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

$2.95 U.S.A.

MICROSYSTEMS™

MAY/JUN 1982 VOL. 3/NO. 3

DATA BASE MANAGEMENT SYSTEMS

See Pages 16-44

Also in this Issue

Cursor Addressing by Fred Gohlke....45
Structured Programming In Basic by Bob Kowitt....50
A Replacement For CP/M SUBMIT by David E. Cortesi....60
Running North Star Basic With CP/M by Steve Leibson....80

and more

Complete Table of Contents on Page 3

MORROW DESIGNS

Morrow DMA Controllers Supercharge Disk Systems.

What EDN said: "The (Disk Jockey™ DMA floppy-disk controller) offers two to three times the performance of comparable controllers." And that's for either 5¼'' or 8'' floppy-disk drives operating from an S-100 (IEEE-696) bus.

How we do it? With channel drive that's almost identical to IBM 370® channel controllers. The DJ/DMA uses 24-bit addressing. The host writes commands into memory. The DJ/DMA picks up commands from the host processor via memory on the system bus, and transfers data during DMA cycles. Channel commands may be located anywhere in the 24-bit address range. Upon completion of the command, the controller returns status. It may also generate an interrupt. Chained commands allow the controller to return status, or to execute a number of commands in succession. The controller board also contains logic which allows other IEEE-696 temporary bus masters to contend for memory cycles. That's the Morrow "channel concept." Fast and Simple.

On-board Z-80A.* By managing both memory and disk transfers, the resident Z-80A allows reading or writing to almost any floppy-disk media. 8'' or 5¼''. Single or double density. Single or double-sided. And up to eight drives per controller board (no more than four of each type).

System compatible. Disk Jockey DMA sub-systems are compatible with all IEEE-696, S-100 systems (such as the Morrow Decision I™). And, with most S-100 like systems.

Faster Winchesters. Hard disks put micros in the mini category. Now, Morrow's channel driven DMA controller concept makes Winchesters lightning fast. Which Winchester? Industry standards—Seagate's 5¼'' ST506/512, or 8'' drives from Shugart/Quantum.

DMA Transfer (Burst Mode). The Morrow Direct Memory Access Hard Disk Controller (HDC/DMA) picks up commands from the host processor via memory on the system bus. Commands are accessed and data is transferred during DMA cycles. Commands and data transfers may occur anywhere in the 24-bit address range.

Interrupts. The controller can generate an interrupt at the end of each command and/or at the end of each command chain.

Imbedded μP. An on-board 8X300 supervises data transfers between the Winchester drive(s) and main memory. Microcode in this 7 MHz bipolar microprocessor implements the command structure of the controller.

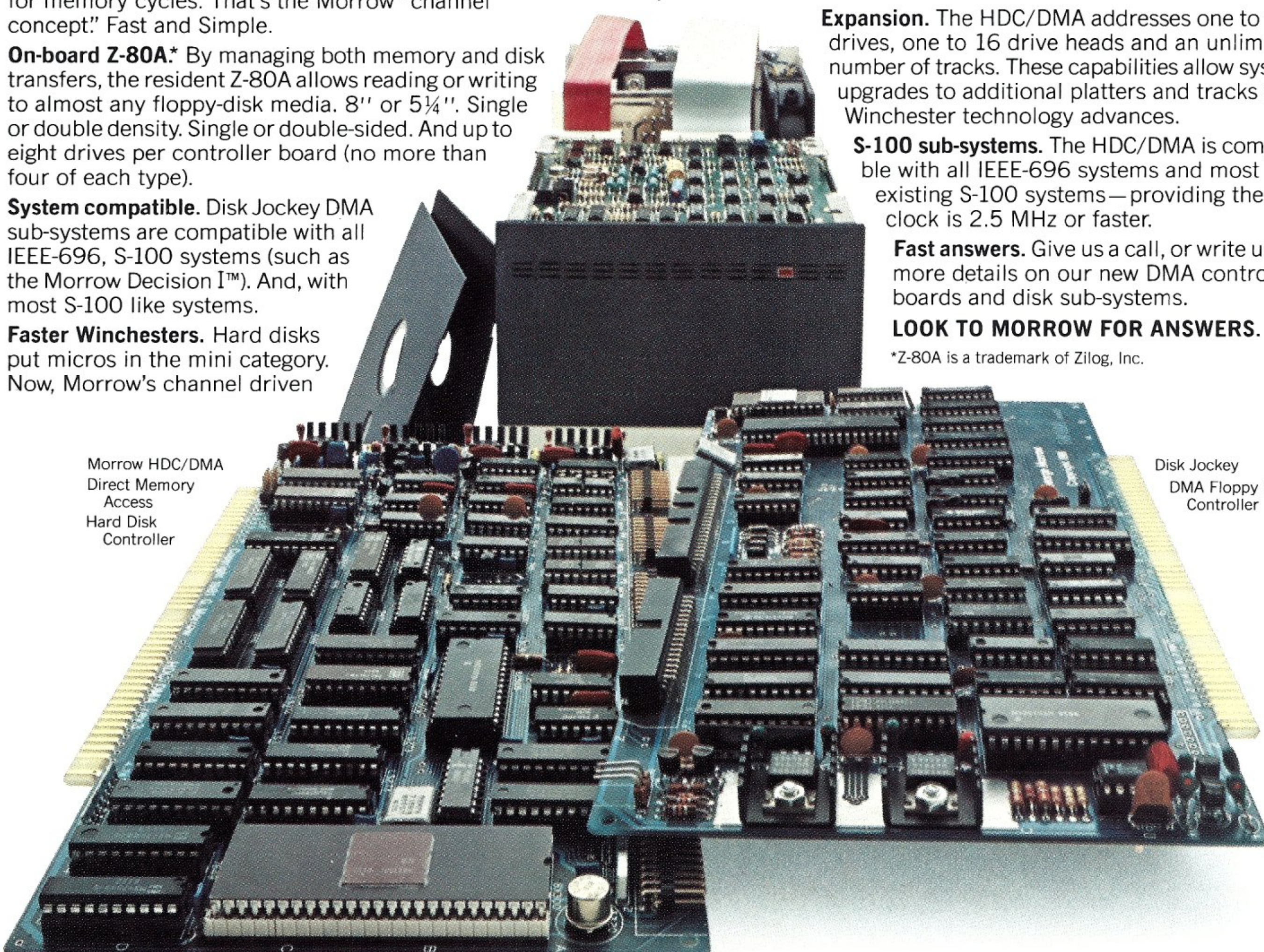
Expansion. The HDC/DMA addresses one to four drives, one to 16 drive heads and an unlimited number of tracks. These capabilities allow system upgrades to additional platters and tracks as Winchester technology advances.

S-100 sub-systems. The HDC/DMA is compatible with all IEEE-696 systems and most existing S-100 systems—providing the bus clock is 2.5 MHz or faster.

Fast answers. Give us a call, or write us for more details on our new DMA controller boards and disk sub-systems.

LOOK TO MORROW FOR ANSWERS.

*Z-80A is a trademark of Zilog, Inc.



IBM/370 is a registered trademark of IBM Corp.
Disk Jockey and Decision I are trademarks of Morrow Designs.

MORROW DESIGNS
5221 Central Avenue, Richmond, CA 94804
(415) 524-2101, TWX: 910-382-8132, (Power Key-Richmond)

POWER & RELIABILITY

Toughest Boards in Town... IEEE696/S-100. Systems too!

256 Kb Dynamic Memory

Available now from Dual Systems: DMEM/256 memory boards. Put the most density of memory ever available on your IEEE/S-100 bus. Get industrial-grade quality with this ruggedly built board: it's been burned-in for 168 hours.

- ☐ FULL 256 Kb on a single board.
- ☐ Two independently addressable 128 Kb blocks.
- ☐ Runs in 8 MHz system.
- ☐ Runs at 4 MHz bus speed (no wait states).
- ☐ Runs in 8 or 16-bit systems with 8 or 16-bit wide data paths.
- ☐ Extended 24-bit addressing.
- ☐ Parity error or detection-generates either interrupt or bus error signals on error.
- ☐ DMEM/256KP (Parity)... $1495

Static Non-Volatile Memory

The Dual Systems CMEM memory boards combine high speed CMOS memories with new 3–10 year lithium batteries to give you the non-volatility of an EPROM board while retaining the instant writability of a high-speed read/write RAM.

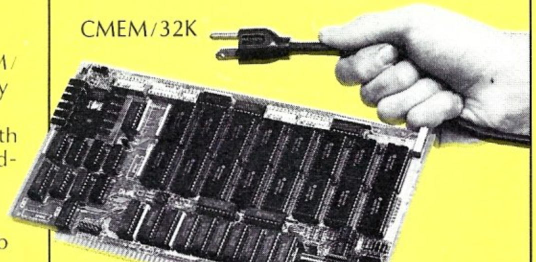
- ☐ Runs at 6 MHz (no wait states).
- ☐ 8 or 16-bit with 8 or 16-bit wide data paths.
- ☐ Extended 24-bit addressing and bank select.
- ☐ CMEM/32K $995
- ☐ CMEM/16K $795
- ☐ CMEM/8K $695

Non-Stop Clock

Keeps time with power off. Our industrial clock utilizes a new lithium battery for 3–8 years use. Easiest clock to program you'll ever see. Runs in all S-100 systems.

- ☐ Year, date, hrs., mins., and secs.
- ☐ Uses new LSI CMOS chip.

CMEM/32K



A/D Converter

IEEE696/S-100 AIM-12 industrial standard module designed for industrial analog-to-digital use.

- ☐ Runs in all S-100 systems.
- ☐ 32-channel, 16-differential.
- ☐ 12-bit resolution/accuracy.
- ☐ 25-microsecond conversions.
- ☐ Instrumentation amplifier.
- ☐ BASIC program provided.
- ☐ AIM-12 w/1-1000 gain transducer amplifier... $785.
- ☐ AIM-12B... $695.

D/A Converter

AOM-12 IEEE696/S-100 industrial level digital-to-analog (D/A) converter.

- 12-bit ± 1/2 L.S.B. accuracy over full 0–70°C temperature range.
- Outputs, 0–10, ± 5, or ± 10 volts.
- Short circuit protection, all outputs.
- Switch-programmable for multiple boards.
- AOM-12: $675.

VIC 4-20

Standard output for industrial control 4-20 mA D/A converter. Used in conjunction with the D/A board.

- VIC4-20, $675.

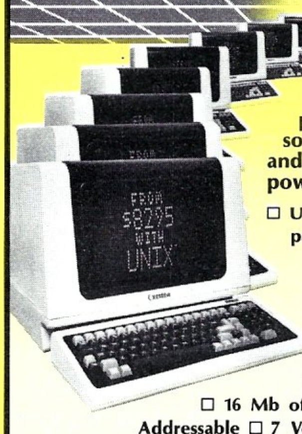
DMA Serial I/O

The SIO-4/DMA is an intelligent, DMA, high performance S100 four port serial input/output board. It conforms to the IEEE-696 specification and is designed to work with older S100 machines. The board is intended to be used with asynchronous terminals or printers in a multi-user system.

- ☐ Program selectable Baud rate.
- ☐ DMA transfers for output.
- ☐ 24-bits of address supported with no restrictions on boundaries.
- ☐ Can be connected to RS-232 terminals or modems.
- ☐ 256-bytes of FIFO buffer for input characters.
- ☐ RS-232 drivers and receivers conform fully to specification.
- ☐ Control-S/Control-Q protocol supported.
- ☐ Printers with 'Buffer full' signal lines supported.
- ☐ Limited synchronous mode capability.
- ☐ Occupies only 16 I/O addresses.

...Hard Power from the Soft Support of

UNIX®



A new and powerful computer has been born... the System 83. The versatile UNIX* operating system pilots the System 83's raw power through a myriad of software such as "C", FORTRAN, PASCAL, BASIC, COBOL, and even Networking. Step into a bold new frontier with more power than you ever dreamed possible.

☐ UNIX V7 configured by UNISOFT ☐ Full IEEE 696/S-100 Compatability ☐ MC68000 8MHz Processor ☐ 32-Bit Data Operations with 32-Bit Internal Registers ☐ 16-Bit Data Transfer Operations ☐ Memory Management Allows Concurrent Use of Mapped and Non-mapped Address space ☐ 16 Mb of Main Memory Directly Addressable ☐ 7 Vectored Interrupt Levels ☐ 256 Kb of RAM with Parity Per Board Slot**

* UNIX is a trademark of Bell Laboratories and is supported on the DUAL System 83 by UNISOFT
**UNISOFT is a trademark of UNISOFT Corporation of Berkeley, CA.

OEM and Dealer pricing is available.

Sales representatives in most metropolitan areas.

DUAL

DUAL SYSTEMS CORPORATION

system reliability/system integrity

720 Channing Way • Berkeley • CA 94710 • (415) 549-3854/3890 • Telex: 172029 SPX

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

MICROSYSTEMS™

Volume 3/Number 3 May/June 1982

Staff

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

For information on commercial advertising, write to: MICROSYSTEMS Advertising Dep't., One Park Avenue, New York, NY 10016, or call Jeff Weiner at (212) 725-7957.

MICROSYSTEMS (ISSN #0199-7955) is published bi-monthly by Ahl Computing, Inc, a subsidiary of Ziff-Davis Publishing Company. David Ahl, President; Elizabeth B. Staples, Vice-President; Selwyn Taubman, Treasurer; Bertram A. Abrams, Secretary.

Second class postage paid at New York, NY 10016 and at additional mailing offices. POSTMASTER: Send address changes to Microsystems, PO Box 789-M, Morristown, NJ 07960.

Subscriptions are $16 a year (6 issues USA). To subscribe, call our toll free number: (800) 631-8112, or (201) 540-0445 in New Jersey.

In This Issue

Data Base Management Systems

An Introduction To DBMS.....16
Dr. Christopher L. Hamlin

DataStar: An Evaluation.....22
Glenn A. Hart

A Review Of MDBS.....28
Bill Machrone

Total Information Management Database Manager..34
Glenn A. Hart

Three Ways To Implement A Mailing List.....40
Bill Machrone

Cursor Addressing.....45
Fred Gohlke

Structured Programing In Basic.....50
Bob Kowitt

Minimizing The Inconvenience Of Compiled Languages Under CP/M.....52
Leonard Schwab

A Replacement For CP/M Submit.....60
David E. Cortesi

CP/M Disk Directory And Table Secrets.....68
Andrew Klossner

More Modifications For The SDS VDB—8024.....72
Richard Bowersox

A Review Of Mince.....76
Chris Terry

Running North Star Basic With CP/M.....80
Steve Leibson

Book Review.....87
Jeff Duntemann

A Bargain Z—80 Debugging Program.....88
Andrew Bender

Departments

News & Views.....4
Letters to the Editor.....6
Editor's Page.....8
Software Directory.....90
New Products.....94
Advertiser Index.....96

NEWS & VIEWS

by Sol Libes

MP/M Slated For Intel Chip

Intel Corp and Digital Research have signed an agreement whereby a special version of MP/M-86 will be developed for the new Intel iAPX-286 microprocessor. The version will be known as MP/M-286 and should become available this fall.

The iAPX-286 will use a new enhanced version of the 8086 to be called the 80286. It will be software compatible with the 8086. The new IC will have an on-chip multi-level protection mechanism and capabilities for memory management and virtual memory address translation. Hence, it is targeted for the multi-user and multi-tasking market. The IC has 68 pins.

Another 8086 upgrade is expected to be released shortly and should be called the 80186. It will have on-chip clock and interrupt controllers, two DMA channels, timers, counters and random chip-select logic.

Apple & Commmodore Adopt CP/M

Both Apple Computer and Commodore International have finally decided to adopt CP/M. Apple will offer CP/M as an option on their newly re-introduced Apple-III. And Commodore will offer a CP/M emulator on a new computer they plan to introduce shortly. A prototype of this new computer was shown at a dealer show recently and reportedly employs the new 6510 microprocessor, an enhanced 6502.

This leaves Radio Shack as the only major supplier of microcomputer systems who has yet to become a CP/M supplier. Of course,CP/M for the TRS-80 is available from several independent sources.

Hence, despite criticism of CP/M and predictions of its demise, the operating system continues to increase its dominant postion in the single-user personal computer marketplace.

CP/M Simulator Available For DEC Computers

Virtual Microsystems, 2409 Telegraph Ave, Berkely, CA 94704, has released a CP/M simulator for DEC PDP/11 and VAX systems, running RSTS, RSX or UNIX. Called "the Bridge" it simulates the running of CP/M, creating "virtual floppies." With an appropriate floppy disk drive on the system, the user can load in CP/M software such as WordStar or Supercalc, and run these programs.

Random News Bits

Ithaca Intersystems' new version of Pascal/Z (4.0) has a unique feature, an interactive symbolic debugger called "Swat"....Intel is reportedly shipping its Ada compiler for the iAXP432 32-bit microprocessor and hopes to have their 432 development system available in the last quarter of this year...Amateur radio enthusiasts might like to know that an S-100 DTMF transceiver board is available from MK Enterprises, 8911 Norwich Rd., Box 29654, Richmond, VA 23229, 804-740-8380. Price is $425.

User Group News

An independent Nevada Cobol users group has been formed to distribute information on applications and routines. For more information contact Bob Blum, Chairman, Nevada Cobol User's Group, 5536 Colbert Trail, Norcross, GA 30092, (404)449-8948.

A Canadian Osborne-1 User Group has been formed called "Ozymandias-II." For more information write to them at Box 65, Station G, Toronto, Ontario, Canada M4M 3E8.

INTRODUCING PERFECT WRITER™. NOW, GET THE FINEST WORD PROCESSOR AT ANY PRICE FOR JUST $289.00

AND GET OUR INCREDIBLE, MONEY-BACK GUARANTEE!

That's right. If you own an Apple II, Heath/Zenith, TRS-80 model II, North Star, Osborne 1, Superbrain, Micropolis/Vector, or CP/M 8," you can now afford the most advanced, yet easy-to-use, word processor on the market today.

But don't take our word for it. See for yourself.

We're so sure you'll be amazed by Perfect Writer's easy-to-use features and capabilities, that we'll let you try it for thirty days and give you your money back if you're not completely satisfied!

FEATURES AND FLEXIBILITY YOU HAVE TO SEE TO BELIEVE.

Looking for the advanced features you'd expect to find in the most expensive word processors? Perfect Writer has them. Like automatic formatting of letters, out-lines, reports, memoranda, articles, books, or other types of documents.

What makes Perfect Writer unique, are the features you won't find in word processors costing many times its price.

Like the incredible sophistication of Virtual Memory Architecture, that lets you edit a full-length novel as easily as you'd edit a letter.

Or a remarkable breakthrough called Split Screen Editing™ that lets you divide the screen into two parts, each showing a different document, and transfer sections of text between the two!

Perfect Writer.
The remarkable word processor combining sophistication and simplicity.

And that's just the beginning. Perfect Writer offers much, much more:

- Automatic numbering of pages, chapters, sections, footnotes, internal page references, and automatic generation of table of contents and indexes.
- Mail merging capability allows generation of form letters and more.
- Automatic carriage return, auto-centering, tab setting, and automatic placement of footnotes.
- Quick and easy cursor commands.
- Easy deleting and inserting procedures.
- Safe and convenient storage of materials.
- Searches for and replaces words and phrases in a document, including near misses.
- Full range of printing options, including right margin justification, micro-spacing, super and subscripts, italics, and bold face.
- And lots more.

MONEY BACK GUARANTEE.

Try Perfect Writer for 30 days at our risk. If you're not completely satisfied, return it for a prompt, complete refund.

PERFECT WRITER IS EASY TO USE. WE GUARANTEE IT.

When we say that Perfect Writer is easy to use, we mean it. That's why we're not afraid to offer a money back guarantee.

You see, Perfect Writer was designed to be simple. In fact, you don't even have to read our crystal-clear manual in order to use Perfect Writer. The pictures, diagrams and illustrations tell you all you need to know, so you're off and running in no time.

There are no difficult and awkward procedures and commands to learn. We guarantee it.

ORDER NOW. WHAT HAVE YOU GOT TO LOSE?

If you'd like to own the most sophisticated word processor on the market today at the current, introductory price of just $289.00, we invite you to order now.

Perfect Writer is completely guaranteed, so there's no risk of any kind. Why not see for yourself what this incredible new software breakthrough can do for you?

TO ORDER CALL TOLL FREE: 1-800-221-2486
IN NEW YORK CALL: 1-212-619-4000

WRITE: COMPUTER SERVICES CORPORATION OF AMERICA
71 MURRAY STREET
NEW YORK, NEW YORK 10007

Yes. I want to see for myself if Perfect Writer is as great as you say it is. So please send it to me, subject to the guarantee printed on this page.

☐ Perfect Writer and Manual $289.00

I also want to order, subject to the same guarantee:

☐ Perfect Speller $189.00 ☐ Perfect Sort $189.00
☐ Perfect Mailer $189.00

Name ____________________

Address ____________________

City, State, Zip ____________________

☐ MasterCard MasterCard ☐ American Express

☐ Visa VISA

Number ____________________ Expires __________

TRY THESE OTHER PERFECT SOFTWARE PRODUCTS
(THE SAME MONEY BACK GUARANTEE APPLIES).

Perfect Speller™: The best automatic proofreader available! Perfect Speller uses its 20,000 word dictionary to proofread your document and catch all your spelling mistakes. (Dictionary size limited only by your system's storage capacity.) You can add words to the dictionary, build separate dictionaries, change the dictionary and more. $189.00 guaranteed.

Perfect Mailer™: It's more than a mail merging or form maintenance system! Perfect Mailer can organize and sort your correspondence, develop sophisticated mailing systems, and keep track of your correspondence. Don't miss this remarkable new software innovation! $189.00 guaranteed.

Perfect Sort™: Here's a sort merge extract program that operates on all kinds of data! Records can be fixed or variable, with a variety of variable length fields. Files can be examined using comparison and logical operators. Want speed and efficiency? You've got to have Perfect Sort! $189.00 guaranteed.

LETTERS TO THE EDITOR

More On CP/M's 'Autoload' Feature

Dear Editor:

Your magazine is a welcome addition to my reading file. After reading the feature concerning "Autoload" by Kelly Smith in the Jan/Feb '82 issue of *Microsystems*, I have the following comments to offer others who may want to use that feature:

1. COM files, whether developed by Basic, Fortran, Cobol, or other languages, may be substituted instead of "MBASIC STARTREK." A good rule to follow is to use "11" or '0BH' as the length of the name of the file to be automatically loaded with the last three characters being "COM" and the first eight characters being the filename. If necessary, pad the filename with trailing blanks in order to make the length equal to eleven characters.

2. Before utilizing the autoload feature, one should insure that the capability to undo it exists, unless one is just going to rewrite the entire disk. One easy way to remove the name of the file to be loaded automatically is to copy the operating system by using SYSGEN22, a utility supplied by Digital Research with CP/M. This will remove the name of the file to be loaded, but will not disturb the directory or the rest of the diskette.

Surry P. Everett
Seabrook, MD

Hints For Whitesmiths C Compiler Users

Dear Editor:

Your recent articles describing the various "C" compilers were interesting and informative. As a former advocate of assembly language programming and now a user of Whitesmiths "C", I have developed a considerable amount of enthusiasm for the language.

Your comments on the Whitesmiths "C" compiler are enjoyable. However, there are two problems with the compiler that your readers should be aware of:

1. During the parser and sometimes the code generator passes (P1 and P2), the compiler gives a useless diagnostic—or none at all—and dies if too many external references are used in the source program.

2. The best method of solving the above problem is to compile each function as a separate module and combine them at a link time. However, the linker limits its command line arguments to 128 bytes, allowing few modules to be specified. It makes some programs impossible to link.

A good solution to problem two is to use the PLINK-II linker by Phoenix Software Associates (sold by Lifeboat). It does not have the command line restrictions, operates four to five times faster than the Whitesmiths linker, and provides good memory maps.

Also, there is supposed to be an improved version of Whitesmiths "C" available for the Z-80 only. It is authored under contract with Whitesmiths by CSSN of Boston, MA. The authors claim that it produces twenty percent less code than the 8080 version.

In future articles, I would like to see some evaluations of the new "C" compilers that are now coming on the market: Aztec, Infosoft, Supersoft, and Telecon.

William D. Briggs
Poway, CA

Note: Reviews of the Supersoft and Infosoft C compilers are now in the work. —Editor.

Short Program to Remove Embedded Characters

Dear Editor:

As a Ham Radio Operator I mostly operate via 2-Meter radioteletype (RTTY). I particularly enjoy sending, receiving and collecting the wide variety of RTTY pictures (pix) created by talented people and relayed all over the world. Since my North Star Horizon computer and my TRS-80 based RTTY station act as "printer" peripherals for each other, it is a relatively simple procedure to have the pix received on the TRS-80, sent to the disk on the North Star and vice versa. However, sometimes the pix have hidden problems that create havoc on the computer.

Many of the pix have imbedded characters that interface with normal operations of the software. A 7FH character acts as a lead-in code to the Hazeltine terminal and distorts the display. An 8AH acts as a line feed and also disrupts the display. 00H is annoying and wasteful of memory and disk space. None of the above characters can be removed easily by such software as is Wordstar. Manual removal is tedious.

After much thought and poking around in the system software, I stumbled across the PIP feature that will optionally filter all formfeeds (0CH) from a file as it is copied. I started searching in PIP.COM (version 1.8) with DDT for the sequence "CPI 0C". I found three instances of this sequence, among them at locations E53H-E54H and E63H-E64H. By experimentation I discovered that if I replaced the 0CH at locations E54H and E64H with the unwanted code to be removed and saved the modified version, I could use the new versions to remove the codes. For example, if I needed to remove several 7FH characters and were using PIP version 1.8 the method used is shown:

```
A>DDT PIP.COM
*FE54,E45,7F   } unwanted code to be removed
*FE64,E64,7F   } unwanted code to be removed
*G0
A>SAVE 29 PIP7F.COM   } creating a new
                        PIP version as
                        PIP7F.COM
```

Now I have a version to remove 7FH (or 00H, 0AH, 0DH, 0CH or any other character) by just typing:

```
A>PIP7F filename.ext=filename.ext[F]
```

This may be pretty basic to some CP/M users, but if you ever find yourself with a text file that has imbedded characters your word processor can't handle, it will certainly save you a lot of time. Incidentally, the PIP version number is found in the beginning part of the PIP as seen under DDT; it doesn't appear on the display as PIP is executed.

Nils R. Olson, N7BCV
Spanaway, WA

by Sol Libes

EDITOR'S PAGE

In the last issue of *Microsystems* I described the wealth of CP/M-based software that is available in the public domain via the CPMUG and SIG/M user groups, at very low cost. In fact most of this software, if not all of it, is in many instances available free of charge (if you do not count the cost of a phone call).

In addition to this software being available from the groups directly and from many local user groups (listed in the last issue) this software is available directly over the telephone line. There are a large number of computer systems operated as "remote" CP/M systems. They usually refer to themselves as either RCPM (Remote CP/M) or RBBS (Remote Bulletin Board System). These systems are operated mostly by individuals who donate their time, effort and their systems to the distribution of public domain CP/M software.

These systems operate primarily as bulletin board systems. Some cater to specific interests (e.g., C language, technical support, etc.). Some serve as a means for micro users in a local area to stay in touch.

In addition to their bulletin board functions, these systems all have facilities for uploading and downloading files. Many of these systems maintain several megabytes of files on-line always available to callers. To access these files and down-load them the caller just calls into the system (rarely is a password required) and follows the procedure that allows him to use the system as a standard CP/M system. A menu is usually given to guide the user.

Once the caller is into the CP/M system he can examine the directory of each disk on the system. To transfer a file, the user must use a transmission protocol that has become a standard on these systems. The protocol was created by Ward Christensen when he and Randy Suess created the first Bulletin Board system to go into operation. The protocol transmits files in 128 byte blocks, with a checksum at the end of the block.

The receiving system checks for errors, and if any are found sends a code back to the transmitting system to retransmit the block. This protocol is part of the MODEM program written by Ward and placed in the public domain via the CPMUG library. Subsequent versions, with enhancements, will also be found in the SIG/M library.

The RCPM/RBBS system has a program called XMODEM which the caller executes to put the system into the file transmit/receive mode of operation. Files can then be transferred between the two systems.

In addition to the RBBS and RCPM systems, file access facilities are available on the COMPUSERVE timesharing system. Although not free, it does provide another means for obtaining much public domain software. This system is part of COMPUSERVE's MICRONET service and is operated by three volunteers (see listing). It also includes a very active CP/M bulletin board. What is particularly interesting about the bulletin board system is that it includes technical representatives from MicroPro, Microsoft, Magnolia Microsystems, Tandy, and several other software and hardware suppliers. Users of the bulletin board can send messages directly to these companies and receive help directly. Not only that, one can read the messages going back and forth between these people....... most interesting! To access the CP/M bulletin board on MicroNet enter (at the command level prompt) "GO CIS-28" and then "R SIGS(CP-MIG)."

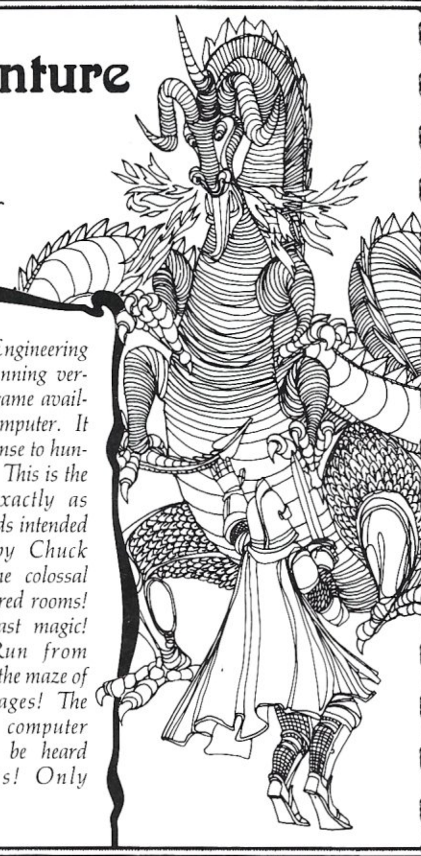
The following list is a highly condensed version of a list of RBBS systems I downloaded from the RBBS system in my local area.

Othello & Gomoku

C.D.E. offers new computerized versions of these two ancient games of tactical skill. **GOMOKU** is played on a 15 by 15 board, and the first player to get five in a row wins. Patterns of play are endless. Complete I/O features include game display, take back, autoplay, referee, set-up and problem modes. **OTHELLO** is played on an 8 by 8 board. Play proceeds by bracketing and "flipping" your opponent's men, and the player with the most squares at the end of the game is the winner. Both games have nine levels of play and are available for virtually any Z-80 based computer, in most CP/M formats or North Star DOS. Also available for TRS-80 Model 1 (running TRS DOS) and Model 2 (running CP/M). Games are available individually at $24.95 each, or $39.95 for both.

Adventure

The classic computer game



California Digital Engineering offers the fastest running version of this classic game available for a microcomputer. It returns instant response to hundreds of commands. This is the complete game, exactly as Crowther and Woods intended it, implemented by Chuck Crayne. Explore the colossal caves—over a hundred rooms! Steal treasures! Cast magic! Kill dwarves! Run from dragons! Get lost in the maze of twisty little passages! The Rubik's Cube of computer games. You won't be heard from for months! Only $35.00 in CP/M.

MAGIC TYPEWRITER™ is C.D.E.'s unique, all-purpose word processing and database management tool. It eliminates the need for separate programs for separate tasks. It offers not only a sophisticated text formatter, but a fast-sort, a mailing list program, and other special purpose features. You may examine any file on the disk without disturbing the file on the screen. Available in CP/M and North Star DOS at a cost of only $175.00.

C.D.E. HAS DISKS

Our disks are certified for two years, and include hard holes. Sold only in boxes of ten.

5¼" double density (Specify soft, 10 or 16 sectors)	
Single sided	$25.95
Double sided	30.95
8" single density	28.95
8" double density	33.95

Sturdy, interlocking DISK BANK cases for safe storage or transporting of disks stack vertically or horizontally to build a useful system to your specifications. DISK BANK holding ten 5¼" disks is $5.95, and for ten 8" disks, $6.95.

Bowling Secretary

A unique program to keep complete league results, compute handicaps, and keep other statistics in order. Only $55 in North Star Basic.

All disks are in the CP/M format except where North Star DOS is indicated. Please let us know whether you require 5¼" or 8" disk, single or double density, single or double sided, soft, 10 or 16 sectors. North Star users indicate CP/M or DOS format. Please include $2 postage and handling with each order. For our complete brochure, write or call (213) 661-2031.

VISA® MasterCard

CDE CALIFORNIA DIGITAL ENGINEERING T.M.
P.O. BOX 526 ★ HOLLYWOOD, CA 90028

Editor's Page, continued...

Remote CP/M Software Exchange Systems

A list of Remote CP/M Software Exchange Systems using XMODEM for program transfers. Operators of new RCPM systems should send information about their systems to Ben Bronson [Hyde Park RCPM (312) 955-4493] or to Kelly Smith [CP/M-Net (805) 527-9321]. Revised October 15, 1981.

EXPLANATION OF CODES USED

- BBS= Bulletin Board System
- B1= 1 Baud rate available (300)
- B2= 2 Baud rates available (110,300)
- B3= 3 Baud rates available (110,300,1200)
- B4= 4 Baud rates available (110,300,450,600)
- B5= 5 Baud rates available (110,300,450,600,710)
- CB= Call Back**
- NALDS= No Alternate Long Distance Service*
- RCPM= Remote CP/M system
- S= Sprint*
- M= MCI*
- I= ITT*

NORTHEAST

Mississauga Ontario RCPM, (Toronto) 416-826-5394, Jud Newell. NALDS;B5,10Mb hard disk; 24 hrs; Sysop now has secondary system (with 2nd PMMI modem) integrated with main system so special arrangements can be made for extensive downloading. All vols of CPM/UG and SIG/M software available on request. Interest in new & new versions of s'ware.

Mississauga Ontario HUG-RCPM, 416-273-3011, Toronto Heath UG. 1800-0900 wkdys, 24 hrs wknds; B5; NALDS;2+Mb files/5 drvs. Sysop plans 1Mb+ of HDOS s'ware & 1Mb+ CP/M software on line.

SuperBrain RCPM, 617-862-0781, Paul Kelly. 1900-0700 wkdys,24 hrs wknds; B3; S,I,M; 300K files on-line. [Lexington, Boston,MA area] (interest in Superbrain-adapted CP/M programs)

Long Island NY RBBS, 516-698-8619, Tim Nicholas, CB, B3, 24 hrs; S,M; 1Mb files/2 drvs. Soon with 2 lines & modems, one half-duplex at 1200 baud)

Valley Stream NY RBBS/RCPM, 516-791-5041, Mike Schiller; 24hrs; B2; S,M, 300k files/2 drvs. May be running 212, 1200-baud modem.

Johnson City, NY SJBBS, 607-797-6416, Charles ---; Eves; etc. B1; 2Mb files/2 drvs, [Upstate New York]

Bearsville Town NY SJBBS, 914-679-6559, Hank Szyszka; B5; NALDS; 2MB files/4 drvs, [Upstate NY]. Installing MP/M. All CPMUG programs available by request.

Brewster NY RBBS, 914-279-5693, Paul Bosshold/Carl Erhorn;5pm-10pm, CB 10am-5pm, up 24hrs wknds; B4; NALDS; 500K files/1 drv. [Downstate New York] (S-100 based. General CP/M software)

Rochester RBBS, 716-223-1100, Arnie McGall; 24hrs; B2; S,M,I; 1.8Mb files/3 drie. [Upstate New York] S-100 based. General CP/M software. RBBS/RCPM system coexists with separate passworded message system called Datastar, which can be entered from CP/M but runs on separate computer. 600 baud expected soon.

EAST CENTRAL

Cranford NJ RBBS/RCPM System, 201-272-1874, Bruce Ratoff, 24hrs, B4; S,M; 2-3Mb files/ 3 drvs. General CP/M software. Amateur Computer Group of NJ & SIG/M RBBS

Allentown Pa RBBS/RCPM System, 215-398-3937, Bill Earnest, 24hrs; B5; S,I.; 4.25Mb files/ hard disk(=4 logical disks). General CP/M software. Lehigh Valley Computer Club RBBS.

Baltimore Md Micro-Mail, 301-655-0393, Rod Hart; CB, Days/Eves until 2200; B5; S,I,M.; 1Mb files/2 drvs. General CP/M software; interest in Ham programs & modem s'ware in PASCAL & C)

Baltimore Md Prodigy Systems RBBS, 301-337-8825. NCB,24 hrs, B1, I,S,M. Down as of 10/16/81. 500K files/2 drvs.

Bel Air Md Nuclear RBBS/RCPM, 301-879-7841, Bob Loesch. NCB, 24 hrs; B2; M,I,S; 1.2Mb files/2 drvs. Down as of 09/21/81.

Grafton Va RBBS, 804-898-7493, Dave Holmes; 24 hrs; B1; NALDS; 200K files/2 drvs. (Tidewater Va.) CP/M, TRS-80 & Apple software; plans dual system (on one line) with LNW-80 & CP/M computer.

MIDWEST

Columbus Oh CBBS, 614-268-2227 [268-CBBS], Ben Miller; 24 hrs; B5; S,I,(M?), 300K files/3 drvs, running MP/M on a Tarbell SD controller; occasional slow response means sysop also using system; interest in BDS-C programs.

Newark Oh RBBS/RCPM, 614-366-3252, Bo McCormick; 24 hrs; B1; NALDS; 500K files/2 drvs; Homebrew S-100. Interest in general software, offbeat graphics; other software may be requested from on-line master catalog.

Westland (Detroit) Mi RBBS/RCPM, 313-729-1905, Ron Fowler; CB; 24 hrsg; B4; S,M,I; 1.4Mb files/2 drvs. Emphasis very recent sftwre.

Detroit Mi, Technical CBBS, 313-846-6127, Dave Hardy; 24hrs; B4; I,S,M; 3Mb files/3 drvs. Emphasis very recent releases. RCPM sysops desiring access to passworded RCPM Clearing House system should leave msg .

Royal Oak (Detroit) Mi,CP/M, 313-759-6569, Keith Petersen; CB, 24hrs; B3; I,S,M; 600K on 2 drvs + 10Mb hard disk (=2 logical drvs). Emphasis on new programs & recent updates of standard progs. 1200-baud 212-type modem available but not regularly on line, use CHAT or leave msg if you want PMMI switched out and 212 switched in.

MINICBBS/Sorcerer's Apprentice Group, 313-535-9186, Bob Hageman; CB; 24hr; B4; I,S,M; 500K/2 drvs. [Michigan] Running on an Exidy Sorcerer. Needs password, "SORCERER". Interest in adapting CP/M software and assorted hardware to Sorcerer systems.

Southfield Mi RBBS/RCPM, 313-559-5326, Howard Booker; NCB, 24 hrs; 300/450 baud; I,S,M; 1Mb/2 drvs. Interest file directories/ catalogs of other RCPM systems & general s'ware)

Valparaiso In, Dick Hill's RBBS, 219-465-1056; 1900-2200 wkdys, 24hrs wkends; B4;NALDS; 4Mb files/4 drvs. S-100 based. General CP/M software.

Chicago Il, Calamity Cliffs Computer Center, 312-234-9257; 1400-0200 daily; B3; I,S,M; hard disk & 2 floppies. Many of CPMUG & SIG/M programs available by request.

Chicago Il, NEI RCPM System, 312-949-6189, Chuck Witbeck; 1800-0100 wkdys, 1200-0100 wknds; B4; M,S,I; 2Mb files/2 drvs. Emphasis on communications ;programs, including versions adapted to non-standard CP/M systems.

Hyde Park Il (Chicago) RCPM/RBBS, 312-955-4493, Ben Bronson; 0100-1700 daily; B5; S,I,M; 2Mb files/2 drvs. Interest hard- & software reviews, C progs, and very recent releases of std progs.

Chicago Il RAPM (Remote Apple CP/M), 312-384-4762, David Moritz; 24 hrs/7 days (sporadic); 300/450 baud; S,I,M; 250K files/2 drvs. Interest in telecom & other utilities for Apple/Softcard CP/M. 450 baud achieved using modified Hayes modem. Sysop may soon substitute a (TEI) S-100 system for the Apple.

Logan Square (Chicago Il) RCPM, 312-252-2136, Earl Bockenfeld; 0100-1900 wkdys, irreg on wkends; B5; S,I,M; 1Mb files/ 2 drvs. Interest recent releases & developing on-line data-bases, with daily change of software on B drive.

Chicago Il HUG-CBBS, 312-671-4992, Paul Mayer & Dave Leonard; 2300-1900, 7 dys; B1; S,I,M; 2Mb files/ 2 drvs. H-89 based, operated for Heath-Zenith UG with interest in H19 & H89 adapted CP/M progs.

Editor's Page, continued...

Palatine (Chicago Il) RCPM, 312-359-8080, Tim Cannon; CB; 1800-2400 wkdys, irreg on wknds; 300/1200 baud; S,M,I; 850K files/4 drvs. 212A 1200 baud modem.

Milwaukee Wi, Rick Martinek's System, 414-774-2683; Eves & wknds; B4; I,S,M; 1200K files/2 drvs.

SOUTH

Fort Mill SC, RIBBS, 803-547-6576, Bill Taylor; 24 hrs; 300/1200 baud; NALDS; 3Mb files/3 drvs. Heath/Zenith-based with 212 modem. Ham stuff, general s'ware, & on-line games.

Louisville Ky, RBBS/RCPM, 502-245-7811, Mike Jung; 0900-2100 wkdys, 24 hrs wknds; B1; S,M; 2.5Mb files/5 drvs. Heath/Zenith-based. Emphasis on BASIC software & some HDOS stuff.

Huntsville Al, NACS/UAH RBBS/RCPM, 205-895-6749, Don Wilkes; CB; 24 hrs; B4; NALDS; 700K files/4 drvs. No Ala Computer Soc @ U of Ala; general CP/M software.

