the CP/M* and S-100 user's journal

$2.00
U.S.A.

MICROSYSTEMS™

MAR/APR 1981

VOL. 2/NO. 2

THREE 'NEW GENERATION' Z-80 CPUs COMPARED

See Pages 22-24

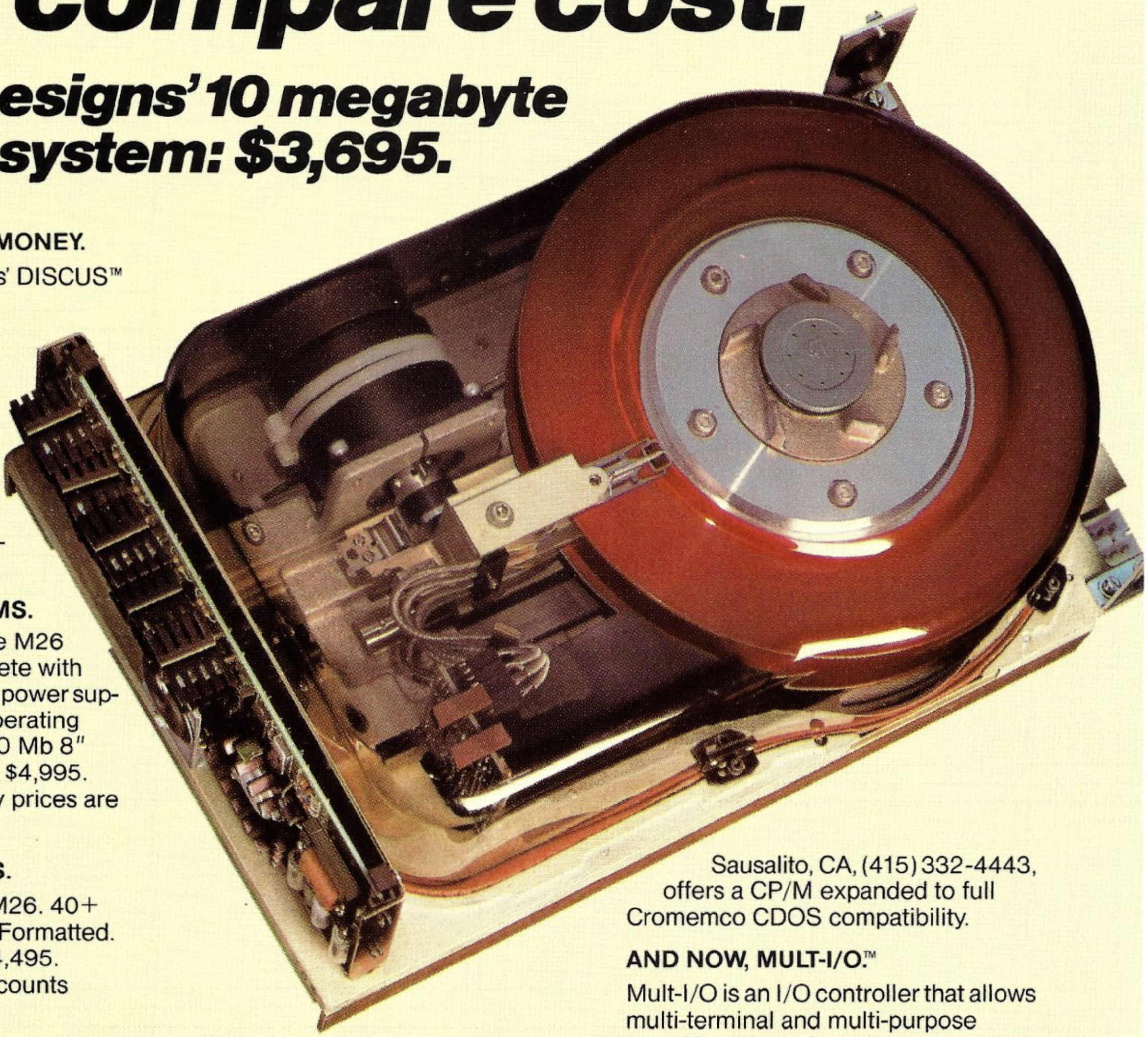
Also in this Issue

and more

Complete Table of Contents on Page 3

First compare quality. Then compare cost.

Morrow Designs' 10 megabyte hard disk system: $3,695.



MORE MEMORY. LESS MONEY.

Compare Morrow Designs' DISCUS™ M26™ hard disk systems to any system available for S-100 or Cromemco machines. First, compare features. Then, compare cost per megabyte. The M26 works out to under $200 a megabyte. And the M10 is about half the cost of competing systems.

COMPLETE SUBSYSTEMS.

Both the M10 (8″), and the M26 (14″), are delivered complete with disk controller, cables, fan, power supply, cabinet and CP/M® operating system. It's your choice: 10 Mb 8″ at $3,695 or 26 Mb 14″ at $4,995. That's single unit. Quantity prices are available.

BUILD TO FOUR DRIVES.

104 Megabytes with the M26. 40+ megabytes with the M10. Formatted. Additional drives: M26: $4,495. M10: $3,195. Quantity discounts available.

S-100, CROMEMCO AND NORTH STAR*

The M26 and M10 are sealed-media hard disk drives. Both S-100 controllers incorporate intelligence to supervise all data transfers through four I/O ports (command, 2 status and data). Transfers between drives and controllers are transparent to the CPU. The controller can also generate interrupts at the completion of each command ...materially increasing system throughput. Sectors are individually write-protectable for multi-use environments. North Star or Cromemco? Call Micro Mike's, Amarillo, TX, (806) 372-3633 for the software package that allows the M26 and M10 to run on North Star DOS. MICAH of Sausalito, CA, (415) 332-4443, offers a CP/M expanded to full Cromemco CDOS compatibility.

AND NOW, MULT-I/O.™

Mult-I/O is an I/O controller that allows multi-terminal and multi-purpose use of S-100 and Cromemco computers. Three serial and two parallel output ports. Real time clock. Fully programmable interrupt controller. Designed with daisy-wheel printers in mind. Price: $299 (kit), $349 assembled and tested.

MAKE HARD COMPARISONS.

You'll find that Morrow Designs' hard disk systems offer the best price/performance ratios available for S-100, Cromemco and North Star computers. See the M26 and M10 hard disk subsystems at your computer dealer. Or, write Morrow Designs. Need information fast? Call us at (415) 524-2101.

Morrow Designs' 26 megabyte hard disk system: $4,995.

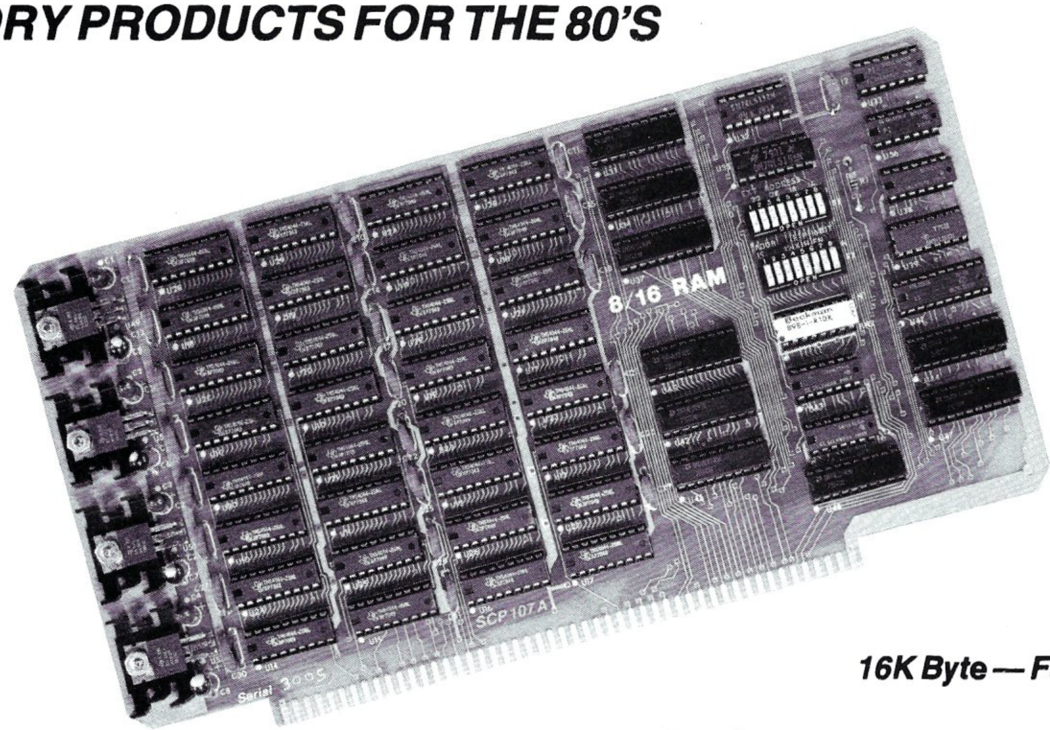
Look to Morrow for answers.

MORROW DESIGNS

5221 Central Avenue
Richmond, CA 94804

*CP/M is a trademark of Digital Research Corp.
*Cromemco is a trademark of Cromemco, Inc.
*North Star is a trademark of North Star Computers, Inc.

MEMORY PRODUCTS FOR THE 80'S



16K Byte — Fully Static

8/16 RAM

Designed with the Future in Mind...

- Able to use the full 24-bit address bus of the IEEE S-100 Standard for a 16-megabyte address range.
- Fully static design eliminates system timing problems. Promotes reliable operation with a wider range of CPU cards and DMA devices.
- Easy to integrate into your system. Addressable on 4K boundaries. PHANTOM, extended addressing, 16-bit operation may be switched off if desired.
- Can act as either an 8-bit or 16-bit wide memory. Dynamic bus switching per the IEEE Standard.
- Fast 200 nanosecond memory chips help you keep up with the ever-rising clock speeds of newer CPUs.
- The 8/16 is the only memory board made which is designed to run at full speed with our 16-bit 8Mhz. 8086 CPU card. And, it has plenty of speed to spare.

From Seattle Computer, the Static Memory Experts

Why static memory? First, compatibility. Most S-100 products do not meet the IEEE Standard. Because of their critical timing requirements, dynamic memory boards must often be configured for a particular CPU — a situation that could make future upgrading difficult. Static memories are inherently more versatile.

Second, speed. The access times of static and dynamic memories are not comparable numbers. Static memories get a substantial head start while dynamic memories wait for a "clock." The IEEE S-100 Standard **guarantees** this head start is **at least** 70 nanoseconds. The result: most of today's dynamic memory boards will not run at full speed with tomorrow's (or even today's) faster CPUs. The 8/16 will.

The 8/16 memory card is fully assembled, tested, guaranteed one full year. Suggested retail price: 1-4, $395; 5-9, $345. Manual only — $3. Overseas purchasers add $15 per board for air shipment.

Seattle Computer Products, Inc.
1114 Industry Drive, Seattle, WA. 98188
(206) 575-1830

16 BIT 8086 MICRO SYSTEM
Tec-86 CP/M™-86

IEEE S-100 Bus — Available Now
Complete system up and operating
8086 System with Dual Floppy Disks
Available Now for only $3990
CP/M-86™ only $250

CP/M-86
TEC - 86
BREAK-THRU

Complete 8086 System for CP/M-86™
CP/M-86 Available Now - $250 • IEEE 696 S-100 Compatible • 5 MHz 8086 CPU (Optional 4 or 8 MHz) • Vectored Interrupts • 24 Parallel I/O lines (three 8 bit ports) • Two RS-232 Ports • Baud rates from 50 to 19200 baud • Independent baud rate generator for each serial port • 32 Kilobytes of static RAM (Expandable) • ROM boot for CP/M-86™ • Dual 8-inch Shugart floppy disks with controller • Attractive, all metal desk top enclosure

Tec-86 / 8086 System
$3990

CP/M-86™ registered trademark of Digital Research

TECMAR INC.

Data Acquisition Systems and
Video Digitization Systems Available
23414 Greenlawn • Cleveland, OH 44122

TECMAR, INC.
(216) 382-7599

the CP/M* and S-100 user's journal

MICROSYSTEMS™

Volume 2 Number 2 March/April 1981

Editorial correspondence is welcomed and should be sent to: Sol Libes, c/o MICROSYSTEMS, Box 1192, Mountainside, NJ 07092. Phone: (201) 522-9347.

Subscriptions are $10 a year (6 issues USA). To subscribe, call our toll free number: (800) 631-8112, or (201) 540-0445 in New Jersey.

For information on commercial advertising, write to: MICROSYSTEMS, 39 East Hanover Ave., Morris Plains, NJ 07950, or call Claudette Moore at (201) 267-4558.

In This Issue

Departments

*CP/M is a registered trademark of Digital Research.

MICROSYSTEMS (ISSN#0199-7955) is published bi-monthly by Microsystems, P.O. Box 789-M, Morristown, NJ 07960. Controlled circulation pending at Allentown, PA.

POSTMASTER: Send address changes to: MICROSYSTEMS, P.O. Box 789-M, Morristown, NJ 07960.

EDITOR'S PAGE

by Sol Libes

IEEE S-100 Standard—Where Art Thou?

The IEEE proposed standard for the 696-bus, better known as the S-100 bus, was developed in the spring of 1979. The official version of the proposed standard was published in the July 1979 issue of *COMPUTER* magazine (the IEEE's official computer magazine) and was also reprinted in the January 1980 issue of *MICROSYSTEMS* magazine. The authors of the standard were Kells Elmquist, Ithaca Intersystems Inc., Howard Fullmer, Parasitic Engineering Inc., David B. Gustavson, Stanford Linear Accelerator Center and George Morrow of Morrow Designs. George Morrow served as the committee chairman. Another important participant in the committee work was Robert Stewart, also the chairman of the IEEE Computer Standards Committees.

Nearly two years have passed since the standard was proposed. It has still not been adopted. The question is—what has happened? What is happening currently and when can we expect the standard to be adopted? After all, most of the other IEEE standards have moved through the adoption process in considerably less time.

It has been difficult for me to determine exactly what has happened with the standard. The primary problem is that I am on the east coast while all the committee work is being done on the west coast, and I am simply not able to attend the meetings. I have over the past year and a half made numerous phone calls to all the gentlemen listed above and have even managed to have personal meetings with two of them. I have come to the following conclusions:

The S-100 bus standard represents a substantially different situation from that of other bus standards such as the 488 and Multibus standards. In those cases the buses were first developed by one company, Hewlett-Packard in the case of 488 and Intel with Multibus, and they had clearly defined the bus before submitting them to the IEEE for adoption as standards. This was not the case with the S-100 bus.

The S-100 bus as developed by MITS in early 1975 was only very loosely defined. In fact there were about 20 undefined lines in the MITS definitions. MITS never saw fit to publish any timing definitions for the bus. Further, during 1975 through 1979 several dozen manufacturers entered the S-100 hardware market and implemented many variations on pin assignments and developed their own timing specifications.

Thus the committee was faced with the task of trying to develop a standard which would cause the least problems for existing products and yet provide features for systems that were now significantly different from the original MITS Altair system (the first computer system to use the S-100 bus). The participants in the committee all had different opinions on how these problems should be dealt with. The problem was compounded by a significant turnover on the committee and the failure of all committee members to attend all meetings. Because of differences that developed, George Morrow left the committee in the fall of 1979, and George Fullmer took over as committee chairman.

The committee had solicited comments and criticism on the proposed standard during the fall of 1979. Close to a hundred responses were received. Some pointed out errors; most made suggestions which were easily integrated into the standard. However, there were a few questions raised on which there were differing opinions among the committee members. These led to numerous discussions during 1980. George Fullmer has informed me that these issues have been resolved and that hopefully the standard will be formally submitted for adoption by the IEEE in the late spring. George promised that I would receive a copy

of the changes to the proposed standard which we could publish in *MICROSYSTEMS*. We will certainly publish these changes as soon as we receive them.

If all goes smoothly, the IEEE-696/S-100 standard will be adopted by the summer of 1981. It is encouraging that many of the S-100 equipment manufacturers have already adopted the standard in their new designs. Regretfully, there are still many products being sold that were designed before the publication of the standard, which as yet do not conform to the standard. Perhaps these manufacturers will redesign their products to conform to the standard.

One last problem is that some manufacturers of S-100 products are claiming to conform to the IEEE-696 standard when in fact they do not. They have taken liberty with some of the pin assignments, particularly the RFU (Reserved For Future Use) pins and have not really checked that their timing conforms to the standard.

It would be really nice to have a "Consumer's Union" type of organization that could review various S-100 products for conformance to the standard. *MICROSYSTEMS* would like to do this someday. We are asking all the authors reviewing hardware for us to check that the pin assignments for the S-100 boards conform to the standard. If the products are not in conformance their reviews should point out the differences. We are also asking reviewers to test the boards with other boards that conform to the IEEE-696 standard so that there is some assurance of compliance with the timing requirements. If we do this much I feel that we can eliminate most of the incompatibility problems that have plagued the S-100 area now.

I am proud to say that I have been elected president of the Amateur Computer Group of New Jersey (ACG-NJ). The group has close to 1,300 members and is one of the largest computer clubs in the world. I am the founder of the club and served as its first president from 1975 through 1978. If you would like to learn about the group's activities and receive a sample copy of its 20-24 page monthly newsletter, send 50 cents (U.S. or Canada) or $1.00 (Elsewhere—cash. postal coupons or U.S. bank check only) to: ACG-NJ, 1776 Raritan Rd., Scotch Plains, NJ 07076. □

LETTERS TO THE EDITOR

Dear Editor:

I must protest the article published in the September/October issue of your new magazine supposedly evaluating the Intel 8088.

To repeat a worn-out cliche, the article is comparing apples to oranges. How can anyone compare a 16-bit machine with an 8-bit data bus to a machine that is only 8 bits wide and only a few crude 16 bit operations. It compares a machine that is a downgrade from a full blown 16-bit micro to an enhancement of a regular 8 bit micro. The Z-80 would have been a better choice to compare to the 6809.

The article states that the 8088 is "more than 4 times faster." Well I hope so! If it's running at 5 MHz, I hope it executes code faster than a machine running at 1 MHz. Does this mean if it ran at 1 MHz it would actually run slower?

Of special interest was the selection of the benchmark program. Most were designed with a 16-bit functions of the 8088 in mind; especially the graphics, 16 bit multiply and word shift routines. This isn't exactly fair to expect an 8-bit machine to execute 16-bit operations as easily as a 16 bitter. After all, isn't that the advantage of the 16-bit machines? And wasn't it curious to note the closeness of the two machines when it came to routines that both machines could run equally well, especially when you consider the extra speed the 8088 is running?

The question of the programmer comes up at this point. Since we don't get to see the actual routines we can only assume an Intel programmer is most familiar with the Motorola device. Sure!

Up to this point I realize that this article was given to S-100. The article may or may not reflect the facts but one is for sure. Your magazine *is* responsible for its publishing. How irresponsible to print an article without even an author's name. Smacks of newspapers and television newsmen's "informed sources." Just another way to report a rumor. I work with Z-80 and 8088 systems and own two 6809 systems and I believe the Motorola device is the best 8-bit micro on the market. It has taken the best of its predecessors and combined them into a powerful device. I am becoming familiar with the 8088 and feel it does have many advantages. I think you could exercise a little responsibility and check facts before you publish.

John Groves
Indiana University/Purdue University/
Indianapolis

Response from the author:

The comparisons between the 8088 and 6809 were extracted from a published Intel benchmark report.

I agree with Mr. Grove's points regarding the choice of benchmarks and the competence of the Intel programmer. These issues are both subject to evaluation by each reader.

The 6809 is an excellent microprocessor. In fact, for many (particularly real-time, interrupt driven) applications, the 6809 is faster *than the 8088. Mr. Groves is in error when he attempts to draw conclusions based on each machine's clock rate. First, the comparisons made were between the 5MHz 8088 an the 2MHz 6809. Second, the 8088 implements a* four clock bus cycle, *while the 6809 uses a* one clock bus cycle. *This makes direct comparisons based on clock rates meaningless.*

T.W. Cantrell
San Jose, CA

Dear Editor:

With reference to "PASCAL Speed Comparisons" by Fred Greeb (Sept/Oct 1980). I want to report the microengine times. This is now appropriate for the S-100 Bus. Program A and B ran at 0.7 seonds, program C ran at 0.8 seconds, program D required 7.1 seconds. Of course this is 16 bits. I work alternately with the IMSAI and have not verified Fred's results for the 8-bit operation.

Robert C. Luckey, MD
Richland, WA

Mr. William Gates
c/o MICROSOFT
10800 NE Eighth
Suite 819
Bellevue, WA 98004

Dear Mr. Gates,

I am writing this letter to you because a very serious question has arisen in my mind about the pricing policies of software from various vendors. I am also certain that the same questions have arisen in the minds of many other S-100 bus and CP/M users. It is for this reason that I am also sending a copy of this letter to Mr. Sol Libes of MICROSYSTEMS Magazine and will also inform him of your reply.

In the November 1980 issue of *BYTE* magazine (page 123) you have an advertisement for mu-MATH and mu-SIMP running under CP/M with ad advertised price of $250 for the package.

In this same issue (page 321) your Consumer Products Division offers ostensibly the same product for the TRS-80 at $74.95. While there is also an editorial by Gregg Williams starting at page 324, in which he alludes to certain supposed differences, I would appreciate your comments and explanation for the vast difference in price.

Furthermore, if there is a difference and the cheaper package is available for the TRS-80, should not this option also be available to those of us who use a Z-80 on the S-100 bus under CP/M?

To be frank, there has been a flurry of articles of late about the concern of various software houses about having their products 'ripped-off' by unscrupulous

NEW! TPM* for TRS-80 Model II
NEW! System/6 Package
Computer Design Labs

Z80* Disk Software

We have acquired the rights to all TDL software (& hardware). TDL software has long had the reputation of being the best in the industry. Computer Design Labs will continue to maintain, evolve and add to this superior line of quality software.

— Carl Galletti and Roger Amidon, owners.

Software with Manual/Manual Alone

All of the software below is available on any of the following media for operation with a Z80 CPU using the CP/M* or similar type disk operating system (such as our own TPM*).

for TRS-80* CP/M (Model I or II)
for 8" CP/M (soft sectored single density)
for 5¼" CP/M (soft sectored single density)
for 5¼" North Star CP/M (single density)
for 5¼" North Star CP/M (double density)

BASIC I

A powerful and fast Z80 Basic interpreter with EDIT, RENUMBER, TRACE, PRINT USING, assembly language subroutine CALL, LOADGO for "chaining", COPY to move text, EXCHANGE, KILL, LINE INPUT, error intercept, sequential file handling in both ASCII and binary formats, and much, much more. It runs in a little over 12 K. An excellent choice for games since the precision was limited to 7 digits in order to make it one of the fastest around. $49.95/$15.

BASIC II

Basic I but with 12 digit precision to make its power available to the business world with only a slight sacrifice in speed. Still runs faster than most other Basics (even those with much less precision). $99.95/$15.

BUSINESS BASIC

The most powerful Basic for business applications. It adds to Basic II with random or sequential disk files in either fixed or variable record lengths, simultaneous access to multiple disk files, PRIVACY command to prohibit user access to source code, global editing, added math functions, and disk file maintenance capability without leaving Basic (list, rename, or delete). $179.95/$25.

ZEDIT

A character oriented text editor with 26 commands and "macro" capability for stringing multiple commands together. Included are a complete array of character move, add, delete, and display function. $49.95./$15.

ZTEL

Z80 Text Editing Language - Not just a text editor. Actually a language which allows you to edit text and also write, save, and recall programs which manipulate text. Commands include conditional branching, subroutine calls, iteration, block move, expression evaluation, and much more. Contains 36 value registers and 10 text registers. Be creative! Manipulate text with commands you write using Ztel. $79.95/$25.

TOP

A Z80 Text Output Processor which will do text formatting for manuals, documents, and other word processing jobs. Works with any text editor. Does justification, page numbering and headings, spacing, centering, and much more! $79.95/$25.

MACRO I

A macro assembler which will generate relocateable or absolute code for the 8080 or Z80 using standard Intel mnemonics plus TDL/Z80 extensions. Functions include 14 conditionals, 16 listing controls, 54 pseudo-ops, 11 arithmetic/logical operations, local and global symbols, chaining files, linking capability with optional linker, and recursive/reiterative macros. This assembler is so powerful you'll think it is doing all the work for you. It actually makes assembly language programming much less of an effort and more creative. $79.95/$20.

MACRO II

Expands upon Macro I's linking capability (which is useful but somewhat limited) thereby being able to take full advantage of the optional Linker. Also a time and date function has been added and the listing capability improved. $99.95/$25.

LINKER

How many times have you written the same subroutine in each new program? Top notch professional programmers compile a library of these subroutines and use a Linker to tie them together at assembly time. Development time is thus drastically reduced and becomes comparable to writing in a high level language but with all the speed of assembly language. So, get the new CDL Linker and start writing programs in a fraction of the time it took before. Linker is compatible with Macro I & II as well as TDL/Xitan assemblers version 2.0 or later. $79.95/$20.

DEBUG I

Many programmers give up on writing in assembly language even though they know their programs would be faster and more powerful. To them assembly language seems difficult to understand and follow, as well as being a nightmare to debug. Well, not with proper tools like Debug I. With Debug I you can easily follow the flow of any Z80 or 8080 program. Trace the program one step at a time or 10 steps or whatever you like. At each step you will be able to see the instruction executed and what it did. If desired, modifications can then be made before continuing. It's all under your control. You can even skip displaying a subroutine call and up to seven breakpoints can be set during execution. Use of Debug I can pay for itself many times over by saving you valuable debugging time. $79.95/$20.

DEBUG II

This is an expanded debugger which has all of the features of Debug I plus many more. You can "trap" (i.e. trace a program until a set of register, flag, and/or memory conditions occur). Also, instructions may be entered and executed immediately. This makes it easy to learn new instructions by examining registers/memory before and after. And a RADIX function allows changing between ASCII, binary, decimal, hex, octal, signed decimal, or split octal. All these features and more add up to give you a very powerful development tool. Both Debug I and II must run on a Z80 but will debug both Z80 and 8080 code. $99.95/$20.

ZAPPLE

A Z80 executive and debug monitor. Capable of search, ASCII put and display, read and write to I/O ports, hex math, breakpoint, execute, move, fill, display, read and write in Intel or binary format tape, and more! on disk $34.95/$15.

APPLE

8080 version of Zapple $34.95/$15.

NEW! TPM now available for TRS-80 Model II!

TPM*

A NEW Z80 disk operation system! This is not CP/M*. It's better! You can still run any program which runs with CP/M* but unlike CP/M* this operating system was written specifically for the Z80* and takes full advantage of its extra powerful instruction set. In other words its not warmed over 8080 code! Available for TRS-80* (Model I or II). Tarbell, Xitan DDDC, SD Sales "VERSA-FLOPPY", North Star (SD&DD), and Digital (Micro) Systems. $79.95/$25.

SYSTEM MONITOR BOARD (SMB II)

A complete I/O board for S-100 systems. 2 serial ports, 2 parallel ports, 1200/2400 baud cassette tape interface, sockets for 2K of RAM, 3-2708/2716 EPROM's or ROM, jump on reset circuitry. Bare board $49.95/$20.

ROM FOR SMB II

2KX8 masked ROM of Zapple monitor. Includes source listing $34.95/$15.

