S-100
MICRO SYSTEMS™
MAR/APR 1980 VOL. 1/NO. 2
$2.00 U.S.A.

LINEAR PROGRAMMING TECHNIQUES IN PASCAL

See Page 14

Also in this Issue

*Registered trademark of Digital Research

and more

Complete Table of Contents on Page 3

S-100
MICRO SYSTEMS
TM

Volume 1 Number 2 **March/April 1980**

Editorial Correspondence should be sent to: S-100 MICROSYSTEMS, BOX 1192, Mountainside, NJ 07092.

S-100 MICROSYSTEMS is seeking articles on S-100 software, hardware and applications. Program listings should be typed on white paper with a new ribbon. Articles should be typed 40 characters/inch at 10 pitch. Author's name, address and phone number should be included on first page of article and all pages should be numbered. Photos are desirable and should be black and white glossy.

Commercial advertising is welcomed. Write to S-100 MICROSYSTEMS, Box 1192, Mountainside, NJ 07092, or phone Sol Libes at 201-277-2063 after 4 PM EST.

*TMK Digital Research

IN THIS ISSUE

DEPARTMENTS

S-100 MICROSYSTEMS is published six times a year by Libes Inc. Controlled circulation postage paid at Westfield, NJ 07090, and application to mail at controlled circulation postage rates is pending at Westfield, NJ. POSTMASTER: Please send Form 3579 to **S-100 MICROSYSTEMS,** P.O. Box 1192, Mountainside, NJ 07092.

The Editor's Page

by Sol Libes

The following is reprinted from the February 1980 issue of BYTE magazine and I feel requires no further comment here.

Battle of the Buses

In the October 1979 "BYTE News," page 107, Sol Libes contends, in an item about the S-100 bus, that "those who wish to have a machine capable of getting the maximum benefits of microprocessors must go the S-100 route." While Mr Libes was comparing the S-100 bus to all-in-one systems, such as the TRS-80 and PET, his statement leaves out a number of computer systems with as much capability as S-100 systems, perhaps more in some cases. For example, the SwTPC S/09 and the Ohio Scientific Challenger III Series are two systems that come to mind. The former uses a 6809 processor with the SS-50 bus (see October BYTE, inside front cover), and the latter uses 6800, 6502, and Z80 processors and apparently OSI's own bus (see back cover, same issue). Both of these systems have a 20-bit address bus for large memories. SwTPC and several other companies make SS-50 bus systems using the 6800. Other non-S-100 bus systems include the Heath H8 and H11. Any of these systems, and probably others that I have left out, can be as good for serious personal computer users as any S-100 bus computer. The S-100 bus is *not* the only possible route.

Mr Libes also writes that "the S-100 bus is not processor dependent." This statement is debatable, in spite of the existence of S-100 boards for a number of microprocessors. Several signals on the S-100 bus are generated ONLY by the 8080. Any other processor must be "bent" into generating (or responding to) these 8080-specific signals.

Personal computing could use a truly processor-independent bus. I feel that the S-100 bus will not be totally satisfactory in this role.

The mention of specific products in this letter does not necessarily constitute endorsement of these products. My point is simply that there are other buses besides the S-100, and that systems using these other buses can be just as capable as S-100 systems.

Jim Howell
5472 Playa Del Rey
San Jose CA 95123

Author Libes replies:

Thank you for your letter regarding my comments on S-100 systems in the October BYTE News column. Despite the views expressed in your letter, I still stand by my view that "those who wish to have a machine capable of getting maximum benefits of microprocessors must go the S-100 route." I agree with you that SS-50 and OSI Challenger III systems offer more power than integrated systems such as the TRS-80, Apple and PET. However, they still leave much to be desired compared to S-100. I will explain shortly.