CALIFORNIA

Bakersfield CA, CP/M-Net (tm), (805) 527-9321, Kelly Smith; 1900-2300 Mon-Fri, 1900 Fri-0700 Mon; B5; NALDS; 20Mb files/2 hard disks (=8 logical disks). System includes SIG/M Vol 1-10 =E:, SIG/M Vol 11-20 =F:, SIG/M Vol 21-25 = G:; XMODEM 'DISKMENU.DOC' for entire system directory (over 2100 files available!).

Pasadena (Los Angeles area) CA, CBBS, 213-799-1632, Dick Mead; 24hrs; B5; I,S,M; 1.5Mb files/2 floppies & 8.3Mb hard disk.

Torrance (Los Angeles area) CA, RCPM, 213-549-9296, Dan Lopez/Alex Valdez; CB?; 1900-2300; B1; I,S,M; 500K files/2 drvs?. RBBS & other RCPM system progs available.

Palos Verdes CA, G.F.R.N.Data Exchange [RBBS], 213-541-2503, Skip Hansen; 24 hrs; 300/1200 baud; S,M,I; 2.4Mb files/2 drvs. Std CP/M s'ware. Interest in ham radio-related progs. Soon (with MP/M) will also be reachable thru 450 mhz radio.

San Diego CA,RCPM, 714-271-5615, Brian Kantor; 24 hrs; 300/1200 baud; I,S,M; 2.4Mb files/ 2 drvs; S-100 based with Auto-Cat modem. General CP/M s'ware with special interest in ham radio.

Siliconia (San Jose) CA, RBBS/RCPM, 408-287-5900, Paul Traina; Wkdys 17:45-08:00, wknds 24hrs; B1; S,M,I; 2.4Mb files/3 drvs. S-100 [Godbout] based. Interest in PASCAL MT+ programs.

RBBS of Marin County (San Francisco area), 415-383-0473, Jim Ayers; Eves & nites wkdys, 24hrs wknd; B4; S,I,M; 1Mb files/2 drvs; S-100 [IMSAI] based; 24hr operation expected soon.

Larkspur (San Francisco area) CA, RBBS/RCPM, 415-461-7726, Jim C.; 24hrs; B5; S,I,M; 2+Mb/2 drvs. TRS-80/Omikron formerly used replaced by Godbout S-100 with PMMI.

Sacramento CA, CBBS/RCPM, 916-483-8718, Sacramento Microcomputer UG; 24hrs; B5; S; 700K+ Files/2 drvs (expansion planned to 1.5M); Joe Bergin, Don Bozarth, John Moorhead, & Bob Ress Sysops. S-100 based; interest in CP/M; disks change bi-monthly.

NORTHWEST

Vancouver BC (Canada) CBBS, 604-???-????, Steve Vinokouroff; 24hrs; B1; NALDS; 2Mb files/ 2 drvs. The system will be down for 6 months starting 10/15. When it comes up again next year it will have a new telno & PMMI modem.

Vancouver BC (Canada) Terry O'Brien's RCPM System, 604-584-2543

Olympia Wa, Yelm RBBS & CP/M, 206-458-3086, Dave Stanhope; CB; 24 hrs; NALDS; 500K files/2 drvs.

Portland Or, Chuck Forsberg's RCPM, 503-621-3193; 24hrs?; B3; NALDS?; 2Mb(?) files/2 drvs. Heath/Magnolia-based, 212a modem. Interest in C-language s'ware.

GENERAL NORTH AMERICA

CP-MIG on CompuServe MicroNet; type 'RSIGS (CP-MIG)';B1; Sysops Dave Kozin, Tom Jorgenson & Charlie Strom are arranging to have MN carry much of new CPMUG and SIG/M software, plus newsletter and CP/M-oriented CBBS.

OVERSEAS

PERTH Western Australia Remote Computer/RBBS, Australian local; 09 457 6059, International; 619 457 6059. Trevor Marshall. Available most daylight hours & evenings. Manual connection only, requires CCITT 300 Baud modem in ANSWER mode for access. Running IOS (CP/M compatible), 64K Z80, 5Mhz system; 2Mb/2 drvs with 48K Cache buffer. All CPMUG and SIG/M volumes available by RBBS request. 1200 Baud Bell 202 will be available in 1 month.

Perth Western Austraila, Paul Kelly's Remote Computer/ RBBS, Australian local; 09 459 3787; Available most evenings. Manual connection only, requires CCITT 300 Baud modem in ANSWER or ORIGINATE mode for access. Running IOS (CP/M compatible), 64K Z80, 5Mhz system; 2MB/2 drvs with 48K Cache buffer. All CPMUG and SIG/M volumes available by RBBS request 1200 Baud, Bell 202 will be available in 1 month.

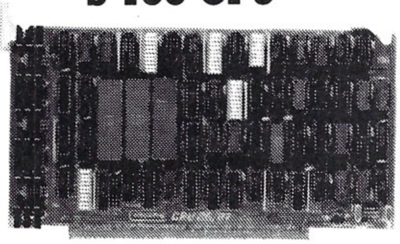
* Alternative long-distance service should be considered when planning transfer of long programs. Charges on SPRINT, ITT/CITY-CALL and MCI are 50-60% of Bell's regular long-distance rates.

** Call-back systems; computer and real people share same telephone line. To contact people, dial & let phone ring until answered. To contact computer: dial, let phone ring once, hang up & re-dial.

All times listed are local time.

PRIORITY ONE ELECTRONICS

S-100 CPU



DUAL PROCESSOR 8086 - 8087 COMPUPRO
Math co-processor

PART NO.	DESCRIPTION	LIST PRICE	OUR PRICE
SFGBT186A	A&T 8086 Only	$695.00	$625.00
SFGBT186C	CSC 8086 Only	$850.00	$765.00
SFGBT8087	8087 Chip		CALL

CPU-Z - GODBOUT
2/4 MHZ Z80 CPU 24 Bit Addressing

SFGBT 160A	A & T	$295.00	$280.00
SFGBT 160C	CSC 3-6 MHZ	$395.00	$375.00

DUAL PROCESSOR 8085-8088 - COMPUPRO
6 or 8 MZ Provides true 16 Bit Power with a standard 8 bit S-100 bus.

SFGBT 1612A	A & T 6 MHZ	$425.00	$399.00
SFGBT 1612C	CSC 6/8 MHZ	$525.00	$498.00

MPM 816™ — COMPUPRO
Runs CPM 80™, CPM 86™, And MPM86™ AT THE SAME TIME! 8 AND/OR 16 BIT TOGETHER! WOW!

SFGBTMPM816	Software		$995.00

See Compupro AD in May **BYTE** *for details*

SOLID STATE DISK DRIVE, 3500% FASTER!
Not Really, But the Next Best Thing For Compupro 8085/88 Users. Call for Details on M-Drive.

SFGBT MD 128K		$1,550.00
SFGBT MD 256K		$3,000.00

2810 Z80 CPU-CA. COMP. SYST.
2/4 MHZ Z80A CPU with RS232C Serial I/O Port complete with Monitor PROM for 2422 Disk Controller

SFCCS 2810A	A & T	$350.00	$280.00

CB2 Z80 CPU - S.S.M.
2/4 MHZ will accept 2716, or 2732, or RAM

SFSSMCB2K	Kit		$260.00
SFSSMCB2A	A & T	$344.00	$310.00
SFSSMZ80M	SSMZ80 Monitor		$89.00

S-100 I/O BOARDS

SYSTEM SUPPORT 1 - GODBOUT
Serial port (software prog baud), 4K EPROM OR RAM provision, 15 levels of interrupt, real time clock, optional math processor

PART NO.	DESCRIPTION	LIST PRICE	OUR PRICE
SFGBT162A	Assembled & Tested	$399.00	$360.00
SFGBT162C	CSC	$495.00	$460.00
SFGBT8231	Math Chip		$195.00
SFGBT8232	Math Chip		$195.00
SFGBT162AM1	A&T with 8231 Math Chip		$555.00
SFGBT162CM1	CSC with 8231 Math Chip		$655.00
SFGBT162AM2	A&T with 8232 Math Chip		$555.00
SFGBT162CM2	CSC with 8232 Math Chip		$655.00

MPX CHANNEL BOARD - COMPUPRO
I/O Multiplexer, using 8085A-2 CPU on board

	With 4K RAM		
SFGBT166A4	A & T	$495.00	$445.00
SFGBT166C4	CSC	$595.00	$535.00
	With 16K RAM		
SFGBT166A16	A & T	$649.00	$585.00
SFGBT166C16	CSC	$749.00	$675.00

INTERFACER I - COMPUPRO
Two Serial I/O

SFGBT133A	A & T	$249.00	$219.00
SFGBT133C	CSC	$324.00	$298.00

INTERFACER II - COMPUPRO
Three parallel, one serial I/O board

SFGBT150A	A & T	$249.00	$219.00
SFGBT150C	CSC	$324.00	$289.00

INTERFACER III - COMPUPRO
Eight channel multi-use serial I/O board

SFGBT1748A	A & T	$699.00	$629.00
SFGBT1748C	CSC 200 hr. 8 Port	$849.00	$750.00
SFGBT1745A	A & T	$599.00	$559.00
SFGBT1745C	CSC 200 hr. 5 Port	$699.00	$629.00

MULTI I/O - MORROW DESIGNS
Three Serial, Two parallel

SFMDSMB3200	A & T	$359.00	$329.00

SWITCHBOARD - MORROW DESIGNS
Two serial I/O, four parallel I/O, one status port, one strobe port

SFMDSSB2411		$299.00	$269.00

I/O4 - SSM
Two serial I/O, two parallel I/O

SFSSMI04K	Kit		$210.00
SFSSMI04A	A & T	$290.00	$260.00

I/O 5 - SSM
2 Serial, 3 Parallel including 1 Centronics

SFSSMI051	A & T	$329.00	$309.00

I/O 8 - SSM
8 Port Serial I/O with Timer

SFSSMI08A	A & T	$550.00	$495.00

2710 4 PORT SERIAL - CCS
4 Full handshaking RS232 ports and optional 2K ROM

SFCCS271001	A & T	$360.00	$310.00

2719 2 SERIAL & 2 PARALLEL - CCS
2 RS232 C ports, 2 8 bit parallel ports, & optional 2K ROM

SFCCS271901	A & T	$360.00	$325.00

2720 4 PORT PARALLEL - CCS
4 8 bit parallel ports and optional 2K ROM

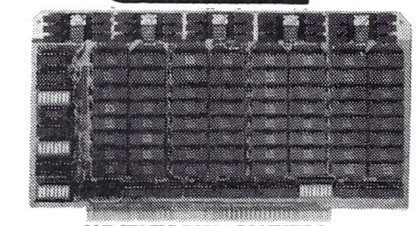
SFCCS272001	A & T	$250.00	$225.00

S-100 10 MHZ STATIC RAM

NEW LOW PRICES!

RAM 20 - 32K

CompuPro™



32K STATIC RAM - COMPUPRO
RAM 20 10 MHZ, 4K byte block disable, bank select or 24 bit addressing available 8, 16, 24 or 32K

PART NO.	DESCRIPTION	LIST PRICE	OUR PRICE
SFGBT164AA8	8K A&T	$210.00	$190.00
SFGBT164AC8	8K CSC	$280.00	$260.00
SFGBT164AA16	16K A&T	$285.00	$260.00
SFGBT164AC16	16K CSC	$355.00	$325.00
SFGBT164AA24	24K A&T	$355.00	$325.00
SFGBT164AC24	24K CSC	$425.00	$385.00
SFGBT164AA32	32K A&T	$425.00	$385.00
SFGBT164AC32	32K CSC	$495.00	$450.00

CMOS STATIC RAM
For a complete analysis of the advantages of CMOS memory, see the "Product Description" on page 416 of the January Issue of **BYTE.**

64K CMOS STATIC RAM - COMPUPRO
RAM 17, 10 MHZ, 2 Watt, DMA Compatable 24 Bit Addressing

SFGBT175A64	64K A&T	$599.00	$550.00
SFGBT175C64	64K CSC 200hr.	$699.00	$650.00

NEW! 32K x 16 BIT CMOS STATIC RAM - COMPUPRO
8 and/or 16 Bit
816 RAM 16 10 MHZ, 32K x 16 or 64K x 8 IEEE/696 16 BIT 2 Watt, 24 Bit Addressing

SFGBT180A	64K A&T	$650.00	$599.00
SFGBT180C	64K CSC	$750.00	$699.00

NEW! 128K NMOS STATIC RAM - COMPUPRO
RAM 21 15MHz. 128K X 8 OR 64K x 16 IEEE/696 8 or 16 Bit 1.2 Amps 24 Bit Addressing

SFGBT190A	128K A&T	$1695.00	$1610.00
SFGBT190C	128K CSC	$1895.00	$1795.00

S-100 PROM

PBI PROM PROGRAMMER - SSM
Programs 2708 or 2716's, operates as a 4K/8K EPROM BOARD AS WELL.

SFSSMPB1K	Kit		$179.00
SFSSMPB1A	A & T	$265.00	$220.00

ECONOROM 2708 - COMPUPRO
16K x 8 EPROM Board using 2708, Power on jump to any 256 byte

SFGBT125A	A & T	$135.00	$120.00
SFGBT125C	CSC	$195.00	$175.00

MB8A - SSM
1K/16K 2708 EPROM board, disable in 1K increments

SFSSMMB8AK	Kit		$114.00
SFSSMMB8AA	A & T	$179.00	$159.00

S-100 VIDEO BOARDS

SPECTRUM - COMPUPRO
Color Graphics board with Parallel I/O

SFGBT144A	A & T	$399.00	$349.00
SFGBT144C	CSC	$449.00	$399.00
SFGBT2D	Sublogic Universal Graphics Interpreter Software		$35.00

VB - 3 S.S.M.
80 x 25 or 50 character video display Memory Mapped, Parallel Keyboard port

SFSSMVB3K24	80 x 24 Kit		$425.00
SFSSMVB3A24	80 x 24 A&T	$499.00	$440.00
SFSSMVB3UP	80 x 50 Line Upgrade		$ 39.00

VB2-S.S.M.
I/O Mapped Video Board, with Parallel Keyboard port 64 x 16

SFSSMVB2K	Kit		$199.00
SFSSMVB2A	A & T	$269.00	$229.00

VBBB - S.S.M
Memory Mapped Video Board 64 x 16 character display or 64 x 16 graphics display

SFSSMVB1K	Kit		$179.00
SFSSMVB1A	A & T	$242.00	$220.00

S-100 CLOCK CALENDAR

SFQTCCCSA	Assembled & Tested	$150.00

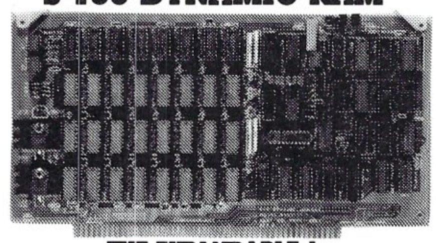
As seen in March Kilobaud Magazine

S-100 MOTHERBOARDS - COMPUPRO

Active termination, 6-12-20 slot

SFGBT153A	A&T 6 slot, 2 lbs	$140.00	$126.00
SFGBT153C	CSC 6 slot, 2 lbs.	$190.00	$175.00
SFGBT154A	A&T 12 slot, 3 lbs.	$175.00	$155.00
SFGBT154C	CSC 12 slot, 3 lbs.	$240.00	$220.00
SFGBT155A	A&T 20 slot, 4 lbs.	$265.00	$235.00
SFGTB155C	CSC 20 slot, 4 lbs.	$340.00	$310.00

S-100 DYNAMIC RAM



THE EXPANDABLE 1
PRIORITY 1 ELECTRONICS

THE EXPANDABLE 1™ 64K Dynamic Ram board provides your S-100 system with 64K of reliable, high-speed dynamic RAM. Compatable with most of the major S-100 systems on the market, including those with front panels, it supports DMA operations and requires no Wait states with current microprocessors.
• User expandable from 16 to 64K • Supports DMA • Designed to IEEE proposed S-100 bus standards • 2 or 4 MHz operation • Operates with either an 8080 or Z-80 based S-100 system, providing processor-transparent refreshes with both • Supports IMSAI-type front panels • Jumper-selectable Phantom input • Uses Popular 4116 RAMS • All ICs in sockets • Any 16K block can be made bank-independent • Fully buffered address and data lines • Fail-safe refresh circuitry for extended Wait states • Board configuration with reliable, easy-to-configure Berg jumpers

SFPRIEXP116	16K Assembled & Tested	$299.00
SFPRIEXP132	32K Assembled & Tested	$339.00
SFPRIEXP148	48K Assembled & Tested	$379.00
SFPRIEXP164	64K Assembled & Tested	$409.00

S-100 DISK CONTROLLERS

2422A - CA. COMP. SYST.
I/O Mapped, controls 8", single or double density A&T with CPM 2.2 8" S.D.

	LIST PRICE	OUR PRICE
SFCCS2422A	$475.00	$375.00

DISK JOCKEY 2D - MORROW
Memory Mapped, controls 8", single or double density, serial I/O

SFMDIDJ2208	A&T with CP/M 2.2	$399.00	$375.00

S-100 DISK SUBSYSTEMS

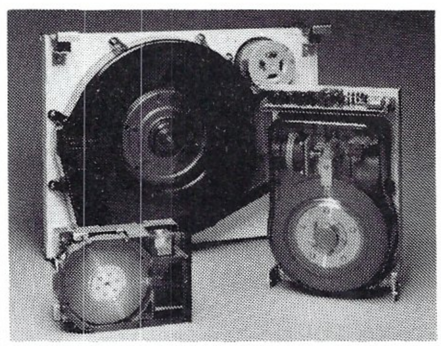
DJ2B DISCUS SINGLE SIDED MORROW
8" DBL Density drives with cabinet, power supply controller, with CP/M 2.2 and Microsoft Basic

SFMDSF1218	Single Drive System	$1095.00	$950.00
SFMDSF1228	Dual Drive System	$1875.00	$1598.00

DJ2B DISCUS DOUBLE SIDED - MORROW
8" DBL Density/sided drives with cabinet Power supply controller, with CP/M 2.2 and Microsoft Basic

SFMDSF2218	Single Drive System	$1395.00	$1250.00
SFMDSF2228	Dual Drive System	$2495.00	$2050.00

S-100 HARD DISK - MORROW

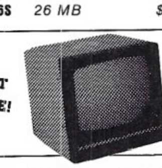


5.25" 5MB, 8" 10 & 20MB, 14" 26MB formatted hard disk complete with cabinet, P.S., Controller, CP/M 2.2 and Microsoft MBASIC 80

		LIST PRICE	SALE PRICE
SFMDSDMAM5	5 MB	$2495.00	$1995.00
SFMDSM10S	10 MB	$3695.00	$2950.00
SFMDSM20S	20 MB	$4795.00	$3825.00
SFMDSM26S	26 MB	$4495.00	$3495.00

WOW!

BEST PRICE!



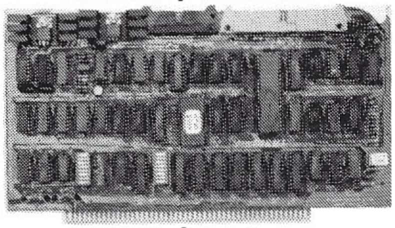
NEW! GREEN 80 x 24

OUTPERFORMS MONITORS OF TWICE THE PRICE!
USI has the competition buckling at their knees! You can have a professional performing Video Monitor for a **fraction of the cost!** *The USI EV2131N is a 12" Green, high resolution data display compatable with 80 x 24 computer formats*

SFUSIEV2131N	24 lbs.	$149.00

PRIORITY ONE ELECTRONICS

COMPUPRO DMA DISK 1 WITH FREE CP/M 2.2 SALE $450.00



SAVE $220.00

Priority 1 Electronics is pleased to offer the COMPUPRO DISK 1 High Performance Disk Controller at our regular low price with CP/M 2.2 and BIOS at no additional cost. That's a savings of $220.00 of the manufacturer's list price.

- Third generation INTEL 8272/NEC 765A LSI floppy disk controller.
- High speed cycle stealing DMA interface for processor independent data transfer between system memory and flexible disk.
- Handles up to four 8 or 5.25 inch floppy disk drives
- Single or double density/single or double sided capability.
- Supports IBM 3740 soft sectored formats.
- 24 bit DMA addressing with data transfer across 64K boundaries for data transfer throughout the 16Mbyte memory map.

PART NO.	DESCRIPTION	LIST PRICE	OUR PRICE
SFPDB171ACPM	A&T w/CP/M 2.2 & BIOS	$670.00	$450.00
SFGBT171C	CSC	$595.00	$555.00
SFGBTCPM80*	CP/M 2.2 for Z80/8085 with manuals & BIOS 8" S/D disk		$175.00
SFGBTOAS8S	Oasis 8 bit single user 8" S/D disk		$500.00
SFGBTOAS8M	Oasis 8 bit multiuser, 8" S/D disk		$850.00

S-100 SYSTEMS
SUPERSIXTEEN — COMPUPRO
HERE IS WHAT EACH PACKAGE INCLUDES:

SFGBT1612A 6 MHz 8085/8088 Dual Processor Board
SFGBT171A High Speed DMA Disk Controller
SFGBT162A System Support 1 Multi-Function Board
SFGBT133A Interfacer 1 Dual Serial I/O
SF128K 10MHz Low Power Static Ram
SFGBTCP/M 86 16 Bit Operating System Ready to Load & Go
Cables and Documentation Three interfacer cables one disk I/O cable, complete documentator for all hardware, and manuals for both CP/M operating systems.
Compu Pro's famous **1 Year** limited warranty.

Now to the best part of all. If purchased separately, these quality components would list for $4,344.00. **BUT SuperSixteen's low package price is an amazing $3,495.00. You save $849.00!** (For boards qualified under the Certified System Component high-reliability program - with extended 2 year warranty, 200 hour burn-in and 6/8 MHz processors - add $600.00 to the package price. Sh. Wt. 15 lbs.

SFPDBGBTSJ	SuperSixteen A&T	$3495.00
SFPDBGBTSK	SuperSixteen CSC	$4095.00

PRINTERS

BEST PRICE!

MICROLINE — OKIDATA WITH FRICTION AND TRACTOR FEED

- BI-DIRECTIONAL - 120 CPS
- 9x9 Matrix (Alphanumeric)
- 6x9 or 12 Matrix for Graphics
- 5,8.3, 10,16 Charactrs p/Inch
- 6 or 8 Lines per Inch
- 80 CPL @ 10 CPI for 82A
- 132 CPL @ 10 CPI for 83A
- Parallel and Serial I/O
- 100 Thru 1200 Baud
- Self Test
- Out of Paper Switch
- Friction or Tractor Feed
- 3" to 14" Top of Form (Switch Selectable)
- 10 Different Character Sets

PART NO.	DESCRIPTION	LIST PRICE	SALE PRICE
SFOKIDAT82AT(26 lbs)	80 CPL @10 CPI	$ 799.00	$539.00
SFOKIDAT83AT (37 lbs)	132 CPL @ 10 CPI	$1195.00	$750.00
SFOKISER2KBF	9600 baud with 2K Serial Buffer upgrade with X-on Y-off		$159.00
SFOKIGRAPH	High Resolution Graphics Prom		$99.00

CALL FOR THE NEW MICROLINE 84

MX80 — EPSON
NEED WE SAY MORE?

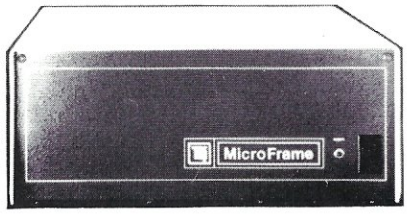
SFEPNMX80	Tractor Feed 17 lbs	$645.00	$450.00
SFEPNMX100	132 Col. Tractor Feed 24 lbs		$725.00

PRINTER INTERFACES - MICROBYTE

SFMBSSEI1	RS232 Serial Conversion for MX80 A & T	$55.00
SFMBSAEI1	Apple Centronics 8 bit parallel interface for Centronics, Epson & OKIDATA printers A & T	$55.00
SFMBSAEC1	Cable for above	$14.95

Printer interfaces & cables sold only with printer purchase

S-100 MAINFRAMES



S-100 MICROFRAME - TEI
110V 60HZ CVT Mainframes, the best money can buy!
12 Slot ±8V 17A±16V @ 2A
22 Slot ±8V @ 30A± 16V @ 4A

PART NO.		LIST PRICE	OUR PRICE 1-9	OUR PRICE 10-24
SFTEIMCS 112	12 Slot Desk	$685.00	$615.00	$570.00
SFTEIMCS 122	22 Slot Desk	$825.00	$760.00	$705.00
SFTEIRM 12	12 Slot Rackmount	$725.00	$720.00	$619.00
SFTEIRM 22	22 Slot Rackmount	$875.00	$850.00	$750.00

Shipping Weight: On 12 Slot Mainframe 45 lbs. On 22 Slot Mainframes 55 lbs.

TEI S-100 FRAMES 3 - 5" DISK CUTOUTS
±8V @ 17±16V @ 1.2A, Internal Cables

			1-9	10-24
SFTEITF12	12 Slot desk	$675.00	$625.00	$580.00
SFTEIRF12	12 Slot Rackmount	$795.00	$715.00	$665.00

Shipping Weight: On 12 Slot Desk 40 lbs. On 12 Slot Rackmount 45 lbs.

DUAL 8" DISK DRIVE CHASSIS - TEI
For Shugart 800/801R or 850/851R with internal power cables provided
+24V @ 1.5A+5V @ 1.0A - 5V @ .25A

			1-9	10-24
SFTEBCFDO	Desk Top	$535.00	$485.00	$455.00
SFTEIRFDO	Rack Mount	$720.00	$670.00	$630.00

S-100 MAINFRAME - COMPUPRO
110V 60HZ CVT Mainframe uses famous 20 slot COMPUPRO Motherboard. 55 lbs.

SFGBTENC20RM	20 Slot Rack Mount	$895.00	$825.00
SFGBTENC20DK	20 Slot Desk Top	$825.00	$760.00

S-100 MAINFRAME - CCS
12-slot motherboard with removable termination card

SFCCS220001	Office Cream	35 lbs	$575.00	$535.00
SFCCS220002	Blue	35 lbs	$575.00	$535.00

SOFTWARE - MICROPRO
All software is supplied on 8" Single Density IBM 3740 CP/M Compatable Diskettes

WORDSTAR
Screen-Oriented, integrated word processing system specifically designed for non-technical personnel
SFMPRWRDSTA1 — $495.00 — **$300.00**

MAIL MERGE WORD STAR OPTION
Powerful file merging tool
SFMPRMLMRGA1 (Requires Word Star 2.1 or later) $250.00 — **$100.00**

SPELLSTAR WORD STAR OPTION
One Step "Proofreader" with compressed 20,000 word dictionary and user-created supplemental dictionaries
SFMPRSPLSTA1 (Requires Word Star 3.0 or later) $250.00 — **$150.00**

SUPERSORT
Sophisticated program that will select and re-arrange variable length information from data files
SFMPRSPRSRA1 — $250.00 — **$150.00**

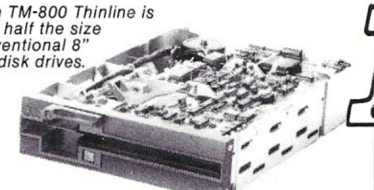
CALC STAR
Sophisticated, easy-to-use, electronic spread sheet and financial planner
SFMPRCLCSTA1 — $295.00 — **$200.00**

DATA STAR
Office-Oriented Data Entry, retrival, and updating system
SFMPRDATSTA1 — $350.00 — **$200.00**

Look in the May BYTE for our 2-Page, Full Color Ad Announcing Our Spring 1982 Engineering Selection Guide

FLOPPY DISC DRIVES

Tandon TM-800 Thinline is exactly half the size of conventional 8" floppy disk drives.



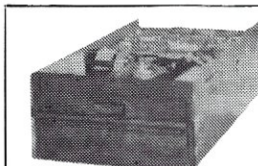
Exactly one-half the height of any other model.
Propietary, high-resolution, read-write heads patented by Tandon
D.C. only operation - no A.C. required
Industry standard interface.
Three millisecond track-to-track access time 9 lbs.

SFTNDTM8481	Single Sided	$425.00	2 or more	$395.00
SFTNDTM8482	Double Sided	$575.00	2 or more	$550.00
SFTNDTM8M	Manual - not included with drive			$ 10.00

801R - SHUGART
Single sided double density most popular 8" drive

SFSHU801R	$425.00 ea or 2 or more (16 lbs) for	$395.00 ea.
SFSHUSA801RM	Manual for 801R drives	$ 10.00

MITSUBISHI ELECTRIC



Better Than QUME! Better Than SHUGART!

8 Inch double-sided, double density

SFMITM289463	Sh. Wt. 16 lbs.	$550.00
SFMITM289463M	Manual	$10.00

2 or more $525.00 each

5¼" DRIVES - TANDON

SFTNDTM1001	Single Sided, 250KB (5 lbs)	$249.00* ea.
	2 or More $220.00	
SFTNDTM1002	Double Sided, 500KB	$325.00 ea.
	2 or More $299.00	
SFTNDTM1003	Single Sided, 500KB	$325.00 ea.
	2 or More $299.00	
SFTNDTM1004	Double Sided, 1000KB	$449.00 ea.
	2 or More $420.00	
SFTNDTM5M	Manual, not included with drive	$ 10.00

*As used in the IBM P.C.

DISK CABINETS
V-100 - VISTA
- Desk or rack mountable • Internal power and data cables
- Drives pull out for easy service and maintenance

SFVISV100 Disk Drive Cabinet (43 lbs) $495.00 **$449.00**

SINGLE 8" - Q.T.
Single 8" cabinet with power supply
SFQTCDDC8 (22 lbs) $249.00 **$225.00**

DUAL 8" - Q.T.
Dual 8' cabinet with power supply
SFQTCDDC88 (24) $395.00 **$349.00**

5" CABINETS - VISTA

SFVIS 9801	Single 5" with P.S.	$75.00
SFVIS 9802	Dual 5" with P.S.	$95.00

TERMINALS

SALE! $525.00

VIEWPOINT — ADDS
Detachable keyboard, RS232 interface and auxiliary port, 80 x 24 display, tiltable screen **SALE**
SFADDVWPR Shipping Weight 30 lbs $699.00 **$525.00**

VT200 — VISUAL TECHNOLOGY
THE MOST RELIABLE TERMINAL WE'VE EVER USED!
Detachable keyboard, RS232C or 20MA interface, 110 to 19200 baud, 12" non glare 80 x 24 display, RS232 Aux. port and composite video out.
SFVSL200 Shipping Weight 55 lbs. **$995.00**

PRIORITY ONE ELECTRONICS©

9161S DEERING AVE • CHATSWORTH, CA 91311

MEMBER MITA — NEDA

ORDER TOLL FREE (800) 423-5922 CA, AK, HI CALL (213) 709-5464

Terms U.S. VISA, MC, BAC, Check, Money Order, U.S. Funds Only. CA residents add 6% Sales Tax. MINIMUM PREPAID ORDER $15.00. Include MINIMUM SHIPPING & HANDLING of $2.50 for the first 3 lbs. plus 25¢ for each additional pound. Orders over 50 lbs. sent freight collect. Just in case, please include your phone number. Prices subject to change without notice. We will do our best to maintain prices through June, 1982. Credit Card orders will be charged appropriate freight. If you don't already have your Spring '82 Engineering Selection Guide, send $1.00 for your copy, today. Sale prices are for prepaid orders only.

An Introduction To Data Base Management Systems

by Dr. Christopher L. Hamlin

A data base management system (DBMS) is a software product whose primary purpose is to allow users of data stored on a computer to work directly with the data items, without having to bother about how they are stored on the computer physically. This sounds like a simple problem to solve, but in fact many years of study have gone into the solution of the theoretical problems entailed in the design and implementation of a practical DBMS.

Another consequence of the complexity of this problem has been considerable inconsistency of vocabulary usage within the field, often as a result of a wish to score marketing points or to make a product appear in a particularly favorable light. This primer will present a consistent frame of reference for the software reviews of DBMS's which follow.

The core concept in DBMS theory and design is that of the separation of logical and physical views of data. For example, a user of a mailing list data set will have a *logical* view of the data set organization, which perhaps construes the data as records whose primary component is the last name of the individual addressed, and for whom the ordering principle involved is alphabetization by last name. The computer, on the other hand, stores the data in a *physical* organization which does not necessarily correspond with the user's logical view. For instance, the address data which the user views as logically alphabetized by last name could be stored in an indexed sequential format by zip code, and the physical contiguity of the records on the mass storage device may have nothing whatever to do with sequential alphabetization by last name.

The basic problem arising here is that of translating the user's logical view into a form which will permit the computer to retrieve the correct data, given some physical organization. In the example given, it would be very simple to write a program to translate the user's logical view to the computer's physical format, and come up with the required data. In fact, if only one or two such logical views of the data must be supported in a given application environment, that is probably the simplest approach to take.

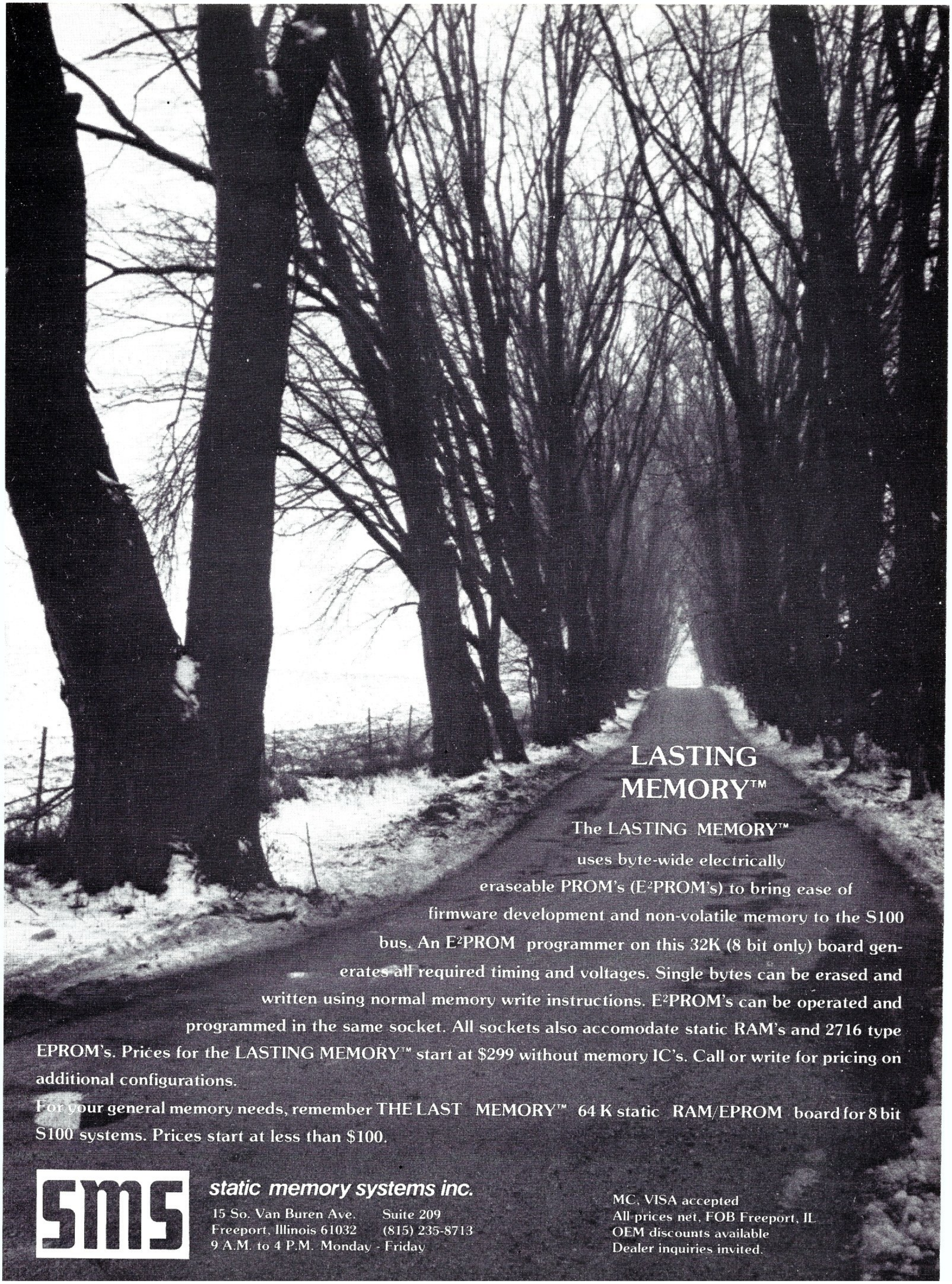
A data base management approach, however, becomes necessary when many different logical views of the content of the data base must all be supported. Notice that the problem could be solved by giving each user a separate set of data and applications programs. This involves great data redundancy in the computer, with attendant high costs, and massive special purpose programming efforts, with further redundancy. Out of the need to avoid such waste was born the first DBMS.

A DBMS offers the user some kind of facility for manipulating only the names of the data items to be operated on, without reference to their storage structure in the computer. This is the principal point of difference between file management systems (which sometimes masquerade as DBMS through over-enthusiastic promotion) and DBMS proper. In traditional file management approaches, the user needs some insight into the physical storage of the data items in order to gain access to them.

The different kinds of DBMS's are distinguished from one another primarily in the degree to which they are successful in allowing the user to operate on any combination of data items, no matter how complex, without having any knowledge of how the data are physically stored. The general term used to describe this kind of freedom in manipulating only the names of the data items is *data independence*—which means, essentially, the complete independence from one of the logical and physical views of the data.

A key notion here, and one which is central in assessing the power and value of a DBMS design, is that of *anticipation*. The degree of anticipation required by a DBMS refers to the extent to which it is necessary for a potential user or set of users to foresee and specify a general logical organization for the data in advance of

Dr. Christopher Hamlin, 8401 Corporate Dr., Landover, MD 20785.


LASTING MEMORY™
The LASTING MEMORY™ uses byte-wide electrically eraseable PROM's (E²PROM's) to bring ease of firmware development and non-volatile memory to the S100 bus. An E²PROM programmer on this 32K (8 bit only) board generates all required timing and voltages. Single bytes can be erased and written using normal memory write instructions. E²PROM's can be operated and programmed in the same socket. All sockets also accomodate static RAM's and 2716 type EPROM's. Prices for the LASTING MEMORY™ start at $299 without memory IC's. Call or write for pricing on additional configurations.
For your general memory needs, remember THE LAST MEMORY™ 64 K static RAM/EPROM board for 8 bit S100 systems. Prices start at less than $100.
SMS
static memory systems inc.
15 So. Van Buren Ave. Suite 209
Freeport, Illinois 61032 (815) 235-8713
9 A.M. to 4 P.M. Monday - Friday
MC, VISA accepted
All prices net, FOB Freeport, IL
OEM discounts available
Dealer inquiries invited.

particular applications. The evolution of DBMS design philosophies has, over the years, been toward reduction of the amount of anticipation required in setting up a data base.

Hierarchical, Network & Relational DBMS Design

This evolution of designs can be loosely described as a movement from *hierarchical* to *network* to *relational* designs. Some of the first DBMS designs to achieve commercial prominence required the user to logically structure the data elements in a hierarchical organization. This organization was communicated to the computer through use of a data language, and the user could then manipulate the data using merely the names as expressed in the hierarchy, without having to know anything about how they were actually represented in physical storage.

In some applications, of course, this worked out very efficiently indeed, both because the problem areas themselves had an inherently hierarchical organization, and because it was often advantageous at the level of the system software to move from a hierarchical logical view to a hierarchical physical storage model. A standard sort of problem, for instance, might be an inventory system in which a rigid hierarchy actually existed (at the logical level), so that a department, its products, the parts from which they were made up, the materials, and so on, could very effectively be represented as a hierarchy.

Two kinds of serious difficulty arose, however, as experience in the use of hierarchical DBMS's was gained. For one thing, it often happened that other users would wish to gain access to the same data sets that had originally been set up for, say, inventory data in ways wholly different from those foreseen when the data base was set up. It might easily prove difficult or impossible for the new application (which had not been anticipated) to be supported by a data arranged in a hierarchy dictated by the logic of the old inventory application's requirements. In short, the logic of unforeseen applications could turn out to be incompatible with the way the data had been defined in the system.

The second kind of problem concerned complexities arising when many users with different purposes were all actively using and modifying the same data base. A number of serious difficulties arose, which are usually referred to as *anomalies* of *insertion, deletion, update*, and *retrieval*. Without going into great detail, it may be said that certain attempts to carry out these activities would prove unexpectedly difficult, or would give rise to incorrect or unpredictable results. As just one simple example, imagine that the inventory data are set up in a tree or hierarchy such that the bottom of the tree (its furthest branches) contains the numbers of the parts making up all the products of all the departments. As long as you are searching the data base going down the levels of the hierarchy from department to product to part, you are in good shape, because the access paths are short. But suppose you wish (unexpectedly) to search for all occurrences of grommets in all products of all divisions. Suddenly the access paths for the search have become a nightmare, requiring that every possible branching path in the data base be traversed in exploring for grommets. Such searches would consume vast amounts of computer time, and become completely disruptive of DBMS activities (in some cases, the search could even be impossible in principle to carry out).

CODASYL DBMS

Hierarchical data base organizations have tended to be susceptible to retrieval and deletion anomalies; for these and other reasons, an organized effort to develop a superior alternative resulted in what are known as the CODASYL DBMS proposals, which have served as the foundation for many network DBMS designs. Network DBMS's allow the user great flexibility and latitude in deciding how the various kinds of data items in the data base are going to be related to one another; it also becomes possible to introduce new connections between data items easily, as the logic of changing applications gives rise to the need for them. Thus the definition of the data can grow and change as the needs of users expand and change. This represents a major advance over hierarchical designs which were enormously cumbersome to reorganize when new needs had to be accommodated.

