIO-1 rather than an SIO-0 for the serial interface. (It was quite a bit cheaper and it was available.) The port A side of the SIO-1 is identical to the SIO-0, the port B side is slightly different. See the illustration for the pinouts. Anyway, I set up serial port B as follows: connect pin 21 to pin 8 (xmit data), pin 11 to pin 12 (rec data), pin 15 to pin 16 (req to send), and pin 19 to pin 20 (clear to send).

I am using port B to drive an Epson MX80. The P.COM and PR.COM programs on user disk #1 provide the software interface.

■ ■ ■





More PFM-80

By Don Retzlaff
6435 Northwood
Dallas, TX 75225

In issue #1 I talked about the entry point table in PFM-80. The first entry point is the cold start which sets up the monitor to go to work. Control is then passed to the warm start entry (which does not reinitialize the system).

Following the warm start, the monitor displays the '*' and waits for a command.

Commands:

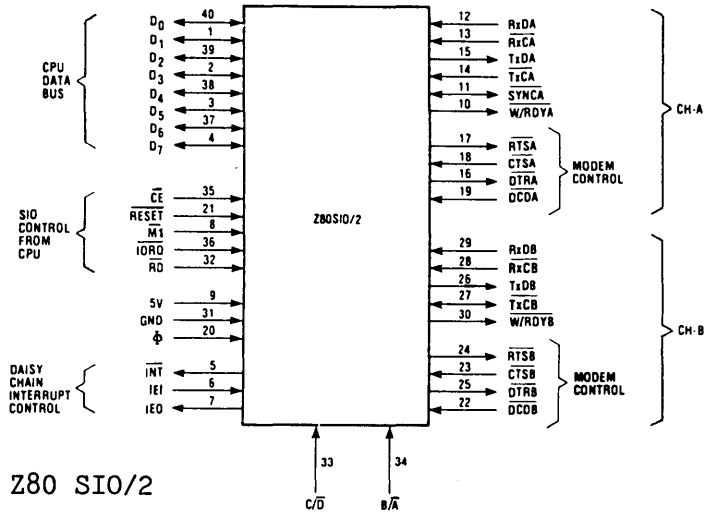
R —Read a disk sector
O—Write to output port
I —Read from input port
G—Go to specified address
T —Test memory
F —Fill memory with character
M—Memory examine/change
C —Move data in memory
B —Boot CP/M
D—Dump memory to display
S —Switch console output

Of the above commands, only the 'S' command is not well documented in the PFM-80 manual. It lets you switch the output of the monitor from the video display to the serial output port B. This makes it easy to bring up the system normally and then switch to a hard copy unit such as a thermal printer or DEC Writer.

When 'S' is typed on the Big Board keyboard, the output is transferred back to the Big Board CRT. Note that only the output is switched, not the input. When 'S' is typed again, the output is switched back to the printer. *(continued next column)*

(More PFM-80 continued)

Another interesting command is 'B' (boot). It is amazingly simple. When 'B' is typed, the monitor jumps to the label BOOT. Drive A is selected when 0 is placed in register C and the monitor calls the subroutine SELECT. PFM calls HOME (which moves the head to track 0) and then puts 0080H in the HL registers (to point to where the first sector will be stored). The first sector is read into memory and then control is passed to 0080H. From here on, CP/M has control of the loading.



Z80 SIO/2

The First 16 Bytes

By David Thompson

Sig Peterson left a note on my bench a while back and on that note was a 16-byte listing. This was a listing of the first 16 bytes of the monitor ROM: the invisible 16 bytes that don't get loaded up into high memory.

This information should help those of you who are having trouble bringing up your boards if you can get a logic analyzer. On power-up, the system sets PIO bit seven (selecting the memory bank with the monitor ROM in it) and jumps to address 0000.

One place to look if your system is not coming up is PIO bit 7. If that isn't getting set to 1 when the reset button is pressed then the Z80 isn't even getting to the ROM.

If the bit is getting set, then check whether the Z80 is jumping to address 0000. If it is, then you should see the registers get loaded and see data being shuffled up into high memory. ■ ■ ■

FIRST 16 BYTES OF THE PFM MONITOR ROM

```
0000    DI                  ;Disable Interupts
0001    LD      HL,0010     ;Source starts at 0010
0004    LD      DE,F000     ;Destination at F000
0007    LD      BC,0800     ;Move 800 (Hex) bytes
000A    LDIR                ;Now move them
000C    JP      F000        ;OK, operate out of it
000F    NOP                 ;A No-Op place holder
```

Using the monitor

The monitor is great for testing memory, exercising the I/O ports, dumping and loading memory, and looking at disk data. *(Editor's note: See Undoing the Fatal Erase and Designer's Corner for examples.)*

You can get into the monitor anytime you are in DDT by jumping to location F003H (GF003). If you enter a second argument in the 'G' command (a breakpoint), then you can return to DDT (and CP/M) by simply jumping to the address you selected as a breakpoint. (Pick an unused address.)

Example: GF003,FFF0

Control is transferred from DDT to the warm start entry in the monitor but DDT is still hanging around in the background just waiting to see the program counter set at FFF0H. If you enter GFFF0 (while in the monitor) you will return to DDT.

Error Department

I have found an error in the PFM-80 article in issue #1. The RAM locations used by PFM for data storage are FF00H through FFC8H, not FFA8H. ■ ■ ■

Translating Your Keyboard

By Daryl Coulthart

532 Lake Bayview Ct
Shoreview MN 55112

This program translates the HEX value of the keyboard characters to whatever value you want. This routine is especially nice for defining single-keystroke control codes such as CNTL/C, etc.

For example, I have a reset key on my keyboard but it generates a HEX A4, so I use my translation table to change it into a CNTRL/C. So my reset key does a reset and a warm boot. Neat huh?

The portion called TABLE is a 256-byte table that contains a value for each character entered at the keyboard. This will be particularly useful if you have a keyboard that has extra keys or if you have keys that generate characters you don't need.

I have an 'Adds' keyboard with an extra 15 keys. These are word processing keys like PREV PAGE, NEXT PAGE, INSERT, and DELETE. However, I have no backspace or carriage return keys. So I wrote this program to change the key codes I didn't need into codes I needed.

One disadvantage of this program is that it overlays the monitor so that is has to be rerun each time the system is powered-up or reset. However, this also makes it easy to do different translations for different software. You could use different tables for Word Star, Word Master, Basic, or whatever.

This program is executed by simply entering its name. When it completes it returns to CP/M by branching to address 0.

I hope this is useful for others in the group—it's useful for me.

Editor's note: This routine opens up a lot of possibilities for customizing keyboards and customizing software that uses CNTL-KEY sequences. For instance, different text editors use different CNTL-KEY sequences to do the same function. It would be easy to set up a custom translation program for each editor so that the same CNTL-KEYs would do the same things.

Also, this program would make it very easy to set up a real custom keyboard such as a Dvorak or American Simplified. Touch typists often manage to double their speed using these new key layouts. We'll cover this subject more thoroughly in a future issue.