PAYROLL (source code only)

The Osborne package. Requires C Basic 2.
5" disks $124.95 (manual not included)
8" disks $ 99.95 (manual not included)
Manual $20.00

ACCOUNTS PAYABLE/RECEIVABLE (source code only)

By Osborne, Requires C Basic 2
5" disks $124.95 (manual not included)
8" $99.95 (manual not included)
Manual $20.00

GENERAL LEDGER (source code only)

By Osborne. Requires C Basic 2
5" disks $99.95 (manual not included)
8" disks $99.95 (manual not included)
Manual $20.00

C BASIC 2

Required for Osborne software. $99.95/$20.

SYSTEM/6

TPM with utilities, Basic I interpreter, Basic E compiler, Macro I assembler, Debug I debugger, and ZEDIT text editor.
Above purchased separately costs $339.75
Special introductory offer: Only $179.75 with coupon!!

$160.

This Coupon Is Worth One Hundred And Sixty Dollars Toward The Full Price Of The SYSTEM/6 Package System/6 with this coupon is only $179.95. This is a limited time offer.

$160.00

ORDERING INFORMATION

Visa, Master Charge and C.O.D. O.K. To order call or write with the following information.

1. Name of Product (e.g. Macro I)
2. Media (e.g. 8" CP/M)
3. Price and method of payment (e.g. C.O.D.) include credit card info. if applicable.
4. Name, Address and Phone number.
5. For TPM orders only: Indicate if for TRS 80, Tarbell, Xitan DDDC, SD Sales (5¼" or 8"). ICOM (5¼" or 8"), North Star (single or double density) or Digital (Micro) Systems.
6. N.J. residents add 5% sales tax.

Manual cost applicable against price of subsequent software purchase in any item except for the Osborne software.

For information and tech queries call
609-599-2146

For phone orders ONLY call toll free
1-800-327-9191
Ext. 676
(Except Florida)

OEMS

Many CDL products are available for licensing to OEMs. Write to Carl Galletti with your requirements.

* Z80 is a trademark of Zilog
* TRS-80 is a trademark for Radio Shack
* TPM is a trademark of Computer Design Labs. It is not CP/M*
* CP/M is a trademark of Digital Research

Prices and specifications subject to change without notice.

DEALER INQUIRIES INVITED.

COMPUTER DESIGN LABS

342 Columbus Avenue
Trenton, N.J. 08629

Letters con't...

hobbyists. I am one of the legitimate purchasers of your MBasic, having paid approximately $350 for it, and so I feel that I have a legitimate right to question this apparent discrepancy.

I simply want to be certain that the same immoral tactics that the software dealers have been accusing all of us of engaging in are not being applied in reverse to a select segment of the microcomputing community customers.

I look forward with great interest to your reply.

Raymond M. Glueck
Pine Brook, NJ

Response from the manufacturer:

Dear Mr. Glueck:

Software pricing is a complex and difficult area. Software is a product that's marginal cost is extremely low while development cost is quite high, and so any pricing strategy is somewhat influenced by the volume of potential sales. For example, IBM's COBOL Compiler licensed over a two year period cost $50,000 where Microsoft CP/M COBOL Compiler is a $750 package. I'm sure these differences lead to confusion in the minds of end-users and tend to undermine the legitimate effort of companies like ourselves to be paid for our work. As such we try to keep to a minimum any inconsistency in our own pricing policy. The particular instance you pointed out where muMATH costs $75 on the TRS-80 in its stripped down form and costs $250 on CP/M in its full form, is a case where we have decided to segment the market by actually having two products. We think there is high demand for a package where you simply get at the built-in capabilities of muMATH without having all description or full access to either the sources or the capilities of muSIMP. Also several of the advanced math capabilies have been restricted to the full version. The full version requires a fairly complex manual that would probably be confusing to the casual user of the package. As such we did define these two versions. Right now we offer both the $75 version and the $250 version into the TRS-80 market. In the CP/M market we should probably make available the $75 version for maximum consistency. However, the volume of sales in the CP/M market and its tilt towards more sophisticated users would probably not make this worthwhile. The difficulty we have at this point is that where we've tried to use the same manual to address two different markets like CP/M and TRS-80, there has been a great deal of criticism, and so it's our policy to modify our manuals extensively and reprint them to reach particular operating environments. Because we are a growing company doing a lot of packages, there is a high opportunity cost of going back and doing that for the smaller version of muMATH at this time.

To summarize, the $75 version is substantially different than the $250 version—both in terms of features, the manual and the sophistication of the user it is directed to. Likewise, our support burden on the complete version has proven to be significantly greater. We have given TRS-80 users the option of picking either of these packages, but as yet cannot justify making the $75 version available to the CP/M market. We are hopeful that the success of our Softcard will increase the number of CP/M users to the point where we are able to justify giving them a greater variety of products than we have available for CP/M today. However, at this point there are more TRS-80 TRSDOS based system than CP/M systems.

Please feel free to give me a call if you have further interest.

Bill Gates
Bellevue, WA

Dear Editor:

While the S-100 bus is *the* hardware bus, it seems that Apple and TRS-80 have most of the game software. I'd really like to read their disks, but my North Star DOS and hard-sectored North Star drive won't (as far as I know) let me. I am in the market for an S-100 system (drive, controller, software patches for DOS) that would let me read these disks. Unfortunately I can't seem to find anything like this on the market. Even if I were to do the software patching, all I find is advertisements that hint but do not promise anything. At $400 to $1000 prices, I can't afford to buy a pig in a poke—I must have promises.

It would seem to me that there *is* a market for this sort of thing. Could you see if any manufacturers are interested in selling to it? If not, then please just publish this letter. Maybe they can take a hint.

Also, if someone out there knows a DOS-patch to let me read those disks, I'd be very happy to hear from them.

John J. Cleary
Wichita, KS

HOW'S YOUR MEMORY?

Our MEMORIES Have Made US Famous!

WHY?

Reliability!

That's the secret of our success. We are dedicated to producing the most reliable memories possible for the S-100 bus and the new IEEE S-100 bus.

Flexibility!

No matter what CPU you use, we have the memory for you.

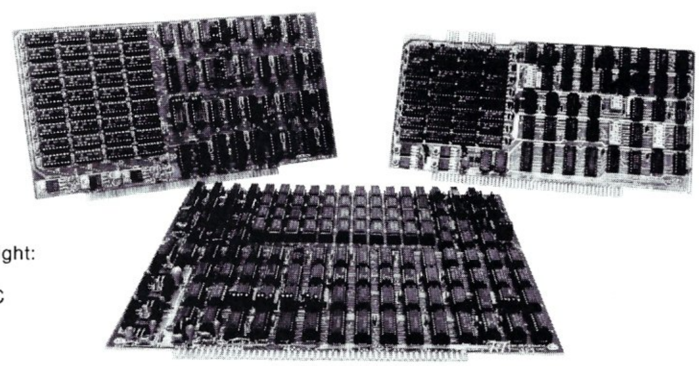
SUPERAM™ 2—We've delivered over 5,000 of these 64K dynamic RAM boards to OEMs worldwide. This board sets the standard for reliable operation.

SUPERAM™ 4—Now available for Z80A and 8085A users, with bank select and optional parity bit.

SUPERAM™ 5—Our forthcoming IEEE-696 (S-100) compatible RAM will provide 64 Kbytes of dynamic RAM as either 32Kx16 or 64Kx8.

With ECC, too!

Announcing the SUPERMEM™ ECC! This is the ultimate in memory data protection: built-in automatic error correction. The SUPERMEM transparently corrects single bit errors and detects double bit errors.



Shown Left to Right:
SUPERAM 4
SUPERMEM-ECC
SUPERAM 2

We'll Design One for You!

If you have a special situation that calls for a custom design involving memory, processors, controllers, or even complete systems, give PIICEON a call. We might have the answer to your product need.

And That's Not All!

We also supply the following Microcomputer Boards:

- C-86 — 16-bit 8086 processor board with multi-processor arbitration logic, on-board PROM and RS-232 port.
- V-100 — 80x24 high-speed video output board.
- D-100 — Coming soon, the answer to 5¼ and 8-inch Winchester control.

WRITE FOR OUR NEW BROCHURE

PIICEON™

OEM Computer Products
2350 Bering Drive
San Jose, CA 95131
(408) 946-8030

NEWS & VIEWS

CP/M & S-100 Sessions At Computer Faire

Sol Libes, Editor of Microsystems, will chair two sessions at the West Coast Computer Faire (see ad in this issue). The panelists and topics are:

Session #1—CP/M & MP/M User Meeting
Dr. Gary Kildall, Digital Research
topic: New CP/M & MP/M developments
Tony Gold, Lifeboat Associates
topic: CP/M applications software
Dr. Bruce Kendall, Corvus Systems Inc.
topic: A user's perspective of CP/M

Session #2—The IEEE 696/S-100 Standard
Howard Fullmer, Parasitic Engineering Chairperson IEEE 696/S-100 Committee
topic: Status of standard's approval
Kels Elmquist, Ithaca Intersystems Inc.
topic: S-100 standard's DMA protocol
Mark Garetz, Godbout Electronics
topic: IEEE-696 compatibility

A 5-MByte Disk System For Under $2K

Shugart Associates is offering a special deal on their SA1002 5-MByte Winchester disk system. For $1,950 you can get the drive, an SA1400 intelligent controller card, cables and documentation. The controller provides for back-up on single or double-sided floppy disks. You will have to do some minimal interfacing to get the system running.

This special deal is really intended as an evaluation offer to OEM's considering the design of the SA1002 into their computer systems. But Shugart has placed no restrictions on the offer, which is called the "Success Kit," and anyone can buy just one unit. To get more information call (800) 852-7888, or in California call (800) 852-7777 (operator 12).

Give Your Printer Ribbon New Life

Printer ribbons wear out pretty quickly and replacement is not cheap, particularly on the closed-loop type ribbons. Therefore, any way to extend their life is a welcome blessing. Here is one method, as told to me by a computer hobbyist friend of mine.

Carefully pry off the lid on your closed-loop ribbon case. Do it slowly so that the ribbon does not pop out of the case. Then spray the ribbon with WD-40 all-purpose lubricant. This will cause the ink from the unused portions of the ribbon to flow down onto the dry, used areas of the ribbon. Be careful not to over-spray the ribbon as this leads to streaking. Close the lid and let the ribbon sit overnight for the lubricant to do its work. You should be able to do this as many as three or four times before the ink in the ribbon is completely spent.

Computer Consultants Association

If you are doing independent computer consulting you should check out the "Independent Computer Consulting Association," Box 27412, St Louis, Missouri 63141, phone (314) 567-9708. They publish a newletter with tips for consultants, an annual directory and provide useful resource materials (e.g. sample contracts). They also have several chapters which hold meetings with speakers.

8086 Version Of Cobol Announced

The 8086 software picture keeps improving. Seattle Computer Products, Seattle WA,has announced a version of Microsoft Basic for the 8086 and now Microsoft, Bellvue WA, has announced Cobol-86 which will run under the CP/M-86 operating system.

It is expected that the execution time of these packages will be three times as fast as the 8080/Z80 versions. Further, because of the global call capabilities of the 8086 the packages will be better suited to multiple user systems than the 8-bit versions.

Pascal/Z User Group Releases Disks

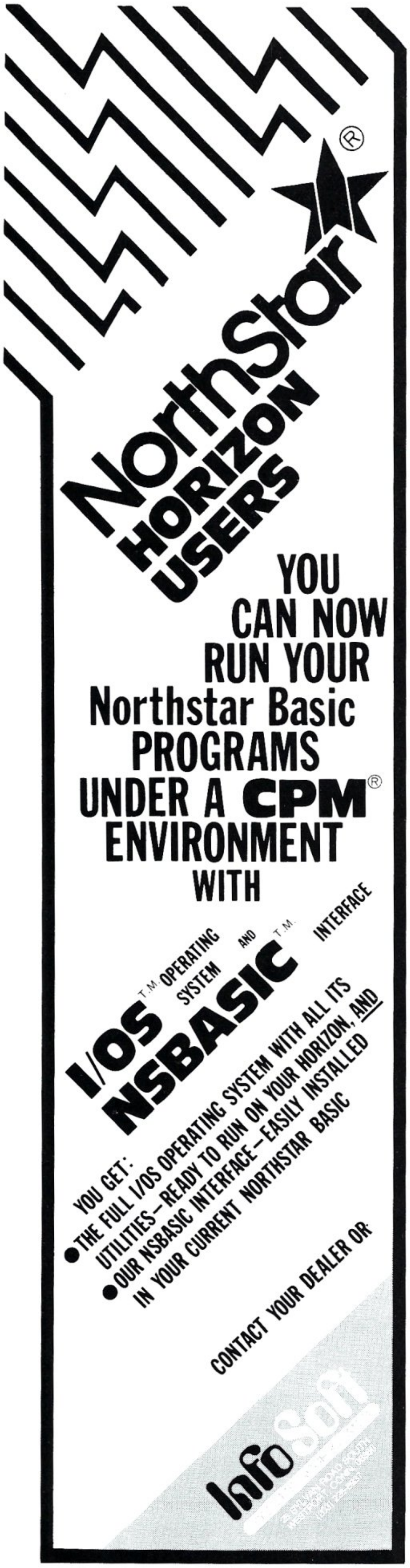
The Pascal/Z User Group is devoted to supporting users of the Ithaca Intersystems Z80 and Z8000 hardware and software products. They have been up and running for about 8 months now. They have already released four 8" disks of public domain software (disk #5 will be released shortly) and have published four copies of their newsletter. The work has been done primarily by Charlie Foster.

The group charges $6 (U.S. funds) to get on the mailing list. They charge $10 (postpaid) for each disk of software.

For more information call or write Charlie Foster c/o ZUG, 7962 Center Parkway, Sacramento, CA 95823. His telephone is: (916)392-2798 (home), or (916)447-6077 (work).

Zilog Z8000 Home Study Course

Zilog has announced a short home-study course for the Z8000. You can study at you own pace at home. The tests you take are sent to Zilog for individual grading and critiquing. The course costs $39 and is available from: Zilog Training and Education, 10460 Bubb Road, Cupertino, CA 95014. □



Is There an Alternative to CP/M — Try it — You'll Like it say the developers. — Westport, CT

InfoSoft Systems, Inc., the developers of I/OS 2, think they have an answer to the complaints about CP/M (the trademarked product of Digital Research). InfoSoft's head of development revealed many of the advanced features that I/OS allows while still supporting all previous CP/M programs. He discussed what the development group learned from their previous development of Cromemco's CDOS, with over 15,000 installations. I/OS 2 is geared, he stated, toward the turnkey system and the developer, yet presents a friendly face to the user. Such features as turnkey startup, local batches, no system on the 'A' disk, and terminal controls are integrated into the design.

Taking concepts from Unix, I/OS 2 has full hard disk support, the ability to mix various sized disks; and a clean documentation format. For the developer, I/OS is the only system available with a full mix of devices and disks supported by the developer, and a question and answer system creation process. End user price is similiar to CP/M, yet the dealer has a greater margin to work with, and far more support from the factory.

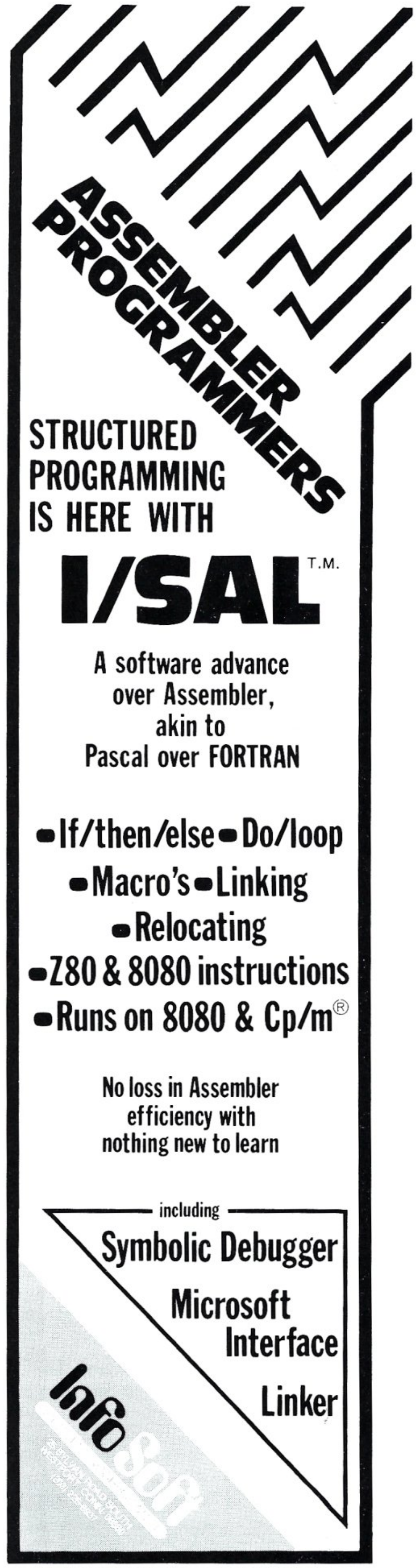
The spokesman refused to comment upon rumors that other, lesser developed programs were being released for the more sophisticated hacker.

• • •

Software Beasts released from the dungeons — Westport, CT

A spokesman for InfoSoft Systems today admitted that strange creatures were being released from their development dungeons below their software sales offices in beautiful downtown Westport. "These creatures are totally unrelated to our normal superb line of micro-processor software", stated the speaker.

Stressing that the beasts were not harmful, she admitted they did tend to bewitch hackers, forcing them to tinker till all hours. Under intense cross-examination he disclosed that these creatures, while only half-formed, were very useful as assistants or homunculi for would-be software wizards. Before disolving into a puff of green smoke, she (???) intimated that full details on capturing the creatures were available from the sales desk.



The CP/M Bus

by Anthony Skjellum

CP/M is used in a wide variety of environments on many types of machines. It provides a powerful operating environment and allows portable software to be created. This regular column will serve to enhance reader knowledge of specific facets of CP/M and applications running on various systems. It will attempt to answer your questions regarding CP/M and related topics. Please mail all questions to:

The CP/M Bus
c/o Microsystems
Box 1192
Mountainside, NJ 07092

Other topics of interest will also be presented from time to time. I also expect to describe interesting software from the CP/M Users' Group as space permits.

I believe that software bugs should be documented as discovered. I will pass on information as I discovered it; if you discover bugs, please don't hesitate to inform me. This month I will discuss: 1) CP/M2, the newest version of CP/M; 2) A Bug report on a problem with Lifeboat's COPY utility and errors in *The CP/M Handbook*.

CP/M—The new release

CP/M 2.2 is the latest release of the operating system. It provides new features which are upwardly compatible with CP/M 1.4 and allows a greater flexibility in software development. CP/M2 was designed with the newer flexible and non-flexible disk technologies in mind. It can handle devices of up to 8 megabytes capacity. It also corrects some of the annoying aspects of release 1, including some nice changes to parameter passing which are more natural for 8080 and Z80 assembly language programmers. Furthermore, file extents are handled more logically to provide more effective use of directory space and greater efficiency in data access. Random access file I/O is also supported. The purpose of this overview is to describe the benefits of release 2. This is not intended to be tutorial so I will not discuss the new BDOS commands, but only discuss the capabilities of the operating system. In a future column I plan to describe additional features which might well be added to the CP/M environment.

Enhancements to CCP and line editing

The CCP contains a new command in CP/M2. This is USER command which sets CP/M's path to one of 16 possible "logical" user areas. Each file on every disk has a user parameter set to a value from zero to fifteen that indicates to which user area it belongs. Each user area is separate from all the others. The enhanced PIP utility permits reading from another user area—but not writing to it from a distinct area. This feature permits compatibility with MP/M which is designed for multiuser applications. I find these user areas useful for organizing data on disk.

For hard disk systems where a large amount of data is involved, this becomes even more useful. Because of the restriction on writing to user areas, it is necessary to move PIP to the new area via a programming "trick." Digital Research recommends that PIP be loaded into the TPA (transient program area) with DDT and then saved in the new user area with the SAVE command at the CCP level. I present an alternative method below which makes use of the "zero program," @.COM. This transient allows programs already in the TPA to be re-executed, since it is of zero length (see my article in DDJ Vol 5 #1, January 1980). I use this technique since it does not require DDT or the less convenient SAVE procedure.

```
We start in USER x and will initialize USER y
where x<>y.

A>PIP ^A                    ; control char causes
                            ; PIP to be loaded but not
                            ; executed.
A>USER y <CR>               ; enter user area y
A>SAVE 0 @.COM <CR>         ; create zero program
A>@ <CR>                    ; execute zero program
                            ; to effectively run PIP.
*PIP.COM=PIP.COM[Gx]<CR>    ; move PIP program into
                            ; current area.

you may now move additional programs as needed.

     nb: x,y are integers on the closed
         interval zero to fifteen.
```

Another nice feature of release 2 is that backspaces are interpreted as a delete request in line input sequences. Actual backspacing is performed instead of echoing. I have found echoing very annoying, especially when running applications programs which use the line input function of CP/M. Also, DIR listings are four columns wide which is very convenient. One needn't use a special program to read the directory in a readable way.

File attributes, enhanced STAT and PIP

File attributes are a nice addition to CP/M. One may make a file read-only, read-write or set the SYS attribute which makes it invisible to directory commands. ED.COM will not allow SYS files to be edited either. The write-protect feature is clearly useful and the SYS attribute allows standard programs to be made "invisible." This makes directory listings more readable, since standard programs don't appear. This may simplify operation for experienced users also. STAT.COM reports the write attributes flag and parenthesises SYS files. Attributes are set and cleared via STAT:

```
STAT *.* $R/W          ; set all files to read-write
STAT *.COM $SYS        ; add system attribute to all
                       ; .COM files
STAT *.COM $DIR        ; remove system attributes
                         (undoes previous example)
```

It is also possible to list file size via the $S option which is useful for files created with random access commands supported by CP/M2. This is not necessary for files created sequentially, since the values will be the same.

STAT also has several new "devices." "DSK:" displays the attributes of all active disks in the system. (Optionally, the user may specify "dn:DSK:" where dn is the disk name.) Remember, in release 2, disks may have very diverse hardware characteristics and capacities. This option displays vital statistics about those characteristics. The "USR:" device returns the active user area and reports which areas have files. By the way, this option cannot be re-executed with the zero program described above, as it reports active user areas incorrectly on the second and subsequent runs of the program.

Several additions have been made to PIP. As illustrated above, the "G" option (get) allows reading files from one user area into another. Because of the write-protect option, the "W" command option has been added. This allows PIP to overwrite read-only files. If this option is not specified, PIP asks for permission to overwrite read-only files. PIP will also ignore system files unless the new "R" option is specified. Here are some examples:

```
PIP B:=A:*.TXT[W]          ; overwrite read-only files
PIP D:=A:*.COM[R]          ; read system attributes files
                           ; as well as regular files
```

DDT, ED and ASM

Unfortunately, these three transients have not been improved significantly from version 1.4. I do not use ED and rarely use ASM in favor of a macro assembler. DDT is useful for modifying program and patching but I prefer a Z80 debugger to this transient which doesn't understand Z80 instructions. In any case, ED refuses to access $SYS or $R/O files in its operation and has line numbering set as the default since this feature is one of its greatest benefits.

SUBMIT and XSUB

On many timesharing systems, batch processes permit console input to originate from the batch file. SUBMIT 1.4 would not permit this but through the use of XSUB, this feature is implemented through SUBMIT 2.2. By placing XSUB as a file to execute in the batch process (.SUB) file, all line input requests by transients will be read from the disk instead of the console.

Summary

CP/M2 provides nice features which make it superior to version 1.4 and the increase in capabilities and efficiency make it worth the cost of upgrading. CP/M2 is a superior environment for hard disks as it provides a more flexible interface to the hardware environment.

II. Bug Reports

In this and succeeding columns, various corrections to CP/M related software and documentation will be made.

Problem with COPY

Since installing CP/M2 on my system, I have noted that the COPY utility provided by Lifeboat Associates (for CP/M2 on Micropolis) has a serious bug. It does not seem to format the last track of the disk correctly. When filling a disk to capacity, bad sector errors appear. These errors do not occur when the disks are initialized with FORMAT under CP/M 1.4 and subsequently used under CP/M2. I have not had time to debug this program but thought that this should be disclosed.

Errors in "The CP/M Handbook with MP/M"

I recently purchased *The CP/M Handbook with MP/M*, by Rodnay Zaks. (This is a Sybex book.) It is intended to be an introductory text for novices and describes both CP/M and MP/M. I found several errors which should be noted:

```
                  Is:                    Should be:

page 70:  A>STAT B: <CR>

BYTES REMAINING ON B: 192K

B:R/O                              This line does not appear.

page 116:

A>A:=B:FILE.INT <CR>               A>PIP A:=B:FILE.INT <CR>

A>B:=A:FILE.INT <CR>               A>PIP B:=A:FILE.INT <CR>
```

I would recommend that beginners obtain a copy of this book and study it carefully. It has sound information about the general operation of a computer room as well as good description of the CP/M and MP/M operating systems.

Conclusion

I hope that this and future installments of "CP/M Bus" prove useful and I would again ask for your suggestions, questions, and ideas about CP/M software and programming. □

The CP/M Connection, Part 4

by Chris Terry

Part IV—How to use CP/M Facilities In Your Own Applications Programs

Previous articles in this series described how the CP/M file management system works, and how the I/O system can be enhanced to give a choice of peripherals. This article discusses how the facilities provided by CP/M can be used in your own application programs, with particular emphasis on portability.

PORTABILITY CONSIDERATIONS
Avoiding System Incompatibility

Let me say at the start that, for programs intended to be run *only* on one's own system, there is no reason not to make use of any and all the facilities of CP/M. Disk I/O primitives and peripheral drivers in the CBIOS may be invoked with subroutine calls—but at a price. The price paid is that the program may not run on any system that uses a different disk format. Prime examples of this are the SAP (Sort And Purge directory), WDIR (Wide DIRectory), and XDIR (extended attribute DIRectory) programs in Volumes 19, 4, and 24 of the CP/M Users' Group library. These work like a charm on systems using CP/M Version 1.4 with 8" IBM-compatible (soft-sectored) drives, and are extremely useful utilities. But they all use direct calls to disk primitives in the CBIOS that assume 77 tracks, 26 sectors per track, and 128 bytes per sector. They do not run on any other disk system, nor under CP/M Version 2. SAP, in particular, destroys the directory if run on a double-density system. All three of these utilities need extensive recoding for double-density or hard sectoring.

There are some advantages in using direct calls, in spite of David E. Cortesi's vigorous letter of protest in the February 1981 issue of *Interface Age*. He rightly points out that direct calls to console drivers lose the facilities provided by CCP (e.g., the ability to suspend/continue an operation with a Control-S). However, there are times when one wishes to use control characters in a manner that conflicts with CCP's use of them; one must then either change the design to use different characters, or employ a direct call to the console driver. Also, direct calls can avoid the need to save registers not used by the primitives.

In general, portability demands that an application program:

1. Perform ALL input and output operations via calls to BDOS, not direct to the CBIOS drivers.
2. Contain all other routines required by the application.
3. Provide adequate stack space of its own, storing the CP/M stack pointer on entry and restoring it on exit.
4. Save all registers containing significant data before each call to BDOS, and restore them on return from the call.

Performing all I/O via BDOS ensures that these operations are system-independent. Regardless of system version or size, a BDOS call takes the function code in the C register and a character or buffer address in the DE register pair; and the BDOS entry point is at 0005H.

Point No. 2 may seem obvious, but sometimes gets forgotten when adapting a private program for publication. I have a 5K ROM monitor that combines and enhances the most useful features of Roger Amidon's Apple monitor and the old Processor Technology Software Package; it contains excellent buffer scanning and code conversion routines that I frequently call from CP/M application programs. But these are peculiar to my system, and when someone would ask me for a copy of one of my utilities, I sometimes would forget to extract one of them. Nowadays I keep the most useful ones in a .LIB file for easy inclusion in programs to run on other systems.

