S-100
$2.00
U.S.A.

MICROSYSTEMS™

NOV/DEC 1980 VOL. 1/NO. 6

IMPLEMENTING THE CP/M* IOBYTE FUNCTION

See Pages 22-26

Also in this Issue

Modifications to CBasic-2 by Andy & Ben Galewsky .... **Page 28**
Running North Star DOS and CP/M Together by Randy Reitz......... **Page 32**
Directory Program for CP/M Systems by Mark M. Zeiger .... **Page 38**
S-100 EPROM Programmer Using the Intel 8255 PPI by Ted Croal ... **Page 46**

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

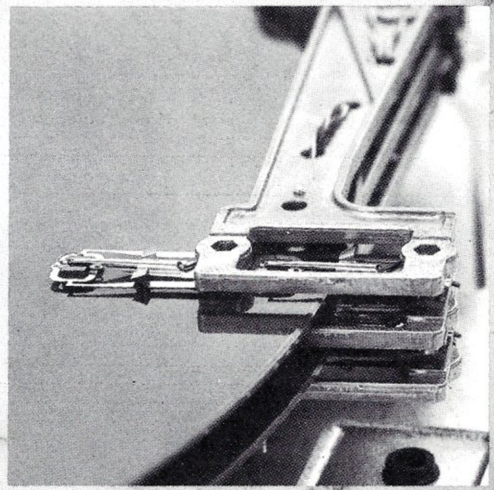
and more

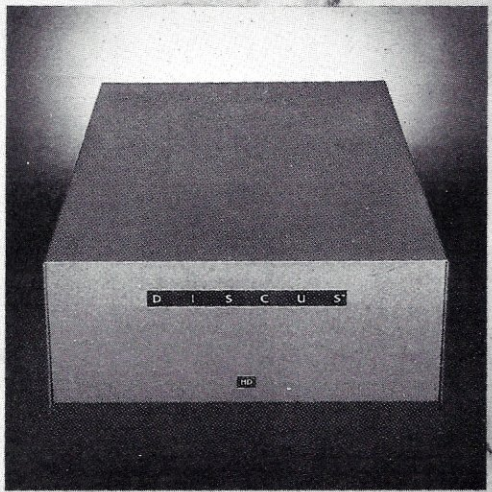
Complete Table of Contents on Page 3

26 MEGABYTES

$4995.

DRIVE A HARD BARGAIN!





Suddenly, S-100 microcomputer systems can easily handle 100 million bytes. Because Morrow Designs™ now offers the first 26 megabyte hard disk memory for S-100 systems—the DISCUS M26™ Hard Disk System.

It has 26 megabytes of useable memory (29 megabytes unformatted). And it's expandable to 104 megabytes.

The DISCUS M26™ system is delivered complete—a 26 megabyte hard disk drive, controller, cables and operating system—for just $4995. Up to three additional drives can be added, $4495 apiece.

The DISCUS M26™ system features the Shugart SA4008 Winchester-type sealed media hard disk drive, in a handsome metal cabinet with fan and power supply.

The single-board S-100 controller incorporates intelligence to supervise all data transfers, communicating with the CPU via three I/O ports (command, status, and data). The controller has the ability to generate interrupts at the completion of each command to increase system throughput. There is a 512 byte sector buffer on-board. And each sector can be individually write-protected for data base security.

The operating system furnished with DISCUS M26™ systems is the widely accepted CP/M* 2.0.

See the biggest, most cost-efficient memory ever introduced for S-100 systems, now at your local computer shop. If unavailable locally, write Morrow Designs,™ 5221 Central Avenue, Richmond, CA 94804. Or call (415) 524-2101, weekdays 10-5 Pacific Time.

*CP/M is a trademark of Digital Research.

MORROW DESIGNS™

Thinker Toys™

S-100
MICROSYSTEMS
TM

Volume 1 Number 6 November/December 1980

Staff

Editorial correspondence is welcomed and should be sent to: Sol Libes, c/o MICROSYSTEMS, Box 1192, Mountainside, NJ 07092.

Subscriptions are $9.50 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

Information Master.....14
Bill Machrone

The Talos Digi-kit-izer Digitizing Tablet Kit.....17
Jon Bondy

The CP/M* Connection.....22
Chris Terry

Modifications to CBasic-2.....28
Andy & Ben Galewsky

North Star Topics.....32
Randy Reitz

Directory Program for CP/M Systems.....38
Mark M. Zeiger

An S-100 Eprom Programmer Using the Intel 8255 PPI.....46
Ted Croal

Departments

Editor's Page.....4
Letters to the Editor.....8
News & Views.....10
CP/M* Bus.....11
New Products.....52
Software Directory.....54
Advertiser Index.....56

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

S-100 MICROSYSTEMS (ISSN#0199-7955) is published bi-monthly by Microsystems, P.O. Box 789-M, Morristown, NJ 07960. Controlled circulation pending at New Berlin, Wisconsin.

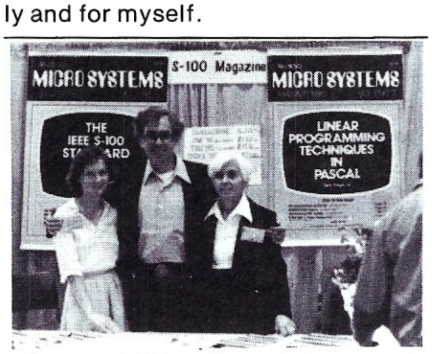
POSTMASTER: Send address changes to: MICROSYSTEMS, P.O. Box 789-M, Morristown, NJ 07960.

by Sol Libes

EDITOR'S PAGE

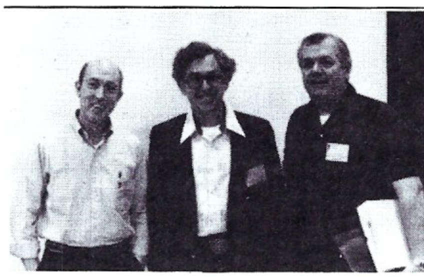
With this issue of MICROSYSTEMS I am retiring from the position of publisher and will only be the editor of the magazine. In other words, I have decided to sell MICROSYSTEMS. The magazine will become a subsidiary of Creative Computing.

I found it necessary to sell the magazine because it had gotten too big for me to handle. MICROSYSTEMS has become a great success. But success comes at a high price. I found that publishing a magazine was an all-consuming undertaking that affected my family relationships, health and job. I was at the point where to continue the magazine meant quitting my job and hiring a staff. My basement looked like a combination of warehouse and production department; we would have needed to move into offices. I examined the alternatives and feel that I have made the right decision—for the readers, my family and for myself.



MICROSYSTEMS staff at PC/80 show in Philadelphia. From left to right: Claudette Moore, Sol and Lennie Libes.

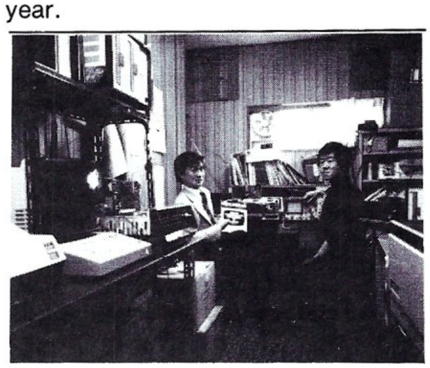
In addition, I am not a businessman. I find it difficult to hire and fire. Also, I like my job and want to keep it (for those that do not know, I am a teacher in a Community College). I found that the major portion of my time was spent in publishing activities rather than editing activities. I cannot describe the time spent dealing with subscribers, dealers, advertisers, printers, typesetters, mailing service and worst of all the U.S. Mail Service (or as Jim Warren says: "The U.S. Snail Service"). This last one alone must have driven my blood pressure up by at least 10 points.



George Morrow, Sol Libes and Bill Godbout get together to discuss the S-100 scene at the PC/80 show in Philadelphia.

Being relieved of the publishing load I will be free to spend more time on editing. I will have total editorial control of MICROSYSTEMS, assuring that it will continue with the same editorial focus as before. All other tasks will now be handled by a professional publishing staff. I am pleased that Dave Ahl, owner of Creative Computing, has organized the staff of MICROSYSTEMS as a separate entity from the Creative Computing organization. Claudette Moore is now the Managing Editor. She handles the day-to-day running of the magazine, including advertising and promotion. Further, she assists me in editorial matters and oversees the production of the magazine. Actual production, typesetting and artwork, subscriptions, warehousing, etc. will be handles by the Creative Computing staff.

Dave Ahl has also undertaken a large promotional effort to increase the circulation of the magazine and its distribution through dealers. These efforts were just not possible with the limited facilities and staff that I had access to previously. Thus, I expect MICROSYSTEMS readership and advertising to increase substantially in the coming year.

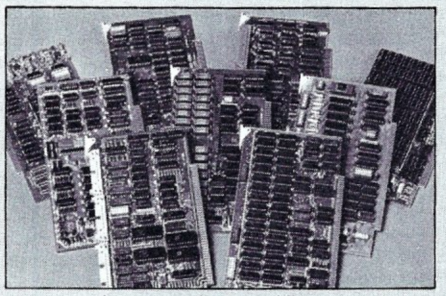


MICROSYSTEMS readers in Japan. Y. Mitazaki (left) and T. Nagayama with a room full of S-100 systems.

You are already seeing results from this new change. First, the size of the magazine has been increased by eight pages to provide more editorial content and advertising space. I expect that size of the magazine will continue to grow with MICROSYSTEMS advertising support. Secondly, we hope to increase the amount of editorial

At Intersystems, "dump" is an instruction. Not a way of life.

(Or, when you're ready for IEEE S-100, will your computer be ready for you?)



We're about to be gadflies again.

While everyone's been busy trying to convince you that large buses housed in strong metal boxes will guarantee versatility and ward off obsolescence, we've been busy with something better. Solving the *real* problem with the first line of computer products *built from the ground up to conform to the new IEEE S-100 Bus Standard.* Offering you extra versatility in 8-bit applications today. And a full 16 bits tomorrow.

We call our new line Series II.™ And even if you don't need the full 24-bit address for up to 16 megabytes (!) of memory right now, they're something to think about. Because of all the performance, flexibility and economy they offer. Whether you're looking at a new mainframe, expanding your present one or upgrading your system with an eye to the future. (Series II boards are compatible with most existing S-100 systems and *all* IEEE S-100 Standard cards as other manufacturers get around to building them.)

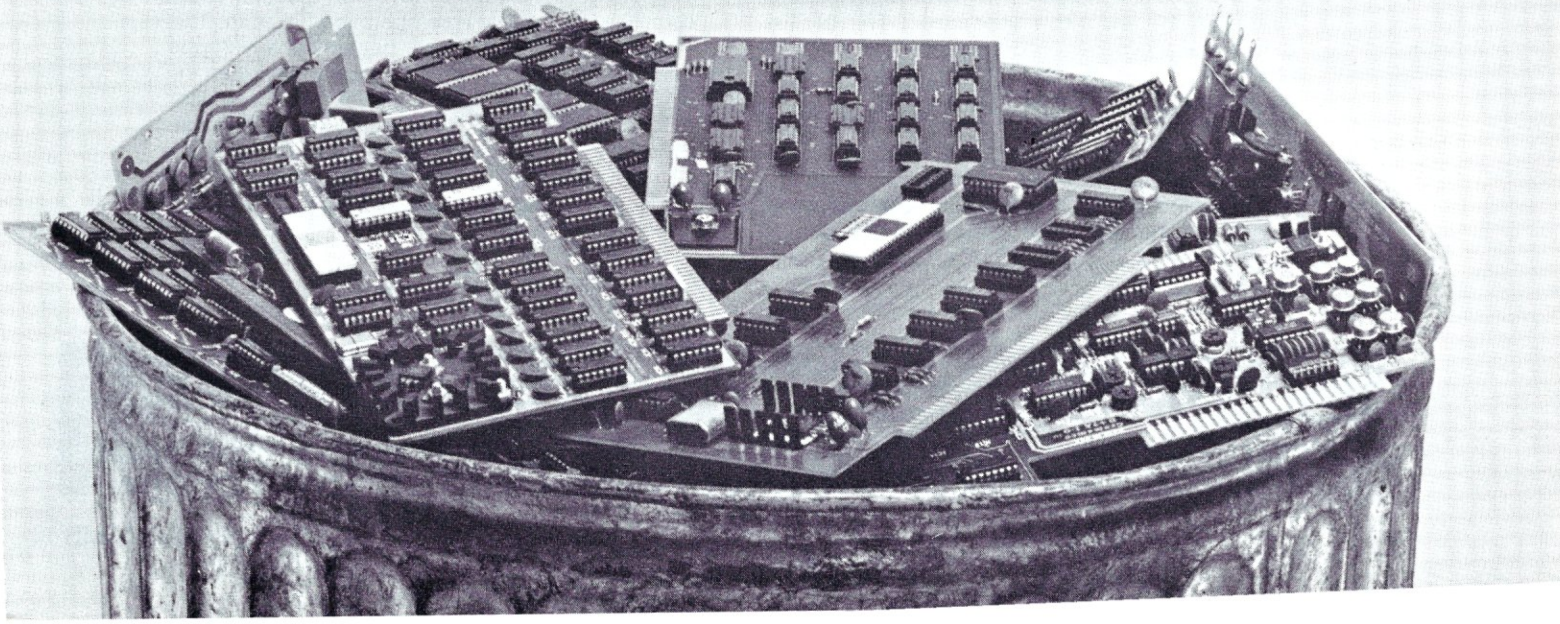
Consider some of the features: Reliable operation to 4MHz and beyond. Full compatibility with 8- and 16-bit CPUs, peripherals and other devices. *Eight* levels of prioritized interrupts. Up to 16 individually-addressable DMA devices, with IEEE Standard overlapped operation. User-selectable functions addressed by DIP-switch or jumpers, eliminating soldering. And that's just for openers.

The best part is that all this heady stuff is available *now*! In our advanced processor—a full IEEE Bus Master featuring Memory Map™ addressing to a full megabyte. Our fast, flexible 16K Static RAM and 64K Dynamic RAM boards. An incredibly versatile and economical 2-serial, 4-parallel Multiple I/O board. 8-bit A/D-D/A converter. Our Double-Density High-Speed Disk Controller. And what is undoubtedly the most flexible front panel in the business. Everything you need for a complete IEEE S-100 system. Available separately, or all together in our new DPS-1 Mainframe!

Whatever your needs, why dump your money into obsolete products labelled "IEEE timing compatible" or other words people use to make up for a lack of product. See the future now, at your Intersystems dealer or call/write for our new catalog. We'll tell you all about Series II and the new IEEE S-100 Bus we helped pioneer. Because it doesn't make sense to buy yesterday's products when tomorrow's are already here.

InterSystems™

Ithaca Intersystems Inc.,
1650 Hanshaw Road/P.O. Box 91,
Ithaca, NY 14850
607-257-0190/TWX: 510 255 4346

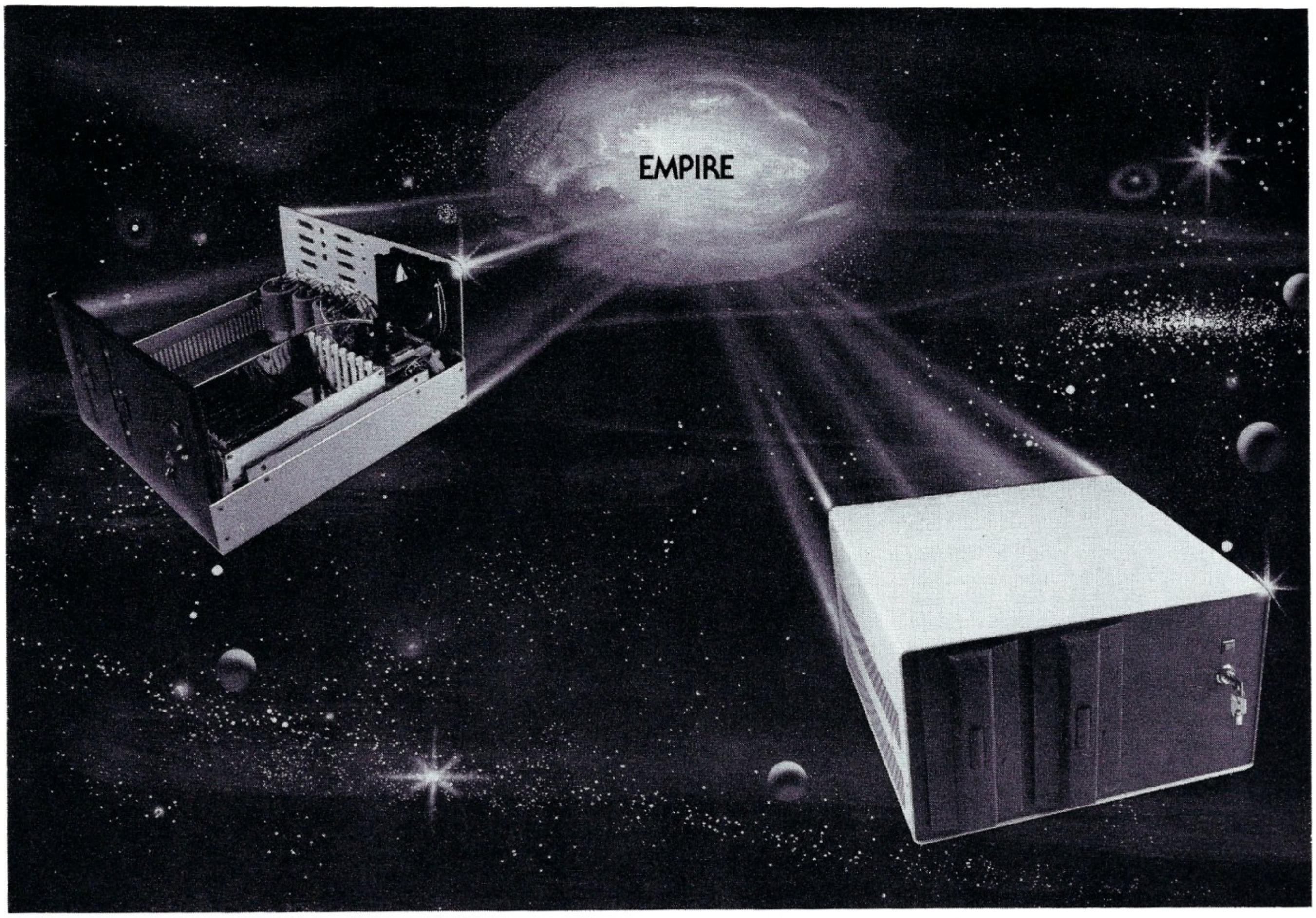


Editor's Page cont'd...

space devoted to product reviews. In this issue we feature one hardware and one software product review. I will try to continue and increase the number of these product reviews in every issue. We are looking for more product reviewers, particularly software product reviewers.

If you have suggestions for improving MICROSYSTEMS do not hesitate to write or call me. (You will find my telephone number on the Table of Contents page.) I wish to thank all the readers, authors and advertisers who helped me get MICROSYSTEMS off the ground, making it a viable publication. I would also like to thank the following friends who helped unselfishly: Russell Gorr, Fred Gohlke, Jon Bondy, Randy Reitz, Jake Epstein, Bill Yarnall, George Lyons, Marty Nichols, Bob Stewart and Larry Stein. I am also indebted to my wife, Lennie and my son and daughter, Don and Susan who shared in the "dirty work". (I should mention that my wife is a professor of mathematics and my children at doctoral candidates at the University of Rochester and M.I.T.) □

The Empire has expanded!

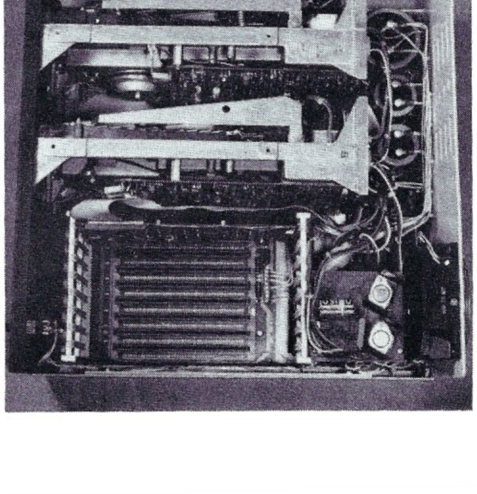


New Mainframe opens more areas for development

In one quantum leap Tarbell has expanded its popular Empire (the vertical disk subsystem) into a full line. This entire series now encompasses 5 variations. Each one contains different components so the S-100 system designer, hobbyist, or serious business user can arrive at the exact custom state he wants and needs.

The basic Empire still includes two Shugart or Siemens 8" disk drives; the compact cabinet with fan and power supply; a Tarbell floppy disk interface; CP/M*; Tarbell BASIC; the necessary cables, connectors and complete documentation. Naturally, it's fully assembled and Tarbell tested.

The new, top of the line Empire contains the basic model's components with the Tarbell design-approved Mainframe. Beside the 8-slot S-100 motherboard with an active terminated bus, there's a cardcage with card guides and a double-density interface.



You're the master of your Empire

You can call the shots in the Empire. Tarbell's made sure of that by offering them as complete subsystem packages . . . or, as separate units. For example, the mainframe may be ordered with 1, 2 or no drives. Whichever way you go, however, you always get the reliability of Tarbell tested components and leadership-engineering.

To get control of your own Empire, see your quality computer store for quick delivery. Or, contact us for dealer locations or further information.

CP/M is a trademark of Digital Research.

Tarbell Electronics

950 Dovlen Place - Suite B
Carson, California, 90746
(213) 538-4251 / 538-2254

LETTERS TO THE EDITOR

Dear Editor:

Small C is a compiler by Ron Cain. It was printed in source form in DDJ, May '80. The article discussed Ron's interest in C, dealing with Tiny-C, but then deciding he wanted a compiler. He did not have "by choice" CP/M. He tried developing the compiler under Tiny-C, but just didn't have the room (in memory). So he "borrowed Unix time" and got it going.

It generates 8080 assembler code directly, and requires a run time library to handle file open, char. read and write (disk and console) 16 bit multiply and divide, and several arithmetic and logical comparison subroutines, and indirect memory reference instructions for fetching local variables.

The compiler, being based on Tiny-C, is quite limited: No FOR, DO, CASE, etc. Just "while" like "teeny-C" as we call it. But it does handle "i + + " and a few things I don't believe (but don't recall) T-C did.

Of course, the big news is that the compiler compiles itself (in about 20 minutes on a 2MHz 8080). It is quite readable code. Seems quite bug-free, although I stumped it with an obscure "pointer-memory addr" which erroneously doubled the memory addr, thus was pointer minus 2* memory addr, which was garbage. I solved it by saying that an int = the pointer. I then had int minus memory reference.

I am having a real ball with Small-C, have gotten the author's permission to put it in the CP/M U.G. and just last night, a friend with a 6809 got it running with my help. Johy Byrns took the source from me (he wrote a compatible modem transfer program for his '09), modified the code generator to put out macros (i.e. machine independent code) and then sent it back to me.

I compiled his compiler with mine, then compiled his compiler with itself, thus outputting macros. I sent them to him, he reviewed the output, asked for some changes (due to 8080 putting low-byte-first in a word, while the '09 does high-byte-first). I compiled it again, sent it to him, and he got it going.

I am very interested in pursuing the addition of a code optimizer to the end of it. Right now it generates "bullet proof" code, i.e. the expression processor does "PUSH H" whenever more is to be processed in the expression. Thus (for Global variable "k") k = k + 1;

generates:

```
LHLD k
PUSH H
LXI H,1
POP D
DAD D
SHLD k
```

If instead of directly producting 8080 code, pseudo code were output, say into a 25 element table, than a pattern recognizing optimizer could see:

PUSH H	and replace	XCHG
LXI H,xxx	it with:	LXI H,xxxx
POP D		

Several more passes thru the optimizer for this block of pseudo code could theoretically recude it to:

```
LHLD k
INX H
SHLD k
```

Just blue skying right now, but you see the "place" for Small-C, namely a super facility for hacking with compilers. Thanks for writing it to Ron Cain, for DDJ for publishing it, and to Alan McNeil and Jim Kirns for getting the copy that I have keyed in and running.

The Code Works is selling a copy on CP/M disks for $15 + postage. The address is: Box 550, Goleta CA 93017. Phone: (805) 967-0905.

Small C doesn't have "for" but it appears it could be added in about an hour. The code for "while" is very similar. Only a routine to test things would have to be changed in the original code, as that routine tests for a condition in " ("")", where as the "for" the test condition isn't in parens.

More easily perhaps to get done, would be adding "do", as that ends in a while test, which is in parens.

Ward Christensen
Dolton, Illinois

Dear Editor:

During the coming year we can expect a proliferation of systems utilizing the Z8000 microprocessor. These systems can be expected to upgrade existing Z80 systems.

Indications are at this time that UNIX will probably become the standard DOS, replacing the present dominance of CP/M.

This presents something of a "retrofit" or old disk file usage problem since UNIX in Z8000 code does not fit on two 8" floppy tracks and it doesn't like CP/M style disks.

A number of CP/M style DOS systems are in the field. Since the authors of these systems have the source code, they can probably be cross compiled into Z8000 code relatively easily. To name a few of these systems, we have K3, SDOS, LDOS, TPM, OS and PDOS — for which we have the source.

CP/M offers a definite advantage over other microcomputer DOS systems in the way it does a DOS entry via address 0005 with the appropriate commands and data in certain registers. This "standardization" enables Basic's, Pascal, Fortran, Cobol, PL, "C" and other high level languages to be written explicitly for CP/M disk file control. This is not to mention all the utility programs — assemblers, debuggers, etc. which are dependent upon the same DOS entry.

Digital Research indicates they have no interest in Z8000 software at this time. Microsoft has a Basic using UNIX ready for release and will also probably offer UNIX. This may still leave the industry with a lot of CP/M files it cannot digest.

Cross Compiling Z80 to Z8000 code results in about a 2.5/I increase in byte size. This is too much for two tracks of single density 8" disk to hold a DOS of CP/M's present complexity. However, re-editing code to take advantage of Z8000 instructions and byte doubling up in messages make it appear that a DOS of the size of CP/M 2.2 with adequate Z8000 I/O handling can be accomodated on two tracks of 8" double density. CP/M type file structures, directories, etc. can therefore be preserved. Format programs can accomodate double density system tracks and single or double data.

Therefore, we propose:

1) Authors of DOS systems and other software useable on Z80's who wish to compile Z8000 code should agree upon a standard.

2) The standard should return CP/M style file structure and entry concepts so that present disks with files will still be readable—at least in single density. There is no double density standard which could apply.

3) Instead of 8080/Z80 registers, the ZILOG Cross assembler registers should be used.