Translate Listing

```
        ORG     0100H       ;START WHERE CP/M LIKES TO
        NOP                 ;BURP
        NOP                 ;BURP
        NOP                 ;BURP
START   LD      BC,0100H    ;MOVE 256 BYTES, 1 PAGE
        LD      DE,0FE00H   ;MOVE TRANSLATE TABLE TO FE00, NOT USED BY MON
        LD      HL,TABLE    ;MOVE THE TRANSLATE TABLE
        LDIR                ;MOVE IT
        LD      BC,000AH    ;MOVE THE 10 BYTE TRANSLATE PROGRAM
        LD      DE,0FD00H   ;MOVE IT TO FD00
        LD      HL,PROG     ;POINT TO TRANSLATE PROGRAM
        LDIR                ;MOVE IT
        LD      BC,0003H    ;MOVE 3 BYTE ZAP THAT OVERLAYS MONITOR
        LD      DE,0F468H   ;OVERLAY ADDRESS F468 'A CALL TO INDEX'
        LD      HL,ZAP      ;POINT TO ZAP
        LDIR                ;MOVE IT WITH OVERKILL
EXIT    JP      00000H      ;RETURN TO CP/M
ZAP     CALL    PROG1       ;CALL THE TRANSLATE ROUTINE AT FD00
PROG1   EQU     0FD00H      ;
PROG    PUSH    HL          ;SAVE
        LD      H,0FEH      ;LOAD PAGE ADDRESS OF TABLE
        LD      L,C         ;LOAD THE CHARACTER TO TRANSLATE FROM KEYBOARD
        LD      C,(HL)      ;DO THE TRANSLATE
        POP     HL          ;RESTORE
INDEX   EQU     0F474H      ;CALL INDEX ROUTINE AT F474
        CALL    INDEX       ;CALL IT BECAUSE WE ZAPPED THIS INSTRUCTION
        RET                 ;NOW BACK TO MONITOR AND CONTIUNE KEYBOARD I/O
```

```
TABLE   DEFB    000H
        DEFB    001H
        DEFB    002H
        DEFB    003H
        DEFB    004H
        DEFB    005H
        DEFB    006H
        DEFB    007H
        DEFB    008H
        DEFB    009H
        DEFB    00AH
        DEFB    00BH
        DEFB    00CH
        DEFB    00DH
        DEFB    00EH
        DEFB    00FH
        DEFB    010H
        DEFB    011H
        DEFB    012H
        DEFB    013H
        DEFB    014H
        DEFB    015H
        DEFB    016H
        DEFB    017H
        DEFB    018H
        DEFB    019H
        DEFB    01AH
        DEFB    01BH
        DEFB    01CH
        DEFB    01DH
        DEFB    01EH
        DEFB    01FH
        DEFB    020H
        DEFB    021H
        DEFB    022H
        DEFB    023H
        DEFB    024H
        DEFB    025H
        DEFB    026H
        DEFB    027H
        DEFB    028H
        DEFB    029H
        DEFB    02AH
        DEFB    02BH
        DEFB    02CH
        DEFB    02DH
        DEFB    02EH
        DEFB    02FH
        DEFB    030H
        DEFB    031H
        DEFB    032H
        DEFB    033H
        DEFB    034H
        DEFB    035H
        DEFB    036H
        DEFB    037H
        DEFB    038H
        DEFB    039H
        DEFB    03AH
        DEFB    03BH
        DEFB    03CH
        DEFB    03DH
        DEFB    03EH
        DEFB    03FH
        DEFB    040H
        DEFB    041H
        DEFB    042H
        DEFB    043H
        DEFB    044H
```

```
DEFB 045H
DEFB 046H
DEFB 047H
DEFB 048H
DEFB 049H
DEFB 04AH
DEFB 04BH
DEFB 04CH
DEFB 04DH
DEFB 04EH
DEFB 04FH
DEFB 050H
DEFB 051H
DEFB 052H
DEFB 053H
DEFB 054H
DEFB 055H
DEFB 056H
DEFB 057H
DEFB 058H
DEFB 059H
DEFB 05AH
DEFB 05BH
DEFB 05CH
DEFB 05DH
DEFB 05EH
DEFB 05FH
DEFB 060H
DEFB 061H
DEFB 062H
DEFB 063H
DEFB 064H
DEFB 065H
DEFB 066H
DEFB 067H
DEFB 068H
DEFB 069H
DEFB 06AH
DEFB 06BH
DEFB 06CH
DEFB 06DH
DEFB 06EH
DEFB 06FH
DEFB 070H
DEFB 071H
DEFB 072H
DEFB 073H
DEFB 074H
DEFB 075H
DEFB 076H
DEFB 077H
DEFB 078H
DEFB 079H
DEFB 07AH
DEFB 07BH
DEFB 07CH
DEFB 07DH
DEFB 07EH
DEFB 017H ; DUP CNTRL W
DEFB 080H
DEFB 081H
DEFB 082H
DEFB 083H
DEFB 084H
DEFB 085H
DEFB 086H
DEFB 087H
DEFB 088H
DEFB 089H
```

```
DEFB 08AH
DEFB 08BH
DEFB 08CH
DEFB 08DH
DEFB 08EH
DEFB 08FH
DEFB 090H    ; STATUS LINE
DEFB 008H    ; ERASE LINE    CNTRL H, BACKSPACE
DEFB 092H    ;
DEFB 093H    ;
DEFB 012H    ; PREV PAGE     CNTRL R, PREV PAGE
DEFB 011H    ; CURR PAGE     CNTRL Q, HELP
DEFB 003H    ; NEXT PAGE     CNTRL C, NEXT PAGE
DEFB 00AH    ; NEW LINE      CNTRL J
DEFB 006H    ; INS CHAR      CNTRL F, INSERT MODE
DEFB 01BH    ; SET ATTR      CNTRL {, LINE ENTER MODE
DEFB 006H    ; INS LINE      CNTRL F, INSERT MODE
DEFB 09BH    ; FORM SELECT
DEFB 009H    ; TAB           CNTRL I, TAB
DEFB 007H    ; DEL CHAR      CNTRL G, DELETE CHARACTER
DEFB 00BH    ; UP ARROW      CNTRL K, UP ARROW
DEFB 019H    ; DEL LINE      CNTRL G, DELETE LINE
DEFB 0A0H    ; PRINT LOCAL
DEFB 008H    ; LEFT ARROW    CNTRL H, LEFT ARROW
DEFB 014H    ; HOME          CNTRL T, CURSOR TOP/BOTTOM
DEFB 00CH    ; RIGHT ARROW   CNTRL L, RIGHT ARROW
DEFB 003H    ; RESET         CNTRL C, NEXT PAGE/ WARM START CP/M
DEFB 00DH    ; ENTER         CNTRL M, CARRIAGE REUTRN
DEFB 004H    ; FWD TAB       CNTRL D, TAB TO NEXT WORD
DEFB 00AH    ; DOWN ARROW    CNTRL J, DOWN ARROW
DEFB 001H    ; BACKTAB       CNTRL A, TAB TO PREV WORD
DEFB 0A9H
DEFB 0AAH
DEFB 0ABH
DEFB 0ACH
DEFB 0ADH
DEFB 0AEH
DEFB 0AFH
DEFB 0B0H
DEFB 0B1H
DEFB 0B2H
DEFB 0B3H
DEFB 0B4H
DEFB 0B5H
DEFB 0B6H
DEFB 0B7H
DEFB 0B8H
DEFB 0B9H
DEFB 0BAH
DEFB 0BBH
DEFB 0BCH
DEFB 0BDH
DEFB 0BEH
DEFB 0BFH
DEFB 0C0H
DEFB 0C1H
DEFB 0C2H
DEFB 0C3H
DEFB 0C4H
DEFB 0C5H
DEFB 0C6H
DEFB 0C7H
DEFB 0C8H
DEFB 0C9H
DEFB 0CAH
DEFB 0CBH
DEFB 0CCH
DEFB 0CDH
DEFB 0CEH
```

```
DEFB 0CFH
DEFB 0D0H
DEFB 0D1H
DEFB 0D2H
DEFB 0D3H
DEFB 0D4H
DEFB 0D5H
DEFB 0D6H
DEFB 0D7H
DEFB 0D8H
DEFB 0D9H
DEFB 0DAH
DEFB 0DBH
DEFB 0DCH
DEFB 0DDH
DEFB 0DEH
DEFB 0DFH
DEFB 0E0H
DEFB 0E1H
DEFB 0E2H
DEFB 0E3H
DEFB 0E4H
DEFB 0E5H
DEFB 0E6H
DEFB 0E7H
DEFB 0E8H
DEFB 0E9H
DEFB 0EAH
DEFB 0EBH
DEFB 0ECH
DEFB 0EDH
DEFB 0EEH
```

```
DEFB 0EFH
DEFB 0F0H
DEFB 0F1H
DEFB 0F2H
DEFB 0F3H
DEFB 0F4H
DEFB 0F5H
DEFB 0F6H
DEFB 0F7H
DEFB 0F8H
DEFB 0F9H
DEFB 0FAH
DEFB 0FBH
DEFB 0FCH
DEFB 0FDH
DEFB 0FEH
DEFB 0FFH
END
```

end

Simple Keyboard Encoder

By Arne A. Henden

7415 Leahy Rd
New Carrollton, MD 20784

My Big Board is enclosed in an old terminal, occupying the space which had been home for the logic board. The old terminal used a simple matrix keyboard which was encoded (not ASCII) by the logic board. So I had to design an encoder for the keyboard.