Point No. 3 is important and is often overlooked. CP/M has a pretty good stack, but a program that does a lot of PUSHing and POPping when subroutines are already nested to considerable depth can exhaust the CP/M stack and crash. I have encountered this problem several times, chiefly because my programming style leans toward a separate subroutine for each logical

Chris Terry, 324 E. 35 St., New York, NY 10016.

function; my main program loop is nothing but a series of subroutine calls to major functions, and those in turn call nested subroutines to perform subfunctions. My programs may run a little more slowly because of this—but then they are I/O limited rather than CPU limited, anyway; I certainly find that this style makes the logic easy to follow when I want to modify a program six months after it was written. Further functions can be added or algorighms changed with very little trouble.

Point No. 4 is also important. The BDOS uses all registers, and even a simple call for console input or output can destroy data in BC or HL if these registers are not saved. If a macro assembler is used, code Save and Restore macros and use them before and after every call to BDOS, remembering that single-byte results are returned in the A register, and double-byte results in BC.

Avoiding Size Incompatibility

CP/M systems come in all sizes from 16K (minimum) up to 64K (maximum without memory Bank switching). Regardless of system size, Page 0 (locations 0000 through 00FFH) and 22 pages at the top of memory are reserved for the use of CP/M. Thus, the area available for application programs and their work space varies from 40 pages (10K) in a minimum system to 248 pages (62K) in a maximum system. This variability has implications both for application programs which make only BDOS calls, and for those which make direct calls to CBIOS routines.

Finding Available Memory Size. Programs which perform extensive searches, such as text editors and formatters, and database update and retrieval programs, run faster if the data to be processed (or a substantial portion of it) is available for processing in memory. Also, telecommunication programs communicating with remote computers generally require large buffers to avoid interruption of data transfers on the line by disk accesses. Programs of this type need to know the last memory location available to them, so that they can define large buffers and workspace without encroaching upon CP/M. Locations 5, 6, and 7 in Page 0 always contain a jump instruction to the BDOS entry point; thus, the last available memory address can be found by the sequence:

```
LHLD 0006H
DCX H
SHLD MEMTOP
```

where MEMTOP is a 2-byte storage location within the application program.

Finding The CBIOS Jump Vector. Programs which make direct calls to CBIOS routines need to know the locations of these routines. One way of doing this is to include in the application source code equate statements that give the addresses contained in the CBIOS jump vector. This is an unsatisfactory method, however, since it ties the executable .COM program to one particular CP/M size; if the system is changed the equates must also be changed and the application program reassembled. It is better to define the CBIOS locations dynamically upon entry to the application program; the program will then run correctly on any system that uses a similar disk format.

There are numerous ways of locating the CBIOS routines at run time; all depend on the fact that CP/M places a jump instruction in location 0000, followed by the address of the Warm Boot routine, which is the second item in the CBIOS jump vector. This vector contains the following items:

```
JMP   BOOT     ;Arrive here from cold start load
JMP   WBOOT    ;Arrive here for warm start
JMP   CONST    ;Check for console input character ready
JMP   CONIN    ;Read character from console input
JMP   CONOUT   ;Write character to console display
JMP   LIST     ;Write a character to List device
JMP   PUNCH    ;Write a character to Punch device
JMP   READER   ;Read a character from Reader device
JMP   HOME     ;Move to track 00 on selected drive
JMP   SELDSK   ;Select a disk drive
JMP   SETTRK   ;Set track number on selected drive
JMP   SETSEC   ;Set sector number on selected track
JMP   SETDMA   ;Set address of disk I/O buffer
JMP   READ     ;Read selected sector
JMP   WRITE    ;Write selected sector
```

Once the address of the jump to WBOOT is known from locations 0001-0001, the application program can compute the address of any other CBIOS routine. The simplest way of doing this is to create an identical jump vector within the application program, as follows:

```
;**********************************************************
;GETVEC.LIB,  A routine to obtain
;CBIOS routine addresses for a local jump vector.
;From CP/M Users' Group Library, Volume 1.
;**********************************************************
;
ORG  100H
;
VECTRS:    JMP  GETVEC
           DS    42            ;This space will contain the local
                               ;jump vector after execution of GETVEC.
;--------------------------------------------------------------
WBOOT:     EQUVECTRS+3         ;Do NOT remove any items
CONST:     EQUVECTRS+6         ;from this list, or addresses
CONIN:     EQUVECTRS+9         ;will not match those in the
CONOUT:    EQUVECTRS+12        ;CBIOS jump vector.
LIST:      EQUVECTRS+15
PUNCH:     EQUVECTRS+18
READER:    EQUVECTRS+21
HOME:      EQUVECTRS+24
SELDSK:    EQUVECTRS+27
SETTRK:    EQUVECTRS+30
SETSEC:    EQUVECTRS+33
SETDMA:    EQUVECTRS+36
READ:      EQUVECTRS+39
WRITE:     EQUVECTRS+42
----------------------------------------------------------------
;
GETVEC:    LXI  D,WBOOT        ;Set Destination to start of local vector.
           LHLD 1              ;Get start address of CBIOS vector.
           MVI  B,42           ;Set byte count (14 jumps X 3).
GETVE1:    MOV  A,M            ;Get a byte from CBIOS vector
           STAX D              ;and copy it to the local vector.
           INX  H              ;Bump the
           INX  D              ;pointers,
           DCR  B              ;check the byte count,
           JNZ  GETVE1         ;and loop till done.
;
;When local vector complete, fall through
;into main body of application program.
;**********************************************************
```

BUFFEREDI/O

Buffered Console I/O

CP/M provides facilities for buffered I/O as well as single-character I/O. Calls to BDOS for buffered I/O have the same form as other BDOS calls—that is, the appropriate function code is placed in the C register and the starting address of the buffer in the DE register pair. A subroutine call to BDOS at the standard entry point (0005H) then initiates the operation.

Buffered console input is used by CP/M for commands and their arguments, using the default I/O buffer at 80H. On input, characters are echoed to the console display device and then accumulated in the buffer, starting at TBUF+2, and the byte count at TBUF+1 is updated. TBUF contains a constant representing the

CP/M Connection cont'd. . .

maximum buffer length (80 characters for the CCP).

Accumulation ends when a Carriage Return is entered, and control is returned to the calling program. Application programs can also use this facility; they can use the default buffer at TBUF, or can define a buffer of up to 256 characters (including the maximum length byte, the current byte count, and the Carriage Return terminator).

Buffered console output is used mainly for messages. The print buffer function is placed in the C register and the address of the string to be printed in the DE register pair. There is no limit on the string length; multiple lines can be printed in one operation by including Carriage Return/Line Feed (ODH, OAH) line separators. Printing stops when a Dollar sign ($, 24H) is encountered, and control is returned to the calling program. This function is strictly for ASCII printable characters; there are no facilities for converting hexadecimal or BCD numbers to ASCII.

Buffered Disk I/O

All data transfers to and from successfully opened disk files take place via a 128-byte (single-density) or 256-byte (double-density) buffer. BDOS is told the starting address of this buffer by a SETDMA function call; all subsequent read and write operation use this one-sector buffer until the address is changed by another SETDMA call to BDOS. If an application program does not execute a SETDMA call to BDOS, reading and writing takes place through the default buffer (TBUF) at 80H. The application program is responsible for filling the buffer before issuing a Write, and for extracting the data from the buffer after a Read; routines to do this are discussed later in the commented listings. The read and write calls must supply the address of a File Control Block; if reading/writing is sequential, BDOS updates the NR (Next Record) byte in the FCB after each operation.

In many applications it is desirable to read more than one sector before processing the data. In such cases, the application must define a buffer of appropriate size (4K is common, and editors may create buffers of 20K or more). A SETDMA call to BDOS establishes the address for the first read, and a further SETDMA augments the buffer address by one sector length after each read. The same procedure is used for writing multiple sectors to the disk.

In the next installment I will present examples and a complete program listing to illustrate the techniques discussed in this article. □

Software Consultants Harken!

Do you do software customizing? Do you do software package installation? Do you provide turnkey systems? Do you run a programming shop? Full-time? Part-time?

Starting with a future issue **MICROSYSTEMS** will publish a directory of "software shops." Places where computer customers can go for software package customizing. Most purchases of computer software packages find that the package does not exactly meet their needs—customizing is required—sometimes this is minor and sometimes it is major. Our directory will provide low cost advertising for "software shops." If you wish to be listed write to MICROSYSTEMS, Box 1192, Mountainside, NJ 07902 for information.

High Tech. — — — Low Prices

"No frills" O.E.M. pricing means real $aving$! Prices listed do not include sales tax (Minnesota residents only) or shipping charges. Prepayment eliminates shipping charges on orders within North America. We require a minimum 25% down payment on all orders. Systems houses, institutions, computer clubs and O.E.M.s are encouraged to inquire for further price reductions on quantity purchases - please contact our office for formal quotations. Yes, we will export to all points of the globe - purchase of an appropriate support/repair/diagnostics kit is suggested.

To preserve the integrity of the products that we sell, we wish to clarify one ever increasing problem:

Due to the difficulty in providing adequate product support cross-country, we cater primarily to the experienced individual or small quantity O.E.M. buyers. We wish to recognize your technical expertice (and the accompanying reduction of support requirements) and to reward it with the lowest possible discount prices. **FULL WARRANTY PROTECTION IS ASSURED!!!**

Furthermore, we wish to discourage the novice from purchasing **ANY** computer equipment cross-country, from **whatever** source. We feel very strongly about the products that we sell, and (with both our appologies and regrets) encourage the first-time-user to purchase these products from their **LOCAL** dealer at his asking price . . . the premium is worth the added support that he can offer. If no local supplier is available, write or give us a call and one of our staff will be glad to offer assistance. Kindly we ask your cooperation in this matter, thank you.
— D. Scott Secor, General Manager

STORAGE . . . "EN MASSE"!

Konan board products, hard disk & tape backup subsystems represent the easiest to integrate and most cost-effective means to expand the storage capacity of your system . . . we add further incentive by discounting their prices 10-15%.

NEW! "David" subsystem - 5 Mb Shugart Tech. ST506 5¼" mini-hard disk in enclosure with P.S., SHUG-100 controller, cabling, etc. (Everybody will want to replace one of their mini-diskette drives with one of these little gems!) **($3550 list)** **$3199.00**

"Hardtape" system - Marksman hard disk drive, 17 Mb tape drive in enclosure with P.S., DAT-100 controller, cabling, etc.
40 Mb system **$7750.00** — 20 Mb system **$6999.00**

"TBS-100" 17 Mb tape backup system in enclosure with P.S., DAT-100 controller, cabling, etc. **$3125.00**

"MDS-100" Marksman disk system, DAT-100 controller, cabling, etc.
40 Mb system **$5599.00** — 20 Mb system **$4999.00**

"KNX-500" Western Dynex 10 Mb cartrige disk drive (5 fixed - 5 removable), KNX-500 controller, cabling, etc. **$5650.00**

"Enhancer" 9-track tape system with CDC 2400' tape unit, BIT-100 controller, cabling, etc. (800/1600 bpi) **$6999.00**

"SMC-100" CDC cartridge disk drive system (16 Mb removable), SMC-100 controller, cabling, etc. (the ultimate answer to your storage problems)
96 Mb system **$11250.00** — 32 Mb system **$7999.00**

SHUG-100 ST-506 disk controller	**($1250 list)**	**$1099.00**
SMC-100 SMD disk controller	**($1650 list)**	**$1399.00**
KNX-500 winchester disk controller	**($1695 list)**	**$1525.00**
BIT-100 9-track tape controller	**($1595 list)**	**$1399.00**
DAT-100 disk & tape controller	AVAILABLE ONLY IN SUBSYSTEMS	
OCTO-PLUS 8-port serial interface	**($795 list)**	**$699.00**
OMNI-PORT 16-port serial interface	**($1235 list)**	**$1099.00**

* * * * *

A.C.T. 5 Mb subsystem ST-506, controller, interface card & cable - specify S100, TRS-80 or Z89/H89 interface. (no case or P.S. included) **each $2750.00**

Vector UniStor single MOD II drive (for VIP - **$895 list**)	**$750.00**
Vector MicroStor dual MOD II drive subsystem	**$1499.00**
Vector DualStor dual 8" Qume (2 Mb) subsystem	**CALL**
Zenith Z77 dual mini-disk (204K) subsystem	**$1099.00**
Zenith Z47 dual 8" (2 Mb) disk subsystem	**($3695 list) $3199.00**

VECTOR GRAPHIC SYSTEMS

The best complete systems packages on the market! Each model includes CP/M 2.2, Microsoft BASIC, all CP/M utilities -plus- SCOPE screen editor (with much inbred W/P technology), RAID debugger (much like the best Development Systems'), ZSM assembler, 64K RAM, 80 x 24 memory-mapped video terminal (a must for rapid screen transfers), disk capacity as listed below, **very thorough** documentation (supplied in 3-ring binders), and more! Vector prices are being raised, effective 2/1/81 (Personally, we feel that they would remain the value leader at even higher prices, but don't let the factory know!)

	(LOCAL) Full Support	Qty. 1 O.E.M. "No frills"
Basic V.I.P. (315K)	**$3995.00**	**$3299.00**
V.I.P. Plus (630K)	**$4890.00**	**$3999.00**
ZIP (a BIS exclusive - 5 Mb)	**$7999.00**	**$6799.00**
Basic System B (630K)	**$5995.00**	**$4899.00**
(Time-share terminal + 64K RAM)	**$2200.00**	**$1750.00**
Basic System 2800 (2 Mb)	**$7995.00**	**$6499.00**
Basic System 3030 (32 Mb)	**$12995.00**	**$10499.00**

We will also custom integrate small business systems, development systems, word processing systems, etc. based upon any model Vector Graphic "Economy Sized Computer" that you may choose. Nobody can beat our integrated systems values! For example: a System B/Q (with Qume printer) add "Peachtree" business accounting & inventory management, Memorite III word processing, the standard software development package, and accessories (total retail value $15000). Our "no frills" price is **$8999** and our fully supported price is **$10499!** Other integrated Vector Graphic systems are priced **$4000** to **$250,000** (from 1 to 64 intelligent terminals). Call or write for a quotation on your particular requirements.

* * * * *

INTERSYSTEMS SPECIALS

The company which helped bring us the blessing of the IEEE, now allows us to offer a generous 15% off list for all mainframes ordered during January!

Front Panel System 2A FP (DPS-1 with 64KDR, MPU-80, VIO & FDC2)
Lists at $3595 — **SPECIAL $2999.00**

Front Paneless System 2A FPL (with cards listed above)
Lists at $3195 — **SPECIAL $2699.00**

Other systems may be custom integrated to your specifications - please call or write for a formal quotation.

SUPERIOR PRINTER VALUES

We will not stock printers that print ILLEGIBLE output (7-wire matrix printers) . . . all will support TRUE lower case decenders.

NEW! EPSON MX-80 9x9 matrix impact printer — Parallel **$575.00**
(80cps) 80/132 column — Serial, GPIB or TRS80 **$645.00**

Anadex 9500 Nx9 SUPER PRINTER! 132/176 column (150-200cps)
Serial, parallel & TTY interfaces ALL standard! **$1439.00**

C. Itoh Starwriter I (25-30cps) daisywheel — Parallel **$1699.00**
(same as Vista, Exidy, etc.) — Serial **$1899.00**

(Add $600 to either of the above prices for 45-50 cps versions)

Qume Sprint (55 cps) daisywheel printers — **CALL**

* * * * *

S100 BUS CIRCUIT CARD ASSEMBLIES:	Retail	OEM (1)
Intersystems DPS Front Panel card	495.00	445.00
Intersystems MPU-8000-1 CPU card	1395.00	1255.00
Intersystems MPU-8000-2 CPU card	1245.00	1120.00
Intersystems MPU-80 card	395.00	355.00
Intersystems 16KSR 16K Static (250ns)	495.00	445.00
Intersystems 64KDR 64K Dyn. RAM card	995.00	895.00
Intersystems FDC2 DD DMA disk cntrlr.	495.00	445.00
Intersystems VIO-1 (4P + 2S) I/O card	445.00	400.00
Intersystems VIO-0 Same w/o interrupts	395.00	355.00
Intersystems 6SIO Serial I/O card	895.00	805.00
Intersystems EP64 64K EPROM card	295.00	265.00
Intersystems EPB-M EPROM programmer	395.00	355.00
Intersystems ADDA 8-bit A/D & D/A card	495.00	445.00
Intersystems WW1 Prototyping card		35.00
Konan SMC-100 Hard Disk Controller	2195.00	1799.00
NEW! Vector Graphics "ZPB" SBC card	395.00	325.00
Vector Z80A CPU card	247.00	199.00
Vector Bitstreamer II card	270.00	225.00
Vector Motherboard, 18 slot, terminated	201.00	160.00
Vector Analog Interface Card	132.00	110.00
Vector Precision Analog Interface Card	449.00	350.00
Vector High Res. Graphics Card	270.00	225.00
Vector Video Digitizer Card	201.00	160.00
Vector Flashwriter II 80 x 24 Video	368.00	299.00
Vector PROM/RAM III Card	247.00	199.00
Vector 8K Static RAM Card	282.00	199.00
Vector 64K dyn. RAM Card	975.00	799.00
Micropolis Controller Card	518.00	425.00

CIRCUIT CARD SPECIALS

Entire line of Vector Graphic circuit cards	20% off
Entire line of Intersystems circuit cards	10% off
Other S100 and Multibus circuit cards available soon!	

MEDIA

Memorex 5¼" diskettes (10 disks per box)

SS/SD - **$26.50** box	**$250.00** 10 boxes	**$2350.00** 100 boxes
SS/DD - **$29.50** box	**$280.00** 10 boxes	**$2650.00** 100 boxes

Memorex 8" diskettes (10 disks per box)

SS/SD - **$32.50** box	**$310.00** 10 boxes	**$2950.00** 100 boxes
SS/DD - **$37.50** box	**$360.00** 10 boxes	**$3450.00** 100 boxes
DS/DD - **$46.00** box	**$450.00** 10 boxes	**$4250.00** 100 boxes

16 Mb #91204 CDC cart. disk pack	**$299.00** each	**$2799.00** qty. 10
5 Mb Western Dynex 5440 disk pack	**$110.00** each	**$999.00** qty. 10
2400' 9-track tape reel		**$18.00** each

* * * * *

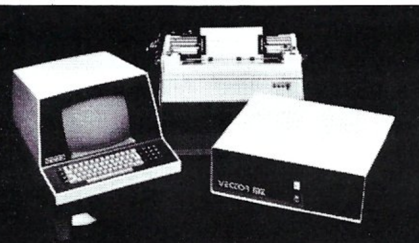
"HACKER SPECIALS" — (no returns please)

Take advantage of a trade-in on a Vector system: Northstar Horizon I double density 16K - looks new, runs well, but needs serial port chip . . . a "super rip" special at only **$999.00**

IBM model 1980 buffered "Selectric" style terminal & model 7441 control unit. About 7 years old - in use by a nationwide Financial Corp. until we bought them in December - quantities available. Options include split platen and many other goodies! (May be used as an intelligent TTY)

Good working condition — **$245.00** qty. 1
(All will be sent freight collect) — **$225.00** 2 up

Zenith data systems — CF&A — InterSystems (Ithaca Intersystems Inc.) — Anadex



Budget InfoSystems

Main offices located at:
1633 NE Highway Ten
Spring Lake Park, MN 55432
Order line open 11 AM to 6 PM
(612) 786-5545

Three 'New Generation' Z-80 CPUs Compared

by Bill Machrone

Much has happened in the S-100 field since the first Z-80 processor cards were introduced. Most significant is the adoption by most manufacturers of the proposed IEEE standard for the bus. The general availability of denser memories, shielded motherboards, octal bus drives and more sophisticated medium-scale ICs has made the designer's job easier, while more sophisitcation on the part of the software engineers has increased the efficiency and utility of systems.

More sophistication on the part of software engineers has increased the efficiency and utility of systems.

We will look at three of the currently available S-100 CPU cards that use the Z-80 microprocessor for its high speed and extended instruction set. I refer to them as "new generation" in that they were all introduced after the IEEE S-100/696 spec was released. I won't attempt to classify this as the second or third generation, just a new one. The three cards reviewed are the Godbout CPU-Z, the Intersystems Series II and the SSM CB2. This selection of manufacturers does not suggest that these are the only viable Z-80 processor cards to use; there just isn't room or time enough to review them all.

The three selected processor cards share several features: They are true CPU cards, meaning that they do not have I/O ports, significant system memory or disk controllers. They provide 24 pin sockets for system monitors and bootstraps, and they provide a means of controlling interrupts. Each implements some form of extended addressing. They provide bus and system controls such as power-on jump, jump on reset, generation of MWRITE, front panel status signals, wait states and clock speed.

From there, the differences begin.

Godbout CPU-2

The Godbout CPU is the newest entry to the Z-80 processor field. It is designed to replace that 8080 in your old system or to be the foundation for a new one, and it fully implements the IEEE S-100/696 specifications. Extended addressing is accomplished through OUT instructions to port 0FDH. A feature of the board is a vectored interrupt mask register, addressable through port 0FEH. It permits the programmer to select which of the S-100 VI0-7 lines will be able to reach the CPU at any point in time. This is a handy way to set the priority of interrupts, as each interrupt handler routine can begin with a mask instruction that determines what, if anything, is allowed to interrupt the routine being executed.

A lot of design time went into the implementation of the on-board ROM sockets. They are designed for two 2716's. 2732's can be accommodated with a few (well-documented) trace cuts and jumpers. Both the socket address and the jump address are selectable via dip-switches. Another switch decides whether the sockets will be present in all the extended pages of memory or just in the base 64K. The bus drivers float when addressing the ROM, so that it co-exists nicely with overlapping memory —without need for the PHANTOM line or an I/O port to disable it. This implementation also gets along well with bootstraps that grab the bus, such as the Tarbell double density controller. With a system monitor ROM in the sockets, the CPU never sees the bootstrap. Reset takes you into your monitor. With the sockets disabled, the bootstrap functions normally.

Bill Machrone, P.O. Box 291, Fanwood, NJ 07023.

We were able to use several different copies of this board during the review period. Two worked flawlessly. One ceased to function if there were two ROMS in the sockets, but functioned normally if just one was present. One copy of the board was reported to (I didn't examine it personally) exhibit sensitivity to the type of memory board present at the bottom of memory. My observation of the two boards still running is that they tend to be "kind" to memory; bus timing is generous enough to permit slower, possibly marginal memories to operate at or near full speed. The Godbout board is available in three versions: "unkit" (sockets and capacitors soldered at the factory), assembled and CSC, which is a combination of burn-in, performance test, priority factory service and extended warranty. The CSC version of the board is shipped with a 6 MHz Z-80B and a 24 MHz crystal which divides down to a 6 MHz clock. Other users of the board report that substitution of the crystal and Z-80B on a normal assembled board gets it right up there at warp factor six. You doubled your processing speed when you went from 2 to 4 MHz. Are you ready for another 50 percent increase? It should be noted that 6 MHz operation is not to IEEE spec since the bus clock line runs at 3 MHz instead of the specified 2 MHz.

Timing is generous enough to permit slower, possibly marginal memories to operate at or near full speed.

A trival thing can alter one's entire perception of a product. Aside from the fact that this board came up running the first time I plugged it in, the thing that sold me on it was the inclusion of card eject levers. In my opinion, they should be part of the physical specifications for S-100 boards. When you're playing around with options and swapping ROMs and the like they're worth their weight in gold. Furthermore, the quality of the sockets used on the boards was the best of the group.

The documentation provided, while adequate, is the skimpiest of the lot.

Ithaca Intersystems' Series II

Ithaca Intersystems' new Z80 is actually almost a year old, and has some interesting features worth studying. It only has one ROM socket, but it will accept anything from a 2708 to a 2732 and can be addressed on any 1K boundary in the address space. Like the Godbout board, it handles interrupts, but uses the sophisticated AMD vectored interrupt controller. In addition to masking it permits setting priorities, rotation and polling. The extended addressing is not IEEE standard in that only four additional address lines are provided out of the eight specified. Thus you can address the first megabyte of the IEEE 16 Mbyte address space, sufficient for most purposes. The unique thing about Intersystems' implementation is that the extended address bits are controlled by two four-bit registers. This, in conjunction with on-board address decoding, provides either one 8K window or two 4K windows into the extended address space. The approach, reminiscent of that used on some DEC LSI-11 implementations, is good for executing programs with small overlay sections. It is not suitable for "banking" multiple 64K pages nor for running large programs in alternate pages, unless specific calls to a kernel are made to supervise address translation. Detailed programming examples are given in the manual.

The manual also goes into extreme detail on bus timing and control signals. The board permits selection of either the partial or fully latched modes of operation, with fully latched mode being preferable for high-reliability or electrically noisy environments. Fully latched mode requires memory with a worst-case access time of 160 nanoseconds or the inclusion of a wait state. The card also has the necessary circuitry to properly manage a DMA transfer on the bus by a temporary master. The objective is to provide the control circuitry on the permanent bus master so that it need not be duplicated by other cards in the system.

Intersystems boards have a reputation for working best in the DPS-1 mainframe with front panel and with other Intersystems products. In actual use, the two copies of the Series II card did not quite live up to the expectations generated by the manual. Neither board worked properly at 4 MHz in an unterminated 8-slot mother board. Neither of them liked the terminated environment in my TEI mainframe, since they would only run while on an old unshielded extender card. They would not run when plugged directly into the bus. There was some conflict between the Tarbell controller bootstrap and the jump on the reset circuitry on the CPU. It was a tossup to see would get the system.

Once running, however, the processor was quite reliable. DMA transfers by the Tarbell board were handled without incident. The CPU did have a distressing tendency to want to keep on running when I had something else in mind, as it largely ignored the reset button. When this happened, the only recourse was to shut it down and regain control through the power on jump. With the memory I was using (Godbout 200 nsec) there was no discernible difference in operation between partially latched and fully latched mode.

SSM Microcomputer Products' CB-2

The SSM processor is truly remarkable. It tries, and nearly succeeds, to be all things to all people. Like other SSM products, jumper options abound. No matter how non-standard your S-100 system may be, there is a very good chance that the SSM board will run in it and make it better at the same time. It came up and ran in my systems with the same alacrity as the Godbout board and exhibited no quirks—with the exception of an occasional dubious response to reset in a 20-slot TEI box. It probably needed a slightly larger capacitor to trigger the circuit properly.

In addition to full extended addressing, the SSM board provides all the other IEEE signals necessary to get along with other state of the art components. But SSM obviously remembers their roots; you can easily get the board up to be compatible with just about anything back to an Altair 8800b. This is accomplished through wire-wrap jumpers. In fact, there are so many of them that areas of the board resemble a bed of nails for

Z80 CPUs cont'd. . .

a miniature swami. Most of the selections you want to make are in 0.10" centers, so you can use shorting plugs instead of wrapping. SSM should (but doesn't) provide a half dozen or so with the board to aid in initial jumpering.

The manual, like so many others, is labeled "preliminary." The difference between this one and all the others is the wealth of detail. The manual could almost be sold separately from the board. It is virtually a living history of the S-100 bus, describing old and new signals in clearly written text and illustrating them with timing diagrams. Granted, the diagrams are hand-drawn, but they're neat and they explain what needs to be explained.

The board has two paddle switches sticking out of the top edge. They perform run/stop and single step for bus cycle-level debugging without a front panel. The board also has an antenna-like header that emanates from a point cryptically labeled J1. There is no reference to J1 in the manual, its only oversight.