Further, I also stand by my statement that "the S-100 bus is not processor dependent." The fact is that presently there are manufacturers selling six different 8-bit processor boards (8080, 8085, Z80, 6502, 6800 and 6809) and five different 16-bit processor boards (9900, LSI-11, 8086, Z8000 and Pascal Microengine) for S-100 systems. This means that eleven microprocessors have already been interfaced to the S-100. I do not know of any other system with this processor independence. Many of these microprocessors could not even be interfaced to buses such as the SS-50 or OSI without sacrificing performance.

When it comes to maximum power and flexibility the S-100 offers the following advantages over all other systems:

- *More software available. There are several times more languages, operating systems and applications packages for S-100 systems than for any other system.*
- *There are currently close to two dozen different manufacturers of S-100 mainframes and about fifty manufacturers of over 400 S-100 plug-in boards. This is many times more than for any other system.*
- *There is greater computer power capability with S-100. What other system has direct addressing of up to 16 megabytes of memory (24 address lines) and 64 K input/output ports (16 address lines), up to eleven vectored interrupt lines, up to sixteen masters on the bus (with priority), up to twenty-three plug-in slots on the motherboard, up to 10 MHz clock on the bus, plug-in operator front panel, and more.*
- *The S-100 bus is now standardized by the Institute of Electrical and Electronic Engineers (IEEE) assuring conformance among manufacturers.*

Regarding your reference to the H8 bus, note that Heath has discontinued production of this unit. Besides, it was dedicated exclusively to the 8080 and therefore was destined to an early death. The Heath H11 is essentially the same as and uses the same bus specifications as a Digital Equipment Corp LSI-11. Few other firms support the LSI-11 with products within the price range of the typical hobbyist. The hardware and software facilities, compared to the S-100, are limited and expensive.

Again, thank you for reading my column and I welcome any further comments you wish to make regarding my opinions.

Sol Libes

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

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

We're about to be gadflies again.

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

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

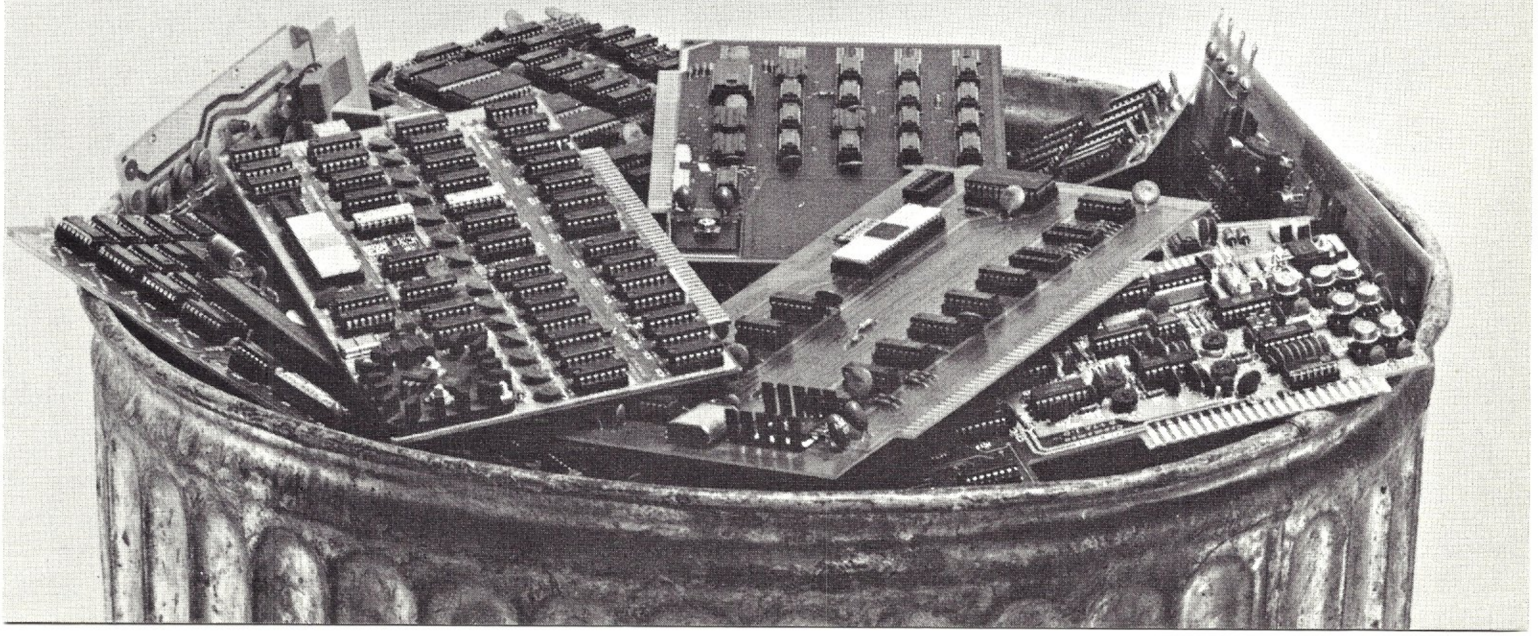
Consider some of the features: Reliable operation to 4MHz and beyond. Full compatibility with 8- and 16-bit CPUs, peripherals and other devices. *Eight* levels of prioritized interrupts. Up to 16 individually-addressable DMA devices, with IEEE Standard overlapped operation. User-selectable functions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