The circuit uses a 50 KHz 555 oscillator to drive a 6-bit binary counter. The 3 high-order bits are used by the 74145 BCD-to-decimal decoder to select one of eight columns. The three low-order bits are used by a 74151 data selector which samples the eight key rows.

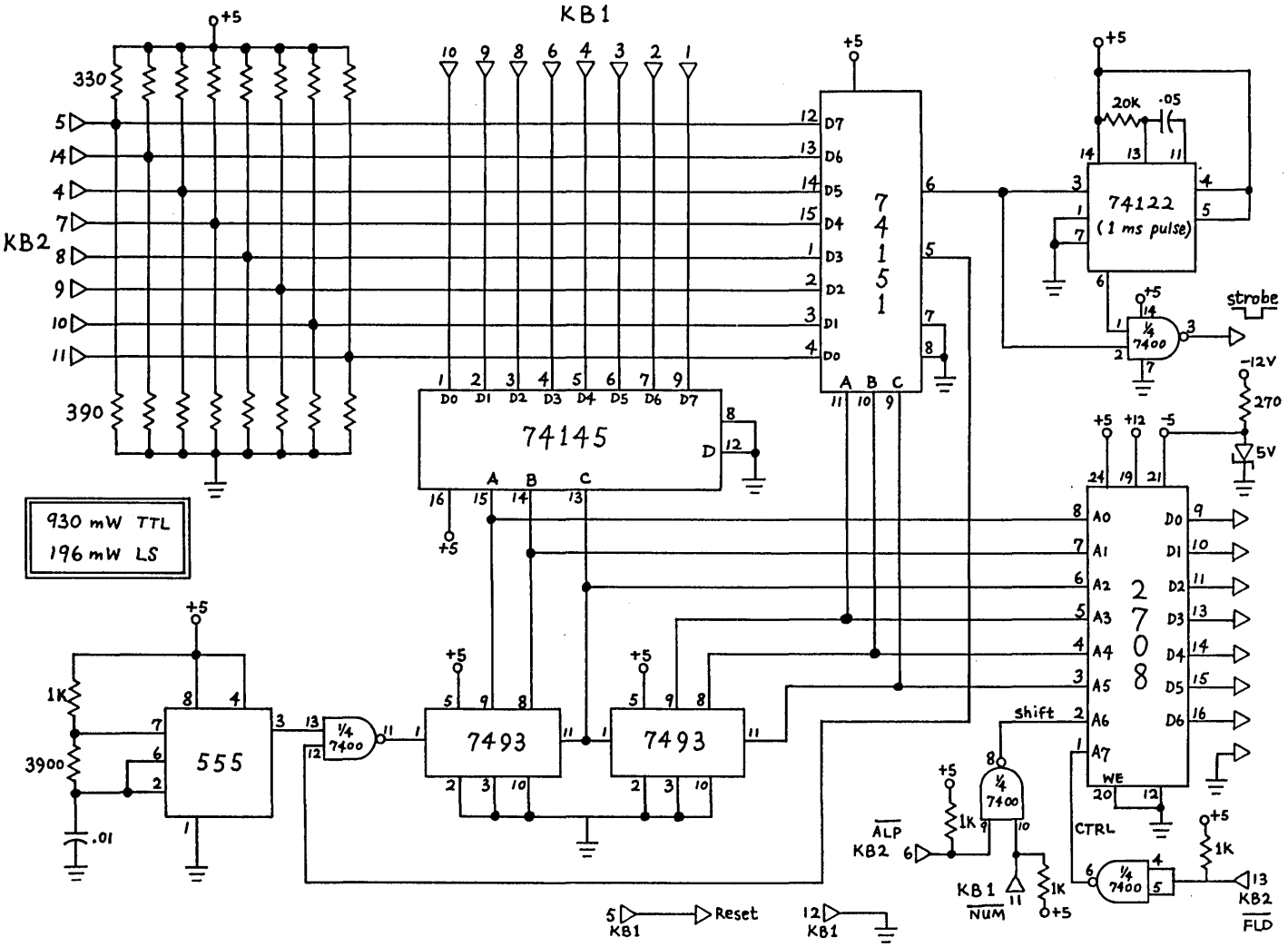
A key closure connects a row and column line, coupling the 74145 output to the 74151 input. When the circuit detects a key closure the 74151 turns off the oscillator and starts a 74122 monostable. The 74122 waits 1 ms (for debounce) and then generates a strobe pulse.

The 6-bit counter output indicates which of 64 keys has been pressed, but this isn't ASCII. So, this 6-bits is combined with the shift and control key status as an address for an EPROM.

You have a choice of EPROMs. A 1702 has sufficient storage for the entire ASCII code. I used a 2708, however, because it is easier to program and is available on the surplus market for around $3.00. A 2716 is overkill but it doesn't require the +12 and -5 V supplies. (It requires +5 V only.)

■ ■ ■

Keyboard Encoder Schematic



4 MHz (More and Less)

By Everyone

3.5 MHz (Easiest mod of all)

This modification is free. It requires no cutting on the board, and produces a symmetrical square wave clock.

1. Remove U96 and lift pin 9 so that when the chip is replaced, pin 9 will be out of the socket.
2. Jumper pin 11 of U96 to pin 8 of U11.
3. Replace U96, making sure that pin 9 does not go into the socket.

The new clock will be 3.4944 MHz. It is derived by dividing the 13.9776 MHz video clock by 4. This works just fine with 250 ns memory chips since the 3.5 MHz period is 286 ns.

Gary Hvizdak
1107 Country Club Ct #19
Bellevue, NE 68005

Inverted 4 MHz

An easy and reliable method of converting your Big Board system to run at 4 MHz requires the following steps.

1. Remove U96 from its socket.
2. Remove U97 from its socket.
3. Obtain a 74LS14 Schmitt trigger buffer.
4. Bend up pins 1, 5, 9, 11, and 13 on the 74LS14.
5. Solder a piece of wire from U97 pin 4 to U96 pin 3.
6. Install the 74LS290 back at U97 and the 74LS14 at U96.

This modification inverts and buffers the clock signal and changes the duty cycle so that the signal is low 60% of the time and high 40% of the time. This allows more time for memory access and so you can use of slower memory devices. My board was shipped with 300 ns TI4116-30 RAMs that worked fine at 4 MHz the first time the system was powered up.

Tom Dilger
5804B Nassau Drive
Austin, TX 78723

5 MHz

There are problems getting all boards to operate using the modification in issue #2 page 4. This is because the output is a non-symmetrical 4 MHz which, at best, is a 40%/60% square wave (100 ns high, 150 ns low). The spec on the Z80A calls for a minimum of 110 ns high and 110 ns low, leaving 30 ns for rise time. (110 ns + 110 ns + 30 ns = 250 ns.) After making this modification, I tried three Z80As in my system but they all worked erratically and were extremely temperature sensitive.

A Z80B solved all the problems and even allowed 5 MHz operation! Pin 5 on U96 produces a nice symmetrical 5 MHz clock (100 ns high, 100 ns low). Just replace the old 2.5 MHz pin 4 signal with the 5 MHz pin 5 signal and you have it.

Gary Oliaro
45 North Mill Rd
BOX 100A, RD #1
Cranbury, NJ 08512

Editor's note:

Each of the above modifications has its own merits. 3.5 MHz is the simplest and safest but you still should have the Z80A, SIOA, PIOA, and CTCA parts. You probably won't need a faster monitor ROM.

The 4 MHz mod does not address the requirements of the Z80A for 110 ns high and low, but I've heard of a number of systems which are violating this 110 ns requirement with no problems at all. (Japanese parts perhaps?)