The detail and the nearly overwhelming number of options make this board a bit more difficult to set up than most. It does not come with the IEEE options already selected, but the standard jumperings are clearly indicated in the manual. Once it was set up, however, it ran reliably and without any fuss.

Conclusions

In summary, these boards represent an advance in the sophisitication of the S-100 bus. Each has its own personality and each is a viable choice in its own right. I recommend the Godbout board if you expect to do a minimum of hardware hacking. The implementation of the ROM disable is worth the price of admission, expecially if you intend to go beyond a single 64K page of memory. Two ROM sockets will prove their value once the price of the 2K X 8 RAMS comes down. I don't know of anyone else at this writing who is offering 6 MHz operation either.

The SSM CB2 is obviously the bit-fiddler's delight. Even the extended address lines come out through a dip header before going to the bus, so you can mix them around to suit old bank select boards and the like. It is also the one to have if your system is old or non-standard and you want a strong cornerstone for entry into the current S-100 world. Or maybe you simply disagree with those guys on the IEEE standards committee and you want to do it your own way.

The Intersystems Series II gave me some hardware hassles, but I know it's a tiger in an environment that it likes. Combined with the front panel in the DPS-1 mainframe and a scope, it is the next best thing to a commercial signature analyzer. It has two test modes that facilitate hardware checkout and debugging. And I dearly love that interrupt controller.

So there you have it; three of the new breed of CPU boards. I can hardly wait for the next generation. □

Godbout Electronics, Building 725, Oakland Airport, CA 94614; (415) 562-0636.

Ithaca Intersystems, Inc., 1650 Hanshaw Road/Box 91, Ithaca, NY 14850; (607) 257-0190.

SSM Microcomputer Products, 2190 Paragon Drive, San Jose, CA 95131; (408) 946-7400.

The SSM VB3 Video Board

by Jon Bondy

The VB3 video board is a 'third-generation' memory-mapped character-oriented video board made by SSM Microcomputer Products. Unlike the first generation (16 by 64) and second generation (24 by 80) display boards, the VB3 offers a dense display (up to 51 lines by 80 characters) with a variety of character 'attributes' such as low intensity, reverse video, underscore, blinking, user-defined characters and graphics characters. In addition, unlike most of the previous generation boards, the VB3 provides a blank select option whereby the refresh memory can be removed from the address space of the computer when not in use (even in-between writing each character to the screen), allowing almost all of the computer's 64K byte address space to be used without interference.

The VB3 comes with a thick (¼ inch) manual which covers a lot of material, including unpacking/assembly instructions, board setup, theory of operation, schematics and board layout, a short section on trouble-shooting, and voluminous listings of I/O drivers. Although I did not build the kit, I did look over the assembly instructions and they appear to take the builder through the process step-by-step, with more warnings and comments than are usually found in such manuals. Board checkout is a staged process, with each section of the board being tested before the IC's for the next stage are inserted. The VB3 uses the SMC CRT5037 and CRT8002 ICs which are programmable video display chips, and the SSM manual discusses how to use their flexibility for a variety of purposes. One of the chief features of the VB3 is the access which the user gets to these programmable IC's.

The VB3 board contains sockets for 8K bytes of memory, of which only 4K bytes were installed on my board. This allowed for storage of 2048 characters (enough for an 80 by 24 character display) and 2048 attribute bytes (to determine if each character is to be displayed normally or with reverse video or with underlining or the like). If one were interested in using the VB3 to display 51 lines by 80 columns, one could install the additional 4K bytes of display memory.

Jon Bondy, Box 148, Ardmore, PA 19003.

In order to use the board properly, one first must initialize the attribute memory for normal display and then clear the character memory to all blanks. One might think that the default value of the attribute memory for 'normal' display would be zero, but SSM implemented the hardware so that the value '3' must be used instead. There are two ways to set the board up, either as a normal memory board (so that it is always in the address space of the computer) or with the bank select option enabled so that the memory on the board can be removed from or brought back to the address space of the computer on command. I started running the board in the former mode for simplicity, and then tried to use the more complex memory management feature later on.

The VB3 video board is a 'third-generation' memory-mapped character-oriented video board.

The VB3 allows the user to configure it in a variety of ways. The user can determine whether the board uses horizontal and vertical sync signals generated on the board itself, or whether it uses externally supplied signals. One can change the character width in a range from 6 to 16 dots, and one can address the VB3's on-board keyboard ports to any port address pairs in the 256 port range. One can instruct the VB3 to latch the S-100 bus address lines during each bus cycle, or to accept the bus data directly (through receiver buffers, of course) without latching, by setting some jumpers. One can set the display memory address to start at any multiple of 8K bytes, and one can set the CRT5037 control register port addresses to start at any multiple of 16. Finally, the bank select option can be enabled or disabled.

The VB3 which I reviewed was sent to MICROSYSTEMS by SSM already assembled, so I did not get a chance to inspect it as closely as I might have had I performed the assembly myself. Upon inspecting the board, however, I noticed a few jumper wires and cut wire runs on the back side, indicating that some sort of bugs in the board layout had been fixed at the factory; I found no documentation on the changes in the manual.

The first thing I tried to do with it was to plug it in and use it simply as a memory board. The board is densely packed, and there is a voltage regulator heat sink at each edge which got in the way of my card guides, making it difficult to insert into my computer; for testing purposes, I inserted the VB3 into an extender board. When I powered up my computer to test the VB3 as a RAM board, my ROM monitor would not work; when the VB3 was removed, and the monitor began to work again. The VB3 was interfering with the rest of my system in some way.

It turned out that the factory 'assembled and tested' unit had a jumper wire missing; it was clearly described in the manual, but I had assumed that the unit would work as delivered, without changing jumpers. With the jumper applied and the various switches correctly set up, I tried again. My ROM monitor worked and so did the VB3, at least as a memory board. It passed my memory tests perfectly, but nothing appeared on the screen.

After some thought I realized that the flexibility which the programmable CRT5037 IC offered was also going to be an inconvenience, since the board was not going to do a thing until I 'programmed' it; and I would have to program it each time I reset my computer. I had encountered a similar situation when I purchased and modified the S.D. Sales VB-8024 (see MICROSYSTEMS, Vol 1 No 1, p. 11), so once I realized that the VB3 was not a simple plug in board like its first- and second-generation predecessors, I began to read the I/O driver listings to see what I had to do.

I keyed in the initialization routine which I found in the manual (about 40 bytes) and ran it, and the screen filled with the garbage which was resident in the memory after it powered up. I filled the attribute memory with '3's, the value which causes the board to display normal characters, and then ran a memory test on the character storage portion of the board. Patterns began to race across the screen as the memory test progressed. Unfortunately, although the patterns were correct, the characters themselves seemed to be shimmering, as if black 'snow' were all over the picture. Deciding that the shimmering was a minor problem, I proceeded to try to use the board with its bank select feature. I never managed to solve the shimmering problem, though, and the characters remained only marginally legible.

The tests which I had run up until now were simple ones run from my ROM monitor, but they were laborious, since I had to key them in each time I powered the computer down. I wanted to continue using the VB3 with my UCSD Pascal system, but to do that I needed all 64KB of my computer as RAM (not as a video board); I needed the bank select feature to work. Unfortunately, for quite a while I could not get that feature to work at all. The VB3 moves the on-board memory into the computer's address space when the keyboard data port is written to (since under normal circumstances this would never happen), and removes it from the address space when the keyboard status port is written to. After much muttering and peering about with my scope, it turned out that the VB3 would not accept just any old port number for the keyboard port value, but if I used a port address of 0FEH, it would work and the on-board memory could be moved into and out of my computer's memory at will. To test it, I had written a short Pascal program, which follows this article, and it finally began to work. In order to get it to do so, however, I had to play with the location where I placed the display memory, since it could neither lie on top of the Pascal program nor the p-machine interpreter. This took some finagling. Unfortunately, it did not work consistently, and it is not clear whether the problem is in the VB3 or in my computer system.

```
program ssmvb3test;
  const
    vtac = 208; { video chip control registers -- 0D0H }
    kbddata = 254; { 0FEH -- write to remove VB3 from address space }
    kbdstat = 255; { 0FFH -- write to put VB3 in address space }
    video = 16384; { 04000H -- address of video ram }
  var
    i : integer; ch : char;

  procedure portwrite( addr : integer; data : integer ); external;
  procedure memwrite( addr : integer; data : integer ); external;

  begin
  portwrite( vtac+14,0 ); { reset video chip }
  portwrite( vtac+10,0 );
  portwrite( vtac,103 ); { character times per line }
  portwrite( vtac+1,188 ); { interlace and synch pulse time information }
  portwrite( vtac+2,109 ); { scans per row and characters per scan }
  portwrite( vtac+3,23 ); { skew bits and data rows per frame }
  portwrite( vtac+4,6 ); { scans per frame }
  portwrite( vtac+5,49 ); { vertical margin }
  portwrite( vtac+6,23 ); { row address of last row on screen (scrolling) }
  portwrite( vtac+14,23 ); { restart timing chain }

  portwrite( kbdstat,0 ); { place VB3 in address space }
  { fill attribute memory with '3' }
  for i := 1 to 2048 do
    memwrite( video + 4096 + i,3 );
  { fill video memory with stuff }
  for i := 1 to 2048 do
    memwrite( video + i,i );
  portwrite( kbddata,0 ); { remove VB3 from address space }
  end.
```

SSM VB3 cont'd. . .

Despite the fact that I have not yet gotten the VB3 to work satisfactorily, I think that I can make some comments about it. My initial reaction to the board (even before I received it) was that it was unwise to put the character memory in the computer's address space, since at this point in the development of S-100 bus computers the software requires more memory resources than it did only a few years ago. SSM solved that problem by allowing the user to remove the VB3 from the address space of the computer except for the times when things are actually being moved into that memory. This allows boards like the VB3 to be used with software systems like UCSD Pascal, as long as the feature works reliably. I have every confidence that my problems with this feature were due to my computer system rather than to a problem with the SSM design.

When I agreed to review the VB3, I had figured that it was just another memory-mapped video board, and that with an afternoon of playing with it I could make some useful observations about it. Unfortunately, it's too smart. I couldn't just plug it in and test it; rather, I had to program it and coddle it in order even to see a few characters on the screen. The complexity of the board is such that one can't just use it for video output. You must initialize it carefully, and then provide a non-trivial driver program to use it.

If you have a requirement for more than 24 lines of text or for a 'terminal' with complex features which you cannot find elsewhere, you should consider the VB3.

If you compare the VB3 with the S.D. Sales VB-8024, you discover that the two boards have similar capabilities (although the VB3 has more memory, more attributes, and more flexibility). The chief differences lie in the fact that the VB-8024 has its own on-board processor and driver ROM; you turn it on and use it, simply and with no tricks or problems. The on-board processor on the VB-8024 also makes complex manipulations of screen data less of a processing burden for the main processor. The VB3 offers more flexibility and capability while at the same time requiring more care on the part of the user. The VB-8024 is a 'black box' which the user plugs in and uses, while the VB3 is a complex but capable peripheral which the user must customize carefully.

In summary, if you have a requirement for more than 24 lines of text or for a 'terminal' with complex features which you cannot find elsewhere, you should consider purchasing the VB3. I think that you will find the VB3 to be more complex and troublesome than it is worth if your application is a simple 80 by 24 terminal; in this case, the VB-8024 will give you the same functions for the same price and with a lot less software headaches.

The SSM VB3 is $375 (kit) and $440 (assembled) and is manufactured by SSM Microcomputer Products, 2190 Paragon Drive, San Jose, CA 95131. □

Choosing Between CRT Output and Printer Output

by Bob Kowitt

Some versions of Basic allow you to specify while running your program whether you want to output to your CRT terminal or to your printer. Unfortunately, one of the most widely used and powerful Basics, Microsoft Basic, does not. If you use the methods proposed in the user's manual, you are told to use the command PRINT to go to the CRT terminal and the command LPRINT when you want to output to your printer.

There is, however, a way you can bypass this deficiency if you are using Microsoft Basic Rel. 5.0 or later, under CP/M. Locations 0000,0001, and 0002 contain the jump to the BIOS in CP/M. Microsoft Basic uses the data stored at these locations to direct your output as you have chosen with the commands PRINT or LPRINT in your program. Using this same information, you can locate the point in memory that contains your routine to write to the terminal or to the printer.

You can bypass the use of LPRINT by fooling the Microsoft interpreter. In Microsoft BASIC 5.0 and higher, this data is stored at a location between 16000 and 18000 (decimal), depending on which release you are using. The location changed during modification of Microsoft Basic to eliminate bugs that were discovered after the original release. By including within your program the following routine, you can at any time within your program direct the output in either direction at runtime rather than being forced to duplicate the code when writing your program. You cannot poke the data directly into the jump table of CP/M because Microsoft Basic does not use this jump table after finding its location.

Lines 60 to 100 define your variables and prepare your program for further input during your program.

Poke F, OT (line 160) should be inserted before each point at which you may want to change the output. Poke F,C (line 180) should be inserted to get output back to your CRT terminal. You must put a copy of line 180 at the end of your program. If you don't, you will be locked into your printer and not your CRT at the end of the program. Your keyboard will still be entering data to your computer but there will be output to the printer and not the CRT. □

Bob Kowitt, 1727 N. Jerusalem Rd., East Meadow, NY 11554.

```
10       '          **************
20       '          SAMPLE PROGRAM
25       '          **************
26       '
27       'To operate printer, fill F with PRINTBYTE
30       'To operate console, fill F with CONSOLEBYTE
31 BIOSBOTTOM=(PEEK(2)*256)+PEEK(1)
32 PRNTBYTELOC=BIOSBOTTOM+13 : CONSOLEBYTELOC=BIOSBOTTOM+10
33 PRNTBYTE=PEEK(PRNTBYTELOC) : CONSOLEBYTE=PEEK(CONSOLEBYTELOC)
34 C=CONSOLEBYTE        '        F is location with MBASIC that directs
35 FOR I=16600 TO 18000
36 IF PEEK(I)=CONSOLEBYTE AND PEEK(I+1)=PEEK(CONSOLEBYTELOC+1) THEN 38
37 NEXT
38       '
110      '
120 INPUT "Do you want P(rinter)  or C(onsole)  ";CHOICE$
130 OT=C
140 IF LEFT$(CHOICE$,1)="P" THEN OT=PRNTBYTE
150      '
160 POKE F,OT
170 PRINT"This is a demonstration of print output selection"
180 POKE F,C
190 END
```

Should you get trapped in printer mode, simply type:

POKE F,C (cr)

to regain control and printout at the console.

Dot Graphics on the IMSAI-VIO

by Gary Sabot

Run TRS-80, Apple or PET graphics programs on your IMSAI-VIO and similar video boards with this program.

Several of the personal computers in wide use today are capable of displaying low resolution graphics. This means that they are able to display dots, lines, pictures, or even animated characters on their screens. The TRS-80 and the Apple have this capability. Because of the widespread use of these two computers, there are many programs available which make use of their graphics capability. This article presents a program that will enable owners of the Imsai VIO (or similar memory-mapped displays, such as the Polymorphics VTI) to utilize these programs. By making a few simple changes, *all* TRS-80 graphics programs, most Apple graphics programs (those programs which do not make extensive use of color), and some PET graphics programs will run on your machine.

The Imasi VIO is capable of displaying special "graphics characters." (See Figure 1.) Each graphics character contains six squares. By using the proper graphics character, it is possible to turn each of these six squares on (white), or off (black) independently. The problem is: how can a single square be turned on, without disturbing the five squares that surround it?

My solution to this problem is in the form of a machine language program. (See listing #1.) It allows a Basic program to quickly and easily plot points using the VIO. It can be modified to work with other memory-mapped display boards, such as the Polymorphics VTI. The program is designed to be used in conjunction with Microsoft Basic and CP/M. (Of course, it can also be used by a machine language program.)

To plot a point, the proper graphics character must be selected; then this character must be placed in the correct memory location. If this process were to be implemented as a Basic program, it would take approximately one-half second to plot each point. If a program that plots several hundred points were run, however, those one-half seconds would add up, delaying the program. I have implemented the plotting program in machine language, because a machine language program is considerably faster than an equivalent program written in Basic. If a Basic program needs to turn a certain square "on," it simply passes its X-Y coordinates to this routine (see Figure 2) and calls it, using the USR function. The routine then turns the square "on," and subsequently returns to the Basic program. Analagous procedures may be used to determine the square's present color (black or white), or to turn it off (black).

Gary Sabot, 38 Woodland Rd., Roslyn, NY 11576.

Utilizing The Program

If you have a 30K CP/M system using the Imsai VIO, you can employ the program just as I assembled it. To use the routine (after you have POKE'd it into memory—see the Basic listing), first POKE the Y coordinate of the desired pixel into location 6889H, then POKE the X coordinate into 688AH, and POKE the function number into 688BH. The function number would be a 1 to set the pixel white. This is equivalent to the TRS-80's SET(X,Y) command. The function number would be a 0 to set the pixel black. This is identical to the TRS-80's RESET(X,Y) command. If the function number is a two, the plotting routine will determine the present status of the pixel, without disturbing it. This is similar to the TRS-80's POINT(X,Y) command. The status of the pixel is retrieved by a PEEK to 688BH. A 0 will be found there if

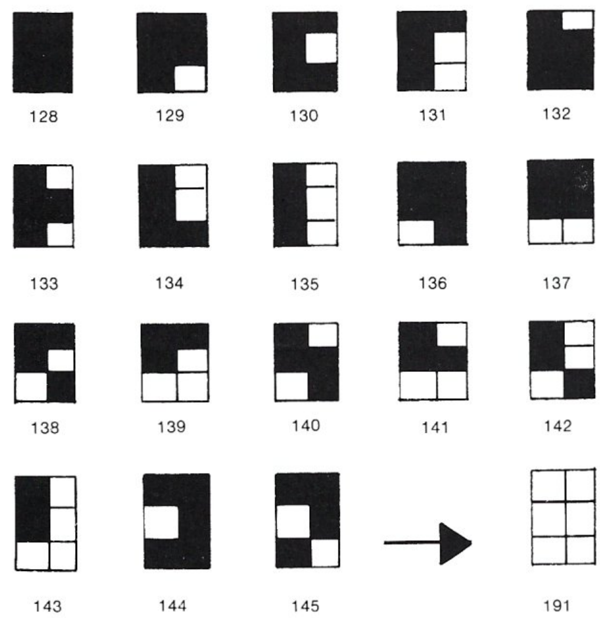


Figure 1. Some of the Imsai VIO's "graphic characters." The number of the desired character can be placed in the VIO's refresh memory; subsequently, the character will appear on the screen.

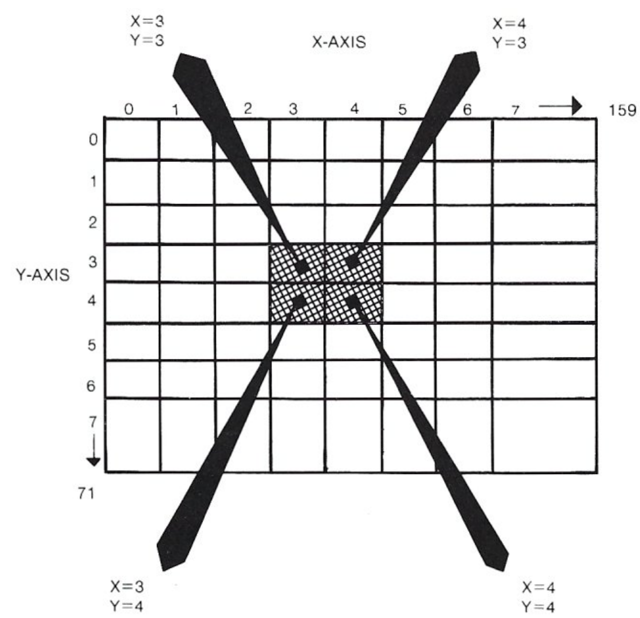


Figure 2. This is the coordinate system used to address the squares, or pixels, on the screen. It is the same as the system used by the TRS-80, except that the VIO's display is larger (160 by 72, compared with the TRS-80's 128 by 48).

the pixel is black and a 1 will be found there if the pixel is white. Once the proper values for the X,Y coordinates, and the function number have been POKED into memory, jump to 6800H using the USR function to execute the plotting routine.

I have provided a sample Basic program which uses my point plotting routine. (See listing #2.) It is a modification of program on page 33 of *Introduction to Low Resolution Graphics*, by Nat Wadsworth. The program draws lines between random points on the screen.

This plotting routine enables you to adapt to your computer the many TRS-80, Apple, and PET programs employing graphics which are in the public domain.

Because the plotting program is written in machine language, it is necessary to reserve memory for it when MBasic is run. To do this, instead of simply typing "MBasic" to run MBasic, type "MBasic /M:&H67FF". This sets 67FF as the highest address available for use by MBasic. The preface "&H" indicates that a hex number will follow.

If your system is larger than 30K, you might want to reassemble the routine at a higher address, in order to fully utilize your memory. For every kilobyte of memory that you have above 30K, add 400H to 67FFH. Then reassemble the routine at the resulting address. Finally, substitute the new origin for the 67FFH in "MBasic /M:&H67FF". Of course, the locations for POKING and PEEKING change each time a different version of this program is assembled.

If you are using a video board other than the VIO, you will probably have to reassemble the program, making one or more of the following changes (most involve the "EQUates" in the beginning of the program):

1) If your video board is addressed at a location other than 61440 (F000H), change the value in the line beginning "SCREENAD EQU..." in the listing to the correct screen address.

2) If the line length of your display is different than 80 characters per line, change the line length in the listing ("LINE EQU...") to the proper value, and reassemble it.

3) If a black pixel is represented by a 1, and not a 0, substitute 1 for 0 in the listing where it now reads "BLK EQU 0".

4) Determine the value of a blank (all black) character cell and substitute this for the 80H in the line "BLKCHR EQU 80H".

5) Find out what the proper CHRAND and CHRCPI values for your display should be (refer to the comments in the program listing) and insert the correct values into the program.

6) If the progression from black to white on your video board is different than that shown in Figure 1, you will have to modify the portion of the program (6843H thru 6863H) that calculates the bit mask. This can be a very involved process!

Now that your computer has dot graphics capability, make it work for you. Programs can be written with output in the form of a graph, instead of in numbers and letters. Programs can even be designed to imitate arcade games. One of the most exciting possibilities of this plotting routine is that it enables you to adapt to your computer the many TRS-80, Apple, and PET programs employing graphics which are in public domain. □

—PROGRAM BEGINS NEXT PAGE—

POINT PLOT LISTING

LISTING #1

```
                    ;******* POINT PLOT ROUTINE FOR MEMORY MAPPED DISPLAYS *******
                    ;******** BY GARY SABOT 11/27/79 ********
F000 =              SCREENAD        EQU     61440   ;ADDRESS OF VIDEO BOARD
0050 =              LINE            EQU     80      ;LINE LENGTH
0000 =              BLK             EQU     0       ;BIT THAT REPRESENTS A
                                                    ;PIXEL THAT IS "OFF" (BLACK)
0080 =              BLKCHR          EQU     80H     ;CONTENTS OF A BLANK (BLACK)
00C0 =              CHRAND          EQU     0C0H
0080 =              CHRCPI          EQU     80H
                    ;CHRAND IS "ANDED" WITH THE CONTENTS OF A CHARACTER CELL.  IF
                    ;THE CELL CONTAINS A VALID GRAPHICS CHARACTER (NON-ALPHABETIC)
                    ;THE RESULT SHOULD EQUAL CHRCPI.  IF IT DOES NOT, A BLANK
                    ;CHARACTER WILL BE PLACED IN THE CELL.
6800                        ORG     6800H
                    ;
                    ;DATA SHOULD BE STORED IN FORM Y, X, FUNCTION #
                    ;FUNCTION #-- 0 MEANS SET BLACK, 1 MEANS SET WHITE,
                    ;ALL ELSE MEANS RETURN DOT STATUS (0=BLACK, 1=WHITE)
                    ;
6800 218968                 LXI     H,DATA          ;SET POINTER TO DATA
6803 0E00                   MVI     C,0             ;LOAD C WITH ZERO
6805 7E                     MOV     A,M             ;LOAD Y INTO A
6806 D603           DIV:    SUI     3               ;DIVIDE Y BY 3,
                                                    ;PUT RESULT IN C
6808 DA1068                 JC      DIVX
680B 0C                     INR     C
680C A7                     ANA     A
680D C20668                 JNZ     DIV
6810 328C68         DIVX:   STA     YREM            ;SAVE REMAINDER FROM
                                                    ;DIVISION OF Y
6813 23                     INX     H               ;READ IN VALUE OF X
6814 7E                     MOV     A,M
6815 0F                     RRC                     ;DIVIDE X BY 2
6816 E67F                   ANI     7FH             ;STRIP OFF FIRST BIT
6818 2100F0                 LXI     H,SCREENAD      ;LOAD ADDRESS OF VIDEO BOARD
681B 5F                     MOV     E,A             ;CALCULATE CHARACTER'S
                                                    ;LOCATION IN MEMORY
681C 1600                   MVI     D,0
681E 19                     DAD     D               ;ADD X TO BASE ADDRESS
681F 115000                 LXI     D,LINE          ;PREPARE TO MULTIPLY
6822 79                     MOV     A,C
6823 A7                     ANA     A
6824 CA3068                 JZ      ADFND
6827 3D             MULT:   DCR     A               ;ADD Y*LINE LENGTH TO
                                                    ;BASE ADDRESS
6828 CA2F68                 JZ      ONEMOR          ;PUT ADDRESS IN HL
682B 19                     DAD     D
682C C32768                 JMP     MULT
682F 19             ONEMOR: DAD     D
                    ;
                    ADFND:
                    ;FIRST CHECK IF A GRAPHICS CHARACTER IS ALREADY IN CHARACTER
                    ;CELL.  IF NOT, STORE A BLACK CHARACTER THERE.
                    ;THEN CALCULATE BIT MASK
6830 7E                     MOV     A,M             ;LOAD CHARACTER
6831 E6C0                   ANI     CHRAND          ;CHECK IF GRAPHICS CHARACTER
6833 FE80                   CPI     CHRCPI
6835 CA3B68                 JZ      FNDBIT          ;YES, NOW CONTINUE
6838 3E80                   MVI     A,BLKCHR        ;NO, STORE AN ALL
```

```
                                                    ;BLACK CHARACTER
683A 77                   MOV     M,A
683B 3A8A68      FNDBIT:  LDA     DATA+1            ;LOAD VALUE OF X
683E E601                 ANI     1                 ;GET REMAINER FROM
                                                    ;DIVISION BY 2
6840 CA4868               JZ      XRZER             ;REMAINDER IS ZERO
6843 3E01                 MVI     A,1               ;BEGIN TO FORM BIT MASK IN A
6845 C34A68               JMP     CONT              ;CONTINUE
6848 3E08        XRZER:   MVI     A,8
684A 47          CONT:    MOV     B,A               ;SAVE PARTIALLY FORMED MASK
684B 3A8C68               LDA     YREM              ;LOAD REMAINER FROM
                                                    ;DIVISION OF Y
684E A7                   ANA     A                 ;SET FLAGS
684F CA6068               JZ      SHIFT2            ;NEEDS TO BE SHIFTED TWICE
6852 3C                   INR     A
6853 CA6468               JZ      MSKFND            ;DOES NOT NEED TO BE SHIFTED
6856 3C                   INR     A
6857 C26068               JNZ     SHIFT2            ;MUST BE SHIFTED TWICE
685A 78          SHIFT1:  MOV     A,B               ;LOAD BIT MASK
685B 07                   RLC                       ;SHIFT LEFT ONCE
685C 47                   MOV     B,A               ;SAVE IN B
685D C36468               JMP     MSKFND
6860 78          SHIFT2:  MOV     A,B               ;LOAD BIT MASK
6861 0707                 RLC ! RLC                 ;SHIFT LEFT TWICE
6863 47                   MOV     B,A               ;SAVE IN B
                 ;
                 ;THE BIT MASK IS IN B AND THE CHARACTER ADDRESS IS IN HL.
                 ;NOW COMPUTE AND PUT PROPER VALUE ON THE SCREEN.
                 MSKFND:
6864 3A8B68               LDA     DATA+2            ;GET FUNCTION #
6867 A7                   ANA     A
6868 CA7E68               JZ      SETBLK            ;0 MEANS SET DOT BLACK
686B 3D                   DCR     A
686C CA8568               JZ      SETWHT            ;1 MEANS SET DOT WHITE
                 ;
                 ;ASSUME CALLER WANTS DOT STATUS
                 ;
686F 7E                   MOV     A,M               ;LOAD CHARACTER CELL
6870 A0                   ANA     B                 ;AND WITH BIT MASK
                          IF      BLK
                          JNZ     RETBLK
                          ENDIF
                          IF      NOT BLK
6871 CA7A68               JZ      RETBLK
                          ENDIF
6874 3E01                 MVI     A,1               ;RETURN A 1 FOR "WHITE"
6876 328B68               STA     DATA+2
6879 C9                   RET
687A 328B68      RETBLK:  STA     DATA+2            ;RETURN A 0 FOR "BLACK"
687D C9                   RET


                 ;THE FOLLOWING ROUTINES ALL LOAD THE CHARACTER CELL,
                 ;MODIFY IT IN THE DESIRED WAY, SAVE IT, AND RETURN.
                 ;
                 ;SET DOT BLACK
                 ;
                 SETBLK:
                          IF      BLK               ;CONDITIONAL ASSEMBLY
                                                    ;WHEN BLACK=1
                          MOV     A,M
                          ORA     B
                          MOV     M,A
                          RET
```

Dot Graphics con't...

```
                              ENDIF
                ;
                              IF        NOT BLK
687E 78                       MOV       A,B
687F 2F                       CMA
6880 47                       MOV       B,A
6881 7E                       MOV       A,M
6882 A0                       ANA       B
6883 77                       MOV       M,A
6884 C9                       RET
                              ENDIF
                ;
                ;SET DOT WHITE
                ;
                SETWHT:
                              IF        BLK
                              MOV       A,B
                              CMA
                              MOV       B,A
                              MOV       A,M
                              ANA       B
                              MOV       M,A
                              RET
                              ENDIF
                ;
                              IF        NOT BLK
6885 7E                       MOV       A,M
6886 B0                       ORA       B
6887 77                       MOV       M,A
6888 C9                       RET
                              ENDIF
6889            DATA          DS        3              ;STORAGE FOR Y,X,
                                                       ;AND FUNCTION NUMBER
688C            YREM          DS        1              ;STORAGE FOR REMAINER OF Y/3
688D                          END       6800H
```

An Ultra-Simple Index For Sequential Files

by Fred Gohlke

File indexing routines are generally complex. There are applications, however, where comprehensive file indexing is overkill. These are characterized by sequential structure, relatively static size, slow access time, and a programmer who'd like to access his data faster without programmed or purchased complexity.