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

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

InterSystems™

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



S-100 MICROSYSTEM NEWS

by Sol Libes

UCSD REVOKES LICENSES TO DISTRIBUTE PASCAL

The University of California, San Diego (UCSD) has sent out a letter to all computer clubs that are licensed to distribute the UCSD Pascal package that their license is terminated effective April 2, 1980. These organizations paid $250 for the license and some of these organizations are considering filing suit against UCSD for breach of contract.

The clubs have distributed the UCSD Pascal package to their members at charges that ranged from $5 to $50. A UCSD Pascal user must now pay $250 to Softech, the new UCSD licensee, to obtain a copy of the package.

UCSD PASCAL NEWSLETTER PUBLISHED

The first issue of the UCSD Pascal Hobby Newsletter has appeared in print. It is 9 pages and is chock full of a lot of valuable informatin that users of UCSD Pascal will find invaluable. To get on the mailing list send $2 to: Jim McCord, 330 Vereda, Leyenda, CA 93017. It's an absolute bargain.

INTERNATIONAL PASCAL USER'S GROUP

If you are into Pascal you will also want to join the "International Pascal User's Group (PUG). You will get an occassional newsletter that is a couple of hundred pages that is a compendium of all the news thats fit to print about Pascal from all over the world. It contains a lot of useful programs, too! To join, send $6 to PUG c/o Dick Shaw, Digital Equipment Corp., 5775 Peachtree Dunwoody Road, Atlanta, Ga. 30342.

Z8000 S-100 CPU BOARDS TO BE OUT SOON

Ithaca Intersystems, Ithaca, N.Y., (formerly Ithaca Audio) will soon introduce a Z8000 CPU board for S-100 based systems. It will meet IEEE S-100 specs and contain an operating system in ROM. II will also have a version of their Pascal/Z software which can compile to either Z80 or Z8000 code. Therefore any present Z80 Pascal/Z software can be immediately recompiled to the new Z8000. II also plans to introduce a Z8000 assembler.

National Multiplex Corp., 78 Oliver Ave, Edison, NJ 08817 has also disclosed that they plan to soon have available a Z8000 CPU card for S-100 systems. No further details are available at this time.

Ithaca Intersystems has also disclosed that they have a prototype CPU card running using the new Motorola 68000, 16-bit, microprocessor IC. Since the 68000 is not yet in production there are no immediate production plans.