If you are determined to go 4 MHz and do it by the book, use Otto Hiller's modification (issue #3 letters to the editor). It gets a true 50/50 clock by replacing the 20 MHz crystal with a 16 MHz crystal and then dividing by 4. The 16 MHz crystal shouldn't cost more than $4.00.

The 5 MHz mod is also very simple but the Z80B, SIOB, etc. parts are not all that cheap or easy to find.

And finally, Christopher Brock is working on a very interesting 6 MHz modification. (Seven MHz anyone?)

With any of these mods you need to move the CAS and MUXC lines on U76. CAS moves to U76 pin 4, and MUXC moves to U76 pin 3. ■ ■ ■

Designer's Corner

The following is a little function I find very useful.

There are some functions that I prefer to do in the PFM monitor such as reading a flaky disk or just poke around a CP/M disk, sector by sector. I wrote a one-liner that exits CP/M and branches to the PFM warm start address. When I return by rebooting, the data stays intact.

Program to enter the monitor from CP/M.

```
JP 0F003H
END
```

Not big, but useful for me.

To get back to CP/M gracefully you can do a 'G0' or a 'B'.

XM-80 (Extended Macro 80)

By Andrew Klossner

Knowlogy
PO Box 283
Wilsonville, OR 97070

Overview

Last issue Dave described the Unica software tool package. This month we'll look at XM-80, the language in which the Unica are written.

XM-80 is Z80 assembly language with additional syntax for modular programming. It is implemented by a translator, which produces straight Z80 assembly code and then calls Microsoft's MACRO-80 assembler to yield object code. XM-80 comes with a library of modules, called software components, which perform a variety of support services. Many of these services resemble those available under the Unix operating system.

Example XM-80 Program

Let's look at the program in figure 1, which reads decimal numbers and displays their hexadecimal equivalents. The call to TPUTF (terminal put, formatted; similar to C's 'printf') prints a question mark to prompt the operator for input. TVGET waits for the operator to type a line (it understands the CP/M line editing commands), and stores the line as a null-terminated string at 'buf:'.

ATOLI (ASCII to long integer) attempts to interpret the contents of 'buf:' as a decimal numeral. If successful, it returns a 32-bit long integer in registers BC and DE, sets HL to point to the delimiter (probably the null), and clears carry. Otherwise it sets carry to indicate an error. (For brevity, this example assumes that the operator will make no errors.) Then TPUTF is called to format a message containing the decimal

Figure 1: the DTOH program.

```
            uses LIB2800                    ; reference the XM-80 library
            uses LIB2801
dtoh:       TPUTF [stk="?"]                 ; prompt for a numeral
            TVGET [hl=buf]                  ; read a line from the console
            ATOLI [hl=buf]->[hl,bc,de]+C    ; long number in BCDE
            TPUTF [stk="%ld decimal is %lh hex^m^j",stk=bc,stk=de,stk=bc,stk=de]
            jr      dtoh                    ; loop forever; stop with ^C
buf:        ds      80                      ; input line buffer
            end dtoh
```

Undoing the Fatal Erase

By David Thompson

It was a dark night (not a speck of sunlight to be seen), my eyes had long before decided that focusing was a waste of effort, so I saved what I was working on, cleaned up the disk, and staggered downstairs to inspect the top side of my pillow.

Unfortunately, that night I had cleaned up more of the disk than I had intended.

I know that the only thing the erase command does is change the first character in a file's directory from 00 to E5. No big deal, half a night's effort still lay intact on the disk with only a piddling little E5 blocking my way. The problem is that CP/M thinks that an E5 is gospel.

However, the PFM monitor doesn't care one whit about E5s. So I got into the monitor, and used the read command to locate the sector on track 2 (track 2 contains the directory) that contained the name of the erased file.

For instance, to PFM, R0 2 1 means read disk 0, track 2, sector 1, which is the first sector of the directory on drive A. To read the second sector of the directory you'd enter R0 2 7 (it goes every 6 sectors in HEX numbers).

I found the name of the erased file with the E5 in front of it. The E5 was located at address 0080 in the disk buffer so I used the memory command to change it.

Enter the underlined characters.

And now, when you get back into CP/M, the file has magically reappeared. ■ ■ ■

```
R0 2 7 <CR>          ;Where I found the file name
M0080 <CR>           ;The E5 was at 0080
0080 E5 00           ;I entered the 00
0081 41 .            ;A non HEX character terminates
MF72F <CR>           ;Get the "READ" character in PFM
F72F 88 A8           ;Change it to "WRITE"
F730 XX .            ;Terminate
R0 2 7 <CR>          ;Now a read will write the
                     ;sector back onto the disk
<RESET> <CR>         ;Do a hardware reset or a boot
                     ;will write over track 0
B <CR>               ;Now you're back
```

and hex representations of the long number. The jump statement causes this behavior to loop until aborted by control-C.

Figure 2 displays the assembly code produced by the translator. When arguments are passed on the stack, a fair amount of overhead is incurred. When arguments are passed and returned in registers, there is very little overhead. You are free to decide, when creating new procedures, whether the application calls for the generality of stacked parameters or the speed of registers.

XM-80 Services

The translator checks every procedure call against its definition. In this example, the library procedure definitions are called by the USES statements. If, for example, you had passed the buffer address to TVGET in register IX instead of HL, the translator would have printed 'input needed in HL' and 'extra input (IX)', and would have skipped the assembly phase.

The services performed by the XM-80 library procedures include dynamic heap memory management, general device-independent, variable buffer size I/O, Unix-like pipes and I/O diversion, wildcard ('ambiguous file name') expansion, data sorting, character string manipulation, and random number generation.

Conclusions

There are three aspects of XM-80 which ease the task of writing assembly programs.

First, the procedure definition/invocation syntax makes subroutines much easier to use since it takes care of parameter passing and stack manipulation with a single statement.

Second, the translator catches many programming errors by flagging incorrect procedure calls.

Third, the XM-80 library provides solutions to a lot of the drudgery of writing assembly code, leaving you free to concentrate on the new aspects of the application. And, by writing new routines as callable modules, you can invent each wheel just once. As your library grows, the effort required to build a new program decreases, until eventually almost no new code is needed.

A place for XM-80

While machine oriented languages such as FORTH and C have their domains, there are some cases where you just have to grunt it out in assembler. These cases include time-critical applications, operating system modules, and situations where you don't want to pay royalties for using someone else's runtime library. XM-80 provides the special power of assembly language while adding that touch of convenience that makes the job easier.

Acknowledgements

XM-80 was designed by Hugh McLarty at Scientific Enterprises. Hugh wrote the translator using the BDS C compiler. The component library was a joint project of Scientific Enterprises and Knowlogy.

■ ■ ■

Figure 2: the assembly code resulting from the DTOH program.

```
        EXTRN   TPUTF   ;+3
        PUSH    HL      ;+3
        LD      HL,..0  ;+3
        EX      (SP),HL ;+3
        CALL    TPUTF   ;+3
        INC     SP      ;+3
        INC     SP      ;+3
        EXTRN   TVGET   ;+4
        LD      HL,BUF  ;+4
        CALL    TVGET   ;+4
        EXTRN   ATOLI   ;+5
        LD      HL,BUF  ;+5
        CALL    ATOLI   ;+5
        PUSH    DE      ;+6
        PUSH    BC      ;+6
        PUSH    DE      ;+6
        PUSH    BC      ;+6
        PUSH    HL      ;+6
        LD      HL,..1  ;+6
        EX      (SP),HL ;+6
        CALL    TPUTF   ;+6
        INC     SP      ;+6
        INC     SP      ;+6
        INC     SP      ;+6
        INC     SP      ;+6
        INC     SP      ;+6
        INC     SP      ;+6
        INC     SP      ;+6
        INC     SP      ;+6
        INC     SP      ;+6
        INC     SP      ;+6
        INC     SP      ;+6
        jr      dtoh    ;loop until ^C
buf:    ds      80      ;input line buffer
..1:    db      "%ld decimal is %lh hex^m^j",0
..0:    db      "?",0
        END     DTOH    ;+9
```

Figure 3: a sample run of the DTOH program.

```
?100000
100000 decimal is 186A0 hex
?1234567890
1234567890 decimal is 499602D2 hex
?-1
-1 decimal is FFFFFFFF hex
?^C
```

FORTHwords

Column by Arne A. Henden

7415 Leahy Road
New Carrollton, MD 20784

Welcome to the second FORTH column! This time, I will discuss the differences between 'standalone' and 'captured' FORTHs. Included separately are reviews of several versions of FORTH available for the Big Board.