4) AF = 0, BC = RH1, DE = RH2, HL = RH3, IX = RHD, IY = RHE, SP = RHF. C, E, L are RL parts of their respective registers. This information is given in the ZILOG translator users guide. Where any intermediate register is required, Register RH 4 can be used. The remaining registers are open to use as well.

At the zero Address:

0000 -0005 Warm Boot

0006 -000B DOS Entry (Functional) (Replaces CP/M) 0005

000C - I/O Byte

000D - Default Drive

000E -004F Interrupts, Restarts* and/or Scratch

0050H - 007F Default file control.

Note that 6 bytes (3 words) are reserved even though the jump mode used may require only two. This allows the software writer to use any mode and to use segments and offsets.

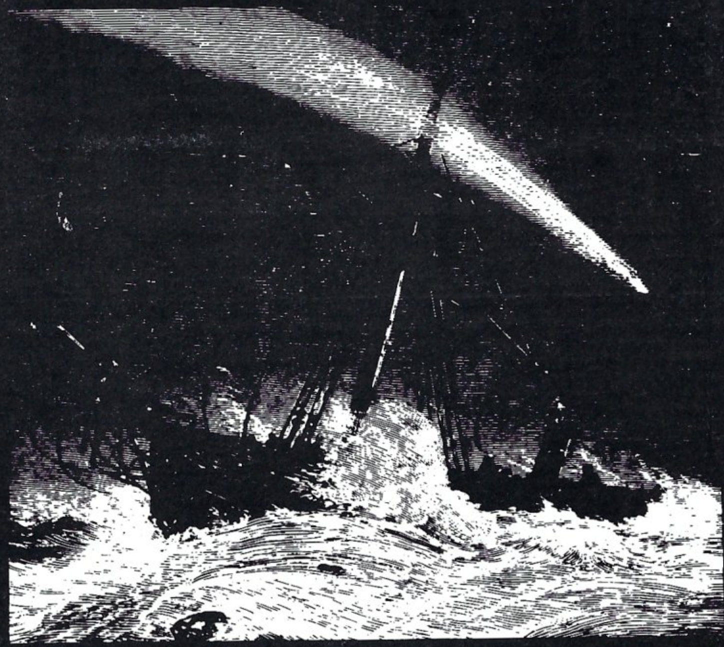
Restarts are not valid in Z8000 code but equivalent addressing concepts can be used.

We have contacted Microsoft about their Basic and they have informed us they see no difficulty in using this concept. The change from UNIX is relatively simple for them. You can expect to see Microsoft Basic at some of the upcoming shows.

The CDL (TDL/XITAN) Basic should be relatively easy to compile after changing the Math routines. The same should be true of their ZTEL and TOP. We have written a MACRO assembler cross compiler which we will make available to them. At this time there are no readily available Z8000 assemblers and text editors other than those sold by AMD software movement than helping it.

Softech (UCSD Pascal) is not interested. No matter, there are much better Pascals around.

National Multiplex Corp.
Middlesex, NJ



NEWS & VIEWS

by Sol Libes

NEW S-100 PRODUCTS SHOWN AT WESCON

The WESCON/80 show, held September 16-18th in Anaheim CA, saw the introduction of some really new S-100 products.

Morrow Designs announced a "Unix-Equivalent" operating system running on a Z-80 based system with a special S-100 CPU board. The board has the Z-80, a floating point processor circuit, interrupt system and task supervisor/memory management circuitry.

Godbout Electronics introduced a digital multiplexer board using an 8085 that provides multiplexed communications with any portion of the computer. It takes over the tasks of transferring storage or I/O memory via DMA cycle stealing, which halts the main processor to transfer data. The device acts as a bus master and uses only about 1% of the available bandwidth. Godbout also showed an S-100 RAM board using 20-pin rather than 18-pin sockets. The board will accommodate the standard 18-pin 4K RAM ICs or the new 20-pin shadow RAMs (EEPROMs) expected to become available next year. Shadow RAMs retain data during power interruptions.

Cromemco showed their new $995 System Zero, a four slot S-100 system using a Z-80 CPU, up to 8K ROM, 1K RAM, one serial and three parallel ports in one slot. The remaining three slots can be used for a floppy/hard disk controller card, 64K RAM card, modem card, etc. Cromemco also showed their new $595 "Quadart" communications card designed for multiplexed communications. The card permits multiple operations, allowing the system to communicate with an IBM mainframe via a variety of user-definable protocols.

Measurement Systems & Controls showed a 64K dynamic RAM card with each of 4 memory banks selectable via I/O ports. Up to 256 banks may thus be selected. The board contains 4 LEDs to indicated when a bank is being accessed.

SOFTWARE REVIEW PUBLISHED

John C. Dvorak is publishing a monthly four page review of software. The primary emphasis is on North Star system software. A subscription is $5/year (foreign $10). The sample issue I received contained a great deal of useful information. John also distributes a wide selection of North Star software. For more information write: Software Review, 704 Solano Ave, Albany, CA 94706.

370 MICRO USER GROUP FORMING

This may be the first user group to be formed before the fact rather than the usual practice of after the after. Rumors are floating around that Intel, Motorola and Fujitsu are working on a microprocessor chip set that executes the IBM 370 instruction set. Mokurai Cherlin (Box 1131, Mt. Shasta CA 96067) has decided to form a user group in anticipation of the 370-micro. Membership is $10 for individuals and $25 for companies. You will get a newsletter and access to a computer database.

There is no doubt that if such an IC is released someone will put it on the S-100 bus. Mokurai points out that "There is more free, public domain software for the 360 than you can buy for all present micros together, including operating systems, languages, tools and applications. There have probably been about a thousand languages brought up on a 360 or 370 over the last 16 years. IBM itself has a list of more than 2500 public domain programs available for a copying and media charge — the universities and users generally have an awful lot more. You don't have to wrestle with IBM operating systems and Job Control Language if you don't want to, because here are numerous software houses that have improved on IBM in all sorts of ways."

MICROPOLIS USER GROUP FORMED

The Micropolis Users Group (MUG) has been formed to maximize the use of the Micropolis supplied software. The group is compiling a directory of all software which runs on MDOS or Micropolis Basic without requiring a second operating system (such as CP/M). The group would appreciate suppliers of such software informing them of their products.

Membership is $12/year for 12 monthly newsletters. Micropolis Users Group, c/o Buzz Rudow, 604 Springwood Circle, Huntsville AL 35803.

UCSD USER'S SOCIETY FORMED

A User's Group has been formed to support UCSD products such as PASCAL. Membership is $20/year. UCSD System User's Society, Chip Chapin Secretary, c/o Softech Microsystems Inc., 9494 Black Mountain Road, San Diego CA 92126.

SOL SOFTWARE AVAILABLE

Processor Technology Company, manufacturer of the SOL computer went out of business over a year ago but its software lives on. PROTEUS (the SOL Users Group) is now distributing the Processor Technology software with hopes of soon having it available on 8" soft sectored disk. They are furnishing the complete source code for PTDOS, Basic, ALS8, Games and Test Routines. The charge range from $50 to $95. PROTEU, 1690 Woodside Drive, Suite 219, Redwood City CA 94061; tel: (415) 368-3331.

RUMOR

It is rumored that early next year Intel will introduce an enhanced 8251 serial I/O chip. It will include baud rate generator, five timer/counters, two full parallel ports and a complete interrupt controller; all in a 40 pin DIP.

MICROSYSTEMS BACK ISSUE NEWS

The premier issue (JAN/FEB) of MICROSYSTEMS has been reprinted and can be ordered by writing: MICROSYSTEMS, Box 789-M, Morristown, NJ 07960. The price is $5.00. Vol. 1/NO.3 (MAY/JUNE) is out of stock. We do not forsee a second printing.

This is the first of what I hope will be a regular column in S-100 Microsystems. The column will serve as a forum on CP/M Readers are encouraged to send in questions about CP/M, which the author will attempt to answer. The questions can be both technical and non-technical. Until the questions are received, here is some news about the CP/M world.

The CP/M* Bus

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

SIG/M-ACG/NJ Generates First Three CP/M Disks

The Special Interest Group for CP/M (SIG/M) of the Amateur Computer Group of New Jersey (ACG-NJ) has produced three disks of CP/M software containing different versions of the ever popular Adventure game. The software has been placed in the public domain. The disks are the following:

SIG/M0001 - Adventure (350 pts) 8080 object code, for 48K RAM system.
SIG/M0002 - Adventure source code in Microsoft Fortran.
SIG/M0003 - 'Super' Adventure (550 pts) Z80 object, for 52K RAM system.

The disks are available for copying at meetings of the ACG-NJ CP/M User Group and at the New York Computer Club Flea Markets. Disks must be supplied and a $1/disk contribution to the club is requested.

The ACJ-NJ is really not prepared to furnish copies via mail. Therefore, it will accept orders only from other bona-fide clubs. Only one copy will be shipped to a club (8" single density); the club will then be responsible for providing copies for its members. It is asked that the club make a contribution to the ACG-NJ of $4/disk plus $2 to cover mailing and handling. ACG-NJ, 1776 Raritan Road, Scotch Plains, NJ 07081.

CP/M User Group News

The Chicago Area Computer Hobbyist Exchange (CACHE) is presently engaged in putting together twenty new 8" disks for the CP/M User Group to distribute. CACHE is the group that put together the last batch of CP/MUG disks. This is strictly a volunteer effort; no one is CACHE will be paid for their efforts. Jim Mills is coordinating CACHE's efforts.

Jim says that these new disks will include the following, among other things:

RATFOR adapted for Fortran-80
Household Budget Management (CBasicII)
Math Primer (CBasicII)
Osborne's A/P, A/R & G/L (CBasicII)
TDL Basic Games, Utilities, tutorials, I/O Drivers and Interactive Retrieval Info System (IRIS)
Loader/Relocator (TDL MAC 6.ASM)
TDLEBAS.AZM (allows TDL Extended Basic to run under CP/M)
CPM.ASC (simulates CP/M mode with TDLEBAS.AZM)
SSM PROM Burner Program (.ASM)
Pictures (ASCII files)
Dungeons & Dragons (MBasic)
Maillist (Ward Christensen's program in MBasic)
And lots more!

INFORMATION MASTER

reviewed by Bill Machrone

Information Master is a CP/M*—compatible information retrieval program oriented towards textual data. The program, available from Island Cybernetics, was originally written to perform retrieval from a large data base of articles and abstracts in the ecological sciences field. It, however, is a generalized program and is adaptable to a number of different retrieval needs.

Information Master operates under a variety of the CP/M-derivative operating systems, such as CDOS and IMDOS. It is "installable," in that some of the operating parameters can be changed for specific applications. The program is fast, since it maintains the dictionary in main memory during retrieval. The console displays are not sophisticated in that there are no cursor controls or even screen clears, but capabilities of that type are usually just window dressing, anyway. It does, however, format text going to the printer and gives you the option of sending it to a disk file for further editing or processing. Within its defined area of operation it does quite a lot, especially for the price, which is $37.50 per copy.

Within its defined area of operation it does quite a lot, especially for the price, which is $37.50 per copy.

One of the unique things about Information Master is that it does not have the input/data entry module that is usual for this type of program. Since its major function is to facilitate access to text, it is specifically intended for use with your current text editor. One advantage to this approach is that it isn't necessary to learn a new set of text editing rules in order to use Information Master. Whatever CP/M-compatible text editor you are familiar with is fine. There is also no reason why you can't use whatever high-level language you have at hand to create prompted input acceptable to Information Master. The files of information you create are considered "raw" text by Information Master; it processes them to build a dictionary of retrieval terms and a pointer file that provides access to the text.

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

Island Cybernetics provides a demonstration data base with the programs, and it is worthwhile to experiment with it before you plunge into creating your own. The data base has extracts from articles which are cross referenced by the topics upon which they are likely to be retrieved. A feature of the program is that only the dictionary and pointer files need be present on one disk. The data itself may be on a completely separate disk, thus maximizing data storage.
maximizing data storage.

The input requirements are simple. There are three "triggers" or delimiters that Information Master looks for in raw text in order to distinguish keywords from text. One of the dilimiters is used to establish a "brief" retrieval heading, such as the title of an article. Below is an example from the Information Master manual:

```
*C
INFORMATION MASTER, Users Manual, Island Cybernetics, 1979 *This short manual describes the
use of the "INFORMATION MASTER" program
for retrieval of text files using Boolean combinations of key words or phrases.
Vendor:
  Island Cybernetics
  P.O. Box 208
  Port Aransas, TX 78373
*K
INFORMATION RETRIEVAL/CP/M/DATA
MANAGEMENT/8080 CPU Z-80 CPU
*E
```

If the above entry (and any number of similarly organized entries) is presented to Information Master as raw text, it will be cataloged and cross-indexed by the key-

*CP/M is a registered trademark of Digital Research words that follow the *K delimiter. The *E signifies the end of the entry. If you use the "short form" of retrieval, the program will display the text from the *C to the first*. If you specify the long form, it will display all of the text down to the *K.

Another nice feature of Information Master is that the output can be directed to either the list device or the console. While we're on the topic of nice features, another that deserves mention is the "not in dictionary" function. If you request a lookup under "Z-80" the program will inform you that there is no corresponding entry in the master dictionary and will then list the close matches to the entry you had specified. This makes it is easy to pick out the entries you want. The "sounds like" algorithm may be a little generous in terms of giving you some matches that aren't even close to what you want, but it's better to have too many than too few.

Actual retrieval from the data base is done by specifying the keywords that you are looking for. Information Master provides a Boolean expression input capability, so that you can logically AND and OR your requirements. This feature alone sets it apart from the usual data retrieval applications written in Basic, which normally do not provide this function. Furthermore, most homegrown retrieval systems are limited in the number of keys that can be stored or retrieved upon. There is no limit to the number of keywords that can be associated with each piece of information, so that the cross-indexing capabilities are endless.

Information Master is unique in its "cataloging" capabilities of text and is adaptable to a variety of storage and retrieval needs.

Now that you know what Information Master does, the inevitable question arises, "What good is it?". Most of us don't have large data bases of articles and books to summarize, but we do have some commonplace data that could stand some organization, and there is the occasional unique application that can benefit from a program such as this. The manual contains some suggestions in addition to data bases of literature, including book collections, correspondence and recipes.

Taking recipes as an example, you can enter your favorite dishes and document where the recipes are located and what variants you have tried. Below is an example of how you might organize these entries:

```
*C
Chicken with walnuts in plum sauce
Bon Appetit, July, 1980 Page 8.
*Use 30% more sauce than recipe calls for. Breast
meat a good substitute for thighs. Goes well over
fried rice and with pina coladas. Simple but impres-
sively good.
*K
CHINESE/CHICKEN/GINGER/HOISAN/WAL-
NUTS/DINNER
*E
*C
Oven fried fish
Better Homes Cookbook, page 260.
*Season bread crumbs with parsley, bouquet
garni, parmesan cheese, dash garlic salt, tarragon,
basil,oregano,orwhatevercomestomind.8-10min-
utes sufficient for thin fillets.
*K
FISH/DINNER/FAST
*E
```

Information Master's short and long form output enables you to list just the recipe titles and the publication or list your comments as well. Any number of entries such as the ones above can be present in the raw text file. Information Master provides the dual advantage of randomaccesswith variable length records for the most efficient possible utilization of your disk storage. If you carefully standardize the usage of keywords you will have no trouble retrieving whatever you want from the data base. For example, you can specify "FAST and DINNER," "CHINESE and CHICKEN and DINNER," or something like "CHICKEN or FISH and DINNER."

Information Master can do things that would otherwise take extensive custom programming or cost far more for a generalized data base management subsystem. It compares very favorably to data managers like WHATSIT and Selector III, especially considering the price.

A totally different potential application is a personal diary or a businessman's calendar. In this mode, you could use the keywords to establish the date, the type of event and meaningful cross-indexes. The short form entry need not be used. Here's an example:

```
*C
10:00 Meeting with Joe Tyler. Discussed new appli-
cations program and suggested that Steve Linden
be appointed as user liaison. Tyler not sure about
Linden; will get back to me by 30 June.
*K
MEETINGS/10/JUN/1980/APPLICATIONS/
TYLER
*E
*C
1:30 Phone with C. Daniels of Hairy Software Inc.
Determined availability of King Kong word
processing system. Version 1.0 will be replaced
in 45—60 days. Field upgrade to existing
licensees is for cost of media and manuals.
*K
10/JUN/1980/KING KONG/WORD PROCES-
SING
*E
```

In this kind of example, Information Master can manage past or future appointments and to-be-done items. With a little ingenuity, follow-up dates could be coded as part of the keyword area, so that an inquiry can tell you almost instantly what needs to be done by a certain date. Anyone can appreciate the permanence of the records and the ability to review a month's meetings or all those held on a given topic or with a specific individual.

Information Master cont'd...

In conclusion, Information Master is unique in its "cataloging" capabilities of text and is adaptable to a variety of storage and retrieval needs. If you don't need to do a lot of field-oriented further processing with the retrieved data and if simple list or console output is sufficient, Information Master can do things that would other wise take extensive custom programming or cost far more for a generalized data base management subsystem. I think that it compares very favorably to data managers like WHATSIT and Selector III, especially considering the price. This is not to say that it would replace either of them; WHATSIT is uniquely capable in expressing hierarchical relationships among data items, while SELECTOR has a full range of report generation capabilities that are quite powerful in themselves. I feel, however, that neither of them could beat Information Master at its own game. It doesn't resort to cute "artificial intelligence" conversations with the user and, depending on how you set up your keywords, can represent hierarchical or relational data structures. It would be nice to see some substring operators so that it wouldn't be necessary to break up the year, month and day, and so you could pick out subcodings like "CPU" from both "Z-80 CPU" and "8080 CPU." A negation operator would be neat, too. Then you could say, in essence, "DINNER but not FISH."

But all this is quibbling. Information Master is a good buy, has no apparent bugs, is reasonably well documented and is both easy and fun to use. It is available from: Island Cybernetics, P.O. Box 208, Port Aransas, Texas 78373, tel: (512) 749-6673. The cost is $37.50. □

The Talos DIGI-KIT-IZER Digitizing Tablet Kit

by Jon Bondy

I have always been fascinated with graphics applications for computers. I therefore purchased an analog to digital converter board just to be able to interface a joystick to my computer as a graphics input device. I expected to be able to designate locations on my graphics CRT screen with the joystick, and hoped that I would be able to draw some simple pictures with it, too. The joystick proved to be far less useful in these capcities than I had imagined, for two reasons.

If one hooks the joystick up so that the position of the stick is directly related to the position of a cursor on the graphics screen, it is difficult to keep one's hand still enough to specify a single point on the screen with any accuracy. If one modifies the joystick so that it represents the direction of motion of the cursor on the screen (i.e., pushing the stick up moves the cursor up), then any point can be designated on the screen with ease, but it is sometimes a slow process to move the cursor to the correct position.

In the midst of this frustration, a friend of mine at Apple computer commented to me that they had selected a digitizing tablet product because it had solved these problems and he felt that it was the best way to input graphic data into a computer. I picked up a manual for their tablet, and it and their software did seem to offer a lot. Unfortunately, I am not an Apple owner, and even if I were, the price (in excess of $700) was more than I felt that I wanted to spend on that portion of my system hardware. I wanted to re-write the software anyway, to learn about how it fits together. I looked at the Summagraphics Bitpad and the Talos Digitizers, but they still were in the $700 and up price range. I then discovered that Talos made a digitizing tablet kit listing for $500 with an active area of 11 by 11 inches, a resolution of 200 coordinates per inch and a data rate of 100 coordinate pairs per second. I decided that this was worth a try, and ordered it from Cushman Associates, Inc. (2735 Skylark Road, Wilmington, Deleware, 19808) who supplied it for $450. You may be able to find it locally for a similar price.

The kit comes with a parallel interface and no power supply, although a serial interface board and a power supply can be purchased as options at about $100 each. I purchased the kit with no options, figuring that I could obtain power from my computer (+15 volts at 250 ma— more on this later) and a parallel interface from my Cromemco TUART board. One thing which was not clear until I received the kit was that the parallel interface really requires 16 bits, taking up two 8-bit parallel input ports rather than the one which I had assumed.

My kit arrived in a box containing two large plastic bags, on for the main kit and one for the parallel output board. This allows Talos to easily mix various output options with the main kit. The main kit consists of two PC boards and some hardware with which to make the digitizing tablet box. Each of the sub-kits (main kit and output board) contains its own instruction manual.

The kit is intended for those who have built kits before, since the manuals do not go into the kind of detail which a HeathKit manual might, but the construction instructions are carefully plotted step by step, and are for the most part easy to follow. There was one typo in their output board manual (on page 3, a 10 ohm resistor was listed as Brown-Black-Brown, whereas it should have been Brown-Black-Black), but the manuals were surprisingly free of printing mistakes. On page 3 of the main manual, the builder is instructed to determine if the pen is a 'style A' or 'style B' pen, but no indication is made as to how to determine which pen style came with the kit. It turns out that there is a small paper tag on the pen cord indicating this. On page 12 or the main manual, the builder is instructed to place the ICs for the MUX board into their sockets, but no list of IC's is provided. It proved easy for this board, since IC's 9 and 10 are 74C42's, and the rest are CD4051's. The section on the final mechanical construction of the tablet (where the PC boards are united with some aluminum shields using nuts, bolts and spacers) was not completely clear, but I figured it out after playing with the pieces for about 30 minutes. The section describing the theory on how the tablet works is rather skimpy and the schematics are not annotated with signal names, so understanding the circuitry is not as easy as one might with. All parts required for construction of the kit were present, down to the solder and wire needed for jumpers. One nice thing about this kit is that the parts are usually called out in the order in which they appear on the board, rather than by part number, so that you usually can locate parts rapidly and accurately. The plastic housing for the completed tablet was adequate for a hobby application, but probably would not stand up to a commercial environment.

Jon Bondy, Box 148, Ardmore, PA 19003

In general, the instructions were good, but I assembled the boards in a slightly different manner than the manuals called for. I always install my sockets first, place a piece of cardboard over them, ivert the cardboard and PC board and then solder all sockets in at once. In order to make sure that they are all in flush to the board, I tack solder them on two pins which are diagonally opposed (say 1 and 9 on a 16 pin socket) and then press them flush to the board while re-heating the two pins from the underside of the board. I find this to be easy and time-saving, as compared with placing a single socket in the PC board, turning the board over, crimping two leads, turning the board over again, etc. Since my technique will not work if any other components are on the board, I had to do this out of the order specified in the Talos manuals. If you do as I did, be careful, two IC positions (U17 and U18) do not use sockets on the parallel output board.

Assembly took 4.5 hours, but I have been building kits for over ten years, so that probably is the shortest time which you could expect. Most of the time was spent stuffing components into the PC boards, but some of it was spent making over 40 jumper wires (archaic, but it really didn't take all that long) and performing the final mechanical assembly. I found that one way to make the

—CONTINUED ON NEXT PAGE—

```
program tablet;

  const
    port1 = 132; { msb's and up/down x/y -- 84 hex }
        { bit 0 -- printer on line
          bit 1 -- printer busy
          bit 2 -- tablet data bit 8
          bit 3 -- tablet data bit 9
          bit 4 -- tablet data bit 10
          bit 5 -- tablet data bit 11
          bit 6 -- tablet stylus up/down
          bit 7 -- tablet x/y data flag }
    port2 = 148; { lsb's -- 94 hex }
        { bit 0 -- tablet data bit 0
          bit 1 -- tablet data bit 1
          bit 2 -- tablet data bit 2
          bit 3 -- tablet data bit 3
          bit 4 -- tablet data bit 4
          bit 5 -- tablet data bit 5
          bit 6 -- tablet data bit 6
          bit 7 -- tablet data bit 7 }
    mask = 128; { select x/y bit }
    msbs = 60; { select msbs (8-11) out of port1 }
    msbshift = 64; { to shift msb's out of byte }
  var
    x, y, i : integer;

  function portread(port : integer) : integer; external;
  function pand(val1, val2 : integer) : integer; external;

  procedure readxy(var x, y : integer);
    begin
    { await y data }
    while (pand(portread(port1),mask) <> 0) do begin end;
    y := portread(port2) + (pand(portread(port1),msbs) * msbshift);
    { await x data }
    while (pand(portread(port1),mask) = 0) do begin end;
    x := portread(port2) + (pand(portread(port1),msbs) * msbshift);
    end; { readxy }

  begin
  for i := 1 to 1000 do begin
    readxy(x,y);
    writeln(x:8,y:8);
    end;
  end.
```

DIGI-KIT-IZER cont'd...

final 34 jumpers rapidly and easily was to wrap the jumper wire around a rod of approximately 1/2 inch diameter and then cut semicircles of wire off the resulting spiral. This enabled me to have uniform jumpers which did not rise above the level of the IC's, ensuring that no short circuits developed.

I am used to resistors with long leads which must be carefully formed in order to place them into the PC board. Talos supplied (for the most part) pre-formed resistors, so that they simply dropped into the holes in the PC board. This was a mixed blessing for me, since the leads were too short to bend easily, so the resistors teneded to drop out of the board when it was turned over for soldering.

When it was complete, I ran the few tests which the manual recommended, using a multimeter, and then sat there staring at it. The tests said that it was O.K., but was it working, or not? I was using one of my TUART's parallel ports to sense the status lines of my printer, but I managed to squeeze enough bits out of the status lines to get the tablet and printer to share two 8-bit input ports. I wired the paralled interface up to my two TUART input ports, and wrote the following program in UCSD Pascal in order to see if the tablet was functioning as I expected. Surprise! Not only did the tablet work, but the program worked the first time I tried it!

A few notes on the tablet which were not mentioned in the manuals. The tablet works by emitting a magnetic field (which changes at a high frequency) from the tip of the pen, and detecting it with a grid of sense lines just under the surface of the tablet's plastic cover. The magnetic field emitted by the pen is strong enough to destroy floppy disks, so be careful when the unit is running. It may also leave enough residual magnetism to cause problems when powered down, although I have not checked this! The power required to drive the magnetic fields in the pen is great enough that the tip of the pen will heat up noticeably, but this doesn't indicate a problem with the kit. If you become worried about the heating, put a scope on the two drive lines to the pen (I believe that these are jumpers R and T from looking at the output board schematic, but the main kit schematic would have you believe that they are jumpers Q and T). The signals should be undistorted sine waves; distortion indicates a parts mismatch in the oscillator, and could cause excess heating. The driver IC's for the pen are protected against thermal overload with automatic shutdown circuits, so even if the drivers were to overheat, they should recover after they cool off. Although the manuals indicate that a voltage of +15 is required to run the tablet, the factory told me that it would run on voltages as high as +17, and I am running it at +16.5 (off of my S-100 bus unregulated power supply). Final tablet assembly involves attaching one of the PC boards to the plastic case with adhesive backed paper. I wanted to avoid that step in order to be able to service the unit if necessary, and it turns out that my kit will press fit' into the case without any adhesive. □

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.