The increased flexibility has its costs, however, in increased consumption of computer resources. In order to create the connections between the different data items in the system, pointers are generally used. Pointers, like data values themselves, require storage. If the set of applications for which the data base was set up is extremely diverse, with many users setting up radically different access paths between different data items, this could mean that the pointer set required could exceed the set of actual data values in size. Thus there is seen to be a direct tradeoff in network DBMS systems between generality (low required anticipation) and storage costs. There are also some rather complex problems involving the insertion and update anomalies which are common to some kinds of network design. In short, while they represented a big step forward, the network designs were far short of ideal in providing users with data independence.

Relational DBMS

A novel and highly influential analysis of this whole issue was published in 1970 by E. F. Codd, who proposed a quite different approach to a solution, which he grounded in mathematical results in the theory of relations. Codd's proposals, which are not easy to explain accurately in concise form, revolve around the idea of "normalization." Normalization is a highly specific procedure for arranging the names of the data elements in simple tables, made up of rows and columns (to use simple terminology in place of the formal language used by relational theory). Codd showed that it is always possible to find an arrangement of normalized tables with which to express the relationships between data elements. He also showed that if data are so organized, it is always possible to express *any* logically admissable combination of them with a simple query system, and that a data base management system based upon such tables and query procedures would be free of anomalies of insertion, deletion, retrieval, and update.

This proposal had profound effects on the DBMS field. It was not, however, easy to translate into practice in efficient, affordable systems. Fully a decade of intensive work has gone into realizing the concepts in practical form. On the other hand, although building a truly relational system was difficult, it wasn't at all hard to use the jargon associated with the theory to dress up advertising copy, instantly transforming older designs into relational Cinderellas. The lack of consistent usage, together with the difficulty of knowing just how a given system actually works, can make debunking claims of this sort very difficult.

> *Relational DBMS claims are almost impossible to evaluate objectively without copies of the source code or detailed descriptions of the systems level functions of the package, and these in turn are generally hard to come by. But given the great inherent complexity of the problem, such claims are probably best treated 'cum grano salis' (with a grain of salt).*

Without entering into the complexities of the problem, we may say the following: It is a necessary but not sufficient condition for a DBMS to be called "relational," that it support the creation of relations in third normal form for the representation of the users' data semantics; that it support a query system (a relational calculus or relational algebra) expressing a first order predicate calculus; and that it be capable of mapping such a query system into physical storage. Which just goes to show that it is not a simple problem.

A considerable number of mainframe and mini-computer relational DBMS's have now been developed, and the market is gradually being sorted out. Many of the entrants in the growing microprocessor DBMS field also identify themselves as relational. These claims are almost impossible to evaluate objectively without copies of the source code or detailed descriptions of the systems level functions of the package, and these in turn are generally hard to come by. But given the great inherent complexity of the problem, such claims are probably best treated *cum grano salis* (with a grain of salt).

Much more to the point is whether the DBMS will do what the application requires, and do it efficiently and accurately. In assessing this, one is back to the older problems of anticipation, data independence, and freedom from anomalies. A DBMS of any kind should be able to flexibly support a wide range of user logical views of data, without requiring any knowledge of the physical storage characteristics. The operations of insertion, deletion, update and retrieval should always be possible for any logically specifiable combination of the data. The operation, no matter how complex, should take place in a reasonable amount of time without consuming excessive storage.

It is interesting that there is no hard and fast answer to the question "What is the best DBMS for me?" *If* your application's logic is quite static, and can easily be expressed hierarchically, and needn't accommodate new applications unexpectedly, it is entirely likely that performance improvements would be realized by choosing a hierarchical DBMS design over a relational one. But if your application entails searching a complex data base for ever-changing combinations of data elements in novel and unexpected combinations, then the logical open-endedness of the relational model is important and you probably couldn't make do with a hierarchical system.

Regardless of the type of DBMS selected, the implementation must be a good one; microprocessor systems, no matter how fast or how large, are pushed to the limits of their capabilities by the demands of DBMS activities, and unless the implementation is very carefully conceived and executed, a DBMS will all too often bog down hopelessly when a certain critical level of data volume and/or complexity is achieved. ■

NEW LOWER PRICES

"THE BIG BOARD"

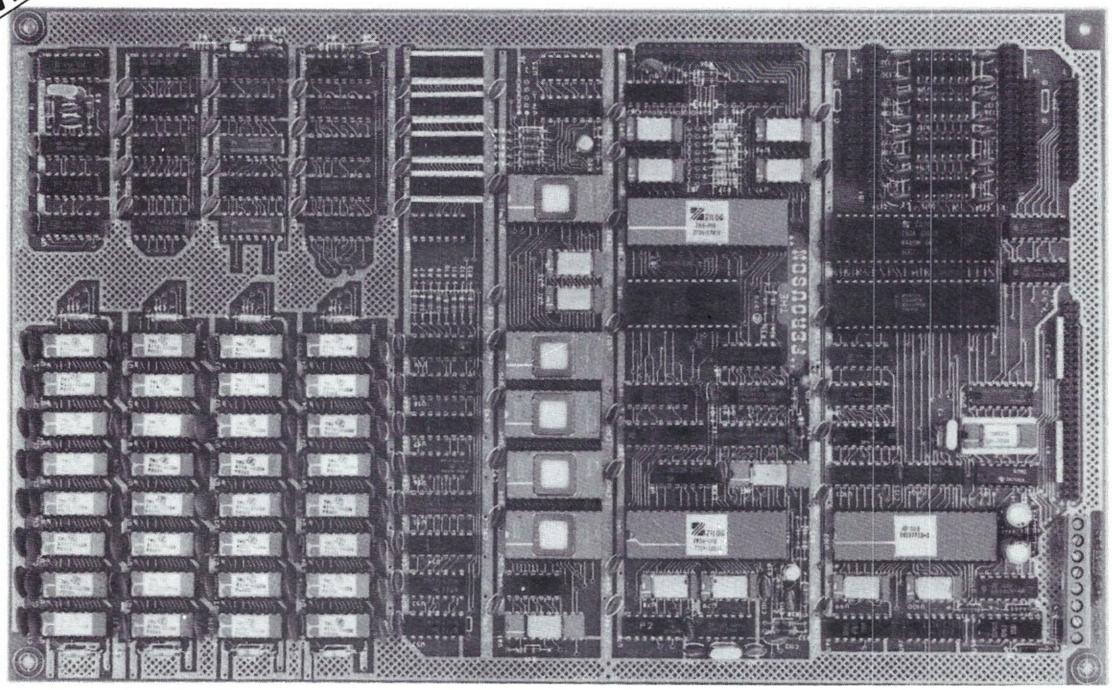
OEM - INDUSTRIAL - BUSINESS - SCIENTIFIC

SINGLE BOARD COMPUTER KIT!

Z-80 CPU! 64K RAM!

NEW!

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



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

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

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

$499.00** (64K KIT BASIC I/0)

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

FULLY SOCKETED!

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

64K RAM

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

Z-80 CPU

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

SERIAL I/O (OPTIONAL)

Full 2 channels using the Z80 SIO and the SMC 8116 Baud Rate Generator. FULL RS232! For synchronous or asynchronous communication. In synchronous mode, the clocks can be transmitted or received by a modem. Both channels can be set up for either data-communication or data-terminals. Supports mode 2 Int. Price for all parts and connectors: $65.

BASIC I/O

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

24 x 80 CHARACTER VIDEO

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

FLOPPY DISC CONTROLLER

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

TWO PORT PARALLEL I/O (OPTIONAL)

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

REAL TIME CLOCK (OPTIONAL)

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

BLANK PC BOARD — $175

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

CP/M* 2.2 FOR BIG BOARD

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

PRICE CUT!

PFM 3.3 2K SYSTEM MONITOR

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

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

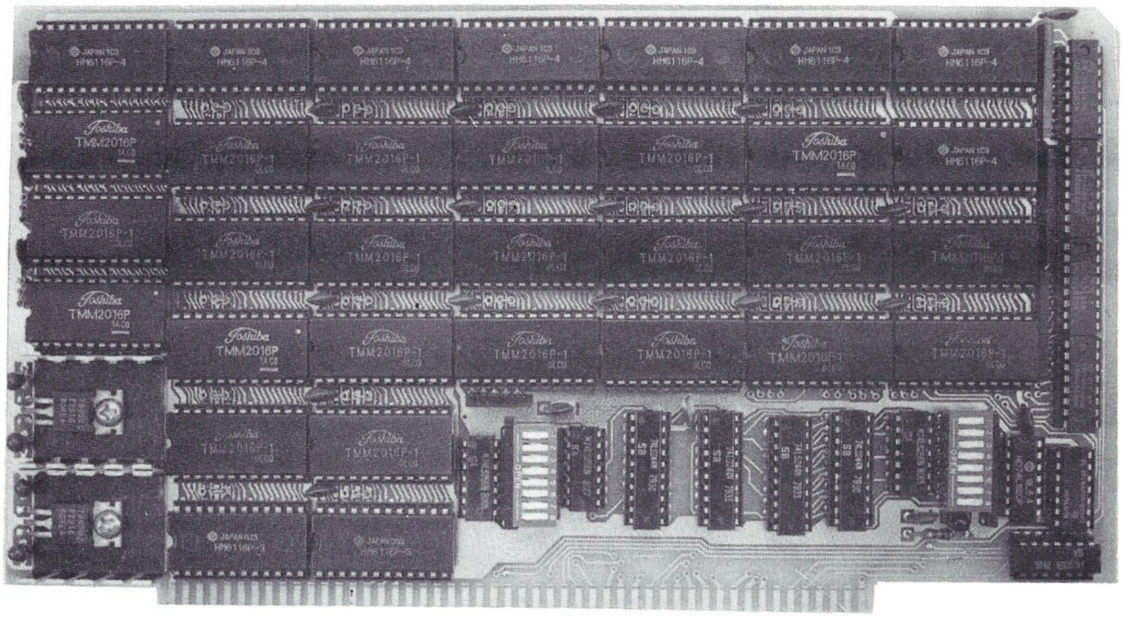
TERMS: Shipments will be made approximately 3 to 6 weeks after we receive your order, VISA, MC, cash accepted. We will accept COD's (for the Big Board only) with a $75 deposit. Balance UPS COD. Add $4.00 shipping.
USA AND CANADA ONLY

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

64K S100 STATIC RAM

NEW! **$399.00 KIT** **NEW!**

LOW POWER!



RAM OR EPROM!

BLANK PC BOARD WITH DOCUMENTATION $55

SUPPORT ICs + CAPS -	$17.50
FULL SOCKET SET -	$14.50

ASSEMBLED AND TESTED **ADD $40**

FEATURES:

FULLY SUPPORTS THE NEW IEEE 696 S100 STANDARD (AS PROPOSED)

- ★ Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
- ★ Fully supports IEEE 696 24 BIT Extended Addressing.
- ★ 64K draws only approximately 500 MA.
- ★ 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
- ★ SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
- ★ 2716 EPROMs may be installed in any of top 48K.
- ★ Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
- ★ Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
- ★ BOARD may be partially populated as 56K.

FOR 56K KIT $349

16K STATIC RAMS?

The new 2K x 8, 24 PIN, static RAMs are the next generation of high density, high speed, low power, RAMs. Pioneered by such companies as HITACHI and TOSHIBA, and soon to be second sourced by most major U.S. manufacturers, these ultra low power parts, feature 2716 compatible pin out. Thus fully interchangeable ROM/RAM boards are at last a reality, and you get BLINDING speed and LOW power thrown in for virtually nothing.

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

TERMS: Add $2.00 postage. We pay balance. Order under $15 add 75¢ handling. No. C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50, add 85¢ for insurance.

DataStar: A Data Management System

by Glenn A. Hart

DataStar is claimed by its authors, MicroPro International, to be a "comprehensive data entry, retrieval and update system." As we'll see, this is a good description of a program which can do many things extraordinarily well and yet doesn't do some things it might.

The issue at hand is whether DataStar is a data base manager in the sense that the term is usually used. The most direct answer is that it is clearly not, since some basic features of DBM such as report generation are not included at all. On the other hand, many of the common functions necessary for a DBM are provided. Thus it is probably more accurate to say that DataStar has the *kernel* of a DBM incorporated into its many features.

I prefer to think of DataStar as an excellent *key-to-disk* utility, meaning that the program excels at providing the operator interface for data acquisition and entry. DataStar makes it very easy to design screen forms to structure the data entry process, has many sophisticated data validation procedures including batch validation for critical applications, accessing to existing files for data validation and many other useful and convenient capabilities.

The DataStar system consists of two main program modules, FORMGEN and DATASTAR. As their names imply, the former is used in defining screens and data entry forms while the latter is used for the actual data entry. An INSTALL utility configures both programs simultaneously for the video terminal to be used. The procedure is very similar to installing MicroPro's WordStar word processor, and is quite easy if the user has one of the many terminals listed. If not, a well planned prompting session elicits the necessary information from a reasonably knowledgeable user.

Once configured, FORMGEN can then be used to define a form. Both DataStar modules use on-screen help menus to either explain program operation or to remind the user of the control codes employed. The screen menus are very reminiscent of WordStar's and are equally helpful to the beginner user. As with WordStar, the menus can be disabled by experienced users to provide more display room for the actual form.

Glenn A. Hart, 51 Church Road, Monsey, N.Y. 10952

The general approach of the user interface is also very similar to WordStar, in that control codes are used for almost all commands. The specific codes are as similar as possible to WordStar's (see Table 1), so users of that excellent word processor will immediately feel right at home. Also as in WordStar, a status line at the top of the screen provides much useful information about what is going on. It displays the line and column number at which the cursor is positioned, the sequential number and length of the current datafield and the position of the cursor within it and the "editing character" associated with the character position of the datafield (more about this soon).

FORMGEN

Designing a form proceeds in two main steps. First, the prompts and the actual data field acceptance areas are simply "painted" onto the screen using cursor movement commands. The data areas are designated with underscore characters, one for each allowable position in the item. DataStar has a default number of lines and columns built in, but simple commands allow expanding the work area. There are no "pages" as such; the display scrolls up to accommodate extra lines beyond the first screenfull. If the video terminal used is capable of some form of highlighting, DataStar can mark any desired area of the screen to appear in the highlighted mode. Editing commands very similar to WordStar allow correcting entries by adding or deleting characters, lines, etc.

Once the basic form is laid out at least one field must be designated as a "key field," meaning that records are sorted according to the data in this field and the data can be used as an index for the record. If more than one field is designated as a key, DataStar combines the data in the multiple fields to generate the key for the records. The key field must be unique (unduplicated by the key field of any other record) if certain searching capabilities of DataStar are to be used to locate the records, otherwise duplicate keys are acceptable. It is tempting to define several fields as keys, but this slows operation greatly.

Now the real fun starts. DataStar includes very comprehensive provisions for editing and validating the incoming data by assigning "attributes" to each data field. This is done by positioning the cursor within a datafield and hitting Control-R, triggering a series of questions which define the desired characteristics of the permissible data. The program displays the sequential number of the field, which the user can alter if he wants data to be entered out of sequence. The attributes of the field being entered can be copied from a previous field if the attributes are very similar or identical, saving time.

If the field is a key, the number of the key is displayed in case there is more than one. This order can also be changed if desired. Duplicate keys can be refused, meaning that when data is being entered DataStar will examine all the keys entered previously and reject any matches. If multiple keys exist, DataStar can be asked to generate a "tie breaker field," which is a special, non-displayed key field containing the smallest integer necessary to guarantee the uniqueness of the combined key fields which otherwise match.

The next question asks if the field is derived, meaning is it generated by the interaction of other existing fields in the data base or from a separate data file. In the first case, both arithmetic and string operators are provided to manipulate the values stored in fields and/or constants. The second situation is even more powerful. A completely separate data file can be created which contains data to be accessed by the form currently being used. Some common key field must exist to allow DataStar to access the correct record in the remote file.

Continuing the question and answer session, the designer can require that data be entered into a given field or can allow the operator to bypass it without entering any data. Data can be either right or left justified within the defined field width. "Pad" and/or "float" characters can be specified to precede or follow the entered data, and these extra characters can be used only to format the displayed data or can be saved with the data itself.

The forms designer can specify range checking for a field and indicate the minimum or maximum value over which the data can range. A check digit scheme is also provided which applies to numeric entries and requires that they be divisible by eleven. I may have missed the point, but I can't quite envision any application for the check digit feature.

More importantly, every character position within a data field can employ an "edit mask" consisting of two parts. The first mask (see Table 2), known as the "Entry Control" mask, controls whether a character must be entered, whether the character will automatically be copied from the previously entered record or will be a constant (usually some punctuation character) and, in either case, whether the operator will be allowed to override the copied value or constant. DataStar can also be instructed to insert constants (again usually a comma or some punctuation) if data exists on either one or both sides of the constant position.

Table 1: *FORMGEN Control Codes.*

```
CURSOR:    ^A=left item    ^S=left char    ^D=right char      ^F=right item
           ^E=up line      ^X=down line    ^U=set/clear tab   ^I=tab
DELETE:    DEL=char left   ^G=char right   ^T=entire column   ^Y=entire line
INSERT:    ^P=line buffer  ^V=char right   ^B=entire column   ^N=entire line
FIELD:     ^Q,_=add/extend field           ^Z=delete field    ^R=define field
HIGHLIGHT:^O=set/clear     ^L=delete block
OTHER:     ^J=rotate help  ^W=list form    ^C=form done       ^K=toggle key
```

Table 2: *Entry Control Character Codes.*

```
! = must enter a character         _ = may enter or leave blank
X = automatic copy                 Ȳ = auto copy/may enter
" = constant in this position      ~ = constant/may enter
' = constant if data on            / = constant if data on one
    both sides of constant             side only
```

Table 3: *Content Control Character Codes.*

```
A = A-Z only                    a = a-z only
B = A-Z, space                  b = a-z, space
C = A-Z, a-z ->A-Z              c = A-Z, a-z
D = A-Z, a-z ->A-Z, space       d = A-Z, a-z, space
E = A-Z, 0-9                    e = a-z, 0-9
F = A-Z, 0-9, space             f = a-z, 0-9, space
G = A-Z, 0-9, a-z ->A-Z         g = A-Z, 0-9, a-z
H = same as G with space        h = A-Z, 0-9, a-z, space
9 = 0-9 only                    8 = 0-9, #$%()*+-.
. = align decimal point         _ = any character allowed
```

Table 4: *DataStar Initial Menu.*

```
A = Add new records         K = select records by KEY      E = Exit current fc
I = SCAN in Index order     D = SCAN in Data file order    M = edit scan Mask
B = select Batch file       V = Verify batch file
F = File maintenance        J = Help
```

The second edit mask is called the "Content Control" mask and specifies the alphanumeric characteristics allowed in the position (see Table 3). A wide variety of combinations is provided, and DataStar will convert lower case to upper if desired. Use of both edit masks can quite precisely specify what characters the operator can enter, resulting in much better data integrity.

DataStar provides for three different types of data verfication in addition to the capabilities outlined above. Sight verification requires the operator to move the cursor through the fields designated for this type of verification a second time, allowing correction if visual inspection reveals some type of mis-keying. More demanding is retype verfication, which blanks the specified field and requires the operator to retype the data. If the second typing agrees with the first the program continues, otherwise the operator must try again. Finally, list verification checks the data in a field against a list of values kept in an external data file, passing only those entries which have a matching entry in the remote file. This can be used for some rather sophisticated checking. Finally, DataStar has special provisions for batch verification, which means that sub-files of data are created, validating in chunks, often by operators other than those which originally entered the data, and then combined or pooled into the final data file. The batch processing features are sometimes used in very critical data entry projects, but are not likely to be used in most reasonable applications.

The remaining FORMGEN commands are primarily housekeeping functions. A completed form can be printed for checking the design or as additional documentation. The print routines produce several pages of output for even a simple form and fully detail every aspect of the forms structure.

DATASTAR

Data is actually entered using the DATASTAR module. When invoked, the menu shown in Table 4 appears. The operator mode is Add for adding records to the data file. The operator merely inputs the data in the provided areas, with various editing and movement commands available to aid in the entry process if required. Data from the previously entered record can be copied to the current record, and the data in a record, either with or without the form template itself, can be sent to the printer for hard copy. When all data is entered the various field checks and range checks are performed and all fields which require entries are checked. Any errors generate explanatory prompts and reposition the cursor to the offending field for reentry or correction.

As each record is entered, DataStar determines the position the record should occupy in the sorted key order for the data base. A separate and distinct index file is created as each record is entered. This index file allows access to the data in key order as well as the sequential order in which the records were originally entered.

The DataStar module allows searching for a specific record in several ways. Searching by key requires the operator to enter the key value (which may be the values of more than one field if multiple key fields were defined) and then locates the first matching value, if any. If duplicate keys were allowed, the user must switch to the scan by index order mode discussed next.

Scanning by index merely moves through the file in the sorted index order instead of the entry order. Unless the file was originally entered in some sorted order, this is usually the preferred method to examine a series of records. DataStar does allow moving through the file in data file order if desired.

The final search method involves defining an edit scan mask for a field or group of fields. This mask is created simply by moving the cursor into a field and typing the characters or words being searched for. DataStar initially sets all positions to asterisks, which match any character. This provision allows finding a record if almost anything is know about its contents and is quite powerful and flexible.

Once a desired record is located with any of these methods, the data can be corrected or modified if desired. DataStar automatically updates the data stored on disk and, if necessary, the pointers stored in the index file.

Performance

If the above descriptions make DataStar sound complex, this is at once correct and at the same time a bit misleading. DataStar certainly has many capabilities, which result in many commands; learning to use the program completely does requires a fair amount of study. On the other hand, once even a small amount of time is spent learning the basic principles employed, designing forms becomes almost trivial and the most difficult forms can be structured quite quickly.

One valid way of looking at the utility of the program is comparing the time necessary to code a DataStar form against the time required to code the same form in a high level language. In designing data entry systems for my consulting clients, I have found that my programming time with DataStar is perhaps one-fourth to one-third of what I would have needed in Basic or Pascal (and I'm pretty fast with both of those). Of course there are some things which can be done with custom programming that DataStar doesn't include, but in general my clients have been happy with DataStar as an entry interface.

I have run across a few problems with DataStar. While it generally works exactly as advertised, a few of my clients who must work with very large data files have found that as more and more data is entered, DataStar begins to require a rather unacceptable amount of time to determine a new record's position within the file and generate the necessary pointer to the index file. DataStar has a file maintenance provision which reorganizes the index file to reduce this delay, but the file maintenance program itself takes a very long time to execute, even on a fast hard disk, and only somewhat reduces the entry delay. In one case, I even had to switch a client back to custom code Basic for the entry process. The main problem lies with the indexing capabilities. MicroPro should add a switch which would disable generation of the index file; while this would prevent searching the file in sorted order, many applications (including this client's) do not need the indexing features. Don't misinterpret the magnitude of this problem; the delay only became annoying after well over ten thosand names were entered into a mailing list. In most normal applications DataStar is more than fast enough.

I've also noticed a few strange events with double sided, double density disks. I think this also may have something to do with how DataStar points to the individual records, but it also might be an interaction with DataStar's internals and the specific disk system used. I've never noticed any problem with standard single sided, single or double density or hard disk formats

Summary

I believe DataStar is an excellent program for designing data entry applications. The tremendous flexibility and sophistication provided allows excellent control over the integrity of data entry, which is one of the primary aspects of many application systems. While DataStar does allow searching for individual records and updating them in some clever ways, it clearly is *not* a data base management program in the sense that term is usually used. I understand that MicroPro will be releasing a report generation program which will work with DataStar to form what MicroPro claims will be a full data base manager. I'll reserve judgment until I see the finished product, but DataStar certainly would form a reasonable foundation upon which to build. ■

G&G ENGINEERING is working wonders with CompuPro™!

You can get *FAST* single and multi-user CompuPro based configurations fully integrated from G&G ENGINEERING.

MULTI-USER SYSTEMS

MP/M™ 8-16*

Are you having trouble deciding whether to go 8 bits (with lots of software that runs *NOW*), or 16 bits (with all that potential power)? Thanks to G&G, it's a split decision—*EVERYBODY WINS!* G&G ENGINEERING's implementation of MP/M 86™ does it all! We call our package **MP/M 8-16** because it uses Digital Research's MP/M 86 to provide a multi-user, multi-tasking environment for running both CP/M 2.2™ and CP/M 86™ compatible programs . . . *AT THE SAME TIME!*

That's right! Imagine a system where one user could be running Wordstar™, another dBASE II™, a third SUPERCALC™ (all 8 bit CP/M 2.2 compatible programs), while a fourth user runs CBASIC 86™ (16 bits and CP/M 86 compatible)! It's all made possible thanks to CompuPro's powerful hardware, including the 8085/8088 processor board and DMA disk controllers, and thanks to G&G Engineering's software expertise.

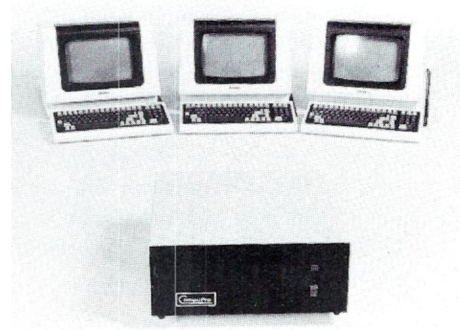
But flexibility isn't the only good news. This system is also *FAST* (naturally, it's by G&G). What's more, it's *BIG.* CP/M 2.2 compatible programs when run under **MP/M 8-16** are not limited to user work areas of 48K, as with 8-bit-only MP/M systems. With MP/M 8-16, your work area for 8 bit software is a whopping 62K!

Plus you can install up to a Megabyte of CompuPro high speed static RAM memory and an assortment of hard disk drives. This system means business.

*MP/M 8-16 is a proprietary implementation of MP/M 86 version 2.0. This implementation takes advantage of the CompuPro 8085/8088 processor card and can not be run on other CPU cards. MP/M 8-16 is a copyright of CompuPro under license from Digital Research. The Loader, Command Line Interpreter, XIOS, and Swap program are copyrighted by G&G Engineering.

UNIX®

DUAL SYSTEMS and CompuPro make a perfect match in the best micro-computer version of UNIX we've seen. This is not "UNIX-LIKE"—this *IS* UNIX, implemented on the MC68000 processor (32 bit internal registers!). For the software connoisseur.



The best keeps getting better!

APPLICATIONS
(for users of our CompuPro based systems)

SUPERSHEET™

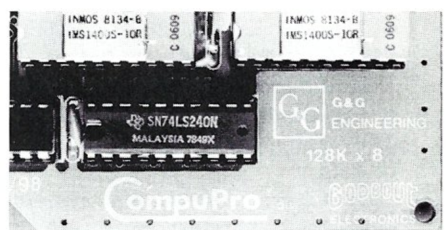
Now you can really spread out with ***SuperSheet,*** G&G's enhancement of Sorcim's SUPERCALC.™ Upward compatible with all your existing SUPERCALC files, ***SuperSheet*** allows up to **15** times more storage! If you are among the many who have been frustrated by the Memory FULL message, ***SuperSheet*** is the answer to your prayers! This is the largest electronic spreadsheet program on the market. Once again, G&G has taken the best and made it even better. (Requires a minimum of 128K RAM.)

WARP DRIVE™

Compute in the *fast* lane with **WARP DRIVE,** G&G's RAM memory disk drive. **WARP DRIVE** is extended address RAM memory configured with CP/M 2.2 to look like a disk drive. Your programs can run from 10 to 30 times faster! When running CP/M 86, **WARP DRIVE** automatically becomes directly addressable main memory. **WARP DRIVE** is available in sizes ranging from 128K to 1 Megabyte.

CPM-IBM

Turn your G&G system into an IBM-ulator! Your system can become a powerful software development tool for the IBM Personal Computer. (We use CompuPro's Dual Processor 8085/88—the same hardware IBM used for their own development!) Our **CPM-IBM** program allows you to transfer CP/M® compatible files to and from 5¼-inch diskettes in the IBM DOS format. The program discerns what kind of file or diskette it is reading, then does the appropriate conversion—IBM to CP/M or CP/M to IBM. G&G makes it easy!



G&G ENGINEERING

1922 Republic Avenue, San Leandro, CA 94577 (415) 895-0798

G&G Engineering is an authorized CompuPro Systems Center. We specialize in CP/M S-100 Software and Hardware integration. Our engineers are factory-trained experts in CompuPro products. All our CompuPro equipment is covered by a 2-year phone exchange warranty.

*COMPUPRO is a trademark of Godbout Electronic; MP/M, MP/M 86, CP/M, CP/M 2.2 and CP/M 86 are trademarks of Digital Research; WORDSTAR is a trademark of MicroPro; dBASE II is a trademark of Ashton-Tate; SUPERCALC and SUPERSHEET are trademarks of Sorcim; CBASIC 86 is a trademark of Compiler Systems Inc.; UNIX is a trademark of Bell Laboratories; WARP DRIVE is a trademark of G&G Engineering.

MDBS: A Database Mangement System

by Bill Machrone

A review of Micro Data Base System's full network DBMS package.

All you need do is open any of the microcomputer magazines these days to realize that a lot of money is being spent on advertising data management systems, data base management systems and data-handling programs. This is attributable to two interrelated facts: Programmers and users need application programs that "understand" and properly handle the relationships among items of data; and high level languages are, for the most part, severely deficient in both storing and facilitating access to the data. For the purposes of this article, I offer the following definitions:

> *Data management systems* provide a means of collecting data from screens presented to the user, storing the data and retrieving the data in user-specified formats, as reports or extracted subfiles of the data. Examples are dBASE II, CBS, Selector, FMS-80 and Condor.
>
> *Data handling programs* are application generation programs such as The Last One and Pearl. They, too, provide collection, storage and retrieval of data.
>
> *Data base management systems* require a host language to handle some phases of their operation, typically data collection and retrieval. MDBS is a data base management system. There are several distinct types of DBMS's, as well, based on the "view" or ways in which the relationships among data items can be described. The three major categories are relational, hierarchical and networking.

Without spending a lot of time on the differences, in *relational data bases,* records are viewed as a collection of lists. Common data items form the connecting link between lists. *Hierarchical data bases* permit records to own other records in a one-to-many manner. In *networking data bases,* records can own other records, themselves or be owned by other records in a one-to-many or many-to-many manner. MDBS is a full networking DBMS.

MDBS is available in two versions, for a large variety of host languages, operating systems and processors. This article will concentrate on the CP/M versions for Microsoft Basic-80, the companion Basic compiler and Digital Research's PL/I. MDBS I is the original product and MDBS III is a new product that complements, rather than replaces, the earlier product. The differences, where significant, will be noted below.

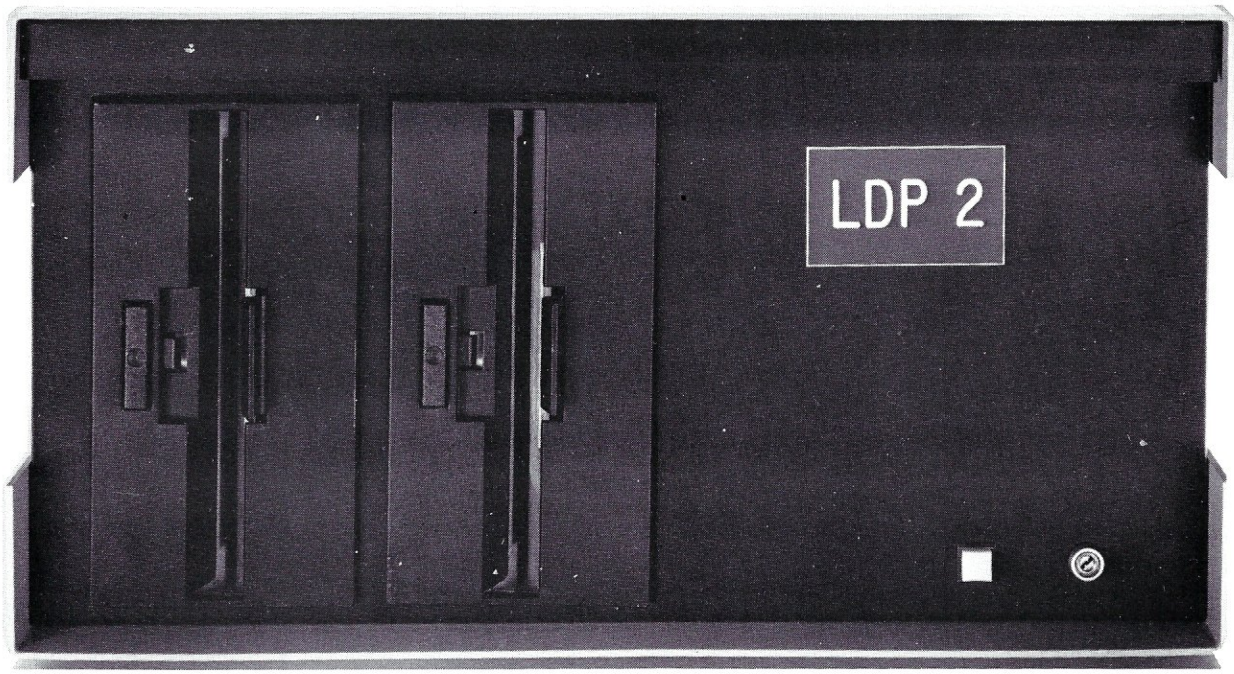
MDBS III is one of the most significant software systems offered to date for a microcomputer. Furthermore, it may well become one of the most significant programs for many minicomputers, too.

A Close Look at MDBS

The first thing that MDBS requires is that you describe the way in which data is to be stored, how large the data items are and what the relationships are among records. This is done with a utility program called DDL, for Data Definition Language. DDL combines a line-oriented editor and a data base processor that scans the entries for syntax and constructs an initial data base from the entered description. The data definition phase also includes operating parameters, such as which drives will

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

YOU CAN'T BUY A FASTER S100 BUS SYSTEM



Presenting the LDP 1/2 S100 Bus System:

Standard features:
- ☐ *Lightning One* CPU
- ☐ 2 Serial Ports
- ☐ 2 Parallel Ports
- ☐ 128K RAM
- ☐ 600K bytes floppy storage
- ☐ MS-DOS** or CP/M-86* operating system

Options available:
- ☐ 10 MHz 8086
- ☐ 8087 – 8089 co-processor (onboard, not tacked on)
- ☐ 128K static RAM
- ☐ Multi-user option including MP/M-86*
- ☐ 1.2M bytes disk storage
- ☐ 2.4M bytes disk storage
- ☐ 11.2M bytes disk storage

Lightning One trademark of Lomas Data Products, Inc.
**CP/M-86 and MP/M-86 trademark of Digital Research.*
***MS-DOS trademark of Microsoft Inc.*

The LDP 1/2 system incorporates the fastest S100 bus CPU, the *Lightning One*™. The *Lightning One*, when coupled with our other fine board products offers a level of performance comparable to many large 16-bit minicomputers. Our multi-user system *doesn't lose steam* as with other 8-bit systems.

In addition to *Lightning* performance, Lomas Data Products has the widest selection of operating systems software. We offer both MS-DOS** and CP/M-86* for single user systems and MP/M-86 for multiuser systems. A wide variety of high-level languages are available including: BASIC (interpreter and compiler), FORTRAN, PASCAL, COBOL, C, and FORTH. Call or write for our benchmark comparisons with other systems.

We have saved the best for last: Our standard system, the LDP 1 with 600K bytes of disk storage is available for just $4150.

Dealer and OEM inquiries invited.

For 16 bit computing on the S100 bus, come to the leader...

LOMAS DATA PRODUCTS, INC.
11 Cross Street
Westborough, Massachusetts 01581
Telephone: 617-366-4335

store the data, the size of the page buffers and passwords. MDBS III permits several more data types, such as date, time and string, in addition to the standard character, integer, real and binary. The string data type is very useful, as it automatically provides data compression and elimination of trailing blanks. The date type performs validity checks that include proper days in short months, including leap years.

The essence of a data base management system is that you, as a programmer, no longer have to worry about where and how the information is stored on disk. The essence of an exceptional data base management system is that it allows you to worry about it if you want to. MDBS III permits clustering of records, which means that associated records will be located as close together as practical on disk in order to minimize retrieval time. MDBS III also has a CALC or calculated record placement option. This is quite similar to hashing, but without the usual space penalty. Retrieval of CALC'ed records is quite fast. When you execute a "find member of set based on sort key" (FMSK), MDBS I performs a binary search on disk if the desired record is not in its memory buffers. MDBS III improves on the binary search with look-ahead tables and sampling of keys.

The line-oriented editor in both MDBS I and MDBS III is decidedly inferior to the screen-oriented editors most of us use on microcomputers. In MDBS I, however, you are virtually forced to use DDL's editor, because the input format is position sensitive. That means that the columns in which parameters appear are important to the correct interpretation of those parameters by the data base processor. DDL eases this particular pain by displaying, at the operator's request, masks that show the proper location of parameters. Line numbers four digits in length are required. Comments and blank lines are illegal.

MDBS III changes all that. DDL permits free-form entry with indentation, comments, blank lines and whatever else is necessary to make the data base definition easy to understand. Comments are implemented PL/I style, beginning with a slash-asterisk and ending with an asterisk-slash. You can use your favorite screen editor to design the data base, but you must still invoke DDL to perform the analysis and initial construction of the data base. Version III also introduces a new storage concept, that of data base "areas." While both versions provide the ability to spread the data base across multiple drives, "areas" can be dynamically allocated to specific drives. This means that the entire data base need not be on line all the time. An application program, after consulting the master data base, can request that a disk be mounted in a specific drive. All of the records and items associated with that area will only be searched for or written on that physical disk. Thus, an accounting data base may have the general ledger and chart of accounts resident in the master data base, while payables, receivables, inventory and open orders are each assigned to a different area, each of which may be on its own disk, any of which may be called for by the main program.

Data Security

Both versions of MDBS have extensive security provisions. MDBS I uses read and write "levels" that tie in to each user's password. Each record, item and set can have a read and a write level associated with it and there may be up to 255 unique levels. The levels associated with your password must be equal to or greater than the levels for the records, items or sets you want to access. MDBS III uses a far more sophisticated scheme, using the first sixteen letters of the alphabet, permitting up to 65,635 unique assignments for read and write access. Each record, item, set and area may be assigned a group or range of values. The user must have one of the values associated with the desired record, item or whatever in his password in order to be granted access. MDBS III also offers encryption on an item by item basis.

Processing Data

The data base analyzers in both versions function similarly, except for the fact that version III's are a great deal smarter. Both of them give meaningful error messages that permit rapid correction of the data base definition.

The processed data base definition is stored as a data dictionary at the beginning of the data base file. It defines the record types, data items within the records and the sets that tie them together. MDBS III provides a more complete data dictionary, as it includes room for a title and one or more synonyms for each item, record or set.

Once the data base has been designed and initiated, you have to write a program which will call the data management system (DMS) routines and manipulate the data as desired. MDBS I provides 65 different DMS calls, while MDBS III has over 100. The increase in flexibility is similar to that of the Z-80's instruction set over that of an 8080: the old one is adequate and the new one is almost an embarrassment of riches. The actual calling syntax differs for each language, but follows a general form in which parameters are passed to the DMS and an error code is returned from the DMS. Basic performs these calls by passing the addresses of the parameters rather than the actual parameters. If the action requested involves moving a block of data (a newly input record, for example) then the DMS must have been informed in advance of the location and size of the variables that will hold the data. String variables must be initialized in the program to the same size at the data items in the data base. In contrast, PL/I is a record-oriented language and provides named data structures. Initialization is done when the structure is declared. Your application program can simply fill up the structure with data, then call the DMS with the name of the structure.

What sorts of DMS commands can be executed? By general category, there are commands that create and delete records, those that find and retrieve records, those that connect and disconnect records through specified sets, commands that modify existing records and utility commands that return counts of records in sets and the like. MDBS III has more commands in each category and has a few additional categories, such as multi-user record locking, recovery and Boolean commands.

The Boolean commands deserve special mention. They permit rapid creation of unique groups of set members based on the members or owners of other sets. That is, they compare the members (or owners) of one set to the members (or owners) of another set and create a third set which represents the members (or owners) in common

(or not in common). While usage of this group of commands can be rather esoteric, they may save many dozens of lines of coding and much repetitive processing in your application program.

So the chief duties of your application program are to provide input and output screens to the user, call the appropriate DMS routines, traverse the data base (find subordinate members of sets), edit input data and respond to error conditions from the DMS. For simple management of a flat file, say a name and address list, there will be no net savings in the number of lines of code written. The benefits begin to show when the organization of the data becomes more complex or access needs to be more flexible. For instance, any simple Basic program can retrieve random records by record number. With a little binary search routine (or a slow sequential search) it can also retrieve based on any data item in the file. Your program starts to go to pot when you need to retrieve on multiple data items or when the file has repetitive data items that may occur many times. MDBS makes these applications a snap because you no longer have to think about how to do the access once you have determined the structure. In essence, your application program never exceeds a certain "ceiling" of complexity as you develop MDBS applications. You wind up putting modified screen I/O and edit segments into the same root program.

Retrieving Data

Retrieval and report generation is another key item. Writing programs to generate hard copy reports or to summarize files is one of the most tedious jobs a programmer would ever want to do. The application, however, is most likely useless without reports. To fill this need, MDBS offers QRS, the Query Retrieval System. QRS can read an MDBS data base and generate reports on an ad hoc basis, and it can store procedure to generate frequently-used reports. The syntax is sufficiently English-like to warrant use by non-technical users. The only catch is that the user must have a "map" of the data base, showing which records own which other records and what data items are in each record. Furthermore, the user must *understand* the map, so that he knows when he is traversing a set backwards.

A typical request to QRS might appear:

```
LIST EMPNAME DEPTNO DEPTNAME SALARY THRU EMPLOYEE SALHIST
```

In this case, we have a hypothetical data base with an employee master record accessible in alphabetical order through the set "EMPLOYEE." Each employee master record owns multiple occurrences of salary history records through the set "SALHIST." Let us say that we have also created a set called "SALAMT" that points to the salary history records from the outside world (not connected to any employees) and maintains them in order of increasing amount. To see the employee roster in salary order, we might enter:

```
LIST EMPNAME DEPTNO DEPTNAME SALARY THRU SALAMT >SALHIST
```

The path statement says to access the data base through the salary amount set and then to traverse the salary history set *backwards* (the ">" flag) to access the employee who owns each amount record.