Standalone vs. Captured FORTH

One of the biggest questions to be answered before a user can purchase a version of FORTH is: 'Should I buy a FORTH that works as a task under an operating system such as CP/M, or should I buy a standalone version?' There are advantages and disadvantages to both. I have used both versions and will now present my opinions on the matter.

FORTH was originally implemented as a standalone system. By this, I mean that standalone FORTH is self-booting from a disk and contains all terminal and disk I/O routines. FORTH was designed for this kind of use, since it contains an editor and assembler and does not need a file system for its disk block transfers.

Advantages of standalone FORTH:

1. You have more usable memory since you don't need CP/M.
2. You have more disk space since FORTH doesn't use a directory. Also, FORTH is a natural for folks with only a single drive.
3. FORTH executes faster since it doesn't need to call CP/M for I/O.
4. You don't need to buy CP/M.

Disadvantage of standalone FORTH:

You must customize standalone FORTH to do I/O on your system. For example, different video controllers and disk controllers require different I/O calls.

Customizing is very much like creating a new BIOS for CP/M. For bus-oriented computers, where the user can put many different kinds of video, disk drives and other peripherals on the bus, creating a standalone FORTH is not possible by the distributor without large expense. For single board computers, however, the customization only has to be performed once.

HART-FORTH and STOIC are examples of complete standalone systems. Timin FORTH is a hybrid standalone system. Because it uses the CP/M BIOS, Timin does not have to customize his FORTH for every installation. However, you must boot CP/M before booting FORTH.

Captured FORTH

FORTH executing as a task is standalone FORTH that has been 'captured' by an operating system. All usual functions are there: editor, assembler, and disk I/O. However, instead of including the I/O functions in the kernel, calls are made to the operating system routines that perform the necessary operations. In addition, FORTH can then access the BDOS and use files for program and data storage.

The disadvantages are, of course, the time and memory overhead of the operating system. The main advantage is that you can create a data file and access it with any language running under CP/M. Z80 FORTH is a captured FORTH. SL5 goes one step further and removes the block structure of disk I/O entirely.

Which version should you buy?

It's a matter of individual preference. At Indiana University we use standalone FORTH exclusively. We need the extra disk space to store our astronomical data and also find very rare need for CP/M (usually just when modifying the FORTH.ASM kernel file). We have disks with bootable systems that come up configured for the specific instrument that we are using at the observatory that night.

At Goddard Space Flight Center, we use a captured FORTH. The PDP-11/44 handles 16 users, of which about half are using FORTH. The rest use other languages, word processors and editors. By using captured FORTH in a multi-user environment, other users are not required to use FORTH and the system disk is available to all types of users. Both facilities are completely satisfied with their versions of FORTH.

HART-FORTH

I am going to provide a standalone FORTH that is configured specifically for the Big Board: HART-FORTH. I just was not totally happy with any of the reviewed FORTHs. While I think HART-FORTH is the only way to go, this is a definitely biased opinion! For this reason, HART-FORTH was not included in the first round of reviews and will be reviewed in the next issue by someone other than myself.

HART-FORTH will come in two versions. Version 1 will be a self-booting FORTH-79, customized for the Z80 and the Big Board. It will include both the double precision integer and assembler extensions, along with many extras such as arrays, screen editor and debugging. Its cost will be $50, and it will be available February 1, 1982 (contact me at the above address).

Version 2 will also include full IEEE-compatible 32-bit floating point with all transcendental functions. Its cost will be $75 and it will be available March 1, 1982. Owners of version 1 will be given complete credit when purchasing version 2.

As you will see, programmers have complete freedom when creating their own implementations of FORTH. The FIG model is often chosen as a starting point because of its ready availability. However, unique FORTHs such as STOIC and SL5 are available and make very interesting contrast to the FIG model.

My personal recommendation is to stay with a standard FORTH so you can exchange software easily. The choice between standalone and captured FORTH is up to you.

Miscellaneous

In the book reviews last time, I mentioned that *Starting FORTH* in paperback form was spiral-bound. This is not the case. I had access to a prepublication edition that was spiral bound.

(continued next page)

Stackwork's FORTH

Review by Arne Henden

Stackwork's FORTH (SL5) is one of the more unusual and interesting implementations of FORTH that I've seen. It matches FORTH-77 (with some minor differences) instead of FORTH-79. While double precision integer arithmetic operations are not included, many normally optional features such as arrays and case statements are available. The SL5 copy that I have is over a year old so this review does not include any recent changes.

File Access

As opposed to Z80 FORTH, which is tied to CP/M but treats the disk just like traditional FORTH (as 250 1024-byte blocks, randomly accessed); SL5 uses sequential access, record oriented files exclusively and has no block structure.

This means that SL5 does not reserve 1024-byte memory blocks to buffer data from the disk, leaving more memory space for application software. One input and one output file can be open simultaneously with the basic system, and more files can easily be added if necessary. SL5 can perform character or buffered read/write operations. Rather than LOAD one application screen at a time, it loads the entire application file.

FORTHwords continued

The next couple of columns will discuss benchmark results for the five FORTHs mentioned in this column, a very accurate hardware timer using CTC#0, and discuss several screens of utility software. Also, there'll be a review of HARTFORTH.

Meanwhile, please send me your questions and comments about FORTH. I would very much like to help you use this unique programmer's language to its fullest advantage!

■ ■ ■

What you get

The SL5 disk comes with one of the best manuals I've seen. Its major entries include a tutorial (6 pages), reference (22 pages), the Z80 assembler (12 pages) and a glossary (20 pages).

The disk includes the kernel (both .ASM and .COM files), the debug package (DEBUG.SL5), and the assembler (ASSEM.SL5). SL5 itself required no modifications and worked the first time. One of the major advantages of SL5 is that all of the source code is included, making system modification simple.

Disk I/O

SL5's non-FORTHian file structure makes SL5 appear quite different from other versions of FORTH. No editor is included because all files can be entered with any standard editor such as ED. This means that debugging changes are not interactive.

Because you cannot examine source files easily while in SL5 (no LIST commands), errors during loading are hard to trace down. You cannot incrementally load your program. More disk activity is necessary since source code is brought in one sector at a time and no extensive buffering is included. At the same time, the ability to manipulate source files with standard editors and other language compilers is an advantage. Leaving out the screen buffers provides more application programming space.

The compiler and assembler

Another advantage of SL5 is that a cross-compiler is built-in. You can easily produce ROMable code for dedicated applications. SL5 does this by separating code and data areas from the symbol table. You can create headerless code by simply removing the symbol table while cross-compiling. Variables and arrays can be stored separately from code segments, easing the problem of creating ROMable FORTH.

New system configurations can be of two types: user applications loaded on top of SL5, creating a new SL5.COM file; and creating entirely new systems using the cross compiler, perhaps modifying the kernel or burning a small application into ROM.

The assembler is excellent. Not only are Zilog mnemonics supported, but high level structures such as IF-ELSE-ENDIF and BEGIN-END are available. Stackworks is careful to point out system register usage to prevent loss of pointers by assembly code produced by the user.

Conclusion

My major complaint with SL5 is its file structure. When you lose the FORTH block/screen, you lose many of the advantages of FORTH. Interactive debugging is hampered, more disk I/O occurs, and less storage space is available on any given disk. SL5 is not easily upgradable to FORTH-79 because of the file structure.

Also, no double precision integer words are included, and floating point is not available as an option.

In all, I think SL5 is a good language that is well documented and comes with a lot of usually optional features. Its unique implementation of FORTH is interesting, but at the same time prevents me from recommending it to a user who wants to learn standard FORTH.