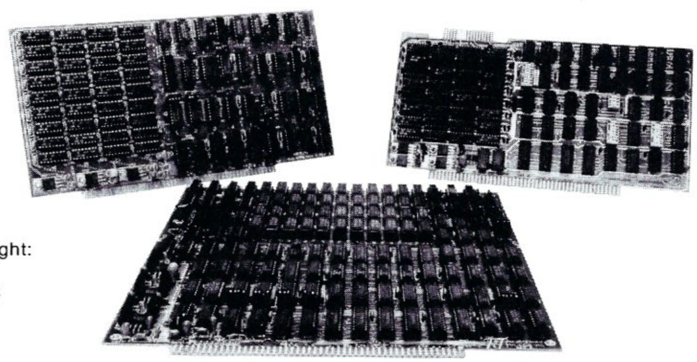
SUPERAM™ 2—We've delivered over 5,000 of these 64K dynamic RAM boards to OEMs worldwide. This board sets the standard for reliable operation.

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

The CP/M Connection

Part III—Implementing the IOBYTE Function

by Chris Terry

The CP/M System Alteration Manual (page 15) notes that "...the user can optionally implement the IOBYTE function which allows reassignment of physical and logical devices." Unfortunately, the clues to the procedure are scattered through the Facilities Manual, the System Alteration Manual and the Interfacing Manual, and no examples are qiven.

Why, in practice, would we want to change the active peripherals? We might, for example, have both a dot-matrix printer (on a parallel port) and a daisy-wheel printer (on a serial port); the IOBYTE function allows us to use the dot-matrix printer for numeric output, but to switch to the daisy-wheel for correspondence. Again, if we normally use an electronic keyboard and VDM as the console, but also have a keyboard/printer serial terminal such as a Teletype or Diablo or TI Silent 700, we can switch all console functions to the serial terminal whenever we wish, and switch them back when desirable.

Logical Devices

The ability to perform this switching implies that we have a logical I/O system in which each kind of I/O operation is performed by a separate logical device - that is, a software routine which controls the flow of data, and may do some formatting and CRC generation or checking, but does not directly talk to a physical I/O device.

Communication between the logical device and a physical device takes place through two intermediaries: a logical driver, which is permanently associated with the logical device, and a physical driver that is permanently associated with a particular physical device (see Figure 1). In the distribution version of CP/M, the logical and physical drivers are one and the same; that is, each logical device is permanently linked to one, and only one physical device.

Chris Terry, 324 East 35th St. New York, NY 10016

However, when the IOBYTE function is implemented, the logical and physical drivers are separated. The logical driver then consists of a switching mechanism that allows its associated logical device to be linked to any one of four physical drivers (and their associated physical devices). The IOBYTE itself is part of this switching mechanism.

CP/M contains four logical devices. For convenience, they are named:

1) CON: 2) RDR: 3) PUN: 4) LST:
(The colons (:) are part of the names.)

The CON: device provides slow-speed communication between the operator and the operating system. It has three logical drivers: CONST, which checks the character ready/not ready status of the currently assigned console input device; CONIN, which fetches a single character from the console input device, and CONOUT, which outputs one character to the currently assigned console display device.

The RDR: device is for input only, from mass storage devices such as a paper tape reader, a cassette playback, a card reader, a badge reader, etc. It has one logical driver, called READER.

The PUN: logical device is for output only to paper tape, cassette recorder, etc. It complements the RDR: device. It has one logical driver, called PUNCH.

The LST: device is for output only. It is not used by the facilities built into the CCP, though it can be linked in tandem with the console display (ctrl-P toggles this link on and off). It is meant for directing the output of application programs to a printer or to mass storage devices other than the disk subsystem. It has one logical driver, called LIST.

Logical Drivers and the IOBYTE

In the distributed system, which does NOT have the IOBYTE function implemented, the logical drivers actually contain the physical drivers. This means that each logical device is linked to one, and only one, peripheral.

When the IOBYTE function is implemented, this situation changes. The physical device drivers become separate routines (TTYIN, TTYOUT, etc.). The logical drivers then become selection routines, each of which may select one out of four possible physical drivers according to the code found in the corresponding section of the IOBYTE.

The IOBYTE is located at 0003H, and is divided into four 2-bit sections (see figure 2), each of which is associated with one of the logical devices. The 2-bit code (00, 01, 10, or 11) found in any given section of the IOBYTE selects one of the four physical drivers that can legally be associated with that particular logical device. Figure 2 also shows the names associated with the codes for each logical device. It is important to note that from the viewpoint of the switch mechanism in the logical driver, only the codes themselves matter. The names are merely identifiers of the legal codes in each section of the IOBYTE, and only become useful when the STAT utility is used to change the contents of the IOBYTE - that is, to assign a new peripheral to a logical device.

There are many possible ways of implementing the selection mechanism. A neat and straightforward solution can be found in a program called VBIOS31, by Jeff Kravitz, which is contained in Volume 1 of the CP/M Users' Group library. Each logical driver has the form shown in figure 3, except that after the LDA IOBYT instruction, the LIST driver has two RLC instructions, the PUNCH driver has four RLC instructions and the READER driver has two RRC instructions. The effect of these is to shift the bits of interest into bit positions 0 and 1 of the A register.

	IOBYTE AT 0003H			
Bit Position-->	7 6	5 4	3 2	1 0
Logical Dev.-->	LST:	PUN:	RDR:	CON:
BINARY DEC	Device names known to PIP & STAT			
00 0	TTY:	TTY:	TTY:	TTY
01 1	CRT:	PTP:	PTR:	CRT:
10 2	LPT:	UP1:	UR1:	BAT:
11 3	UL1:	UP2:	UR2:	UC1:

Figure 2. Device Selection Codes in IOBYTE

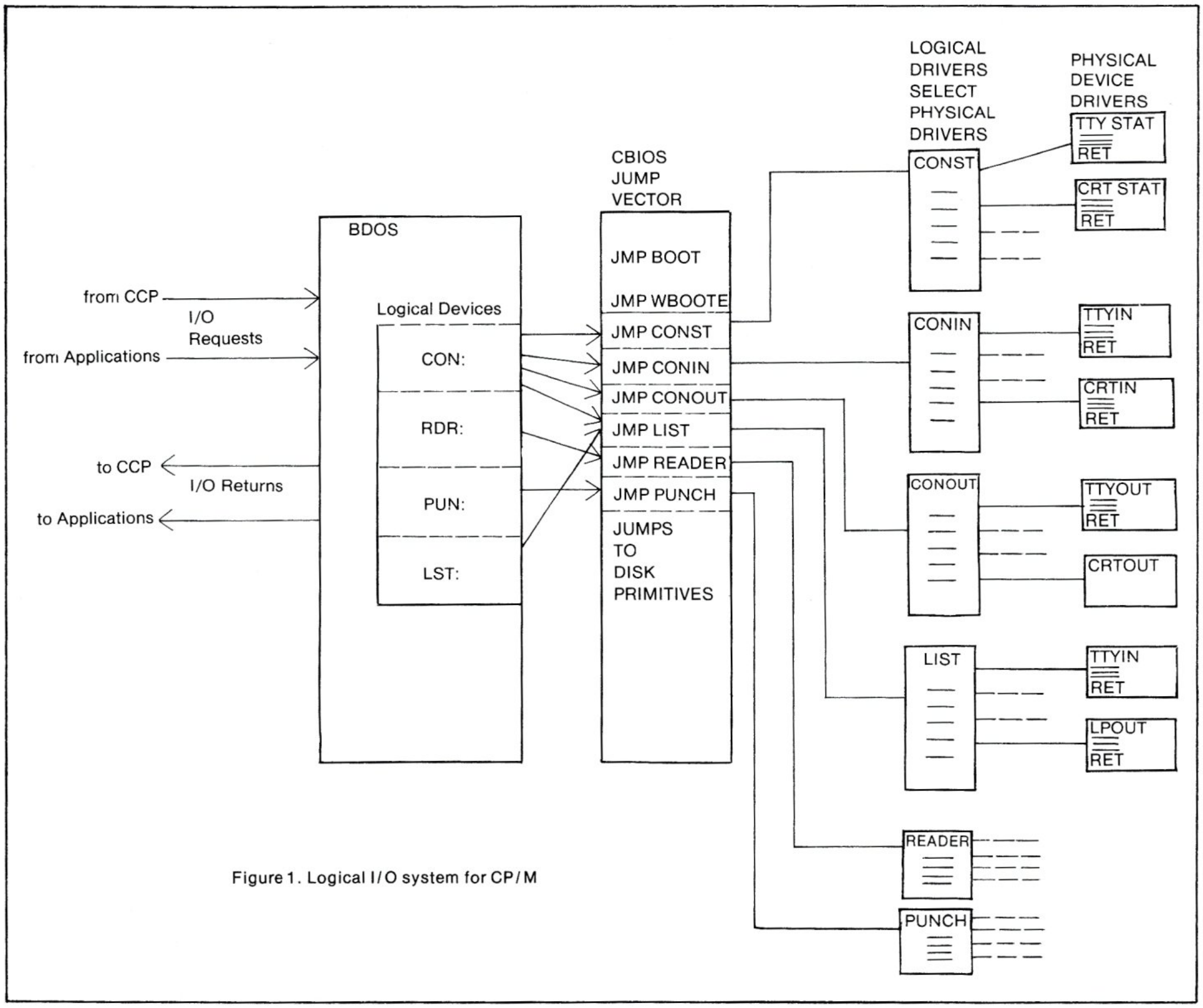


Figure 1. Logical I/O system for CP/M

CP/M Connection cont'd...

The CALL to the common I/O Dispatcher (IOCAL) puts the address of the first entry in the table of physical drivers on the stack as the Return address, although it will not be used as such. IOCAL's job is to find which table entry to use, and then to branch to the address contained in the entry. To do this, it uses the IOBYTE code as an offset to be added to the address of the first table entry. The original IOBYTE code ranges from 0 through 3; however, each table entry is two bytes long, and therefore our offset must be doubled so that its possible values are 0, 2, 4, or 6. This is done by the single RLC at the start of IOCAL. Now we set bits 0 and 3 through 7 of the A register to zero with the ANI 6 instruction, which leaves the absolute value of our doubled code in the register to be used as the offset.

The XTHL instruction saves the current contents of the HL register pair on the stack and brings what was on the top of the stack (the address of the first table entry) into HL. To this (after saving the contents of DE) we double-add our offset by clearing D, moving the offset from A into E, and doing a DAD D. The HL register pair now points to the table entry containing the address of the desired physical driver. The next five instructions bring the driver address itself into HL and restore the original contents of DE. The XTHL again swap HL and the top of the stack, so that the physical driver address goes on the stack and the original contents of HL are restored. Finally, the RETurn instruction pops the driver address off the stack into the Program Counter, and we start executing the selected driver. The RETurn instruction at the end of the driver itself passes control back to whichever routine requested the I/O operation.

Thus, every I/O call, whether to BDOS or directly to any one of the logical drivers, causes the IOBYTE to be inspected and control to be passed to the physical driver specified in the appropriate section of the IOBYTE.

Other Considerations

The LST:, PUN:, and RDR: are one-way logical devices, and assigning a new physical device to one of them does not affect either of the other two. The only restriction is the obvious one that it is useless trying to obtain input from an output-only device, and vice versa. Care must be taken, however, in the assignments to CON:, on a two-way logical device. Every assignment to this device changes all three of the associated physical drivers simultaneously; that is, the status driver, the character input driver and the character output driver. The branch tables for these drivers must be set up so that a mistaken reassignment

```
CONIN:  LDA     IOBYTE  ;Gets the complete IOBYTE
        (RLCs or RRCs   ;as needed to shift code into
                ;bits 0-1 of A. None needed for CON:)
        CALL    IOCAL   ;Puts the address of CITBL on stack
CITBL:  DW      TTYIN
        DW      CRTIN
        DW      RDRIN
        DW      UC1IN

IOCAL:  RLC             ;Double the code bits of interest
        ANI     6       ;Mask out all other codes
        XTHL            ;Save HL, get address of XXTBL
        PUSH    D
        MOV     E,A     ;Put doubled code in E
        MVI     D,0     ;and clear D
        DAD     D       ;Add doubled code to XXTBL address
                        ;to find address of required entry
        MOV     A,M     ;Get low byte of entry
        INX     H       ;Now point to high byte of entry
        MOV     H,M     ;and put it into H
        MOV     L,A     ;Put the low byte into L
        POP     D       ;Restore DE
        XTHL            ;Put entry address on stack, restore HL
        RET             ;Pop entry address into PC to start
                        ;executing the required driver (TTYIN)

TTYIN:  ---
        ---
        ---
        RET             ;Returns control to original caller
```

Figure 3. Typical Code for one Logical Driver (CONIN), an associated Physical Driver (TTYIN), and the common I/O Dispatcher (IOCAL).

UCSD Pascal*

Until now, getting U.C.S.D. Pascal running on your system has been at best a chancey proposition.

There have been several versions of the system available for CP/M systems and one which requires extensive assembly language programming to bootstrap.

The U.C.S.D. Pascal system II.0 is now available ready to run for systems using the Thinker Toys* (Morrow Designs) Discus II system. This outstanding 8" disk system offers 587,008 bytes of storage per drive as configured for U.C.S.D. Pascal and is exceptionally easy to install and use as well as offering a good deal of reliability.

The UCSD Pascal system now offered by Northwest Comm. includes the total Pascal system as released by the University of California as well as additional utility programs developed by Northwest Communications such as an interface to a Thinker Toys* M-26 hard disk(26 megabytes capacity) and a number of Pascal utility programs to assist you in the development of good software.

This software package is particularly easy to run on systems using Thinker Toys* Discus II controller boards because of Morrow Design's thoughtful inclusion of a serial I/O port on the disk controller board.
When you first receive your UCSD Pascal package from Northwest Communications, you may immediately boot the system into a program which will lead you through the steps necessary to configure the software for your system.
The program asks, in plain language, what the memory configuration is, and asks you which of several included I/O modules you wish to use.

The I/O modules supplied include one for only the on board serial port, one which uses a Thinker Toys* Switchboard as ports for a printer and remote unit and one which is similiar to the I/O supplied with a NorthStar Horizon computer. Several other useful I/O modules are included as well as one which you may use to tailor the software to your own unique hardware if necessary. This module is blocked out for you in advance with all disk IO complete and requires that you just insert your own I/O routines and assemble using the excellent editor assembler included with the Pascal package. Use of the Disk Jockey serial port for this tailoring makes it very easy to implement. Remember, if you have a North Star Horizon, or wish to use only the on board serial port, or use a Thinker Toys Switchboard you have no programming to do, just answer yes to the IO question section of the configuration program and you're ready to go. Once the I/O and memory configuration has been set, the Pascal system will use the new information in all future bootstraps. You may alter the IO any time to reflect changes in your system.

Future releases planned are interfaces to North Star and Tarbell disk control boards as well as modules for IO involving many of the popular Video control boards and other useful hardware such as Mountain Hardware's Modem board. Also, we will introduce UCSD Pascal for the Heath/ Zenith H - 89 tabletop computer in January, 1981.

If you want to be informed when these programs are available please send us a letter or postcard with your name and address and a description of your system. We will put your name on our mailing list and keep you posted on all future releases.

These enhancements will be available at a low cost to registered owners of our UCSD Pascal package.

TECHNICAL INFORMATION

Our UCSD Pascal system will run on any computer with a minimum of 48k of RAM located anywhere between 0000 hex and DFFE hex. The computer must have at least 1 DiscusII controller and 1 or more 8" drives. The software will support up to 4 Discus II 8" drives or 2 8" drives and 1 M-26 hard disk. Any IO configuration may be used. The system initially bootstraps using the Discus II serial port.

DOCUMENTATION

System documentation is available in two forms. The standard system includes Printed Documentation on the Northwest Communications UCSD Pascal system, as well as the complete UCSD Pascal system documentation on a Pascal format disk. You may access this material either by scanning through it using the Pascal system editor, or by printing it out on a line printer. If you prefer, we will send you the complete printed documentation package in an indexed 3 ring binder with room to store several disks in looseleaf holders.
As an introductory offer, on all orders for the UCSD Pascal system placed before Feb 1st, 1980, we will include at no extra charge our 'PRINTER.CODE' program. This program converts your Pascal system into a very powerful and easy to use Word processor. The program operates by the detection of symbols which you place into the text file. It operates on the text to format it to your specifications.
The PRINTER.CODE program normally sells for 99.95 but is free with the purchase of a Pascal system. Why not get on board the Pascal Bandwagon?

U C S D   P a s c a l

UCSD Pascal System for Discus II	349.95
System for Discus II and M-26 Hard disk	379.95
Standard Documentation pack	n/c with system
Complete printed documentation with binder and dividers if ordered with system	39.95
Printed documentation without system	69.95
PRINTER.CODE text formatting/word processing	99.95
Beginners guide to the UCSD Pascal system by Dr. Kenneth Bowles	11.95

Northwest Communications
2 airport plaza box 2454
Idaho Falls, ID 83401

TOLL FREE ORDER LINE
800-635-4706
MC/VISA

*Thinker Toys/Discus are trade marks of Morrow Designs. CP/M is a trade mark of Digital Research
*UCSD Pascal is a registered trade mark of the Regents of the University of California

CP/M Connection cont'd...

command does not cause loss of all communication between the operator and the operating system.

Suppose we have a keyboard and VDM as our standard console, a serial CRT terminal as the alternative device and do not intend to use the BAT: (input from RDR:, output to LST:) or UC1: (user-defined) logical devices. Then in our CONST and CONIN logical drivers, the first table entry will branch to TTYST and TTYIN drivers, and in CONOUT the first table entry will branch to the VDM driver software. The second table entry in CONST and CONIN will branch to the CRTST and CRTIN routines, and the second entry in the CONOUT table will branch to CRTOUT. For the third (BAT:) and fourth (UC1:) table entries, we have two possibilities:

a. In each table, make the 3rd and 4th entries the same as the first. This will automatically default them to the standard device.
b. Put branches to an error handling routine. This might merely be a null input routine that returns a NULL (00) and a null output routine that copies C into A and then returns; or the error handling routine might include an error message.

Initialization. As we have seen, communication between the operator and the computer is now totally dependent upon having the correct code in bits 0 and 1 of the IOBYTE. At power-on time, this byte contains a random bit pattern. It is therefore essential that the CP/M Coldstart portion of the boot procedure be modified to include proper initialization of the IOBYTE. If the assignments are set up so that the first entry in each table (code 00) sets up our normal system configuration, the initializing code merely clears the A register (XRA A) and deposits this 00H in IOBYTE (STA IOBYT).

Device Assignment from the Console

The STAT utility has the ability to list the legal device assignments for each of the four logical devices, to list the current assignments, and to change the current assignments. STAT does not know (or care) how the logical driver tables are set up in the CBIOS; it is concerned only with examining the contents of the IOBYTE at location 0003H, reporting what it finds there, and changing specific bits in the IOBYTE while leaving the remainder untouched.

To obtain the list of legal assignments, we type the command:

```
A>STAT VAL:
```

which generates the response:

```
CON:= TTY: CRT: BAT: UC1
RDR: = TTY: PTR: UR1: UR2
PUN: = TTY: PTP: UP1: UP2:
LST: = TTY: CRT: LPT: UL1:
```

If we wish to know the current device assignments, we type the command:

```
A>STAT DEV:
```

If the IOBYTE contains the bit pattern 10 11 01 00, the response will be:

```
CON: IS TTY:
RDR: IS PTR:
PUN: IS UP2:
LST: IS LPT:
```

Now, if we wish to change the reader assignment from PTR: (which could be a fast paper-tape reader) to UR1: (which could be a cassette), we type the command:

```
A>STAT RDR:=UR1:
```

STAT would then change the code in bits 2 and 3 of the IOBYTE from 01 to 10. All subsequent requests for Reader input would then access the cassette instead of the paper-tape reader.

If we want to change more than one assignment, we can put up to four such commands on the same line, separating them with commas, e.g.:

```
A>STAT RDR:=UR2:,LST:=TTY:
```

STAT will detect and deny any request to assign an input physical driver to an output logical device, or to assign device names that are unknown to it, with the error message:

```
INVALID ASSIGNMENT
```

Space on the System Tracks

There is one last item which we must take into account: the space on the system tracks (0 and 1) that is available to expand the CBIOS. The standard CBIOS begins at 3E00 in a 16K system or 7E00 in a 32K system. Thus, we have 512 bytes available for all CBIOS functions (including the disk primitives). If our expanded CBIOS (with the new IOBYTE function) requires more than 512 bytes, then we shall have to move CP/M downward by 1K in order to fit the new CBIOS between the top of the BDOS and the top of available memory. This creates a space of 512+1024=1536 bytes above the top of the BDOS. We cannot use all of this space, however, since only 9 sectors (1152 bytes) are available for the CBIOS on System Track 1. We must therefore ensure that the last byte of object code in our expanded CBIOS has a memory address no greater than XE7F (where CBIOS starts at XA00). There is nothing to prevent us from using the space XE80 thru XFFF as scratchpad memory.

If memory space is tight, and we only require (say) 640 bytes for the new CBIOS, we could move CPM downward by only one page (256 bytes). However, a shift of less than 1K will make computation of the ORG address of CBIOS and of the offset less convenient.

Introduction To CP/M

Part V of the "Introduction To CP/M," by Jake Epstein, will be continued in the next issue of S-100 MICROSYSTEMS. Regretfully, the manuscript arrived too late for inclusion in this issue.

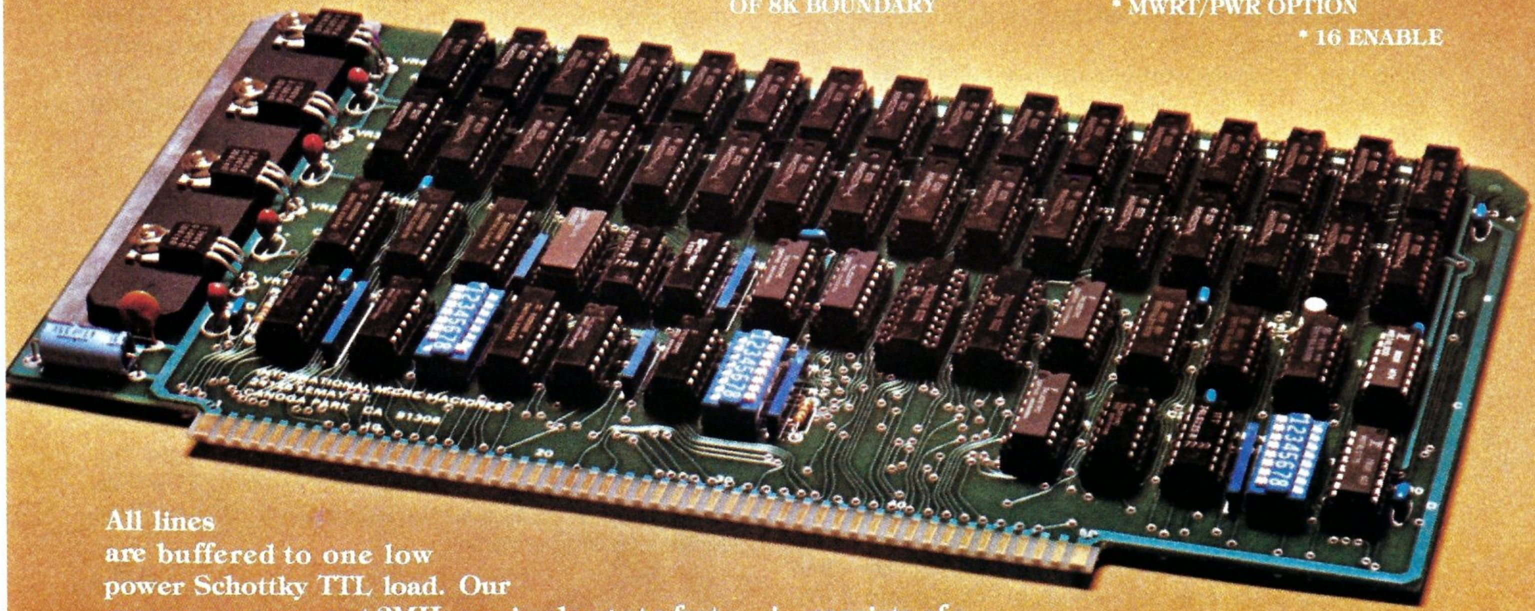
a WORD to the wise...

IMM technology has developed a new innovative memory board with your future in mind. This compact static memory will operate in your present 8 bit S-100 system with built-in flexibility to handle 16 bit transactions when you are ready for it.

Sound engineering and common sense were the basic ingredients in our design parameters for the IMM-816. Your needs were our guidelines. The end result is a reliable memory with remarkable features at competitive prices.

- FULL IEEE COMPLIANCE
- LOW POWER 5257 STATIC MEMORIES
- 8 BIT EXTENDED ADDRESS OPTION
- ADDRESSABLE IN 2 SEPARATE BLOCKS OF 8K BOUNDARY
- OPERATES IN 8 or 16 BIT S-100 SYSTEMS
- IEEE OR NON IEEE SYSTEMS
- PHANTOM LINE ENABLE
- WAIT STATE ENABLE OPTIONS (0-2)
- MWRT/PWR OPTION
- 16 ENABLE

MODEL IMM-816



All lines are buffered to one low power Schottky TTL load. Our memory can coast at 2MHz or zip along at a fast 4 MHz clip. All functions are DIP switch selectible without wire jumpers. We used high quality parts such as the Augat face-grip, low profile sockets to assure maximum component retention.

If your system employs Bank Select operation, our Memory Manager Card (Model IMM-900) may be used to emulate virtually any bank select format currently in existence; ie. Cromemco, Alpha Micro, Horizon, etc. The memory management card consists of a simple I/0 port, switch selectable to any 8 bit address. Data is written to 8 IEEE extended address lines (A16-A23) to set the desired active bank.

IMM 816K-45	2MHz KIT	$ 290
IMM 816K-25	4MHz KIT	$ 315
IMM 816A-45	2MHz Assembled & Tested	$ 325
IMM 816A-25	4MHz Assembled & Tested	$ 350
IMM 900K	KIT	$ 65
IMM 900A	Assembled & Tested	$ 95

Charge Cards are accepted

IMM

INTERNATIONAL MICRO MACHINES

24125 Lamay Street

Canoga Park, California 91306 ● (213) 887-5737

Modifications to CBasic2

by Ben and Andy Galewsky

CBasic by Software Systems is a good language for many applications, especially in the business environment. The CBasic language comes as a package of two programs. The Basic source is entered into a file using a text editor, then compiled into intermediate code by the program CBAS2. The intermediate code is executed by invoking CRUN2.

Unfortunately, the language has one major shortcoming. There is no provision for outputing a single character to the console at the current cursor position; a buffer must be filled and then printed. This creates problems for users with memory mapped video displays. Formated screens and other special programs also become difficult (i.e. Osborne and Associates' Payroll with Cost Accounting).