Of course, QRS understands conditionals and subtotals, so that employees can be grouped on a page-per-department basis, or a search limited to those making more than $20,000 per year or whatever. An entire report can be stored as a one-word macro that can execute multiple QRS statements. A macro can, for example, turn the printer on, insert a report title, run a report, turn the printer off, change the title, run a second report to the screen and return control to the user. The macros are stored right in the data base file, so that a macro library can be established for all users.

QRS can also output to disk, either in report format, raw data format or, for MDBS III, the format of the host language. Version III also includes special commands for searching a specific data base area and for displaying the expanded data dictionary. The new version also has several mechanisms for handling nested conditional selection criteria, both explicitly and implicitly. That means that the search can have nested conditional selection criteria that you tell it about in advance or ones that it learns about as it is executing the search.

The biggest single enhancement for Version III however, is the internal sort. In MDBS I QRS you must have declared a set in every order which you will ever want to see data. While the impact on the data base's size is minimal, it does add measurably to the DMS processing time for an addition when the record being added is the member of many (say eight or more) sets. Version III minimizes the need for sets that facilitate the order of reports by optionally sorting the records between the selection and display steps. Of course, this greatly slows the report phase and requires temporary disk storage, but is desirable when the tradeoff is having the interactive portion of the system run slow.

Version I QRS has an interactive DMS feature that allows you to enter DMS commands directly. It is ideal for modeling and experimenting with access paths before you commit yourself to a large production data base design. It can also be used for limited production work, say to enter data into records that will be used as look-up tables by an application program. It saves you the need of writing an application to load and modify that portion of the data base. Version III QRS, with all its additional features, did not have room for the interactive DMS feature, but a stand-alone interactive DMS program is available.

Both versions of MDBS also allow use of a real-time transaction logging module, RTL. It keeps a copy of every data base transaction in a log file which can be used to back up or selectively restore the data base. Version III has a number of commands which maximize the utility of RTL and provide a "belt and suspenders" level of data base integrity.

Existing data bases can be modified without compromising their integrity. Version I has DRS, which can add or delete records, sets or items, in addition to the more prosaic maintenance of user names and passwords. Version III, at this time, has a lesser DMU utility program, which can only change users and passwords and the like. A data base restructuring program for Version III will be offered in the near future.

The operating environments for the two versions differ somewhat. The DMS routines in Version I must be co-resident in memory with your application program: they take up 20K and use another 4K minimum for buffers. After getting over the initial shock of having to give up that much memory, you begin to ask yourself, "20K?! How did they fit all that into just 20K?" One thing is certain — your application language had better be capable of segmenting or overlays. Version III, with all of its additional features, has the potential to be a real memory killer. Fortunately, the folks at MDBS realized that nobody uses all the features of the DMS at once and provides selective linking, so that you only link in the routines that your application actually uses.

A Look at the Documentation

The documentation won't teach you how to be a data base expert if you're ignorant of the subject, but it will fill in a lot of blanks and give you some good examples and illustrations if you are somewhat versed in the terminology. The manuals are quite thorough and provide good descriptive passages that tell you exactly what each command will do, the associated error conditions and an example of how the call is structured. Advanced capabilities and esoteric features are noted with a vertical bar

The application designer who uses MDBS III has more target machines than any other DBMS, and his application is especially portable if it is written in C.

in the margin next to the descriptive text. This gives you fair warning when you may be about to get in over your head. MDBS gives one- to three-day seminars in various parts of the country on use of their products. They complement the manuals nicely and greatly accelerate the learning curve for a system of this complexity.

What Does it Cost?

MDBS Version I costs about $1500 with all the modules described above. Version III goes for a little more than twice that for the single user micro version, and up to $30,000 for a large UNIX machine. That's enough to give any hobbyist second thoughts about using MDBS for cataloging his library. All that money — and it isn't even easy to use. Well, not easy compared with dBASE II and the rest. The professional applications designer tends not to care so much about front end effort as long as the end result is a robust system with high capability. If it's portable, so much the better. He isn't looking for instant results. But MDBS has cast an eye towards the "instant database" market as well. Due out in the next few months is a front end program which will build screens, edit data and communicate directly to an MDBS data base. Not all that much is known about it right now, but we'll keep you posted.

In Conclusion

Wrapping it all up, MDBS III is one of the most significant software systems offered to date for a microcomputer. Furthermore, it may well become one of the most significant programs for many minicomputers, too. Version III is actually available as two separate products: Version IIIa and Version IIIc. The "a" version is written in assembler for all the popular Z-80, 8080, 8086, and Z8000 microprocessors. There are versions for CP/M, MP/M, PCDOS, CP/M-86, MP/M-86 and OASIS-16. The "c" version is written in the "C" language and is designed for transport to virtually any C environment, including PDP-11, and most UNIX or XENIX machines. The host languages range from assembler to all the Microsoft languages to BDS C to CBasic through the popular Pascals.

As UNIX and C have grown in popularity, so will MDBS III, giving it the potential to become the single most widely used data base. Since the product does not compete with the data management programs mentioned at the outset, what is its competition? How about Total, IDMS, Image, System 2000 and ADABAS? These are all mainframe and minicomputer data base management systems and frankly, MDBS does *more* and is easier to use. Right now, the application designer who uses MDBS III has more target machines than any other DBMS, and his application is especially portable if it is written in C. Given the proliferation of new microprocessors and the blurred distinctions between micros and minis, MDBS looks like not only a survivor, but a strong leader in the software shakeout that the future will surely bring. ■

Total Information Management Database Manager

by Glenn A. Hart

The Total Information Management system (invariably designated by its acronym T.I.M.) is an unusually comprehensive data base management program designed and sold by Innovative Software, 9300 West 110th Street, Overland Park, KS 66210. T.I.M. is designed to run under the CP/M operating system and requires at least 56K of RAM, an 80 x 24 video terminal and a hard copy printer. At least one disk drive is mandatory, and the large amount of code in the system really makes multiple drives essential for convenient use and storage of any serious data.

The newest Revision 3 of T.I.M. (released in December, 1981) is rather different from previous versions. Unlike earlier releases, the Microsoft 5.3 Basic compiler is used and only object code is supplied; source code is no longer available. The system uses the BRUN provision of the new compiler to reduce significantly the size of the compiled object modules.

All data base managers have some restrictions on the data format they will accept. T.I.M.'s limitations are less restrictive than some and not as flexible as others: a data file may contain up to 32,767 records (disk space permitting), but each data record may now contain up to 40 data fields. Each field can be no more than 60 characters long, with a total record limitation of 2400 characters. These limits are significantly greater than in earlier T.I.M. implementations. These record limitations occur in many other DBM's, and the constraints imposed by T.I.M. are not a major problem with most normal applications.

T.I.M. is easily installed in most computer systems with a comprehensive, but easy to use new configuration program. Several common terminals are supported directly, and an interactive dialog can be used to program general system parameters (printer width, number of disk drives, company data for initial prompting and the type of word processor in use), terminal cursor movement routines, terminal attributes (initialization code, if any, video attributes to use for prompting messages, status lines and error messages) and function key codes. This last option defines the cursor control codes to more or less match WordStar, Vector Graphic or custom key sequences. The results of all these initializations are kept in a special data file and can be readily changed at any time.

One of T.I.M.'s strong points is the human engineering that has been incorporated into every aspect of its operation. The program is completely menu driven, meaning that the various options available at any time are clearly displayed. Most commands are single keystrokes, with either upper or lower case accepted. Operator prompts are generally clear and self-explanatory. Such careful design makes for simple user training and easy operation and greatly reduces the need to refer to the program documentation.

At program startup, an initial banner is displayed and the master menu program is loaded. Whenever T.I.M. is loading another program module or is performing some time-consuming function, this fact is displayed on the screen, which helps reduce operator anxiety when it appears that nothing is happening. It can take quite a while for T.I.M. to chain in each new program module because of the large size of the program modules produced by the Microsoft compiler.

The Main T.I.M. Menu is shown in Table 1. Each of the main functions available can be reached through this menu, and each module returns to this menu upon completion of its task. While the listing is alphabetical, we will consider the modules in the order in which they would normally be used.

The first step is to create a file. As with all DBM's, the system must be told how the data files are organized, what each field's name, length and data type are, etc. The new T.I.M. recognizes eight field types: alphanumeric,

Glenn A. Hart, 51 Church Road, Monsey, NY 10952.

numbers, dollar amounts, dates, inverted names, calculated fields, sequential fields and total fields.

The first four are standard, but the last four are not. *Inverted names* are a great convenience which allows entering a name in the normal John J. Jones order but which instructs the system to invert the order to Jones, John J. for sorting purposes. This allows names to be arranged in alphabetical order by last name without the necessity of entering them artificially. If this inversion is not needed or desired, a normal string field can be used instead.

Calculated fields are the result of arithmetic manipulation of two other fields or one field and a constant. Thus a field could be designated TOTAL VALUE and be calculated by multiplying ORDER QUANTITY and UNIT PRICE fields. Calculated and number fields may have up to four decimal place precision.

Sequential fields are numeric fields whose entries are automatically entered by the computer. As each record is added to a data file, the contents of a sequential field are incremented by one. This can be quite useful in assigning consecutive invoice numbers, customer I.D.'s, etc. *Total fields* are also calculated by the system and are the total of up to 16 other fields as defined by the user.

Access to the file can be limited by establishing a four position password. If a password is defined no operations can be performed on a file without providing the correct input. The password can be changed with the File Maintenance utility.

Records are stored on disk in the sequential order in which they are entered. Any field can be designated a Key Field, which means that the individual records can be accessed in the sorted sequence of the key value. The main, or "major," key can be followed by any number of secondary, or "minor," keys to further specify the exact sorting order desired. T.I.M. creates special key files which contain numbers which are pointers to the next record in the sorted order (either ascending or descending) of the chosen key. This "linked list" method provides fast access to any desired record.

While it is tempting to define all fields as keys to handle any possible sorting contingency, a T.I.M. user soon learns that increasing the number of keys results in much slower sorting and merging operations. Most DBM's require that keys be designated only when the file is structured. T.I.M. provides the unusual ability to define keys after the file is defined and data has been entered. A Sort utility is provided for this specific purpose; normally all sorting and merging operations are completely automatic.

Table 1: *Main T.I.M. Menu.*

A = Add/Inspect/Update a record
C = Create a new file
D = Display T.I.M. directory
F = File specifications
H = Help menu
I = word processor Interface
L = List generation
M = file Maintenance
R = Report generation
S = Select records from a file
U = Utility commands
X = eXit to operating system

The process of actually creating the file is quite simple. The user is prompted for the field name, length and type, key status, etc. It is easy to correct errors and make other changes during the creation phase. The new T.I.M. includes a truly excellent screen definition module. Two 80 by 20 screens can be defined, and the user has complete control over screen layout. Entering screen data is extremely easy; the cursor is merely moved to the position desired for the new field. A series of simple commands facilitates the process and comprehensive help screens aid the user at each step.

Table 2: *Add/Inspect/Update Menu.*

A = Add records
C = Change current file
D = Delete current record
F = go to First record
H = display Help menu
K = change Key-field
N = go to record Number
P = Print current record
R = Redisplay a current record
S = Search for a record
T = Toggle screen
U = Update current record
X = eXit to main menu
CR = Step to next record
+n = Jump forward direction
-n = Jump backward direction

Once a file structure has been defined and an actual data file created with the Create module, data can be entered with the Add/Inspect/Update module (see Table 2). The fields to be entered are displayed using the previously defined screen format, with the defined length of each field displayed as a row of dashes. The screen format is well done, and uses extensive cursor movement, prompting and status lines and alternative help menus. Entries are checked for appropriate data type and length. Information from previous records can be used again by entering a Control-L, mistakes can be corrected before a field is completely entered, and the record can be redisplayed after corrections. Any calculated, sequential or totaled fields are computed immediately and displayed on the screen. When record entry is complete the newly entered data are merged into the index files in their proper order. The merging process is surprisingly fast, although more time is consumed as the number of key fields increases.

T.I.M.'s provisions for modifying existing data are flexible and powerful. When the module is first loaded, records are displayed in sequential order, but any key field can be designated as the search field and T.I.M. will then use that sorted order for display. Several commands permit jumping around in the file to specified records, moving forward or backwards any number of records, etc. Any record can be located by searching for specific data in any of its fields. Parts of a field can be input so the entire contents of a field do not have to be entered. A record can be deleted, in which case it is marked as deleted but not removed from the file unless a File Compression utility in the File Maintenance program is used. This allows deleted records to be reclaimed if necessary.

Each record is displayed with its complete field titles and the specific data for that record. In addition, the file name, both the actual and relative record number, status (deleted or non-deleted), direction (forward or backward movement), the search mode (sequential or key-field order), screen number and field type and length are displayed. The Beginning-of-File or End-of-File records are also so indicated. This information is very helpful in keeping track of what is happening.

Updating a record is as simple as entering "U" and changing the data in any field. Fields which are to be left unchanged are merely skipped over with a carriage return. If any records are updated, T.I.M. automatically merges the new information into the data file at the completion of the editing process.

An important role of a DBM is to selectively extract specific data from a large data base. T.I.M.'s Select Records from a File module performs this function. Extracted data can be displayed on the user terminal, printed on the system list device or routed to a separate file containing only the desired data. If printed data are requested, T.I.M. prompts for the fields the user wishes to print and outputs a nicely tabulated report. Search criteria can be stored in a Library file for re-use in performing similar extractions in the future or on other data files.

The search criteria can be built using eight types of relational statements. Deleted records or non-deleted records can be specified; these are convenient in determining which records have been marked for deletion prior to compressing the file or for listing only records which have not been so marked. The designated field can be compared to either a constant or to the contents of another field using the relational operators GT (greater than), LT (less than), EQ (equal to), NE (not equal to), GE (greater than or equal to), LE (less than or equal to) and BV (between two values). Sub-fields, indicated by the starting and ending positions of a string, can be related to a constant, and a string can be searched to determine if it contains a specified constant. These commands can be combined in any complexity, and provide a very flexible searching and extraction facility.

Once all the data has been entered, updated, extracted, etc., the output most often desired is either a report of some kind, mailing labels or form letters.

Mailing labels are created with the List Generation module (a slightly confusing nomenclature). Up to four labels across can be produced. A label is considered a block of text, and T.I.M. allows defining a block in many useful ways. Each line of a block can contain any field from the data file, truncated to a user specified length if desired, or a constant string for identification purposes. The tabular position of a field or string on each line can also be specified to allow more than one field to appear on each line of a block. A list format can be stored in a List Library for future use.

The normal tabular report generation module is equally powerful. The user can specify which fields to include, constant or variable (entered at the time the report is generated) titles and several other factors. T.I.M. will automatically determine the best tabulation and form layout. Twenty different fields may be summarized and evaluated using a broad spectrum of analytical tools, including sub-totals, totals, grand-totals, record counts, etc. These summarizations can be specified for major and minor break points on each of the summarized fields. As usual, a Report Library stores report formats for review and future use.

The Report program in the new T.I.M. can access a secondary data file in addition to the main file. The system requires that there be an identical "linkage field" in both files to synchronize access to the appropriate records in both files. This capacity allows T.I.M. to handle certain applications previously impossible.

Earlier versions of T.I.M. included a slow and idiosyncratic word processor. While it could be used to produce normal correspondence and other text, its main application was to produce form letters with data merged from T.I.M. data files. Innovative Software realized the limitations of this module and the probability that most users of T.I.M. would have their own word processor with which they were already familiar, and would probably prefer to use. Thus the T.I.M. word processor is no longer provided. In its place is an interface module which converts T.I.M. data files to formats directly usable by WordStar's MailMerge module and Vector Graphics Memorite III. Other formats can be handled as well.

The other modules listed on the Master Menu are mainly housekeeping utilities. T.I.M. maintains a special directory of files which it recognizes; this is *not* the same as the directory of the files on the disk itself. The "D" command allows inspection of this T.I.M. directory. The "F" command displays three screens of information on selected data files, including field definitions, date created, key fields, calculated fields, etc. The File Maintenance module (see Table 3) handles renaming or deletion of files, displaying the disk directory, changing or eliminating passwords, removing records with duplicate keys, changing the titles of specific fields and compressing a file by removing records which have been tagged for deletion. This module also can sort files. Sorting is not normally necessary, since T.I.M. automatically indexes data, but the free-standing sort provision can be used to add key fields after file definition. This is an unusual and useful provision not often found in DBM's.

The T.I.M. utilities (Table 4) include several very powerful commands as well as some commands which merely shuffle T.I.M. files around (necessary because of the special T.I.M. directory, which prohibits normal movement of T.I.M. files). The backup program considers the disk storage capacity of the destination drives and makes provisions for any necessary disk swapping, and allows either segmenting data files into smaller units or concatenating files with equivalent formats together. Most powerful of all is the Restructure command, which allows modification of practically any data file parameters. It works by moving the data in a file to a new file with a completely different data structure, with provisions for reassignment and data type conversions along the way. This is quite unusual and gives the user an added degree of confidence when designing files; if something is forgotten or needs change in the future T.I.M. can accommodate the change without destroying existing data.

Table 3: *File Maintenance Menu.*

A = Alter sequential count
C = Compress a file
D = Display disk directory
E = Erase a key-field
K = Kill a T.I.M. file
M = reMove duplicate records
P = Password update
R = Rename a T.I.M. file
S = Sort a file
U = Update field titles
X = eXit to main menu

Table 4: *T.I.M. Utilities Menu.*

A = ASCII file to T.I.M. conversion
B = Back-up or concatenate T.I.M. files
C = Convert T.I.M. 1.xx or 2.xx files to T.I.M. 3.xx
R = Restructure T.I.M. data files
T = Transfer T.I.M. file between disks
X = eXit to main menu

User Evaluation

T.I.M. is a comprehensive software system which performs its data base management tasks smoothly and efficiently. Prompting and on-screen menus and information have been extremely well designed, so it is remarkably easy to use the system, even without reference to the documentation. This is not often the case with such complex software, and Innovative Software is to be commended on its careful attention to human engineering. The documentation deserves special praise; it is very well organized and helpful (although a complete index would be nice).

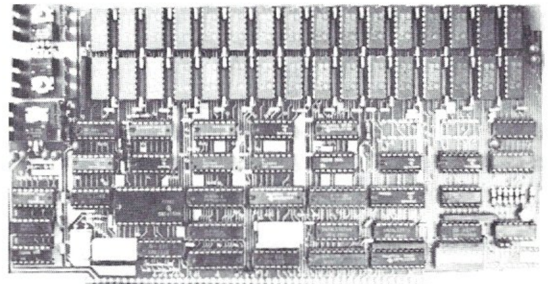
Innovative Software has upgraded the program significantly over earlier versions. Extensive use of cursor addressing and screen attributes adds professionalism and makes the program even easier to use. T.I.M. is probably the best of the available microcomputer data base managers in its human engineering.

I think the overwhelming majority of micro users would find T.I.M. more than powerful enough for all their intended applications. T.I.M. is still basically a single data file system, although the linking feature of the report generator allows at least some access to a secondary file. T.I.M. also restricts the user to the provisions which have been included by the program designers; there is no applications development language or any other way to modify or extend the program (although source code is provided for the experienced programmer so T.I.M. files can be accessed or manipulated by custom Basic programs).

Previous T.I.M. versions represented a very good price/performance ratio. With T.I.M. now listing at $695, the enhancements made have not come cheaply, but I think T.I.M. is still fairly priced in light of its excellent performance.

256K RAM IN 4K BLOCKS

1/4 Megabyte $1495



OVERVIEW

The BSR 64/256 is an 8 bit bank selectable dynamic random access memory card designed to operate in a Z-80 based S-100 computer system with a CPU clock frequency of up to 4 MHz(A model) or 6 MHz(B model).

Individual 64K banks are selected via the IEEE 696 8 bit address us extension. If the host system is not capable of driving the extended address bus, one of the BSR 64/256 cards in the system may be configured to drive it through an onboard latched output port.

System area is allocated in 4K blocks by writing a system mask out to two latched output ports. Another port allows any one of up to eight cards to be assigned as the current system master. Logically, up to 64 cards may be addressed in a single computer system.

Assembled & Tested Boards

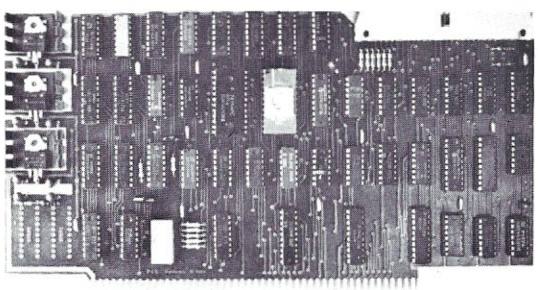
(4164)	64k	$770
	128k	$1015
	192k	$1260
	256k	$1495

Memory prices fluctuate, call for price

FEATURES

MAX STORAGE	256k bytes (32 ea. 64k X 1 chips) or 64k bytes (32 ea. 16k X 1 chips)
BLOCK SELECTION	Any combination of 4k blocks in any 64k bank, software selectable
SYSTEM AREA	Any combination of 4k blocks in low order bank of current system master card, software selectable
SYSTEM MASTERS	Any one of up to 8 cards software selectable, one card jumper selectable for system power-up or reset
CARDS/SYSTEM	Logically, up to 64 64k or 256k cards
BANK SELECTION	Uses or implements IEEE 696 (S-100) extended address bus
CPU SPEED	4 MHz or 6 MHz with no wait states
REFRESH MODE	Invisible
OUTPUT PORTS	4 Consecutive ports for entire system, selectable on any 4 port boundary

14″ WINCHESTER FIXED DISK CONTROLLER & SUBSYSTEM



FDC 4000

14″ Winchester Fixed Disk Controller

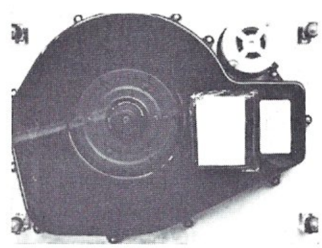
Controls Shugart 4000 & 4100 series drives
Up to 174 megabytes per controller
I/O mapped buffer - no wait states
Includes MP/M* XIOS & CP/M* CBIOS

Assembled & tested $895

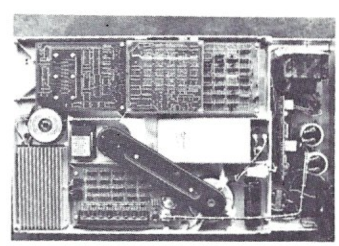
29 Megabyte Disk Subsystem $3840

The fixed disk subsystem includes a 29 million byte Shugart fixed disk (SA 4008), and an assembled and tested S-100 disk controller (FDC 4000). Also included is an unfinished rack mountable aluminum case, punched and ready for assembly. To complete this kit we have included power supply, fan, wiring harness, signal cable, and MP/M* XIOS and CP/M* CBIOS.

(SA 4008)



(Assembled Subsystem)



PCC SYSTEMS

"Contact Us For More Information"

5232 Manzanita Ave. Carmichael Ca 95608 (916) 338-5454

Price subject to change

*Trademark of Digital Research

CBS, MDBS and MOPIB*: Three Ways to Implement a Mailing List

by Bill Machrone

With all of the packages and applications available to manage a mailing list, would you believe that I'm still searching for the perfect one? A bit of background is in order: Back in 1975, before I had my first microcomputer system, I managed a club's mailing list with stolen time on an IBM 370. The system was strictly batch, with keypunch card input, but a flexible little report generator (GRS, for those of you who may of heard of it) gave me mailing labels and membership rosters in pretty much any format I wanted. I lived in constant fear of the EDP auditors and yearned for a system that I could access on-line, at my leisure and which would give me exactly the reports I needed.

By 1977, when I had acquired my first micro, home implementation of that mailing list was one of my goals. When I entered the microcomputer field as a business, my major systems were extended mailing list management systems, with interfaces to word processing software and (usually) the ability to keep track of the number of times and reasons that a given address in the mailing list was used. This is where the commercially available products fell short and caused me to write my own software.

That first system was a TDL Xitan with 16K and cassette tape. After the usual spate of game programs I began work on my first mailing list system in earnest. It was written in TDL Basic and held a sorted array of new records in memory, then read the master file from one cassette and rewrote it with the new records in the right places on a second cassette. It was painfully slow, but it was mine. I never actually produced labels from that first system, because by the time I got the bugs out I had upgraded to disk. The initial disk-based system was only marginally better than tape because the Basic I was using regarded disk as nothing more than a fast cassette machine. Random access was out of the question.

Then came Microsoft Basic. I rewrote the system based on fixed-length records, stored on disk in alphabetic order. The program appended new records to the end of the file and tagged ones to be deleted. The entire file was then sorted by MicroPro's Super Sort. Super Sort could easily be the topic of an entire article, as it is unquestionably the finest sort utility available for the CP/M environment. I wrote a binary search subroutine that permits random access by name and was somewhat disappointed in its speed in traversing a 600 record file. So I improved it by doing a look-ahead at program startup which divides the file into four quartiles which can then be binary searched (see Figure 1). A larger file can be subdivided even further to keep access speed high. The important thing in keeping retrieval speed high is to eliminate the first several long seeks by storing the sort keys and record numbers of reference records in memory. Quartiles have been adequate for files with 1500 250 byte records to date, with access speed on the order of two to three seconds, worst case.

The file management system described in Figure 1 works well and is still in daily use. The only real drawback to it is reporting. Each time you need a different report a program must be written, or at least modified. It may also entail resorting the master file if a different order is required. On the other hand, the reports and output files that you create yourself have exactly the format that you require — no need to compromise. Another drawback is

*My Own Program in Basic

Bill Machrone, Box 291, Fanwood, NJ 07023.

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

Z80* Disk Software

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

— Carl Galletti and Roger Amidon, owners.

Software with Manual/Manual Alone

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

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

BASIC I

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

BASIC II

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

BUSINESS BASIC

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

ZEDIT

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

ZTEL

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

TOP

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

MACRO I

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

MACRO II

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

LINKER

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

DEBUG I

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

DEBUG II

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

ZAPPLE

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

APPLE

8080 version of Zapple $34.95/$15.

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

TPM*

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

SYSTEM MONITOR BOARD (SMB II)

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

ROM FOR SMB II

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

PAYROLL (source code only)

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

ACCOUNTS PAYABLE/RECEIVABLE (source code only)

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

GENERAL LEDGER (source code only)

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

C BASIC 2

Required for Osborne software. $99.95/$20.

SYSTEM/6

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

$160.

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

$160.00

ORDERING INFORMATION

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

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

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

For information and tech queries call
609-599-2146

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

OEMS

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

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

Prices and specifications subject to change without notice.

DEALER INQUIRIES INVITED.

COMPUTER DESIGN LABS
342 Columbus Avenue
Trenton, N.J. 08629

that Basic does not easily support keyed access to files. You can program around it, but what the heck, high level languages are supposed to make things easier, not tougher.

This caused me to become interested in the Configurable Business System (CBS) from DMA Associates, marketed by Lifeboat Associates. I was fortunate to receive an evaluation copy of CBS, so the mailing list application constitutes a "hands-on" review of the package. CBS is a series of Fortran programs that interact with the user to create files, menus and updating procedures. It also includes a general purpose report generator. The file organization is ISAM, which keeps the need for sorting to a minimum. The programs are entirely menu-driven, with comprehensible error messages. The documentation is also quite good, heavily laced with examples and definitions. If you have any prejudices against Fortran, I'd like to state for the record that there is no negative impact on the end result.

The user interaction is logical in its flow. You begin by selecting a name for the master file, then proceed into naming each field in the file, specifying length, type of data, minimum and maximum number of characters acceptable for entry, etc. You then create a main user menu that controls selection of all the options available to the user. CBS makes a sharp distinction between "designer" and "application" functions, permitting a programmer to set up a series of applications for a user, then remove the utilities that create or modify menus, files and indexes. You don't have to be a programmer to design, but it will increase productivity if you know something about CP/M file structures and operations. One potential cause for confusion on the part of the non-CP/M initiated user is the multiplicity of files generated and maintained by CBS. While they are explained in the manual, the non-programmer may be distressed by the apparent complexity when, in fact, there is no need for him to know the number or purpose of the files under most circumstances.

Figure 1: *Look-ahead binary search in Microsoft Basic.*

```
4000 'Pre-read file to determine location of quartiles.
4005 'Executed once at beginning of program.
4010 GET #1,1                  'First record has total records on file.
4020 NUMRECS = VAL(NUMRECS$)
4030 IF NUMRECS = 0 THEN NUMRECS = 1 'Initialize new files only.
4040 LO.POINTER = 2: HI.POINTER = NUMRECS
4050 INT.POINTER = 1
4060 MID.POINTER = (LO.POINTER + HI.POINTER) /2 'Get middle of file.
4070 GET #1,MID.POINTER
4080 'The field on which the file is sorted is ELEM$(6).
4090 MID.PNAME$ = ELEM$(6)
4100 QT1.POINTER = (LO.POINTER + MID.POINTER) /2 'First quartile.
4110 GET #1,QT1.POINTER
4120 QT1.PNAME$ = ELEM$(6)
4130 QT3.POINTER = (HI.POINTER + MID.POINTER) /2 'Third quartile.
4140 GET #1,QT3.POINTER
4150 QT3.PNAME$ = ELEM$(6)
4160 RETURN

5000 'Predetermine quartile of file.
5010 'Parameter passed from "find" routine is REQUEST$.
5020 COMPARE$ = SPACE$(16)  'Set up comparison field.
5030 LSET COMPARE$ = REQUEST$
5040 IF COMPARE$ < QT1.PNAME$ THEN
          HI.POINTER = QT1.POINTER:
          LO.POINTER = 2:
          RETURN
5050 IF COMPARE$ > QT3.PNAME$ THEN
          LO.POINTER = QT3.POINTER:
          HI.POINTER = NUMRECS:
          RETURN
5060 IF COMPARE$ < MID.PNAME$ THEN
          LO.POINTER = QT1.POINTER:
          HI.POINTER = MID.POINTER:
          RETURN
5070 IF COMPARE$ => MID.PNAME$ THEN
          LO.POINTER = MID.POINTER:
          HI.POINTER = QT3.POINTER:
          RETURN

6000 'Execute binary search.  INT.POINTER gets as close as possible
6010 'to the requested record.  FOUND = 1 on match, 0 on no match.
6020 INT.POINTER = (LO.POINTER + HI.POINTER) /2
6030 GET #2,INT.POINTER
6040 IF COMPARE$ = ELEM$(6) THEN
          FOUND = 1:
          RETURN
6050 IF COMPARE$ < ELEM$(6) THEN
          HI.POINTER = INT.POINTER - 1
6060 IF COMPARE$ > ELEM$(6) THEN
          LO.POINTER = INT.POINTER + 1
6070 IF HI.POINTER < LO.POINTER THEN
          FOUND = 0:
          RETURN      'No Match.
6080 GOTO 6020        'Loop until match or definite no match.
```

I had no difficulty setting up a menu and file for my mailing list. Retrieval via the index was slightly faster than that of my random access Basic program. Entries, changes and deletions are all handled easily by the CBS utility programs. Deleted records are simply flagged on the file and removed at a later time by the CBS Cleanup utility. Before this is done, a deleted record may be restored to active status via a command in the update program. Another utility from the designer menu gave me a concise printed summary of the fields, lengths and attributes in my file. This summary is a necessary tool for formulating reports with the report generator.

CBS has several capabilities that most marketers would tout as "relational" (whatever that means). In addition to creating multiple indexes per file, files can point to other files via indexes. An option in the update specification program allows entry of a field to cause changes in multiple (up to 20) files. Once the indexed relationship is established, the flow of information can go either way. That is, an existing file can be the source of information for the record you are entering, permitting "fill in the blank" operations or table lookups. You have to be careful in using this feature, though, because the update inquiry program can only access and display one file at a time despite the presence of links to other files. The system can also create and read external ASCII files. This provides a form of communication with other steps in CBS processing or with external programs. The manual assured me that the output format of carriage return-line feed delimited fields (one per line) was compatible with most word processing software and programming languages. More on this later.

Armed with the printed summary of field lengths, names and what-not, report generation is an easy task. CBS provides the capability to create and store a report format or to modify an existing one. You can then call up the desired report by selecting it from the menu. The report generator is powerful enough for most purposes, and is better than most at formatting, since it permits alteration of the number of blanks preceding or trailing a field. Further, it permits you to specify the number of lines to skip after a field is printed. With these two features plus calculation-derived work fields, you can move to any X-Y coordinate on a printed page, making it easy to set up preprinted forms. It allows three levels of control subtotals, sufficient for most applications.

One of the places where CBS (as with most other data managers) is deficient is in the inability to specify input edits. The program I had written makes a number of validity checks on data, preventing most errors due to carelessness. It's easy for garbage to creep into your files without tight edits, especially when the system is to be used by an operator several times removed from caring about the effects of bad data. I had also programmed some automatically derived fields into my system, so that when you enter the addressee name, say "Mr. John Smith," the salutation field is automatically filled in with "Mr. Smith" and the user is able to override it with "John" or "Jack" if the individual is known personally. Bells and whistles of this sort are impossible to accommodate in a general purpose data management system, so you have the old trade-off: exact fit and function versus fast and easy implementation.

Neither of these limitations in any way makes CBS a bad system. On the contrary, it is quite a good system. The specific problem that I had with it was its inability to create an output file suitable for WordStar mail merge. All of my clients are using WordStar, with great success. Introduction of another text editor for merge operations would throw a monkey wrench into the works. For those of you not familiar with WordStar's mail merge system, it can read ASCII files comprised of fields and records and merge the fields into a document as it prints. It requires fields to be separated by commas, and records to be separated by carriage return, line feed sequences. Furthermore, if a field contains an embedded comma it must be surrounded by quotation marks to prevent confusion. An example is the company name field, which often takes the form, "Twit Engineering, Inc." Of course, this is not a limitation when programming in Basic, since this is the way Basic likes to write its sequential files. But CBS can't produce that format, so it would have required a conversion program to reformat the file with commas and quotation marks. I chose to stick with my trusty old Basic program.

At the same time, however, I became enamored of MDBS, the Codasyl data base management system produced by Micro Data Base Systems, Inc. Called from a host language such as Basic or PL/I, it handles all the storage and retrieval of records, files and data fields. It permits the user (programmer) to set up highly complex relationships between record types (subfiles) which are linked by sets (pointers). Unlike ISAM-based data management systems, MDBS stores *everything* in one big file. An advantage is simple backup — one PIP will do it. The disadvantage is that there's bound to be some wasted space in that file, so be prepared to use lots of disk space. MDBS also includes utility programs for creation of the data base, logging of transactions, restoration of the data base, modification of the data base and query/report generation.

My first system using MDBS was fairly easy. I just took my tried and true mailing list system and replaced all of the disk I/O routines written in Basic with equivalent calls to MDBS. After getting around some of the hassles of interfacing with Microsoft Basic it worked fine. It is difficult or impossible to load or unload an array directly from the data base because MBASIC keeps moving the arrays around in memory as it performs "garbage collection." MDBS shares available memory with the host program and uses as much as is available to buffer incoming and outgoing data. The result is that accesses such as getting the next record or finding (binary searching) a record that is fairly close to the current record is quite fast, often without any disk access.

As with CBS, I found that you have to be careful in using some of MDBS's features. For example, an address record can belong to many sets, each acting as an index that maintains a sort in a different order. If you have too many of these, say eight or so, MDBS begins to spend a lot of time updating pointers as new records are entered. By "a lot of time" I mean two or three seconds on a single density 8" diskette, one to two seconds on a double density diskette with 1K sectors. This is certainly not bad performance, especially if you have worked with online interactive systems on mainframe computers, which are usually much slower. But micros spoil us because we generally get what we want when we want it. The delay, however, can be disconcerting to a data entry operator who wants to get as many records in as possible in the shortest time frame. I tried the same program on my Godbout 6 MHz system with a Morrow M10 hard disk, and the delay just about disappeared. Remember, even with the slower disk accesses, *all* file maintenance is done at this point. There is no need for a maintenance step after the interactive session with the user.

MDBS's main attraction to you as a programmer is that it relieves you of nearly all the hassle of storing the data once you have decided how each data item relates to every other data item. Then you run the DDL (Data Definition Language) program to tell MDBS about it. The technology level of DDL's user interaction is far below that of most other products on the microcomputer scene. The input is based on "card" images, where column position is critical, yet there is no provision for automatically prompting or tabbing to the correct column position for field entry. There is a "mask" display capability that shows you where everything should go for each card type, but it is a primitive method. DDL has a built-in line editor that shares the evils of every other line editor in the world. Where DDL shines is in analyzing the data base you are trying to create, and giving you helpful error messages so that it can create a properly initialized data base.

Once you get used to traversing the records and sets of an MDBS data base you may never be satisfied with anything else again. There are no practical limits on how the data can be stored and how it can be retrieved. One valid criticism of MDBS is that it requires the user to be somewhat familiar with the structure of the data base. This is not a limitation in the application programs because the programmer takes care of traversing the sets to get at data. Retrieval is really what it's all about. While MDBS's QRS query/retrieval system is probably the best report generation system of its type available for micros, it's not perfect. It is good enough for most purposes, however. The problem with queries is that you have to tell QRS how you want to traverse the data base. No problem for the designer, since he knows exactly what fields are in which records, and what sets connect the records in what sequence. But the non-programming user is likely to have limited success in traversing a complex data base. This can be alleviated to some extent by use of QRS'S macro/synonym capability. It permits the substitution of a key word for an entire phrase. The system designer can set up a bunch of useful phrases that will execute some of the more difficult path excursions and simply provide a list of key words to the user. A more satisfactory solution would be a preprocessor that prepared a QRS inquiry from interactive screens presented to the user. Another limitation of QRS is its lack of control over over print formatting. One approach is to include a record in each data base that is filled with spaces, line feeds and tabs for inserting into the printed output, but it is at best a clumsy and incomplete solution.

So my MDBS implementation finally gave me the perfect mailing list system, right? Well, almost. QRS provides a disk output option but, like CBS, does not produce a file compatible with WordStar's mail merge system. So I was faced once again with the option of writing a conversion program to create a proper file with quoted strings, deblanked fields and delimiters, and writing the extraction program in Basic or PL/I. I elected to write the conversion program because the power of QRS is just too good not to use. This way, the user can test his retrieval criteria with QRS, then invoke the disk output option and conversion program. Additionally, QRS is much more flexible than any extraction or inquiry program that I could write.

By the way, virtually all the quibbles I have had with MDBS (except for automatic or easier pathfinding) have been resolved or greatly improved in MDBS III. It is so radically changed that it is considered a new product rather than an upgrade, yet MDBS I systems are upward-compatible with it. It's breathtakingly expensive, designed as it is for the OEM or systems house, but it's *good*.

So here I sit with a mailing list system that has evolved continually over five years and, while it is pretty good, it isn't perfect. Will it ever be? Probably not. But I'll keep on improving it and trying new technology as it becomes available. ■

Cursor Addressing

by Fred L. Gohlke

A technique for cursor addressing on 16 x 64 memory mapped displays in 8080 Assembly Language.

Display manipulation is normally handled by a system monitor. However, in some cases, the monitor imposes limitations the programmer would like to circumvent. Of these, direct cursor addressing can be one of the most cumbersome to implement. Understanding the numeric relations involved in a 16 x 64 screen may help in solving the problem.

It is important to notice that each 64 character line is equal to one quarter of a 256 byte block of memory, and each 256 byte block is equal to one quarter of the total screen memory. Thus, the 16-bit address of the cursor may be viewed like this:

```
        HIGH ORDER BYTE     LOW ORDER BYTE
           111111 11           11 111111
AREAS:       A     B            C     D

Area A - 6 Bits - Screen Base Address
Area B - 2 Bits - Quarter Screen Number
Area C - 2 Bits - Line in quarter screen
Area D - 6 Bits - Character position
```

With this representation in mind, you can write a short routine to calculate screen addresses. The routine will require three or four one-byte parameters, depending on whether the system has hardware scroll capability. The example uses four parameters: character position; line number; scroll offset; and base address of screen memory which are stored in the DE and HL registers:

```
                PARAMETER STORAGE

LINECHAR   EQU   $
CHARPOS    DB    0      Set in 'E' register
LINENUM    DB    0      Set in 'D' register

BASOFSET   EQU   $
OFFSET     DB    0      Set in 'L' register
BASE       DB    0nnH   Set in 'H' register
```

Systems without hardware scrolling can keep a zero in the OFFSET byte.

Fred L. Gohlke, 1000 Blair Rd., Carteret, NJ 07008.

This routine requires only 22 bytes, plus four bytes for parameters. Once understood it provides a simple, efficient way of calculating screen addresses. The following is the routine:

```
                  ORG    nnnn
         CURSOR   LHLD   LINECHAR
                  XCHG
         ADDRESS  LHLD   BASOFSET
                  MOV    A,D
                  ADD    L
                  RRC
                  RRC
                  MOV    L,A
                  ANI    3
                  ADD    H
                  MOV    H,A
                  MOV    A,L
                  ANI    0C0H
                  ADD    E
                  MOV    L,A
                  RET

     LINECHAR     EQU    $
     CHARPOS      DB     0
     LINENUM      DB     0
     BASOFSET     EQU    $
     OFFSET       DB     0
     BASE         DB     0nnH
```

Entry at CURSOR calculates the cursor address from the stored parameters, loads the line number and character position, and exchanges them into the DE register pair, with line number in D and character position in E.

```
     CURSOR     LHLD   LINECHAR
                XCHG
```

FRED GOHLKE has worked with microcomputer software development for five years. He has two micros which he uses primarily for software development and two others which he uses in his truck dispatching business. During his free time Fred teaches, writes articles, and helps others work through the problems of setting up small micro systems. Fred actively participates in the Amateur Computer Group of New Jersey.

Cursor Addressing, continued...