The approach suggested here covers such cases. It isn't intended as a definitive solution to file indexing problems, but as an aid to finding data in certain types of files. It requires that the files be maintained in alphabetical order and that they be randomly accessible. It doesn't index an individual record directly, instead it finds a small group of records which contain the record sought. The fact that the files must be maintained alphabetically means that adding and deleting records is slow. So, it works best when such changes are infrequent, as in name and address, inventory, and similar files which are frequently read and/or updated.

The method is to maintain the number of entries in the file beginning with each letter of the alphabet, and to use these amounts to find the first entry in the file beginning with the first letter of the entry sought.

Implementation will vary with the language and dialect used. The following description uses CBASIC statements and assumes:

1 - A randomly accessible alphabetized file called STOCK.FIL.
2 - A randomly accessible file called INDEX.FIL containing the number of entries in STOCK.FIL for each letter of the alphabet.
3 - The data on which STOCK.FIL is alphabetized are the leftmost characters of each record.
4 - The key for the record sought is in a variable called KEY.DATA.
5 - The length of the key is in a variable called KEY.SIZE.

The first step is to create a pointer into INDEX.FIL by isolating the first letter of the data sought and removing it's ASCII bias:

```
KEY.LETTER$ = LEFT$(KEY.DATA$,1)
POINTER%    = ASC(KEY.LETTER$) - 64
```

Fred L. Golke, 1000 Blair Road, Carteret, NJ 07008.

The first instruction isolates the first letter of the data sought, and the second creates POINTER equal to the KEY.LETTER's position in the alphabet, i.e. A=, B=2, ... Z=26.

Next, POINTER is used to calculate the offset from the beginning of STOCK.FILE for the entries starting with KEY.LETTER:

```
FILE.KEY% = 0
SIZE%     = 0
FOR    I% = 1 TO POINTER%
  FILE.KEY% = FILE.KEY% + SIZE%
  READ #INDEX.FILE%,I%;SIZE%
NEXT I%
```

FILE.KEY and SIZE are cleared. A loop adds the SIZE for each letter to FILE.KEY until the SIZE of KEY.LETTER is read. When these instructions have executed, FILE.KEY is equal to the number of entries in STOCK.FIL which are alphabetically lower than KEY.LETTER.

Processing beyond this point is sequential:

```
    IF SIZE%       = 0 THEN GO TO xxx
100 FILE.KEY%      = FILE.KEY% + 1
    READ #STOCK.FILE%,FILE.KEY%;FILE.DATA$
    IF KEY.DATA$ = LEFT$(FILE.DATA$,KEY.SIZE%)
        THEN GO TO xxx
    SIZE%          = SIZE% - 1
    IF SIZE%     GT    0      GO TO 100
```

If there are no entries for this KEY.LETTER, handle that condition. Otherwise, read the file data sequentially until the entry sought is found or all entries are exhausted. If found, the routine exits with FILE.KEY pointing to the correct record. If updated, it can be re-written using the same key.

Of course, each time a new record is added to STOCK.FIL the INDEX.FIL must be updated. That's easily done:

```
KEY.LETTER$ = LEFT$(KEY.DATA$,1)
POINTER%    = ASC(KEY.LETTER$) - 64
READ   #INDEX.FILE%,POINTER%;SIZE%
SIZE%       = SIZE% + 1
PRINT #INDEX.FILE%,POINTER%;SIZE%
```

handles the job nicely. For deletions, reduce SIZE by 1.

Patching A CP/M Diskette On A North Star System

by Tom Wiens

Every computer system manual at some point includes a pointed warning—always back up you diskettes! But after logging months of work without a fatal error, I became lax and not a little bit careless, until the inevitable happened: late one night I finished off an additional ten pages of writing on a word processor running under CP/M, and began to write to disk—ERROR: DISK FULL was the program's response. All my attempts to salvage the situation only got me in more trouble. Finally, in desperation, I rebooted without ending the run or closing the file in the hope that some temporary files could still be found on the diskette and used to reconstruct the text. But the CP/M DIR command produced no evidence that any part of my creation was alive or well in any form.

Having found in the past that erased CP/M files could easily be salvaged (see *The CP/M Connection*, Part II, in Vol. 1/ No. 5), and groaning at the prospect of many hours wasted recreating lost ideas, I turned immediately to North Star DOS and the North Star monitor to have a look at the CP/M directory. On my version (Lifeboat Associates' single-density CP/M 1.4 for North Star diskettes), the latter is stored as 8 blocks beginning at disk address 30, which can be read into memory using the DOS RD command, and then examined and modified with the aid of the monitor. In this instance I found that the directory contained one "live" old version of the document I was working on (minus the ten pages added that evening) and one "erased" new version. Since CP/M erases by placing an E5H in the first byte of the file entry in the directory, I used the North Star monitor to replace the E5 with a 00H, modified the filename to prevent confusion with the old version, and used the DOS WR command to write the directory back to diskette. Re-booting CP/M and running the word processor, I was further dismayed to find that the restored file included little of the lost text.

Not ready to give up, I returned to DOS and again examined the CP/M directory entries. I was particularly interested in the group numbers—since the word processor has crashed while trying to write to disk, perhaps it had not written the file entry back to disk. Perhaps some of the "unused" groups not included in the directory entries for the old or restored versions of the file contained parts of the new text.

To check out this possibility, I first had to be able to translate the group numbering system used by this version of CP/M into the corresponding North Star disk addresses, so that I could RD or WR particular groups at will. Since the groups were numbered 2-79 and I knew that my CP/M files began at North Star disk address 38 and could run through disk address 349, it was easy to deduce that if E were the group number, the corresponding North Star disk address would be (E-2)*4+38. With this knowledge, I was able to RD in the unused sectors and then examine the text with the monitor. A slow process to be sure, but within an hour I had found most of my missing text, determined the ordering of the groups (which was sequential with some jumps), and then "created" a new file entry into which I placed the series of groups. When I returned to CP/M and the word processor and tried to read this text, I found that it would only read a small piece of the reconstructed file. Returning to DOS and examining the end of this piece, I found a series of 1AH's—Control-Z's, the CP/M end-of-file mark. Zeroing these and rewriting the affected block to disk solved the problem. I went to bed reassured that the total loss of text had been reduced to one paragraph instead of ten pages!

Let a Program Do the Dirty Work

While the above procedure will work in a pinch, it is rather tedious—why not automate the whole process, taking advantage of what I now understand about CP/M diskette structure? This notion led to the North Star Basic program CP/MDP given in the listing. The program is written for the Lifeboat single-density version of CP/M 1.4, Release 5 single-density DOS and North Star Basic, and furthermore exploits the display control features of the IMSAI VIO-C memory-mapped video board. Obviously a bit of careful patching is required before the program will run correctly on other systems.

What does it do? First, it automatically provides a listing of a CP/M diskette directory, both "live" and "erased" entries, giving filenames (exactly as written on disk) and group numbers (translated into North Star disk addresses). Those files which are "live" are highlighted on the screen; if any groups of "erased" files have not been overwritten by "live" files, the first appearances of those groups are also highlighted to indicate that they may contain salvageable information (though not necessarily the information you might expect !). After this listing is displayed on the screen, the user may also request a hardcopy printout.

Thomas B. Wiens, 2025 Westwood Terrace, Vienna, VA 22180.

Following the above, the directory information is used to display a "diskette map," which simply lists the contents of the diskette in order of disk address, indicating also where empty sectors occur.

A menu is then displayed, which allows the user to choose to:

1.REPEAT SECTOR MAPPING
2.MODIFY DIRECTORY
3.READ A SECTOR (HEX)
4.READ A SECTOR (ALPHA)
5.JUMP TO CP/M
6.JUMP TO DOS

The first option allows the user to either list the directory of another CP/M diskette or relist the previous one, perhaps to confirm that changes have been made as desired.

The second option requests the number of the directory entry to be changed (as previously listed), then allows the user to restore or erase the file, change the filename, change the number of records and/or change the group of numbers. One can, for example, take an exisiting erased entry, restore it to life, put in a new name, number of records (128 bytes per record; 8 records per group; and 128 records per entry) and set of disk addresses, thereby creating an entirely new file from groups which have not been committed to another live file (as I did when salvaging my text as described above). The program does not currently allow you to modify the extent numbers, although that could easily be added—they lie in the directory byte immediately following the filename.

The third option requests a North Star disk address and reads in 4 blocks of hex data. The user may view this by moving the cursor forward or backward on the screen with keyboard left and right arrows (or some other control keys, if statements 1040 and 1050 are modified). Typing in any two hex characters (0-9,A-F) replaces the pair under the cursor. Typing ESCAPE writes the modified data back to diskette; typing RETURN returns one to the menu. The most frequent use of this option is in patching .COM files. It can be used, for example, to examine and patch your BIOS I-O routines, the location of which on diskette is usually described in the implementation manual for any version of CP/M.

The fourth option also requests a disk address, reads in 4 blocks of ASCII text and displays it all on the screen (with ? marks for non-printable characters). This option can be used to search for missing or garbaged pieces of files. Unlike CP/M, it is not stopped by Control-Z's in the text, and so it can be used to pinpoint their locations. However, the third option must be used if the user wishes to modify the text (e.g., to remove garbage).

Options five and six provide jumps to a cold boot of CP/M, assuming that a CP/M diskette has been placed in drive 1, or to North Star DOS (the entry points used are given in the CALLs at statements 795-800).

What's the Gimmick?

North Star Basic, like CP/M, won't let you read or write from disk independent of the file structuring system. So how do we accomplish the above? The gimmick is to exploit North Star DOS' auto-start facility (see North Star *System Software Manual*, Rev. 2.1, pp. F-1 to F-2). To use CP/MDP, the DOS byte which initiates this

Patching cont'd. . .

facility must be set to zero (it is address 2030H; a FILL 8240,0 statement may be added to the beginning of the program to initialize this byte). One must also locate the DOS command input buffer (its location is stored at 2031-2032H; in the single-density version 5.0, the buffer begins at address 10109 decimal). A number of statements in the program stuff commands into this buffer, and may need modification for different version of DOS (statements 50, 980, and 1235-1245). Finally, we need an entry point into DOS which, if called, initiates an auto-start. This entry point is an "LDA 2030H" statement which, in my DOS, is found at address 8379 decimal; it may be located in other versions using the monitor command, SM 2000-2CFF 3A,30,20. Statements 70, 1090, and 1100 should be modified if the address is not the same. With these adjustments, when a read or write of the CP/M diskette is necssary, the program will stuff the appropriate command into the DOS buffer and call the auto-start entry point, which completes the command, leaving the user in DOS. To return to the program, it is necessary to type "JP 2A04" (2D04 in double-density versions) and "RUN n", where n is the desired CP/MDP entry point (all this is prompted by the program before it enters DOS.)

When the program reads blocks of text into memory, it stores them beginning at address 01. Since every RUN command re-initializes all variables, the program uses its own buffer at address 4091 decimal as temporary storage. Both these locations are arbitrary and can be changed if necessary.

As the program is written for my system, it exploits the power of the IMSAI VIO-C video board for an attractive display which highlights significant data (using character-by-character reverse video). The relevant video control codes, which will need to be replaced for other terminals/displays, are as follows:

CHR$(26) = protect/unprotect reverse video fields;
CHR$(22) = turn on/off individual character reverse video;
CHR$(16) = protect/unprotect reverse video fields;
CHR$(27)+" = "+CHR$(31+Y)+CHR$(31+X) positions the cursor at line Y and column X.

Also, for keyboards without positional keys (arrows), the ASCII numbers "9" and "8" in statements 1040 and 1050 will need to be replaced.

The formulas converting CP/M group numbers to North Star disk addresses are found in statements 235, 730, 925, and 935. For different versions of CP/M or DOS, these formulas may need to be modified. The first part of the article suggests how to deduce the required conversion formula.

In reading the CP/M directory, CP/MDP determines the number of file entries by examining the first byte of each potential entry to see if it is a 0 or E5H; if it is not, the program assumes that the last entry has been reached. Should the directory itself get garbaged, this assumption may be incorrect, and the program may fail to read portions of the directory. In that case, use option 5 to jump to DOS and any of the monitors other than M0000 to examine the directory (DA 1,90 will do for the first block) and place a zero or E5H in the garbaged byte which is blocking the full directory read. After returning to DOS, use the WR command to write the directory back to diskette (8 blocks).

CP/MDP maps the diskette by setting up a vector X with 79 entries corresponding to groups 2-79 (the dimensions may need to be changed for other version of CP/M—see statements 140 and 720). If a group appears in "live" file, the entry number for that file is written as a positive number in X; if it appears in an "erased" file, it is written as a negative number. If the group appears in no directory entry, it is treated as "empty."

Saved Again

Ironically, I had written two-thirds of the text for this article when my word processor again crashed—this time because I had mistakenly directed the new text file to a crowded diskette. With CP/MDP available, no worry: I moved the injured diskette to disk 2, booted DOS on disk 1 and ran CP/MDP. The directory entry to the text file contained no group information, but I noted the disk addresses where there were blocks which were potentially "alive" but uncommitted to an unerased file. Using the fourth finding, I checked each of these, finding that they contained all but a small part of the lost text (of course, had the diskette been completely full, nothing could have been saved). Returning to option 2, I entered 8 addresses in the file directory entry, and computed and entered the number of records as 8x8 = 64. After having the directory written back to diskette, I moved the CP/M diskette back to drive 1, chose option 5 to boot CP/M, and was back in business. □

```
*****************************************************************
*                                                               *
*               LISTING: CP/MDP   AUTHOR:  T. WIENS             *
*                                                               *
*****************************************************************

1REM THIS PROGRAM USES SOME VIDEO CONTROL CHARS. WHICH WILL NEED MODIFICA-
2REM TION FOR OTHER SYSTEMS--CHECK ALL !CHR$() STATEMENTS.  DOS 5.0 SINGLE
3REM DENSITY AND CP/M 1.4 SINGLE DENSITY (LIFEBOAT ASSOC. VERSION) IS ASSUMED.
4REM FOR OTHER VERSIONS, CHECK CAREFULLY THE DOS LOCATIONS CALLED OR FILLED,
5REM AND THE LAYOUT OF CP/M DISKETTES.  SOME KEY PARAMETERS\
6REM 10109+ -- DOS COMMAND INPUT BUFFER,  8379 -- DOS AUTOSTART ENTRY POINT
7REM 2A04H -- BASIC CONTINUE ENTRY POINT, 8232 -- DOS REENTRY POINT
8REM 59648 -- COLD BOOT ENTRY, 4091+ -- BUFFER USER BY PROGRAM
9REM 22146 -- TERMINAL CHAR.POINTER, 30 -- SECTOR WHERE CP/M DIR. BEGINS
```

```
10REM 38 -- FIRST SECTOR FOR CP/M FILES, 349 -- LAST SECTOR
11REM  2 -- FIRST CP/M GROUP NO., 79 -- LAST CP/M GROUP NUMBER
19 LINE#1,84\REM TO DEFEAT AUTOMATIC CARRIAGE RETURNS
20 !CHR$(26),"CP/M DISKETTE PROCESSOR by Thomas B. Wiens"
21 !"This program provides a means of reading & modifying CP/M diskettes,"
22 !"including editing CP/M directories to recover erased files or destroy"
23 !"garbage (even to piece together new files from pieces scattered around"
24 !"the diskette), reading individual blocks of text or hex files, patching"
25 !".COM files, and screen or hardcopy mapping of diskette structure."
26 !"    RAM starting at 01H is used as buffer space.  References to sector"
27 !"numbers are based on NS sector format, converted back and forth from"
28 !"CP/M's group numbering scheme.  For some commands, the program stuffs"
29 !"commands into DOS' input buffer and calls DOS to execute; to return"
30 !"to the program, follow the printed instructions EXACTLY."
33 !"    BEWARE OF MODIFYING THE CP/M DIRECTORY UNTIL YOU KNOW WHAT YOU"
34 !"ARE DOING--PRACTICE ON A DUPLICATE DISKETTE."
35 !"   NOTE: CP/M AND ITS PROGRAMS TAKES CONTROL-Z (1AH) AS AN END-OF-FILE"
36 !"MARK, AND WILL NOT READ PAST SAME, WHEREAS THIS PROGRAM WILL.  IF "
37 !"DESIRED, YOU CAN USE THE ROUTINES HEREIN TO ELIMINATE THE CONTROL-Z'S."
39 !\!"PRESS ANY CHARACTER TO CONTINUE..."\Y$=INCHAR$(0)
40 !CHR$(26),"INSERT YOUR CP/M DISKETTE IN DRIVE 2, THEN PRESS ANY KEY"
41 Y$=INCHAR$(0)
45 DATA 82,68,32,51,48,44,50,32,49,32,56,13
46 DIM Z(12)\REM CONTAINS ASCII CODES FOR DOS RD COMMAND
50 FORI=1TO12\READZ(I)\FILL 10108+I,Z(I)\NEXT
60 !\!"NOW, TYPE:  JP 2A04"
65 !"             RUN 100"
70 Z=CALL(8379)\REM JUMP TO DOS AUTOSTART
100 !CHR$(26),"*******  DIRECTORY LISTING FOLLOWS *******"
101!"ACTIVE FILES ARE LISTED IN REVERSE VIDEO; ERASED FILES IN NORMAL VIDEO"
102!"ERASED BUT PERHAPS SALVAGEABLE BLOCKS ARE ALSO IN REVERSE VIDEO!"
105B=0\L1=32\L2=64\P1=0
106 ERRSET 4000,E1,E2
110FORI=1TOL2*32STEP32
115 X=EXAM(I)
120IFX<>229ANDX<>0THEN EXIT130\REM 229=ERASED FILE EXTANT
122 X=EXAM(I+1)\IF X=229THENEXIT130
125NEXTI
130L2=(I-1)/32
140DIMD(L2),R(L2),N$(11*L2),M(L2,16),A$(11),X(79)\N$=""
150FORI=1TOL2
152K=(I-1)*32+1
155D(I)=EXAM(K)\IFD(I)=229THEND(I)=1\REM FILE EXTANT DEAD OR ALIVE
160FORJ=K+1TOK+11\N$=N$+CHR$(EXAM(J))\NEXTJ\REM FILE NAMES (11 CHARS)
165R(I)=EXAM(K+15)\REM NO. OF RECORDS IN EXTANT
167L=0
170FORJ=K+16TOK+31
172L=L+1
175X=EXAM(J)\IFX=0THENEXIT185
180 M(I,L)=X\REM CP/M BLOCK NO. (16 PER EXTANT)
182 NEXT J
185NEXTI
190 GOSUB 400
200!"NO. FILENAME   DISK SECTORS"
210FORI=1TOL2
215 IFD(I)=0THEN!CHR$(22),\REM REVERSE VIDEO IF "LIVE" ENTRY
216 IF P1=1 ANDD(I)=1THEN!#P1,CHR$(91),\REM BRACKET DEAD FILES
220A$=N$((I-1)*11+1,(I-1)*11+11)
225!#P1,%2I,I,%#3I," ",A$," ",
230FORJ=1TO16\IFM(I,J)=0THENEXIT240
232 IFM(I,J)<0THEN!CHR$(22),
235X=(ABS(M(I,J))-2)*4+38\!#P1,X," ",\IFM(I,J)<0THEN!CHR$(22),
238 NEXTJ
240 IF P1=1 ANDD(I)=0THEN!#1,CHR$(93),\REM RIGHT BRACKET
241IFD(I)=0THEN!CHR$(22),\!#P1\REM TURN OFF REV.VIDEO
```

```
242 IF I=20 OR I=40 THEN GOSUB 280
245NEXTI
247 !"PRESS ANY KEY FOR MEMORYMAP:"\Y$=INCHAR$(0)
250 GOTO 700
280 !"PRESS ANY KEY FOR NEXT DISPLAY; 'P' FOR HARD COPY:"
290Y$=INCHAR$(0)\IF Y$="P" THEN 305
300P1=0\RETURN
305P1=1\I=I-20
310!#P1,"NO. FILENAME   DISK SECTORS"\RETURN
400 FOR I=1TOL2
410 IF D(I)<>0THEN450
420 FOR J=1TO16\X(M(I,J))=I\NEXTJ
450 NEXTI
460 FOR I=1TOL2
470 IF D(I)<>1THEN550
480 FOR J=1TO16\IFX(M(I,J))<>0THEN500
490 X(M(I,J))=-I\M(I,J)=-M(I,J)\REM X MAPS DISKETTE, M NEG. IF BLOCK
500 NEXTJ\REM ERASED.  IF FILE ERASED BUT BLOCK NOT USED BY AN ACTIVE
550 NEXT I\REM FILE, TREAT THE BLOCK AS POTENTIALLY RECOVERABLE
560RETURN\REM THE 1ST TIME IT'S FOUND, THIS MAY BE INCORRECT!!
700 REM MEMORY MAP
701 REM USES PROTECTION FOR INVERSE VIDEO FIELDS TO FORMAT OUTPUT
702!CHR$(26),CHR$(16),CHR$(22),
705 !"**********        DISK MAP (BY NS SECTOR)    *****************"\!
710 A$="            "
720 FOR I=2TO79\REM RANGE OF CP/M BLOCK NOS. IN VERS. 1.4
725 IF X(I)=X(I-1)THEN750
726 IF X(I)<>0THEN 727 ELSE A$="**EMPTY**  "\GOTO730
727 A$=N$((ABS(X(I))-1)*11+1,(ABS(X(I))-1)*11+11)
730 !(I-2)*4+38,"-",A$,"  || ",CHR$(13),
750 NEXT I
760 !CHR$(22),CHR$(16)
770 !\!"PRESS ANY KEY TO CONTINUE:"\Y$=INCHAR$(0)
775 !CHR$(26)\!"CHOOSE ANY OF THE FOLLOWING:"
780 !"1 -- REPEAT SECTOR MAPPING"\!"2 -- MODIFY DIRECTORY"
785 !"3 -- READ A SECTOR (HEX)"\!"4 -- READ A SECTOR (ALPHA)"
787 !"5 -- JUMP TO CP/M"\!"6 -- JUMP TO DOS"
790 !"   YOUR CHOICE?"\Y$=INCHAR$(0)
795 X=VAL(Y$)\IF X=1THEN40\IF X=6 THENZ=CALL(8232)
800 IFX=5THENZ=CALL(59648)\IFX=2THEN850\IFX=3THEN1000\IFX=4THEN1100
810 GOTO 775
850 !CHR$(26),\INPUT"ENTRY NO.",N1
852 ERRSET 4001,E1,E2
855 K=(N1-1)*32+1\A$=N$((N1-1)*11+1,(N1-1)*11+11)
860 IF D(N1)=0 THEN865ELSE!"FILE ",A$," HAS BEEN ERASED. RESTORE IT (Y/N)?",
862 Y$=INCHAR$(0)\IF Y$<>"Y" THEN 870 ELSE FILL K,0\D(N1)=0\GOTO870
865 !"FILE ",A$," IS ACTIVE. KILL IT (Y/N)?",
867 Y$=INCHAR$(0)\IF Y$<>"Y" THEN 870 ELSE FILL K,229\D(N1)=1
870 !\!"(PUSH CR IF NO CHANGE)         OLD                NEW"
875 !"FILENAME (11 CHARS.):     ",A$,"      ",\INPUT A$
880 IF LEN(A$)<>11 THEN 900\N$((N1-1)*11+1,(N1-1)*11+11)=A$
885 FOR J=1TO11\FILLK+J,ASC(A$(J,J))\NEXTJ
900 !"RECORDS COUNT (128 PER EXTANT,"
901 !" OR 8 PER GROUP)              ",R(N1),"             ",\INPUT Y$
910 IF Y$=""THEN 920 ELSE R(N1)=VAL(Y$)
915 IF R(N1)<=128 THEN FILL K+15,R(N1)
920 !"FILE GROUPS (4 SECTORS EACH):"
925 FOR J=1 TO 16\X=(ABS(M(N1,J))-2)*4+38\IFX=30THENX=0
930 !"   # ",J,"                   ",X,"                  ",\INPUT Y$
935 IF Y$=""THEN945 ELSE X=(VAL(Y$)-38)/4+2
940 IF X<80 AND X>1 THEN FILL K+15+J,X ELSE 942\M(N1,J)=X\GOTO 945
942  FILL K+15+J,0 \M(N1,J)=0
945 NEXT J
950 !\!"WRITE DIRECT. TO DISK (Y/N) OR CR TO REVIEW ENTRY:"\Y$=INCHAR$(0)
955 IF Y$="Y" THEN 975\IF Y$="N" THEN 775 ELSE 855
```

```
970DATA 87,82,32,51,48,44,50,32,49,32,56,13
975 RESTORE 970
980FORI=1TO7\READ Z(I)\FILL 10108+I,Z(I)\NEXT
990 GOTO 60
1000 GOSUB 1200\!"              RUN 1010"\Z=CALL(8379)
1010 !CHR$(26),"USE LEFT OR RIGHT ARROWS TO VIEW OR POSITION CURSOR"
1015 !"TYPE TWO HEX CHARS. TO REPLACE THOSE UNDER CURSOR; TYPE AN "
1020 !"ESCAPE TO WRITE MODIFIED FILE BACK TO DISK; CR TO RETURN TO"
1025 !"MENU."\!CHR$(22),\I1=1\Y$="  "\Y1=36\X1=32
1030 !CHR$(27)+"="+CHR$(Y1)+CHR$(X1),\REM PLACE CURSOR ON SCREEN
1035 Y$(1,1)=INCHAR$(0)\X=EXAM(I1)\FILL 22146,0\X2=ASC(Y$)
1040 IF X2<>9THEN1050ELSE!FNH$(X),\I1=I1+1\X1=X1+3
1045 IF X1<=104THEN1030ELSEY1=Y1+1\X1=32\GOTO 1030
1050 IF X2<>8THEN1060ELSEI1=I1-1\X1=X1-3
1055 IFX1>=32THEN1030ELSEX1=104\Y1=Y1-1
1056 IFY1>35THEN1030ELSEY1=36\X1=32\I1=I1+1\GOTO1030
1060 IF X2<>13 THEN 1065\!CHR$(22)\GOTO 775
1065 IF X2=27THEN1080ELSE IFX2<48ORX2>70THEN1035
1070  Y$(2,2)=INCHAR$(0)\X2=FND(Y$)\IFX2>=0THENFILL I1,X2
1075 !Y$,\I1=I1+1\X1=X1+3\GOTO1045
1077DATA 87,82,32,44,50,32,49,32,52,13
1080RESTORE 1077\!CHR$(22)
1082 B$=""\X=EXAM(4090)\FORI=1TOX\B$=B$+CHR$(EXAM(4090+I))\NEXT
1084 !"WRITING SECTOR ",B$
1085 FORI=1TO10\READ Z(I)\NEXT
1090 GOSUB1235\!"               RUN 775"\Z=CALL(8379)
1100 GOSUB 1200\!"             RUN 1110"\Z=CALL(8379)
1110 !CHR$(26),"SECTOR READS:"\!
1120 FORI=1TO1024\X=EXAM(I)
1130IF(X>31AND X<128)OR(X>9ANDX<14)THEN!CHR$(X), ELSE!"?",\NEXT
1140GOTO770\REM NON-ALPHA CHARS. PRINT AS '?'
1200 INPUT "SECTOR NO. FOR 4-SECTOR READ? ",X
1210DATA 82,68,32,44,50,32,49,32,52,13
1220 RESTORE 1210
1230 FOR I=1TO10\READ Z(I)\NEXT\B$=STR$(X)
1235 FOR I=1TO3\FILL 10108+I,Z(I)\NEXT\FILL 4090,LEN(B$)
1240 FOR I=1TOLEN(B$)\X=ASC(B$(I,LEN(B$)))\Z=10111+I
1243 FILL Z,X\FILL 4090+I,X\NEXT
1245 FOR I=4TO10\FILL Z+I-3,Z(I)\NEXT
1250 !\!"NOW, TYPE:   JP 2A04"
1260RETURN
1600 DEF FND(H$)\REM HEX/DEC CONVERTER FROM NS MANUAL
1620IFH$=""THEN1675\T=0
1630FORE=LEN(H$)TO1STEP-1\C=ASC(H$(E,E))
1640IF(C<ASC("0"))OR(C>ASC("F"))THENEXIT1675
1645IF(C>=ASC("0"))AND(C<=ASC("9"))THENC=C-48
1650IF(C>=ASC("A"))AND(C<=ASC("F"))THENC=C-55
1655IF(C>ASC("9"))AND(C<ASC("A"))THENEXIT1675
1660T=T+C*(16^(LEN(H$)-E))
1665NEXT
1670RETURNT
1675RETURN-1
1680FNEND
1700DEF FNH$(D)
1730H1$=""
1740 FORI=1TO0STEP-1
1745D2=INT(D/(16^I))
1750 IF D2>=10 THEN H1$=H1$+CHR$(ASC("A")+D2-10)
1755IFD2<10THENH1$=H1$+CHR$(ASC("0")+D2)
1760D=D-(D2*(16^I))
1765NEXT
1770RETURNH1$
1775FNEND
4000 IF E1=106 THEN 150\REM TO PREVENT REDIMENSIONING ERROR
4001 IF E1<>85  STOP ELSE !"MUST RE-READ DIRECTORY!"\GOTO40
```