It is possible to write a machine language subroutine to output a single character and have CBasic load the program every time it is run. This has its own attendant problems. The solution presented in this article is a modification to CRUN2. A machine language subroutine is inserted into an unused portion of CRUN2. The character to be placed on the screen is POKEd into a memory location specified by the subroutine. The subroutine is then CALLed from Basic. Then the subroutine makes a call to CP/M to display the character at the cursor position. This eliminates the need to load the routine from disk every time the program is run because it travels along with CRUN2.

The second modification involves the CRUN2 sign-on message, allowing a more elegant and custom finish, as well as making computer operation easier for an inexperienced user.

Making The Modification

Before attempting to modify any program, especially expensive or irreplaceable software, a copy should be made and kept in a safe place free from magnetic radiation and high temperature.

Galewsky, 7705 Calder Ave, Beaumont, TX 77706

With the backup made, it is now possible to begin the modifications. For this you will need to use DDT (Dynamic Debugging Tool) supplied with your CP/M system, or a similar program. First invoke DDT by typing DDT CRUN2.COM. DDT will return with the following prompt:

```
DDT VERS 1.4
NEXT PC
4300 0100
```

The 4300 under the NEXT shows the next available address after CRUN2. The 100 under PC tells the location of the program counter.

Starting around 110 hex is an embedded copyright notice. This area can be displayed by typing D100 (figure 1). It is here that the new machine language subroutine will be placed. To load the program into memory, the in-memory assembly function of DDT will be used. Type A120, to start the assembly at 120 hex. Type in the following program:

```
120 MVU C,02
122 LDA 130
125 MOV E,A
129 CALL 0005
129 RET
```

Key <RETURN> to end the in-memory assembly function. When called, this program loads the CP/M code for print character (2) in to the C register of the microprocessor. Then it fetches the character out of memory location 130 hex and moves it to the E register to be passed to CP/M. Finally CP/M is called at 0005 hex to place the character on the screen. This solves the single character output problem.

The next modification is to the sign-on message. This message is found at 2147 hex (on CRUN vs 2.05). Display this message by typing D2100 (figure 2). The new message may be up to 18 characters long including a terminal dollar sign (the extra dollar signs may be written over). In our cas we decided to have CRUN2 clear the screen and print

Please wait...

Which is a little less confusing and more reassuring to the inexperienced operator then the usual

CRUN VER 2.05

message. The revised message is shown in Figure 3. To put the proper characters into memory, use the DDT S command. This displays the memory contents and allows you to change it. Type S2147 and key in the proper ASCII codes (see figure 4). The 04 code at the begining is the screen clear code for the Vector Mindless Terminal. Use whatever screen clear character you particular terminal uses. The remaining codes are for the Please wait... message. End the message with a "$" (ASCII code 24). The dollar sign is the terminator of a message string used by CP/M. Type <Q> to Quit the change mode. Display the message again with the command D2147 to check for proper coding (see figure 4). This ends the modification.

The modified CRUN2 must be saved on the disk. To do this type control C. This does a warm start and returns to the A> prompt. Type SAVE 72 RUN.COM. This saves 72 256 byte pages into the file RUN.COM We use the name RUN.COM to make programs easier to run. The operator only has to type RUN > filename.

Testing The Modifications

As with any program, all chages must be thoroughly tested. Testing the sign-on message is easily done; simply run any CBasic program and your sign-on message should be displayed in lieu of CRUN VS 2.OX

To test the single character printing, a short program will have to be written. The program in listing 1 is an example. This program uses the CONCHAR% function of CBasic. It will input a line of characters and then allow **the editing of this line. The functions supported are:**

<space bar>	advances to the next character
D	deletes the current character
C	changes the current character
<return>	inputs a new line to edit

This program is quite useful as an editor of input data in a program. Function PRT uses the single character print routine to display the argument DISP$.

In Conclusion

These modifications overcome some of CBasic's limitations. Combining these changes with the turnkey CP/M system described in December, 1979 *Creative Computing* will aid in the operation of your application programs. □

—PROGRAM ON NEXT PAGE—

```
-D100
0100 C3 A0 26 C3 00 00 00 00 0E 0D C3 05 00 C3 00 00 ..&.............
0110 50 00 43 4F 50 59 52 49 47 48 54 20 28 43 29 20 P.COPYRIGHT (C)
0120 31 39 37 37 2C 20 31 39 37 38 2C 20 31 39 37 39 1977, 1978, 1979
0130 20 43 4F 4D 50 49 4C 45 52 20 53 59 53 54 45 4D  COMPILER SYSTEM
0140 53 20 49 4E 43 05 00 00 00 00 00 00 00 00 00 00 S INC...........
0150 00 00 00 00 00 00 00 00 00 00 E1 E3 78 B1 C8 1A .............x...
0160 77 0B 13 23 C3 5C 01 0A 12 03 13 0A 12 03 13 0A w..#.\..........
0170 12 03 13 0A 12 03 13 0A 12 03 13 0A 12 03 13 0A ................
0180 12 03 13 0A 12 C9 21 8B 01 71 DB 00 C9 21 93 01 ......!..q...!..
0190 71 7B D3 00 C9 60 69 E9 21 B2 01 4E 23 46 2B CD q{...`i.!..N#F+.
01A0 B6 01 21 B4 01 CD D3 01 21 B2 01 71 23 78 E6 7F ..!.....!..q#x..
01B0 77 C9 00 00 19 36 16 09 CD C7 01 CD D3 01 16 02 w....6..........
-
```

Figure 1

```
D2100
2100 0F 0F 0F E6 60 C6 0F 2A 05 1A 4F 06 00 09 7E C6 ....`..*..O...~.
2110 07 0F 0F 0F E6 1F 4F E1 09 E5 0E 12 C3 F8 20 CD ......O....... .
2120 5E 1A E1 AF 32 1A 1A C9 4E 4F 20 49 4E 54 45 52 ^...2...NO INTER
2130 4D 45 44 49 41 54 45 20 4C 41 4E 47 55 41 47 45 MEDIATE LANGUAGE
2140 20 46 49 4C 45 20 24 43 52 55 4E 20 56 45 52 20  FILE $CRUN VER
2150 32 2E 30 35 24 24 24 24 24 24 24 21 00 01 22 24 2.05$$$$$$$!.."$
2160 41 21 5C 00 22 26 41 3A 45 41 1F DA 86 21 CD E6 A!\."&A:EA...!..
2170 17 01 47 21 CD 1A 27 0E 0D CD C4 26 0E 0A CD C4 ..G!..'....&....
2180 26 21 EF 01 36 00 CD CC 24 21 0E 43 22 FC 40 CD &!..6...$!.C".@.
2190 5D 25 4F 3E 02 B9 D2 A2 21 01 56 49 CD 1D 28 CD ]%O>....!.VI..(.
21A0 0D 01 CD 5D 25 32 02 41 FE 2A CA FD 21 21 01 41 ...]%2.A.*..!!.A
21B0 36 00 CD 5D 25 F5 3A 01 41 3C 32 01 41 4F 06 00 6..]%.:.A<2.AO..
-
```

Figure 2

```
D2100
2100 0F 0F 0F E6 60 C6 0F 2A 05 1A 4F 06 00 09 7E C6 ....`..*..O...~.
2110 07 0F 0F 0F E6 1F 4F E1 09 E5 0E 12 C3 F8 20 CD ......O....... .
2120 5E 1A E1 AF 32 1A 1A C9 4E 4F 20 49 4E 54 45 52 ^...2...NO INTER
2130 4D 45 44 49 41 54 45 20 4C 41 4E 47 55 41 47 45 MEDIATE LANGUAGE
2140 20 46 49 4C 45 20 24 04 50 6C 65 61 73 65 20 77  FILE $.Please w
2150 61 69 74 2E 2E 2E 24 24 24 24 24 21 00 01 22 24 ait...$$$$$!.."$
2160 41 21 5C 00 22 26 41 3A 45 41 1F DA 86 21 CD E6 A!\."&A:EA...!..
2170 17 01 47 21 CD 1A 27 0E 0D CD C4 26 0E 0A CD C4 ..G!..'....&....
2180 26 21 EF 01 36 00 CD CC 24 21 0E 43 22 FC 40 CD &!..6...$!.C".@.
2190 5D 25 4F 3E 02 B9 D2 A2 21 01 56 49 CD 1D 28 CD ]%O>....!.VI..(.
21A0 0D 01 CD 5D 25 32 02 41 FE 2A CA FD 21 21 01 41 ...]%2.A.*..!!.A
21B0 36 00 CD 5D 25 F5 3A 01 41 3C 32 01 41 4F 06 00 6..]%.:.A<2.AO..
-
```

Figure 3

```
S2147
2147 43 04
2148 52 50
2149 55 6C
214A 4E 65
214B 20 61
214C 56 73
214D 45 65
214E 52 20
214F 20 77
2150 32 61
2151 2E 69
2152 30 74
2153 35 2E
2154 24 2E
2155 24 2E
2156 24 X
?
-
```

Figure 4

Modifications cont'd...

```
REM     *****************************************\
        *     CBASIC LINE EDITOR.               *\
        * THIS PROGRAM DEMONSTRATES THE CBASIC  *\
        * SINGLE CHARACTER OUTPUT ROUTINE       *\
        *                                       *\
        *       WRITTEN JULY, 1980              *\
        *       ANDY & BEN GALEWSKY             *\
        *****************************************
REM     FIRST DEFINE THE PRINT SINGLE CHARACTER FUNCTION

        DEF FN.PRN(DISP$)
                POKE 130H,ASC(DISP$)           REM CHARACTER TO BE PRINTED
                                               REM IS PASSED IN 130 HEX
                CALL 120H                      REM CALL ROUTINE
                RETURN
        FEND
CLR$=CHR$(4)    REM SCREEN CLEAR CHARACTER FOR VECTOR MINDLESS TERMINAL
                REM CHANGE TO SCREEN CLEAR ON YOUR TERMINAL
REM     START OF PROGRAM
5       PRINT "ENTER LINE TO EDIT "
        INPUT " "; LINE EDIT$
        PRINT CLR$
        PRINT EDIT$      REM DISPLAY LINE AT TOP OF SCREEN
        POINTER%=1       REM CHARACTER POINTER
        DUM=FN.PRN("^") REM PLACE MARKER ON SCREEN
10      INKEY%=CONCHAR% REM GET ONE KEYBOARD CHARACTER
        IF INKEY%=32 THEN 20    REM SPACE BAR
        IF INKEY%=ASC("D") THEN 30      REM DELETE CHARACTER
        IF INKEY%=ASC("C") THEN 40      REM CHANGE
        IF INKEY%=13 THEN 50            REM RETURN
        GOTO 10
20      POINTER%=POINTER%+1             REM INCREMENT POINTER
        DUM=FN.PRN(CHR$(8))             REM MOVE CURSOR BACK
        DUM=FN.PRN(CHR$(8))
        DUM=FN.PRN(" ")                REM ERASE OLD MARKER AND GO FORWARD
        DUM=FN.PRN("^")                 REM PRINT NEW MARKER
        GOTO 10

REM     DELETE CHARACTER

30      LEF$=LEFT$(EDIT$,POINTER%-1)     REM GET LEFT OF DELETION
        RIG$=MID$(EDIT$,POINTER%+1,LEN(EDIT$))  REM GET RIGHT
        PRINT CLR$
        EDIT$=LEF$+RIG$                         REM REBUILD STRING
        PRINT EDIT$                             REM PRINT IT
        POINTER%=POINTER%-1                     REM DECREMENT CHARACTER POIN
        FOR MOV%=1 TO POINTER%-1                REM REPOSITION CURSOR
        DUM=FN.PRN(" ")
        NEXT MOV%
        DUM=FN.PRN("^")                         REM PRINT MARKER
        GOTO 10

REM     CHANGE CHARACTER

40      REPL$=CHR$(CONCHAR%)                    REM GET CHANGE
        LEF$=LEFT$(EDIT$,POINTER%-1)            REM LEFT PART
        RIG$=MID$(EDIT$,POINTER%+1,LEN(EDIT$))  REM RIGHT PART
        PRINT CLR$
        EDIT$=LEF$+REPL$+RIG$                   REM REBUILD STRING
        PRINT EDIT$                             REM PRINT IT

        FOR MOV%=1 TO POINTER%-1                REM REPOSITION POINTER
        DUM=FN.PRN(" ")
        NEXT MOV%
        DUM=FN.PRN("^")                         REM DISPLAY MARKER
        GOTO 10

REM     GET NEW LINE TO EDIT

50      PRINT CLR$
        GOTO 5
```

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)

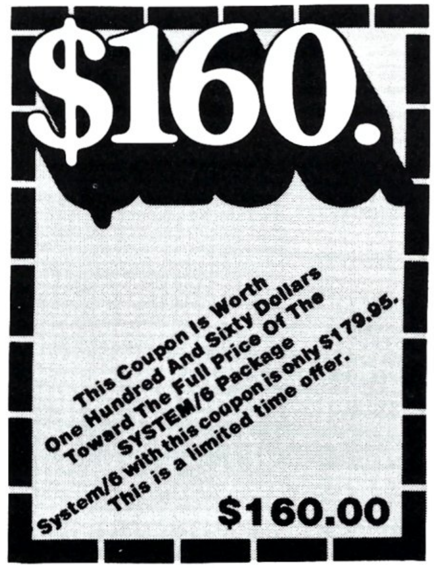
By Osborne. Requires C Basic 2
5" disks $99.95 (manual not included)
8" disks $99.95 (manual not included)
Manual $20.00

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!!



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

NorthStar Topics

*Running North Star DOS and CP/M * Together*

by Randy Reitz

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

I have always been interested in CP/M and its dynamic file management system. Last year I started to experiment with the Lifeboat implementation of CP/M for the North Star controller. Using CP/M is a great change from the North Star DOS. The North Star disk operating system (DOS) only performs directory maintenance and low level disk access whereas CP/M has features that should be found in a DOS such as file open, file close, etc. I became interested in how I could use my North Star programs under CP/M as well as use the CP/M editor to prepare text for my North Star Basic programs. The North Star system has some useful programs for poking around and I thought they would be helpful for exploring CP/M. For example, the North Star Monitor program easily dumps and modifies memory, while with the North Star RD command, the contents of a disk can be examined. So the natural outcome of this was to experiment with CP/M using North Star DOS, North Star Monitor and eventually North Star Basic.

Since the CP/M programs FDOS (Basic Input Output System—BIOS plus the Basic DOS—BDOS) and the Console Command Processor (CCP) are in high memory and the North Star DOS and its programs are at 2000H (in the Transient Program Area—TPA), my first idea was to load both systems and switch between them whenever desired. However, I quickly realized that simply running an unmodified North Star DOS in CP/M's TPA did not work well. When I did something in North Star DOS that required a disk access and then returned to CP/M, I would get unpredictable results. Since computers are supposed to be very predictable, I set out to find what was causing the incompatibility between North Star DOS and CP/M.

I didn't have to look too long to find 4 bytes in the North Star DOS that were causing the problem. I could say the problem was really with the North Star Micro-Disk System (MDS) controller. The North Star controller is simplicity itself. You may have noticed there is no "big" LSI chip on the North Star controller board. Everything on the board is simple, ordinary TTL stuff. This simplicity is deceiving since simple hardware usually requires complicated software. Now, I don't want to say that the North Star DOS software is all that complicated; but since the North Star controller board is simple, the North Star software must do more than software that uses a TARBELL controller. For example, there is no way to query the North Star controller board to find out what drive or track is currently selected. This extremely relevant information must be maintained by the software.

Randy Reitz, 26 Maple St., Chatham Township, NJ 07928

Since the controller board is simple, the North Star DOS software must do more than software that uses a TARBELL controller.

Now, about those 4 bytes in the North Star DOS. In locations 2000H through 2002H, the current track number for each drive in the 3-drive North Star system is stored. In location 2003H, the number of the currently selected drive is stored. CP/M on North Star has to keep this same information in software, but since 2000H-2003H is smack in the middle of the TPA, Lifeboat's BIOS keeps this information elsewhere. This is the problem with running two systems together—these bytes need to be synchronized. You can imagine what happens when CP/M's BIOS looks and sees the drive motors are running (which means a drive has been selected) and checks its memory and finds the requested drive (or track) is selected, then proceeds when North Star DOS just finished with a different drive (or track). This condition guarantees unpredictable results.

Fortunately, the solution for this problem is straight forward. The folks at Lifeboat merely lifted the North Star DOS disk drivers that are in the ROM on the controller board and dropped the software unmodified into their BIOS. It didn't take too much work with a disassembler to find where the drivers were in

Lifeboat's BIOS. Lifeboat tried to discourage me since they inserted an extra byte after each RET and JMP instruction. This drives a disassembler wild; but once I figured out what was going on I could correct for it. It's hard to keep secrets from a good disassembler and a persistant software hack.

The following are the steps required to modify North Star DOS to use the CP/M disk drivers so one set of memory locations are used to keep track of the disk system status. I have been using this "patched" North Star DOS for a while and I can say that it is well behaved. I usually run North Star DOS with CP/M also resident and bounce back and forth easily.

Lifeboat tried to discourage me since they inserted an extra byte after each RET and JMP instruction. This drives a disassembler wild; but once I figured out what was going on I could correct for it. It's hard to keep secrets from a good disassembler and a persistent software hack.

The first 4 steps set up the environment to patch the North Star DOS. The Dynamic Debugging Tool (DDT) program in CP/M is well suited for this work. The Assembly and List (disassemble) commands are useful and the ASCII interpretation in the Dump command is also helpful.

1. Cold start (boot) CP/M for North Star (Lifeboat CP/M 1.4).

2. Cold start North Star DOS release 4.0 or 5.1S.

3. Insert the CP/M disk in drive A (North Star drive 1) and give the North Star DOS command JP 0. (i.e. get back to CP/M).

4. Issue the CP/M DDT command (i.e. start the CP/M dynamic debugging tool).

   Now comes the point of this whole exercise.

5. Patch the North Star DOS in RAM at the following addresses:

4.0	5.1S	Change to	Comment
22C5H		MVI M,0E5H	for N* IN command
22D6H		LXI H,0E500H	for IN command
243BH	2381H	CALL BIOS+480H	use CP/M drivers
2497H	241DH	LDA BIOS+5FAH	use CP/M current
24A2H	2428H	STA BIOS+5FAH	drive selected

The value of BIOS above is calculated as MSIZE* 1024-512 where MSIZE is the size of your CP/M in kilobytes. For example, I have 56K of memory so the largest CP/M I can run is 52K since the Lifeboat BIOS is 4K larger than the regular CP/M BIOS. Hence BIOS = 52*1024-512 = 52736 (CE00H). These changes can be made easily with the DDT A(assemble) command.

6. Patch the North Star I/O area to use FDOS I/O functions. I will discuss below a suggested patch to use.

7. M2000, 2A00,100 (move North Star DOS to 100H).

8. Enter the following DOS mover program with the A(assemble) command:

```
100H    JMP    8E0H        use N*DOS buffer area
8E0H    LXI    H,BIOS+566H    for mover program
8E3H    LXI    D,906H      patch BIOS for 7 bytes
8E6H    MVI    C,7         with code at 906H
8E8H    LDAX   D
8E9H    MOV    M,A
8EAH    INX    H
8EBH    INX    D
8ECH    DCR    C
8EDH    JNZ    8E8H
8F0H    LXI    H,100H      now move N*DOS to
8F3H    LXI    D,2000H     proper location
8F6H    LXI    B,F600H     this is -A00H
8F9H    MOV    A,M
8FAH    STAX   D
8FBH    INX    H
8FCH    INX    D
8FDH    INX    B
8FEH    MOV    A,B
8FFH    ORA    A
900H    JNZ    8F9H
903H    JMP    20A8H       start N*DOS V5.1S
906H    MVI    A,FE        look for -1 command
908H    CMP    C           this is 5.1S single
909H    JZ     2740H       density disk init
90CH    NOP
```

9. Exit DDT with a control-C and execute the CP/M command "SAVE 10 NSTAR.COM".

The reason for moving the North Star DOS program to 100H is to create the NSTAR command on the CP/M disk. When in CP/M, typing NSTAR will load the 20 records (10 pages) saved above and then the CCP jumps to location 100H. At 100H is a JMP 8E0H that executes the patch and mover program (step 8). This program patches 7 bytes in the CP/M BIOS to accomodate the new North Star DOS 5.1S command (-1) for single density disk initialization. This command was added to the DCOM entry in 5.1S so a disk could be initialized without using a buffer outside the North Star DOS.This BIOS patch isn't needed if you are using release 4.0, but it can still be put in since release 4.0 will not recognize the -1 command. After patching BIOS, the program moves the North Star DOS to 2000H and starts the DOS at the point in the cold start routine that calls TINIT. This will execute whatever initialization routine you have provided as well as check the "auto" start byte. Hence, you could have the DOS do a "GO BASIC, 2" immediately. I located this patch and mover program in the middle of the disk buffer in the North Star DOS. The jump at 903H to start the North Star DOS should be JMP 208AH if you are using release 4.

When running North Star DOS with the CP/M disk driver, you should not have any problems if you are careful not to disturb the CP/M FDOS (BDOS + BIOS) and the bytes at 0-3 that contain the JMP WARM to get back to CP/M and the IOBYTE.

One motivation for running North Star DOS and CP/M together is to use the North Star monitor and

North Star, cont'd...

North Star Basic programs to experiment with CP/M. Another reason is to use North Star Basic to move files from a North Star disk to CP/M and vice versa. Following is a segment from a North Star Basic program that allows North Star Basic access to the CP/M FDOS facilities. The most important feature of this program is the assembly routine that provides the North Star Basic interface to CP/M. The segment of the program that does this is given below:

```
2  DEF FNC(N,D)
3      FILL 64,N
4  RETURN CALL (65,D)
5  FNEND
6  DATA 58,64,0,79,205,5,0,96,111,201,0
7  F=92
8  FILL F,0
9  FOR I=1 TO 11
10     FILL F+I,ASC("?")
11     READ X \ FILL 64+I,X
12 NEXT
13 R=FNC(13,F)
```

North Star Basic provides a method for accessing user written assembly language subroutines by using the CALL command. The CP/M FDOS can be considered such a subroutine. So a North Star Basic program can use FDOS to do the disk functions necessary to manipulate CP/M files. The FDOS cannot be called directly by North Star Basic since the conventions for passing arguments in the 8080 registers don't agree for North Star and CP/M. Hence, another small assembly language program is needed to adjust the 8080 registers.

The North Star Basic CALL command can contain one or two arguments. The first argument is a numeric value between 0 and 65535 that is the decimal value of the memory address at the beginning of the assembly language subroutine. If a second argument is used, it will be converted to an integer value between 0 and 65535 and placed in the DE register pair. Since the CALL command is a Basic function, it will return a value. The value returned is an integer from 0 to 65535 that represents the value in the HL register pair when the assembly language subroutine returns.

The CP/M FDOS entry point is at address 5. CP/M requires a function number in the C register. Any address information that the CP/M function requires should be in the DE register pair. CP/M returns single byte results in the A register. If a double byte result is returned, the high order byte is in the B register and the low order byte is in the A register. Now that the register conventions are known, it is simple to write a assembly language program that North Star Basic can use to access CP/M FDOS:

```
ADDRESS-CODE  LABEL     OPCODE ARGUMENTS

40H 00        FUNCTION  DB           0
41H 3A4000    DOCPM     LDA          FUNCTION
44H 4F                  MOV          C,A
45H CD0500              CALL         5
48H 60                  MOV          H,B
49H 6F                  MOV          L,A
4AH C9                  RET
```

The first byte of this program is used to pass the FDOS function value. This value is put in the C register and FDOS is called. When CP/M returns, the return code is put in the HL register pair. Now look at the multi-line function in the North Star Basic program (lines 2-5). This function expects two arguments, N and D. The first argument, N, is the CP/M FDOS function number and is "poked" into the "FUNCTION" byte in the assembly language program above. The next argument, D, is used for address information. The CALL to the interface program is made with D as the second argument. Recall that the North Star Basic CALL command will put the second argument in the DE register pair, just where CP/M FDOS expects the argument to be,so no adjustment is required. The Basic "FNC" function expects the interface program to be at address 64. Address 64 stores the CP/M function value for the interface program and the CALL is made to address 65. CP/M provides a 16-byte space starting at 64 for the user's CBIOS. If your CBIOS doesn't use these 16-bytes, you can use it for the interface program. Finally, the interface program sets up the HL register pair with the CP/M return code and returns to North Star Basic.

The other important part of the Basic program is line 6, that contains the assembly code for the interface program, and lines 8-12 that put the assembly code into memory starting at address 65. This program is using the CP/M default FCB that is at address 5CH (92 decimal).

Now I'll show you a North Star Basic program that will move a text file from North Star disk to a CP/M disk in drive A (North Star drive 1).

This program begins the same way as the last one. On line 13 the name of the North Star file is requested. If the file does not exist, the name is requested again. Next, the CP/M filename is requested. This name must be less than 12 characters to be valid. Lines 18 to 22 move the CP/M filename from the string C$ to the FCB. Notice that a "." is removed and the CP/M file type is loaded into "FT" field of the FCB. Lines 23-25 set up the CP/M file. If desired, this section of the program could detect if the CP/M file already exists, and if so, request permission to delete it. The CP/M file will be created on drive A. Line 27 opens the North Star file.

The file is transferred one byte at a time in the main loop (lines 28 to 39). Each CP/M sector of 128 bytes is loaded into the default buffer (lines 28 to 34) and then written to the CP/M disk (line 37). If more data remains (test in line 39) the main loop continues. Finally, the CP/M file is closed in line 40.