■ ■ ■

Name:	SL5
Authors:	The Stackworks Mike Brothers Larry Mongin Dave DeLauter
Type:	FORTH for the Z80
Distributor:	Supersoft Associates P. O. Box 1628 Champaign, IL 61820
Price:	$175
Requires:	CP/M operating system Z80 processor 24K bytes RAM
Manual:	110 page 8 1/2 x 11" spiral bound

Timin FORTH

Review by Arne Henden

Mitchell Timin has modified FIG FORTH to include a screen editor, fast disk I/O, system configuration and array handling. The system is in 8080 machine code, but a Z80/8080 assembler is provided for programming in the Z80 superset. Timin uses CP/M to boot FORTH and to provide BIOS calls, but all disk storage is in a special format and not compatible with CP/M.

The Manual

The 66-page user's manual covers Timin's additions nicely. Plus, it provides information on how to install the system.

About 20 pages are devoted to a tutorial, 11 pages to the assembler, and 22 pages to the glossary. Documentation for the screen editor, CP/M utility, and floating point are on disk. Included on the disk are the FORTH.COM file and 20+ screens of FORTH, mostly the editor and examples.

Installation

Timin FORTH is harder to install than Z-80 FORTH. It isn't a standalone system, but it doesn't operate under CP/M either. Usually you put the FORTH.COM file on a CP/M disk, boot FORTH, and then replace the boot disk with a FORTH-format disk. You can also use a different drive for the FORTH-format disk, or write FORTH screens onto the CP/M disk, but I highly recommend against it. You would have two operating systems sharing a common disk and it just doesn't work. For example, the message screens 4 and 5 are on track 1, or in the middle of the CP/M system.

As mentioned above, Timin FORTH is a hybrid standalone system. One problem with this kind of system is that you must swap disks after loading FORTH. This is a seemingly minor inconvenience, but FORTH tends to crash often. For example, you can attempt to store a value in a variable and get the stack order wrong, thereby storing non-executable garbage somewhere in the kernel. Then you must reload the system, which with Timin FORTH means swapping disks.

If you have a two-disk drive system, you can leave the CP/M disk in drive A and execute your applications software from drive B.

Some Details: Timin's Z80 assembler is not Zilog-compatible. It uses 8080 mnemonics except for the Z80 additions. Timin advertises fast disk I/O, and achieves it at the expense of about 10 percent of disk storage. Only 24 sectors are used on each track, with interleaving every third sector.

While this format allows a full FORTH screen read in one disk revolution, it is not optimized for sequential screen reads (at least not at 2.5 MHz). I found that reading 100 sequential screens took about the same length of time for Timin and Z80 FORTH, probably because the sector timing mark. Also, the Timin drive-switching words AD and BD do not flush updated screens before switching, a potentially dangerous omission.

Floating Point

Options for Timin FORTH include software floating point and the CP/M utility. The floating point uses hexadecimal exponents, providing a range of 10**76 at the expense of up to 1 significant digit in the mantissa. It provides the usual arithmetic and comparison functions. I don't like the way Timin implements the floating point. He uses a mode control flag to switch between integer and floating point modes.

While in floating point, all operators (e.g. '*', '0=') are redefined. All numbers, whether entered as '3', '3.00', or '3.00E0', are considered floating point. This is fine until you realize that DO-LOOP parameters are integers and there are many, many cases of mixing modes within a FORTH definition. It would have been much better to have just created new words called F*, F0=, etc.

Timin's CP/M utility handles transfer of CP/M files to and from FORTH blocks. If the FORTH screens are text, the utility inserts the CR/LF at the end of each line during the transfer. Other functions include saving and retrieving parts of memory, CP/M BDOS calls, and LOADing from a CP/M text file. This utility only transfers whole files. It does not allow you to build the file a record at a time with read/writes. It occupies 19 blocks on the basic disk.

The Editor

The screen editor was easily modified to run on the Big Board. The 'CLR' function is the only command that needs to be changed. Similar cursor commands are available as on the Z-80 FORTH full screen editor. I like the Z-80 FORTH editor better, since it lets you switch blocks without having to drop out of the edit mode. It also has a nifty command menu. However, the Timin editor is resident rather than a separate program as is the Z80 FORTH editor.

Conclusion

Timin has a good FORTH, but the $235 basic price is a little steep. He has offered version 3 without the CP/M utility to Big Board users for $75, a more competitive price. Check with Timin before ordering to be sure the offer still stands.

■ ■ ■

Name:	Timin FORTH
Author:	Mitchell E. Timin
Type:	8080 FIG-FORTH
Distributor:	Timin Engineering Co. 9575 Genesee Avenue San Diego, CA 92121
Price:	$235
FP	$100 extra
CP/M utils	$50 extra
Util disk	$75 extra
Requires:	CP/M 8080 or Z80 24K bytes RAM
Manual:	66 page 7 x 8 1/2" stapled booklet

Z80 FORTH

Review by Arne Henden

Ray Duncan created Z80 FORTH by taking FIG FORTH and extensively modifying it to take advantage of the additional Z80 registers and instructions. He claims an 8-12 percent speed advantage over the original 8080 FIG implementation. His system operates completely under CP/M, with FORTH as a .COM file and the screen storage using CP/M 2.2 random access disk files. Because of this, FORTH applications and data software can co-exist with CP/M files.

What you get

The FORTH disk comes nicely packaged with a 90-page user's manual in a soft plastic binder. The documentation is very professional looking. It contains all of the information necessary to install FORTH in your system, and information on Duncan's extensions to the basic model. And though it includes a complete Z80 FORTH glossary, there is little information on how to use FORTH.

The Z80 FORTH disk contains five utilities to provide access to the screen file without entering FORTH. In addition, Duncan has included two general purpose CP/M utility programs. FCOPY copies one file from one disk to another (with verification), and FVFY, which performs only the verification operation of FCOPY.

The FORTH screen file provided is remarkably full, containing over 140 screens. Some of the major entries include: two editors, FORTH extensions such as arrays, games (including Breakforth and Life), 8080 and Z80 assemblers, a CP/M interface utility, time and date for the Dual Clock-24 board, and debugging options.

Name:	Z80 FORTH
Author:	Ray Duncan
Type:	Z80 optimized FIG-FORTH
Distributor:	Laboratory Microsystems
	4147 Beethoven Street
	Los Angeles, CA 90066
Price:	$50
FP	$100 extra
Cross-comp.	$150 extra
Requires:	CP/M 2.2
	Z80
	24K bytes RAM
Manual:	95 pages 8 1/2 x 11"
	Looseleaf binder

Running it

I had absolutely no problems bringing up Z80 FORTH. There are only a few minor complaints. Full CP/M control character input is not implemented. That is, you can backspace, but have no CNTL/R, CNTL/U functions. This can be very frustrating when trying to delete lines.

There is a discrepancy between the concepts of 'block' and 'screen.' A Z80 FORTH 'block' is 128 bytes long, while all 'screens' are 1024 bytes long. FIG-FORTH makes no size restrictions on 'blocks,' but FORTH-79 requires blocks and screens to both be 1024 bytes long. I suggest redefining Duncan's BLOCK, UPDATE, FLUSH, etc. words to operate on 1024-byte buffers. Because of the CP/M structure, the screen file is limited to about 200 screens plus the directory and FORTH.COM.

Editors

Three editors are included. A FIG-FORTH portable line editor allows configuration of the two cursor-controlled editors: the mini-screen editor and the full-screen editor. All three editors come in high-level FORTH so that you can easily modify them.

The full-screen editor is only available as a separate entity. It allows cursor positioning, character insertion and deletion, and allows moving from one screen to another with a single keystroke. It is very useful when entering several screens of a new application.

The mini-screen is a subset of the full-screen editor, and is available while you are in the FORTH program. Duncan provides screens for configuring the editors for various terminals. I supplied him with the proper screen for the Big Board/ADM-3 so there should be no problems bringing up the full-screen editor.

There is one potential problem with the screen editors: CNTL/C is intercepted by the operating system instead of the editor. This means that if you inadvertently hit CNTL/C when meaning to hit CNTL/V, you can lose all edited screens that have not been flushed to disk!