Entry at ADDRESS calculates the cursor address from the values preset in the DE register pair, which must conform to the convention outlined above. At entry, the screen base address is loaded into the H register, and the scroll offset into the L register.

```
ADDRESS    LHLD   BASOFSET
```

The line number is brought into the A register, and the scroll offset is added to it. Line number and offset are counted from zero and the screen contains 16 lines, so the highest legal value for either is 15. Since four bits represent a base-16 number, the right-most four bits of their sum are all that are significant for identifying the actual line number.

```
MOV    A,D
ADD    L
```

Now that the line number has been isolated, rotate the accumulator right two positions to divide it by four. The whole number portion of the quotient (the two low order bits in the accumulator) represents the quarter-screen in which the cursor is located, and the remainder (the two high order bits in the accumulator) represents the line number in the quarter screen.

```
RRC
RRC
```

Save a copy of the value in the accumulator in the L register.

```
MOV    L,A
```

Isolate the quarter screen by ANDing away all but the two low order bits of the accumulator.

```
ANI    3
```


Add the base address of the screen memory, which is held in the H register, to the quarter screen.

```
ADD    H
```

This completes the high order byte of the screen address, so place it in the H register.

```
MOV    H,A
```

Recover the value we saved in the L register, and isolate the line number in the quarter screen by ANDing away all but the two high order bits.

```
MOV    A,L
ANI    0C0H
```

Add the character position, which is held in the E register, to the line number in the quarter screen.

```
ADD    E
```

This completes the low order byte of the screen address, so place it in the L register. The HL register pair now holds screen addresses of the cursor, and the routine ends.

```
MOV    L,A
RET
```

The Converse

In cases where the cursor address is known, but the line number and character position aren't (such as light pen returns), extracting them from the address is a snap.

Entry at FINDPARAMS calculates the cursor parameters from the address in the "HL" register pair, and returns the character position in the "E" register, and the line number in the "D" register.

```
FINDPARAMS  MOV    A,L
            ANI    3FH
            MOV    E,A
            DAD    H
            DAD    H
            MOV    A,H
            ANI    0FH
            MOV    D,A
            RET
```

Bring the low order byte of the address into the "A" register, and AND away the two high order bits. The remainder is the character position on the line, place it in "E."

```
FINDPARAMS  MOV    A,L
            ANI    3FH
            MOV    E,A
```

Shift the "HL" register pair left two positions (multiply "HL" by four) to place the line number in the "H" register.

```
DAD    H
DAD    H
```

Bring the line number into the "A" register, and AND away the four high order bits. The remainder is the line number, place it in "D," and return.

```
MOV    A,H
ANI    0FH
MOV    D,A
RET
```

That's all there is to it — provided, of course, the offset is zero. If you're trying to determine the line number while scrolling, please ask your psychiatrist's permission before writing to me for help. ■

JADE Computer Products
3313 S. Bristol St.
Santa Ana, Calif.
714-549-7108

JADE Computer Products
1291 W. El Camino Real
Sunnyvale, Calif.
415-965-7980

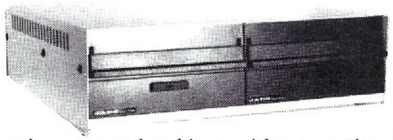
JADE Computer Products
4344 Convoy Street
Kearny Mesa, Calif.
714-268-4661

PLACE ORDERS TOLL FREE.

Continental U.S. - 800-421-5500
Inside California - 800-262-1710
Los Angeles Area - 213-973-7707

JADE Disk Sub-Systems

Shugart, Siemens, Qume



Handsome metal cabinet with proportionally balanced air flow system • Rugged dual drive power supply • Power cable kit • Power switch, line cord, fuse holder, cooling fan • Never-Mar rubber feet • All necessary hardware to mount 2-8″ disk drives, power supply, and fan • Does not include signal cable

Dual 8″ Subassembly Cabinet

END-000420	*Bare cabinet*	$59.95
END-000421	*Cabinet kit*	$225.00
END-000431	*A & T*	$359.95

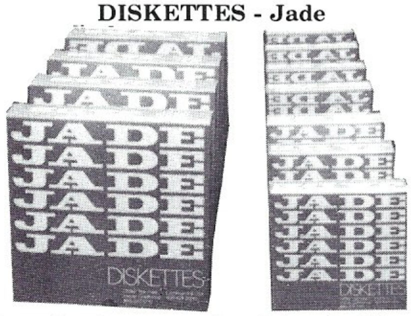
8″ Disk Drive Subsystems Single Sided, Double Density

END-000423	*Kit w/2 FD100-8Ds*	$924.95
END-000424	*A & T w/2 FD100-8Ds*	$1124.95
END-000433	*Kit w/2 SA-801Rs*	$999.95
END-000434	*A & T w/2 SA-801Rs*	$1195.00

8″ Disk Drive Subsystems Double Sided, Double Density

END-000426	*Kit w/2 DT-8s*	$1224.95
END-000427	*A & T w/2 DT-8s*	$1424.95
END-000436	*Kit w/2 SA-851Rs*	$1295.00
END-000437	*A & T w/2 SA-851Rs*	$1495.00

DISKETTES - Jade



We proudly put our name on these high quality diskettes guaranteed to satisfy you or your money back.

5¼″ Diskettes, Box of Ten

MMD—5110103	*SS, SD, 01S*	$29.00
MMD—5111003	*SS, SD, 10S*	$29.00
MMD—5111603	*SS, SD, 16S*	$29.00
MMD—5120103	*SS, DD, 01S*	$31.00
MMD—5121003	*SS, DD, 010*	$31.00
MMD—5121603	*SS, DD, 16S*	$31.00

8″ Diskettes, Box of Ten

MMD—8110103	*SS, SD, 01S*	$31.00
MMD—8120103	*SS, DD, 01S*	$39.00
MMD—8220103	*DS, DD, 01S*	$48.00

EPROM ERASER - Spectronics

Ultra-violet EPROM erasers

XME-3100A	*With out timer*	$69.50
XME-3101	*With timer*	$94.50
XME-3200	*Economy Model*	$39.95

16K MEMORY UPGRADE

Add 16K of RAM to your TRS-80, Apple, or Exidy in just minutes. We've sold thousands of these 16K RAM upgrades which include the appropriate memory chips (as specified by the manufacturer), all necessary jumper blocks, fool-proof instructions, and our 1 year guarantee.

MEX-16100K	*TRS-80 kit*	$25.00
MEX-16101K	*Apple kit*	$25.00
MEX-16102K	*Exidy kit*	$25.00

16K RAM CARD - for Apple II

Expand your Apple to 64K, 1 year warranty

MEX-16700A	*Save $125.00 !!!*	$74.95

Z-80* CARD for APPLE

Two computers in one, Z-80 & 6502, more than doubles the power & potential of your Apple, includes Z-80 CPU card, CP/M 2.2, & BASIC-80*

CPX-30800A	*A & T*	$299.95

8″ DISK CONTROLLER

New from Vista Computer, single or double sided, single or double density, compatible with DOS 3.2/3.3, Pascal, & CPM 2.2, Shugart & Qume compatible

IOD-2700A	*A & T*	$499.95

2 MEGABYTES for Apple II

Complete package includes: Two 8″ double-density disk drives, Vista double-density 8″ disk controller, cabinet, power supply, & cables, DOS 3.2/3.3, CP/M 2.2, & Pascal compatible.

1 MegaByte Package	*(Kit)*	$1495.00
1 MegaByte Package	*(A & T)*	$1695.00
2 MegaByte Package	*(Kit)*	$1795.00
2 MegaByte Package	*(A & T)*	$1995.95

Apple-CAT - Novation

Software selectable 1200 or 300 baud, direct connect, auto-answer/auto-dial, auxiliary 3-wire RS232C serial port for printer.

IOM-5232A	*Save $50.00!!!*	$334.95

DISK DRIVES - Micro Sci

Inexpensive disk drives for your Apple

A2 *Direct replacement for Apple Disk II, works with Apple II controller as first or second drive.*

MSM-123101	*Micro Sci A2*	$429.95
MSM-123101C	*A2 with contrlr*	$499.95

A40 *40 track drive for Apple II. Improved storage capacity and speed over Apple Brand drives - requires Micro Sci controller.*

IOD-2340A	*Micro Sci A40*	$399.95

A70 *70 track drive for Apple II. Twice the storage capacity and three times faster than Apple Brand drives - requires Micro Sci controller*

IOD-2370A	*Micro Sci A70*	$499.95

Micro Sci Controller *Disk controller for up to two Micro Sci A40 or A70 disk drives, DOS 3.2, 3.3, Pascal, and Z-80 SoftCard compatible, includes utility disk and 40/70 track patch.*

IOD-2300A	*Micro Sci controller*	$95.00

VISION 80 - Vista Computer

80 column x 24 line video card for Apple II, 128 ASCII characters, upper and lower case, 9 x 10 dot matrix with 3 dot descenders, standard data media terminal control codes, CP/M Pascal & Fortran compatible, 50/60 Hz

IOV-2400A	*Vista Vision 80*	$375.00

Joystick - T G Products

A better joystick for your Apple II

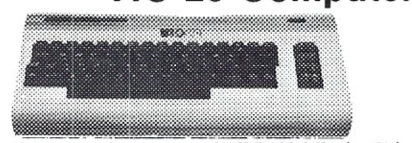
SYA-1512A	*A & T*	$49.95

CPS MULTICARD - Mtn. Computer

Three cards in one! Real time clock/calendar, serial interface, & parallel interface - all on one card.

IOX-2300A	*A & T*	$179.95

commodore VIC-20 Computer



Complete personal computer with 5K RAM, full color, 61 key keyboard, 4 dual special-function keys, serial ports, cassette port, composite video output (connects to standard color TV set), BASIC language, & expansion port.

COM-VIC20	*VIC-20*	$279.95

Z-80 STARTER KIT - SD Systems

Complete Z-80 microcomputer with RAM, ROM, I/O, keyboard, display, kludge area, manual, & workbook

CPS-30100K	*KIT*	$299.95
CPS-30100A	*A & T*	$469.95

AIM-65 - Rockwell

6502 computer with alphanumeric display, printer, & keyboard, and complete instructional manuals

CPK-50165A	*1K AIM-65*	$424.95
CPK-50465A	*4K AIM-65*	$474.95
SFK-74600008E	*8K BASIC ROM*	$64.95
SFK-64600004E	*4K Assembler ROM*	$43.95
PSX-030A	*Power Supply*	$64.95
ENX-000002	*Enclosure*	$54.95
SFK-74600020E	*PL/65 ROM*	$84.95
SFK-74600010E	*Forth ROM*	$64.95
SFK-74600030E	*Instant Pascal*	$99.95

Special Packages

4K AIM-65, 8K BASIC, power supply & enclosure

Special Package Price		$649.95

Video Monitors

HI-RES 12″ GREEN - Zenith

15 MHz bandwidth, 700 lines/inch, P31 green phosphor, switchable 40 or 80 columns, small, light-weight & portable.

VDM-201201	*List price $150.00*	$118.95

12″ GREEN SCREEN - NEC

20 MHz, P31 phosphor video monitor with audio, exceptionally high resolution - A fantastic monitor at a very reasonable price

VDM-651200	*Special Sale Price*	$169.95

12″ COLOR MONITOR - NEC

Hi-res monitor with audio & sculptured case

VDC-651212	*Color Monitor*	$479.95
NEC-1202D	*RGB color monitor*	$1045.00

Leedex / Amdek

Reasonably priced video monitors

VDC-801310	*13″ Color I*	$379.95
VDC-801320	*Color II*	$895.00
IOV-2300A	*DVM board for Apple*	$199.95

Prices may be slightly higher at our retail locations. Please call the store nearest you for local price and availability.

Structured Programming In Basic

by Bob Kowitt

Structured programming techniques are a blessing to the programmer who writes large programs. Many languages are designed specifically to be structured. Basic, originally intended to be a beginner's language, was not designed for structured programming use. However, several versions of Basic do permit the use of structured programming techniques. I use Microsoft Basic80, and have found it lends itself easily to structured programs.

Generally, in Basic, structured programming means the liberal use of GOSUBS to all routines. To take full advantage of this, I use a vector table of GOSUBs with all referenced subroutines located at the beginning of the program.

The CP/M BIOS jump table is an example of structured programming. It is structured in such a way that the CP/M BDOS always goes to the same address for its functions. From that address it jumps to the location within the BIOS area where the function code actually resides. Therefore, by knowing the base of the jump table you can immediately locate the particular function needed.

The same thing can be done with a Basic program and, as a by-product, gain a tremendous programming time saving.

This means keeping a standard list of the locations of utility subroutines so that you always know where they are. Then when you want, for example, to center a message line, you GOSUB to that line. It is much simpler when these routines are always located in the same place in your programs.

I have constructed a file I call HEADER.BAS which I save with the MBasic toggle,A. This causes the file to be saved in ASCII form and it is then mergable into any other Basic program. I use lines 1-98 for elementary housekeeping such as clear screen codes, form-feed codes, CLEAR, RESET, the title, etc.

Line 99 must contain the line:

```
99    GOTO 1000        '       to start of program
```

to jump past the jump table and all of the utility subroutines.

Lines 200-950 (more about lines 950-999 in another article) are reserved for the utility subroutines used by all programs such as:

(1) Choosing console or printer display
(2) Center the message on the display
(3) Cursor addressing

and so on. With judicious use of the renumber function, I am able to get them all in. This area is not used for subroutines that specifically relate to the program itself. These will be in the body of the program.

Now for the best part, Lines 100-199. These lines are a series of GOTOs, i.e., a Basic copy of the jump vector table as used in CP/M. Now if you have a jump table at the beginning of your Basic programs which contains the following:

```
125          GOTO 425      '          normal centering
130          GOTO 390      '          emphasized centering
 ...
 ...
165          GOTO 375      '          modular tabbing
     etc.
```

you can always remember that GOSUB 120 will *always* mean (if you have been careful when renumbering) that you will branch to 120, then jump from there to the actual subroutine and return. The actual location need never concern you because when you modify a routine's location you must only modify the jump table GOTO address.

In my HEADER.BAS file, I typically have as many as 40 to 50 jump vectors. After merging HEADER.BAS into my new Basic program, sub-routines that are not needed are eliminated to save memory. It is easier to erase the subroutines than to write them in when needed. My procedure consists of putting in my title remarks, then MERGEing "HEADER." I do this by typing:

```
MERGE "HEADER"
```

In minutes, I have installed my first 1000 lines and housekeeping is attended to. If your Basic does not allow merging, you will have to LOAD "HEADER," insert any title you want before it, or at worst type in HEADER each time. This should take no more time, than your previous method and the benefits of uniformity from program to program will remain.

A portion of my HEADER.BAS file is reproduced below. Yours, of course, may have other subroutine calls. I have included for illustration my "modular tab" routine. Space limitations do not permit including all the subroutines.

For example, the following short program is non-structured coding to read in data from a sequential file and print in on the printer:

```
10      '      HEADER DEMO PROGRAM
20      '      9/25/81
30      '
40 LAST.ITEM=100
50 GOTO 1000
60      '
```

Bob Kowitt, 1727 N. Jerusalem Rd., E. Meadow, N.Y. 11554

```
998     '       ++++++++++++++++++++++++++++
999     '       START OF PROGRAM
1000 OPEN "I",#1,"FILENAME"
1010 FOR N = 1 TO LAST.ITEM
1020 IF EOF(1) THEN 1060
1030 INPUT #1,A$(N)
1040 LPRINT A$(N),
1050 NEXT N
1060 the rest of the program after this point
```

Between each data item, you will get the number of spaces needed by the printer to reach the next comma-tab position, depending on the actual value of the A$ variable. It would be desirable to have each data item printed at a tab position related to its position in the data list. We may arbitrarily decide that we will print a data item every 25 spaces. That is, #1 will print at the left margin; #2 will print 25 spaces in and #2 will print at 50 spaces in. However, look what happens to #3. It will start at position 75, and, unless it is less than five characters long, will wrap around to next line. Worse yet, is what happens to #4, at theoretical position 100 on an 80 character printer.

The MODTAB subroutine provides columnar presentation of data. For example, assume that we know we will have a maximum of 15 characters in A$. Insert the following into your program:

```
1035 S=20           'allow 20 character spacing
1036 M=3            'allow 3 per line

1040 ITEM$=A$(K) : GOSUB 165
                    'call modtab printer routine
```

The MODulus operator returns the integer remainder of an integer division. Therefore, 22 Mod 4 yields 2; 14 MOD 5 yields 4. For example: when N = 1 the tab position is ((N-1) MOD 3) *20 or equivalent to TAB(0); when N=2 the tab position will equal ((N-1) MOD 3) *20 or 2*20, the equivalent of TAB(40); when N=11, the tab position will be ((11-1) MOD 3) * 20 spaces which reduces to (10 MOD 3) * 20 or TAB(20).

If you should decide to vary the number per line or if you find that the variable does not need as much space as you had assigned it, change the values of S and M instead of going through an elaborate line editing procedure. ■

```
10      '       HEADER          5/1/81
20 C$=CHR$(126)+CHR$(28)+CHR$(26)+CHR$(4)     'combination clear screen
30 V$=CHR$(20)  'reverse video toggle
35 ON ERROR GOTO 715
40      '       ******************************
99      GOTO 1000  '*** TO START OF PROGRAM ***
100     '       ******************************
105     '       ******   JUMP VECTORS   *******
110     '       ******************************
115     GOTO 225        '       display client name data
120     GOTO 260        '       display byline and title
125     GOTO 425        '       normal centering
130     GOTO 390        '       emphasized centering
135     GOTO 590        '       find output location byte
140     GOTO 540        '       select console or printer output
145     GOTO 610        '       enable console
150     GOTO 615        '       enable selected display mode
155     GOTO 315        '       wait control
160     GOTO 335        '       instructions for wait control
165     GOTO 375        '       modular tabbing
170     GOTO 435        '       select terminal for cursor addressing
175     GOTO 465        '       enable cursor addressing
180     GOTO 495        '       strip nulls from random file strings
185     GOTO 520        '       reverse first/last names
190     GOTO 650        '       flashing cursor routine
195     GOTO 695        '       delay of 100 loops
200     GOTO 705        '       main menu load message
205     GOTO 710        '       disc access message
 ...
 ...
 ...
360     '       ++++++++++++++++++++++++++++
365     '       modular tabbing ON PRINTER from array level
370     '       S=spacing       M=how many on line
375     LPRINT TAB(S*((N-1) MOD M)) N;" ";ITEM$; : RETURN
 ...
 ...
 ...
1000    '       *************************
1001    '       *       START           *
1002    '       *************************
1003    '
   ***********************************************************
```

Minimizing The Inconvenience Of Compiled Languages Under CP/M

by Leonard C. Schwab

If you have ever developed a lengthy program in Fortran, CBasic or assembly language, you will probably agree that the steppy processes involved are boring, frustrating and inconvenient.

Each time that you want to make even the most minimal change in your program, you must first load and execute a text-editor program to modify your program source-code file. Then you must load and execute the compiler program to generate some kind of code file which, in turn, may be executed by the computer (object-code) or by a run-time interpreter program (pseudo-code or P-code). For the purposes of this article, we will group several types of programming languages under the rubric "compiler" which technically may not belong in that category. The grouping is justified by the fact that all of the systems included in the group require that the programmer go through at least three steps in each programming cycle.

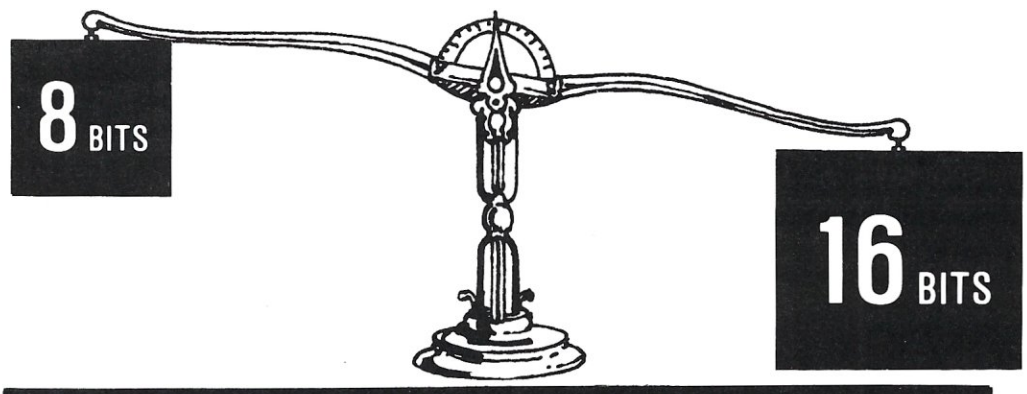
While the compiler is executing, you sit hypnotized, staring into an unchanging CRT tube, waiting for the compiler to pass judgement on your efforts. Many programmers will testify that these periods of inactivity, waiting for the computer to complete its tasks, are the most boring, energy-draining part of the programming process.

Leonard C. Schwab, 2728 Garber Street, Berkeley, CA 94705.

It is well acknowledged that programming in a compiled high-level language is less convenient than in a interpreted language. When using interpreted languages, including most microcomputer Basics, one may stop the execution of a program, examine values in variables, change a line or two of code and resume execution at the start of the program or at any specific line number. You can even test the effect of a proposed line of code by executing it in the command mode, without a line number.

On the other hand, there are several reasons why compiled languages, such as Fortran, or semi-compiled languages, like CBasic-2, may be preferable for particular applications, in spite of their inconveniences.

The compiled language is likely to encourage the writing of code which is more easily read, and which can have better structure than interpreted code. This results from a number of factors. The source-code for a compiler will probably not have a number for each line, giving the programmer some additional freedom in organizing and reorganizing the code. The use of a powerful full-screen text-editor instead of the relatively primitive text-editing facility usually found in an interpreted language may allow the programmer to enter and edit code more efficiently. Compiled languages allow richer commentary than interpreted languages because the compiler strips the comments and remarks from the program which are superfluous to the computer.



MICROCOMPUTERS

THE CHOICE JUST GOT EASIER!

PRICE $4390 Gets you a Tec-86 Complete 16 Bit 8086 System, including 64K Bytes of Memory, Dual 8 Inch Double Density Floppy Disks, and more. . . (See complete Tec-86 specs below.)

PERFORMANCE Faster Processing, Increased Address Space, up to One Megabyte of available memory, and the Increased throughput of the the TEC-86 16 Bit System means that you can now do work on a microcomputer never before practical with the slower 8 Bit systems.

SOFTWARE CP/M-86™ Brings the World of CP/M software up to the 16 bit systems. BASIC-86™, MS-Fortran™, MS-PASCAL™ are just a few of the Languages currently available.

EXPANSION Memory Expansion up to ONE MEGABYTE. Disk Storage can be increased to over 30 Megabytes with our Winchester Disk Subsystem. Multi User Systems available soon!

QUALITY Full ONE YEAR WARRANTY is standard on TECMAR components. This Warranty is possible due to our High Engineering Standards, and Thorough Product Testing.

DELIVERY We have been delivering 16 Bit Boards and Systems for nearly two years. Most orders are shipped from stock or within 30 days.

S100 A/D $765
Analog to Digital Converter with Timer/Counter

12 Bit Accuracy, 16 Channels, 30 KHz Conversion, Five Programmable Timer/Counters for:
- Time of Day
- Triggering Conversions
- Counting Conversions
- Counting External Signals
- Interrupting the CPU

Options: Programmable Gain; 14, 16 Bit Accuracy; 40, 100, 125 KHz Conversion; Expansion to 256 Channels

$4390 TEC-86 $4390
16 BIT 8086 MICROCOMPUTER SYSTEM

FEATURES: 5 MHz 8086 CPU * 64K Bytes RAM * S100 IEEE-696 * Dual 8 Inch Double Density Floppy Disk Drives * Two RS 232 Serial Ports, 50 to 19200 Baud * Three Parallel Ports * ROM Boot for CP/M-86 * Heavy Duty Power Supply * Attractive Industrial Quality Desk Top Cabinet * Completely Assembled and Tested * **OPTIONS:** Up to One Megabyte of Memory, Double Sided Double Density Floppy Disks * Up to 31 Megabytes of Winchester Fixed Disk Storage. * CP/M-86 * BASIC-86 * MS-FORTRAN * MS-PASCAL * CBASIC/86™ * PASCAL/M86™

$10990 TEC-86W $10990

Same Great Features as our TEC-86 except that the TEC-86W has 256K of RAM, and our 31 Megabyte Winchester Disk Subsystem.

NEW — 64K/256K S100 MEMORY $595/$1695

8 and 16 Bit Transfer for 8 and 16 Bit Systems
MP/M and MP/M-86 Compatible, Up to 7 users on one 256K board.
Parity Detection with Interrupt and LED Indication
Hardware Write Protect
Full DMA Capability
24/20/16 bit Address Decoding
Enable/Disable 8K/32K Segments
Bank Select in 8K/32K Segments
Meets IEEE-696 Specifications

See our catalog for many other fine S-100 and Apple Boards including: — Analog to Digital Converters *(16-256 channels; 12, 14, or 16 bit accuracy; 30, 40, 100, or 125 KHz; programmable gain; timer/counters)* **— Digital to Analog Converters** *(12 bit accuracy, 3 microsecond conversion)* **— 8086 CPU Board — I/O Boards — 16K Static RAM — Real-time Video Digitizer and Display.**

TECMAR INC.

DEALER INQUIRIES INVITED

TECMAR, INC.
23600 Mercantile Rd. • Cleveland, OH 44122
(216) 464-7410

Add to the above the possibilities that loading times and execution times may be shorter, and that compiled, semi-compiled or pseudo-compiled code may offer some protection against software theft or unauthorized tinkering with the program, one can see why compiled languages continue to be preferred by many programmers for serious commercial applications.

Edit-Compile-Test Cycle

A programmer working in a compiled high-level language must work through each of the following steps each time a change is made in the program:

1. Make the changes or additions to the source code file, using a text-editor.
2. Compile the source code into object code or pseudo-code, using the compiler.

(At this point, subsequent processing steps will depend on the results reported by the compiler. If syntax errors have been reported, then the process will have to be re-started at step 1; otherwise, the processsor will proceed to step 3.)

3. Test the program for logical correctness. Note any "bugs" or changes needed.
4. Repeat (iterate) steps 1 through 3 as necessary until program is finished.

Use Short Filenames

The process outlined above will require the programmer to type in a certain amount of information at the keyboard. The first step to take to minimize the effort involved in the overall process is simply to reduce the number of keystrokes that are needed to perform the various steps. The files and programs that are used most often should be given short names, single-characters, if possible. For example:

Type of program	New, short name
Text-editor	E.COM
CBasic compiler	G.COM (for 'G'enerate)
CBasic run-time package	R.COM

The program or module in process at any given time may also be given a short name, eg. "X.BAS". When complete, the program can be renamed with a longer, more descriptive filename.

Using these short names, the process of compiling and testing a CBasic program may be reduced to the following keystrokes:

```
E X.BAS    (edit source file "X.BAS")
G X $BE    (compile "X", switches 'B' and 'E' on)
R X        (run "X.INT", for de-bugging)
```

Create And Use SUBMIT To Control The Process

Many (most/all) CP/M systems include a utility program named "SUBMIT.COM". Using SUBMIT, a series of programs (called a "procedure") can be started with a single command from the console.

In addition to controlling a sequence of programs, SUBMIT can also pass parameters entered by the operator at the time the procedure is initiated, to the programs named in the procedure. The process for building and using a procedure file is as follows:

1. Create a procedure file:

The procedure file must carry a filename extension of ".SUB". It will be an ordinary ASCII text file and will consist of a series of lines, each of which contains the command that the operator would enter from the console when the CP/M prompt (>) appeared on the screen. (There will be some differences if parameters are to be passed to the procedure, per paragraph two below.)

This file is called the "submit file" or "procedure file" or "proc file." Following the short name convention described previously, the procedure file for developing a CBasic program might be named "C.SUB". A procedure for developing a Fortran program could be called "F.SUB", etc.

To start the execution of a procedure file when no parameters are to be passed to the procedure, the operator simply types "SUBMIT filename," where "filename" is the basic filename of the procedure file, e.g., "SUBMIT C."

2. Arrange for name of program to be substituted into the procedure commands:

Using the parameter passsing feature of SUBMIT, a generalized procedure file may be created to drive any edit-compile-run sequence with the parameter to be passed being the name of the program being developed.

When parameters are to be passed to the procedure, the portions of command lines in the procedure file which will receive the parameter values must be represented symbolically. The symbol consists of a dollar sign and a digit ("$1"). This symbol is placed into the procedure command line exactly where the parameter substitution is desired. Since the dollar sign will be taken by the SUBMIT program as an indicator of a substitution point, any literal dollar signs which may be needed in the final console command, such as are needed when CBasic compilation switches are invoked, must be represented by *two* dollar signs ("$$").

For example, the procedure file line resulting in a console command that will allow the CBasic source file (whose name was supplied as a parameter at the start of the procedure) to be edited is:

```
E $1.BAS
```

As an additional example, the procedure file line which will cause the parameterized source file-name to be compiled by the CBasic compiler (G.COM), with the "B" switch (suppress console print-out during compilation) and the "E" switch (include trace ability in the generated code) turned on is

```
G $1 $$BE
```

To execute a procedure file containing the above two lines, the operator must type:

SUBMIT C <parameter-value-1>

where, in place of <parameter-value-1> , the operator types the name of the program in development.

For example, if the operator typed "SUBMIT C X", the SUBMIT program would create the following two console commands from the procedure lines shown above:

Procedure file line	Resulting console Command
E $1.BAS	E X.BAS
G $1 $$BE	G X $BE

A procedure may include one or more parameters, or no parameters to be substituted for. The first parameter is "$1", the second, "$2", etc. On invocation of the procedure, the several parameters are supplied in the command line separated by spaces, as in ("SUBMIT REPORT JAN 81"). ("JAN" and "81" will be substituted for "$1" and "$2" respectively.)

Listing 1 contains a sample SUBMIT procedure file for driving the iterative process involved in developing a program under CBasic.

If the procedure in Listing 1 is invoked with the command:

SUBMIT C TEST

the following happens:

1. SUBMIT creates the command: E TEST.BAS
2. Editor is loaded, along with the program source file.
3. Exit the editor when finished.
4. SUBMIT creates the command: G TEST $BE
5. Compiler is loaded and compiled TEST.BAS into TEST.INT.
6. SUBMIT creates the command: R TEST
7. CRUN is loaded. It loads and executes TEST.
8. Upon exiting from TEST, SUBMIT repeats the cycle starting at step 1.

As each command is read from the procedure file, it is printed on the console, so you may observe what the system is doing at all times.

Terminating Or Modifying An Executing Procedure

With my version of SUBMIT (Imsai, 1977) the procedure will continue, line by line, until the end of the procedure file is reached or until I hit any key on the keyboard. The SUBMIT program will terminate if, while processing any new line from the procedure files, it finds that a console key has been depressed.

An additional control that is available with my hardware (an Imsai VDP-80) allows me to skip a step or steps in the proc sequence by pushing the interrupt button when the command appears on the screen. Thus, if a compile results in compile-time errors, I can skip the RUN portion of the cycle, going on to the edit portion to correct the errors. This action may be duplicated on other hardware configurations.

In addition, the control-S feature of CP/M may be used to freeze the screen at any time, in order to digest information displayed or to make notes about compile-time errors. Otherwise, the procedure will continue and the information on the screen may be lost when the next program in the sequence is loaded.

For example, if a compilation results in compile-time errors, the programmer may enter control-S to stop the procedure while the error messages are noted. Then the depression of any key on the keyboard will result in the procedure being resumed.

In Listings 2 and 3, you will find sample procedure files for controlling the edit-compile-test run cycle for Microsoft's Fortran compiler and M80 assembler.

Modify CBasic To Signal the End of Compilation

Compilers require time to complete their work. The exact amount of time that a compiler will take depends on the efficiency of the compiler and the size of the source code being compiled. CBasic is not a particularly fast compiler, and a large program (one that results in a p-code file of 16 kilobytes or so) may take two or three minutes to complete.

Two or three minutes may not seem like a significant period of time, but when the programmer is involved in an iterative cycle of edit-compile-run, this short delay in active work can seem endless. It would be nice to be able to either turn one's attention to other work or to relax during the compilation process, but in order not to lose precious time at the completion of a compilation, the programmer must be made aware of the fact that the compiler program has terminated.

Most especially, when using a procedure file to drive the edit-compile-run cycle, one could easily turn away from the computer while the compiler is executing. However, if you happen not to be looking when the compiler finishes its part of the cycle, you may easily miss the CBasic exit report, with its important information regarding syntax errors in the source file and partition sizes.

Obviously, some kind of audible signal is needed to alert the programmer when compiler execution is done.

My first approach to this situation was to write a simple assembly language program, which I called "BEEP.COM." BEEP did nothing but sound my console alarm. I inserted a command line into the procedure to call this little program after the compiler exited and before CRUN was loaded.

This solution was not satisfactory because I felt that the time required to load and execute "BEEP" was excessive. Also, the solution didn't apply to the times when I was compiling without the use of a procedure file. What was needed was for CBasic to sound the alarm itself as it exited to CP/M. I decided to attempt to patch a signaling routine into CBasic itself.

I loaded CBASIC2.COM, version 2.06, under DDT, noting that the "next" location, signifying the length of the program, was at 5200. (All memory address references in this section are in hexadecimal.) I was looking for a "JMP 0000" statement, the accepted method for exiting from an application program and returning to CP/M control.

Using the disassembler feature of DDT, I quickly found the statement at address 010B, near the start of the program. This was evidence of the excellent programming techniques of the people at Compiler Systems, Inc. Using the breakpoint facility of DDT, I was able to confirm that 010B was indeed the only exit from CBasic. Thank you, CSI!

Listing 1: *The following is a sample procedure file for driving the edit-compile-test run cycle while developing programs under CBasic. The comments, at the right, must* not *be included in the procedure file.*

```
E $1.BAS          (first
G $1 $$BE         (  iteration
R $1              (    of cycle.
E $1.BAS          (second
G $1 $$BE         (  iteration
R $1              (    of cycle.
:                 (the basic three line
:                 (  cycle may be repeated
:                 (    as many times as desired.
:
```

Listing 2: *The following is a sample procedure file for driving the edit-compile-test run cycle while developing programs under Microsoft Fortran (F80). The comments, at the right, must* not *be included in the procedure file.*

```
E $1.FOR           (edit the FORTRAN source file.
F80 =$1            (compile the program.
L80 $1/N,$1/E      (link the program, save after exit.
DDT $1.COM         (load for testing under DDT.
:                  (
:                  (repeat the basic four line
:                  (  cycle as often as desired.
:
```

Listing 3: *The following is a sample procedure file for driving the edit-compile-test run cycle while developing programs under Microsoft macro assembler (M80). The comments, at the right, must* not *be included in the procedure file.*

```
E $1.FOR           (edit the assembly language source.
M80 =$1            (assemble the program.
L80 $1/N,$1/E      (link the program, save after exit.
DDT $1.COM         (load for testing under DDT.
:                  (
:                  (repeat the basic four line
:                  (  cycle as often as desired.
```

Listing 4: *Routine patched into CBASIC2.COM, version 2.06. This routine beeps the Imsai IKB-2 keyboard speaker four times before exiting to CP/M. The keyboard is configured so that its status port is at 014h.*

Hex Address	Mnemonics	Comments
010B	XRA A	null byte to send to IKB-2
010C	MVI B, 0004	counter for 4 beeps
010E	OUT 14	beep !!!
0110	LXI H, 0000	delay timer initialized
0113	LXI D, 0002	timer increment value
0116	DAD D	update counter
0117	JNZ 0116	... till overflow occurs
011A	DCR B	count down number of beeps
011B	JNZ 010E	beep again, if not finished
0110	JMP 0000	finished — exit to CP/M

Listing 5: *Suggested routines to be patched into CBASIC2.COM, version 2.06, to sound the alarm signal on a "standard" CRT terminal before exiting to CP/M.*

Hex Address	Mnemonics	Comments
(a) sound one beep:		
010B	MVI E, 07	ASCII "BEL" character
010D	MVI C, 02	CP/M CONOUT function call
010F	CALL 0005	call CP/M to send BEL to terminal
0112	JMP 0000	exit to CP/M
(b) sound 4 beeps:		
010B	LXI B, 0402	B = beep count, C = CONOUT call
010E	LXI H, 0001	Timer count increment amount (note)
0111	MVI E, 07	ASCII BEL character
0113	CALL 0005	beep !!!
0116	XCHG	put increment value into DE
0117	LXI H, 0000	initialize timer
011A	DAD D	update timer count
011B	JNC 011A	... until overflow
011E	XCHG	stash the increment amount in HL
011F	DCR B	count down number of beeps
0120	JNZ 0111	beep again if not through
0123	JMP 0000	finished - exit to CP/M

Once I found the exit point, it was necessary to locate a spot in the program area where I might patch in a routine of 30 bytes or so. Fortunately (and incredibly) the exit statement was found to be followed by two null bytes and the Compiler Systems copyright statement. Unless the program used this area for data storage during execution, I could use it for my alarm routine. The only way to determine if the area was safe would be to actually make the patch and then test the operation of the compiler.

I had already developed a machine language routine which would sound my keyboard alarm, in connection with BEEP. The display facility of my machine is an Imsai VIO-D video board. It has no alarm or bell, but the IKB-2 keyboard does have a speaker which sounds off when a null byte is sent to the keyboard status port (14 hex). The routine I wrote for the VDP-80 sounds four quick beeps as CBasic exits to CP/M. The routine is shown in Listing 4.

A more common method for sounding a terminal alarm is to send an ASCII "BEL" control character (07h) to the terminal. This will work for most "outboard" or serial CRT terminals. Listing 5 includes suggested patches for these units.

In the beep routine, note the use of the HL register as a 16-bit delay timer. This is necessary because of the duration of the beep sounded by the keyboard. The timer is initialized at zero and then incremented until it overflows (exceeds FFFF hex). The timer increment is stored as a 16-bit integer in the DE register. Increments are accomplished using the "DAD D" statement rather than by using "INX D" statements. INX does not set the CARRY flag on overflow, so there would be no easy way of sensing the end of the delay period. DAD does set the CARRY flag on overflow.

I entered the patch routine into memory using the assembly facility of DDT. When I had finished entering the patch, I tested its effect with the trace and go-with-breakpoint features of DDT. At this time, I adjusted the timer increment value from one to two to three before finally settling on two.

Satisfied that the routine appeared to be working well, I exited from DDT by typing "GO". The modified program was still in memory, starting at 0100h and ending one byte below 5200. (Recall that 5200 was the "next" address reported by DDT when CBASIC.COM was first loaded.)

In order to be able to command CP/M to save the modified CBASIC.COM, I had to compute the number of 256 byte "pages" contained in the program. CP/M saves .COM files in 256 byte blocks. The number of blocks is computed by multiplying the decimal value of the fourth significant digit in the "next" address (here "5") by 16, and adding the decimal value of the third significant digit (here "2"). If the last two digits are "00" as in this case, you may subtract one from the result. Following this process, I determined that 81 "pages" (5 times 16 plus 2 minus 1) must be saved.

I then typed "SAVE 81 GX.COM" to save the modified CBasic compiler to disk under a new filename. I certainly didn't want to wipe out the unmodified program file until I was absolutely certain that the modifications worked.

After a certain period of using the modified compiler under varying conditions, I was satisfied that I had done no harm to the main functioning of the compiler. I renamed the file "G.COM" and now use it regularly with great satisfaction.

Summary

The techniques discussed in this article won't make compilers as convenient to use as interpreters but they help to reduce the drudge work and wasted time involved in the iterative edit-compile-run cycle. I know that it is much more pleasant for me to let the procedure file load the programs and I really like the freedom I get from the simple modification of CBasic. Now I can turn my attention away from the computer and, when the alarm sounds, I merely glance over to see if any errors were found. If so, I push the interrupt button to skip over the test run; otherwise, I am ready to conduct a test of whatever feature I happen to be working on at the time.

LEONARD SCHWAB was born in Birmingham and attended the University of Alabama. He earned MBA and Law degrees at the University of California in Berkeley, where he now lives. As a CPA and consultant, he has been involved in accounting, systems and management at various levels and with a number of firms. He is an Associate Professor at Cal State University at Hayward where he teaches computing and accounting.

A Replacement For CP/M SUBMIT

by David E. Cortesi

The power of the SUBMIT and XSUB commands in CP/M 2.2 are limited by problems in SUBMIT. A replacement for the SUBMIT command has been written in Pascal (Pascal/Z). It corrects two major shortcomings in SUBMIT: the inability to submit an empty line, and the inability to submit lower-case characters. The program also demonstrates the internal operation of SUBMIT.

The SUBMIT command in CP/M is potentially a very powerful tool, especially when combined with the XSUB command provided in version 2. Unfortunately, that potential is stunted by some problems in the SUBMIT command. This article presents a replacement for SUBMIT that corrects its major shortcomings. The program is written in Pascal as implemented by the Pascal/Z compiler from Ithaca Intersystems.

The Use of SUBMIT

The SUBMIT command accepts as input a file of commands that you'd like to have performed, and causes those commands to be executed in the given order by the Console Command Processor (CCP). The submitted commands can be parameterized; wherever the sequence $n (n digit from 0 to 9) appears in the file, SUBMIT replaces it with the nth sequence of letters from the command line that invoked SUBMIT. Control characters can be submitted by entering ^X (X capital letter from A to Z) in the file; the corresponding control character will be submitted.

The operation of SUBMIT is described fairly well in the CP/M documentation, although in some details of parameter substitution the command and the documentation do not agree. For example, the sequence $0 will be replaced with the name of the submit file; the documentation suggests that $0 would be invalid.

SUBMIT gives you a very useful ability to submit any stereotyped sequence of commands for automatic execution while you carry on with other things.

The Use of XSUB

The XSUB command was added to CP/M with version 2; it adds greatly to the uses of SUBMIT. When called within a submitted file, XSUB alters the CP/M BDOS so that a program's request for a line of console input will be answered with the next line from the active submit file, rather than from the terminal. With SUBMIT alone you can only automate command lines. With XSUB you can supply program input lines following the commands line that calls the program.