This program expects the North Star file to contain text that is separated by carriage returns. The program inserts a line feed character after each carriage return so the CP/M editor can be used. The North Star end-of-file is an SOH character (ASCII 1). When this is found, the CP/M end-of-file SUB character (ASCII 26) is substituted. The record loop from line 28 to 34 could be changed to accomodate any North Star data file format you desire. The program ends in line 43 by returning to CP/M. I included this to show that since the CP/M warm start entry doesn't require any data in the 8080 registers, the interface program is not required.

```
1  REM MOVE N* TEXT FILE TO CP/M
2  DEF FNC(N,D)
3      FILL 64,N
```

```
4  RETURN CALL (65,D)
5  FNEND
6  DATA 58,64,0,79,205,5,0,96,111,201,0
7  F=92 \ O=0 \ B1=0 \ W=0 \ DIM C$(16)
8  FILL F,0
9  FOR I=1 TO 11
10     FILL F+I,32
11     READ X \ FILL 64+I,X
12 NEXT
13 INPUT "N* FILENAME= ",I$
14 T=FILE(I$) \ IF T>0 THEN 16
15 PRINT I$," -- NOT FOUND" \ GOTO 13
16 INPUT "CP/M FILENAME= ",C$
17 IF LEN(C$)>12 THEN 46
18 FOR I=1 TO LEN(C$)
19 IF C$(I,I)="." THEN 21
20 FILL F+O+I,ASC(C$(I,I)) \ GOTO 22
21 O=8-I
22 NEXT
23 R=FNC(13,0) \ REM RESET CP/M
24 R=FNC(19,F) \ REM DELETE CP/M FILE
25 R=FNC(22,F) \ REM CREATE CP/M FILE
26 T$="CREATE" \ IF R>=128 THEN 44
27 OPEN #0%T,I$,L \ L=2*L
28 FOR I=128 TO 255
29 IF B1=13 THEN 33 \ REM END-OF-LINE
30 READ #0,&B1 \ PRINT CHR$(B1),
31 FILL I,B1 \ IF B1>1 THEN 34
32 FILL I,26 \ EXIT 35 \ REM END-OF-FILE
33 FILL I,10 \ B1=0 \ PRINT \ REM ADD LINE FEED
34 NEXT
35 PRINT
36 PRINT "WRITTING CP/M RECORD # ",EXAM(F+32)
37 R=FNC(21,F)
38 T$="WRITE" \ IF R>0 THEN 44
39 W=W+1 \ L=L-1 \ IF L>0 AND B1<>1 THEN 28
40 R=FNC(16,F)
41 T$="CLOSE" \ IF R=255 THEN 44
42 PRINT "TRANSFER COMPLETE,",W," CP/M RECORDS"
43 PRINT "RETURNING TO CP/M" \ W=CALL(0)
44 PRINT "CP/M ERROR ON ",T$,
45 PRINT "RETURN CODE =",R \ STOP
46 PRINT "CP/M FILENAME TOO LONG"
47 STOP
48 END
```

This is a small example to show what can be done with North Star Basic by using the CP/M FDOS. It isn't difficult to modify this program to transfer files in either direction. So you might suspect that you could use the CP/M editor to prepare the text for a North Star Basic program and then transfer the program to North Star DOS. However, getting the text of the program into North Star Basic is a little tricky.

As a final topic, let me discuss the possibilities that arise when the CP/M FDOS facility is used to implement the North Star input/output routines. The North Star DOS provides a one-page (256 bytes) block at the end of the DOS to carry out four I/O functions:

- COUT —character output
- CIN —character input
- TINIT —terminal initialization
- CONTC —control-C detection.

At entry to the COUT and CIN routines, the A-reg contains a number that represents the device the routine should use to do the I/O. The CP/M FDOS facility uses a code in the C-reg to indicate the function requested. So, one consideration is simply to set up the information in the proper 8080 registers. For example, here is what the North Star COUT routine could look like:

```
                ORG     2900H
        ;
        ;       N* I/O USING CP/M BIOS
        ;
COUT:   PUSH    B               ;B-reg contains char
        ORA     A               ;to output, A-reg is
        MVI     C,2             ;0 for console so use
        JZ      $+5             ;FDOS function 2
        MVI     C,5             ;function 5 otherwise
        LDA     ECHO            ;check if this character
        CMP     B               ;was just read with CIN
        CNZ     DOCPM           ;output character if not
        MVI     A,0FFH          ;reset ECHO flag
        STA     ECHO
        POP     B
        MOV     A,B             ;N* expects char in A-reg
        RET
```

The B-reg contains the character to output when the COUT routine is called. Since CP/M FDOS expects to find the character in the E-reg when function code 2 (output to console) or code 5 (output to list) is used, the DOCPM routine will make the adjustment. The only other consideration is that FDOS automatically "echoes" characters typed so the COUT routine should not. Therefore, the COUT routine compares the character it is about to output with the last character received by CIN.

Next, consider the character input routine:

```
CIN:    PUSH    B               ;can't destroy any regs
        MOV     C,A             ;save input device number
        LDA     NOFILE          ;check if an "input file"
        ORA     A               ;is active
        JZ      READCPM         ;take all input from file
                                ;if active
READEV: MOV     A,C             ;restore input device
        ORA     A               ;input from console if 0
        MVI     C,1             ;use FDOS function code 1
        JZ      $+5             ;for console, code 3
        MVI     C,3             ;otherwise
        CALL    DOCPM
        STA     ECHO            ;set ECHO flag
        POP     B
        RET
```

Again, this routine is straightforward. The feature added to normal character input is the capability to read a CP/M file. In the implementation of North Star DOS for CP/M that I have been presenting, the North Star DOS exists as a CP/M command (COM) file. When the CP/M console command program (CCP) receives a string that does not start with the name of any built-in command, the CCP assumes that there is a file on the currently logged-in disk with the name given and an extension of COM. If this is so, the CCP loads the contents of the file into the TPA and sets up the default buffer at 80H with the remaining characters that were typed before the carriage return. CP/M programs usually understand these "arguments" to be CP/M file name(s). Hence, the North Star DOS can be considered a CP/M command that executes in the TPA and will accept a CP/M file name as an argument.

When the North Star DOS begins execution, the terminal initialization routine first gets control. Since CP/M has been running, the terminal doesn't need to be initialized. This routine can be used to check if a CP/M file name has been passed as an argument. For example:

```
TINIT:  MVI     A,0FFH          ;initialize some flags
        STA     CASE            ;my upper/lower case flag
        STA     ECHO
        STA     NOFILE          ;assume no CP/M file
        OUT     0FFH            ;IMSAI front panel lights
        LDA     BUFF            ;look in default buffer
        ORA     A               ;for a CP/M file name
        RZ                      ;all done if no file
        LXI     D,FCB           ;prepare to "open" file
        MVI     C,15
        CALL    FDOS
        CPI     255             ;check if successful
        RZ                      ;return if no good
        XRA     A               ;else indicate a CP/M
        STA     NOFILE          ;file is active
        STA     FCBCR           ;start at record 0
        MVI     A,80H           ;force file read routine
        STA     IBP             ;to get a new sector
        RET
```

North Star, cont'd...

If the North Star DOS was "called" by the CCP with a file name as an argument, the TINIT routine will open the file and if the open is successful, TINIT will reset the NOFILE flag so CIN is forced to read characters from the given CP/M file.

Notice the CIN routine above will jump to the READCPM routine if a CP/M file is active. This routine follows:

```
READCPM:PUSH  B        ;save all registers
        PUSH  D
        PUSH  H
        MVI   C,11     ;check if any key has
        CALL  FDOS     ;been hit on keyboard
        RRC            ;abort CP/M file if so
        JC    FINIS
SKIPLF:LDA    IBP      ;get address of next
        CPI   80H      ;char in input buffer
        CZ    DISKR    ;read another sector if
        MOV   E,A      ;required - set up (DE)
        MVI   D,0      ;to offset of char in
        INR   A        ;buffer - update input
        STA   IBP      ;buffer pointer
        LXI   H,BUFF   ;address of default buf
        DAD   D        ;(HL) now points to next
        MOV   A,M      ;character
        CPI   10       ;look for line feed
        JZ    SKIPLF   ;skip line feeds
        CPI   26       ;look for CP/M eof
        JZ    FINIS    ;terminate CP/M file
        POP   H
        POP   D
        POP   B        ;BC pushed in CIN
        POP   B        ;restore registers and
        RET            ;return char to N*
```

When a CP/M file is active, all input requested by North Star will be taken one character at a time from the CP/M file until the end of file is reached. Subsequent input will then be taken from the input device specified in the A-reg when CIN is called. The READCPM routine uses two subroutines:

```
DISKR:  LXI   D,FCB    ;read the next sector
        MVI   C,20     ;of the CP/M file into
        CALL  FDOS     ;the default buffer
        CPI   0        ;check for read error
        RZ             ;return if no problem
        POP   H        ;clear return address
FINIS:  MVI   A,0FFH   ;set NOFILE flag
        STA   NOFILE
        LXI   D,FCB    ;close CP/M file
        MVI   C,16
        CALL  FDOS
        POP   H        ;restore registers
        POP   D
        POP   B
        JMP   READEV   ;return to CIN routine
DOCPM:  PUSH  D        ;do FDOS function
        PUSH  H
        MOV   E,B      ;adjust register
        CALL  FDOS
        POP   H
        POP   D
        RET
```

The DOCPM routine is used by COUT and CIN to execute the selected FDOS function. The only tricky code above is that when DISKR returns successfully, the A-reg is 0 so IBP will be properly initialized for the new sector.

The only other routine required in the North Star DOS I/O is the control-C detection routine. Here it is:

```
CONTC: MVI    C,11     ;see if a key has been
       CALL   FDOS     ;hit on the keyboard
       ANI    1        ;return with Z-flag
       XRI    1        ;reset if no ^C
       RNZ
       MVI    C,1      ;a key has been hit
       CALL   FDOS     ;get character
       CPI    3        ;look for ^C
       STC             ;tell CP/M char was read
       RET
```

These routines use the following symbols:

```
ECHO    DB    0       ;don't ECHO character
NOFILE  DB    0       ;no CP/M file when 0
IBP     DB    0       ;input buffer pointer
CASE    EQU   0D5F9   ;subject for future
FDOS    EQU   5       ;CP/M entry point
FCB     EQU   5CH     ;default FCB address
BUFF    EQU   80H     ;default buffer addr
```

At this point you might question the usefulness of this discussion. I mentioned above that I could use the CP/M text editor to prepare North Star Basic programs. With this driver for the North Star DOS I/O, you can do the following:

1. Prepare the text of a North Star Basic program using the CP/M editor.
2. Make the first line of this file the North Star command "GO BASIC, 2".
3. Type the CP/M command "NSTAR <filename> .TXT".
4. Sit back and watch North Star Basic read in the program you have prepared.

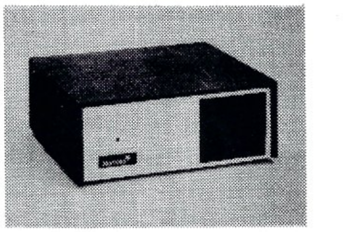
The North Star DOS and any programs running under it will treat the CP/M file as a command file. The file will be read until an end-of-file condition is encountered, then all input will be taken from the device specified in the A-reg when CIN is entered.

This completes my discussions of using North Star DOS and CP/M together. If you would like to try this, but don't want to do it yourself, for $15 I will supply a diskette containing the NSTAR command and some North Star Basic programs to demonstrate what can be done. You must have the Lifeboat CP/M 1.4 for single density North Star. □

SAVE MORE THAN 20%

NORTH STAR — INTERTUBE — MICROTEK
ZENITH — RCA-COSMAC — ITHACA
THINKER TOYS — GODBOUT — SUPERBRAIN

The smartest computers at the smartest price



FACTORY ASSEMBLED & TESTED	LIST	ONLY
HORIZON-1-32K-DOUBLE DEN	$2695	$1994
HORIZON-2-32K-DOUBLE DEN	3095	2299
HORIZON-2-32K-QUAD DENSITY	3595	2699
HORIZON-2-64K-QUAD+HARD DISK	9329	7199
HORIZON MEMORY ASSM	16K 389	32K 579
HORIZON MEMORY KIT	16K 359	32K 535
NORTH STAR HARD DISK 18 Mb	4999	3939
PASCAL FOR NORTH STAR ON DISK	199	190
Powerful NORTH STAR BASIC..The Best		FREE
2 NORTH STAR SOFTWARE DISKS w/HORIZON		FREE
NSSE 1-22 & P01 TERRIFIC PROGRAMS	ONLY	10
NORTHWORD 299 MAILMAN 239 INFOMAN		369
RCA-COSMAC VP-111 99 RCA-COSMAC VP-711		189
COLOR! RAINBOW 385 CAT-100 1369 SPECTRUM		289
ITHACA FRONT PANEL COMPUTER 64K	3195	2695
Z-8001 CPU CARD 16-bit ITHACA S-100 8Mb		1179
ITHACA MEMORY 8/16-bit 64K	995	845
SEATTLE 8086 CPU 16 bit 10 x faster		556
SEATTLE MEMORY 8/16 BIT 16K 4Mhz		356
SSM KITS Z-80 CPU 221 VIDEO BRD VB3 4Mhz		412
MEASUREMENT MEMORY 64K A & T 4mHz		599
MEASUREMENT MEMORY 64K BANK SELECT		789
ECONORAM XIV UNKIT 16K	299	254
CENTRAL DATA 64K RAM	665	599
DISCUS/2D A & T + CP/M	1199	975
THINKER TOYS HARD DISK 26 Mb	4995	3995
DISCUS/2+2 1.2 Mbytes A & T	1549	1285
TARBELL DISK CONTROLLER DD	495	445
TARBELL CASETTE INTERFACE KIT	120	109
SUPERBRAIN	2995	2395
SUPERBRAIN QUAD DENSITY	3995	2995

ZENITH-HEATH Z-89 48K	2895	2495
INTERTUBE II SMART TERMINAL	995	725
ZENITH-HEATH SMART TERMINAL	995	739
ZENITH-HEATH WH-11 16bit COMPUTER		2995
CAT NOVATION MODEM	179	169
MICROTEK PRINTER	795	675
ANADEX PRINTER DP-8000	995	865
ANADEX PRINTER DP-9500-1	1650	1389
NEC PRINTER Fast Typewriter Quality	2915	2799
SECRETARY WORD PROCESSOR The Best!	85	77
TEXTWRITER III Book Writing Program	125	112
GOFAST NORTH STAR BASIC Speeder Upper	79	71
PDS SUPER Z-80 ASSEMBLER & More	99	89
SUPER BASIC DEBUGGER $89 COMPILER 135 HDS		40
EZ-80 MACHINE LANGUAGE TUTOR 25 STATISTICS		190
EZ-CODER Translates English to BASIC	79	71
ECOSOFT FULL ACCOUNTING PKG	350	315
BOX OF DISKETTES 29 IN PLASTIC CASE		30
Which Computers are BEST? BROCHURE		FREE
North Star Documentation refundable w/HRZ		20

ORDER 2 or more COMPUTERS....BIGGER DISCOUNTS
YES WE WILL BEAT OUR COMPETITION'S PRICE!
FACTORY ASSEMBLED & FACTORY WARRANTY

AMERICAN SQUARE COMPUTERS

KIVETT DR • JAMESTOWN NC 27282
(919)-889-4577

CATCH THE S-100 INC. BUS!

	LIST PRICE EACH	OUR SPECIAL CASH PRICE
TARBELL DOUBLE DENSITY DISK CONTROLLER — A & T	495.00	399.00
S.D. SYSTEMS VERSAFLOPPY II — KIT	350.00	299.00
GODBOUT CPU-Z — A & T	295.00	249.00
MULLEN EXTENDER CARD W/PROBE — KIT	59.00	49.00
POTOMAC MICRO MAGIC MODEM — A & T	399.00	341.00
3M "SCOTCH" 8" 740-0 DISKETTES — 10	50.00	30.00
CROMEMCO 32K BYTESAVER — A & T	295.00	249.00
CCS 64K DYNAMIC RAM — A & T	700.00	549.00

Subject to Available Quantities • Prices Quoted Include Cash Discounts.
Shipping & Insurance Extra.

We carry all major lines such as
S.D. Systems, Cromemco, Ithaca Intersystems, North Star,
Sanyo, ECT, TEI, Godbout, Thinker Toys, SSM.
For a special cash price, telephone us.

S-100, inc.

7 White Place, Clark, N.J. 07066
201-382-1318
Hours: Mon. - Fri. — 10 a.m. to 6 p.m.

CP/M Systems Compatable 8080/Z80 Software

		$SOURCE	/OBJECT
	STANDARD UTILITIES		
COMMX	—Intercomputer and timeshare communications package	250.00	75.00
D	—Directory files entry/space/disk status 4 column sort	40.00	20.00
DDB	—Disk directory database of your programs. Update/inquiry	60.00	25.00
DCOMP	—Disk file compare with another disk file	30.00	20.00
MCOMP	—Memory compare with memory (EPROM to RAM or EPROM)	30.00	20.00
MTEST	—Memory test with before/after write error bits +pass #	30.00	20.00
	ADVANCED UTILITIES		
CDIR	—Comprehensive directory/file allocation verification	30.00	20.00
COPSEQ	—Define and copy disk file to CP/M file	30.00	20.00
DASM	—8080 object dis-assembler with symbol table/XREF/map	100.00	40.00
GEFIT	—One pass gang replacement non-destructive editor	50.00	20.00
PREDIT	—Program version number maintenance at pre-edit time	40.00	20.00
RELOC	—8080 Object code relocator (known data areas entered)	30.00	20.00
	SUPER SUBROUTINES		
DLOG	—Data buffer logged direct to end of disk file requested	40.00	
POLAR	—Converts origin/angle/radius to X/Y coordinates fast!	40.00	
REDIX	—Alphanumeric radix bin data sort with no data movement	30.00	
SHELL	—Shell metzner data sort with minimal overhead/movement	30.00	
	MEMORY MAPPED VIDEO		
CGEN	—Video character generator/editor for on screen updates	50.00	20.00
DXAM	—Disk examine in ASCII/EBDIC/HEX with sector update	40.00	20.00
VBASIC	—9K Video/disk Basic + full screen basic source editor		100.00
PMIS	—Program management information system written in VBasic	200.00	
VIDEO	—Flexible multi-user/window super video driver	50.00	
VDRAW	—Vector line draw and plot subroutine for fast graphics	30.00	
CHESS	—Graphic games for IMSAI VIO + VG Flashwriter 2 + SSM VB3	30.00	20.00
INVADERS	—Zap!/sound effects/joystick or buttons or console	50.00	20.00
STARTREK	—Realtime action/sound effects with host of commands.	40.00	20.00
TARGET	—Moving aircraft shooting gallery with speed options	40.00	20.00

HAWKEYE GRAFIX
23914 Mobile
Canoga Park, CA 91307

Disk $7.50 extra — CA residents add 6% sales tax. Send your disk! — S. DEN 8" +MICROP+APPLE+NSTAR. Dial (213) 348-7909 to get free product brochure.

Directory Program for CP/M® Systems

by Mark M. Zeiger

Like most computerists, I like things to be neat and orderly. I also like convenience, and having a large disk directory scroll off the screen before I can find what I'm looking for is not convenient. Therefore I was overjoyed when I discovered the CP/M Users Library had a program called **XDIR** that would output an alphabetized directory using the whole screen. I got a copy and literally ran home to try it out on my North Star CP/M system. Goodbye disk! I then tried it on a friend's eight inch double density system. While it didn't blow the disk, it also did not list the directory. Evidently the program was not CP/M compatible. However, once I had seen such a program, I had to have one for myself.

The program I have written is completely CP/M compatible. This means it does everything by using standard calls to the CP/M BDOS (which on most systems has its entry point at address 5 - if it doesn't, then it is not a "true" CP/M system). The only changes that have to be made from system to system are the commands that clear the screen and the tab control character (although the latter is pretty standard on most hardware).

The program has a number of "goodies"; the nicest being the Shell-Metzner sort. This sort is a fourteen line Basic program and it is not that much longer in assembly language. For a maximum of sixty-four items (the largest directory allowable in CP/M) almost any machine language sort would have been unnoticable timewise. As near as I can tell, the Shell-Metzner sort takes less than one-quarter of a second.

The maximum of sixty-four entries is perfect for a 80 x 24 screen. The heading and the line skipped after it along with the twenty lines of names will fit on the screen and still allow the CP/M prompt to be shown at the bottom without the screen scrolling. If there is a sixty-fourth entry, it will be shown at the bottom of the third column. If you would like to adapt the program to a VDM, it would be easy to do if there are not more than forty entries in the directory. More than that will defeat the purpose of the entire program unless you put a pause after sixteen lines are printed.

I did try to make the program structured, and at first it was very much so. But naturally, as a few more things were added, the structure started to disappear. Below are the major routines that are called sequentially at the beginning of the program and some of the more important subroutines:

```
CKDRIVE  =======>          See if drive is requested,
                           else use logged-in drive.
                           Store drive name in head-
                           ing.
SIGNON ========>           Print heading message.
GETNAME =======>           Checks to see if file name
                           and/or type was request-
                           ed; else finds all files.
CLEARBUFF  =====>          Clears RAM where names
                           are to be stored for sorting
                           and output.
SEARCHRT ======>           Searches directory for
 [                         names.
 [- MOD4 -------------->   Finds address of directory
 [                         FCB in DMA
 [- TRANS ------------>    Moves name and number
                           of records to area in RAM
                           where all names are to be
                           stored contiguously.

EXTSCH ========>           Searches for file extents.
 [                         Notes the existence of
 [                         extent, the number of
 [                         extents, and the number
 [                         of records in the last
 [                         extent.
 [- MATCH ------------>    Searches buffer for match-
                           ing 0th extent.
RECINDEC  ======>          Divides by two to get the
 [                         numbers of sectors (256
 [                         bytes). Then calculates
 [                         total number of records in
 [                         file in decimal. Puts num-
 [                         ber next to name with
 [                         leading zeros surpressed.
 [- ADDEXT ----------->    Adds 64 decimal to num-
                           ber of sectors for each
                           extent.
SORT  =========>           Shell-Metzner Sort
 [- COMPARE ------->       Compares the two names.
 [- SWITCH ---------->     Switches names in buffer
                           if required.
PRINTOUT  ======>          Prints names in three col-
 [                         umns.
 [- WRITENAME ---->        Checks to see if name in
                           directory buffer has been
                           output to screen.
```

Mark M. Zeiger, 198-01B 67th St., Flushing, NY 11365

The only routine I will explain in detail is the "search" routine. When a search or "search next" is requested, CP/M loads the directory file control blocks into the DMA address (defaults in this program to 80H) in groups of four. These FCB's include files which have been erased as well as extents (which are not usually contiguous with the zeroth extent on the disk). The accumulator then returns a number which when divided by four will give a remainder that is the thirty-two byte part of the DMA address where the directory FCB is located. Thus, the remainders of 0, 1, 2, or 3 will correspond to 80H, 0A0H, 0C0H, 0E0H as the location of the FCB in the DMA address if the address is set at 80H. The MOD4 routine does this calculation. A 0FFH means the file does not exist. Extents have to be searched for as different files. Therefore, when first searching for the occurence of a file, the DE registers must point to a RAM address containing the name and extent of the file(s) being sought. The "search next" routine will then get other occurences of that file name (assuming, of course, that the filename is a wildcard). To search for extents, the DE registers must again point to the filename with the new extent and the initial search and the "search next(s)" must be requested.