Selecting Functions Via Tables In 8080 Assembly Language

by Fred Gohlke

It is often necessary to write a routine which will perform one of several functions, depending on a code character. The beginning programmer may first notice this need when providing for selections from the keyboard. For this reason, these routines are frequently written to take the first character of a mnemonic command:

```
'E' from Execute
'P' from Print
'T' from Tape
```

and use it as the code character. Execution of the routine selected is usually accomplished by putting the code character in the 'A' register and proceeding something like this:

```
 .
 .
CPI   Code-for-execute
JZ    Execute-routine
CPI   Code-for-print
JZ    Print-routine
CPI   Code-for-Tape
JZ    Tape-Routine
 .
 .
```

While effective for selecting from a few alternatives, this style becomes cumbersome and slow as the system grows in size: The selection of suitable mnemonics becomes increasingly difficult; there is no logical relationship between the code characters and the routines they invoke (in the sense that the 4th code invokes the 4th routine, rather than the letter indicating the routine name); each new set of codes and routines requires a new set of decodings and invocations; and, it fails to recognize that selecting between alternative routines is a central part of the programmer's task which should be handled systematically.

There is another approach to this problem which allows great flexibility and lends itself to modular program design. The first step is to design a module which can be used from anywhere in the system to distinguish between alternative functions. The second step is to set up the means of invoking this module when necessary.

Fred L. Golke, 1000 Blair Road, Carteret, NJ 07008

The module will require 2 parameters: the address of a table of function addresses, and a function pointer. The SELECTOR routine which follows expects the table address in the 'HL' register pair and the function pointer in the 'A' register:

```
SELECTOR ADD   A      Double the function pointer
         ADD   L      Add low table address byte
         MOV   L,A    Restore to 'L'
         JNC   LOAD   No carry, no overflow
         INR   H      Overflow, increment 'H'
LOAD     MOV   A,M    Pick up low function addr
         INX   H      Point to high function addr
         MOV   H,M    Set high function address
         MOV   L,A    Set low  function address
         PCHL
```

Note that the routine ends by jumping directly to the selected function. This requires that SELECTOR be CALLed, and that each function end with a RETurn instruction, which tends to maintain modularity. Also note that a function pointer of zero points to the first function. The routine simply uses the function pointer as an offset from the beginning of the table of routine addresses. Finally, note that the 'BC' and 'DE' register pairs are available for passing parameters to functions.

With the SELECTOR routine as a base, all that's left is to provide for its invocation by table construction and code selection. Table construction is neatly handled by assemblers. Assuming the appropriate routines exist somewhere in the system:

```
TABLE1  DW  RTN1     Routine invoked by zero
        DW  RTN2     Routine invoked by 1
        DW  RTN3     Routine invoked by 2
         .

TABLE2  DW  XCHRA    Routine invoked by zero
        DW  XCHRB    Routine invoked by 1
        DW  XCHRC    Routine invoked by 2
         .
         .
```

will create the necessary tables.

Code selection can be handled several ways. One way is direct conversion of the code character to a function pointer. This can be illustrated by selecting a sequence of ASCII characters, such as 'J' through 'V'. If the code character is between 'J' and 'V', and the ASCII value for 'J' (4A Hex) is subtracted from it, the result will be between zero and 12 (decimal). Thus, the letter 'J' causes execution of the first routine, 'K' the second, 'L' the third, and so on. This method is simple and direct, and lends itself to code validation. If the result of the subtraction is greater than the difference between the first and last character in the sequence, the result is invalid:

```
004A =    FIRSTCH  EQU  'J'
0056 =    LASTCH   EQU  'V'
000C =    LIMIT    EQU  LASTCH - FIRSTCH
```

with the code character in the 'A' register:

```
CODES1  DB   'E'    Code to be converted to zero
        DB   'P'    Code to be converted to 1
        DB   'T'    Code to be converted to 2
        DB   'D'    Code to be converted to 3
         .
        DB    0     Table terminator
```

performs the conversion and validates the result. This leaves the offset in the 'A' register, so SELECTOR can be called as soon as the 'HL' register pair is set to the appropriate table.

If a code sequence isn't practical, the assembler will create a code table:

```
D64A                  SUI  FIRSTCH
FE0D                  CPI  LIMIT + 1
D2nnnn                JNC  Error Routine
```

In this case, with the code character in the 'A' register, and the 'DE' register pair holding the code table address, the following instructions will set up the call to SELECTOR:

```
CHECK    MOV  B,A      Save for comparison
         XRA  A        Clear 'A' register
         MOV  C,A      Clear 'C' register
CHECK1   LDAX D        Pick up code letter
         ORA  A        Check for terminator
         STC           Use carry as error flag
         RZ            Error return, no match
         CMP  B        Compare code character
         MOV  A,C      Prepare for SELECTOR
         RZ            Valid, offset in 'A'
         INR  C        Increment the offset
         INX  D        Increment code pointer
         JMP  CHECK1
```

Variations of this routine will accomodate multi-character codes for tasks. They must simply insure an unambiguous error flag for invalid codes, and a zero offset for the first routine.

The procedures recommended here seem more complex than a simple CPI followed by JZ. But don't be put off by the verbosity of this presentation. What it boils down to is that, after getting the code character into the 'A' register, either:

```
SUI   FIRSTCH
CPI   LIMIT + 1
JNC   Invalid-code-routine
LXI   H,TABLEn
CALL  SELECTOR
 .
 .
```

or:

```
LXI   D,CODELISTn
CALL  CHECK
JC    Invalid-code-routine
LXI   H,TABLEn
CALL  SELECTOR
 .
 .
```

handles all the processing. As confidence in this method grows, the selection of codes and creation of tables becomes almost trivial. □

New Life For An Old Text Editor

by Bill Machrone

How to use ZTEL (Z-80 Text Editing Language) for letter and address merging. Also, how ZTEL can be used with other text editors such as "Word Star".

Since I fired up my first microcomputer in 1977, the majority of my time has been spent editing text of one kind of another. That first system was a TDL Alpha box with Z-80, 16K, the VDB terminal board and 1200 baud cassette. It came with the TDL software package, which included the excellent macro assembler, Basic, the Text Output Processor and a complicated text editor called ZTEL.

ZTEL supports a potentially bewildering array of macro instructions, conditional branches, iterations and options.

I say complicated because ZTEL (Z-80 Text Editing Language) supports a potentially bewildering array of macro instructions, conditional branches, iterations and options in addition to the normal text editing functions. These additional features permit you to actually write programs that manipulate text, prompt the console and perform logical operations. I eventually mastered it and came to regard it as a close friend even in the slow tape environment. As time passed I added more memory, then disk. Desiring the maximum software compatibility, I selected the TDL disk utilities package to supplement those that came with CP/M . The package included a new version of ZTEL, with some useful additions for the CP/M environment.

ZTEL is an editor that encourages proficiency on the part of the user. It is neither simple nor 'friendly.' While its basic command structure resembles Digital Research's ED, it was modelled after DEC's TECO, long considered to be one of the best 'programmer's' text editors. In fact, ZTEL supports most of all of the useful features of TECO, including text registers, value registers and macro capability. It uses virtually all the same commands, so that an experienced TECO user could move right over to ZTEL and be comfortable with it. ZTEL's syntax consists primarily of one-or two-letter commands that are fairly mnemonic in terms of what they do. For instance, 'i' puts the editor in insert mode, 's' searches for a text string, 'd' deletes a character and 'k' kills a line of text. Most of the commands accept a numeric parameter, so that '5d' deletes 5 characters. This is fairly standard nomenclature for a text editor of this type, but where ZTEL goes beyond is in the ability to modify some of the parameters in more complex ways and to store whole strings of commands in one of ten macro registers which can be invoked with just two keystrokes, such as 'm5'.

The contents of macro 5 could possibly tell the editor to move the pointer back 40 lines, then display 20 lines. Another macro might move the pointer ahead 20 lines, then display the next 20 lines. Thus, you can implement your own "page forward" and "page back" commands. As I mentioned above, testing and conditional branching is supported. This means that you can write macros that do things like find all occurrences of the word "boat" and change them to "ship." With a little ingenuity, you can construct the macro so that it automatically capitalizes

Bill Machrone, Sales Productivity Systems, Inc., 2121 Saw Mill Rd., White Plains, NY 10607.

any occurrence at the beginning of a sentence. The ZTEL manual gives examples of macros that convert from decimal to hexadecimal, reverse all the lines in the file and tell you how many lines or words there are in a file.

An experienced TECO user could move right over to ZTEL and be comfortable with it.

The biggest problem with ZTEL, as with Digital Research's ED, is that it is oriented towards a hard copy terminal such as a teletype instead of a CRT. Thus you have to end every command with an additional command that displays a few lines and the pointer, so you can see the results of your editing. This can be built into a macro, but the pointer and the cursor are in two different places on the screen. This makes text insertion somewhat difficult.

So, when I saw a demonstration of WordStar, I couldn't live without it. WordStar seemed to overcome all of the operating disadvantages of ZTEL, primarily through its full screen edit, real-time justification and margination and concurrent printing. Of course, it takes three times as much memory as ZTEL, but everything has its price.

As I began to use WordStar more, I realized that it, too, was less than perfect. I missed being able to construct macros that would move large pieces of text around dynamically. WordStar is limited to one marked block of text that can be moved, deleted, copied or written out. ZTEL offers up to ten text registers which can hold a macro, a single character or an entire file's worth of text. When you delete a string with ZTEL, you often delete to a text register so that you can get the deleted text back if you change your mind. It also sports ten value registers which may hold the decimal equivalents of ASCII characters or the results of calulations. ZTEL performs the four arithmetic functions, so that calculation is possible. It is my understanding that other text editors, such as Magic Wand and Wordmaster, have some of the features that ZTEL offers, but none of them seem to do everything you want.

As time went on I found myself using WordStar more and more, using ZTEL infrequently, perhaps for an intensive search-and-replace job such as converting a Basic program to run on another interpreter. ZTEL is tremendously handy here because it can be programmed to create line numbers if desired, so that a Basic-E program, which does not require a number on every program line, can be easily converted to run with an interpreter that requires that every line be numbered (as most do).

As a hobbyist who was able to turn microcomputing into a business, I have maintained a vigil on most new hardware and software releases so that I could ride the crest of constantly improving technology and offer customers the best product at a good price with minimum effort. Faced with a need to do a letter and address merge program, I eagerly awaited the release of the mail/merge version of WordStar. As with most new software products, its release date came and went and we had an eager customer who wanted the mail/merge function as soon as possible. Of course, all the then commercially available mailing list packages were incompatible with their needs. We had to choose between writing an assembly language program to do the job, with its attendant long debug and development times, or doing it in compiled Microsoft Basic. Or did we? Out came the old ZTEL manual, and a quick perusal confirmed my suspicion: ZTEL could do the job.

Our requirements were straightforward: Read a fixed-length random access file with names, addresses and other data, compress extra spaces out of the city, state and zip fields, merge the output with a WordStar-generated one-page letter.

The CP/M version of ZTEL reads a file called "INIT.TEL" from the logged in disk, loads it into text register zero and executes it. This is normally used to initialize value registers that set the width of the screen, TTY or CRT mode or to automatically load your favorite macros into the appropriate registers. The need was for the mailing list merge to begin without intervention from the (non-programmer) user, who definitely would not understand the complexities of ZTEL.

By using the CP/M Submit function, we could dynamically rename a macro specifically constructed to merge addresses or type envelopes as INIT.TEL. Further, the user could use the familiar WordStar to create the letter that would receive the addresses. Better still, the macros would prompt the user so that he would know when to insert a new envelope or sheet or letterhead into the printer.

Out came the old ZTEL manual, and a quick perusal confirmed my suspicion: ZTEL could do the job

The system is described below, with appropriate listings for those of you who would like to implement a similar function on your own system. The program that extracts the addresses from the mailing list master file outputs the addresses as three or four lines terminated with carriage return/line feed sequences. Since part of the objective in a high-quality direct mailing campaign is to eliminate blank lines and fixed-length fields, the first character or of the address is the number '3' or '4', which fixed to variable length conversion is done at extract time. The output file, EXTRACT.ADR, looks like this:

```
3ABC Corporation
2150 Topanga Canyon Road
Torrance, CA 94503
4Tweedle Instruments, Inc.
Building 5
345 Maple Street
White Plains, NY 10607
3Underwood Manufacturing Co.
Merritt Industrial Park
Union, NJ 07083
```

Text Editor con't...

The application is set up on a two drive system so that the file EXTRACT.ADR and the letter for merging, which can be any name, are on drive B. Drive A is the logged-in disk and contains the following files:

```
MRGMACRO.TEL
ENVMACRO.TEL
ZTEL.COM
SUBMIT.COM
MERGE.SUB
ENVELOPE.SUB
MARGIN.COM
```

Below is the listing for MERGE.SUB, the submit file that controls the merging of addresses with a letter specified by the user. It is invoked by typing,

```
SUBMIT MERGE FILENAME.TYP nn
```

where the filename.typ is the name and type of the letter file created with WordStar on drive B:

```
margin $2
ren init.tel=mrgmacro.tel
ren b:letter.mas=b:$1
ztel
ren b:$1=b:letter.mas
ren mrgmacro.tel=init.tel
margin 0
```

A brief narrative of what happens is as follows: A short program, MARGIN.COM, is invoked. This assembly language program takes an ASCII numeric parameter from the CP/M default buffer and converts it into an escape sequence which positions the daisywheel printhead at the specified column and sets the margin there, too. SUBMIT then renames MRGMACRO.TEL to INIT.TEL, so that ZTEL will find, load and execute it when it is invoked. The letter specified by the user is renamed "LETTER.MAS" for the duration of the job so that ZTEL can find it easily. It then performs the merge, quits upon detecting an out-of-date condition and allows SUBMIT to restore everything back the way it was. Moving the margin is necessary because ZTEL is printing a raw WordStar file, which has not been indented or otherwise manipulated by the WS print module.

The file below, ENVELOPE.SUB, does essentially the same thing, except that the user does not specify a letter file:

```
SUBMIT ENVELOPE nn
```

For a 10-pitch typewheel, the user normally enters an indentation of 30 or 35. The ability to input this parameter is necessary so that the user can align 12-pitch addresses properly on the envelope.

```
margin $1
ren init.tel=envmacro.tel
ztel
ren envmacro.tel=init.tel
margin 0
```

Below is a listing of MRGMACRO.TEL. If you are not familiar with ZTEL it might be a little confusing, because the expressions are so terse. I've done some interpreting to aid in comprehension.

```
   @!/MRGMACRO.TEL/
          80wl
          124wy
          @er/b:letter.mas/
          0a
          h:pl
          @b/READ/
   @!/MESG/@x/
          ?? Type RETURN, ^R or ^C.    /
         @b/KEY/
  @!/READ/
         @er/b:extract.adr/
  @!/LOOP/@x/
  Align paper in printer and type 'return'/
 @!/KEY/ ek4                 @!/Get char from kybd./
         v4-3"e@b/DONE/' @!/^C gets you out,/
         v4-18"e@b/REPT/'@!/^R repeats letter./
         v4-13"e@b/EVAL/'@!/CR merges next adr./
         @b/MESG/
 @!/EVAL/
         hk                  @!/Clean slate./
         a                   @!/Get a line, but/
         ve-1"e@b/DONE/' @!/branch if EOF./
         :ecl                @!/Put first char. in vr 1/
         vl-51"e@b/TWO/' @!/"3" = 2 lines, "4" = 3 lines./
         vl-52"e@b/THREE/' @b/ERROR/
 @!/TWO/
         2a
         @b/MERGE/
 @!/THREE/
         3a
 @!/MERGE/
         h:p2                @!/Put address in tr 2/
         gl                  @!/Get the letter back/
         j                   @!/and get to top./
         @s/ADR/             @!/Find the place./
         0lk                 @!/Kill the line./
         g2                  @!/Get the address./
         zj                  @!/Go to the bottom,/
         12ei                @!/insert a form feed./
 @!/REPT/
         lwa                 @!/Interprets form feed./
         hel                 @!/List it out./
         0wa                 @!/Turn off A flag./
         @b/LOOP/            @!/Next address./
 @!/ERROR/                   @!/If not 3 or 4.../
         @x/
 File Construction Error- Ending processing.
    /
   eq
@!/DONE/
   @x/
No more addresses- ending processing./
   eq
```

Here's a synopsis of what's happening in the above program: After some initialization, ZTEL reads a file called LETTER.MAS and appends its contents. The letter is then deleted from the console and ZTEL accepts a character of input, which is stored in value register 4. The contents of that register are evaluated for a return, which causes the merge process to begin, or either of two control characters which force and end of job or cause the letter to be reprinted. Input of other than the three recognized characters causes the error message at label MESG to be displayed. ZTEL then reads the EXTRACT.ADR file and appends one line. Before doing anything else, it checks to see if the append operation set an end-of file flag. If it did, the macro branches to the label "DONE" and tells ZTEL to quit. If not, it extracts the first character from the file and puts it into a value register where it can be evaluated. Similar to an assembler CPI instruction, ZTEL subtracts the decimal value of the character '3' or '4' from the character in the value register. If the result is zero, ZTEL branches to the appropriate label. If neither value is found, there is an error condition and a branch is made to the error message routine, which then falls through to "DONE".

At this point the editor has either three or four lines of an address in the file buffer. Beginning at the "MERGE" label, the editor places the address in text register 2 and gets a copy of the letter back into the file buffer from text register 1. It then searches for the string "ADR", which is the location at which to replace the string with the address. It does this by killing the line that contains the "ADR" string and getting the address from text register 2. It then inserts a formfeed at the bottom of the text so that the printer will eject the letter when it is done.

The merged letter is actually typed by using ZTEL's 'el' (extended list) function which can send a copy of the file buffer or any portion thereof to the list device. The macro then branches back to "LOOP" to begin again. Although not illustrated in the listing above, any branch to the end of a macro takes the user back to the command mode of the editor. This is not particularly useful for the mailing list program, since the user has no idea how to operate ZTEL, but is normal procedure for most ZTEL macros.

```
@!/ENVMACRO.TEL/
        80wl
        124wy
        hk              @!/clean slate./
        13ei 10ei       @!/Insert CRLF and/
        h:p2            @!/put it in tr 2./
        @er/b:extract.adr/
        @b/LOOP/
@!/MESG/
        @x/
        ??  Typè RETURN, ^R or ^C.
        /
        @b/KEY/
@!/LOOP/@x/
        Align envelope in printer and type 'return'
         /
        @!/KEY/ ek4             @!/Get char from kybd./
                v4-3"e@b/DONE/' @!/^C gets you out,/
                v4-18"e@b/REPT/'@!/^R repeat last addr./
                v4-13"e@b/EVAL/'@!/CR prints next addr./
                @b/MESG/
        @!/EVAL/
                hk              @!/Clean slate./
                a               @!/Get a line, but/
                ve-1"e@b/DONE/' @!/branch if EOF./
                :ecl            @!/Put first char. in vr 1/
                v1-51"e@b/TWO/' @!/"3" = 2 lines, "4" = 3 lines./
                v1-52"e@b/THREE/' @b/ERROR/
        @!/TWO/
                2a
                @b/PRINT/
        @!/THREE/
                3a
        @!/PRINT/
                zj
                12ei            @!/Insert a form feed./
                j               @!/Go to top and/
                10<g2>          @!/get 10 CRLFs./
        @!/REPT/
                lwa             @!/Interprets form feed./
                hel             @!/List it out./
                0wa             @!/Turn off A flag./
                @b/LOOP/        @!/Next address./
        @!/ERROR/               @!/If not 3 or 4.../
                @x/
                File Construction Error- Ending processing.
                 /
          @!/DONE/
                  @x/
                  No more addresses- ending processing./
                  eq
```

The ENVELOPE macro is very similar to the MERGE macro, except that it does not read the letter file. As part of the initialization, the macro places a carriage return and a line feed in text register 2. Once it inserts a formfeed after the address, it jumps back to the top of the file and uses ZTEL's iteration feature to get the CRLF sequence ten times. Thus, the envelope is spaced up ten lines before the address is printed, an alignment convenience for the user.

While ZTEL provides an interesting array of conditional branch tests that are well suited to text processing (only the branch if zero and branch if alphabetic tests are used here), it does not support subroutine constructs. Therefore, you definitely won't be able to write structured-code macros! Be that as it may, the old fall-through constructs have served programmers well for a couple of decades now, and they still aren't obsolete. You might even consider ZTEL to be a sort of APL for text editing. The expressions are economical, powerful and all but incomprehensible to the uninitiated.

The above examples, while indicative of ZTEL's capabilities, do not really tell the whole story. I'm sure there are TECO and ZTEL macros out there that would make these look paltry by comparison. You may also be wondering why I'm bothering to praise an editor offered by a company that is out of business. Well, the software lives on. Computer Design Labs, (342 Columbus Ave., Trenton, NJ 08629) has the rights to ZTEL and is still selling it. And the price, the last time I checked, was only $79.95, which is quite reasonable for what it can do.

In conclusion, ZTEL is not everybody's ideal text editor. This example, though, shows that you can accomplish a lot without spending hundreds of dollars for a mailing list management system without tedious programming. It is a unique utility for unique applications. Maybe it belongs in your personal arsenal of system software. □

***

NOTE: All of the software mentioned in this article, with the exception of ZTEL, is available on disk from Sales Productivity Systems, Inc. for $25. This disk includes source code for MARGIN.COM, a complete documentation file and a "second generation" macro that will print addresses up to 9 lines long and subsitute a salutation in the heading and, optionally, in the body of the letter.

An 88-S4K Design Update

by W. Howard Adams

As a matter of economy, the least expensive memory available is the memory already owned. As the S-100 bus has evolved, some older pioneering equipment has lost its ability to function due to increasing bus expectations. Such is the 88-S4K, a dynamic memory manufactured by MITS for their Altair system. In its original design, this card will not self-refresh during multiple WAIT states such as North Star disk systems utilize during disk reads and writes.

This update can render some memory useful again, providing the 88-S4Ks began dropping bits ironically after a new peripheral was added to the system. Be advised, however, to judge personal technical ability by reading this article completely. While dynamic memory is not the evil thing static ram board manufacturers would have believed, they are more difficult to understand and troubleshoot.

Before correcting the design for multiple WAIT state refresh interference problems, the REV level must be assured. An addendum written by MITS and dated November 1976 relates a factory directive modifying how the T4 state, or the so-called 'transparent refresh' state is detected. (Yes, it was pioneering MITS who probably found the 'transparent refresh' capability that the modern dynamic memory boards brag about!) The original 88-S4K simply counted the free-running Phase 1 clock pulses after PSYNC to find T4; the MITS' modification calls for using the trailing edge of PDBIN to clock a flip flop, thereby providing an error free way of finding T4.