XSUB is limited in that it can only respond to requests for complete input lines (BDOS service request 10, Read Console Buffer). Some programs request their input from the BDOS one letter at a time (or worse, get it by direct BIOS calls). Such input requests are not supplied

David E. Cortesi, 2340 Tasso Street, Palo Alto, CA 94301.

from the active submit file; they cause a read from the terminal. It can be quite a challenge to work out which method a program uses.

Fortunately, two of CP/M's most useful utilities take their input one line at a time, and so can be automated with XSUB. PIP, when called with no operands, reads its transfer instructions one line at a time. ED also reads its commands one line at a time, so a complete ED edit session can be automated with SUBMIT and XSUB. Well, almost; ED appears to read bulk insertion text one character at a time, so you can only automate the insertion of single lines. All other ED commands can be submitted for automatic execution. This opens the way to programming the script of a very complex file modification, including parameterized changes, and submitting it to run unattended. You could set up a submit file to customize a form letter, supplying the variable parts of the form letter as operands of the SUBMIT command.

The Shortcomings of SUBMIT

SUBMIT has two design defects that limit the potential uses of SUBMIT and XSUB. First, SUBMIT converts all its input to upper case in the submitted file. This was acceptable when you could only submit command lines; the CCP treats command lines as upper case anyway. The addition of XSUB changed this; program input lines must often be given in lower case. ED commands in particular are sensitive to the case of the command letter, and without lower case input you cannot automate editing of a document.

Second, SUBMIT is not capable of submitting a null line. When its input file contains a zero-length line, SUBMIT produces invalid error messages, looping, or crashing in other entertaining ways. Digital Research is aware of the problem, but claims that the fix is too extensive to be released as a patch.

Since SUBMIT can't handle a null line, SUBMIT and XSUB can't be used to automate any program interaction that relies on a null line to signal "the end." For instance, you can't submit a file like this:

```
XSUB
PIP
B:=A:FILE1
B:=A:FILE2
B:=A:FILE3
((null line))
STAT B:
```

PIP requires the null line to tell it to end. The same command sequence could be achieved by submitting three separate PIP commands. However, the sequence shown executes much faster as a single command, since PIP does not have to be loaded for each transfer (some people create a .COM file of zero length and use it to re-invoke PIP in this case; a single command is still faster).

A Replacement For SUBMIT

The Pascal program shown as Listing 1 performs the same functions as SUBMIT, and eliminates both of SUBMIT's shortcomings. It retains lower case in its output, and it will submit null lines for execution. It will accept a control character signal in upper or lower case, instead of requiring upper case. Except for these improvements, and except that its error messages are a bit more specific, the program behaves as much like the original SUBMIT as possible.

Using the Program

If you have the Pascal/Z compiler, you can put the program to work by typing it in and compiling and linking it. The resulting .COM file occupies 10K bytes; it loads and executes just a bit more slowly than the original SUBMIT. The constant "mxsub" determines the number of lines that may be submitted and therefore the amount of storage the program requires. The upper limit of "mxsub" in a system with single-density disks is 128.

The program should work with another Pascal compiler essentially as given. The few points at which the code relies on (or is inhibited by) peculiarities of Pascal/Z are noted in the comments; read the code carefully and substitute the corresponding features of your system. The only place where you might meet a problem is in reading the command "tail," the characters that follow the program's name in the command line that invokes it. If your Pascal won't supply them, you might have to create an assembly language subroutine to return them from location 80H where the CCP leaves them.

If you don't have access to any Pascal, the program can still act as a design template for your own version of SUBMIT in assembly language. It would be interesting to see how much longer an assembly language version with identical logic would be, and how readable it would be with comparable commenting.

Technical Stuff: SUBMIT and the CCP

The SUBMIT command is just a utility program; most of the logic of submit processing is in the Console Command Processor (CCP). SUBMIT copies an ASCII file of command lines to an output file named $$$.SUB, changing the format of each line as it goes. The output file has fixed-length records, each of which is an image of how the CCP's console buffer would look if the corresponding command had been entered from the terminal.

The CCP is loaded during each warm start. There is no safe place in storage where the CCP would preserve information from one warm start to the next. Each time it is loaded, the CCP has to determine afresh whether there is an active submit file and, if there is, which line of it should be done next. The first test is done by searching for a file A:$$$.SUB. If it exists, submit processing is active, else not.

The second task—keeping track of the next line in $$$.SUB—is done in a way that is at once ingenious and restrictive. The directory entry for any file extent contains a count of the number of records in that extent. The CCP used that count to read the *last* record of $$$.SUB. After reading it, the CCP updates the directory entry to show that the file has one less record in it, effectively dropping the last record from the file.

You can demonstrate this CCP action for yourself. Prepare a submit file that contains a number of identical commands like this:

```
STAT $$$$$$.SUB
```

(Six dollar signs are needed in order to submit three of them.) Submit the file for execution. The reported number of records in $$$.SUB decreases by one with each command executed. The amount of space the file occupies does not change, indicating that the CCP doesn't actually release the allocated space, but only changes the record count.

The CCP's scheme is restrictive in two ways. It requires that the submitted commands appear in $$$.SUB in reverse order; the record that is physically last will be executed first. A more serious restriction is that the file may not exceed one extent; that is, it may not be so large as to require more than one directory entry. An extent is usually 16KB, or 128, 128-byte records. On a double-density disk CP/M can use 32KB extents, in which case $$$.SUB could be allowed to contain 256 records (presuming there was enough room for the buffer). I don't know whether the SUBMIT command will actually allow this to happen. The program given here allows as many records as specified by the constant "mxsub." ■

Listing 1.

```
program sub;{ A version of the CP/M Submit command in Pascal/Z }
{$c+,e+,f-,i+,m-,p-,r-,s-,t-,u-  PASCAL/Z compiler options    }
{****************************************************************
 *                                                              *
 *                    C O N S T A N T S                         *
 *                                                              *
 ****************************************************************}
const
   mxsub     = 64;          { number of records in buffer     }
   neos      =  0;          { ord(NUL), end of string         }
   ntab      =  9;          { ord(TAB)                        }
   mnctok    ='0';          { smallest replace signal: $0     }
   mxctok    ='9';          { largest ditto: $9               }
   mxtok     =  9;          { highest token index             }
   mxline    = 127;         { number of characters in line    }
   mxsubline= 126;          { max char.s in a submit record   }
   blank     = ' ';         { blank comparand                 }
{****************************************************************
```

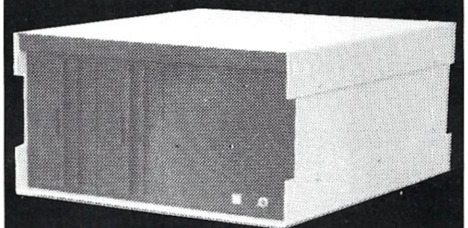
Main/Frames

Main/Frames

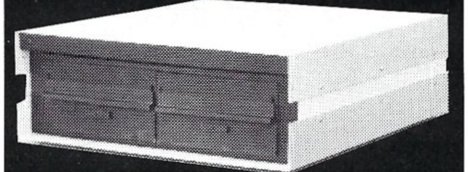
from

$200

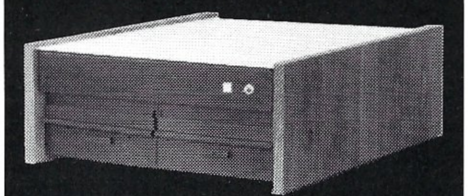
- 30 Models of Enclosures
- Assembled and tested
- Quasi-Coax Motherboards
- Power Supply
- Card cage and guides
- Fan, line, cord, fuse, power & reset switches



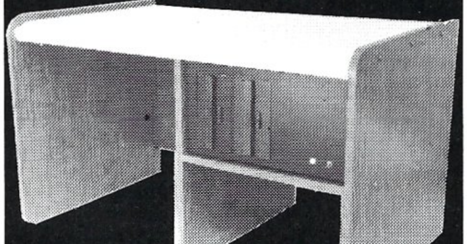
8" Floppy Main/Frame



8" Disc Enclosure



Phase/80 8" Floppy Mainframe



Phase/80 Desk + Mainframe

Write or call for our brochure which includes our application note: "Building Computers — A Recipe"

INTEGRAND

8620 Roosevelt Ave. • Visalia, CA 93291
209/733-9288

We accept BankAmericard/Visa and MasterCharge

FREE with software purchase — One CPM Handbook

DISCOUNT Ad#25 SOFTWARE

ULTIMATE SOFTWARE PLAN

We'll match any advertised price on any item that we carry. And if you find a lower price on what you bought within 30 days of buying it, just show us the ad and we'll refund the difference. It's that simple.

Combine our price protection with the availability of full professional support and our automatic update service and you have the Ultimate Software Plan.

It's a convenient, uncomplicated, logical way to get your software.

✓ **(New items or new prices)**
*** Special price of the month.**
CP/M users:
specify disk systems and formats. Most formats available.

CP/M® DISK WITH MANUAL / MANUAL ONLY

ARTIFICIAL INTELLIGENCE®
Medical(PAS-3) $849/$40
Dental (PAS-3) $849/$40

ASYST DESIGN®
Prof Time Accounting $549/$40
General Subroutine $269/$40
Application Utilities $439/$40

COMPLETE BUS. SYSTEMS®
*Creator $199/$25
*Reporter $129/$20
*Both $299/$45

COMPUTER CONTROL®
*Fabs (B-tree) $119/$20
*UltraSort II $119/$25

COMPUTER PATHWAYS®
Pearl (level 1) $ 99/$25
Pearl (level 2) $299/$40
Pearl (level 3) $549/$50

DIGITAL RESEARCH®
CP/M 2.2
NorthStar $149/$25
TRS-80 Model II (P+T) $159/$35
Micropolis $169/$25
PL/I-80 $459/$35
BT-80 $179/$30
Mac $ 85/$15
Sid $ 65/$15
Z-Sid $ 90/$15
Tex $ 90/$15
DeSpool $ 50/$10
CB-80 $459/$35
CBasic-2 $ 98/$20

D.M.A.
Ascom $149/$15
Formula $539/$45

GRAHAM-DORIAN®
General Ledger $729/$40
Acct Receivable $729/$40
Acct Payable $729/$40
Job Costing $729/$40
Payroll II $729/$40
Inventory II $729/$40
Payroll $493/$40
Inventory $493/$40
Cash Register $493/$40
Apartment Mgt $493/$40

MICRO-AP®
S-Basic $269/$25
Selector IV $295/$35
Selector V $495/$50

MICRO DATA BASE SYSTEMS®
HDBS $269/$35
MDBS $795/$40
DRS or QRS or RTL $269/$10
MDBS PKG $1295/$60

MICROPRO®
WordStar $319/$60
Customization Notes $429/$na
Mail-Merge $109/$25
WordStar/Mail-Merge $419/$85
DataStar $249/$60
WordMaster $119/$40
SuperSort I $199/$40
Spell Star $175/$40
CalcStar $259/$na

MICROSOFT®
Basic-80 $298
Basic Compiler $329
Fortran-80 $349
Cobol-80 $629
✓ M-Sort $175
✓ Macro-80 $144
Edit-80 $ 84
MuSimp/MuMath $224
MuLisp-80 $174
Multi Plan Call
Manager Series Call

ORGANIC SOFTWARE®
TextWriter III $111/$25
DateBook II $269/$25
Milestone $269/$30

OSBORNE®
General Ledger $ 59/$20
Acct Rec/Acct Pay $ 59/$20
Payroll w/Cost $ 59/$20
All 3 $129/$60
All 3 + CBASIC-2 $199/$75
Enhanced Osborne $269/$60

PEACHTREE®
General Ledger $399/$40
Acct Receivable $399/$40
Acct Payable $399/$40
Payroll $399/$40
Inventory $399/$40
Surveyor $399/$40
Property Mgt $799/$40
CPA Client Write-up $799/$40
P5 Version Add $129
MagiCalc $269/$25
Other less 10%

SOFTWARE WORKS®
*Adapt (CDOS to CP/M) $ 49/$na
*Ratfor $ 68/$na

SOHO GROUP®
*MatchMaker $ 89/$20
*WorkSheet $159/$20

STRUCTURED SYSTEMS®
Business Packages, Call for Price

SORCIM®
SuperCalc $269/$na
Trans 86 $115
Act $157

TCS®
GL or AR or AP or Pay $ 79/$25
All 4 $269/$99
Compiled each $ 99/$25
Inventory $ 99/$25

SUPERSOFT®
Diagnostic I $ 49/$20
Diagnostic II $ 84/$20
Disk Doctor $ 84/$20
Forth (8080 or Z80) $149/$30
Fortran $219/$30
Fortran w/Ratfor $289/$35
C Compiler $174/$20
Star Edit $189/$30
Other less 10%

UNICORN®
Mince $149/$25
Scribble $149/$25
Both $249/$50

WHITESMITHS®
"C" Compiler $600/$30
Pascal (incl "C") $850/$45

"PASCAL"
Pascal/MT+ Pkg $429/$30
Compiler $315
Sp Prog $175
Pascal/Z $349/$30
Pascal/UCSD 4.0 $670/$50
Pascal/M $355/$20

"DATA BASE"
FMS-80 $649/$45
dBASE II $595/$50
Condor II $899/$50

"WORD PROCESSING"
WordSearch $179/$50
SpellGuard $229/$25
VTS/80 $259/$65
Magic Wand $289/$45
Magic Spell $269/$25
Spell Binder $349/$45
Select $495/$na
The Word $ 65/$na

"OTHER GOODIES"
Micro Plan $419/$na
Plan 80 $269/$30
Target $189/$30
BSTAM $149/$na
BSTMS $149/$na
Tiny "C" $ 89/$50
Tiny "C" Compiler $229/$50
Nevada Cobol $179/$25
MicroStat $224/$25
Vedit $130/$15
MiniModel $449/$50
StatPak $449/$40
Micro B+ $229/$20
Raid $224/$35
String/80 $ 84/$20
String/80 (source) $279/$na
ISIS CP/M Utility $199/$50
Lynx $199/$20
Supervyz $ 95/$na
✓ CP/M Power $ 75/$na
✓ Mathe Magic $ 95/$na

APPLE II®

INFO UNLIMITED®
EasyWriter $199
Datadex $249
EasyMailer $128
Other less 15%

MICROSOFT®
Softcard (Z-80 CP/M) $298
Fortran $179
Cobol $499
Tasc $139

MICROPRO®
Wordstar $269
MailMerge $ 99
Wordstar/MailMerge $349
SuperSort I $159
Spellstar $129

PERSONAL SOFTWARE/ VISICORP®
Visicalc 3.3 $225
Desktop/Plan II $225
Visiterm $ 90
Visidex $225
Visiplot $180
Visitrend/Visiplot $300
Visifile $225

PEACHTREE®
G/L, A/R, A/P, Pay or Inventory (each) $224/$40

"OTHER GOODIES"
*VU #3R (use w/Visicalc) $ 49
*Context Connector (use w/Visicalc) $ 99
Micro Courier $219
Super-Text II $127
Data Factory $134
DB Master $184

IBM PC SOFTWARE

✓ Business/Med/WP Call for Price

8086 SOFTWARE
✓ CBASIC 86 $300/$na
✓ Pascal MT + 86 Pkg $730/$na
✓ Pascal UCSD 86 $700/$na
✓ Macro 86 $259/$na

ORDERS ONLY—CALL TOLL FREE VISA • MASTERCHARGE

1-800-854-2003 ext. 823 • Calif. 1-800-522-1500 ext. 823

Outside Continental U.S.—add $10 plus Air Parcel Post • Add $3.50 postage and handling per each item • California residents add 6% sales tax • Allow 2 weeks on checks, C.O.D. $3.00 extra • Prices subject to change without notice. All items subject to availability • ®—Mfgs. Trademark. Blue Label $3.00 additional per item.

THE DISCOUNT SOFTWARE GROUP
6520 Selma Ave. Suite 309 • Los Angeles, Ca. 90028 • (213) 837-5141
Int'l TELEX 499-0446 DISCSOFT LSA • USA TELEX 194-634 (Attn: 499-0446)
TWX 910-321-3597 (Attn: 499-0446)

```
 *                                                            *
 *                        T Y P E S                           *
 *                                                            *
 *************************************************************}
type
        $STRING0 = STRING 0;      { for PASCAL/Z string functions}
        $STRING255 = STRING 255;
{*************************************************************
* Definitions of submit records and submit buffer.  The buffer *
* is set to 64, 128-byte records (8K).  In CP/M 2.2 the maximum*
* buffer depends on the size of one extent.  8K is the smallest*
* The format of one record is: a byte giving the length of the *
* actual data, from zero to 126 bytes of characters, followed  *
* by a byte of zero which is not counted in the length.        *
*************************************************************}
   subrec    = record
                  sublen   : 0..127;
                  subdata : array[1..127] of char
               end;
   subspan   = 1..mxsub;
   subuffr   = array[subspan] of subrec;
{*************************************************************
* Defining a line and a token.  Some games have to be played   *
* with the types of line indices, since it is useful to start  *
* an index at zero or let it run past 127.  A single token may *
* reach 120+ bytes, so a token is a full 127-byte line.        *
*************************************************************}
   linespan = 1..mxline;           { span of a line              }
   linecnt  = 0..mxline;           { count of bytes in a line    }
   lineover = 1..128;              { index of line + 1           }
   line     = array[linespan] of char; {console line or token }
   token    = line;
   tokens   = array[0..mxtok] of token;

{*************************************************************
 *                                                            *
 *                    V A R I A B L E S                       *
 *                                                            *
 *************************************************************}
var
   okfile    : boolean;            { master return code          }
   buffer    : subuffr;            { sequence of submit records  }
   eos       : char;               { end of string constant      }
   tab       : char;               { tab-comparand constant      }
   recount   : 0..mxsub;           { count of submit records     }
   replist   : tokens;             { list of 10 tokens           }
   input     : line;               { command tail, input records }
   infile    : file of char;       { input submit file           }
   outfile   : file of subrec;     { $$$.SUB                     }
{*************************************************************
 *                                                            *
 *                 P R O C E D U R E S                        *
 *                                                            *
 *************************************************************}
{  PASCAL/Z STRING PROCEDURES   }
FUNCTION LENGTH(X:$STRING255):INTEGER; EXTERNAL;
PROCEDURE SETLENGTH(VAR X:$STRING0; Y:INTEGER); EXTERNAL;
{*************************************************************
* Procedure to clear the buffer and token list, and to set up  *
* some constants.  No need to clear the whole submit record,   *
* because CP/M and XSUB stop reading at the 'eos' at the end   *
* of the data (or maybe after 'sublen' bytes -- who knows?).   *
* The terminating eos is put in when the record is written.    *
```



```
*************************************************************}
procedure clear;                   { initialize buffer and tokens }
   var bx : subspan;               { index over buffer            }
       rx : 0..mxtok;              { index over token list        }
       lx : linespan;              { index over a line            }
       nulline : line;             { a line of eoses to clear toks}
   begin
      eos := chr(neos);            { const eos=chr(0)             }
      tab := chr(ntab);            { const tab=chr(9)             }
      for bx := 1 to mxsub do      { all records start empty      }
         buffer[bx].sublen := 0;
      for lx := 1 to mxline do nulline[lx] := eos;
      for rx := 0 to mxtok do      { all tokens start null...     }
         replist[rx] := nulline    { ...in all bytes              }
   end{clear};
{*************************************************************
* Procedure to set up the command tail in 'input'.  PASCAL/Z   *
* returns the command tail on the first read/readln, provided  *
* no other files are opened first.  On a readln of a character *
* array, PASCAL/Z fills the array to the end with blanks.      *
*************************************************************}
procedure get_tail;                { get command tail to input    }
   begin
      readln(input)
   end;

{*************************************************************
* Function to scan backwards over a line and return the count  *
* of active characters -- the length of the line less the      *
* number of trailing blanks -- which could be zero!            *
*************************************************************}
function scanback(l:line):linecnt;
   var q : linecnt;
   begin
      q := mxline;
      while (q>0) and (l[q]=blank) do
         q := q-1;
      scanback := q
   end;
{*************************************************************
* Function to run an index ahead over white space (blanks and  *
* tabs).  Returns the index of the next non-white character.   *
* If there isn't one, the returned value will be mxline+1 (128)*
*************************************************************}
function skibble(l:line;q:lineover):lineover;
   var x : lineover;
   begin
      x := q;
      while (x<=mxline) and ((l[x]=blank) or (l[x]=tab)) do
         x := x+1;
      skibble := x
   end;
{*************************************************************
* Procedure to load all tokens from the command tail (now in   *
* 'input') to the token array 'replist'.  A series of short    *
* tokens could fill the list even with a short input line.  If *
* that happens, give a message.  The submit will go on.        *
*************************************************************}
procedure load_tok;                { load token list from tail    }
   var lx : lineover;              { index over 'input'           }
       lz : linecnt;               { count of useful characters   }
       rx : 0..10;                 { index over 'replist'         }
       tx : linespan;              { index over a token           }
```

```
   begin {load_tok}
      lz := scanback(input);      { count of useful characters    }
      lx := 1;                    { set for 1st byte              }
      rx := 0;                    { set for 1st token             }
      while (lx<=lz) and (rx<=mxtok) do
         begin                    { have data, list isn't full..  }
            lx := skibble(input,lx); { next (first) data          }
            tx := 1;              { token presently full of eos   }
            while (lx<=lz) and
                 (input[lx]<>blank) and (input[lx]<>tab) do
               begin              { non-blank data left...        }
                  replist[rx,tx] := input[lx];
                  tx := tx+1;     { copy a byte and step.         }
                  lx := lx+1
               end;
            rx := rx+1            { advance token index           }
         end;
      if lx<=lz then              { ran out of token-space        }
         writeln('Ignoring command after column ',lx:3)
   end;
{*************************************************************
* Procedure to open the input file.  The name comes from the   *
* zero-th token in 'replist'.  If no dot is seen, the filetype *
* .SUB is appended to it.  A few invalid filenames could get   *
* past this routine's checks, but Pascal would trap on them.   *
*************************************************************}
procedure open_in;                { open input file (token 0)     }
   var fname : STRING 14;
        f     : 1..14;
        sawdot: boolean;
   begin
      okfile := false;            { set return code for failure   }
      sawdot := false;            { haven't seen a dot yet        }
      SETLENGTH(fname,0);         { clear filename string         }
      for f := 1 to 14 do         { allow 'x:filename.typ'        }
         if replist[0,f]<>eos then  { more data coming    }
            begin
               APPEND(fname,replist[0,f]);
               if replist[0,f]='.' then sawdot:=true
            end;
      f := LENGTH(fname);         { length of filename found      }
      if f>0 then                 { got some name bytes but...    }
         if ( f<11 )  or          { not over A:FILENAME[.XYZ]     }
            (sawdot and (replist[0,15]=eos)) then
            begin
               if not sawdot then APPEND(fname,'.SUB');
               reset(fname,infile);
               if not eof(infile) then okfile := true
               else
                 writeln(fname,' -- not found or empty.')
            end
         else                     { name too long.                }
            writeln(fname,' -- is too long a filename.')
      else                        { no token zero data at all     }
         writeln('A filename is required.')
   end;
{*************************************************************
* Function to put one character into a submit record.  The     *
* data length of the chosen record acts as both a length and   *
* as an index to the *last* byte put in the record.  If the    *
* record fills up -- 126 bytes max -- return false.            *
*************************************************************}
function stowbyte(b:subspan;c:char): boolean;
   var q : linecnt;
   begin
      with buffer[b] do
         begin
            q := sublen;
            if q<mxsubline then { ok to stow the byte              }
               begin
                  q := q+1;
                  subdata[q] := c;
                  sublen := q;
                  stowbyte := true
               end
            else                   { this record is full           }
               stowbyte := false
         end
   end;
{*************************************************************
* Function to place one replacement token in a submit record.  *
* The input character is assumed mnctok..mxctok.  If the       *
* submit record fills up, return false.                        *
*************************************************************}
function replace(b:subspan;c:char):boolean;
   var r : 0..mxtok;
       t : linespan;
       f : boolean;
   begin
      f := true;                  { record not full (yet)         }
      r := ord(c)-ord(mnctok);    { org-0 index to replist        }
      t := 1;                     { set for 1st byte              }
      while (replist[r,t]<>eos) and f do
         begin                    { token data left, space left..}
            f := stowbyte(b,replist[r,t]);
            t := t+1              { copy and step                 }
         end;
      replace := f
   end;
{*************************************************************
* Function to place a control character in a submit record.    *
* The input character is assumed to be in @..DEL.  If the      *
* record fills up, return false.                               *
*************************************************************}
function control(b:subspan;c:char):boolean;
   var n : 0..127;
   begin
      n := ord(c);
      if c>'_' then n := n-32;    { lower case to upper case      }
      n := n-64;                  { make control character        }
      control := stowbyte(b,chr(n))
   end;
{*************************************************************
* Procedure to load one record from 'input' to the current     *
* submit record 'buffer[b]'.  Handle $ replacement and control *
* characters.  If the record fills up (stowbyte returns false) *
* then set 'okfile' false and write a message.                 *
*************************************************************}
procedure load_one(b:subspan;l:line);
   var lx : lineover;             { index over line (to 128)      }
       lz : linecnt;              { count of good characters      }
        c : char;                 { current character             }
        f : boolean;              { record-full flag              }
   begin
```

```
lz := scanback(l);              { drop trailing blanks          }
lx := l;                        { set for 1st character         }
f  := true;                     { record not full (yet)         }
while (lx<=lz) and f do         { good bytes until full         }
   begin
   case l[lx] of                { split '$', '^', other          }
   '$':{ ***** substitution for '$' ******************* }
        begin
          if lx<lz then          { there is another character }
            c := l[lx+1]         { check it out: $ or 0..9    }
          else                   { $ at end of line...        }
            c := eos;            { ...force an error message  }
          if ((c>=mnctok) and (c<=mxctok)) or (c='$') then
            begin
              if c<>'$' then f:=replace(b,c)
              else           f:=stowbyte(b,c);
              lx := lx+2;
            end
          else                   { not a valid dollar sign    }
            begin
          writeln('Ignoring $, line',b:3,' column',lx:3);
              okfile := false;
              lx := lx+1
            end
        end
    '^':{ ***** substitution for '^' ******************* }
        begin
          if lx<lz then         { there is another character  }
            c := l[lx+1]        { check it: @..DEL or ^       }
          else                  { at end of line...           }
            c := eos;           { ...force an error           }
          if ((c>='@') and (c<=chr(127))) or (c='^') then
            begin               { valid substitution          }
              if c<>'^' then f:= control(b,c)
              else           f:= stowbyte(b,c);
              lx := lx+2
            end
          else                  { not a valid up-arrow        }
            begin
          writeln('Ignoring ^, line',b:3,' column',lx:3);
              okfile := false;
              lx := lx+1
            end
        end
   else:{ ***** transmission of any other character **** }
        begin                   { ordinary character          }
          f := stowbyte(b,l[lx]);
          lx := lx+1
        end
   end {case};
   end {while};

      if not f then              { the line overflowed           }
         begin
            okfile := false;     { keep on, but don't submit     }
     writeln('line ',b:3,' too long, truncated at ',lx-1:3)
         end
   end{load_one};
{*************************************************************
* Procedure to load the input file into the array of submit    *
* records 'buffer'.  If more than mxsub lines are received,    *
* set okfile false and write a message.  Count the records.    *
* At this point it is known that eof(infile)=false.            *
**************************************************************}
procedure load_in;                { load records to buffer         }
   begin
      recount := 0;               { no records so far              }
      repeat
         begin
            readln(infile,input); { next line                     }
            recount := recount+1;{next slot                       }
            load_one(recount,input); { load a line to a record }
         end
      until (eof(infile)) or (recount=mxsub);
      if not eof(infile) then
         begin
            okfile := false;      { don't submit                   }
          writeln('Too many records, only ',mxsub:3,' allowed')
         end
   end;
{*************************************************************
* Procedure to write all records in *reverse* order to the     *
* file $$$.SUB.  Count of records is in 'recount'.  The null   *
* byte that terminates the data in each line is inserted here. *
**************************************************************}
procedure writeall;               { write the submit file          }
   var bx : subspan;              { index over buffer              }
       lx : linecnt;              { from 0 to 127 over subdata     }
   begin
      rewrite('$$$.SUB',outfile);
{
 the following glitch compensates for a bug? feature? in
 PASCAL/Z.  It buffers 256 bytes of any file, and always writes
 the last 256 bytes.  If this program writes an odd number of
 128-byte records, PASCAL/Z will effectively add one more 128-
 byte record at the end of the file, filled with ^Z bytes.  That
 is the first record processed from $$$.SUB.  The unsatisfactory
 solution is to write an even number of records including an
 empty one if necessary.
}
      if odd(recount) then recount := recount+1;
      for bx := recount downto 1 do
         with buffer[bx] do begin
            lx := sublen+1;
            subdata[lx] := eos;   { stick in eos after data        }
            write(outfile,buffer[bx]){ write to $$$.SUB            }
         end
   end;
{*************************************************************
*                                                              *
*               M A I N     P R O C E D U R E                  *
*                                                              *
**************************************************************}
begin
   clear;                         { initialize buffer and tokens }
   get_tail;                      { get command tail               }
   load_tok;                      { load token list from tail      }
   open_in;                       { open the input file (token 0)}
   if okfile then load_in;        { load input records to buffer   }
   if okfile then writeall        { write the submit file          }
   else           writeln('No submit done.')
end.
```

CP/M Disk Directory and Table Secrets Revealed

by Andrew Klossner

One of the less fortunate aspects of the documentation distributed with CP/M version 2.2 is its taciturn descriptions of internal system structures, such as the information contained in the disk directory and the disk tables within the BIOS. Although the "search for first" and "search for next" BDOS calls provide a means of reading the directory, the only clue to its contents consists of hints that it resembles the first 32 bytes of a File Control Block (FCB). The *CP/M 2.0 Alteration Guide* gives just enough information about the resident disk tables to allow a custom BIOS programmer to fabricate new tables without understanding their meaning or use.

In order to write programs such as intelligent directory listers or disk backup/restore software, an understanding of the directory structure is necessary. This in turn requires an understanding of the BIOS disk tables, which direct interpretation of the directory fields. The presentation of such detailed information is the purpose of this article.

Disk Format

CP/M considers a disk to consist of a fixed number of tracks, each of which contains a fixed number of sectors, each of which is of a fixed size. Tracks are numbered sequentially starting at 0; sectors within a track are numbered sequentially starting at 1. A single-sided, single-density, 8" soft-sectored disk (the standard for transporting CP/M files) contains 77 tracks, each of which holds 26 sectors, each 128 bytes in length. Such a disk is referred to herein as a "standard floppy."

CP/M skips the first few tracks on a disk, so that data such as a bootable system image can be placed there. The number of tracks skipped depends upon the particular disk drive and BIOS, but is always two for a standard floppy.

With floppy disks of dual or greater density and with hard disks, the number of bytes in a sector is often a multiple of 128 such as 256, 512, or 1024. In this case the BIOS "deblocks" these physical sectors into records, maintaining the illusion of 128 byte sectors for the BDOS and the application program. This frees the programmer from the need to provide for records of different lengths on different systems.

In a completely independent process, the BDOS groups adjacent sectors into "blocks" (also called "clusters" or "BLS units") for purposes of allocation. Rather than allocate one sector at a time when a file is extended, CP/M allocates several sectors at once. This decreases fragmentation of the file, the condition in which successive sectors are in wildly different parts of the disk, causing many head seeks when the file is accessed. It also decreases the total amount of information that must be stored to locate a file, and thus cuts down on the amount of directory access needed to read the file. For a standard floppy the cluster size is eight, and there are eight logical sectors per block. For larger disks, the blocking factor is often 32 or greater. Blocks are numbered sequentially from 0.

The first few blocks are reserved for the disk directory. This consists of a number of 32-byte "entries" or "slots," which contain the file name and user number, and describe which sectors on the disk correspond to which records of the file. A large file can require several such 32-byte slots. Each slot corresponds to a physical extent.

The Disk Parameter Header

There is a separate Disk Parameter Header (DPH) within the BIOS for each drive. This table is 16 bytes long, and consists of pointers to other tables describing various aspects of the disk subsystem. The address of a DPH can be obtained in an assembler program by loading the number of the disk drive (0 for disk A, 1 for disk B, etc.) into register C and then calling the SELDSK BIOS entry, as with the following code fragment:

```
          MVI C,0         ; Get DPH for disk A
          LHLD 0          ; HL points to the BIOS
          MVI L,1B        ; HL -> SELDSK entry point
          CALL CALLHL     ; Call that routine
                          ; Here with HL -> DPH for disk A
          ...
          ...
CALLHL:   PCHL
```

Andrew Klossner, P.O. Box 283, Wilsonville, OR 97070.

If the disk drive specified by the value of register C does not exist, SELDSK is supposed to return with HL containing zero. However, many custom BIOS versions do not adequately check for this contingency, and return a nonsense pointer.

The structure of the DPH is as follows:

```
XLT:     DS      2          ; Address of sector translation table
SCR1:    DS      2          ; BDOS scratchpad word
SCR2:    DS      2          ;  "       "       "
SCR3:    DS      2          ;  "       "       "
DIRBUF:  DS      2          ; Address of the directory sector buffer
DPB:     DS      2          ; Address of the disk parameter block
CSV:     DS      2          ; Address of the directory checksum vector
ALV:     DS      2          ; Address of the allocation vector
```

The sector translation table, addressed through XLT, specifies the sector interleave factor for a particular disk drive to the BDOS. To interleave sectors on a disk is to assign successive data to each n'th sector, where the number, n, is called the *interleave factor*. For example, the format for a standard floppy includes an interleave factor of 6. This means that the first sector accessed in a track is number 1; the second is number 7; the third is number 13; and so on.

By skipping several records in this mapping of logical sectors onto physical sectors, the BDOS gives the application program time to digest a record and request the next before it goes by the disk head. If no sectors were skipped(the interleave factor=1),the program would normally have to wait an entire revolution for each successive sector, leading to a condition known among operating systems programmers as "blowing revs."

Since all interleaving is done by the BDOS, which "believes" that all physical sectors are 128 bytes in length, the interleave formula maps logical 128-byte records onto pseudo-physical 128-byte sectors. After the BDOS performs this mapping, it requests the sector from the BIOS, which performs deblocking for disks whose sectors are larger than 128 bytes.

The sector translation table is used to perform the logical-to-physical mapping without incurring the overhead of multiplication by the interleave factor and division by the number of sectors in a track. The table for a standard floppy contains 26 bytes, one for each sector in a track; the first byte contains the physical sector number for logical sector one, the second the sector for logical sector two, and so on. The entire table is:

```
XLT0:   DB      1,7,13,19,25,5,11,17,23,3,9,15,21
        DB      2,8,14,20,26,6,12,18,24,4,10,16,22
```

If two disk drives have the same interleave factor, the *XLT* fields in their respective DPH's can point to the same sector translation table. The XLT field can be zero, which indicates to the BDOS that the interleave factor is 1 (i.e., logical sector n is mapped to physical sector n).

The *DIRBUF* field in the DPH points to a 128 byte "scratchpad area," which is used by the BDOS to hold records from the directory. All DIRBUF fields point to the same area.

The *DPB* field points to the Disk Parameter Block for the drive. This block contains a number of one- and two-byte values which describe the format of the disk, and is presented in detail later. Two disks can point to the same DPB.

The *CSV* field addresses a "checksum vector." By computing the checksum of each 128-byte directory record whenever the disk's directory is accessed, and comparing those checksums with their previous values from the CSV, the BDOS can make a guess as to whether the disk volume has been changed without its knowledge, and set the software write lock flag for that disk if so. The number of directory records to be so checked is one of the numbers in the DPB, and is typically maximal for a floppy disk and zero for a hard disk whose volume cannot be removed (and for which checking is unnecessary). The checksum vector consists of one byte for each checked directory record. Since a record is 128 bytes, it contains four directory entries.

The *ALV* field points to an "allocation vector." This consists of one bit for each block on the disk, and is used to record whether the disk is free for allocation. The ALV may indicate that blocks which do not appear in any directory entry are not free, because they have recently been added to an output file which has not yet been closed.

The Disk Parameter Block

The DPB for a disk can be found in either of two ways. One is to locate the disk's DPH, then follow the address in the DPB field. The other is to select that disk with the BDOS "select disk" call (number 14), and then issue the "get addr(disk parms)" call, number 31.

The structure of the DPB is as follows:

```
SPT:    DS      2          ; Sectors per track
BSH:    DS      1          ; Block shift factor
BLM:    DS      1          ; Block maximum
EXM:    DS      1          ; Extent mask
DSM:    DS      2          ; Drive storage maximum
DRM:    DS      2          ; Directory slot maximum
AL0:    DS      1          ; First byte of directory allocation
AL1:    DS      1          ; Second byte of directory allocation
CKS:    DS      2          ; Check vector size
OFF:    DS      2          ; Number of reserved tracks
```

SPT contains the number of pseudo-physical 128-byte sectors per track. This is 26 for a standard floppy. In the case of a dual density disk with, for example, 15 sectors of 512 bytes apiece, SPT would be 60, because there are four 128-byte "sectors" in each physical sector.

BSH is the base two logarithm of the number of records in an allocation block. It is the number of bits to left-shift a block number in order to turn it into a logical record number. For a standard floppy, with eight records per block, BSH is 3.

BLM is the maximum record number within a block (where records are numbered from 0), and is therefore one less than the number of records within a block. It is also a mask which, when applied to a record number, yields the relative record number within its block. BLM is 7 for a standard floppy.

EXM is the so-called "extent mask," and is used in interpretation of the format of a directory slot. It is 0 for a standard floppy.

DSM is the maximum block number on a disk, or one less than the number of blocks. DSM is 242 for a standard floppy.

DRM is the maximum directory slot number, or one less than the number of directory slots. DRM is 63 for a standard floppy, which has 64 directory slots.

AL0 and *AL1* are used to indicate which blocks contain the directory. If the high bit of AL0 is on, it means that block 0 is within the directory; the next-to-high bit of AL0 corresponds to block 1, and so on to the least significant bit of AL0, which corresponds to block 7. Similarly, the bits of AL1 correspond to blocks 8 through 15. For a standard floppy, AL0 is 192 (the two high bits are on) and AL1 is 0, indicating that the first two blocks comprise the directory.

CKS is the size of the directory checksum vector, which contains one byte for each record of directory which contains checked slots. CKS is one-fourth the number of checked directory entries. For a standard floppy CKS is 16 (64/4).

OFF is the number of reserved tracks at the start of the disk. On a standard floppy, tracks 0 and 1 are reserved for the bootable CP/M image, and so OFF is 2.

Directory Slot Structure

Each directory slot is either unused or describes one physical extent of a file. The first physical extent, number 0, is termed the "base extent."

A logical extent consists of 128 records or 16Kbytes. The number of logical extents in a physical extent on a given disk is (1 SHL EXM), or 1 shifted left EXM bits, where EXM is the extent mask field of the DPB for that disk. For example, on disk with EXM=1, there are 2 (1 SHL 1) logical extents per physical extent, and so a directory entry describes two logical extents, or up to 32K of data.

The 32 bytes of a directory entry are structured as follows:

```
USERNO: DS    1     ; User number, or hex E5 for free slot
FILENM: DS    8     ; File name, padded with blanks
FILETP: DS    3     ; File type (extension), padded with blanks
EX:     DS    1     ; Extent and record count overflow field
S1:     DS    1     ; Reserved for system, normally 0
S2:     DS    1     ; High bits of extent number
RC:     DS    1     ; Non-overflow portion of record count
INDEX:  DS    16    ; Allocation block indices
```

The *USERNO* field is used both to indicate whether a slot is free and to specify the user number for the file. The value E5 (hex) designates a free slot. This value was chosen because it is the value of every byte on a freshly formatted disk, which therefore looks like an empty disk instead of a disk full of garbage, as it does on other systems. User numbers range from 0 to 15. The USERNO field for each extent of a file is the same.

FILENM and *FILETP* appear in the directory slot exactly as they do in a File Control Block: eight characters of file name and three characters of file type. The high bit of the first filetype byte is the read/only flag, and the high bit of the second filetype byte is the SYS file flag. The FILENM and FILETP fields are the same for each extent of a file.

EX, *S2*, and *RC* determine the extent number and the number of records in the physical extent. The number of the first logical extent contained in the physical extent is given by (EX AND (NOT EXM))+(S2*32); that is, the EX value is AND'ed (masked) with the logical complement of the extent mask EXM (from the DPB), then added to the product of the S2 field and 32. For compatibility with CP/M version 1, EX contains only the low five bits of the extent number. The overflow in S2 allows for files greater than 512Kbytes (32 logical extents).

The number of 128 byte records in the physical extent is given by the formula ((EX AND EXM)*128)+RC; that is, the EX value is AND'ed with the extent mask EXM, multiplied by 128 (or shifted left seven bits), and added to RC. When EXM is 0, as for a standard floppy, this formula degenerates to simply RC. This figure is only correct when the extent does not contain random access "holes."

The last 16 bytes of the directory slot, *INDEX*, are block indices. They are interpreted in either of two ways. If the disk contains 256 or less blocks (the DSM field in the DPB is 255 or less), they are sixteen one-byte block numbers; that is, each byte contains a block index. If the disk contains 257 or more blocks (the DSM is 256 or greater), they are eight two-byte block numbers; each pair of bytes is treated as a word, with the low byte before the high byte, and the word is a block index. Thus, a physical extent can contain a maximum of eight or sixteen allocation blocks, depending upon the disk parameters. Unused block indices, which correspond to unallocated blocks (holes), contain zero.

Directory Programming Techniques

The best way to scan a disk directory is to use the "Select Disk" BDOS function (number 14) to designate the disk to be scanned, then fill an FCB with ASCII question marks and call the "Search for First" and "Search for Next" BDOS functions, numbers 17 and 18. If the first byte of the FCB contains a question mark, the BDOS returns all directory slots, including unallocated slots, slots for other users, and slots corresponding to non-base extents, with the exception that the free slots after the last allocated slot are not returned.