I hope that you will enjoy the convience of this program as much as I have. One of the nice things about it is that it is slightly less than 1K of object code. This means that it will use the minimum amount of disk space (important for those of us who have mini-floppies). And also, is there anyone out there who knows how to calculate the amount of space left on a disk? The CP/M STAT program does it with a call to BDOS using 27 in the C-register, but I can not figure out the details of the routine. I would like to put it in this program. If anyone knows, I would appreciate hearing from you .

```
;CP/M DIRECTORY LIST PROGRAM

;COPYRIGHT 1979 BY MARK M. ZEIGER

;THIS PROGRAM WILL LIST AN ALPHABETIZED DIRECTORY OF A
;CP/M 1.4 DISK IN A FORMATTED OUTPUT ON A 80 X 24 SCREEN.
;NEXT TO EACH FILENAME IS THE NUMBER OF 256 BYTE PAGES
;IN THE FILE. THIS PROGRAM WILL WORK FOR ANY TYPE OF CP/M,
;WHETHER THE DISKS ARE IBM COMPATABLE FORMAT OR NOT, BECAUSE
;ALL DISK ACCESSES ARE DONE BY STANDARD CP/M FUNCTION CALLS.

;TO USE THE PROGRAM, JUST TYPE "XDIR". ALL FILES ON THE DEFAULT
;DRIVE WILL BE LISTED. IF YOU WISH TO EXAMINE ANOTHER DRIVE,
;SAY DRIVE B, TYPE "XDIR B:". IF YOU WISH TO LIST ONLY CERTAIN
;FILES, SUCH AS ALL COM FILES, TYPE "XDIR *.COM".

;REVISED 9/80 BY HARVEY FISHMAN TO WORK FOR CP/M 2 EXTENSION FORMATS
```

```
0100                        ORG 100H

0100 C32D01                 JMP START

0005 =          BDOS        EQU 5
0011 =          SEARCH      EQU 17
0012 =          NXTSCH      EQU 18
0009 =          WRTBUF      EQU 9
0002 =          CONOUT      EQU 2
005C =          FCB         EQU 5CH
0015 =          NOLINES     EQU 21

0103 3F         PRNTCNT     DB 63
0104 00         DIRCNT      DB 0
0105            DESAVE      DS 2
0107            STKSV       DS 2
0109            WRTNUM      DS 1

    ;FCB FOR SEARCH ROUTINE. SEARCHES FOR
    ;ALL FILES UNLESS CHANGED BY GETNAME.

010A 003F3F3F   ANYNAME DB 0,'???????????',0,0,0,0,0
011B 00000000           DB 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

012D 210000     START   LXI H,0                     ;SAVE STACK
0130 39                 DAD SP
0131 220701             SHLD STKSV
0134 319B05             LXI SP,NEWSTK
```

```
0137 CD5701            CALL CKDRIVE
013A CD7301            CALL SIGNON
013D CDCA01            CALL GETNAME
0140 CDB601            CALL CLEARBUFF
0143 CDDC01            CALL SEARCHRT        ;MAIN PROGRAM
0146 CD6202            CALL EXTSCH
0149 CDE602            CALL RECINDEC
014C CD4204            CALL SORT
014F CD6103            CALL PRINTOUT

0152 2A0701            LHLD STKSV           ;RELOAD CP/M'S STACK
0155 F9                SPHL
0156 C9                RET                  ;RETURN TO CP/M

0157 3A5C00    CKDRIVE LDA FCB              ;GET DRIVE NUMBER IN FCB
015A 320A01            STA ANYNAME
015D FE00              CPI 0                ;IF DRIVE IS ZERO, THEN..
015F CA6801            JZ LOGDSK            ;..CALCULATE LOGGED-IN DRIVE.
0162 C640              ADI 40H              ;CHANGE TO ASCII
0164 329701            STA DRMSG
0167 C9                RET
0168 0E19      LOGDSK  MVI C,25             ;CP/M GET CURRENT DRIVE CALL
016A CD0500            CALL BDOS
016D C641              ADI 41H              ;CHANGE TO ASCII
016F 329701            STA DRMSG
0172 C9                RET

0173 117A01    SIGNON  LXI D,ONMSG          ;SCREEN CLEAR AND..
0176 CDFB03            CALL WRITOUT         ;..PRINT HEADING.
0179 C9                RET

017A 1A0C2044  ONMSG   DB 1AH, 0CH, ' Directory      ',9,'     ','Drive  '

0197 =         DRMSG   EQU $-1

0198 0909436F          DB 9,9, 'Copyright 1979  M. Zeiger',0DH,0AH,0

CLEARBUFF

   ;DIRECTORY BUFFER FILLED WITH SPACES. BUFFER IS 5 PAGES
   ;AT TOP OF PROGRAM.

01B6 3E20              MVI A,' '
01B8 219C05            LXI H,DIRBUFF
01BB 1605              MVI D,5
01BD 1E00      LP1     MVI E,0
01BF 77        LP2     MOV M,A
01C0 23                INX H
01C1 1C                INR E
01C2 C2BF01            JNZ LP2
01C5 15                DCR D
01C6 C2BD01            JNZ LP1
01C9 C9                RET

GETNAME

   ;PUTS FILE NAME AMD/OR TYPE INTO SEARCH FCB. IF DEFAULT FCB
   ;OF COMMAND LINE IS BLANK, THEN LEAVE SEARCH FCB WITH "?"S.

01CA 3A5D00            LDA FCB+1
01CD FE20              CPI ' '
01CF C8                RZ
01D0 215D00            LXI H,FCB+1
01D3 110B01            LXI D,ANYNAME+1
01D6 060B              MVI B,11
01D8 CDF904            CALL HTD
01DB C9                RET

SEARCHRT

   ;SEARCHES FOR NAMES AND TRANSFERS TO BUFFER ABOVE PROGRAM.

01DC 219C05            LXI H,DIRBUFF        ;SAVE ADDRESS..
01DF 220501            SHLD DESAVE          ;..OF DIRBUFF
01E2 0E11              MVI C,SEARCH         ;SEARCH FOR FIRST..
01E4 110A01            LXI D,ANYNAME        ;..OCCURANCE.
01E7 CD0500            CALL BDOS
01EA FEFF      CHECK1  CPI 0FFH             ;IF FIRST SEARCH FAILS..
01EC CA1302            JZ NODIR             ;..NO ENTRY EXISTS.

   ;FIND NEXT VALID FILENAME IN DMA. MULTIPLY IT BY 32, THE LENGTH
   ;OF A DISK DIRECTORY. CALCULATE ITS ADDRESS IN DMA IN HL REG.

01EF CD5502    LOOP2   CALL MOD4
01F2 E5                PUSH H               ;ZERO THE 3 BYTES AFTER NAME
01F3 D5                PUSH D
01F4 110C00            LXI D,12
01F7 19                DAD D
01F8 3600              MVI M,0
01FA 23                INX H
01FB 3600              MVI M,0
01FD 23                INX H
01FE 3600              MVI M,0
0200 D1                POP D
0201 E1                POP H
0202 CD3402            CALL TRANS           ;TRANSFER TO DIRBUFF
0205 0E12              MVI C,NXTSCH         ;SEARCH FOR NEXT ENTRY
0207 110A01            LXI D,ANYNAME
020A CD0500            CALL BDOS
020D FEFF              CPI 0FFH             ;NO MORE NAMES IF 0FFH
020F C2EF01            JNZ LOOP2
0212 C9                RET

0213 112002    NODIR   LXI D,NODIRMSG       ;COULDN'T FIND THE ENTRY
0216 0E09              MVI C,WRTBUF
0218 CD0500            CALL BDOS
021B 2A0701            LHLD STKSV
021E F9                SPHL
021F C9                RET

0220 0D0A0A4E  NODIRMSG DB 0DH,0AH,0AH,'No entry found',0DH,0AH,'$'

TRANS

   ;STORES A DISK FCB ALONG WITH NUMBER OF RECORDS IN THE
   ;NEXT 16 BYTES OF DIRBUFF.

0234 F5                PUSH PSW
0235 D5                PUSH D
0236 0610              MVI B,16
0238 E5                PUSH H               ;GET THE NEXT ADDRESS..
0239 2A0501            LHLD DESAVE          ;..OF FILE IN DIRBUFF.
023C EB                XCHG                 ;PUT IT INTO DE REG
023D E1                POP H
023E 7E        LOOP1   MOV A,M              ;DO THE TRANSFER
023F 12                STAX D
0240 23                INX H
0241 13                INX D
0242 05                DCR B
0243 C23E02            JNZ LOOP1
0246 EB                XCHG                 ;SAVE THE LAST ADDRESS..
0247 220501            SHLD DESAVE          ;..OF DIRBUFF USED.
024A EB                XCHG
024B D1                POP D
```

```
024C 3A0401          LDA DIRCNT             ;COUNT THE NUMBER OF..
024F 3C              INR A                  ;..DIRECTORY ENTRIES.
0250 320401          STA DIRCNT
0253 F1              POP PSW
0254 C9              RET

MOD4

  ;CALCULATE THE ADDRESS OF THE DIRECTORY FCB IN DMA

0255 E603            ANI 03H                ;GET DIR ENTRY MODULO 4
0257 070707          RLC ! RLC ! RLC        ;MULT BY 32
025A 0707            RLC ! RLC
025C 218000          LXI H,80H              ;GET ADDR IN DMA (80H)
025F 85              ADD L
0260 6F              MOV L,A
0261 C9              RET

EXTSCH

  ;SEARCH FOR ALL FILES WITH EXTENT 1

0262 211601          LXI H,ANYNAME + 12     ;INCREASE EXTENT NUMBER
0265 34              INR M
0266 0E11            MVI C,SEARCH
0268 110A01          LXI D,ANYNAME
026B CD0500          CALL BDOS
026E FEFF            CPI 0FFH               ;IF THERE IS NOT A FIRST EXTENT..
0270 C8              RZ                     ;..THEN DONE WITH SEARCHES.

0271 47              MOV B,A                ;CODE TO NEXT BLANK LINE..
0272 3A1601          LDA ANYNAME+12         ;..ADDED FOR CP/M 2 BY..
0275 E601            ANI 1                  ;..HARVEY FISHMAN.
0277 78              MOV A,B
0278 CA8802          JZ LOOP5
027B CD5502          CALL MOD4
027E C60C            ADI 12
0280 6F              MOV L,A
0281 7E              MOV A,M
0282 E601            ANI 1
0284 CAA802          JZ NXT1
0287 78              MOV A,B                ;END OF CODE BY H.F.

0288 CD5502   LOOP5  CALL MOD4              ;GET ADDR IN HL
028B EB              XCHG                   ;PUT IT IN DE.
028C 219C05          LXI H,DIRBUFF
028F CDCF02   LOOP4  CALL MATCH             ;COMPARES EXTENTS WITH 0TH EXT.
0292 DA9C02          JC FOUNDMATCH          ;ITS ADDRESS IS IN HL REG
0295 011000          LXI B,16
0298 09              DAD B                  ;TEST THE NEXT ADDRESS IN DIRBUFF
0299 C38F02          JMP LOOP4

FOUNDMATCH

029C EB              XCHG                   ;PUT DMA ADDRESS IN HL REG
029D 010F00          LXI B,15
02A0 09              DAD B
02A1 EB              XCHG                   ;DE POINTS TO # RECS IN EXT ENTRY
02A2 09              DAD B                  ;HL POINTS TO # RECORDS IN DIRBUFF
02A3 1A              LDAX D                 ;GET # RECORDS IN EXTENT
02A4 77              MOV M,A                ;PUT IT IN # RECORDS OF DIRBUFF
02A5 2B              DCX H                  ;POINT HL TO A BYTE TO BE..
02A6 2B              DCX H                  ;..DESIGNATED AS NUMBER OF EXTS.
02A7 34              INR M                  ;COUNT NUMBER OF EXTENTS
02A8 0E12     NXT1   MVI C,NXTSCH           ;SEARCH FOR NEXT FILE
02AA 110A01          LXI D,ANYNAME
```

```
02AD CD0500          CALL BDOS
02B0 FEFF            CPI 0FFH

02B2 CA6202          JZ EXTSCH              ;CODE TO NEXT BLANK LINE ADDED..
02B5 47              MOV B,A                ;..FOR CP/M 2 BY HARVEY FISHMAN.
02B6 3A1601          LDA ANYNAME+12
02B9 E601            ANI 1
02BB 78              MOV A,B
02BC CA8802          JZ LOOP5
02BF CD5502          CALL MOD4
02C2 C60C            ADI 12
02C4 6F              MOV L,A
02C5 7E              MOV A,M
02C6 E601            ANI 1
02C8 CAA802          JZ NXT1
02CB 78              MOV A,B
02CC C38802          JMP LOOP5                      ;END OF CODE BY H.F.

MATCH

  ;CALLED WITH DE POINTING TO FILENAME IN FCB AND HL POINTING
  ;TO FILE NAME IN DIRBUFF. RETURNS WITH CARRY SET IF THE
  ;NAMES ARE THE SAME.

02CF E5              PUSH H
02D0 D5              PUSH D
02D1 0E0B            MVI C,11
02D3 1A       LOOP3  LDAX D
02D4 BE              CMP M
02D5 C2E202          JNZ CLRCRY             ;THEY'RE NOT EQUAL. CLEAR CARRY
02D8 23              INX H                  ;CHECK NEXT CHARACTERS
02D9 13              INX D
02DA 0D              DCR C
02DB C2D302          JNZ LOOP3              ;CHECK 11 CHARACTERS.
02DE D1              POP D                  ;THEY'RE EQUAL
02DF E1              POP H
02E0 37              STC
02E1 C9              RET
02E2 B7       CLRCRY ORA A                  ;CLEAR CARRY
02E3 D1              POP D
02E4 E1              POP H
02E5 C9              RET

RECINDEC

  ;GETS DECIMAL NUMBER OF PAGES (256 BYTES) IN FILE EXTENTS.

02E6 3A0401          LDA DIRCNT
02E9 47              MOV B,A                ;NUMBER OF DIRECTORY ENTRIES IN B
02EA 21AB05          LXI H,DIRBUFF + 15     ;POINT TO NUMBER OF RECORDS..
02ED 7E       LOOP8  MOV A,M                ;..IN FIRST DIRBUFF ENTRY.
02EE 3C              INR A                  ;INCREASE BY 1, THEN DIVIDE BY 2..
02EF B7              ORA A                  ;..TO CHANGE RECORDS TO PAGES.
02F0 1F              RAR
02F1 CD2B03          CALL DECIMAL           ;CHANGE BINARY TO DECIMAL
02F4 2B              DCX H                  ;POINT TO NUMBER OF..
02F5 2B              DCX H                  ;..EXTENTS PER FILE.
02F6 7E              MOV A,M
02F7 CD3B03          CALL ADDEXT            ;RETURNS WITH UNITS IN E..
02FA 2B              DCX H                  ;..TENS IN D, HUNDREDS IN C.
02FB 79              MOV A,C
02FC FE00            CPI 0                  ;BLANK IF 100'S IS ZERO
02FE C21503          JNZ SKIP4              ;MAKE IT A DIGIT
0301 F620            ORI 20H                ;MAKE IT A BLANK
0303 77              MOV M,A                ;PUT IT BACK IN DIRBUFF
0304 23              INX H                  ;IF HUNDRED'S ARE BLANK..
```

```
0305 7A                MOV A,D                ;..TEST TO SEE IF TEN'S..
0306 FE00              CPI 0                  ;..SHOULD BE BLANK.
0308 C21003            JNZ SKIP9
030B F620              ORI 20H
030D C31C03            JMP SKIP10
0310 F630      SKIP9   ORI 30H
0312 C31C03            JMP SKIP10
0315 F630      SKIP4   ORI 30H                ;MAKE IT ASCII
0317 77        SKIP5   MOV M,A
0318 23                INX H
0319 7A                MOV A,D                ;TEN'S DIGIT
031A F630              ORI 30H                ;MAKE IT ASCII
031C 77        SKIP10  MOV M,A
031D 23                INX H
031E 7B                MOV A,E
031F F630              ORI 30H
0321 77                MOV M,A
0322 111100            LXI D,17               ;READY FOR NEXT ENTRY..
0325 19                DAD D                  ;..IN DIRBUFF.
0326 05                DCR B
0327 C2ED02            JNZ LOOP8
032A C9                RET
```

DECIMAL

```
;CHANGES A-REG TO DECIMAL. RETURNS WITH TEN'S IN D,
;UNIT'S IN E.

032B 110000            LXI D,0
032E F5                PUSH PSW
032F D60A      LOOP9   SUI 10
0331 14                INR D                  ;COUNT NUMBER OF TENS
0332 D22F03            JNC LOOP9
0335 15                DCR D
0336 C60A              ADI 10
0338 5F                MOV E,A                ;UNIT'S DIGIT IN E
0339 F1                POP PSW
033A C9                RET
```

ADDEXT

```
;FOR EVERY EXTENT, ADD 64 DECIMAL TO NUMBER OF PAGES.
;HUNDRED'S IN C, TEN'S IN D, AND UNIT'S IN E.

033B 0E00              MVI C,0
033D FE00              CPI 0                  ;IF NO EXTENTS, EXIT
033F C8                RZ
0340 F5                PUSH PSW               ;SAVE NUMBER OF EXTENTS
0341 7A                MOV A,D                ;PUT TEN'S IN A(REG)
0342 0707              RLC ! RLC              ;GET IT IN UPPER NIBBLE
0344 0707              RLC ! RLC
0346 E6F0              ANI 0F0H               ;BLANK OUT LOWER NIBBLE
0348 B3                ORA E                  ;GET UNIT'S IN LOWER NIBBLE
0349 C664              ADI 64H                ;EACH EXTENT IS 64 PAGES
034B 27                DAA
034C D25003            JNC SKIP2
034F 0C                INR C                  ;COUNT HUNDREDS PLACE
0350 F5        SKIP2   PUSH PSW
0351 E60F              ANI 0FH                ;BLANK OUT TEN'S
0353 5F                MOV E,A                ;MOVE UNIT'S TO E
0354 F1                POP PSW
0355 0F0F              RRC ! RRC
0357 0F0F              RRC ! RRC              ;PUT TEN'S IN LOWER NIBBLE
0359 E60F              ANI 0FH                ;BLANK OUT UPPER NIBBLE
035B 57                MOV D,A                ;PUT TEN'S IN D
035C F1                POP PSW                ;GET BACK NUMBER OF..
035D 3D                DCR A                  ;..EXTENTS AND DECREASE.
035E C33D03            JMP ADDEXT + 2
```

PRINTOUT

```
;WRITES DIRBUFF TO 80 X 24 SCREEN. WRITES IN 3 COLUMNS. IF
;64 ENTRIES, WRITES 64TH AT BOTTOM OF THIRD COLUMN. WRITES
;THE 1ST, 22ND, 43RD RECORD, THEN THE 2ND, 23RD, 44TH RECORD,
;ETC. IF THE RECORD HAS BEEN BLANKED IN DIRBUFF BECAUSE IT
;HAS ALREADY BEEN WRITTEN OR THERE ARE NO MORE RECORDS, IT
;DOES NOT GET WRITTEN.

0361 3A0401            LDA DIRCNT
0364 320901            STA WRTNUM
0367 CD2404            CALL CRLF
036A 3E00              MVI A,0
036C 321605    LOOP6   STA RECNO
036F CDB103            CALL WRITENAME
0372 C615              ADI NOLINES
0374 CDB103            CALL WRITENAME
0377 C615              ADI NOLINES
0379 CDB103            CALL WRITENAME
037C 3A0301            LDA PRNTCNT
037F D603              SUI 3
0381 320301            STA PRNTCNT
0384 FE00              CPI 0                  ;CHECK FOR 64TH ENTRY
0386 CA9803            JZ PRINTEND
0389 3A0901            LDA WRTNUM             ;IF ALL ENTRIES ARE..
038C B7                ORA A                  ;..PRINTED, RETURN.
038D C8                RZ
038E CD2404            CALL CRLF
0391 3A1605            LDA RECNO
0394 3C                INR A
0395 C36C03            JMP LOOP6
```

PRINTEND   ;MUST GET THE 64TH ENTRY..

```
0398 3A0401            LDA DIRCNT             ;..IF THERE IS ONE.
039B FE40              CPI 64
039D C0                RNZ
039E 113A04            LXI D,TAB7             ;TAB TO LAST COLUMN FOR LAST ENTRY
03A1 0E09              MVI C,WRTBUF
03A3 CD0500            CALL BDOS
03A6 3E00              MVI A,0                ;STIFFLE ANYMORE TABS
03A8 322204            STA NULLIT
03AB 3E3F              MVI A,63
03AD CDB103            CALL WRITENAME
03B0 C9                RET
```

WRITENAME

```
;CHANGES "RECNO" TO (DIRBUFF+1) + 16*RECON. IT THEN STORES
;THAT ADDRESS IN HL AND PRINTS THE NAME AT THAT ADDRESS
;UNLESS IT IS BLANK.

03B1 C5                PUSH B
03B2 019D05            LXI B,DIRBUFF+1
03B5 6F                MOV L,A
03B6 2600              MVI H,0
03B8 2929              DAD H ! DAD H
03BA 2929              DAD H ! DAD H
03BC 09                DAD B

03BD F5                PUSH PSW
03BE 7E                MOV A,M                ;TEST TO SEE IF NAME WAS..
03BF FE20              CPI ' '                ;..PREVIOUSLY OVERWRITTEN.
```

```
03C1 CAED03          JZ NOWRITE
03C4 3A0901          LDA WRTNUM
03C7 3D              DCR A
03C8 320901          STA WRTNUM
03CB 0E08            MVI C,8              ;TRANSFER NAME TO OUTPUT BUFFER
03CD 110C04          LXI D,OUTBUFF
03D0 CDF003          CALL TRANS2
03D3 3E20            MVI A,' '            ;AT LEAST ONE SPACE BETWEEN..
03D5 12              STAX D               ;..NAME AND TYPE.
03D6 13              INX D
03D7 0E03            MVI C,3              ;TRANSFER TYPE
03D9 CDF003          CALL TRANS2
03DC 3E20            MVI A,' '
03DE 12              STAX D
03DF 13              INX D
03E0 12              STAX D
03E1 13              INX D
03E2 0E03            MVI C,3
03E4 CDF003          CALL TRANS2
03E7 110C04          LXI D,OUTBUFF
03EA CDFB03          CALL WRITOUT
03ED F1     NOWRITE  POP PSW
03EE C1              POP B
03EF C9              RET

03F0 7E     TRANS2   MOV A,M              ;TRANSFER ROUTINE TO OUTPUT BUFFER
03F1 12              STAX D
03F2 3620            MVI M,' '            ;OVERWRITE NAME WITH BLANKS
03F4 23              INX H
03F5 13              INX D
03F6 0D              DCR C
03F7 C2F003          JNZ TRANS2
03FA C9              RET

03FB 1A     WRITOUT  LDAX D               ;WRITES OUT EACH CHAR..
03FC B7              ORA A                ;..INDIVIDUALLY SINCE USING..
03FD C8              RZ                   ;..CP/M'S PRINT STRING..
03FE C5              PUSH B               ;..ROUTINE WILL NOT PRINT $'S..
03FF D5              PUSH D               ;..WHICH ARE SOMETIMES FILETYPES.
0400 0E02            MVI C,CONOUT
0402 5F              MOV E,A
0403 CD0500          CALL BDOS
0406 D1              POP D
0407 C1              POP B
0408 13              INX D
0409 C3FB03          JMP WRITOUT

040C        OUTBUFF  DS 17
041D 20207C20        DB '  |  '           ;SEPARATER OF FILE NAMES..
0422 0900   NULLIT   DB 9,0               ;..FOR EACH COLUMN.

0424 F5     CRLF     PUSH PSW
0425 D5              PUSH D
0426 C5              PUSH B
0427 E5              PUSH H
0428 113704          LXI D,CARLFD
042B 0E09            MVI C,WRTBUF
042D CD0500          CALL BDOS
0430 E1              POP H
0431 C1              POP B
0432 D1              POP D
0433 F1              POP PSW
0434 0603            MVI B,3
0436 C9              RET

0437 0D0A24 CARLFD   DB 0DH,0AH,'$'
```

```
043A 090909   TAB7   DB 9,9,9,9,9,9,9,'$'
```

SORT

```
 ;THE FOLLOWING IS AN ADAPTION OF THE SHELL-METZNER SORT
 ;TAKEN FROM A BASIC PROGRAM.

0442 3A0401          LDA DIRCNT           ;N = M = NUMBER OF ITEMS
0445 321605          STA RECNO
0448 321705          STA HALFREC
044B 3A1705  HALVE   LDA HALFREC          ;M = INT(M/2)
044E B7              ORA A
044F 1F              RAR
0450 321705          STA HALFREC
0453 C8              RZ                   ;M = 0 ?   YES - EXIT SORT
0454 3A1705          LDA HALFREC          ;K = N - M
0457 47              MOV B,A
0458 3A1605          LDA RECNO
045B 90              SUB B
045C 321805          STA SPREAD
045F 3E00            MVI A,0              ;J = 0
0461 321B05          STA J
0464 3A1B05  X2      LDA J                ;I = J
0467 321905          STA FIRSTREC
046A 3A1905  X1      LDA FIRSTREC         ;L = I + M
046D 47              MOV B,A
046E 3A1705          LDA HALFREC
0471 80              ADD B
0472 321A05          STA SECONDREC
0475 CDA404          CALL CHTOAD          ;CHANGE RECORD TO ADDRESS IN DIRBUFF
0478 CDBF04          CALL COMPARE         ;D(I) > D(J) ?
047B D29204          JNC SKIPTRANS        ;IF NO CARRY SET, THEN DO NOT SWITCH
047E CDD904          CALL SWITCHHD        ;ELSE MAKE THE SWITCH
0481 3A1705          LDA HALFREC          ;I = I - M
0484 47              MOV B,A
0485 3A1905          LDA FIRSTREC
0488 90              SUB B
0489 321905          STA FIRSTREC
048C FA9204          JM SKIPTRANS         ;IF I < 0 THEN LOOP BACK
048F C36A04          JMP X1
```

SKIPTRANS

```
0492 3A1B05          LDA J                ;J = J + 1
0495 3C              INR A
0496 321B05          STA J
0499 47              MOV B,A              ;IS J > K
049A 3A1805          LDA SPREAD
049D B8              CMP B
049E DA4B04          JC HALVE             ;IF J > K GOTO 'HALVE'
04A1 C36404          JMP X2               ;IF J <= K GOTO 'X2'
```

CHTOAD

```
 ;CHANGES "RECNO" TO CORRECT ADDRESS IN DIRBUFF. HL POINTS
 ;TO FIRST RECORD AND DE POINTS TO SECOND.

04A4 3A1905          LDA FIRSTREC
04A7 CDB304          CALL ADJUST
04AA EB              XCHG
04AB 3A1A05          LDA SECONDREC
04AE CDB304          CALL ADJUST
04B1 EB              XCHG
04B2 C9              RET
```

ADJUST

```
   ;MULTIPLIES "RECNO" BY 16 AND PUTS IN HL TO POINT
   ;TO NAME IN DIRBUFF.

 04B3 019C05             LXI B,DIRBUFF
 04B6 6F                 MOV L,A
 04B7 2600               MVI H,0
 04B9 2929               DAD H ! DAD H
 04BB 2929               DAD H ! DAD H
 04BD 09                 DAD B
 04BE C9                 RET
```

COMPARE

```
   ;COMPARES THE NAMES IN THE FIRST AND SECOND ADDRESS. IF THE
   ;FIRST IS LARGER THAN THE SECOND, IT INDICATES A SWITCH
   ;SHOULD BE MADE BY SETTING THE CARRY.

 04BF E5                 PUSH H
 04C0 D5                 PUSH D
 04C1 0E0B               MVI C,11
 04C3 46                 MOV B,M
 04C4 1A                 LDAX D
 04C5 B8                 CMP B
 04C6 DACD04             JC RETWC
 04C9 CAD004             JZ INCREASE
 04CC B7                 ORA A                  ;CLEAR CARRY
 04CD D1         RETWC   POP D
 04CE E1                 POP H
 04CF C9                 RET
```

INCREASE ;CHARACTERS WERE EQUAL. DO..

```
 04D0 23                 INX H                  ;..ANOTHER COMPARE.
 04D1 13                 INX D
 04D2 0D                 DCR C
 04D3 CACB04             JZ RETWC-2
 04D6 C3C304             JMP COMPARE+4
```

SWITCHHD

```
   ;THIS ROUTINE SWITCHES THE FIRST RECORD WITH THE SECOND.

 04D9 D5                 PUSH D
 04DA E5                 PUSH H
 04DB 110605             LXI D,TEMP             ;PUT THE SECOND RECORD IN..
 04DE 0610               MVI B,16               ;.. TEMPORARY STORAGE.
 04E0 CDF904             CALL HTD
 04E3 E1                 POP H                  ;PUT THE FIRST RECORD..
 04E4 D1                 POP D                  ;..IN THE SECOND.
 04E5 EB                 XCHG
 04E6 0610               MVI B,16
 04E8 CDF904             CALL HTD
 04EB EB                 XCHG                   ;PUT THE TEMPORARY (SECOND)..
 04EC E5                 PUSH H                 ;..IN THE FIRST.
 04ED D5                 PUSH D
 04EE 210605             LXI H,TEMP
 04F1 0610               MVI B,16
 04F3 CDF904             CALL HTD
 04F6 D1                 POP D
 04F7 E1                 POP H
 04F8 C9                 RET
```

HTD

```
   ;THIS ROUTINE DOES THE TRANSFER. MOVES RECORD ADDRESSED BY
   ;HL TO RECORD ADDRESSED BYE DE.

 04F9 D5                 PUSH D
 04FA E5                 PUSH H
 04FB 7E                 MOV A,M
 04FC 12                 STAX D
 04FD 23                 INX H
 04FE 13                 INX D
 04FF 05                 DCR B
 0500 C2FB04             JNZ HTD+2
 0503 E1                 POP H
 0504 D1                 POP D
 0505 C9                 RET

 0506            TEMP    DS 16
 0516            RECNO   DS 1
 0517            HALFREC DS 1
 0518            SPREAD  DS 1
 0519            FIRSTREC DS 1
 051A            SECONDREC DS 1
 051B            J       DS 1

 051C            STACK   DS 80H
 059B =          NEWSTK  EQU $-1

 059C            DIRBUFF DS 16*64

 099C                    END
```

An S-100 Eprom Programmer Using the Intel 8255 PPI

by Ted Croal

The 8255 programmable peripheral interface is a convenient interface for an EPROM programmer.

For my first scratch-built computer project to add to my 8080A system, I selected the EPROM programmer circuit provided by Steve Ciarcia in the March 1978 issue of "BYTE". I was reading Adam Osborne's "An Introduction to Microcomputers, Vol. II" at the time and I thought that the 8255PPI would be a good way to provide the four parallel I/O ports required for the project. When the board was about half completed, I realized that I did not have specific instructions for connecting to the S-100 bus. After some trial and error, I arrived at a configuration that worked well for programming but would not read data in from the 2708 EPROM. It was not until I read David Condra's article in the October 1979 issue of "BYTE", that I realized that I did not properly isolate the data lines of the 8255 from the data in lines of the S-100 bus. I added tristate buffers to the data outlines and now the board is working as intended.