Disk I/O

A full CP/M interface utility is included, providing file create/delete, open/close, and read/write functions for sequential and random-access files. Access to the BDOS is provided with the word 'FDOS.' However, Duncan's documentation is unclear about how to read and write from an arbitrary buffer address. The Z80 assembler is complete and uses Zilog mnemonics, but I have strong reservations about it.

Duncan uses a mixture of postfix and infix notation which can be very confusing. The software helps by doing a lot of error checking but that slows down the compilation process.

Floating Point

The floating point option includes the four basic functions, comparisons, float/fix, and square root. Duncan plans on providing transcendental functions at no additional cost in the near future. While his floating point implementation works, all floating point numbers must be in the form x.xxx E xx, (the exponents and spaces are required).

Conclusion

Overall, I am very impressed with Z80 FORTH. I think Ray Duncan has done an excellent job of providing a quality system with plenty of application software at a reasonable cost. If you want a FORTH that executes under an operating system such as CP/M, I don't think you can beat Z80 FORTH.

■ ■ ■

Bringing Up a Reluctant Big Board

By Charles VanLeeuwen

3335 SW 116
Beaverton, OR 97005

When I ordered it last April, the Big Board from Digital Research Computers (DRC) seemed like the easiest and cheapest way to get a capable micro system. Because of the way I went about it, I turned out to be wrong on both counts. It is now October and I've spent about twice what I had intended for parts.

I bought the bare board since I already had quite a few parts and thought I knew how to get a good deal on the rest. Most were easy to get, and since I did some comparison shopping, they were reasonably priced. A few low power Shottky (74LSXX) parts were hard to find, and the 8116 baud rate generator was impossible. I finally ordered it from DRC.

Construction of the board went fairly well. I had nearly all the sockets on hand, so my board is fully socketed (more on that later). I'm an old hand at soldering small stuff so, with the right equipment, that wasn't a problem. The only comment I have is that there are a lot of sockets to insert and hold in place while soldering. After that, there are a lot of IC pins to try to get into the proper sockets.

I was a little disappointed, but not surprised, when I powered it up with my video monitor and nothing happened. I didn't even have a raster on my monitor because the Big Board was not generating a horizontal output signal. The culprit turned out to be a resistor I had left out of the monitor circuit. At this point I found I had no verticle output. This problem was traced to a bad TTL IC (my scope displayed the IC's poor switching characteristics as I traced back up the video timing circuit). Once I had the video generator problems cleaned up, my screen was filled with lines of

B@B@B@B@B@B@B@B.

A couple of observations made me believe that things were fairly close to functional. The display blinked about once a second as the video memory was being rewritten. If I powered on and off very quickly, the display would show a semi-random set of characters. A slower turn-on after power-off (but quick enough so video was still available) showed a randomly filled display being rewritten with the alternating characters. The reset button would sometimes stop the rewriting and sometimes was able to restart it.

The next hundred hours spent trying to get locate the source of this problem have become a nightmarish blur in my mind. My late-night efforts were endured by my wife, but my technical ego kept dragging lower and lower. Over the weeks, I got volunteer help from technically experienced friends. Together we found some bad TTL and some errors I had started making in the search. Rather than a blow-by-blow description, the following are specific things I found and a list of some of the blind alleys I encountered.

My tools included a tried and true VOM (volt-ohm-meter), a digital voltmeter, a 50 MHz dual trace scope, a 10 MHz dual trace Heathkit scope, and a byte-wide, 16-words-deep Paratronics logic analyzer. The most useful tool was the Heathkit scope because of its relatively good triggering. I spent a lot of time using the logic analyzer, but I don't recall finding a single problem with it, it just was not wide or deep enough to give meaningful information. You really need to see the data and the addresses at the same time. The observation of invalid signal levels, floating nodes, and poor rise/fall times on a scope pinpointed all the problems.

I found at least four bad ICs. Two were TTL and two were MOS devices. The gate I found quickly was unable to pull down a load. The second piece was an address driver for the dynamic RAM (it had a very poor rise time). One of the 2114s had a bad output, but that may have been due to handling while debugging. I had the 2114s out at least six times. The fourth problem IC was a PIO which I had plugged into the SIO socket sometime after midnight one frustrating evening. The 12 V applied to the SIO socket didn't set well with the PIO. It began generating noise on the bus and writing semi-random characters to the video RAM.

After finding the first three bad ICs, the display changed to alternating 'D' and ':'. The ASCII code for 'D' and ':' don't have the same bit overlap as 'B' and '@'. At this point, I really got sidetracked as I tried to figure out what made the display show this pattern. I originally thought I knew how it worked but today I'm still confused. I found I got the same display even though I grounded lines that had to go high to write a 'D'.

I used the logic analyzer (LA) to confirm that at least the first 16 words of the instruction set data path were reasonable on power-up. The system initializes and then goes into the block move from EPROM to high RAM. I also used the LA to verify that bus lines were toggling and not tied together.

My inability to see where or when things were falling apart made me look for a way to single-step the processor. However, RAM refresh depends on the processor so it appears impossible to do without a full-fledged in-circuit-emulator with real-time refresh.

Victory finally came as I was preparing to give up. I had talked to Tanner and he suggested that I send the board back to him. His estimated price for repair was certainly reasonable but it was really hard to accept the fact that I could not find the problem. I finally made the commitment to myself (and voiced to several others to keep myself honest) that I would work on it one more Saturday and send it back to Tanner if I weren't successful.

After a full day of non-progress, I took one final desperate pass. Starting at the bottom left corner of the board, I went pin by pin with the scope looking for anything suspicious. I was down to the last eight ICs without finding anything un-

On Modems, SIOs, and Lync

By David Thompson

usual (except for the numerous inputs which are floating since they are not used) when I landed on the ground pin of U8. It was not ground, but rather a strange noisy signal which floated up and down very slowly.

Careful inspection, after prying the plastic top off the socket, showed that the two springs which are supposed to grip the side of the IC's pin were pointing straight down and not making contact. (Earlier we had had this and numerous other sockets off while searching for the problem.) My needle nose pliers quickly fixed the contact and to my utter amazement when I powered up, I got:

SYSTEM MONITOR 3.3

After celebrating with a good night's sleep and a pumpkin pie with candles freshly baked by my wife, I quickly debugged my keyboard's inclination to produce the same character from several different keys. (The ribbon cable was shorted by a commercially attached connector.)

The same socket may have been the problem when I blew out two other RAM chips while swapping RAM chips around while trying the RAM test.

In summary, the Big Board is really for those with experience. Simple problems can be very difficult to trace down without a lot of experience and truly sophisticated equipment (which DRC has and will use for a very reasonable fee). Sockets made it possible for me to change components without messing up the board but they were the root of my most serious problem. Sockets which grip the edge rather than the side of the pin are best.

An alternating set of characters on the video most likely means a processor RAM problem. Nothing can beat a decent scope for looking at signals on all pins.

I want to express my sincere appreciation to Vance Navarrette, Gregg Will, Dave Thompson, and my wife, Leona, for their support and assistance. ■ ■ ■

I purchased a brand new Vadic 3451 modem a few months ago. I got it because it is supposed to talk to just about any other modem alive. It speaks Bell 103 (300 baud full duplex), Bell 212A (1200 baud full duplex) and Vadic 3400 (also 1200 baud full duplex). For those of you who have had bad luck trying to transfer data at 300 baud, I suppose 1200 baud over a standard phone line sounds like folly, but it's actually much more immune to garbage on the line than 300 baud.

Anyway, last spring I started out with the modem manual, the Z80 SIO Product Specification, and the CPM User's Group Modem 7 program. A week later I decided that I had better get back to more important things like getting a magazine out. I didn't get Modem 7 running because I hadn't successfully initialized the SIO.

A couple of months ago I received a copy of Lync from Computer-Aid; 1122 De La Vina, Santa Barbara, Ca, 93101. Lync does a lot of very nice things like let you remotely control another computer or let another machine remotely control yours (when both are running Lync). Plus it handles data verification, storage on disk and other niceties. It, of course, also talks to un-Lync-ed systems and can verify that kind of exchange by having the data sent twice.