Table 1 relates the necessary MITS modification. Whether the multiple WAIT state modification to be discussed later is used or not, the MITS modification should be incorporated! There is at least one S-100 processor card on the market which allows WAIT state selections to be made on the processor board. Without the MITS mod, the WAIT state pulled by the processor card while an 88-S4K is selected will do a very nice job of completely ignoring the refresh cycle. From all indications the board will seem to have refreshed. Upon investigating the 4060 memory element it is readily seen that addresses are latched into the memory chip on the rising leading edge of chip enable. Assume the 88-S4K is selected, so the appropriate address bus signals are latched into the 4060s with chip enable. The processor card then pulls its own WAIT, but the memory card does not have the MITS mod.

When the old method of detecting T4 is used, the refresh addresses are presented to the 4060s, a refresh cycle is pulled and then cleared. Refresh is not seen by the 4060s because its chip enable is still asserted from the memory cycle still in progress; the 4060 never saw the refresh addresses. The MITS mod allows for the WAIT state (the instruction fetch cycle) and with the trailing edge of PDBIN allows a reassertion of chip enable which latches up the refresh addresses. A bona fide refresh cycle can therefore take place at a very safe place (i.e., after an instruction fetch, but only if IC X-3 has just previously gone high).

Table 1: MITS modification (refresh at trailing edge of PDBIN)

TRACE RUN:	CUT AT:	SIDE OF BOARD:
N-3 to N-10	N-3	Component Side
V-10 to L-10	V-10	Component Side
V-8 to M-1	V-8	Component Side
V-13 to N-8	V-13	Component Side

ADD JUMPERS:

V-10 to V-13 to V-14
K-3 to G-11
N-8 to G-12
V-8 to N-10
G-8 to M-1
G-13 to F-8

Simply ohm out the jumper list in Table 1 on every 88-S4K. If the jumper list ohms out, the MITS modification is already in the board's artwork, and the board is of a later REV level. No further consideration of Table 1 need be made.

Should the jumper list not ohm out, then the cuts in Table 1 should be made, then the jumper list must be

W. Howard Adams, 1590 So. Krameria, Denver, CO 80224.

incorporated. Please read the modification considerations in Table 3 before attempting to do any work.

After the MITS modification is made, the boards should be checked for refresh retention. With a system monitor, write the entire board with 0's. Do not access the board for several minutes. Now read the memory and assure that no bits have toggled to a logic 1 state. Repeat this process by writing all 1's, waiting a few minutes, and reading the memory assuring no bit has toggled to a zero. This check must be made of each board modified.

If a problem does occur, check the rework carefully — but go no further into the next modification until the MITS modification is working. The problem will only become more complicated. Please refer to Figure 1 for the necessary documentation to update a schematic for the MITS mod.

of a refresh cycle being pulled, and never by a free-running system clock!)

The second flaw is that PWAIT, the processor signal indicating a WAIT state is being pulled, is not a part of the 88-S4K's refresh arbitration scheme. The modification to be described now corrects both these flaws.

The modification criteria includes these five very important attributes necessary for reliable operation:

1. The MITS mod must have been incorporated. This mod frees up a gate necessary for this WAIT mod's use.

2. The memory card should be responsive to all signals designed into it by MITS. Namely, RUN will still be active, and the 88-S4K will still operate in an Altair system.

3. The board should not refresh during PWAIT if the board is currently selected. As pointed out earlier, there is at least one Z80 processor card which has wait state selectability on board. This criteria point is necessary. If

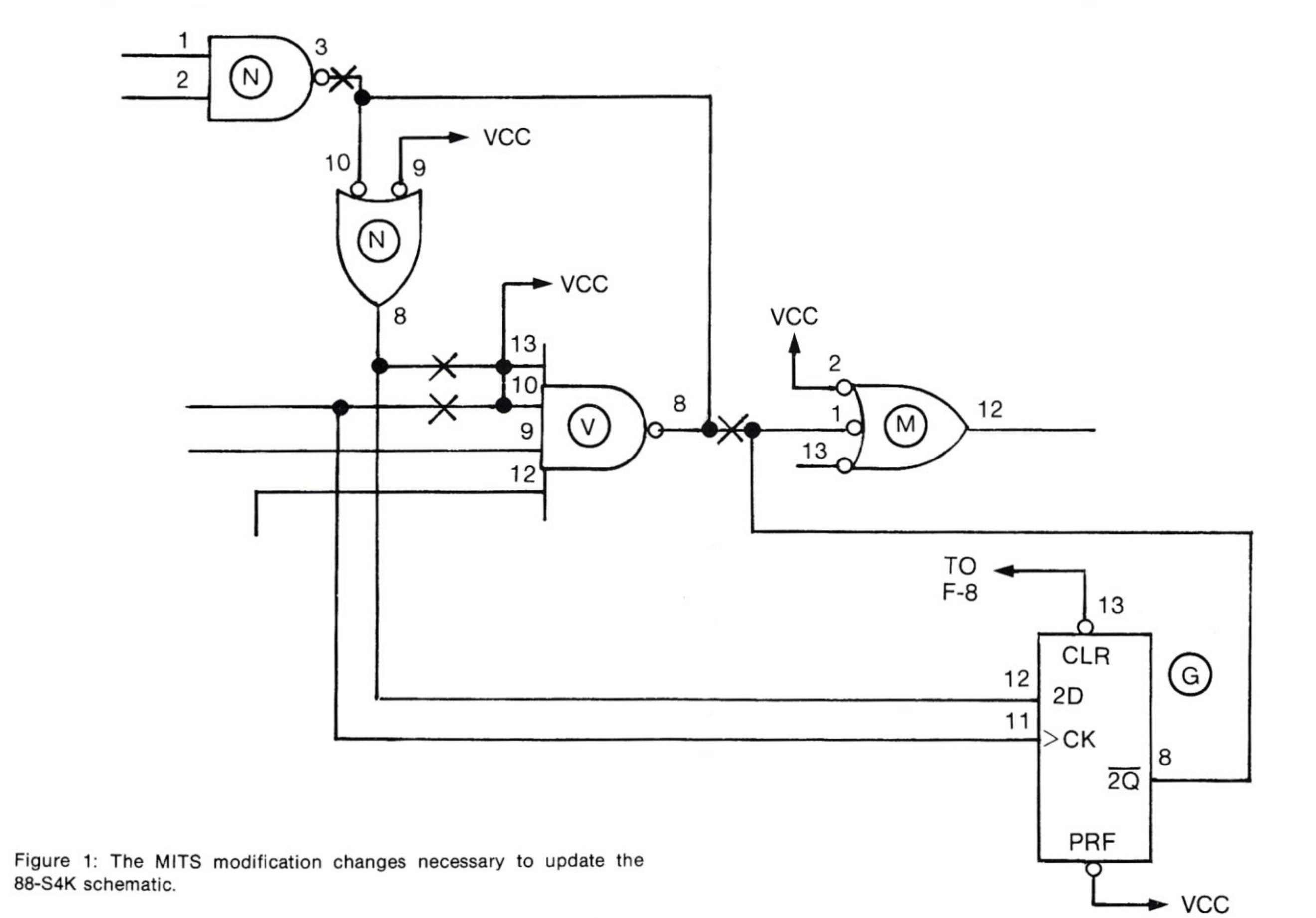


Figure 1: The MITS modification changes necessary to update the 88-S4K schematic.

With the MITS' mod now functioning, it's evident that the 88-S4K is a fine memory card, except that the stricter bus requirements imposed upon it reveals two major design flaws.

First, the refresh address counter is incremented by the free running Phase 2 system clock. Assume a refresh cycle is pending, but the processor is undergoing multiple WAIT states as the refresh address counter is being constantly incremented. The refresh is finally granted, but is pulled not at the address needing the refresh, but at an address higher up in the domain. (For information purposes, consider that modern design criteria would have the refresh address counter incremented as a result

the board were allowed to refresh during PWAIT, and the 88-S4K were selected, the memory elements would not respond. The same discussion for the MITS mod earlier is also appropriate here.

4. Only those ICs already on board will be used. No "dead bugs" or "piggy backing" is necessary nor is it desirable.

5. The modification should allow refreshing during PWAITs at a 32 micro-second rate, as per 4060 memory element specifications, and as per the existing design of the 88-S4K. (As a point of interest, dynamic memory is quite desirable because of its high density and low power. These elements should not be allowed to refresh in an

Design Update cont'd...

S-100 bus every T4 cycle because access rate determines power dissipation. Refresh should occur only during T4 AND at the memory element's specified refresh rate. There is only 'modern' S-100 64K dynamic now marketed which refreshes every T4; from a power dissipation point-of-view, this short cut refresh scheme is not as modern as the 'old' 88-S4K!)

Table 2: Multiple WAIT refresh modification.

CUTS: (to be made at IC pin noted)

T-12 to T-10 to GND	Component Side, under IC T (SEE TEXT)
N-1	Component Side
N-2	Component Side
M-2	Solder Side

ADD JUMPERS: (Signals defined and active state)

I/O-27 to N-1	(PWAIT-H)
V-6 to N-2	(BOARD SELECT -L)
N-3 to T-13 to T-12	(INVERTER)
T-11 to T-10	
S-3 to T-9	(REFRESH-H)
T-8 to M-2	(REFRESH DURING PWAIT-L)

NOTE: Observe N-3 was cut in the MITS MOD earlier, and should be open!

See Table 2 for the cuts and jumper list necessary for this modification. Please be familiar with Table 3 for modification considerations. Figure 2 shows the necessary documentation to update the schematic.

Table 3: Rework Considerations

1. Jumpers should be 26-30 gauge single stranded wire. Wire wrap wire would be a good selection.
2. Jumpers should be as short as possible while maintaining a good workmanship standard.
3. Jumpers should never be installed on the solder side of a board. These can easily snag when installing and removing boards in the system.
4. Jumpers should be run on the component side of the board, and under caps and resistors. This assures jumpers can't lift up and become a snag problem.
5. The jumper ends should be stripped just enough so a shallow 'U' can be formed to hook about the IC pin without shorting to the adjacent pin.
6. Watch soldering technique. Solder bridges are readily formed when soldering jumpers to ICs.
7. Make trace cuts cleanly and well defined by insisting upon a sharp X-ACTO blade.
8. Check all rework before reapplying power in a system.

NOTE: The IC at location T has traces to be cut located under the chip. Do yourself a favor and spend the twenty cents on a new 74LS00. Cut the old one out, desolder the IC's feedthroughs, cut the traces, and solder in the new IC. The results will be far better than trying half-hearted board surgery to save a readily available TTL chip!

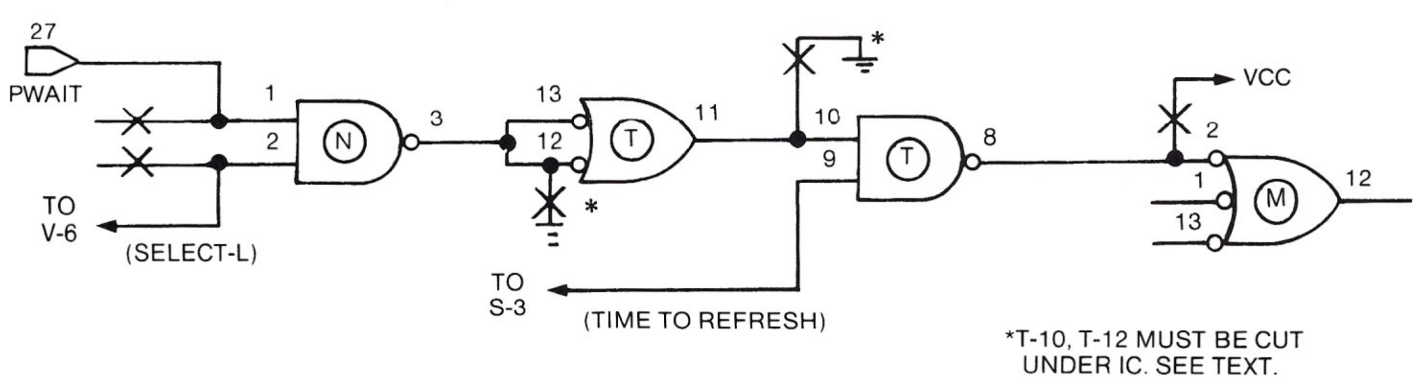
The modification is now complete, and the 88-S4K should function like its modern designed counterparts in every way. Put it to the refresh retention test spoken about earlier. Be absolutely sure that the board is not dropping bits. If there are problems, check the rework carefully. See the appendix for the theory of operation for the multiple WAIT state modification.

Properly functioning memory is the key paramount issue in computing. Eagles Computer Works offers a comprehensive memory diagnostic which was used to detect the 88-S4K problem and help find the solution. MEMDOC contains 14 tests and an oscilloscope loop which can turn your processor into a digital signal generator. It is available assembled at five locations on North Star single density diskette, or as a hex image dump ready for entry for $34.95. A manual is included. The non-hardware oriented computer user can qualify computer memory periodically, and allow troubleshooting by board swapping to assure good computing. □

APPENDIX — Theory of Operation for the multiple WAIT state modification. (Refer to Figure 2).

PWAIT is the processor signal relating that a wait state is in progress. Gate N-1 allows PWAIT to become part of the refresh arbiter logic. N-2 is low if the memory board is selected, and this disables PWAIT from asserting N-3 low. Gate T-13, 12, and 11 serves only as an inverter for T-10. T-9 is wired to S-3, and this line is active high at a 32 micro-second rate. This enables the PWAIT refresh cycles only at the appropriate time, assuring the board uses the least amount of power necessary to maintain memory element specifications. The output at T-8 is then active low for a PWAIT refresh, and this drives M-2. Any low on M-2, 1, or 13 is a refresh request command, and the modification is seen to work readily and that all text criteria points are met.

Figure 2: Adding DWAIT to the 88-S4K ARBITER LOGIC. This figure is necessary to update the schematic.



Book Review

Two Approaches to CP/M

by Steven H. Leibson

Digital Research's CP/M operating system is the standard for the microcomputer industry. It was introduced in 1975 and has since been adapted to almost every 8080, 8085 and Z80 based microcomputer available. CP/M will control the operation of your computer *if* you can figure out how to give the proper commands.

Until now, the only documentation available to acquaint you with the operation of CP/M has been the Digital Research manuals. Unfortunately, these manuals assume that you know quite a bit about computers, operating systems and other associated subjects. The organization of these manuals is sometimes quite difficult to follow; it can take a long time to find the section you need.

Six years later, books are appearing that will help you to understand the CP/M operating system and use it. Two of these are *Using CP/M: A Self-Teaching Guide* by Judi N. Fernandez and Ruth Ashley, and *The CP/M Handbook With MP/M* by Rodnay Zaks. I bought these books in a last attempt to understand CP/M. I succeeded.

Using CP/M: A Self-Teaching Guide, by Judi N. Fernandez and Ruth Ashley. John Wiley and Sons, Inc., 605 Third Avenue, New York, NY 10158. Softcover, 241 pp. plus index, $8.95. 1980.

The title is accurate; this book will really help you teach yourself how to use CP/M. The authors are specialists in writing computer-related education materials, and have published more than 100 courses. Their book on CP/M is a very readable tutorial in all of the facets of the CP/M operating system and utilities.

Each chapter of *Using CP/M* is written in the same format. A chapter starts with an introduction explaining what it will cover. This is followed by a list of objectives. The main body of each chapter is divided into "frames" composed of one or two paragraphs, followed by a series of questions about the material just covered. Chapters end with a self-test and a list of machine exercises to try. I found this format delightful after the usual computer user's diet of dry, cryptic and poorly written software manuals.

There are ten chapters in *Using CP/M*. The first is an introduction not just to CP/M but to computers, disks and buzz words as well. The basic structure of CP/M is covered. The FDOS, which is composed of the BIOS and BDOS, is explained so that the reader understands what parts of CP/M perform which tasks. A simple memory map is shown. This is the first time I have seen such a map presented so clearly. The chapter ends with a self test and machine exerises that concentrate on getting your CP/M system up and running.

Subsequent chapters cover command formats and control characters, Built-In (Intrinsic) Commands, and Transient Programs. Separate chapters are devoted to using the STAT, PIP, ED and SUBMIT utility programs. Each chapter is well written; explanations are clear. You may pace yourself, stopping whenever you need to. I found that I could set the book down and come back to it without problem. This is not always true of technical books.

ED, the CP/M text editor is key to most of the things you might do with CP/M. It is used for editing text to be printed, and for assembly language and high-level language source programs. Since this utility is so important, *Using CP/M* devotes three chapters to it: an introduction, instructions in editing existing files and advanced features. As with the rest of the book, the reader is led slowly at first, then more complex topics are covered until the entire subject has been explained.

The chapter on SUBMIT explains both the SUBMIT utility and the XSUB program supplied in CP/M versions 2.0 and later. SUBMIT allows you to create a file of commands using the text editor. This can save a lot of repetitive typing if there is a group of commands you use frequently. XSUB allows you to type information into the CP/M system in the middle of executing a SUBMIT file. Thus if you have a submit file that assembles a program, lists it and then runs it, XSUB could be used to accept the names of the source and object files. That allows you to write a general purpose submit file that can be varied when you run it.

There are three rather short appendices at the end of the book. The first, "Changing Device Assignments," is too superficial to be of use to the novice. Unless you can determine which drivers have been installed in your CP/M, appendix A won't help you. Appendix B summarizes the PIP parameters, including H,I, O, V and

Steven H. Leibson, 4040 Greenbrian Blvd., Boulder, CO 80803.

Book Reviews cont'd. . .

Z—which aren't explained in the book. Appendix C summarizes all CP/M commands.

I feel that *Using CP/M: A Self-Teaching Guide* is the best book on CP/M I have ever seen. Though I am not a novice, I got a truly thorough understanding of the operating system without feeling that I was being talked down to. Novices will find the book easy to read and all concepts used in the book are first explained. I highly recommend *Using CP/M* to anyone with a CP/M operating system.

The CP/M Handbook With MP/M, Rodnay Zaks. Sybex, 2344 Sixth St., Berkeley, CA 94710. Softcover, 317 pp. plus index, $13.95. 1980.

Dr. Rodnay Zaks is one of the most prolific authors writing about microcomputers today. It was to be expected that he would make a contribution towards filling the gap in CP/M documentation. *The CP/M Handbook With MP/M* is a very thorough user's manual on the CP/M operating system.

The first four chapters cover much the same material as *Using CP/M*. The two computer systems used as examples are the Processor Technology SOL and the Cromemco System 3. The SOL is defunct for the time being and the Cromemco is at the high end of the microcomputer product family. I don't feel these two systems were a good choice for teaching you how to get CP/M up and running.

The writing in *The CP/M Handbook* is less readable than that in *Using CP/M*. It's not poorly written, but the style used requires more concentration. The real value of the book for me is contained in the later chapters.

Chapter 5 is "Inside CP/M (and MP/M)." It gives a thorough explanation of how the operating system is organized, how it works and how to modify it for your needs. A turn-key system is offered as one reason you might want to modify CP/M. Turn-key systems assume very little knowledge on the part of the user. Turn the computer on and it asks you questions using menus until your input is no longer needed. The procedure is thoroughly documented. Examples such as this give the reader the confidence to go into the system and do some fiddling around.

A complete command reference for CP/M and MP/M is contained in chapter 6. The commands are listed alphabetically, with one command per page. Each page has a large print heading of the command and the upper right corner of the page shows whether the command is in CP/M version 1.4, version 2.2 or in MP/M. Each listing gives the format, arguments used with the command, a description, and explanation of how and why to use the command and some examples. A collection of CP/M commands, thoroughly explained and all in one place is very helpful to me since I don't use the commands frequently enough to have memorized them.

Chapter 7 is called "Practical Hints." It could have been called "Good Procedures for Maintaining Sanity in the Computer Room." Helpful suggestions for operating your computer, disk system and printer are given. These have nothing to do specifically with CP/M, but could come in handy if you want to get along with your computer.

The last chapter devotes three pages to conjecture about the direction of microcomputer operating systems in the future. It is followed by fifteen appendices. Some are useful and some aren't. They are:

Appendix A: *Common CP/M Error Messages*
Appendix B: *Hexadecimal Conversion Table*
Appendix C: *ASCII Conversion Table*
Appendix D: *ED Control Characters*
Appendix E: *ED Commands*
Appendix F: *PIP Device Names*
Appendix G: *PIP Keywords*
Appendix H: *PIP Parameters*
Appendix I: *CP/M (and MP/M) Commands*
Appendix J: *Command Editing Controls*
Appendix K: *CP/M Extension Types*
Appendix L: *Supplies (Checklist)*
Appendix M: *Computer Room Organization (Checklist)*
Appendix N: *Failure Checklist*
Appendix O: *Basic Troubleshooting Rules*

I felt that some of these appendices serve only to make *The CP/M Handbook* a little thicker, but others may find them more helpful.

I conclude that both *Using CP/M* and *The CP/M Handbook* belong in your library if CP/M is in your computer. I used the former to learn how to use the operating system, and the latter as a reference book. If you have to pick only one, *Using CP/M* is definitely the better written and, I feel, the most helpful. □

SOFTWARE DIRECTORY

Program Name: Z-80 FORTH
Hardware System: CP/M 2.0 or MP/M 1.0
Minimum Memory Size: 32K, may be reconfigured by user to take advantage of larger memory sizes.
Language: Z-80 Assembler
Description: Optimized fig-FORTH for Z-80 microcomputers. Uses standard CP/M random access disk files for screen storage. Extensions allow use of all CP/M functions. Distribution diskette includes: interpreter, line editor, screen editor, decompiler, debugging aids, utilities, several demonstration programs and over 100 Kbytes of documentation. Source code provided to extend vocabulary to meet FORTH-79 Standard.
Release: Currently available
Price: $50.00
Included with price: Eight inch soft-sectored, single-density diskette, 55 page user manual
Where to purchase it:
Laboratory Microsystems
4147 Beethoven Street
Los Angeles, CA 90066

Program Name: Wiremaster
Hardware System: Any Z-80 CP/M system
Minimum Memory Size: 48 Kbytes
Language: Written in ZSPL (Pete Ridley Software)
Description: Wiremaster is a software tool to aid in the design, layout, and construction of electronic hardware. Its inputs are easily derived from the schematic diagram and fed to Wiremaster in a CP/M text file. Outputs include a network map graphically showing all pins and wires, a wirelist sorted by lengths and levels, a parts list, and check lists that detect all wiring errors. The resulting information is then used for layout, error checking, wiring, component stuffing, and system debugging. Together with the schematic, this forms a complete and easily updated documentation package for an electronic product, and results in substantial savings of time.
Release: November 1980
Price: $75; manual $4
Authors: Jim and Gary Gilbreath
Where to purchase it:
Afterthought Engineering
7266 Courtney Dr.
San Diego, CA 92111

Program Name: Comstar
Hardware System: Double or quad density North Star System
Minimum Memory Size: 32K
Language: Machine Language
Description: Full compiler system for North Stas Basic (type 2) programs. Compiled programs run faster and original source is protected. Variable dimensions and disk file numbers must be decimal constants.
Release: December 1980
Price: $400
Included with price: Documentation and disk
Author: Allen Ashley
Where to purchase it:
Allen Ashley
395 Sierra Madre Villa
Pasdena, CA 91107

Program Name: Alpha APL Version 2.0
Hardware System: Alpha Micro
Language: Assembler
Description: Implementation of APL language functionally compatible with large mainframes. Runs as a multi-user system. Has system variables, system functions, I-beam, component I/O and other features. Runs under Alpha Micro operating system. Can be used with either ASCII or APL terminals. Assembler subrontines can be called directly. Source code for many external subroutines and assembler sub-routine development aids are included.
Release: November 1980
Price: $500; manual $25
Included with price: Disk and user's manual
Where to purchase it:
Softworks Limited
607 W. Wellington
Chicago, IL 60657

Program Name: D80
Hardware System: CP/M with 8" drive
Minimum Memory Size: 24K (CP/M System Size)
Language: 8080, 8085, and Z80 compatible machine code
Description: D80 is a flexible and powerful disassembler for 8080, 8085, and Z80 machine code programs. It accepts a machine code program from disk or memory and produces a disk file of the disassembled code using either the Intel or Zilog mnemonics. In the created source file, D80 will produce a map of the object file, symbol table, up to four types of symbolic labels, and uses the ORG, EQU, and ND pseudo-opcodes. The created source code is not held in memory, therefore very large programs can be disassembled. Also, all information about a disassembly can be stored in a disk file and then reloaded at a later time to pick-up where you left off. Also runs in interrupt and MP/M environments.
Release: November 1980
Price: $85.00(disk and manual); $75.00 (disk only); $10.00(manual only)
Included with price: Disk contains D80 with sample disassemblies and original source code of an included utility
Author: Dennis Gallagher
Where to purchase it:
DG Software
P.O. Box 1035
Iowa City, IA 52244

Program Name: FORTH
Hardware System: 8080/8085/Z-80 CP/M or CDOS
Minimum Memory Size: 24K
Description: Enhanced version of FIG* FORTH for CP/M or CDOS users. Supplied on CP/M format diskette, ready to run. A FORTH style editor with 20 commands is included, as well as a virtual memory subsystem or disk I/O. These allow the user to easily create new FORTH software which is permanently stored on diskettes, then loaded when needed. The user may also make permanent additions to the resident FORTH vocabulary. A Z-80/8080 assembler is also included, allowing the user to create new FORTH definitions which compile directly into machine code. All Z-80 or 8080 instructions may be used. The IF. . . ELSE . . . UNTIL, and BEGIN . . WHILE . . . control structures may be included in assembler definitions; these will automatically compile into appropriate machine code. Other enhancements include an interleaved disk format that minimizes the time required for disk access. A 1024 byte disk block may be read or written in as little as 1/6 second. Eight of these blocks are maintained in RAM for immediate access and automatically swapped with others on the disk as they are needed. FIG FORTH was originally defined by the FORTH INTEREST GROUP and is very close to the FORTH-79 international standard.
*FORTH Interest Group
Release: October 1980
Price: $95 IBM compatible 8" single density disk (other disk formats $110)
Included with price: CP/M compatible disk, users manual suitable for beginners or experienced users, and shipping.
Authors: Mitchell E. Timin, and FIG
Where to purchase it:
Mitchell E. Timin Engineering Company
9575 Genesee Avenue, Suite E2
San Diego, CA 92121

Program Name: ENTRY
Hardware System: CP/M
Minimum Memory Size: 52K bytes
Language: Microsoft Basic
Description: The UNIVERSAL DATA ENTRY System provides interactive definition of data files complete with CRT format and prompts. Once defined, data entry becomes a 'fill in the blank' operation. A wide range of validity checks reduces error to a minimum.
Release: Currently available
Price: $195; License Agreement Required
Included with price: Diskette, manual, examples, support
Author: The Software Store
Where to purchase it:
The Software Store
706 Chippewa Square
Marquette, MI 49855

Program Name: Z-80 DES
Hardware System: Z-80 based
Minimum Memory Size: 16K
Language: Z-80 assembler
Description: High Speed Implementation of the NBS data encrypton algorithm. Modular and user oriented. Fully documented source code supplied for the algorithm. Special run-time package for TRS-80. Database protection, password scrambling, telecommunications security from remotely-accessed data files. Protect sensitive or proprietary files. Easily adapted by user for custom uses.
Release: April 1979
Price: $34.95 + $9.95
Included with price: Documented source code listing
Author: ITM
Where to purchase it:
Interface Technology of Maryland
P.O. Box 745
College Park, MD 20740

Program Name: Infomedia System (IMS) Ver 1.1
Hardware System: S-100 (Vector MZ), Micropolis Drives, CRT with cursor controls
Minimum Memory Size: 48K
Language: MDOS Micropolis Basic of CP/M-CBasic2
Description: A menu driven data base and file management system, plus report writer. All user created data base formats, data files, and report formats are listed on a system directory. Allows up to 20 user defined data formats and reports per disk. The user can define up to 24 numeric or alphanumeric fields for file records. File functions include: Create, Delete, Duplicate, Add/Modify, and List. Record functions include: Add, Update, Delete, Scan, List, Sort, Compact, and Create, Delete, Add/Modify, List and Print. User selectable column or label format, titles, fields, subtotals, total, and printing of selectable records. Fields may be specified as mathematical calculations. of other fields (+, -, *, &, /) are supported. IMS stores up to 20 different files and up to 999 records per disk with full file management capability.
Release: January 1980
Price: $195
Included with price: Eight programs and users manual send $3 for brochure and sample print-outs, or $195 plus $2 shopping. CA residents add 6% sales tax.
Where to purchase it:
Investment Analysis Systems
P.O. Box 282
Palos Verdes Estates, CA 90274

Program Name: Property Analysis System (PAS) Ver 2.11
Hardware System: S-100 (Vector MZ), Micropolis Drives, CRT with Cursor controls
Minimum Memory Size: 48K
Language: MDOS-Micropolis Basic or CP/M-CBasic2
Description: Property Analysis System, for both residential and income property. Analyzes the effects of financing, income, expenses, depreciation, taxes, and inflation on the return on investment for nine years. PAS produces a three-page report consisting of initial conditions and a nine year projection of property value, liabilities, equity, gross income, expenses, net income and cash and percentage return on investment before and after taxes. PAS also provides percentage return on equity

New Products cont'd...

for before and after taxes. PAS was designed for ease of use without prior computer experience—field tested since mid 1979. All data can be changed at any time with the effects immediately displayed for review, allowing the user to model an investment property and ask "What if?" PAS stores up to 20 different properties per disk with full file management capability (create, delete, read, & write/update).