NEW S-100 8-BIT CPU CARDS TO BE AVAILABLE

Godbout Electonics, Oakland Airport, CA, plans to produce an S-100 card with dual microprocessors. The card will contain both 8085A and 8088 microprocessors. Both are Intel ICs. The 8085A is the new improved version of the 8080A and hence can execute all the present 8080 software. It will be clocked at 8MHz. The 8088 is the new Intel 16-bit micro with 8-bit I/O. Hence, it executes Intel's 8086 code. Thus a user can run standard CP/M written in 8080 code and software (e.g. 8086 BASIC) written in 8086 code. The board will also contain a memory manager circuit to provide extended memory addressing.

Tarbell Electronics, Carson, CA, will soon introduce a Z80 CPU card for S-100 systems. Don Tarbell said that he feels that the Z80, with its huge software base, will continue to be the dominant microprocessor for the next few years. The software base for 16-bit micros, he feels, will take at least a year or two to develop and hence he decided to introduce an 8-bit CPU card rather than a 16-bit CPU card.

CP/M USER GROUP NEWS

After over a year of no new disks, the CPM User Group will add eight new disks to the CPM User Group Software Library. The new disks were prepared by Bob Van Valzah of the Chicago Area Computer Hobbyist Exchange club (CACHE). This will bring the CP/M User Group library up to a total of 42 8-inch floppy disks.

The CP/M User Group will distribute copies to clubs, as in the past, for copying. This software will be available for copying at the CP/M User Group meeting at the Trenton Computer Festival, Trenton, NJ, April 19-20 (donation of $1/disk is asked for).

CP/M SOFTWARE DIRECTORY AVAILABLE

The Small Systems Group, Box 5429, Santa Monica, CA 90405 has prepared a directory which lists "all" CP/M applications programs and classifies them by type with the name address of the vendor. To get a copy send $2 or include a large self-addressed stamped envelope with $1.

XITAN/TDL NEWSLETTER PUBLISHED

All those TLD/Xitan system owners may no longer have TDL or Xitan for user support (what little there was of it) but they do have the XITAN NEWSLETTER published irregularly by: Dennis Thovson, 243 McMane Ave., Berkeley Heights NJ 07922. Send $1 for a sample copy.

MICROCOMPUTER SOFTWARE HONORED

For the first time a microcomputer software package has placed on the prestigous DATAMATION magazine "Honor Roll of Software Packages". Naturally it was CP/M, a product of Digital Research. Microsoft BASIC and UCSD PASCAL received honorable mention.

IMSAI LIVES ANEW

IMSAI, one of the pioneers in the microcomputer field is alive and functioning as the the IMSAI COMPUTER Division of Fischer-Freitas Corp., San Leandro CA. FFC was the outfit handling IMSAI's production and warranty service at the time that IMSAI went into bankruptcy, last year. Actually, the new IC division is staffed with all former IMSAI employees from the top down. The company is housed in two buildings totalling almost 12,000 sq. ft. and has 12 employees.

Fischer-Freitas purchased almost 90% of the finished IMSAI stock and the rights to use the IMSAI and IMDOS trademarks. The IC division is manufacturing and selling the full line of IMSAI Hardware products. They are presently selling copies of the IMDOS operating system which was purchased from IMSAI. However, they are still negotiating with Digital Research regarding updating and marketing of IMDOS, in the future. Todd Fischer, FFC director, said that he expects to have this resolved shortly. In the meantime, IC is supporting all present registered owners of the IMDOS software and is providing updates as needed.

Their address is: IMSAI COMPUTER Div, Fischer-Freitas Corp., 2175 Adams Ave, San Leandro CA.

- CONTINUED ON PAGE 56 -

LETTERS TO THE EDITOR

Sol:

First issue looks great! Thanks. Hope it goes well.

Jack Mathis
Union Carbide
Bound Brook NJ

Dear Sol:

I received the first issue of **MICROSYSTEMS** and was quite pleased with it. The magazine is still quite rough around the edges but shows a great deal of promise.

There are quite a few topics I would like to see covered. One of these is graphics for the S100. I think this is its weakest point. Also, a comparison of the S-100 with others, such as the Apple and TRS-80. Other topics might be a troubleshooting column, diagnostics, getting started with Pascal, software other than BASIC and many others.