The 8255PPI is a 24 pin general purpose interface that can be programmed in a variety of ways. In my board it is used in the simplest mode (Mode O, for basic I/O). Included in the device are three data ports, referred to as A, B, and C and a fourth port which is used for programming the 8255. By writing into this control port (the highest of the four consecutive port addresses used by the 8255), ports A, B, and C can be assigned to either input or output. Individual pins of port C can also be set or reset by appropriate control bytes.

Programmer Hardware Structure

I use port A for data to be loaded into the 2708 EPROM, or read from it, port B for the eight lowest address lines of the EPROM, and the two lowest bits of port C for the two highest address lines of the EPROM (page address). I use bit 2 of port C for programming mode (set for programming, reset for reading) and bit 3 for the programming pulse, leaving bits 4-7 of port C available for other uses.

Ted Croal, 40 Lorne Cresent, Brantford, ONT, Canada, N5

Port assignments are made by writing a control byte in which bit 7 is set. Bits 6 and 5 are reset to indicate mode O for port A and bit 2 is reset to indicate mode O for port B. Bits 4, 3, 1, and O control the direction of port A, the upper half of port C, port B, and the lower half of port C respectively. They are set for input and reset for output.

Port C can be used in the usual manner by writing a byte to the port address or the bits of port C can be controlled individually by writing to the control port a byte in which bit 7 is reset. Bits 3, 2, and 1 of this byte form a binary number specifying the bit number of port C to be controlled and bit O specifies the state (reset for O, set for 1). Bits 6, 5, and 4 of the control byte are ignored.

The control bytes used in the listings are:

***80H (200Q):** Ports A, B and C are defined as out ports and mode O is selected. This is in preparation for programming.

***90H (220Q):** Port A is defined as an in port and B and C as out ports. Mode O is selected. This prepares for reading the EPROM.

***9BH (233Q):** Ports A, B and C are assigned as in ports. This is intended to discourage a false pulse from reaching the EPROM when the power is turned off, but is not always effective. The 27V supply should be turned on after the main power supply and off before the main power supply.

***07H (007Q):** This sets bit 3 port C, initiating a programming pulse.

The following control bytes are useful in checking the operation of the board from the front panel or in a test program. (Port C must already be defined as an out port.)

***05H (005Q):** bit 2 of port C is set and the LED comes on to indicate "program mode".

***04H (004Q) :** Bit 2 of port C is reset and the LED goes off to indicate "read mode".

The "program mode" LED also comes on when the 8255 is reset, as at power on, or when port C is defined as an in port.

OPINION

The Shadow, Buck Rogers, and the Home Computer — *Gardner* 2
The State of the Art — *Helmers* 5
Could a Computer Take Over — *Rush* 8

THEORY AND TECHNOLOGY

A Systems Approach to a Personal Microprocessor — *Suding* 14
Frankenstein Emulation — *Murray* 17
Programming for the Beginner — *Herman* 22
What is a Character — *Peshka* 27
Friends, Humans, and Countryrobots: Lend me your Ears — *Rice* 36
Magnetic Recording for Computers — *Manly* 44

COMPUTER KITS

Assembling an Altair 8800 — *Zarrella* 56
Build a 6800 System With This Kit — *Kay* 59
More on the SWTPC 6800 System — *Kay* 64
The New Altair 680 — *Vice* 68
A Date With KIM — *Simpson* 72
True Confessions: How I Relate to KIM — *Gupta* 76
Zilog Z80 — *Hashizume* 81
The Digital Equipment LSI-11 — *Baker* 86
Cromemco TV Dazzler 94

HARDWARE

Flip Flops Exposed — *Browning* 98
Recycling Used ICs — *Mikkelsen* 102
Powerless IC Test Clip — *Errico and Baker* 104
Parallel Output Interfaces in Memory Address Space — *Helmers* 106
Son of Motorola — *Fylstra* 110
Data Paths — *Liming* 117
Build a TTL Pulse Catcher — *Walde* 124
Dressing Up Front Panels — *Walters* 125
Deciphering Mystery Keyboards — *Helmers* 126
A Quick Test of Keyboards — *Walters* 134
Keyboard Modification — *Macomber* 135
Serialize Those Bits From Your Mystery Keyboard — *Halber* 136
Build a Television Display — *Gantt* 138
The "Ignorance Is Bliss" Television Drive Circuit — *Barbier* 144
Build a TV Readout Device for Your Microprocessor — *Suding* 145
Let There Be Light Pens — *Loomis* 153
Build an Oscilloscope Graphics Interface — *Hogenson* 158
An Introduction to Addressing Methods — *Zarrella* 169
Interface an ASCII Keyboard to a 60mA TTY Loop — *Cotton* 174
Interfacing the 60 mA Current Loop — *King* 175
The Complete Tape Cassette Interface — *Hemenway* 177
Digital Data on Cassette Recorders — *Mauch* 184
Build a Fast Cassette Interface — *Suding* 190
Technology Update 197
What's In a Video Display Terminal? — *Walters* 198
Pot Position Digitizing Idea — *Schulein* 199
Read Only Memories in Microcomputer Memory Address Space — *Eichbauer* 200
More Information on PROMs — *Smith* 203
Getting Input from Joysticks and Slide Pots — *Helmers* 210
Logic Probes — Hardware Bug Chasers — *Burr* 213
Controlling External Devices With Hobbyist Computers — *Bosen* 218
Microprocessor Based Analog/Digital Conversion — *Frank* 222
Add a Kluge Harp to Your Computer — *Helmers* 226
The Time Has Come to Talk — *Atmar* 231
Make Your Own Printed Circuits — *Hogenson* 238

SOFTWARE

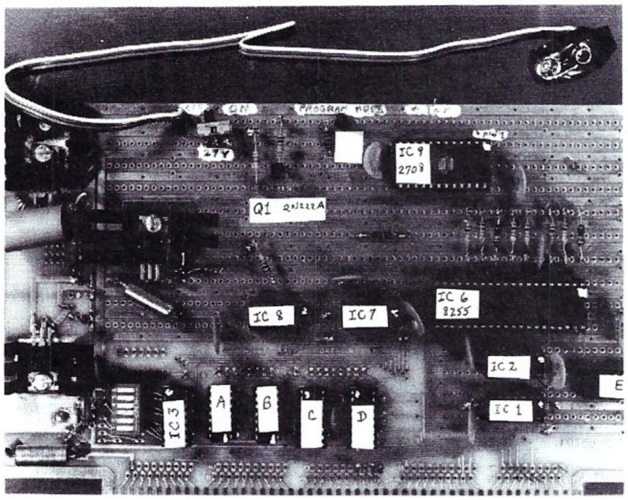
Write Your Own Assembler — *Fylstra* 246
Simplify Your Homemade Assembler — *Jewell* 255
Interact With an ELM — *Gable* 261
Design an On Line Debugger — *Wier and Brown* 268
Processing Algebraic Expressions — *Maurer* 275
The "My Dear Aunt Sally Algorithm" — *Grappel* 286
Can YOUR Computer Tell Time? — *Hogenson* 294
A Plot Is Incomplete Without Characters — *Lerseth* 300
Hexpawn: A Beginning Project in Artificial Intelligence — *Wier* 309
Shooting Stars — *Nico* 314
Biorythm for Computers — *Fox* 322
Life Line — *Helmers* 326

APPLICATIONS

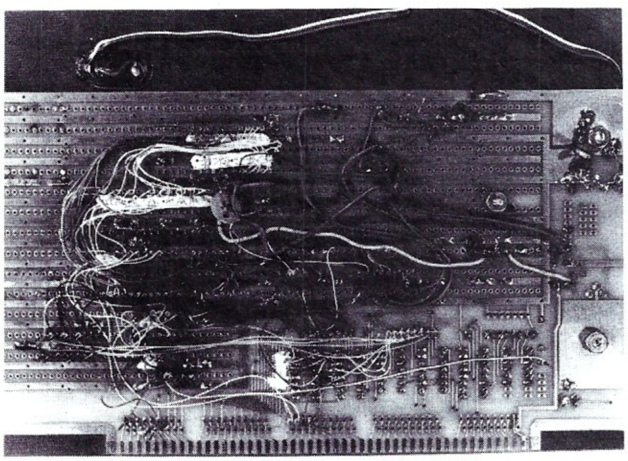
Total Kitchen Information System — *Lau* 360
A Small Business Accounting System — *Lehman* 364
Chips Found Floating Down Silicon Slough — *Trumbull* 369

RESOURCES

Books of Interest 372
Magazines 375



Component side of board.
This board does not correspond to the schematic because it incorporates addressing decoding and data bus buffering. The data out lines are buffered with 2 input AND gates (74LS08, IC's A and B0. This proved to be unsuitable for proper operation of the internal bidirectional data bus. Adding an additional tri-state buffer (IC E) to the data out lines and employing the unused gates of IC C and IC D (Tri-state hex buffers on the data in lines) corrected the problem. A, B, C, D, and E serve the same function as IC's 4 and 5 in the schematic. It would be simpler to build on a completely bare board, following the schematic which can be found at the end of this article.



Wire side of board.
The heavy wiring is for the extra power supply connections required by the 2708 EPROM. The +5V supply is to pin 24 of the EPROM. The +12V supply is pin 19 and the −5V supply is pin 21. Pin 12 is the ground. Some of the bypass capacitors have been placed on this side.

Number	Type	+5V	GND
IC 1	74LS04	14	7
IC 2	74LS00	14	7
IC 3	DM8131	16	8
IC 4	DS8833	16	8
IC 5	DS8833	16	8
IC 6	8255PPI	26	7
IC 7	74121	14	7
IC 8	7406	14	7
IC 9	2708	24	12

(IC 9 also requires +12V (pin 19) and -5V (pin 21)

Power Supply Connections

Programmer Software

There are two subroutines, BURN (listing 1) and READ (listing 2), which address the EPROM. The subroutine CYCLE (listing 3) calls on BURN to program the block of memory specified in three memory words into the EPROM, starting at the address in the EPROM specified by a fourth memory word.

```
BURN:    MOV   A,D      ; High byte of EPROM adr
         ANI   03H      ; Mask all but bits 0,1
         ORI   04H      ; Set bit 2 for program mode
         OUT   PORTC    ; Two highest adr lines of EPROM
         MOV   A,E      ; Low byte of EPROM adr
         OUT   PORTB    ; Remaining EPROM adr lines
         MOV   A,M      ; Data byte
         OUT   PORTA    ; Output to data lines
         MVI   A,07H    ; Control byte to set bit 3 port C
         OUT   PORTD    ; Write strobe
         MVI   B,85H    ; Delay count
DELAY:   DCR   B        ; One millisecond delay
         JNZ   DELAY    ; during write strobe
         XRA   A        ; Clear accumulator
         OUT   PORTC    ; Clear port C
         RET            ; Return from BURN
```

Listing 1

Subroutine to program a byte of memory specified by register pair HL at an address in EPROM specified by register pair DE. "PORT A" is the lowest of the four consecutive port addresses used by the 8255PPI. "Port B" and "Port C" are the next two and "PORT D" is the highest address, that is, the address of the control port. Ports A, B, and C must first be defined as out ports.

```
READ:    MOV   A,D      ; High byte of EPROM adr
         ANI   03H      ; Mask all but bits 0,1
         OUT   PORTC    ; Two highest adr lines of EPROM
         MOV   A,E      ; Low byte of EPROM adr
         OUT   PORTB    ; Remaining EPROM adr lines
         IN    PORTA    ; Data byte
         RET            ; Return from READ
```

Listing 2

Subroutine to read a byte from the 2708 EPROM. Port A must first be defined as an in port, and ports B and C as out ports.

```
CYCLE:   MVI   A,80H    ; Code byte to define A, B, C
         OUT   PORTD    ; as out ports
         LHLD  STDES    ; Starting adr in EPROM
         XCHG           ; Place in DE
         LHLD  STSRC    ; Starting adr of source block in HL
WLOOP:   CALL  BURN     ; Program a byte of EPROM
         INX   D        ; Increment EPROM pointer
         INX   H        ; Increment source pointer
         LDA   ENSRC    ; Adr of end of block
         CMP   L        ; Test low byte source pointer
         JNZ   WLOOP    ; Loop until same
         LDA   ENSRC+1  ; Get high byte of end adr
         CMP   H        ; Test high byte source pointer
         JNZ   WLOOP    ; Loop until same
         RET            ; Return from CYCLE
```

Listing 3

Subroutine to program a block of memory into a given location in a 2708 EPROM. Each specified location in EPROM is given one programming pulse.

I have a routine (not listed) common to several programs that loads these memory locations. STSRC is the starting address of the source, that is, the starting address of the program to be loaded into EPROM. LNSRC is the length of the block and ENSRC is the first address past the end of the block (or STSRC + LNSRC). STDES is the address of the start of the block in the EPROM. Bits 7-2 of the high order byte of STDES are ignored so the address of the intended site for the EPROM can be used.

Before each programming run, the subroutine TSTER (listing 4) is called to see if the block to receive the program is blank. The status of the EPROM block is displayed. The message "error" indicates that at least

EPROM Programmer cont'd...

one bit of the block is reset. The operator may then abort the run or proceed to load over the block. (A set bit may be reset but a reset bit must be erased by ultra-violet light.)

```
TSTER:      MVI     A,90H       ; Code byte to define A as in, and
            OUT     PORTD       ; B and C as out ports
            LHLD    STDES       ; Starting adr in EPROM
            XCHG                ; Place in DE
            LHLD    LNSRC       ; Length of block in HL
TLOOP:      CALL    READ        ; Read a byte of EPROM
            INR     A           ; If byte erased, reg A is 0 after
            RNZ                 ; increment, else return
            INX     D           ; Increment EPROM pointer
            DCX     H           ; Decrement block length
            MOV     A,L         ; Test for end of block
            ORA     H           ; Zero flag is set if HL is 0
            JNZ     TLOOP       ; Loop until done
            RET                 ; Return with zero flag set
```

Listing 4

Subroutine to test a block of EPROM for complete erasure. If the block is erased, the zero flag is set on return. Else it is reset.

A full load (listing 5) of one kilobyte is complete in less than two minutes, and a quick load (listing 6) requires only a few seconds, and usually consists of only 2 cycles. The quick-loaded EPROM seems to work just as well as the full-loaded one but you may wish to copy the program with a full load after you have debugged the program which you are loading.

```
FLLOD:      MVI     C,100       ; Place count of 100 (64H,144Q) in reg C
            CALL    CARST       ; Clear and reset TV screen
            LXI     H,MES7      ; Inform user that
            CALL    MESGE       ; loading is in progress
CLOOP:      CALL    CYCLE       ; Program the block once
            DCR     C           ; Decrement count
            JNZ     CLOOP       ; Loop until done
            MVI     C,100       ; Restore count for display later
            JMP     LDONE       ; See listing 6
```

Listing 5

Subroutine to make a full load of a block of memory, consisting of 100 programming pulses of 1 ms each to each byte in the block. Register C starts and ends with the number 100 to indicate the number of cycles used.

```
QKLOD:      MVI     C,OH        ; Clear register to receive count
            CALL    CARST       ; Clear and reset TV screen
            LXI     H,MES7      ; Inform user that
            CALL    MESGE       ; loading is in progress
AGAIN:      CALL    CYCLE       ; Program the block once
            INR     C           ; Increment count
            MVI     A,9H        ; Ten cycles is maximum
            CMP     C           ; Compare count to 9
            JC      LDONE       ; If exceded abort run
            CALL    VERFY       ; Compare load with source
            JNZ     AGAIN       ; If different, loop
            CALL    CYCLE       ; Else program once more
            INR     C           ; Increment count
LDONE:      CALL    CDISP       ; Display count
            CALL    VERFY       ; Verify the load
            JNZ     ERMSG       ; If incorrect display "error" message
            LXI     H,MES5      ; Else point to "verified" message
STTUS:      CALL    MESGE       ; Display message
            LXI     H,MES8      ; Point to "done" message
            CALL    MESGE       ; Display it
            MVI     A,9BH       ; Code byte to define A, B, C as in
            OUT     PORTD       ; to prevent false program pulse
            CALL    INPUT       ; Wait until key pressed (read display)
            JMP     UTILY       ; Go to utility directory
ERMSG:      LXI     H,MES6      ; Point to "error" message
            JMP     STTUS       ; and display
```

Listing 6

Subroutine to make a quick load of a block of memory. The 2708 is checked after each cycle to see if the loading is correct. One additional cycle is made and the count of the cycles is displayed and the load verified. The run is aborted after ten unseccessful tries (you probably forgot to turn on the programming power supply).

At the end of the run, the EPROM is compared with the source block by the subroutine VERFY (listing 7) and the status displayed. The number of programming cycles used is also displayed.

```
VERFY:      MVI     A,90H       ; Code byte to define  A as in, and
            OUT     PORTD       ; B and C as out ports
            LHLD    STDES       ; Starting adr in EPROM
            XCHG                ; Place in DE
            LHLD    STSRC       ; Start of source in HL
RLOOP:      CALL    READ        ; Read a byte
            CMP     M           ; Compare with source
            RNZ                 ; Return if different, zero is reset
            INX     D           ; Increment EPROM pointer
            INX     H           ; Increment source pointer
            LDA     ENSRC       ; Adr of end of block
            CMP     L           ; Test low byte source pointer
            JNZ     RLOOP       ; Loop until same
            LDA     ENSRC+1     ; Get high byte of end adr
            CMP     H           ; Test high byte source pointer
            JNZ     RLOOP       ; Loop until same
            RET                 ; Return from VERFY
```

Listing 7

Subroutine to verify the program loaded into the 2708 EPROM with the soruce in memory. If a byte is found that does not match, the zero flag is reset on return. If the load is verified, the zero flag is set.

I use a "menu" display in which the user can select the desired program. At the end of the run, the computer returns to the "utility directory" for selection of another program. Listing 8 shows how the subroutines can be combined to form a useful program. CARST (not listed) clears and resets the TV screen. CDISP (not listed) displays the contents of register C. Other programs not listed in this article display the contents of the EPROM and copy the EPROM into memory. These call on READ and can be written in the same manner as the routines listed. Several messages are used to inform the user of the progress of the program (listing 9). These are displayed by the subroutine MESGE (not listed). The routine at MENU (not listed) displays the choices, receives a keyboard entry, and jumps to the address specified in the table at MENTB (not listed).

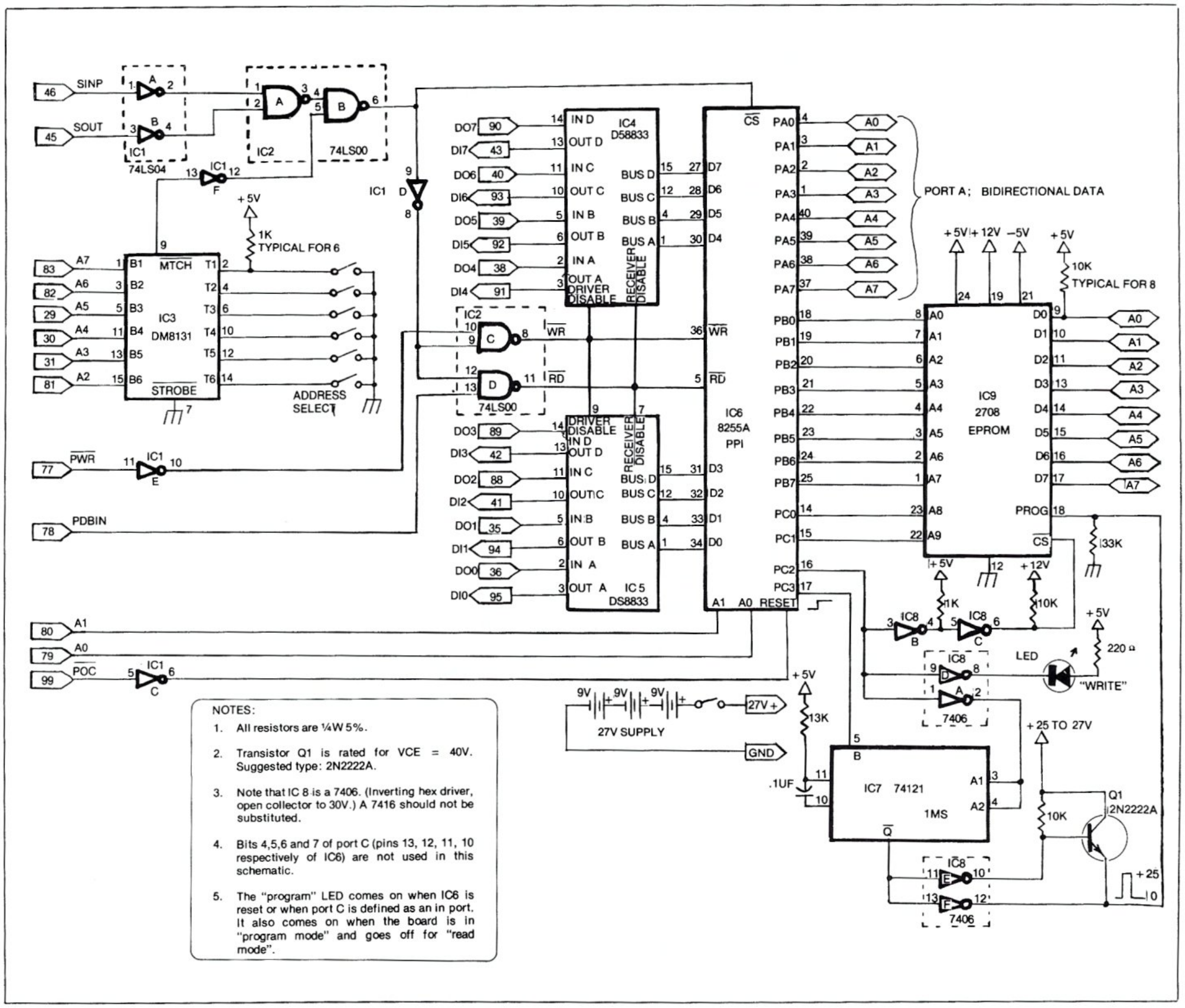
```
LODIT:      CALL    CARST       ; Clear and reset TV display
            CALL    TSTER       ; Test for erased block
            JNZ     NOTER       ; If not erased point to "not"
            LXI     H,MES3      ; Else point to "erased"
INFRM:      CALL    MESGE       ; Display message
CHUSE:      LXI     H,MENTB     ; Point to address of menu table
            JMP     MENU        ; Display menu and jump to choice
NOTER:      LXI     H,MES4      ; Point to "not erased"
            JMP     INFRM       ; and display
```

Listing 8

Program to select either a full load or a quick load for a block of memory. Choices are (1) for quick load, (2) for full load, or (ESC) for return to start of program (not listed) to redefine the block. All other entries cause a return to (CHUSE).

```
MES 1       1.  Quick load (used by MENU)
MES 2       2.  Full load (used by MENU)
MES 3       Block erased
MES 4       Block NOT erased
MES 5       Load verified
MES 6       Error!
MES 7       Loading in progress
MES 8       Done
```

Listing 9 Messages Used



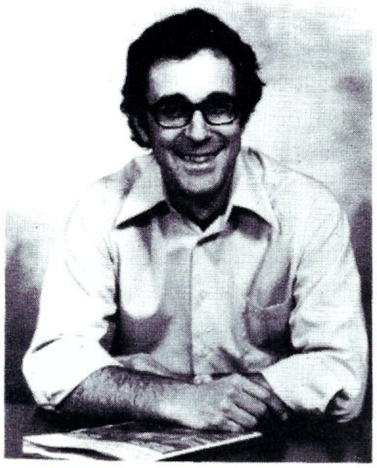
If you are building this circuit, note that IC4 in Steve Ciarcia's article should be a 7406 not a 7407 (do not substitute a 7416). Note also that in Dave Condra's article there should be an inverter between pin 9 of IC3 (DM8131) and pin 5 of IC2 (74LS00). Use a section of IC1 (74Ls04); connect pin 9 of IC3 to pin 13 of IC1 and pin 13 of IC1 to pin 5 of IC2. Although I did not use the DS8833 bus transceivers I would do so if I were building the circuit again.

References

1. Program Your Next EROM in BASIC; Ciarcia, Steve; BYTE March 1978, page 84.

2. Interfacing the S-100 Bus with the Intel 8255; Condra, David L.; BYTE October 1979, page 124.

3. An Introduction to Microcomputers Vol. II; Adam Osborne and Associates, page 4-133. □



David Ahl, Founder and Publisher of Creative Computing

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 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 accessible 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 your satisfaction or we will refund the unfulfiled portion 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

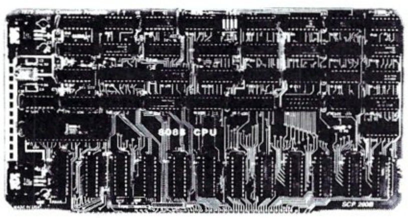
P.O. Box 789-M
Morristown, NJ 07960
Toll-free **800-631-8112**
(In NJ 201-540-0445)

NEW PRODUCTS

16-BIT 8086 CPU PRODUCTS

Seattle Computer Products has released several products which comprise an Intel 8086 based computer system. Included are the following:

16-BIT 8086 CPU CARD: The SPC200 is an S-100 8086 based CPU compatible with older 8-bit memories and newer 16-bit memories, or the two combined. It meets the IEEE S-100 bus standard. Addressing range is one megabyte. The price is $695 (4 MHz) ir $895 (8 MHz/4 MHz).



CPU SUPPORT CARD: The SPC 300 is a companion to the SPC 200 and is necessary for a multi-user system. It provides 2K ROM (contains system monitor), serial I/O (programmable band rate), form programmable times and more. The price is $395 (with ROM) or $370 (with disk instead of ROM).

MEMORY CARD: The SPC 107 is an IEEE compatible memory will automatically configures itself as 8K by 16 bits to the SPC 200 or 16K by 8 bits to 8-bit CPUs. It utilizes 200 nsec chips which can operate at 8 MHz with no wait states. Price: $455.

BASIC-86: Identical to Microsoft's version 5.0 of Basic. Does not require an operating system as all disk and I/O drivers are included. Available on 8" IBM disk with Cromemco 4FDC drivers. Price: $350 (SCP owers), $500 (others).

CROSS-ASSEMBLER: A Z80 program that rans under CP/M and creates 8086 codes. Includes a translator program which converts Z80 source code to 8086 source code. Available in 5" soft-sectored, 5" North Star and 8" IBM disk. Price: $250.

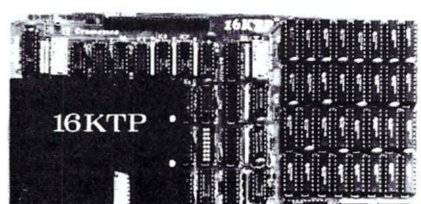
DISK OPERATING SYSTEM: The SCP 86-DOS is a complete operating system package. Provides a high level interface to I/O devices and disk system. Includes resident assembler, Z80 to 8086 source code translator, a utiliby to read CP/M files, a line editor and disk maintenance utilities. Price: $95 (SCP owers), $195 (others).