Price: $195

Included With price: Five programs and 60 pg. users manual, send $3 for brochure and sample printouts, or $195 plus $2 shipping. CA residents add 6% sales tax.

Where to purchase it:
Investment Analysis Systems
P.O. Box 282
Palos Verdes Estates, CA 90274

Program Name: HayesSys

Hardware System: PolyMorphic Systems 8813 single density

Minimum Memory Size: 8K (uses system RAM)

Language: 8080A Machine Language

Description: HayesSys is a D.C. Hayes MICROMODEM 100 terminal operating system consisting of two programs and a modified version of the operating system which allows operation of the D.C. Hayes MICROMODEM 100 board. The system includes complete software control of all parameters, auto dial, and all other features of the D.C. Hayes board except auto answer. The system includes the ability to download to disk files and to send files from disk, as well as the ability to log the incoming text to a printer. Available for Exec/78 and for Exec/83 (specify which).

Release: September 1980

Price: $85 postpaid

Included with price: Installation instructions & disk.

Where to purchase it:
Ralph E. Kenyon Jr.
145-103 S. Budding Ave
Virginia Beach, VA 23452

Program Name: DF

Hardware System: CP/M

Minimum Memory Size: 30K CP/M Configuration

Language: Compiled from C (BDS version)

Description: Shows all differences between two versions of a printable file, such as a source program. Re-synchronizes and continues after reporting deletions and insertions. Comparisons are on a line basis, but you can specify the line delimiter and a character to be ignored so that PCL and other text block files can be compared. As compiled, it can handle differences as large as 8K in files of any length.

Release: Available now

Price: $20 (check or money order)

Included with price: CP/M COM and C files on CUTS cassette or a paper listing. Or send a Micropolis Mod II diskette. Can arrange for conversion to standard 8" diskette for $5 extra. Modem?

Where to purchase:
Richard Greenlaw
251 Colony Ct.
Gahanna, Ohio 43230

The Sixth
Trenton Computer Festival
TCF-81

April 25 & 26, 1981
at TRENTON STATE COLLEGE
Trenton, New Jersey

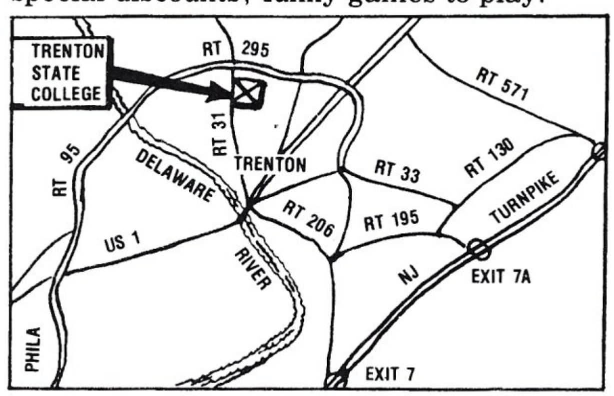
10 AM to 6 PM
Saturday, 25th

10 AM to 4 PM
Sunday, 26th

Super Outdoor Flea Market
Surplus computer gear, bargains galore, over 5 acres of space (5/spot, no electricity).

Indoor Commercial Exhibit Area
90 exhibitor booths showing newest products; special discounts; funky games to play.

Forums, Talks & Seminars
Meet the leading experts and hear sessions on robots, computer music, amateur radio, etc.



Convenient to NY, PA, MD & DEL
Easy to get to; free parking for over 6,000 cars.

Free Short Courses on Sunday
Hundreds of Door Prizes • Banquet Saturday Night

For additional information call 609-771-2487

Admission $5 - Students $2

Banquet $10. **Avoid disappointment – pre-register for the banquet!** Send your check for $10 per person to: TCF 81, Trenton State College, Trenton, NJ 08625

Sponsored by:
Amateur Computer Group of New Jersey
Philadelphia Area Computer Society
Trenton State College Digital Computer Society
Dept. of Engineering Technology, Trenton State College
I.E.E.E., Princeton Section

NEW PRODUCTS

New 8/16-Bit Memory Board By SSM Microcomputer Products

SSM Microcomputer Products has announced the MB10—a 4-MHz, 16Kbyte static RAM memory board that supports the proposed IEEE S-100 bus standard. For those applications requiring bank switching, the MB10 is jumper selectable to any I/O address with enable/disable on power-up or reset (Cromemco-compatible). The board uses the popular, low power 2114L memory chip, but the on-board logic is designed to upgrade easily to the use of faster memory chips as they become available in the future. Option selection is via wire-wrap jumpers.

Suggested retail price of $399. SSM Microcomputer Products, 2190 Paragon Drive, San Jose, CA 95131. (408) 946-7400.

---

64 Channel D/A Converter Card

The Model SB-64 D/A converter board has 64 output channels accomplished by multiplexing a single 8-bit converter. Each output drives a 1K ohm load, and has ±.4% absolute accuracy from 0V to 10.625V. Address decoding allows memory-mapped I/O to any 256 byte sector in the 65K. IEEE specifications for S-100 Bus interface devices are fully met. Options available are: 10 digital outputs, fast 12-bit DAC with 3 microsecond settling, ±15V tracking regulator with 600ma output, and 24-bit address decoding.

The price is $514; Digital Multi-Media Control, 2338 Patterson #12, Eugene, OR 97405.

ADA Manual Available

A manual on ADA, the defense department's(DOD) new standard programming language is now available from Pony Express Computer Services, 100 West 57th Street, New York, N.Y. 10019.

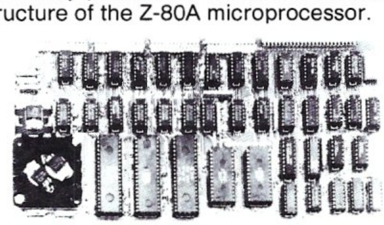
In addition to the ADA manual, Pony Express offers a complete update service which keeps subscribers fully informed of ADA developments and program availability.

The ADA, manual costs $100 exclusive of shipping. The complete update service is extra. Call (212) 265-7446 for more information.

New Cromenco Serial Communications Interface Card

Cromemco announces the new QUADART serial communications S-100 interface card, with four serial channels. Any channel can support Asynchronous or Synchronous Byte mode (Bisync) or Bit mode (SDLC) communication protocols under software control, with handshaking provided for modems. Baud rates for each may be software-selected from 0 to 300K baud (asynchronous to 19,200 baud).

The QUADART also supports the powerful internally-prioritized vectored interrupt structure of the Z-80A microprocessor.

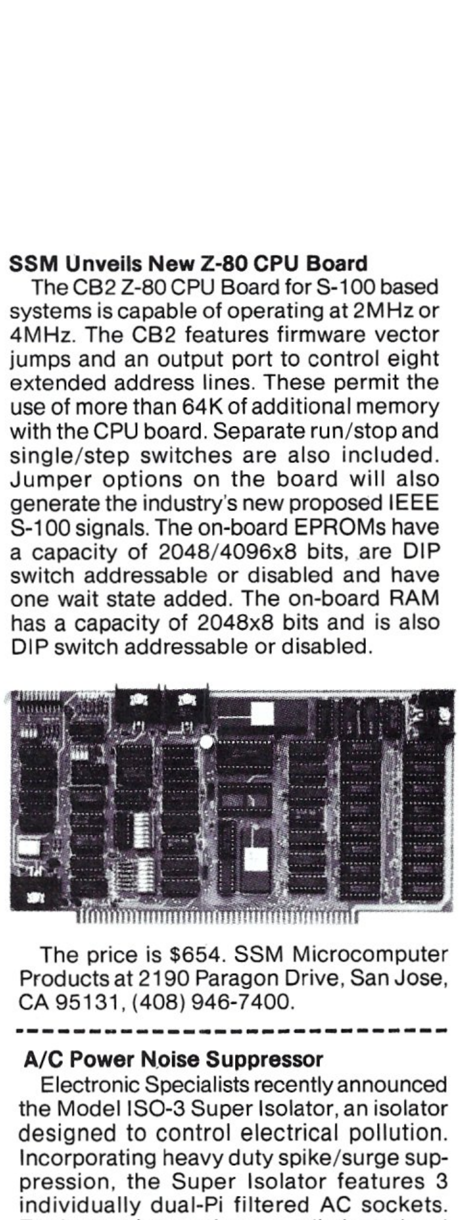


The QUADART has four interval timers each having periods as small as 4.00 microseconds. Up to three timers can be cascaded to provide a 1.000-second time interval.

The QUADART Serial Communications Interface (Model QDRT) is $595 from: Cromemco, Inc., 280 Bernardo Avenue, Mountain View, CA 94043, (415) 964-7400.

SSM Unveils New Z-80 CPU Board

The CB2 Z-80 CPU Board for S-100 based systems is capable of operating at 2MHz or 4MHz. The CB2 features firmware vector jumps and an output port to control eight extended address lines. These permit the use of more than 64K of additional memory with the CPU board. Separate run/stop and single/step switches are also included. Jumper options on the board will also generate the industry's new proposed IEEE S-100 signals. The on-board EPROMs have a capacity of 2048/4096x8 bits, are DIP switch addressable or disabled and have one wait state added. The on-board RAM has a capacity of 2048x8 bits and is also DIP switch addressable or disabled.



The price is $654. SSM Microcomputer Products at 2190 Paragon Drive, San Jose, CA 95131, (408) 946-7400.

A/C Power Noise Suppressor

Electronic Specialists recently announced the Model ISO-3 Super Isolator, an isolator designed to control electrical pollution. Incorporating heavy duty spike/surge suppression, the Super Isolator features 3 individually dual-Pi filtered AC sockets. Equipment interaetions are eliminated and disruptive, damaging power line pollution is controlled. The Super Isolator will control pollution for an 1875 watt load. Each socket can handle a 1000 watt load. Price is $94.95. Electronic Specialists, Inc., 171 South Main Street, Natick, MA 01760. Phone: (617) 655-1532.

New QT System Offers Up to 16 Megabyte Storage Capacity

QT Computer Systems has introduced their new QT System™ which provides standard two Megabyte storage, plus availability of up to 16 Megabytes storage. The new system includes mainframe, two 8" disk drives, double-sided double-density 5 1/4" & 8" floppy disk controller, power supply, fan and Televideo 920C terminal.

This 4MHz/Z80/CPU System includes such features as: 48K Dynamic Memory (expandable to 64K); 2K Monitor program and Disk Bios on 2716 EPROM; RAM/ROM/PROM in any combination up to 8K; two RS-232C Serial I/O ports; two Parallel I/O ports; Hard disk compatible; Real time clock; CP/M 2.2 Operating System; Power-on/Reset jump to monitor program.

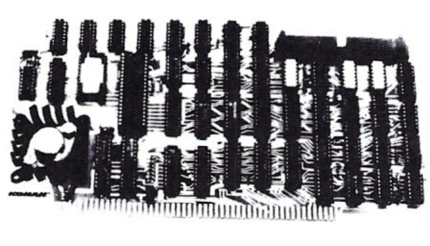
The one Megabyte single-sided, double-density system sells for $4,295, while the two Megabyte double-sided, double-density unit is priced at $4,995. The units are assembled, tested and burned in at the factory. Each system includes documentation and has a one year warranty against defects in material and workmanship.

For additional information, contact QT Computer Systems, Inc., 15620 South Inglewood Avenue, Lawndale, CA 90260. Call (213) 970-0952. The toll-free number (except California, Alaska and Hawaii) is 800-421-5150.

Konan Unveils New Controller

Konan's KNX-500 is a new hard disk controller which is software compatible with the Alpha Micro AM-500*. KNX-500™ interfaces with the S-100 Bus, plugs in the same as AM-500 and uses the same software drivers. It will control four drives from one board and is designed to work with most 5440 drives, typically 10 to 20 megabyte, such as Western Dynex, CDC Hawk, Diablo, and Pertec.

Bundled with a Western Dynex drive, KNX-500™ will be offered as subsystem—including cable, disk pack, drive, and controller—for $4500 O.E.M., quantity one. (Additional disk packs are $95 each.) Alone, KNX-500™ is priced at $1200 O.E.M., quantity one, with quantities of 200 or more priced at $695 each. The 50 pin cable is 10' length, sells for $80, with a total daisy chainable length of approximately 40'. KNX-500™ will have a CP/M support package and full documentation. Each will be backed by continuing diagnostic support and will carry Konan's one year warranty.

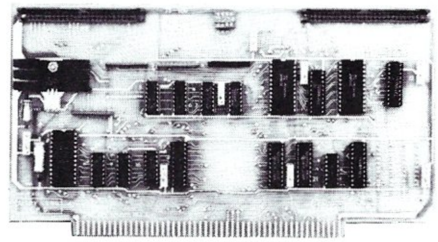


Konan Corporation, 1448 N. 27th Avenue, Phoenix, Arizona; 800-528-4563.
*trademark of Alpha Micro Systems.

S-100 Controller Supports IBM-Compatible 9-Track Tape

The Alloy Engineering TS-100 Controller permits any IBM-compatible 9-track formatted tape drive, including the new Cipher Microstreamer™, to be interfaced with the S-100 Bus. MTIP software provides backup/restore under CP/M™ and MP/M™ operating systems. An .ASM file containing the tape control subroutines is included to facilitate customization.

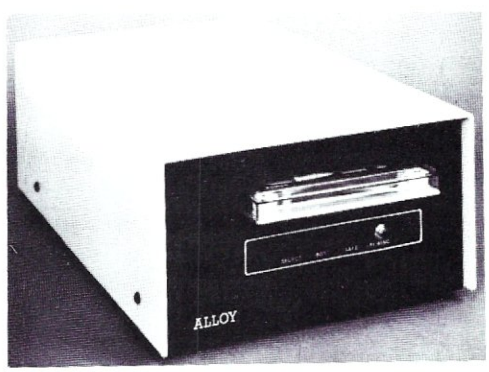
Linked via 16 I/O ports with switch selectable base addressing, the Alloy Engineering TS-100 Contrroller handles up to 25 IPS @ 1600 BPI or 50 IPS @ 800 BPI, and also supports bidirectional spacing at 100 IPS. Software is distributed on single sided, single density 8" floppy disk in CP/M format.



The TS-100 Controller is $600, including software, and dual 8 ft. data cables with 50 pin connectors. Contact: Alloy Engineering Company, Inc., Computer Products Division, 85 Speen Street, Framingham, MA 01701 (617) 620-1710.

Winchester Disk Backup Cartridge Tape Subsystem.

The Alloy Engineering TIP (Tape Interchange Package) is an S-100 compatible cartridge tape subsystem and software utility that permits transfer of programs and data files from a Winchester disk to an easy-to-handle 13.4 MB 1/4" tape cartridge. Facilitating off-premises data base storage and shipping, the subsystem features comprehensive menu driven software, and links to the Winchester disk under Digital Research CP/M and MP/M Operating Systems.

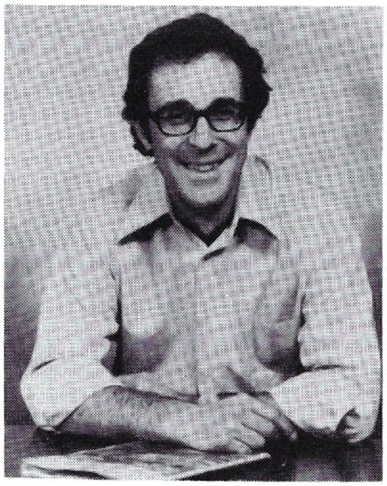


TIP consists of a DS-100 Controller, 6400 BPI cartridge tape drive, and rack or table mount power supply. Software is distributed on a single sided, single density 8" floppy disk in CP/M format.

TIP is $2100 each (rack mount) and $2200 each (table mount) in lots of 25. Contact Alloy Engineering Company, Inc., Computer Products Division, 85 Speen Street, Framingham, MA 01701; (617) 620-1710.

A REMARKABLE MAGAZINE

creative computing



David Ahl, Founder and Publisher of Creative Computing

"The beat covered by Creative Computing is one of the most important, explosive and fast-changing."—Alvin Toffler

You might think the term "creative computing" is a contradiction. How can something as precise and logical as electronic computing possibly be creative? We think it can be. Consider the way computers are being used to create special effects in movies—image generation, coloring and computer-driven cameras and props. Or an electronic "sketchpad" for your home computer that adds animation, coloring and shading at your direction. How about a computer simulation of an invasion of killer bees with you trying to find a way of keeping them under control?

Beyond Our Dreams

Computers are not creative per se. But the way in which they are used can be highly creative and imaginative. Five years ago when *Creative Computing* magazine first billed itself as "The number 1 magazine of computer applications and software," we had no idea how far that idea would take us. Today, these applications are becoming so broad, so all-encompassing that the computer field will soon include virtually everything!

In light of this generality, we take "application" to mean whatever can be done with computers, *ought* to be done with computers or *might* be done with computers. That is the meat of *Creative Computing*.

Alvin Toffler, author of *Future Shock* and *The Third Wave* says, "I read *Creative Computing* not only for information about how to make the most of my own equipment but to keep an eye on how the whole field is emerging.

Creative Computing, the company as well as the magazine, is uniquely lighthearted but also seriously interested in all aspects of computing. Ours is the magazine of software, graphics, games and simulations for beginners and relaxing professionals. We try to present the new and important ideas of the field in a way that a 14-year old or a Cobol programmer can understand them. Things like text editing, social simulations, control of household devices, animation and graphics, and communications networks.

Understandable Yet Challenging

As the premier magazine for beginners, it is our solemn responsibility to make what we publish comprehensible to the newcomer. That does not mean easy; our readers like to be challenged. It means providing the reader who has no preparation with every possible means to seize the subject matter and make it his own.

However, we don't want the experts in our audience to be bored. So we try to publish articles of interest to beginners and experts at the same time. Ideally, we would like every piece to have instructional or informative content—and some depth—even when communicated humorously or playfully. Thus, our favorite kind of piece is acessible to the beginner, theoretically non-trivial, interesting on more than one level, and perhaps even humorous.

David Gerrold of *Star Trek* fame says, "*Creative Computing* with its unpretentious, down-to-earth lucidity encourages the computer user to have fun. *Creative Computing* makes it possible for me to learn basic programming skills and use the computer better than any other source.

Hard-hitting Evaluations

At *Creative Computing* we obtain new computer systems, peripherals, and software as soon as they are announced. We put them through their paces in our Software Development Center and also in the environment for which they are intended—home, business, laboratory, or school.

Our evaluations are unbiased and accurate. We compared word processing printers and found two losers among highly promoted makes. Conversely, we found one computer had far more than its advertised capability. Of 16 educational packages, only seven offered solid learning value.

When we say unbiased reviews we mean it. More than once, our honesty has cost us an advertiser—temporarily. But we feel that our first obligation is to our readers and that editorial excellence and integrity are our highest goals.

Karl Zinn at the University of Michigan feels we are meeting these goals when he writes. "*Creative Computing* consistently provides value in articles, product reviews and systems comparisons . . . in a magazine that is fun to read."

Order Today

To order your subscription to *Creative Computing*, send $20 for one year (12 issues), $37 for two years (24 issues) or $53 for three years (36 issues). If you prefer, call our toll-free number, **800-631-8112** (in NJ 201-540-0445) to put your subscription on your MasterCard, Visa or American Express card. Canadian and other foreign surface subscriptions are $29 per year, and must be prepaid. We guarantee that you will be completely satisfied or we will refund the entire amount of your subscription.

Join over 80,000 subscribers like Ann Lewin, Director of the Capital Children's Museum who says, "I am very much impressed with *Creative Computing*. It is helping to demystify the computer. Its articles are helpful, humorous and humane. The world needs *Creative Computing*."

creative computing

ATTN: RHODA
P.O. Box 789-M
Morristown, NJ 07960
Toll-free **800-631-8112**
(In NJ 201-540-0445)

New Record Retrieval System For PL/I-80

Digital Research announces BT-80™, a comprehensive single user record retrieval system. Based on the B-Tree index organization technique, BT-80 is designed for use in PL/I-80 applications where single or multi-keyed access to data records is required.

BT-80 facilities may be accessed from PL/I-80 or assembly language application programs via two BT-80 system procedures which handle all data file and index maintenance. The BT-80 system also includes utilities that provide access to commonly needed facilities at the command level.

BT-80 runs on CP/M version 2, MP/M™ and CP/NET™ operating systems, and requires the PL/I-80 Run-time Library and LINK-80 Linkage Editor to operate.

For information contact: Digital Research, Post Office Box 579, 801 Lighthouse Avenue, Pacific Grove, CA 93950: (408) 649-3896.

Amateur Radio Interface Card

The Snow Micro Systems AR.01 card is a versatile S-100 Bus card designed to facilitate the merging of amateur radio and the personal computer. It contains a digital and an analog section. The digital section may be used as a serial I/O port at RS-232 levels at 45.5, 50, 56, 74, 75 and 110 Bauds using ASCII or BAUDOT codes. The analog section comprises a modem and can be set up for any commonly used tone pairs. The AR.01 is normally supplied set up for the standard 2125-2295 tone pairs commonly used in the amateur bands.

The board can thus be used as a serial I/O port driving an ASCII or BAUDOT printer, as an interface to existing RTTY equipment such as an ST-5, as a full RTTY terminal unit, as an interface between a radio receiver and a computer, for morse code reception software, and as an RTTY modulator/demodulator (TU) for an existing printer in a non-computer environment.

The AR.01 is available as follows: Bare board and documentation $45, kit $245, assembled and tested $349. For information contact: Snow Micro Systems, Inc., P.O. Box 1704, Silver Spring, MD 20902, Phone: (301) 622-2194.

Station Controller Card

The Snow Micro Systems SC.01 Station Controller card is a versatile S-100 card designed for process control applications. It can be used in almost any control type of environment such as Energy Management, Alarm Monitoring and Amateur Radio.

The specifications of the card include: S-100 bus compatible, hardware time of day 24 hour clock, 6 Reed relays with isolated contacts, 7 Analog input channels (5 channels have a maximum input level of 2 Volts, 2 channels are scaled via 100K potentiometers. The Analog channels are multiplexed by a 4051 CMOS switch, and converted to digital levels by a 3 1/2 digit ADC device (14433). The time of day clock uses a low cost 3.579 MHz color TV crystal and has provision for battery back up. The PC. board may be used with several kinds of reed relays.

The SC.01 is available as a bare board for $45, as a kit for $245 and assembled and tested $349. For information contact: Snow Micro Systems, Inc., P.O. Box 1704, Silver Spring, MD 20902, Phone: (301) 622-2194.

SSM Introduces Eprom Board

SSM Microcomputer Products has introduced the MB8A* 1K—16K EPROM Board. The new MB8A provides sockets to support up to sixteen 2708 EPROMs. Simply by removing the 1K EPROMs the board can be disabled in 1K increments. The user can overlay RAM and ROM at the same address in any desired increment. This provides increased flexibility when the board is used with RAM boards equipped with Phantom Disable.

For complete information and specifications on the MB8A EPROM Board, contact your local computer store or call or write SSM Microcomputer Products, 2190 Paragon Drive, San Jose, CA 95131, (408) 946-7400.

8" Floppy Subsystem for North Star

John D. Owens Associates, Inc. announces an 8 inch floppy subsystem for North Star users. In response to the need of North Star users for greater auxiliary storage and their need to transfer files from 5 1/4 inch media to 8 inch media or vice versa, a software-hardware package, fully integrated, is offered for a total cost of $1,910. The hardware requires simple, plug-in installation and is completely compatible with North Star hardware. The package is designed to operate in single or double density on 8-inch media in IBM compatible format.

The hardware includes a Tarbell double density controller, dual Shugart 800R drives housed in a cabinet with power supply, fan and all by all necessary cabling. The software, produced by Dynamic Microprocessor Associates, is accompanied by a detailed, comprehensive 80 page manual. An export model strapped for 50 hz. 220 volts is also offered for an additional $100.

The software alone is $150 from John D. Owens Associates, Inc., 12 Schubert Street, Staten Island, NY 10305. Telephone: (212) 448-6283 or (212) 448-2913. Overseas callers can reach the export desk only at (212) 448-6298 or TWX 710 588 2844.

Advertiser Index

Software Shops

Systems And Services: Custom software in assembly language & Basic for S100 Z80/8080 CP/M systems. Contact Buz Koenig C/O Systems And Services, P.O.B. 961 Hurst, TX 76053 (817) 268-2938 Eve & W.E.

Theta Labs, Inc.: Software in Basic, Pascal and 8080/Z80 assembly. Process control, automatic test equipment, factory time-keeping systems. Consulting or turn-key. Box 20337, Dallas, TX 75220; (214) 241-1090.

Provar Inc.: Specialists in multi-user and networking systems. Currently marketing an MP/M utility package. Experienced in Assembler, Pascal, and Basic. We also design and build process control systems. Contack Mark Winkler, 6217 Kennedy Ave., Hammond, Ind. 46323; (219) 844-0370.

Maximum Functions Minimum Space Maximum Versatility Minimum Price:

SYSTEM SUPPORT 1

If you ever need to crunch numbers, time intervals or events, create sequenced programs, check for power interruption, shuttle information through an RS-232C serial port at up to 19,200 Baud, tell your computer to do something at 02:47:37 AM on March 24th of 1982, service 15 levels of interrupts (8 vectored interrupts from the bus and 7 on-board interrupts), or need EPROM or battery-backup RAM to accomplish any of the above. . .and want to do all this while fully conforming to all IEEE 696/S-100 standards. . .**System Support 1** is here.

Call **(415) 562-0636**, 9 AM to 5 PM PST, for the name of a **CompuPro** retailer near you.

System Support 1 prices: **$295 Unkit, $395 A/T, $495** qualified under the **C**ertified **S**ystem **C**omponent high-reliability program. Math processor and ROM/RAM optional at extra cost. Prices shown do not include dealer installation and support services.

CompuPro™
OAKLAND AIRPORT, CA 94614

division of
GODBOUT ELECTRONICS