Above all, keep it interesting. Thanks.

Hector M. Smith
Fountain Valley CA

Dear Sirs:

Please find enclosed my check for $14.00 for a two year subscription, as mentioned is the announcement in Infoworld.

I look forward with enthusiasm to the first issue, and hope you receive the widespread support a publication with your aids and audience deserves.

William T. Hole MD

To Sol Libes & Russell Gorr:

Bless you!!

The S-100 hobbyist has been increasingly taking a back seat in the micro world. The stores and mags are catering to either the small business market or the Pet/TRS-80/Apple crowd. I can understand it, because that's where the big bucks are. But that doesn't mean that I like it. Your magazine is a welcome move back toward the group that started it all.

William C. Burns
Palo Alto CA

Dear Sol:

Congratulations on the forthcoming S-100 **MICROSYSTEMS.** I am enclosing a check for a three year subscription. I'll await the first issue eagerly (but patiently).

I first heard that you were thinking about an S-100 publication from John Dilks around Christmas and it had been on my mind to write my best wishes. However, my clairvoyent card is not working so my printer never followed through. Then I read of **S-100 MICROSYSTEMS** in Computer World. You have your public relations act together -- an area that ON LINE could have paid more attention to. John suggested also that you might be looking for people and that I might contact you in that regard. With studied sanity, I have avoided doing that -- I am currently enjoying retirement (to just a full time job). However, if I can be of help in a pinch, let me know. When I come out of retirement, I may try my hand at writing. If so, you'll be hearing from me. If you're looking for software reviewers, keep me in mind.

Anyway, I hope that **S-100 MICROSYSTEMS** turns out to be as successful as I think it will. My one suggestion is to put heavy emphasis on CP/M as the 'software bus' that has made S-100 systems so usable. But I'm sure that you are ahead of me on that thinking.

My very best wishes for success. And thanks for your help and support with ON-LINE.

Dave Beetle
Los Gatos CA

Editor: Dave is the former publisher of ON-LINE, which until recently sold to Computer Shopper, was the buy-line type publication for computer hobbyists. Dave did a great job with it.

Dear Mr. Libes:

As a long time reader I congratulate you on your new venture. With your extensive preparation and professional style I look forward to your great success.

I am writing to tell you of another CPU chip operating on the S-100 bus (the 12th by my count). The Signetics 8X300 is a very fast 8-bit machine currently offered as a slave processor for BASIC (FORTRAN and PASCAL coming).

I'm not sure of the part number, but this board is also the first S-100 slave processor that I know of. I heard a rumor about a slave device using Western Digital's micro but have nothing concrete.

The BASIC processor is called a DLX-10 and is marketed by Alaska Computer Systems, 12759 Poway Rd, Poway CA, 92064.

I offer this information for your

- CONTINUED ON PAGE 55 -

CP/M* VERSION 2 FOR TRS-80★ MODEL II NOW AVAILABLE

Prices reflect distribution on 8" single density diskettes. If a format is requested which requires additional diskettes, a surcharge of $8. per additional diskette will be added.

All Lifeboat programs require CP/M, unless otherwise stated.

Software with Manual / Manual Alone

DIGITAL RESEARCH

☐ **CP/M FLOPPY DISKETTE OPERATING SYSTEM** — Ⓜ Packages supplied on diskette complete with 8080 assembler, text editor, 8080 debugger and various utilities plus full documentation. CP/M available configured for most popular computer/disk systems including: North Star Single, Double or Quad density, Altair 8" disks, Helios II, Exidy Sorcerer, Vector MZ, Heath H17† or H89†, TRS-80†, iCOM 3712 and iCOM Micro Disk plus many other configurations available off the shelf .... $145/$25
CP/M version 2 (not all formats available immediately) .... $170/$25

☐ **MP/M*** .... $300/$50

Lower prices!