Somehow, the folks at Computer-Aid managed to put the entire Lync manual on two sides of a single 8-1/2 X 14 sheet of paper (including the configuration instructions). They mentioned that they were working on better documentation. I'm sure it will be an improvement but I can't complain too much—they explain the system as well on one sheet as some folks manage in a bound manual. In fact, they send out the 'manual' to folks who request information about the system. So if you are interested in a new modem program, let them know. Lync costs $95 and is certainly worth investigating if Modem 7 isn't doing it for you.

Meanwhile, I couldn't get Lync running because of the (#%'&%) SIO. Fortunately I got some help this time. Bill Randle, a long time friend and fellow Big Board owner, came to my aid and I now have Lync running. (See the serial port A configuration below.)

I've actually talked to the local CBBS (nothing intelligible, you understand, but you have to consider the source). But even more important is that now I have the ability to send and receive articles and software by phone. (Anyone have any suggestions on one of the commercial computer networks?)

■ ■ ■

```
Serial port A configuration for Lync

     data port = Q4
stat/cont port = 06
   recvr ready = 01
      xmit rdy = 04
        drives = 02
     backspace = 08
  powerup mode = 01
term mode exit = 01
view cont char = 00
         clock = 02
     init mode = 03
```

```
usart init bytes
: 00 05 06 18 06 01 06 00 06 04 06 44 06 03 06 C1 06 05 06 EA 04 FF
   |     |                                                  |  |
   |     |                                                  |  |
   |     |this area initializes the SIO .....................|  |
   |     |(the 06's are the address of port A control)          |
   |                                                            |
   |00 05 = set baud rate generator (00) to 300 baud (05)       |
   |        (00 07 would set rate to 1200 baud)                 |
   |                                                            |
   |                              stuff first character (FF) into SIO (at 04)
```

(Editorial continued)

Thanks

I don't get flustered often but I definitely did when Jarel Hambenne called one afternoon and asked to place a simple ad. He wanted an ad in Issue #3 saying, 'Best Wishes Micro C'. Well I don't know if I said 'thank you' then or not, but I am definitely saying it now. Thank you, Jarel.

Coming Up

We've got a couple of very interesting issues coming up (you've been bored so far, right?). Issue #5 will concentrate on word processing.

If you are using a text editor that you particularly like or dislike just jot the name of it on a post card along with a few comments about it and we'll run some of the comments in the letters column.

Shortly thereafter, (either issue #6 or #7) we will do a block buster C review. We will dissect just about every C on the market that runs under CP/M: features, speed, code size, bugs, and so forth. Anyone wanting to get involved in that issue, please get in touch with me right away. (I will be selecting a core group of evaluators shortly.)

Two more versions of C have crossed my desk: Q-C and C/80. They are both very interesting extensions of Ron Cain's Small-C, although neither supports structures, unions, or floating point. (At least not yet.)

Q-C includes source code and an excellent manual. In fact, if you ever wanted to extend a compiler, or just wanted to find out how one works, this software and 88-page manual is exactly what you need. Plus, an understanding of the compiler really helps you master the language.

Jim Colvin, the author of both the manual and the software, has a knack for organization and comfortable informality that will have you reading this manual for the pure enjoyment. After the first dozen pages, I definitely wanted to meet him. (For more information see the Code Works Ad in this issue.)

C/80 is the other interesting version of C. It's claim to fame is price ($39.95 including utilities and an assembler). It also compiles a substantial portion of the C language, and does it very well. For instance, it supports initialization and conditional compilation, features which are not usually available in the small versions of C and not yet available in Q-C.

C/80 does not include source. The 25 page manual is very complete, although it is not in the same league as the Q-C manual (but then, no other manual has been either). C/80 is available from The Software Toolworks, 14478 Glorietta Dr., Sherman Oaks, CA, 91423.

When do you expire?

If you are curious about when your subscription will expire, just check the last number on the top line of your mailing label. That number is the issue number of the last issue you will receive on your present subscription. When your time is at hand you might consider renewing (consider the alternative).

David Thompson
Editor and Publisher

ANNOUNCING

DOUBLE DENSITY DISK INTERFACE FOR THE BIG BOARD

New floppy interface package for the Big Board lets you read and write single and double density disks with 128, 256, 512, and 1024 Bytes/sector.

The package includes:

1. Fully assembled and tested board, cable and connector to replace the 1771. Board contains 5 ICs including a Western Digital 1795.

2. An extended monitor in two 2716s.

3. A disk containing:
   - Disk formatting program. (128, 256,512,1024 bytes/sector)
   - Disk copying program.
   - An overlay for MOVCPM.COM
   - A double density SYSGEN

4. Documentation
   - Dependable 4MHz mod
   - Jumpers to add to the Big Board.

Sector size is determined by how the disk was formatted and is totally transparent to the user.

Disk capacity ranges from: 241K for SS, SD, 128 bytes/sector to 668K for SS, DD, 1024 bytes/sector

Requires minor modification to Big Board and requires that Big Board run 4 MHz.

Available December 15, 1981.

Price: $200

Otto Hiller Co.
Scientific Equipment
P.O. Box 1294
Madison, WI 53701
608-271-4747 3-5 pm

Bring the flavor of Unix To your Z80-based CP/M system with Unica

"Unicum: a thing unique in its kind, especially an example of writing. Unica: the plural of unicum."

The Unica: a unique collection of programs supporting many features of the Unix operating system never before available under CP/M. The Unica are more than software tools; they are finely crafted instruments of surgical quality. Some of the Unica are:

bc	-	binary file compare
cat	-	catenate files
cp	-	copy one or more files
dm	-	disk map and statistics
hc	-	horizontal file catenation
ln	-	create file links (aliases)
ls	-	directory lister
mv	-	move (rename) files, even across users
rm	-	remove files
sc	-	source file compare, with resynchronization
srt	-	in-memory file sorter
sr	-	search multiple files for a pattern
sp	-	spelling error detector, with 20,000 word dictionary

Each Unicum understands several flags ("options" or"switches") which control program alternatives. No special"shell" is needed; Unica commands are typed to the standard CP/M command interpreter. The Unica package supports several Unix-like facilities, like filename user numbers:

sc data.bas;2 data.bas;3

(compares files belonging to user 2 and user 3);
Wildcard patterns:

rm *tmp* -v

(types each filename containing the letters TMP and asks whether to delete the file);
I/O redirection:

ls -a ►list

(writes a directory listing of all files to file"list");
Pipes:

cat chap* ! sp ! srt ►lst:

(concatenates each file whose name starts with"chap", makes a list of mispelled words, sorts the list, and prints it on the listing device).

The Unica are written in XM-80, a low level language which combines rigorously checked procedure definition and invocation with the versatility of Z80 assembly language. XM-80 includes a language translator which turns XM-80 programs into source code for MACRO-80, the industry standard assembler from Microsoft. It also includes a MACRO-80 object library with over forty "software components", subroutine packages which are called to perform services such as piping, wildcard matching, output formatting, and device-independent I/O with buffers of any size from 1 to 64k bytes.

The source code for each Unicum main program (but not for the software component library) is provided. With the Unica and XM-80, you can customize each utility to your installation, and write your own applications quickly and efficiently. Programs which you write using XM-80 components are not subject to any licensing fee.

Extensive documentation includes tutorials, reference manuals, individual spec sheets for each component, and thorough descriptions of each Unicum.

Update policy: each Unica owner is informed when new Unica or components become available. At any time, and as often as you like, you can return the distribution disk with a $10 handling fee and get the current versions of the Unica and XM-80, with documentation for all new or changed software.

The Unica and XM-80 (which requires MACRO-80) are priced at $195, or $25 for the documentation. The Unica alone are supplied as *.COM executable files and are priced at $95 for the set, or $15 for the documentation. Software is distributed on 8" floppy disks for Z80 CP/M version 2 systems.

Knowlogy

"Shaping Knowledge for Evolving Worlds"
P.O. Box 283
Wilsonville, Oregon 97070

Visa/Mastercard customers call (503) 635-5701 after hours for next day shipment.

MICRO CORNUCOPIA
Journal of the Big Board Users Group

11740 NW WEST ROAD
PORTLAND, OREGON 97229