Making changes to a directory is harder. If the desired change is something that a BDOS call will perform, such as deleting a file or changing its attributes, that call is the easiest and safest means of making the change. For unorthodox directory manipulation, such as collecting bad disk blocks and allocating them to a hidden file to get them out of the way, it is possible to create a file with a legitimate FCB, then set the INDEX fields to the desired block numbers and close the FCB. The first fifteen bytes should be left unchanged between the create and the close, so that CP/M will find the directory entry at close time. Also, the high bit of S2 should be cleared just before the close. Experimental evidence indicates that, when set, this bit directs CP/M to take no action when a close is issued. The bit is cleared by read/write calls.

Some applications will require the programmer to call upon the BIOS to write individual sectors directly into the directory. In doing this, the number of reserved tracks and the interleave factor must be taken into account. The SECTRAN entry point within the BIOS can be called to compute the logical-to-physical interleave sector mapping.

One convenient use of directory patching is to give the same program image (.COM file) to all users without having to make a separate copy for each. This can make quite a bit of difference when there are several users of a non-removable hard disk. The technique is to make a duplicate copy of each directory slot used by the file, changing only the USERNO field. This fools the BDOS into thinking there is a file for each user, but each file occupies the same disk blocks. The technical term for the additional directory entry is "link," and is a standard feature in the Unix and Data General RDOS operating systems. It's a good idea to make each link "read/only," and to be careful to remove links with directory patching, never by asking CP/M to delete the file, since this would cause a disk block to appear to be both free and allocated. ■

ANDREW KLOSSNER is chief engineer at Knowlogy, where he develops systems software for personal computing environments. With an M.S., Computer Science degree from UCLA, he specializes in the design and implementation of algorithmic programming languages. His best creative ideas come to him when backpacking in the Cascade Mountains.

More Modifications For The SDS VDB-8024

by Richard Bowersox

> This article describes hardware and software mods for the SDS VDB-8024 that allow it to return cursor information to the host processor. Routines adding two new functions to the video board—Insert Character and Delete Character—are also included. Note: All routines listed will assemble under Intersystems IASM Z80 assembler and will fit into the same 2708 EPROM as the original control firmware.

The SD Systems VDB-8024 video board is one of many on the market. It is distinguished by its use of an onboard Z80 to control an SMC 5027 video controller circuit. This board and some software additions have been previously described in *Microsystems* ("Modifying the SDS VDB-8024," by Jon Bondy, Jan/Feb 1980).

I have used this board for about two years and have been generally pleased with its operation. There were, however, several features it did not offer that I wanted:

1) Erase to end of line
2) Erase to end of screen
3) Insert character at cursor location
4) Delete character at cursor location
5) Determine cursor location
6) Determine character under cursor

The first four routines are simple additions to the VDB-8024 firmware. Mr. Bondy described the implementation of the first two items in his article. In my version, I replaced two of the original VDB functions with these. What was "set control bit 1" is now erase to EOL and "clear control bit 1" is erase to EOS. The firmware already detects escape sequences, so I have used additional ones to call the remaining routines. Listing 1 is the code fragment to be inserted into the cursor control routine section that will decode the new escape codes.

Listing 2 contains the two routines to INSERT or DELETE a character from the current line. A space is inserted at the cursor position and a character is deleted from the position under the cursor. In this way, the functions are complimentary and a delete "un-does" an insert. These functions affect only the line on which they occur, so an insert on an 80 character line will send the last character off into the bit bucket. The delete routine will not operate on the last character in the row and is ignored if the cursor is in the first column. If desired this function could be easily changed to delete the character under the cursor.

After I had the first four functions running, all that was left was to implement the ones that I wanted the most and were the least readily attainable. SD Systems apparently decided that users of this board would not need to know where on the screen the cursor was located or what character was under the cursor. To simplify my other programming efforts I wanted that information available. The VDB-8024 has provisions for attaching a keyboard to be used for the system console device. With a little additional software and a some extra board wiring the keyboard port can be used to return data to the S-100 bus. Since my system uses a separate I/O card for keyboard interfacing I chose to use this method.

The hardware modifications consist of connecting the onboard data bus to the inputs of the keyboard latch/buffer chip (U35 on my revision 1 board) and supplying a pulse to U37 to simulate a key pressed strobe. I picked the data lines off the outputs of U34, which is the input data buffer chip. The strobe is provided by pin 12 of U26 (a two line to four line decoder chip). The clock input of U37 is positive edge-triggered and the output of the decoder is active low so an inverter must be used to provide the correct strobe polarity. U2 has an unused inverter stage which I used for this function. Below is a list of interconnections to be made:

Signal	**From**	**To**
D0	34.5	35.18
D1	34.2	35.17
D2	34.9	35.4
D3	34.12	35.8
D4	34.6	35.3
D5	34.15	35.7
D6	34.19	35.13
D7	34.16	35.14
Strobe*	26.12	2.9
Strobe	2.8	13.6

(Where 34.5 indicates IC U34 pin 5.)

Richard L. Bowersox, 460 North Armando, Apt. H5, Anaheim, CA 92806.

The routines to retrieve cursor row and column address and character under cursor are given in Listing 3. Details of their operation are given in the comments, but a brief explanation of the board design is needed to understand their function.

Since the board was not originally set up to send information back to the host system there is no provision for checking to see if data sent by it has been read. This constrains the software to require that the system ask for one character at a time, and so dictates the use of separate row and column returning routines. Also, the port addresses of the on-board Z80 are only partially decoded. Bits 5 and 6 are used to address either the data input port (20h) the keyboard ready flip-flop (30h) or the video attributes control latch (10h). There is no port addressed at 00, so that address is used to strobe data into the output (keyboard) latch for transmission to the main computer. This addressing method does have one problem, the CRT controller chip is addressed at ports 80h-8Fh. Anytime the controller is accessed, bits 5 and 6 go low and data will be strobed into the latch. The routines get around this problem by initially reading from the keyboard flip-flop address which clears the flip-flop and output latch.

The software required to access these new functions is simple and can follow one of two methods. The first method is to send, say, a read row command (ESC R) to the video board, and check the "keyboard" status flag until the character is available for reading. However, since some time is needed for the board to decode the instruction and clear the flag (from controller writes, see above) a time delay must be used between the command and flag test. The second method is to simply read from the board (clearing the flag and latch) ignore the result and then go through the read sequence without the delay between command and flag checking stages.

The value returned from a call for the position is one byte in the range 0-17h for row and 0-7Fh for the column. The original VDB firmware requires an offset of 20h when sending a new cursor position to the board, but I saw no use in returning the position with this offset. Of course a simple change of the code could be used to add any offset desired.

The board now performs all of the operations that I originally wanted—and performs them well. The hardware changes were not overly involved, and there is still a fair amount of room left in the onboard EPROM. Now I'm beginning to wonder about the other functions available in the CRT-5027. ■

RICK BOWERSOX is on the technical staff at Rockwell International. He regularly uses minicomputers for modeling electromagnetic effects in satellites. He first fell in love with computers when he was exposed to a programming class in the sixth grade. His S-100 system began with a single card computer four years ago and has expanded considerably since then.

Listing 1: *Additions to cursor control routine to decode escape sequence to access new functions.*

```
Crtin   equ     20h         ; Input port from which characters
Crtout  equ     0           ; Output port to which characters

;
;       Cursor control routines
;
ESC:    STC
        CMC                         ; CLEAR CARRY FLAG
        EI                          ; ENABLE INTERUPTS
        HLT                         ; WAIT FOR NEXT CHAR (= MAYBE)
        IN      CRTIN               ; GET IT
        RES     7,A                 ; MAKE SURE HIGH BIT IS OFF
        CPI     'I'                 ; TEST FOR INSERT
        JZ      INSERT
        CPI     'D'                 ; TEST FOR DELETE
        JZ      DELETE
        CPI     'R'                 ; TEST FOR ROW
        JZ      GETROW
        CPI     'C'                 ; TEST FOR COLUMN
        JZ      GETCOL
        CPI     'W'                 ; TEST FOR 'CHARACTER UNDER CURSOR ?'
        JZ      GETCHR
        CPI     '='                 ; TEST FOR '='
        RNZ                         ; NOPE -- IGNORE IT AND RETURN
        STC                         ; WAIT FOR NEXT CHAR
```

Listing 2: *Routines to INSERT space at cursor location and DELETE character before cursor.*

```
Numcol  equ     80          ; Number of columns
;
;       Insert character at cursor position
;
INSERT: MVI     A,NUMCOL-1          ; ARE WE AT END OF ROW ?
        SUB     D
        RZ                          ; IF SO, RETURN
        PUSH    D                   ; SAVE REGISTERS 'DE' AND 'BC'
        PUSH    B                   ;  REGISTER 'HL' END UP IN CORRECT POSITION
        MOV     C,A                 ; 'C' NOW CONTAINS NUMBER OF CHARACTERS
        XRA     A                   ;   FROM CURSOR TO END OF LINE
        MOV     B,A
        DAD     B                   ; INDEX 'HL' TO END OF LINE
        MOV     D,H                 ; MOVE THIS VALUE TO 'DE'
        MOV     E,L
        DCX     H                   ; POINT 'HL' TO START OF STUFF TO MOVE
        LDDR                        ; BLOCK MOVE THE STUFF
        INX     H                   ; RETURN M TO PROPER PLACE
        MVI     M,' '               ; INSERT A SPACE
        POP     B                   ; RESTORE REGISTERS
        POP     D
        RET
;
;       Delete character from position before cursor
;
DELETE: XRA     A                   ; SET UP TO CHECK POSITION IN ROW
        ADD     D                   ; ARE WE AT BEGINNING OF ROW ?
        RZ                          ; IF SO, RETURN
        MVI     A,NUMCOL            ; CALCULATE NUMBER OF CHARACTERS
        SUB     D                   ;  TO END OF LINE
        PUSH    H
        PUSH    D                   ; SAVE REGISTERS
        PUSH    B
        MOV     C,A                 ; SET UP TO BLOCK MOVE CHARACTERS
        XRA     A                   ;  FROM CURSOR TO THE END OF LINE
        MOV     B,A                 ;  ONE POSITION LEFT
        MOV     D,H
        MOV     E,L
        DCX     D
        LDIR                        ; BLOCK MOVE STUFF
        DCX     H                   ; POINT HL TO PROPER POSITION
        MVI     M,20H               ; PUT SPACE AT END OF ROW
        POP     B
        POP     D                   ; RESTORE REGISTERS
        POP     H
        JMP     CTRH                ; JUMP TO CURSOR BACKSPACE ROUTINE
```

Listing 3: *Read cursor row and column and character under cursor routines.*

```
KbFlop  equ     30h             ; Keyboard flip-flop clear port
Crtout  equ     0               ; Output port to which characters
                                ;  are sent back to the computer

Numrow  equ     24              ; Number of rows
Numcol  equ     80              ; Number of columns
;
;       Get cursor row address
;
GETROW:
        IN      KBFLOP          ; CLEAR/RESET OUTPUT LATCH
        MOV     A,B             ; GET LAST DISPLAYED ROW
        CMP     E               ; IS IT >= CURRENT LINE
        JP      LGEC            ; YES, BRANCH
        MOV     A,E             ; GET CURRENT ROW
        SUB     B               ; SUBTRACT LAST DISPLAYED LINE
        DCR     A               ; ADJUST TO SHOW ACTUAL ROW ON SCREEN
        JMPR    GR2             ; GO TO OUTPUT ROUTINE
LGEC:   MVI     A,NUMROW-1      ; NO, GET NUMBER OF ROWS - 1 INTO A
        SUB     B               ; SUBTRACT LAST DISPLAYED ROW
        ADD     E               ; ADD CURRENT ROW (TO COME UP WITH DISPLAYED R
OW
GR2:    OUT     CRTOUT          ; SEND DISPLAYED ROW TO COMPUTER
        RET
;
;       Get cursor column address
;
GETCOL:
        IN      KBFLOP          ; CLEAR/RESET OUTPUT LATCH
        MOV     A,D             ; GET CURSOR COLUMN ADDRESS
        OUT     CRTOUT          ; SEND IT TO THE COMPUTER
        RET
;
;       Get character under cursor
;
GETCHR:
        IN      KBFLOP          ; CLEAR/RESET OUTPUT LATCH
        MOV     A,M             ; GET THE CHARACTER AT THE CURSOR POSITION
        OUT     CRTOUT          ; SENT IT TO THE COMPUTER
        RET
```

Mince - A New Text Editor

by Chris Terry

Mince is the latest flower to bloom in the fertile fields of microcomputer word-processing. It is the text editor portion of a package called AMETHYST, supplied by Mark of the Unicorn. The complete AMETHYST ($350) consists of:

- MINCE, a text editor.
- SCRIBBLE, a text formatter (partial source code is also included).
- The C language source code for the Mince command set and support routines for the commands and terminal. The code does not include the multiple window display routines, buffered file I/O, and other routines at the core of Mince.
- The BDS C compiler.

The components of the package are available separately. If you already have the BDS C compiler you can get the rest of AMETHYST for $275. If you don't want to tinker with the C code, you can order MINCE at $175, SCRIBBLE at $175, or both together at $275. This software is available from many software suppliers, or can be purchased directly from: Mark of the Unicorn, P.O. Box 423, Arlington, MA 02174,(617)489-1378.

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

What It Will Do For You?

Having experimented with Mince for a few weeks, I can confidently say that Mince is excellent value for the money. It is by no means "just another editor," but a very fine one. If Scribble is as powerful, flexible, and easy to use as Mince, then the combination at $275 will be an unbeatable value. I expect Amethyst to prove a formidable rival to WordStar and Magic Wand. Mince has all of the features that one generally expects of a good editor—fast cursor movements, global search, global replacement, the marking of start and end of a text region to move it bodily to another place or kill it, the ability to incorporate all or part of another file, and so on. In addition, it has one feature that I have not seen in other CP/M editors; that is, the ability to read several files into buffers and work on any two of them simultaneously by means of a dual window on the screen. All of the editing features are available in both windows, and text can be transferred from one window to the other via the Kill buffer. I do have a few reservations, mostly connected with my personal preferences (prejudices?) about commands, and I will discuss these later, since they may be relevant to you too. Meanwhile, I'd like to describe the features of Mince in some detail.

System Requirements

A 48K CP/M system is required. The terminal or video display board must have computer-controllable cursor positioning.

Performance

Reliability. Mince keeps a close and constant watch on the keyboard. During all display modifications, frequent keyboard checks are made, and the 80-character input buffer is big enough to hold what you type while the screen is changing. The only time you may lose characters is when disk operations are in progress. During a sector read or write, the CPU has no time to check the keyboard. However, Mince waits until the keyboard has been idle for a few seconds (you can set this delay during configuration) before starting a disk operation, and puts a message on the screen asking you to wait. This is no hardship.

Compatibility. Mince creates standard CP/M ASCII files, with a hard CR-LF at the end of each line. I have had no trouble at all in displaying or processing Mince files with TYPE, ASM, LINKASM, XREF, and other CP/M utilities, as well as Basic-80 version 5.2 and the TSC text formatter. People who send me raw WordStar files over a telecommunications link get my curses, because of the absence of line feeds—I can't read them or print them without WordStar, which in a 48K system (the minimum for Word Star) performs with all the speed and elegance of a pregnant hippopotamus. Mince files will never give you these problems.

Multiple Files. Mince does all its composing in a series of "Swap" buffers kept on the disk. You can display, edit, and combine portions of buffers that you have loaded from several different disk files. You can have two windows on the screen at once, each looking into a different buffer. Normally the windows are equal in size, but you can temporarily enlarge one and reduce the other. You can move either window to any point in its associated file and apply all editing commands to either window, independently of the other. You can also move text from one window to the other, to combine portions of the files, via the Kill buffer.

Backup. Mince does not automatically rename the source file as a BAK file and write a new, changed file. Instead, it prompts you for a filename; if none is given (just hit Return) the last filename used for a read of write is used.

If you want to keep the original file as a backup, you must write to a file with a different name. So be careful if you have been merging portions of different files; if you don't specify a filename for the Write, you could overwrite one of your sources. On the other hand, disks don't fill up quite so quickly. With ED and its derivatives (such as Word Master, which has been my favorite for three years), files multiply like rabbits. But the backup file has saved my neck once or twice when I had memory problems that made printer's pie of my newly edited version just before it was written to disk. Writing to a Mince file does not cause an exit to CP/M; that requires a separate command.

The Write command saves the work you have done so far and leaves you in Mince ready to do more editing—it is like the H command of ED/Word Master.

Principal Editing Features

Fast Cursor Movements. Mince provides horizontal cursor movements forward or backward by character, by word, by sentence, and by paragraph. In addition any command can be repeated any number of times by the Universal Argument (Ctrl-U followed by the number of repetitions). Vertical movement is by line or by screen, or directly to the beginning or end of the file.

Insertion. In the normal entry mode, insertion at the cursor is automatic, pushing existing text to the right. In the page mode (see below), existing text is overwritten; characters to be inserted must be preceded by a Ctrl-Q.

Deletion. Deletion can take place forward or backward from the cursor, by character, by word, or by sentence.

Kill. There is a difference between deletion and killing. Deletion removes the specified text entirely, and it is lost. Killing one or more lines removes them from the screen and the current buffer, but saves them in a special Kill buffer from which they may later be recovered by a Yank command, provided that no movement or insertion commands were given after the Kill. Availability for yanking is indicated by a + sign in the command line. However, groups of lines may be assembled in the Kill buffer and retrieved by "turning on" the + sign; after this has been done, movements and insertions do not affect the +. Thereafter, killing additional lines appends them to the Kill buffer instead of replacing what was there before.

Transpose. This is a nifty command which transposes the character/word at the cursor and the character/word before it. It is useful when one is tired and starts typing "hte" and so on.

Regional Marking. The NUL (Ctrl-@, 00H) is used to mark the start of a region of any size; the end of the region is marked by the cursor. The Wipe Region command kills the marked region, saving the killed text in the Kill buffer. The Copy Region command copies the marked region to the Kill buffer, but leaves the text unchanged in its original position. An Exchange command swaps the marker and the cursor; this is useful for determining the boundaries of a region that is to be wiped or copied.

Searching. The Search command is a standard search from the cursor to the end of file for the specified string.

Replacement. The Replace commands are more sophisticated than in most editors. The Query/Replace command prompts you for Oldstring and Newstring, then searches from the cursor to end of file for Oldstring. At each match, the user can specify Replace and Find Next; Don't Replace but Find Next; Replace and Request Confirmation (useful if you try something to see how it looks but don't like the result); Replace Rest which replaces all remaining occurrences of Oldstring without stopping for confirmation; or Exit, which terminates the operation without searching any further. If unconditional replacement is desired, the Replace command replaces all occurrences without stopping for confirmation.

Tabbing. The Set Tab Spacing command operates in two ways. If a numeric argument is given, tab points are set every N columns across the screen. If no argument is given, a tab point is set at the cursor position; thus, tabs can be individually set at arbitrary intervals. This is a great improvement on Word Master's fixed 8-character interval, and combines the tab setting features of both Document and Non-Document modes of WordStar.

Case Conversion. With the cursor under any letter of a word, the Capitalize Word command capitalizes the first letter of that word; the Uppercase Word command converts the word to all capitals; and the Lower Case Word command converts all capitals in the word to lower case.

Formatting Features

Fill Mode. This allows a line width to be set with the Set Fill Line command. When you have entered a complete paragraph, you can even up the lines with the Fill Paragraph command, which adjusts the text so that each line contains as many words as possible without exceeding the line width. A word that would overflow the specified width is pushed down into the next line (this is called word wraparound). The right margin is NOT justified, but the result looks quite good.

Center Line. This command centers the current line between limits set by the Set Indent and Set Fill Column commands, when the Fill Mode is on.

Page Mode. This allows continuous entry with word wraparound, filling the lines to the specified width. A hard CR-LF is automatically inserted to terminate each line, so the operator can type hell-for-leather without ever hitting the Return key until the end of the paragraph. Subsequent insertions are not handled automatically—the paragraph must be recomposed with the Fill Paragraph command to produce word wraparound as necessary.

Command Structure

The structure of the Mince command set is designed for easy learning and retention. Simple operations have simple commands, more complex operations have more complex commands and, for the most part, command letters are mnemonically related to the operations they perform. In the most general terms, a command consisting of a single control character operates on a small region of text; for example, Ctrl-F (written as C-F) moves the cursor one character Forward, C-B moves it one character Backward, C-N moves it to the Next line, C-P moves it to the Previous line. The C-K command Kills one line from the cursor to the end of line. For operations on larger text regions, Meta-commands are employed, consisting of the ESC character (written as M-) and a letter; for example, M-F moves the cursor Forward one word, M-B moves it Backward one word. The M-K command Kills text from the cursor to the end of the sentence. And so on.

Not all the commands fit neatly into this structure. Some consist of ESC plus a control character; the Query Replace String command is ESC + Ctrl-R, written M-C-R. And there is a fairly large group, mainly concerned with reading and writing files, changing windows, and other potentially destructive operations, that consist of C-X followed by another control or printable character. The C-X prefix is, in effect, a warning to be careful how you use these commands.

Installation and Documentation

Installation. The configuration program provided on the Mince distribution disk is excellent. It is set up for quick installation with most of the standard terminals (ADM-3A, Hazeltine, Soroc, etc) and some memory-mapped video boards. If you have an unusual terminal or board, the program prompts you for information about it. And before you freeze the configuration, a complex display is put on the screen, including a frame around the edges, material that should be stable, and material that should disappear after a few seconds. If your display does not match the very clear description, you can go back and start over. Once your configuration display matches and you say Go, the configured version of Mince will perform exactly as the manual describes.

Documentation. The manual is good Section 1, on Installation, is clear and describes exactly how to use the CONFIG program.

Section 2 is a good tutorial for users unfamiliar with any editor, and is presented in eight lessons. These are supported by text files on the disk, so that you can follow each step of a lesson exactly and see immediately if you make a mistake. The early lessons cover simple cursor movements and deletion/insertion, moving by word and sentence, buffer commands and, in Lesson 3, reading and writing files. At this point, the user should be able to create, edit, and retrieve straightforward documents. The remaining five lessons deal with all the refinements.

Section 3 is a brief Programmer's Introduction to Mince, also supported by a disk file for exercises. This is intended for users who are already familiar with at least one text editor.

Section 4 is the Mince User's Guide, explaining the general principles that one should understand in order to make effective use of Mince. It includes a general description of Mince, a glossary of terms, the parts of the display and how they are used, the commands, I/O operations, text buffers, the various modes, and the use of windows. The Command Cross-Reference at the end of this section is almost an index to the commands, by subject, showing the actual commands and cross-referencing related topics. It is good as it stands, but would have been even more useful if page numbers had been included.

Section 5 is a detailed list of the Mince commands, defining exactly how each one operates. I have not yet used every single one of the commands, but all those that I have used perform exactly as described in this section. Section 6 is a Command Summary, ordered by command character(s).

Some General Comments

Why is this editor called Mince? Well, I don't think there's any connection with mince pie (though it tastes as good). Another enthusiastic reviewer tells us that "Mince Is Not Complete EMACS"—EMACS being an interactively extensible text editor developed at MIT, on which Mince is patterned . That is, EMACS allows you to define your own commands, or to modify the action of those provided, during a terminal session. This is one big difference between Mince and EMACS— Mince is compiled, not interpreted, so that to change existing commands or add new ones you must change the source code and recompile it. I haven't seen the 'C' source code, so I don't know just how easy changes would be—but I suspect you would need to be a pretty good programmer and very familiar with 'C'.

Word processing and text editing techniques have come a long way in the last five years, and seem to generate almost as much enthusiasm, argument, and downright fanaticism as politics or religion. The points at issue are mainly to do with the human interface, and although there are many points of view on this, I have found three broad categories of user to be the most vocal:

> The hot-shot professional typographer who is used to a huge dedicated system, a display of 60 lines by 80 or 100 columns, and all the accompanying bells and whistles. This kind of user needs to compose his page on the screen and wants labeled function keys, multiple windows, and elaborate video tools to help him. He is intolerant, even contemptuous, of the software available for microcomputers and of the use of control characters for commands, which, he says, slow him down I doubt this. Studies have shown that hitting any function key off the main keypad is equivalent to four or five keystrokes for a touch typist; control characters, right on the keypad, should not take longer than three regular keystrokes on a well laid out keyboard. This user just wants the Bible printed on a postage stamp, every word readable to the naked eye.
>
> The idealist who is looking for the perfect integration of man and machine. He too, tends to despise current word processing packages, and perhaps rightly, because we really have not yet gotten around to making the machine do what is most natural to the user, instead of forcing the user to do what is convenient for the machine. The idealist often prefers English-like commands on a command line.
>
> The pragmatic user, a moderately good typist who wants reasonable facilities, and is willing to put up with learning which control characters do what. This user likes some macro capability—for example, to display each changed item during a global replacement—but is unwilling to spend five minutes working out a complex macro that might save a minute or so of execution time. He would rather perform each function separately, right on the screen where he can see what he is doing. "Keep it simple and get it right" is his working principle.

Most of us are a mixture of these three types, with one that is dominant. In me, No 3 is the dominant type, perhaps because much of my technical writing has been about hardware, and word processing has had to conform to a limited budget. But because I have done layout work, I understand the typographer's contempt, and am delighted when an idealist who is also a neat programmer figures out how to give us another bell or whistle, or how to make the machine work better with us, while still keeping the program manageable in size and the operation easy. I think Mark of the Unicorn must have a few of these around— and also a few secretarial users to test out the result and keep human (and especially non-technical human) needs in front of the idealists.

Conclusions

Mince is my kind of editor. It won't appeal to the typographer, because it is only an editor, with very little formatting capability. It probably won't appeal to the idealist, because it has no macro capability at all and uses control characters and letters as commands. But it is a well-designed, workmanlike job, and has some very useful features that no other editor can yet provide.

My personal reservations about Mince relate to the command structure. I have been using the original Word Master for several years, and love it. All the Word Master on-screen operations are single control characters, and I have brought out the most commonly used operations (cursor left, right, up down, Insert on/off, delete character, delete line, etc) to function keys which generate the appropriate ASCII code. I am reluctant to forego this convenience, or to modify my function key hardware extensively to handle a new set of codes and generate the Mince 2-character meta-commands.

If you are about to purchase your first screen editor, note that Mince provides facilities (Transpose, Copy Region, Capitalize, Page Mode, Mark a region, variable tab points, and others) that Word Master does not have, or does less elegantly.

Other editors each have some of these features, but I know of no other that has all of them as Mince does. On the other hand, Word Master does some things (such as yanking a disk file for inclusion in the current text, or overwriting existing text) a little more simply than Mince, and its command mode has the very powerful macro capability of ED.

If you must have a package that formats your page for you right on the screen, neither Mince nor Word Master is for you; in that case you need Word Star or Magic Wand, provided that you have at least a 48K system (64K if the formatting is not to be abominably slow). But if you like your editor and formatter to be separate programs (as I do), then look closely at both Mince and Word Master (which is equivalent to the editor portion of Word Star). Even if you have some other editor that you are happy with, look closely at Mince anyway, and consider the tradeoffs — it may be time for you to switch, because Mince does more for you than any other CP/M editor that I have seen so far.

If you want a formatter, too, SCRIBBLE may be just the job — what I have heard of it makes it an easy winner over almost anything except the TSC formatter, which for the power and flexibility knocks spots off TEX, TEXTWRITER, and in some respects even Word Star. I hope to be reviewing SCRIBBLE in a future article. ■

Running North Star Basic With CP/M

by Steve Leibson

A review of three software packages which permit the powerful North Star Basic to run with CP/M

Many languages are available to run on the CP/M operating system. There are a large number of Basics, Pascals, Fortrans, Cobols and Forths, to name a few. One language which is popular in spite of its CP/M incompatibility is North Star Basic. There are now three software packages which end this incompatibility.

Since North Star has its own disk operating system which it supplies to every purchaser of North Star computers, there was little hope that CP/M compatibility was forthcoming. Yet North Star Basic is such a powerful Basic interpreter that we all lost on not having the ability to run the two software packages together.

North Star DOS is not as powerful as CP/M because it does not have dynamic file allocation. North Star users are all too familiar with taking care not to overflow disk files, and periodically repacking disks to reclaim space from deleted files.

Advantage of North Star Basic

Why does CP/M need another Basic? First there was E-Basic followed by C-Basic which became the "standard" CP/M Basic. Then when Microsoft could market their Basic for computers other than the MITS Altair, they introduced MBasic which is now available in both interpreter and compiler versions.

With all these Basics, plus others I haven't named, why adapt North Star Basic to CP/M? There are three reasons: speed, accuracy and power. In every benchmark, North Star Basic is at the top of the heap. It is a fast interpreter made faster by a hardware floating-point board available from North Star. To my knowledge, this is the only microcomputer Basic interpreter which has been mated to a hardware floating-point facility.

Steve Leibson, 4040 Greenbriar Blvd., Boulder, CO 80303.

Along with its blazing speed, North Star Basic has accuracy. Computations are done in Binary Coded Decimal (BCD) instead of binary as with the other Basics. Binary routines must be written to first convert a number to binary, perform the computation, and then convert the binary result back into decimal if the answer is to be printed. Binary computation can be faster than BCD computation, but not necessarily so.

BCD format encodes each decimal digit into a four-bit binary number. The math routines are written to calculate in decimal. Thus there is no round-off or truncation error in the decimal-to-binary and binary-to-decimal routines since these routines do not exist.

Some binary-math Basics show computation error easily. To see if your system has such error, try the following short program.

```
10   FOR I=0 to 1 STEP .1
20   PRINT I
30   NEXT I
40   END
```

If you don't get the printout you expect, chances are good that your Basic performs math in binary. The number .1 cannot be exactly represented in binary, just as the number 1/3 can only be approximated by the decimal representation .3333.

North Star Basic also has power. Computationally, the most powerful feature of this language is the multi-line, user-definable function. These functions accept several arguments to be passed in the function call. While in the function itself, only local variables are used. This feature isolates complex subroutines from main programs by creating whole new sets of variables while leaving mainline variables alone.

String variables are another powerful feature of North Star Basic. Gone are RIGHT$, LEFT$ and MID$. In their place, the string variable name is used with the starting character position and the ending character position specified. For example, if A$ was equal to the string "ABCDE" then A$(3,3) is "C" and A$(2,5) is "BCDE." There are string functions built into the language to concatenate strings, to transform numbers into strings and back again, and a very useful pair of functions which allow numbers representing ASCII characters to be transformed into string characters, and vice versa.

The Conversion Packages

There are now three ways to adapt North Star Basic to the CP/M operating system. The trick is to accept console, printer and disk calls from North Star Basic and convert these to the protocols required by CP/M.

NSCPM48 From SIG/M

The SIG/M group is a CP/M users group in New Jersey (P.O. Box 97, Iselin, NJ 08830) that offers low cost, public domain software. Currently this group is offering fifty-eight volumes. Volume number 10 contains the North Star Basic to CP/M conversion program NSCPM48, and it costs about $10.

This program is designed to adapt release 4 of North StarBasic to CP/M versions 1.4 and 2. It will also work with CDOS V1.07 with four patches made to the CDOS. NSCPM48 requires that your copy of release 4 Basic be relocated to 800 hex using a North Star Users Group program.

The NS48CPM occupies the memory space between 100 hex and 800 hex. It allows space for up to ten open disk files. North Star Basic makes some assumptions on the amount of record keeping that the DOS is supposed to do. All of the interface programs have to allocate space for this record keeping. Disk files have only a single character extent, as opposed to normal CP/M files which may have up to three characters. Thus FILE1.BAS and FILE1.BAK are equivalent.

The I/O devices are configured as:

CP/M Device	North Star Device Number
Console	0
List	1
Punch	2

Though the SIG/M program will adapt North Star Basic to CP/M, there are two reasons why you may wish to buy one of the other two software packages. First, Release 4 Basic is not the latest version. Release 5.2 is North Star's relocatable Basic with some of the earlier bugs removed. Second, both of the other packages allow conversions between ASCII text files and Basic program files.

North Star Basic program files are tokenized. Thus, a keyword such as PRINT is represented by a single "token" byte. Each keyword is converted to a token as it is typed in. Tokenization is one of the keys to North Star Basic speed. Unfortunately, text editors and word processors cannot create token files easily, and so the North Star editor has to be used if the SIG/M interface is selected.

NS Basic Interface From InfoSoft

This software package is designed to link North Star Basic Release 5.2 to CP/M 1.4 or 2.X, InfoSoft's I/OS operating system versions 1 or 2, CDOS versions 1 or 2, or TSA/OS. The North Star Basic must be Release 5.2, and only the eight-digit precision, non-hardware, floating-point Basic can be used.

The cost of the package (available from InfoSoft Systems, 25 Sylvan Rd. South, Westport, CT 06880) is $100 plus $8 for shipping and handling. A combination of the NS Basic Interface plus the InfoSoft I/OS operating system configured for North Star Horizon I/O is $140. The version of I/OS included in this price is equivalent to CP/M 1.4.

Three other programs are provided. NSDD will move files from North Star DOS compatible disks to CP/M disks. You must have two disk drives for this program. NSLIST and NSENTER are the programs which allow interchange between North Star Basic files and ASCII text files. NSLIST converts programs to ASCII and NSENTER reverses the process.

The conversion process is well documented and fairly simple. The steps are:

1. Move the North Star Basic file to a CP/M disk using NSDD. The CP/M file has to be named NBASIC5.NDT. Also transfer the files NS5CNVT. COM and NS5PAT.DAT using PIP.
2. Run NS5CNVT. This creates a file called NS5BAS.COM which is the execute file for the converted North Star Basic. This system requires at least 32K of RAM.

That is all there is to the conversion.

The NS Basic Interface uses two special extents for North Star related files. Program files use the extent .NBP for North Star Basic Program and data files use the extent .NDT for North Star DaTa.

NSLIST, the program-to-text file converter creates files with the extent .BAS from files with the .NBP extent. Other versions of NSLIST are available for Release 3 North Star Basic and Poly 88 Basic files.

The NSENTER program takes text files with .BAS extents and converts them into .NBP files. Other versions of NSENTER are available for Release 3 and 4 North Star Basics and Poly 88 Basic.

Features Of InfoSoft's Interface

The hybrid of North Star Basic and the NS Basic Interface supports several new features. The Basic statement MEMSET loses its significance because program memory will automatically be sized to the maximum allowable space. This is between Basic and CP/M's BDOS. The CP/M control-P printer toggle and control-S screen control characters are supported while in Basic. Typing BYE <CR> will return you to CP/M.

The peripheral devices in North Star Basic have been assigned to CP/M devices as follows:

CP/M Device	North Star Device Number
Console	0
List	1
Reader(in)/Punch(out)	2

Disk number extents which are 1,2,3 and 4 in North Star Basic are automatically converted to A,B,C, and D. Eight open disk files are supported. NSAVE, the North Star command to create and then save a new file, has been eliminated. LOAD and SAVE are all that are required for programs. This is due to the dynamic disk allocation of CP/M.

One new quirk of disk operation is that CP/M disks have to be logged in. This is the control-C operation all CP/M users are familiar with. A new command which has replaced NSAVE in the Basic command table is RESET. This command will perform the log-in of the disk but not from the program. InfoSoft has added a special CALL to log in the disk from the running program. This is discussed in detail later in the article.

CREATE also has changed slightly. In North Star Basic, a file size is specified. Dynamic file allocation makes this argument in the CREATE statement extraneous, but a dummy length may still be specified.

Since CP/M has dynamic file allocation, this has an impact on data files which are to be randomly accessed. The file must be fully allocated by sequential writes before it is randomly accessed, because CP/M creates files with no blocks allocated. Refer to Listing 1 for InfoSoft's program that will perform such an allocation.

The OPEN statement is used to associate a number with a file name. In the standard North Star operating system, any type of file; data, program or whatever, could be OPENed. With the NS Basic Interface, only data files (extent = .NDT) may be OPENed. Attempts to open other types of files will produce "non-existent file" errors.

If this presents a problem, simply change the file name extent to .NDT while in CP/M, run your Basic program, exit Basic and change the extent back to what it was. This is admittedly a clumsy method of programming, but at least the problem has a solution.

Two functions added to North Star Release 5.2 Basic were FILESIZE and FILEPTR. The FILESIZE function no longer works due to the dynamic file sizes in CP/M. The function will always return 65535. The FILEPTR function still returns the location of the file pointer for the specified file.

A new function InfoSoft has added to the Basic is the "^" function. This function returns the absolute memory location of the specified variable. This can be especially useful for passing information to user-created assembly language routines. Normally, only one or two bytes can be passed to a machine-language routine. With the "^" function floating-point numbers, arrays and strings can also be passed, by address.

One peculiarity of moving North Star Basic to CP/M is that operation of the Basic is no longer tied to North Star hardware except by North Star's licensing agreement. Since some users will be sure to try North Star programs on other hardware, there is one thing to keep in mind.

The random number generator RND in Basic is tied to the North Star Disk Controller. RND(-1) causes the random number seed to be regenerated based on the time to the next sector pulse received by the disk controller. If the system has no North Star Disk controller, this function call will cause the program to hang indefinitely.

Finally, InfoSoft has made major changes to the machine-language call facility in Basic, the CALL function. The standard function was:

```
CALL (<absolute address>,<optional passed
     parameter>)
```

InfoSoft considered this function too much of an unsheathed knife and decided to make it "safer." The new syntax is:

```
CALL (<Subroutine #>,<Parameter 1>,<Parameter 2>)
```

The parameters are optional. There are six standard subroutines supplied by InfoSoft. If the subroutine number specified is greater than 128, the call will be made as in the original North Star CALL, but only if "extended calls" are enabled.

The new standard calls are:

Subroutine 1: Returns a character from console or 0 in none has been typed. Examples:

```
Z=CALL(1)      Get a character from the console.
Z=CALL(1,0)    Get a character from the console.
Z=CALL(1,1)    Read the console status.  If Z=0,
               console not ready.  If Z=255,
               console is ready with a character.
```

The only problem with this subroutine is that at the end of execution of each line of North Star Basic, the interpreter checks to see if a control-C has been typed to halt the program. If a character was typed but isn't a control-C, the character is discarded. This control-C check can't be disabled. Thus the only way to take full advantage of a CALL to subroutine 1 is to use multiple statements on one line. Then keyboard inputs can be processed exclusively by the subroutine.

Subroutine 2: Input a character from the reader. Example:

```
C=CALL(2)
```

A full eight-bit value is returned. Subroutine 2 will wait until a character is obtained from whatever device is configured as the reader.

Subroutine 3: Output a character to the punch. Example:

```
Z=CALL(3,C)
```

The second parameter in the CALL specifies a byte to be output to the device configured as the punch.

Subroutine 4: Output a character to the LST device. Example:

```
Z=CALL(4,C)
```

The second parameter in the CALL specifies a byte to be output to the list device. In order to print a string, you have to use the program lines:

```
FOR I=1 TO LEN(A$)
Z=CALL(3,ASC(A$(I,I)))
NEXT I
```

Subroutine 5: Log-in new disk. Example:

```
Z=CALL(5)      Log-in current disk
Z=CALL(5,X)    Log-in disk drive X
```

Listing 1.

```
  10 REM THIS PROGRAM WILL FULLY ALLOCATE A NS BASIC DATA FILE
  20 REM FOR PROGRAMS THAT NEED PRE-ALLOCATED FILES
  30 REM INITIALIZE BUFFER
  40 DIM A$(254)
  50 FOR I=1 TO 254\A$(I,I)=CHR$(0)\NEXT I
  60 REM
  70 INPUT "FILE ",F$
  80 DESTROY F$
  90 CREATE F$\REM NOTE NO DUMMY LENGTH
 100 REM
 110 OPEN #0,F$
 120 INPUT "NUMBER OF RECORDS: ",X
 130 FOR I=1 TO X
 140 WRITE #0,A$
 150 NEXT I
 160 CLOSE #0
```

Listing 2.

```
1000 REM *****************************************************
1010 REM *                                                   *
1020 REM * BASIC PROGRAM LINE NUMBER GENERATOR FOR NORTH STAR *
1030 REM * ON CP/M PROGRAMS CREATED WITH TEXT EDITORS OR WORD *
1040 REM * PROCESSORS.  THIS PROGRAM REPLACES THE AUTOMATIC   *
1050 REM * LINE NUMBER GENERATION OF THE NORTH STAR BASIC     *
1060 REM * EDITOR                                             *
1070 REM *                                                   *
1080 REM *   STEVE LEIBSON     SEPTEMBER 7, 1981              *
1090 REM *                                                   *
1100 REM *****************************************************
1110 REM
1120 REM FIRST CLEAR THE SCREEN
1130 C$=CHR$(27)+"+"\REM CLEAR SCREEN STRING
1140 PRINT C$,
1150 REM
1160 REM NOW, GET BEGINNING AND ENDING LINE NUMBERS AND STEP SIZE
1170 REM
1180 INPUT "WHAT LINE NUMBER DO YOU WISH TO START WITH? ",B
1190 PRINT
1200 INPUT "WHAT LINE NUMBER DO YOU WISH TO END WITH?",E
1210 PRINT
1220 IF E>=B THEN 1250
1230 PRINT "QUIT FOOLING AROUND PLEASE!"
1240 GOTO 1180
1250 INPUT "WHAT LINE NUMBER SPACING DO YOU WANT? ",S
1260 B=INT(B)
1270 E=INT(E)
1280 S=INT(S)
1290 REM
1300 REM NOW ASK FOR THE FILE TO PUT THE LINE NUMBERS IN
1310 REM
1320 PRINT
1330 INPUT "WHAT IS THE NAME OF THE FILE TO PLACE THE LINE NUMBERS
     IN? ",F$
1340 PRINT
1350 IF LEN(F$)>0 AND LEN(F$)<7 THEN 1390
1360 PRINT "FILE NAMES MUST BE LESS THAN OR EQUAL TO 6 CHARACTERS
     IN LENGTH"
1370 PRINT
1380 GOTO 1330
1390 REM
1400 REM CREATE AND OPEN THE DESIGNATED FILE
1410 REM
1420 OPEN #1, F$+".D"
1430 REM
1440 REM FILL IT WITH LINE NUMBERS
1450 REM
1460 FOR I=B TO E STEP S
1470 A$=STR$(I)
1480 FOR J=1 TO LEN(A$)
1490 WRITE #1,&(ASC(A$(J,J)))
1500 NEXT J
1510 WRITE #1,&32,&13,&10,NOENDMARK
1520 NEXT I
1530 END
```

This subroutine can be used as a programmable RESET command.