For more information contact: Seattle Computer Products, Inc., 1114 Industry Drive, Seattle WA 98188; (206) 575-1830.

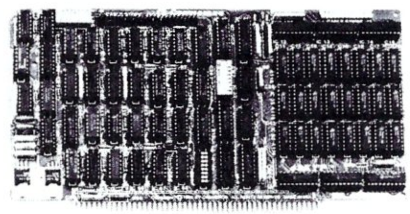
16K and 48K TWO PORT MEMORY BOARDS

Cromemco introduces two new two-port memory boards, the 16KTP and the 48KTP, for use with their Model SDI high resolution, color graphics interface.

These two-port memory boards have two sets of address and data lines which give them the ability to process the SDI's memory refresh requests while the CPU simultaneously and independently executes a user program. Picture information is accessible by the SDI through a connector on the top of the memory boards. This direct connection of the SDI and the two-port memory bypasses the S-100 bus so the CPU accesses the two-port memory as though the SDI were not present. Consequently, use of the special two-port memory in a graphics system assures 75% to 100% CPU utilization, depending on the application software.



These two-port memory boards are also designed to work with Cromemco's powerful graphics software package. This graphics package provides a full range of powerful, human oriented commands that operate from such common high-level languages as Basic, Fortran and Ratfor. The graphics software package will operate with one or two pages of two-port memory. Two pages of 48K bytes of RAM are required for complete utilization of all available software options.



Cromemco's two-board color graphics interface (Model SDI) is available for $595. The 16K two-port memory board (Model 16KTP) is available for $795 and the 48K two-port memory board (Model 48KTP) is available for $1785. Cromemco's graphics software package is available on either 8" (Model SGS-L) or 5" (Model SGS-S) floppy diskette for $295. For additional information, contact Cromemco, Inc., 280 Bernardo Avenue, Mountain View, CA (415) 964-7400.

HIGH RESOLUTION COLOR GRAPHICS FROM CROMEMCO

Cromemco has just announced a new software package for it's high resolution graphics system. The Cromemco graphics system can be used to display color or black-and-white images with up to 756 x 482 point resolution on a high quality RGB monitor.

The cromemco system provides commands that operate from such common high-level languages as Basic, Fortran and Ratfor.

The graphics software package is designed to work with Cromemco's 48KTP and 16KTP (two port) memory boards and will operate with one or two pages of two port memory. Two pages of 48K bytes of RAM are required for complete utilization of all available software options.

For those using the graphics software package, the subroutine calls provided are sufficient to fully utilize all the capabilities of the Cromemco SDI high resolution graphics interface board. These subroutines allow the user a number of powerful capabilities including: fast line generation; fast generation of regular shapes such as circles, rectangles, and polygons; area fill of these shapes in a designated color at video rates; text generation and rotation; the ability to open and close windows in the page of memory being displayed; the ability to simulate motion (animation); the ability to CLIP which eliminates problems which might arise from trying to plot outside the screen area; and the ability to scale the display area of the work page.

The programmer can generate and display an image in high resolution (756 x 482 points) as well as the 16 color medium resolution (378 x 241 points) using the same system. In addition, the programmer has the choice of plotting explicitly (i.e., specifying within a call all needed location and color information) or implicitly (i.e., specifying needed location information with regard to an implied curson).

The software and hardware permit the user to select 16 colors for the color map from a palette of 4096 colors. The contents of any color in this color map can be modified by the user with a imple call define color command. In addition, when programming in Fortran or Assembly language, the programmer has the option of creating color maps using the command CMAPGEN.

The SDI color graphics software package is available from Cromemco on either 5" (Model SGS-S) or 8" (Model SGS-L) diskette for $295. For additional information, contact Cromemco, Inc., 280 Bernardo Avenue, Mountain View, CA 94043, (415) 964-7400.

THE DS MICRO TRANSLATOR SYSTEM TRANSLATES BRAILLE & DOES WORD PROCESSING

DUXBURY SYSTEMS, INC., of Maynard, Mass, has introuced a microcomputer-based, word processing an braille translating system. It includes a North Star Z-80 based microcomputer with dual, quad-capacity diskettes for translation and storage; a screen terminal with keyboard for input; the Triformation Systems' LED-120 (Line Embossing Device, 120 characters per second) for braille production; Duxbury Systems software for text editing, braille translation, and formatting; and an optional letter-quality printing terminal for inkprint production. In addition to the LED-120 translated tect can be transferred to any of the Triformation LED series embossing devices, or the press braille plate embosser, model PED-30, or the hand-held Digicassette braille recorder.

In operation, the system permits a sighted person with no knowledge of Standard English Braille, which is a complex, semi-phonetic code, to enter and edit material from newsletters, memors, and class notes to full-length books. Entered text is automatically translated and formatted by the computer to produce high quality braille copies of the material, or formatted without translation for equivalent print copies.

The Braille translator follows the same internal procedures as Duxbury's minicomputer-based systems, currently in use by large braille production houses in North America and Australia; consequently the same close conformity to Standard Braille Code is maintained. Different language reference tables are available to produce Grade I (uncontracted) or Grade II (contracted) braille according to American rules or the slightly different British braille code. Grade I and Grade II braille are also available in Spanish, along with Grade I in four other languages. The micro translator renders text into braille at the rate of more than 300 characters per second. The system is also designed to be used eventually for large print production (for persons with low vision) and for translation from braille to print (e.g., from cassette braille recorded by a blind person).

The cost of terminal, computer, and software is $8,950. The LED embosser costs between $7,400 and $14,500. An optional, letter quality printer adds $2,500. For further information please contact: Duxbury Systems, 56 Main St., Maynard, Massachusetts, 01754.

COMPLETE DATA ACQUISITION AND VIDEO S-100 MICROCOMPUTER SYSTEMS

Tecmar Inc. announces two S-100 stand-alone microcomputer systems for general use as well as for data acquisition and/or video applications. These systems are ideal for use in laboratories for research, pulmonary function analysis, stress testing, radiology, production lines, quality control, surveillance, etc. The systems are fully assembled, tested, and warranteed.

Each system consists of:

1. Z-80
2. 64 Kbyte of 300 nsec static RAM
3. ADM-3A alphanumeric terminal (80 characters x 24 lines, 19,200 baud)
4. Dual 8" floppy disks (Shugart)
5. Disk controller
6. 24 parallel lines (for end-user)
7. Desk-top enclosure with power supplies and cables
8. Supports CP/M 1.4 or 2.0
9. Supports high level languages such as Fortran, Cobal, Basic, C, Pascal, etc.

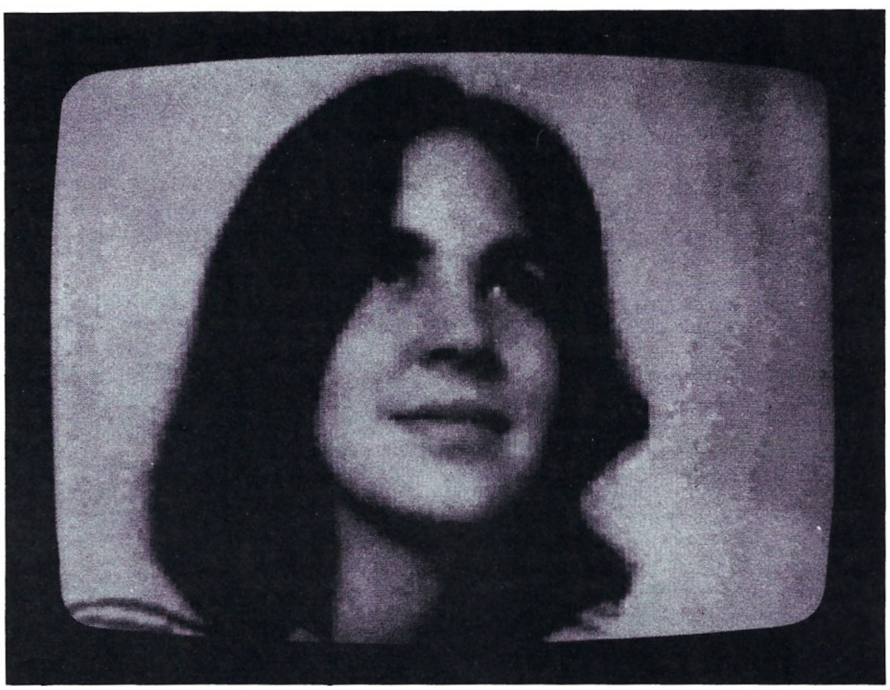
The TECMAR COMPLETE DATA ACQUISITION MICROCOMPUTER SYSTEM includes the above basic system plus:

1. One Tecmar 16 channel, 12 bit, high speed (30 KHz throughput) A/D
2. Two Tecmar 4 channel, 12 bit, high speed (3 uses settling time) D/A boards for a total of eight D/A channels

The TECMAR REAL-TIME VIDEO MICROCOMPUTER SYSTEM includes the above system plus:

1. Tecmar's Real-Time Video Digitizer and Monitor Interface featuring digitization in 1/60th sec and maximum resolution of 512 x 240
2. Video Camera
3. 9" video monitor

The Data Acquisition Microcomputer System and the Video Microcomputer System are each priced at $6,950.00. The options include rack mount units, printer/plotters, and additional S-100 boards. TECMAR, INC. 23414 Greenlawn Avenue, Cleveland, Ohio 44122 (216) 382-7599.



Portrait digitized with Tecmar's Video Microcomputer System (resolution:128x128)

In each issue of S-100 MICROSYSTEMS we will have this catalog listing of S-100 system software. If you have a software package you are offering for sale and want to be listed then send us the information in the format shown. All information must be included. We reserve the right to edit and/or reject any submission.

SOFTWARE DIRECTORY

Program Name: APL Version 2.3
Hardware System: CP/M system
Minimum Memory Size: 48K
Language: 8080 Machine Code
Description: An APL implementation having most of the functions and operators of full APL, including n-dimensional inner and outer product, reduction, compression, general transpose, reversal, take, drop, execute, format, logarithm, exponential, power, and the circular functions sine, cosine, tangent and arctangent. It has system variables, system functions and shared variables for CP/M disk I/O. The interpreter will run in ASCII using CP/M standard I/O. In addition, it supports typewriter and bit-pairing ASCII-APL character sets and can run with user-supplied I/O drivers. A driver for a video display with programmable character generator is included. SOFTRONICS APL uses Abrams' descriptor calculus and shared data storage to save memory space and execution time.
Release: Available now
Price: $350
Included with price: CP/M disk, 112 page user manual which includes an APL tutorial
Where to purchase it:
Softronics
36 Homestead Lane
Roosevelt NJ 08555

Program Name: ISSCAI SYSTEM
Hardware System: Standard CP/M or MP/M
Minimum Memory Size: 8K
Language: 8080 Assembler
Description: This is a set of three programs CAIGEN, TUTOR, and ENROLL which provide, with the use of a system editor, a complete COMPUTER AIDED INSTRUCTION system. CAIGEN formats a editor written text file to the requirements of TUTOR and creates an enrollment file for the course if needed. TUTOR is the heart of the system providing foward and reverse linking of text, prompting for answers even where there might be several that are correct, responding on correct or incorrect answers with replys if wanted, chaining to next lesson, scoring, passwords, comments and several other functions. ENROLL provides complete enrollment file maintenance and teacher monitoring of student progress in a course, lesson by lesson, this program has password level access.
Release: Currently available
Price: $250.00
Included in price: Object of three programs and users manual. System is available for RESALE LICENSE.
Author: G.B. Shaffstall
Where to purchase it:
International Software Service
13050 W. Cedar Drive #15
Lakewood, Colorado 80228

Program Name: Apparel Management System
Hardware System: CP/M 48K, 2-8" Drives
Language: CBASIC-2
Description: This system is designed to help management make decisions about their stores. Items to reorder that will still make the season, items to be moved from one store to another and items to be marked down are some of the daily tools provided. A detailed inventory report by department shows inventory information (units, dollars in stock, etc.) and monthly sales information. A monthly analysis is done by store/department showing sales, COGS, profit, annual inventory turns, stock to sales ratio and sales compared with budgets. The annual report follows the key mothly analysis figures for a year, again for your comparison abilities. Other major reports include daily sales by department, yearly budgets and physical inventory taking sheets.
Release: Available now
Price: $960.00
Included with price: User documentation, 31 programs warranty
Author: Keystone System, Inc.
Where to purchase it:
Keystone Systems, Inc.
P.O. Box 767
Spokane, WA 99210

Program Name: SCREENMASTER
Hardware System: 48K CP/M system. Dumb terminal = Hazeltine, ADM3A, TRS-80 II. Others easily accomodated.
Minimum Memory Size: 40K, 48K recommended.
Language: CBasic-2. Distributed in source code.
Description: Intended for programmers only, Screenmaster allows user to describe multi-screen input via data to the program. Program returns an array of responses, edited for validity. Programmer has pre-/post-input and submit exits where editing and control code may be inserted, commands = 90 to m, back n, forward n, prior (screen 0, next (screen), submit etc. End-user can also be given the commands. Flexible design allows any input scheme to b implemented in minutes rather than days.
Release: Available now
Price: $295. Compiled Demo $50. User manual alone $25.
Author: Dr. Laird Whitehill & Joel Wittenburg.
Where to purchase it:
Micro-computer Business Systems
161 W. 75 St.
New York, NY 10023

Program Name: MWP-SEL
Hardware System: CP/M
Minimum Memory Size: 48K bytes
Language: Microsoft Basic
Description: Allows sophisticated selected records. Example: Select all records with 'AMOUNT DUE' greater than or equal to (=) 'CREDIT LIMIT'. You may combine selection criteria with AND' and 'OR' for complex task.
Release: Available now
Price: $95; Licence 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

THE ONLY MAGAZINE BY AND FOR S-100 SYSTEM USERS!

S-100 MICROSYSTEMS™

At last there is a magazine written exclusively for S-100 system users. No other publication is devoted to supporting S-100 system users. No longer will you have to hunt through other magazines for an occasional S-100, CP/M* or PASCAL article. Now find it all in one publication. Find it in **S-100 MICROSYSTEMS.**

Every issue of **S-100 MICROSYSTEMS** brings you the latest in the S-100 world. **Articles on applications, tutorials, software development, letters to the editor, newsletter columns, book reviews, new products, etc.** Material to keep you on top of the ever changing microcomputer scene.

SOFTWARE
CP/M*
Assembler
BASIC
PASCAL
applications
and lots more

SYSTEMS
Cromemco
Intersystems
North Star
IMSAI
SOL
and lots more

HARDWARE
8 bit & 16 bit CPUs
interfacing
hardware mods
bulletin board systems
multiprocessors
and lots more

*TMK Digital Research

Edited by Sol Libes
Published every other month

USA	Canada, Mexico	Other Foreign (Air)
	THREE YEARS (18 issues)	
☐ $23.00	☐ $32.00	☐ $65.00
	TWO YEARS (12 issues)	
☐ $17.00	☐ $23.00	☐ $45.00
	ONE YEAR (6 issues)	
☐ $9.50	☐ $12.50	☐ $23.50

☐ New ☐ Renewal
☐ Payment Enclosed
☐ Visa
☐ MasterCard
☐ American Express
Signature ____________________
Card No. ____________________
Expiration date ____________________

☐ Please bill me ($1.00 billing fee will be added; foreign orders must be prepaid)

S-100 MICROSYSTEMS
P.O. Box 789-M, Morristown, NJ 07960

S-100 MICROSYSTEMS™
ORDER FORM (SMLG)

Name ____________________
Address ____________________
____
City ____________ State ________ Zip ________

BACK ISSUES

☐ 1-1 Jan/Feb 1980 $5.00
☐ 1-2 Mar/Apr 1980 $2.50
~~1-3 May/Jun 1980 $2.50~~ *Sold out!*
☐ 1-4 Jul/Aug 1980 $2.50
☐ 1-5 Sep/Oct 1980 $2.50
☐ 1-6 Nov/Dec 1980 $2.50

Postpaid in USA; add $1.00 per issue foreign postage. Subscriptions start the month following receipt of order. Subscriptions cannot start with earlier issues.

Program Name: D—Directory and Disk Status
Hardware System: any CP/M system
Minimum Memory Size: 16K
Language: 8080 Assembler
Desription: This program works with single or double density systems on any selectable disk drive. The directory is presented in 4 columns sorted into alphabetical order (the number of columns is equate selectable in the source program). The first line contains the following disk information: Disk:? Files:? Entries:? (? left) Space used:? K (? K left).
Release: Available now
Price: $40.00 Source $20.00 Object
Included with price: Program and documentation.
Author: Hawkeye Grafix
Where to purchase it:
Hawkeye Grafix
23914 Mobile St.
Canoga Park, CA. 91307

Program Name: Tarbell Dual-Density DMA Support Package
Hardware System:8080/Z80 S100 system with Tarbell DD/DMA disk controller.
Minimum Memory Size: N/A
Language: 8080 Assembly Language (ASM or MAC)
Description: CP/M 2.0 compatible BOOT and BIOS for Tarbell Dual Density disk controller, including all support programs required for normal operation (FORMAT, Disk validation, Fast absolute copy, auto-density sysgen, etc.) Not compatible with public domain code from Tarbell, this is all new code which supports IBM standard gaps and header information, has no known bugs, and is very clearly written. Currently supports CP/M with 128 byte sectors only, but will allow user to format and validate diskettes in any of the following formats (sectors/track x bytes/sector): Single: 26 x 128, 13 x 256, 8 x 512, 4 x 1024 Double: 48 x 128, 26 x 256, 15 x 512, 8 x 1024 Supports standard IOBYTE, remote console auto answer dial-in access, etc. Console/printer I/O currently uses IMSAI SI02-2 (very easy to modify).
Release: Available now
Price: $50.00
Included with price: 8" 3740 CP/M style disk with source for BOOT, BIOS, FORMAT, VALDSK, ADCOPY and SYSGEN. Note: CP/M 2.0 from Digital Research required.
Authors: Lawrence E. Highes and Sam H. Adams
Where to purchase it:
Mycroft Labs
P.O. Box 6045
Tallahassee, Fla 32301

Program Name: PRO-TYPE Word Processor
Hardware System: CP/M North Star, MECA ALPHA TAPE
Minimum Memory Size:16K
Language: Basex
Description: IMI's PRO-TYPE is a powerful word processor that is easy to learn and simple to use. Its comprehensive 72-page manual will guide you from beginner, to intermediate and on to advanced applications. PRO-TYPE packs all of these convenient features into a single 8K program that supports fully interactive text entry, editing, and print formatting: Works with ANY type of terminal (memory mapped or not). Floating tabs and underlining. Change left and right margin, line spacing while printing, double text buffers for form letters, etc. Multiple print modes (justification, line fill, verify) Embedded "STOP" codes allow special text insertion command malow special text insertion command macros for repeated command execution.
Release: Available now
Price: North Star 5" SD & DD disk (with manual) $25. MECA ALPHA tape (with manual) $75. CP/M 8" disk (with manual) $75. Add $.75 Special 4th Class or $1.50 Special Handling or UPS
Included with price:72 page manual and disk or tape
Author: Paul K. Warme
Where to purchase it:
Interactive Microware
P.O. Box 771
State College, PA 16801

ADVERTISER INDEX

Advertiser	Page
Ackerman Digital Systems	11
American Square Computers	37
Budget Infosystems	Cover 3
Casheab	6
Computer Design Labs	31
Computer Mart of New Jersey	6
Creative Computing	47,51
Digital Graphic Systems	11
Electronic Control Technology	1
Electronic Specialists	16
Godbout Electronics	Cover 4
Hawkeye Grafix	37
International Micro Machines	27
Ithaca Intersystems	5
JES Grafics	20
Konan Corporation	39
Lifeboat Associates	12,13
Merrimack Systems	16
Microsystems	55
Morrow Designs	Cover 2
Northwest Communications	25
Piiceon	21
Potomac Micro-Magic	56
Quasar Data Products	2
S-100, Inc	37
Sensational Software	9
Tarbell Electronics	7
Tecmar	19
Theta Labs	36

Budget InfoSystems

Why Pay More?

Budget InfoSystems maintains a price competative inventory of several major manufacturer's product lines - all attractively discount priced to $ave you money. . .institutions & OEMs will find our prices less than most local distributors, and we don't require "blanket purchase orders!" Local prices may be somewhat higher on systems purchases only. Most items shipped from STOCK to 2 weeks - Inter-Systems products may run to 8 weeks. Minnesota residents must add 4% sales tax, unless purchased for resale. PREPAY by bank transfer or other means saves normal freight charges.

VECTOR GRAPHIC S100 SYSTEMS:

All of the following Vector Graphic Systems include 64K RAM, "ZCB" single board computer (with I/O ports, space for 16K ROM, etc.), 80 x 24 Flashwriter II video, disk controller, CP/M 2.X, Microsoft BASIC-80, ZSM assembler, the incomparable SCOPE screen editor (with horizontal scroll), RAID debugger, and more! System B utilizes Vector's "Mindless Terminal" & an "MZ" mainframe, while all other systems use an intelligent "Vector 3" terminal with integral 6-slot S100 card cage (4 slots used). The most comprehensive documentation in the industry accompanies each system. For a limited time, if you Order a Qume printer with your system and we'll include MEMORITE Word Mgmnt. software FREE!

NEW! "VIP" (single Micropolis 315K disk)	3695.00	2999.00
NEW! "VIP Plus" (BIS exclusive - dual 315K disks & more!)		3799.00
System B (dual 315K disks)	5463.00	4399.00
80 x 24 Time-share terminals w/64K RAM	1750.00	1450.00
System 2800 (dual 1MB Qume drives)	7295.00	5899.00
System 3030 (Micropolis 32MB Hard Disk)	11945.00	9699.00

(ask about our "nearly FREE" Peachtree accounting & inventory software packages for ALL Vector systems)

MODULAR S100 MICROCOMPUTER SYSTEMS (MAINFRAMES ONLY):

Vector Graphic MZ(64K, CPM2, & much more!)	4313.00	3499.00
Intersystems DPS-1(w/front panel)	1795.00	1625.00
Intersystems DPS-1FPL (front Paneless)	1395.00	1275.00
Intersystems System 1A (w/64K, Z80&I/O)	3195.00	2875.00
Intersystems System 2A (same plus FDC2)	3595.00	3250.00
Intersystems System DPX-1 (1A w/Z8001)	4895.00	4425.00
Intersystems System DPX-2 (2A w/Z8002)	4795.00	4325.00
(other configurations available on request)		
Northstar Horizon I DD 16K - DEMO - (one only)		only 1299.00

CRT TERMINALS:

Microterm 2A	995.00	850.00
Microterm 5A	945.00	799.00
Microterm 314	895.00	750.00
Microterm 100 (DEC VT1000 emulator)	1895.00	1699.00
TVI 912B & 912C	to 895.00	799.00
TVI 920B & 920C	to 995.00	850.00

MATRIX PRINTERS (RO):

Anadex 9500 printer	1650.00	1450.00
Centronics 779TF	1350.00	1099.00
Centronics 737 (add $99 for serial vers.)	995.00	850.00

DAISYWHEEL CHARACTER PRINTERS (RO):

BudgetRiter III (Qume 3/55 w/S100 interface card)		2499.00
Qume Sprint 5/55	3195.00	2899.00
Plessy CP4282 (by Data Products)	3595.00	2499.00
Vector/Qume Sprint 3/55 (for System B)		2799.00
Vector/Qume Sprint 3/55 (for VIP, 2800 & 3030)		2999.00

MASS STORAGE SUBSYSTEMS:

BudgetStor I (dual Micropolis 315K drives)		1599.00
BudgetStor II (dual 8" drives - 1.2MB)		1499.00
BudgetStor III (dual 8" Qume drives - 2MB)		1899.00
KONAN "Hard Tape" controller, with Century Data 10MB disk & DEI 15.5MB tape drive	6455.00	5999.00
Same with 20MB disk drive	6915.00	6399.00
Same with 40MB disk drive	7685.00	6999.00

CF & A DATA DESKS:

Universal Printer Stand (all metal)	130.00	115.00
CRT data desk 48" wide	178.00	160.00
Data desk return 18" x 36"	145.00	135.00
Microshelf (specify mainframe size)	110.00	95.00
Microshelf (vented for Vector MZ)	135.00	120.00

28" Ht. and 30" depth are standard; returns are 26" ht.

Top colors: Teak, Walnut, Pecan, Black, White, Putty, Champagne
Nonstandard tops are $25.00 extra (any std. Formica colors)
Sheetmetal colors: Champagne, Black, Blue, Yellow, Putty, Brown, Butterscotch; legs are black (chrome $50.00 extra).

SOFTWARE:

Vector Graphic MEMORITE III Word Management System - NOW CP/M 2.2/Microsoft BASIC compatible, with proportional spacing, and many new features! The finest "canned" word processor available CALL FOR PRICE

"Peachtree" accounting software most systems — $500.00 per module
SPECIAL for existing Vector Graphic systems — only $250.00 per module

OS-1 "UNIX-like" Z80 Optimized Operating System & Utilities (fully CP/M compatible) by Software Labs - shipped on three 8" CP/M disks $225.00

10% off the complete line of Microsoft software!

MDBS Data Base Management System (CODASYL-compatible) the last word in DBMS's! available in a wide variety of formats, with or w/o optional modules $750.00
HDBS Data Base Management System - the 2nd to the last word!!! $350.00

We have a wide variety of development and applications packages from several of the major software houses, all at discount prices!

S100 BUS CIRCUIT CARD ASSEMBLIES:

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

MISCELLANY:

Budget ansr./orig. modem		150.00
Novation CAT ansr./orig. modem	189.95	175.00
Qume Single sheet paper feeder	1390.00	1199.00

OPUS SD/SS Mini-diskettes (10 in plastic box) — SPECIAL 27.50
(specify 0, 10, or 16 sectors & system) — 100 in bulk $255.00

OPUS SD/SS 8" Diskettes (10 in plastic box) — SPECIAL 29.50
(specify 0 or 32 sectors & system) — 100 in bulk $275.00

OPUS RECORDING MEDIA ARE A PREMIUM GRADE FULLY GUARANTEED PRODUCT OFFERING SUPERIOR VALUE. . .DOUBLE DENSITY OR SPECIAL FORMATS AVAILABLE AT SLIGHTLY HIGHER PRICES.

CompuClean head cleaning diskette 5¼" — 19.00
CompuClean 8" — 19.00
(Specify single or double sided) — three packs only 49.00
These are a dry process - no mess & much safer!

BUDGET INFOSYSTEMS
1633 N.E. Highway Ten
Spring Lake Park, MN 55432
(612) 786-5545
Normal Order Line Hours Are: 12:00 to 6:00 p.m. CST