Subroutine 6: Terminal Controls. This subroutine does not work on CP/M. The operating system must know the escape and control sequences of your terminal and CP/M doesn't. I/OS, CDOS, TSA/OS or SDOS do, and this subroutine works with those operating systems.

```
Z=CALL(6,R,C)
```

Personally, I don't care for the modifications to the CALL function. They are not compatible with previous North Star programs and do not seem to materially add to the language usefulness. InfoSoft does give a simple program to enable normal operation of this function. It is:

```
X=CALL(0)+2

FOR I=X TO X+2\FILL I,0\NEXT I
```

This uses the undocumented subroutine 0 to return the address of the subroutine table which is then zeroed out.

Performance Of The InfoSoft NS Basic Interface

The interface between CP/M and North Star Basic seems to have been properly made. Programs SAVE and LOAD correctly, data files are accessed properly and the console and printer interfaces work. It felt odd not to have to worry about whether the North Star Basic program would fit in the file or not. Yet the old comfortable feel of North Star Basic was there.

I think I would be more excited about the InfoSoft product if I had one of the other operating systems, instead of CP/M. To finally have terminal independence would go a long way to easing operator interface in business programming. I do not like the changes InfoSoft has made to the CALL statement for just that reason. Since I do not have an operating system which allows the extended CALLs to work, they are merely in the way. A statement which I find useful has a slightly different syntax to accomodate features I can't use.

Matchmaker II From The SoHo Group

The second commercial product available which converts North Star Basic to operate on the CP/M operating system is called Matchmaker II. It is from The SoHo Group (140 Thompson St., Suite 4-B, New York, NY 10012), and costs $109.95 postpaid. The SoHo Group will sell you North Star Basic, version 5.2 already on a CP/M-readable disk for an additional $24.95 (North Star's retail price).

Like InfoSoft's NS Basic Interface, Matchmaker II is intended to convert North Star's version 5.2 Basic, with the origin at 0E00 hex. Unlike InfoSoft's product, versions of Matchmaker II are available which will convert any precision North Star Basic in either standard or floating-point variants.

The only other program supplied with Matchmaker II is a SUBMIT file. The function of InfoSoft's NSDD, bringing files over from North Star's DOS, is handled in a simple manner. For files less than 31 blocks, North Star DOS is used to load the file starting at location 100. Then CP/M is loaded and the file is stored on disk with the CP/M SAVE command.

A considerably more complex process is required to move larger files over. First you must create a version of North Star DOS that runs in high memory. Since version 5.2 of North Star DOS is relocatable, this is not a great problem. Then CP/M is booted, immediately followed by a DOS boot. DOS is then used to load the desired file in starting at 100 hex. Finally, a JP 0 DOS command issued after inserting the CP/M disk in drive A: will warm-boot the already loaded CP/M, and the file may be saved.

This technique should allow you to load in any Basic program you can run in your system but will not help for large data files. Those will have to be loaded in segments and recombined with CP/M's PIP.

The functions of InfoSoft's other two programs, NSLIST and NSENTER are built into the Basic by Matchmaker II. The ability to cross between North Star Basic tokenized format and text file format is a blessing if you have a good word processor such as WordStar, or if you are adept at using ED.

If you are using CP/M 2.2, installation is simple. First make a copy of the disk with The SoHo Group's software and add DDT, SUBMIT and XSUB. Then type "SUBMIT SOHO". When the disk stops spinning, the conversion is complete.

Owners of CP/M 1.4 do not have XSUB and therefore must manually do the conversion. This is made easy by the documentation. The procedure is to print out the SUBMIT file SOHO.SUB, strike out the line XSUB and type in the remaining commands. Since the SOHO.SUB file is used to perform the conversion for CP/M 2.2, the only chance for error is in your typing. A little care in typing, and the conversion is complete.

One program that The SoHo Group claims is included is called REE.COM. This is supposed to restart Basic without reloading it. It wasn't on the disk I received, but thanks to several articles in past issues of Microsystems, creation of this file is simple. You simply type:

```
SAVE 0 REE.COM
```

Features Of The SoHo Group Matchmaker II

Just as with the InfoSoft product, Matchmaker II has added extents to file names. The difference is that files under Matchmaker II have single letter extents and you may pick the letter for data files. For example, a data file under the NS Basic Interface must have the extent .NDT while Matchmaker II suggests using .D (though you may choose another letter).

Likewise, Basic programs have .NBP extents under the NS Basic Interface, while programs under Matchmaker II must have an extent of .B. Thus, even though Matchmaker II extents are limited to a single letter, there is better flexibility than InfoSoft's three-letter extents. However, flexibility is lost because filenames under Matchmaker II may only be six characters long. The remaining two characters normally found in a North Star file name have been given to the period and single letter in the extent thus forming an eight-character filename.

Disk drive specifiers remain as in North Star's DOS. The file name is followed by a comma and a drive number such as FILE.D,2. This will save a lot of program rewriting.

CREATE, DESTROY and NSAVE have been eliminated from the command set. Any use of these commands will result in a "SYNTAX ERROR" message. Instead, any reference to a file using SAVE will automatically create the necessary file and directory entry. A side effect of this approach is that any attempt to LOAD a program which does not exist will result in the creation of that file instead of an error.

MEMSET is still functional but unnecessary. Matchmaker II will automatically set memory size for your system, assuming your CP/M is correctly configured and all pointers are in their proper places.

In addition, FILESIZE will always return 8192 since the concept of absolute file size is a little strange in a dynamic file allocation system such as CP/M. FILEPTR will work however, so you will not need to keep track of file pointers yourself. Also, the function FILE, which is supposed to return the file type will return type 2 (Basic program) if the file extent is .B and type 3 (data file) otherwise.

Peripheral device assignments are somewhat inflexible. North Star device 0 has been assigned to the CP/M Console and devices 1 through 6 are assigned to the LST: device. The SoHo Group suggests that you "may manipulate the IOBYTE" if you have more than two peripherals. As with most software, the I/O is primitive and mostly unimplemented. This is not the fault of the SoHo Group. With as many configurations as there are users, it is not possible to build a "universal interface routine." I would prefer to see better documentation on the I/O however, such as how one "may manipulate the IOBYTE."

North Star peripheral device number seven is reserved for the ASCSAV and ALOAD features which Matchmaker II adds to the North Star Basic command set. ASCSAV will take a memory-resident program and save it as a text file with an extent of .SOH. You may only save to the currently logged CP/M drive. After saving a program in this manner, you may edit it with any text editor or word processor.

The process is reversed with the ALOAD command. Again, only the currently logged CP/M drive may be accessed. ALOAD will also merge two programs by loading or ALOADing one program and then ALOADing the second on top of the first. There are some "funnies" in this process which The SoHo Group details in the manual. These stem from the use of the keyboard buffer for the two commands. It is best if you don't play with the keyboard while disk accesses are taking place during the use of ASCSAV or ALOAD.

One limitation which Matchmaker II has is the ability to only have five files open at one time while North Star Basic may have eight open at once. This is purely the result of a decision on The SoHo Group's part as to how much space to allocate for file buffers. In my applications, I never open more than three files at once, but this limitation may greatly impact your programs if you open more than five files at once.

In conjunction with this limitation, the file count has to be manually reset to allow more files to be OPENed. After closing all files, the magic statement:

```
FILL 299,0
```

will clear the file buffers and allow another five files to be opened. File buffers are also cleared automatically when a program stops running and Basic returns to the READY mode.

Matchmaker II creates a Basic totally independent of North Star hardware. An unmentioned exception to this is that, as with the NS Basic Interface, any use of RND(-1) will hang your system up until you reset it for the same reason mentioned for the InfoSoft product.

Performance Of Matchmaker II

I was able to thoroughly test the disk capabilities of Matchmaker II, on itself. After configuring North Star Basic with Matchmaker II, you will see your name in the log-on message when Basic is first activated. Unfortunately, The SoHo Group got my name wrong. I am used to the routine reversal of "e" and "i" in my name by humans, but I won't stand for it from my own computer!

All that was required was to write a simple Basic program which sequentially reads each byte of the BASIC.COM file (Renamed BASIC.C to conform to Matchmaker II syntax) until my mispelled name was found. This tested sequential byte reads. Then, using a random byte write, I corrected my name. The Matchmaker II worked flawlessly.

I was most eager to test out the ASCSAV and ALOAD features because I have WordStar which has proved itself to be quite a program editor for assembly language programs. Without thoroughly reading the manual, I was able to create a file called FTEST.BA.SOH by ASCSAVing a file called FTEST.BA. This file could not be erased by the CP/M ERA command except by ERAsing all files with .SOH extents. Moral: Read the manual.

Once I did get ASCSAV and ALOAD working, a problem became apparent. I had lost one of the best features of the North Star editor, automatic line numbering. This could not be tolerated. If changing North Star Basic over to CP/M meant losing features, something had to be done. My solution is shown in Listing 2, a program I call NUMBER.B.

The NUMBER.B program creates a text file of line numbers. WordStar or another editor can then be used to add program statements to the line numbers to create the program. If you run out of line numbers, simply exit the editor, saving the program created thus far, run NUMBER.B to create a file with more line numbers, re-enter the editor and append the new line number text file to your program file.

Listing 2 also exhibits another advantage of ASCSAV and ALOAD. The program was created with the North Star editor (old habits are hard to change). Then, ASCSAV was used to create a text file of the program which was then merged with this article.

Overall Impressions And Recommendations

I am delighted to have North Star Basic running on CP/M. I still feel it is the best Basic available for microcomputers. The conversions don't seem to affect performance except that CP/M is constantly performing directory reads that North Star DOS didn't have to do. That is a function of the operating system not the converter routines.

If you just want North Star Basic on CP/M, already have release 4 North Star Basic (you can't but it any more), don't mind a bit of work in converting Basic and don't want to use a text editor or word processor to edit programs, the SIG/M software may be for you. Low cost is its major feature.

Both the NS Basic Interface and Matchmaker II are good products, yet both are flawed though in different ways. The products cost about the same.

The NS Basic Interface has limited file extent capability, a modified syntax for the CALL statement, only works with 8-digit, non-floating point Basic, uses separate programs for saving and loading text files and is somewhat more complex to install. Advantages include no extra limitation on the number of open disk files, the "∧" function and a slightly better I/O implementation. If you have one of the operating systems supported by the extended CALL of the NS Basic interface, this may sway you. Remember however that this can no longer be called "North Star Basic."

Matchmaker II's chief limitations are the five-file limit on open disk files and the six-character limitation on filenames. Advantages include the ASCSAV and ALOAD commands built into the Basic, the ability to convert any variant of North Star Basic version 5.2 and simple installation.

My preference is for The SoHo Group's Matchmaker II. The five disk file limit doesn't affect me and there are no syntax modifications to worry about. I wrote to both The SoHo group and InfoSoft and both sent speedy replies to my questions. In the end, your choice will depend on which set of limitations bothers you the least.

STEVE LEIBSON is an electrical engineer designing graphics workstations for Auto-trol Technology Corporation. He specializes in interfacing to microcomputers. Steve has written "Leibson on I/O," a pamphlet published by Hewlett-Packard and self-published a book, "The Great Small Business Computer Ripoff." Steve is also editor of "Interrupt," the newsletter of the Denver Amateur Computer Society.

Hayden's 8086 Primer

by Jeff Duntemann

The 8086 Primer, by Stephen P. Morse. Hayden Book Co., Rochelle Park, NJ, softcover, 197 pp. plus index, $8.95. 1980.

In the computer book business, it is all too easy for an author to look at a new language or processor and describe it without explaining it. Most of us can read manufacturer's specs, but what we want is insight into why a piece of software or hardware was created and how it can best be used. *The 8086 Primer* is built on such insight, which is not surprising since Morse was the man perhaps most singly responsible for designing the 8086 architecture.

After the obligatory "Introduction to Computers" section (registers, hexadecimal notation, ASCII codes) Morse explains how and why the 8086 architecture does what it does. Here the book is at its best; the author points out the deficiencies in the 8080 and explains the 8086 organization in terms of the evolution of the 8080 family. Done this way, the 8086 begins to make sense in a way that helps cement the reader's understanding of machine specifics in later chapters. The tone of the book is enthusiastic and informal. I was surprised and thankful for that; the tone of microprocessor texts in general (and the Osborne line in particular) is notoriously dry.

Only after establishing this rational overview does Morse take on the instruction set, opcode by opcode. Here, too, he explains *why* every instruction was included in terms of a recognized need, rather than simply describing what each instruction does. Furthermore, his willingness to use informal metaphor is striking, and very effective:

> "There may be any number of reasons (let's say 256) for an interrupt on the INTR pin, while there is only one reason (impending doom) for an NMI interrupt."

This slightly offbeat way of presenting the dry facts of microcomputer logic occurs again and again through the book.

Along the way Morse explains how the 8086 instruction set lends itself toward use in single-stepping, debugging, multiprocessing, and arithmetic-intensive applications. Coverage of the 8086 interrupt handling logic is particularly good; this subject is poorly covered elsewhere, even for the venerable 8080.

The single chapter on hardware interface is brief and goes for breadth rather than depth. Morse goes into hardware only enough to show the prospective programmer how different hardware configurations can affect the design of the system software. Hardware design deserves more time and detail than Morse can afford here; this is, after all, a primer.

The final two chapters are introductions to ASM-86, an Intel macroassembler, and PL/M-86, Intel's higher level language with roots in PL/I. On a detail level these chapters are of less use than the rest of the book, since few of us will ever own the actual Intel software. However, the insight here into what a macroassembler of high-level language actually must do to turn English into efficient binary machine code is worth the space taken. Morse makes it fairly plain that the 8086 was created with PL/M foremost in mind, and the idea of designing a microprocessor "backwards" from a language spec was fascinating. Even if the reader never uses PL/M or 8086 assembler, as general introductions to "thinking in a language" these chapters are excellent.

Aside from a (relieved) feeling that the 8086 is approachable by a hobbyist with some 8080 experience, two lasting impressions come from *The 8086 Primer*. First, the chip's architecture was designed to make implementation of high level languages efficient, *not* to make assembly language programming any easier. The complexities of the instruction set will, if anything, make mastering an 8086 assembler difficult, and hand-assembly of 8086 code virtually impossible. Second, this book and the architecture of the 8086 were labors of love on Stephen Morse's part. His enthusiasm for the chip was infectious, and helped the whole picture of the 8086 gel in my mind.

The 8086 Primer is an excellent preface to the CP/M-86 documentation. It is a great deal easier on the reader than the slightly older *The 8086 Book* by Rector and Alexy. Better books than both will probably appear in time, but for now, there is no better place to start than here. ■

Jeff Duntemann, 301 Susquehanna Road, Rochester, NY 14618.

ZDM: A Bargain Z-80 Debugging Program

by Andrew L. Bender

A debugging program is a tool used to diagnose troubles with programs in development as well as during checkout. This review concerns itself with a Z-80 debugger called ZDM or ZDMZ. ZDM is used to debug programs where the programmer is using the "TDL/CDL" mnemonics. ZDMZ is for those using the Zilog mnemonics. For the purposes of editorial clarity I will use the designation ZDM, and if you are inclined towards the Zilog mnemonics substitute ZDMZ in your mind.

ZDM is supplied in several formats for the standard CP/M system as well as North Star, Micropolis and Apple CP/M systems. I tested the standard CP/M version on my Z-80 system. ZDM will not execute on an 8080 system. The package is supplied by RD Software, 1290 Monument Street, Pacific Palisades, CA 90272. The price is $45 which, in this reviewer's opinion, is a bargain in these days of $500 compilers.

The documentation supplied with ZDM consists of short, unambiguous descriptions of each command contained in a ten page booklet. The booklet is punched to fit a three-ring notebook—I put mine in my CP/M notebook. You can't learn how to debug a program with this book any more than you can with Digital Research's DDT manual. To learn to debug programs you have to write them, lots of them, then you can learn to fix them. No amount of reading can substitute for the experience gained from hours of staring at listings, register dumps and memory dumps.

ZDM parallels DDT in most commands, making it very easy to learn and use. Just pick up your ZDM manual for the new commands. After you have read them a few times you will remember them and won't need to refer to the manual. First I will tell you the thing DDT has that ZDM doesn't have: The "A" command (assemble symbolic code) which was never of much value to me personally. I will shed no tears, and I am certain that most will agree that it isn't an essential command. Next, ZDM has only one breakpoint. This is unfortunate because two breakpoints are nice when a program reaches a branch point and you don't know which way it will go. Two breakpoints let you cover both possibilities and give you comfort in the knowledge that the debugger will always get control again.

There are things ZDM has that DDT doesn't: It can disable and enable interrupts—valuable if you are trying to debug a program which is running with interrupts enabled, such as a real-time clock program or interrupt driven input/output routines. It can read or write to or from any of the machine input/output ports. You can also examine the alternate Z-80 register set. It is smaller than DDT by one page and it overlays the CP/M CCP as does DDT. If the mnemonic portion of ZDM is disrupted by a program which overlays it, ZDM print out opcodes during trace, list and examine in hexadecimal rather than mnemonics. If the nucleus of ZDM is overlaid during loading of a program with ZDM, then an error message is printed. DDT gives no indication what went wrong, although those used to using it always know why it failed. That isn't nice behavior for a program which is supposed to be gentle and forgiving.

In summary, ZDM is inexpensive and useful in debugging Z-80 programs which execute under CP/M. It has a command structure similar to DDT, making it easy to learn in a few minutes by those familiar with DDT. It lacks the double breakpoint capability of DDT, but has other valuable commands which make the debugging of input-output routines easier. ■

Andrew Bender, 614 Victoria Terrace, Paramus, NJ 07652

ANDREW L. BENDER is a practicing physician specializing in neurological diseases. He has over twenty years experience in assembly language programming. Recently, he has been designing a database for seizures and epilepsy. He likes kids, cats and Bach but not necessarily in that order.

SOFTWARE DIRECTORY

Program Name: Footnote and Pair
Hardware System: 48K RAM, 8080/Z80, WordStar, CP/M
Minimum Memory Size: 48K RAM
Language: Machine Language
Description: Footnote automatically numbers footnote calls and footnotes, and formats the text, placing footnotes at the bottom of the correct page. Footnotes can be entered singly or in groups in the middle or at the end of paragraphs, in a completely separate note file. After running Footnote, the user can re-number and re-format the Wordstar file. Pair checks that printer commands to Underline or set in Boldface are properly terminated.
Release: December 1981
Price: $125.00, $15.00 for manual alone
Included with price: Disk and User's Manual.
Where to purchase it:
Digital Marketing
2670 Cherry Lane
Walnut Creek, CA 94596
(415)938-2880

Program Name: NANSII
Hardware: CP/M
Minimum Memory Size: 56K
Language: Assembler
Description: NANSII is a data base system using English word commands. System allows up to record lengths up to 2048 characters, up to 100 fields per record, and up to 32000 records per file. Math function allows up to 100 math operations on records. Search function allows record retrieval by up to 100 logical selection criteria. Sort on up to eight fields at a time in ascending or descending order. Sort routine requires no extra disk space. NANSII can be used with one or more floppy or hard disk drives. 200 page instruction manual includes step by step example.
Release: December 1981
Price: $249.95, Demo is $29.95.
Included with price: NANSII program and manual.
Where to purchase it:
XPS, Inc.
323 York Road
Carlisle, PA 17013
1-800-233-7512/ PA (717)243-5373

Program Name: "C" Compiler
Hardware System: CP/M based incl. Altos, Superbrain,TRS-80 MOD II
Minimum Memory Size: 48K
Language: "C"
Description: A three pass true C compiler that directly produces 8080 object code in relocatable format. Symbolic debug displays global variables, functions and source lines. Accepts a sub-set of "C" language as defined in *The C Programming Language* by Kernighan and Richie. Full extended run-time library including "PRINT," I/O Re-Direct, Random Files, Quick Sort and B-Tree Access.
Release: January 1981
Price: $50.00 end user. Requires I/SAL. Source $250.00. Call for OEM and dealer pricing.
Included with price: User manual and three month subscriber service for updates.
Where to purchase it:
InfoSoft Systems Inc.
25 Sylvan Road South
Westport, CT 06880
(203)226-8937

Program Name: Phonedex i
Hardware System: 8" disk based CP/M, 64 x 16 or larger CRT, Hayes Smartmodem
Minimum Memory Size: 48K
Language: .COM file; no language required
Description: Name/address/phone list manager with autodial/data communuications capabilities. Maintains up to 1200 records on a SD 8" disk. Records can be sorted, searched, or extracted on any of eleven fields including second address line and two user-definable fields. Phonedex prints mailing labels up to four across with up to 50 repeats of each label. There is also a "little black book" hardcopy format which prints records to memo-book sized pages. Dials through the Smartmodem with either pulse or tone dialing. Terminal mode can be entered for communicating with time-sharing systems and computer bulletin boards.
Release: January 1982
Price: $49.95; User guide $15.00.
Included with price: Object files, user guide, and several example data files.
Where to purchase it:
Starside Engineering
PO Box 8306
Rochester, NY 14618
(716)473-2986

Program Name: TRANSFER
Hardware System: 16K
Language: 8080 Assembler
Description: Utility to link one CP/M system to another also using TRANSFER. Allows file transfers at full data speed (no conversion to hex), with CRC block control check for

very reliable error detection, and inter-active retry. Full wildcard capability to send *.*. Transfer at 9600 baud with wire, 300 baud with phone connection. Both ends need TRANSFER. Includes 8080 Source Code.
Release: January 1981
Price: $179.00
Included with price: Source code on 8" SS/SD disk, and a 25 page manual with program flow charts.
Where to purchase it:
Starr Computer Systems Inc.
6126 Melissa Lane
Omaha, NB 68125
(402)571-1722

Program Name: OMEGA
Hardware System: 8080/Z80 disassembler CP/M system with at least one disk drive
Minimum Memory Size: 50K
Language: Machine Language
Description: Assembler utilizing external mnemonic tables. XITIAN/TDL/PASM, ZILOG, and INTEL source code are supported. User may define mnemonics to meet the requirements of any particular assembler. A preconditioner program allows user to specify address ranges that contain ASCII data which will force DB or BYTE statements to be generated. User can optionally direct disassembly to include the actual start address of the instruction being decoded. Provision is made to allow the user to externally define up to 50 common systems locations and variables as name symbols.
Release: First release Nov. 1981; Ver 1.6 released Feb. 1982
Price: $150.00 prepaid, add $5.00 forOD.
Included with price: 5" or 8" disk and operators manual.
Where to purchase it:
Computer Toolbox, Inc.
1325 East Main St.
Waterbury, CT 06705
(203)754-4197

Program Name: Data Base CHECKER for IC-IRS (Info. Retrieval Sys)
Hardware System: 8" and North Star CP/M Systems
Minimum Memory Size: 40K
Language: MBasic or CBasic 2
Description: An Information Retrieval Program designed to handle a large body of rather static information where flexible access is required. Data Base files are created with any CP/M editor and use identifers for start of record, start of data, start and end of Search Keys. The CHECKDB.BAS program reports any missing identifiers. This helps eliminate major data base errors.
Release: January 1982
Price: Included at no charge with IC-IRS at $59.95 + $1.50 S&H.
Included with price: Users Manual and disk containing Demo Data Base and two CHECKDB.BAS programs.
Where to purchase it:
Elliam Associates
24000 Bessemer Street
Woodland Hills, CA 91367
(213)348-4278

Program Name: IBIOS.ASM (CP/M interactive BIOS)
Hardware System: Any CP/M system with a user-modifiable BIOS
Minimum Memory Size: 16K (CP/M 1.4); 20K (CP/M 2.2)
Language: 8080 assembly language
Description: IBIOS adds a powerful software-interrupt capability to any CP/M-based computer system. It permits immediate execution of user-definable systems commands from any program environment that performs I/O. With IBIOS the user is just a keystroke or two away from accessing utility programs in high memory, reassigning I/O devices, altering I/O device parameters, bank-switching memory, etc. IBIOS is transparent to currently running program and to CP/M. Because it is located at the bottom of the program hierarcy, IBIOS gives you continuous control of your computer.
Release: November 1981
Price: $60(US); $75 (foreign) single user, non-commercial use only
Included with price: Commented 8080 source code listings with command examples and installation instructions.
Where to purchase it:
Miken Optical Company
53 Abbett Avenue
Morristown, NJ 07960
(201)267-1210 or (201)543-7372.

Program Name: REFORMATTER CP/M—DEC File Conversion Program
Hardware System: Multi-drive, CP/M-based systems
Minimum Memory Size: 24K
Language: Assembly Language (8080)
Description: REFORMATTER is an intelligent bi-directional file conversion program. Reads and writes DEC RT-11 single density disks and gives CP/M users the ability to exchange files on floppy disk with DEC equipment. Provides complete facilities for file reorganization and directory maintenance. Users have the ability to alter any fields in the DEC RT-11 directory, to list the DEC directory, and display unused areas of the disk. A "squeeze" function consolidates "fragmented" DEC disks into a continuous data area. Can also be used for file exchange to DEC equipment running RSTS/E, RSX-11M and VAX/VMS operating systems as internal utilities on those systems' read and write RT-11 format.
Release: November 1980
Price: $195.00
Included with price: 8" disk, manual, and telephone consulation if required.
Where to purchase it:
MicroTech Exports, Inc.
467 Hamilton Avenue
Palo Alto, CA 94301
(415)324-9114

Program Name: PMS-II, Project Management System
Hardware System: Any system with CP/M and CBasic2
Minimum Memory Size: 64K
Language: CBasic2
Description: PMS-II is a project management system using critical path analysis to calculate early and late start and finish and float time for activity networks of up to 1000 activities. PMS-II allows 3, 4, 5, 6, or 7 day work-weeks, will schedule on per cent complete or actual start/finish. It takes budgeted vs. actual material, labor, and burden dollars. The user can define up to 100 holidays which PMS-II will schedule around. Reporting features: GANTT chart, activity-on-arc diagram, "net change" report for "what if" comparisons and an activity report writer with multiple sort/select options. Meets APSR 7-604.7 and C.E. ER 1-1-11 procurement and reporting requirements.
Release: June 1981
Price: $995.00
Included with price: User manual with tutorial; one year free update service; phone-in consulation.
Where to purchase it:
North America MICA, Inc.
11772 Sorrento Valley Rd. #240
San Diego, CA 92121
(714)481-6998

Program Name: TRANSLATE
Hardware System: Z-80, 8080 or 8085 system
Minimum Memory Size: Any size CP/M system
Language: Assembler
Description: Translates 8080 source code to 8086 source code. It preserves flags as they would be under the 8080. Attempts to remove colons from labels that reference DB'S, DW'S and DS'S as required by CP/M-86. Also puts colons on all code labels. Produces mnemonics that are used by CP/M-86. Saves considerable time over hand-translating existing 8080 source code.
Release: August 1981
Price: $45.00 plus postage
Included with price: Following files are provided on 8" SD disk. TRANS.COM, TRANS.CMD and TRANS.DOC
Where to purchase it:
Linmar
541 Ingraham Ave.
Calumet City, IL 60409
(312)868-4866 (ask for Mark)

Program Name: North Star Interface
Hardware System: Z80/8080/8085 incl. Altos, Superbrain, TRS-80 MOD II
Minimum Memory Size: 32K
Language: Assembler, "C"
Description: Converts North Star Basic to an executable program under I/OS, UNI/OS, MultI/OS, CP/M, CDOS or SDOS environments. Included is an NSDOS to CP/M file mover which moves files and programs from an SDOS disk to a CP/M disk. A source conversion program allows other Basic's and advanced system editors to be used with programs from North Star Basic. Two versions are available; one for NS single density Basic Rev. 4 and another for NS double density Basic Rev. 5.2.
Release: May 1977
Price: $100.00 end user. Call for OEM and dealer pricing
Included with price: User manual and three month subscriber service for updates.
Where to purchase it:
InfoSoft Systems Inc.
25 Sylvan Road South
Westport, CT 06880
(203)226-8937

Plain Talk About Business Computers

Can a Small Computer Really Save You Time?

Time is Money

Theophrastus said time was the most valuable thing a man could spend. Fifteen centuries later Haliburton agreed saying, "we reckon hours and minutes to be dollars and cents." Today, time is more valuable than ever—and more fleeting.

About the only way to gain time is to use it more efficiently and effectively. That's where we come in.

Small Business Computers—by the way, the "small" refers to computers, not to business—will dramatically increase your effectiveness and help save you time and money. How so?

You get flagrantly honest evaluations and reviews of computers and software. We don't just tell you what a program can do; we tell you what it doesn't do, what it does poorly, and what it should do for the price. If advertisers don't like that, we don't want their business, and you're better off without them. Fortunately, most companies appreciate our honesty. In fact, one of our reviewers has gained a reputation because of the many software houses that have incorporated his suggestions into their products. We're proud of that.

Photo courtesy of Alanthus Data Communications Corp.

Plain Talk

Small Business Computers explains the complexity of today's computerized business world without the technical jargon and doubletalk that may have held you back before. In its easily comprehensible "how-to" style, *Small Business Computers* answers your questions while providing the information you need to make some tough decisions. As you select, purchase, and install your computer system, *Small Business Computers* will guide you through each step calmly and comfortably—helping you to evaluate your computer needs and avoid unnecessary pitfalls. As you use your computer, be it mini or micro, *Small Business Computers* will be there to help you do so efficiently and with confidence while informing you of the latest developments and future possibilities of computers in business.

For Example

You have just purchased a mailing list program. Everything is fine until the file has to be sorted by zip code. If the program has that capability, all is well. If not, you have a big problem. If you had just invested a few hours reading *Small Business Computers*, you would have known what functions to look for before buying the program; you would have known how to plan for future needs. That's just one example. Expand this concept into other areas, other programs and systems, and you can see what you get for your investment.

Added Expertise

As the newest member of the Creative Computing family of fine computer publications, *Small Business Computers* will be expanding to offer subscribers more valuable information than ever before. *Creative Computing* editors and contributors will be unleashing their business expertise in *Small Business Computers* through articles, evaluations and applications of particular interest to the business person. *Creative Computing* has a reputation of editorial excellence and integrity built on unbiased, in-depth product evaluations; articles by top thinkers in the field; and pragmatic, innovative applications.

One management consulting firm, for example, used the Shell-Metzner sort described in *Creative*, and saved $3000 a month, and we still receive letters thanking us for the hardhitting, candid, evaluation of word processing printers we published over a year ago, and which, incidentally, cost us several advertisers.

All this knowledge and experience will now be available to business people in *Small Business Computers*.

So, don't let anyone give you that old story about how complicated and difficult computers are. We don't buy that. Our magazine—our whole philosophy—revolves around the sharing of honest information. If you don't know where to start, we'll put you on the right track. If you're already on the road, we'll show you the best route.

For Any Size Business

Whatever your business—manufacturing or banking, retail or research—*Small Business Computers* will increase your efficiency and help save you time and money.

Subscribe today; *Small Business Computers* is the best consultant your business will ever have.

SAVE up to 33%

Small **Business Computers** Magazine 8H01
P.O. Box 789-M
Morristown, N.J. 07960

YES! Send me *Small Business Computers Magazine* for the term checked:

- ☐ One year (6 issues) $12.97—I save 19%!
- ☐ Two years (12 issues) $22.97—I save 28%!
- ☐ Three years (18 issues) $31.97—I save 33%!

Savings based on full one-year subscription price of $16.

CHECK ONE: ☐ Payment enclosed. ☐ Bill me later.

Mr.
Mrs.
Ms.______________________
(please print full name)

Address______________________Apt.____

City______________________

State______________Zip__________

Foreign postage: Add $3 a year for Canada. Add $5 a year (cash payment in U.S. currency only) for all other countries outside U.S. and possessions. Please allow 60 to 90 days for delivery of first issue.

NEW PRODUCTS

S-100 Board with Eight RS-232 Interfaces

SSM Microcomputer Products Inc. has introduced the IO8 which contains eight asynchronous serial RS-232 ports, each individually accessible and programmable with individually programmable baud rates (110 to 19,200). Data activity is monitored via send/receive LED indicators on each line. A timer function, which provides master interrupt clocking (50 Hz/60 Hz), supports real-time or multiuser applications.

The IO8 also provides multiple interrupt modes, including priority, vectored, daisy chain, and maskable. Optional terminal and modem cables provide complete interfacing possibilities.

Price: $550; SSM Microcomputer Products Inc., 2190 Paragon Dr., San Jose, CA 95131, (408)946-7400.

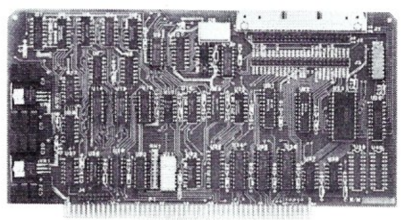
S-100 Computer System

Beavercreek Computer Systems announces a new IEEE compatible S-100 bus computer. It includes a heavy duty power supply, 8-slot motherboard, two Shugart 801R disk drives, 4MHz Z-80 CPU, single and double density disk controller with standard IBM formats, 64K dynamic RAM, five serial ports all under software control and CP/M 2.2. The unit is compatible with RS-232 CRT terminals and printers. A computer desk has been designed for the unit as an option for office applications. Double sided and double density disk drives, Winchester hard disk drives, and multi-user systems will be available in the future.

The retail cost, including manuals, is $4299.00 with quanity discounts available. Contact Beavercreek Computer Systems, 2465 Banyon Dr., Beavercreek, OH 45431, Phone: (513) 426-7670

S-100 Cartridge Disk Controller

Processor Interfaces, Inc. announces the Cartridge Disk Controller board containing the necessary logic to add a type 2315 or 5440 hard disk to an S-100 bus computer system. Depending upon the drive used, storage capacities from 1.25 to 10 Mbytes can be realized. Parallel I/O controls the disk drive, reads status of interface and transfers data to and from an on-board sector buffer memory. Controller logic does parallel and serial data conversions for reading and writing the disk.



The CPU commands the controller through a set of simple subroutines to Seek, Read, Write and move data to and from the sector buffer memory. Provided on floppy disk are disk formatter program, controller diagnostic program (the controller design includes diagnostic paths) and a CBIOS for Digital Research's CP/M-80 version 2.X. The CBIOS includes the above subroutines as well as additional routines such as sector blocking/deblocking and sector error checking.

Cartridge Disk Controller board, CBIOS, disk formatter, diagnostic and manual are $595.00. Introductory Price is $495.00 (until May 31, 1982). Available from Processor Interfaces, Inc., P.O. Box 154-A, Elm Grove, WI, 53122. Phone: (414)785-1245.

5-1/4" Winchester Disk For NorthStar Horizon

NorthStar Computers, Inc., announced its new 5-1/4" Winchester disk drive designed for the Horizon Series of microcomputers. The new hard disk provides greater data storage capacity and faster response times, and lets users expand their system to include multiuser capabilities.

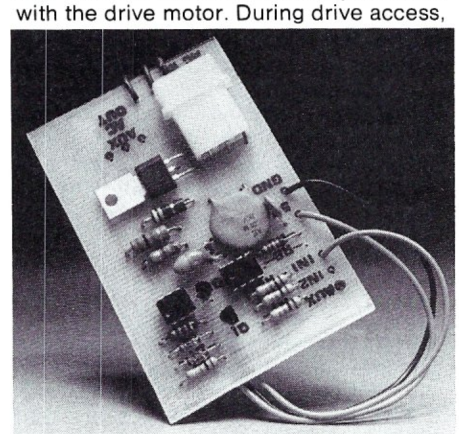
The 5Mb HDS-5 hard disk includes controller and power regulator boards, cables and NorthStar's Hard Disk Operating System (HDOS) software, and has been designed to replace one of the system's two disk drives. The system's remaining single quad disk drive remains in place.

The NorthStar TSS/A operating system runs the company's proprietary software for both word and data processing. TSS/C provides both multiuser CP/M and HDOS. With TSS/C, the multiuser Horizon Series can run any single-user CP/M application program without modification. It also supports NorthStar's Basic, MicroSoft Fortran and Cobol. Also unique to TSS/C is its 32K of extra memory, used to hold additional operating system features. It provides each user with a program area of more than 50K.

Users can expand their Horizon systems to support up to five simultaneous users. The HDS-5 is priced at $2,999, and is available from NorthStar Computers, Inc., 14440 Catalina St., San Leandro, CA.

Automatic 8" Floppy On/Off Control

Optronics Technology has introduced a new product for 8" Floppy Disk users which provides automatic on/off control for the drive motor, eliminating drive noise and significantly reducing media wear. The Drive Control Unit (D.C.U.) is designed for easy installation and has connectors which allow it to fit within the drive assembly in series with the drive motor. During drive access,

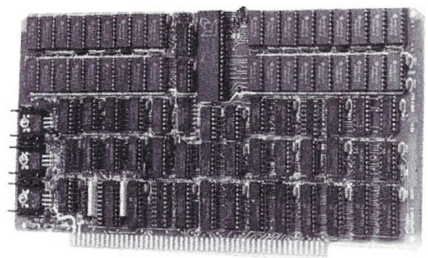


the motor is energized at zero-crossing for low noise and will turn itself off after eight seconds (adjustable) of idle time. The kit is $18.95 or $29.95 assembled; order from Optronics Technology, P.O. Box 81, Pittsford, NY 14534.

256K S-100 RAM Card

Piiceon's SUPERAM 50 RAM board provides 256K RAM with parity. The IEEE-696/S-100 bus standard level of compliance is: Slave D8/16 M24 T225 or T275.

SUPERAM 50 can be configured for either bank select or extended addressing. Extended addressing allows memory placement at any 256K boundary segment in a 16Mb address field. Bank select offers 64K banks for memory protect applications. Each bank may be individually set to any 64K block of the 256K address space, and may be enabled or disabled by I/O writing to a jumper-settable I/O address. The board responds dynamically to sXTRQ* line while operating in a byte mode (8-bit) or word mode (16-bit).



New Products, continued...

SUPERAM 50 is available in two speeds. SUPERAM 50-P: access time of 275 ns max; SUPERAM 50-AP: access time of 225 ns max.

The parity bit feature detects single bit errors per byte. If a parity error is detected, the ERROR* line is pulled low and an LED is lit until a SYSTEM RESET (POC* or RESET*) or a SLAVE CLR* is effected.

Power requirements: +8V @ 0.8A max., +16V @ 0.5A max., -16V @ 0.2A max.

The OEM quantity 100 price is $1300 for SUPERAM 50-P and $1400 for SUPERAM 50-AP. Piiceon, Inc., 2350 Bering Dr., San Jose, CA 95131.

Advertiser Index

ABM Products 10
Ackerman Digital Systems 11
Advanced Micro Digital .... cover 3
Avocet Systems 13
Blat R & D 58
Bower-Stewart & Assoc 92
California Digital Engineering 9
Cer-Tek, Inc. 73
CodeWork, The 71
Computer Design Labs 41
Computer Innovations 73
Computer Services Corp. 5
Computer Toolbox 61
Computing Interface 95
Daman 91
Digatek 92
Digicomp Research 96
Digital Research of Texas 20,21
Discount Software Group 63
Dual Systems 1
Dupre Enterprises 46
D & W Digital 19
Electronic Control Technology 35
ET Software Systems 95
Executive Data Systems 75
Fischer—Freitas 13
Hawkeye Grafix 95
InfoSoft Systems 59
Inner Access 31
Integrand 63
Jade Computer Products 47,48,49
JRT Systems 25
Knowlogy 51
Laboratory Computer Systems 71
Laboratory Microsystems 55
Lomas Data Products 29
Manx Software Systems 67
Meta Engineering 67
Microsystems 89
Micro Tech Exports 96
Morrow Designs .... cover 2
Omni Software Systems 86
PCE Systems 39
Potomac Micro—Magic 91
Priority One Electronics 14,15
Processor Interfaces 95
Protech 33
RD Software 67
Rosetta Stone, The 71
RR Software 2
S—100, Inc. 75
Scitronics 23
Small Business Computers 93
Software Banc 83
Software Connection, The 37
Software Toolworks, The 44
SPC Technologies 86
Starside Engineering 43
Static Memory Systems 17
Stok Computer Interface 12
SZ Software Systems 91,95
TecMar 53
Theta Labs 25

Performance, Quality, Reliability.

CPU 8086/87, with sockets for the 8087 math co-processor and 80130 interval timer/interrupt controller, is an advanced 16 bit processor designed for exceptionally efficient operation. And, it's compatible with 8 bit IEEE 696 peripherals as well as 16 bit equivalents. Looking for a 16 bit CPU that stands out above the crowd?

CPU 8086/87 is here.

Pictured above:
CPU 8086/87, Disk 2 Hard Disk Controller, Selector Channel, RAM 17 low power memory, and MPX 1 multiplexer for multi-user environments.

For business, scientific, and industrial computing solutions, contact your authorized CompuPro Systems Center.

CompuPro division, Godbout Electronics. Oakland Airport, CA 94614-0355.

Performance, Quality, Reliability.

CPU 8086/87, with sockets for the 8087 math co-processor and 80130 interval timer/interrupt controller, is an advanced 16 bit processor designed for exceptionally efficient operation. And, it's compatible with 8 bit IEEE 696 peripherals as well as 16 bit equivalents. Looking for a 16 bit CPU that stands out above the crowd?

CPU 8086/87 is here.

Pictured above:
CPU 8086/87, Disk 2 Hard Disk Controller, Selector Channel, RAM 17 low power memory, and MPX 1 multiplexer for multi-user environments.

For business, scientific, and industrial computing solutions, contact your authorized CompuPro Systems Center.

CompuPro division, Godbout Electronics. Oakland Airport, CA 94614-0355.