☐ **MAC** — 8080 Macro Assembler. Full Intel macro definitions. Pseudo Ops include RPC, IRP, REPT, TITLE, PAGE, and MACLIB. Z80 library included. Produces Intel absolute hex output plus symbols file for use by SID (see below) .... $85/$15

☐ **SID** — 8080 symbolic debugger. Full trace, pass count and break-point program testing system with back-trace and histogram utilities. When used with MAC, provides full symbolic display of memory labels and equated values .... $70/$15

☐ **ZSID** — As above for Z80. Requires Z80 CPU $95/$25

☐ **TEX** — Text formatter to create paginated, page-numbered and justified copy from source text files, directable to disk or printer .... $70/$15

☐ **DESPOOL** — Program to permit simultaneous printing of data from disk while user executes another program from the console .... $45/$5

All Microsoft prices are discounted!

MICROSOFT

☐ **BASIC-80** — Disk Extended BASIC, ANSI compatible Ⓛ with long variable names, WHILE/WEND, chaining, Ⓜ variable length file records .... $300/$25

☐ **BASIC COMPILER** — Language compatible with Ⓛ BASIC-80 and 3-10 times faster execution. Produces Ⓜ standard Microsoft relocatable binary output. Includes Macro-80. Also linkable to FORTRAN-80 or COBOL-80 code modules .... $350/$25

☐ **FORTRAN-80** — ANSI 66 (except for COMPLEX) plus Ⓛ many extensions. Includes relocatable object compiler, linking loader, library with manager. Also includes MACRO-80 (see below) .... $400/$25

☐ **COBOL-80** — Level 1 ANSI '74 standard COBOL plus Ⓛ most of Level 2. Full sequential, relative, and indexed file support with variable file names. STRING, UNSTRING, COMPUTE, VARYING/UNTIL, EXTEND, CALL, COPY, SEARCH, 3-dimensional arrays, compound and abbreviated conditions, nested IF. Powerful interactive screen-handling extensions. Includes compatible assembler, linking loader, and relocatable library manager as described under MACRO-80 .... $625/$25

☐ **MACRO-80** — 8080/Z80 Macro Assembler. Intel and Ⓛ Zilog mnemonics supported. Relocatable linkable Ⓜ output. Loader, Library Manager and Cross Reference List utilities included .... $149/$15

☐ **XMACRO-86** — 8086 cross assembler. All Macro and Ⓛ utility features of MACRO-80 package. Mnemonics slightly modified from Intel ASM86. Compatibility data sheet available .... $275/$25

☐ **EDIT-80** — Very fast random access text editor for text Ⓛ with or without line numbers. Global and intra-line commands supported. File compare utility included. .... $89/$15

MICRO FOCUS

☐ **STANDARD CIS COBOL** — ANSI '74 COBOL standard compiler fully validated by U.S. Navy tests to ANSI level 1. Supports many features to level 2 including dynamic loading of COBOL modules and a full ISAM file facility. Also, program segmentation, interactive debug and powerful interactive extensions to support protected and unprotected CRT screen formatting from COBOL programs used with any dumb terminal .... $850/$50

☐ **FORMS 2** — CRT screen editor. Output is COBOL data Ⓛ descriptions for copying into CIS COBOL programs. Automatically creates a query and update program of indexed files using CRT protected and unprotected screen formats. No programming experience needed. Output program directly compiled by CIS COBOL (standard) .... $200/$20

☐ **HDBS** — Hierarchical Data Base System. CODASYL oriented with FILEs, SETs, RECORDs and ITEMs which are all user defined. ADD, DELETE, UPDATE, SEARCH, and TRAVERSE commands supported. SET ordering is sorted, FIFO, LIFO, next or prior. One to many set relationship supported. Read/Write protection at the FILE level. Supports FILEs which extend over multiple floppy or hard disk devices.

☐ **MDBS** — Micro Data Base System. Full network data base with all features of HDBS plus multi-level Read/Write protection for FILE, SET, RECORD and ITEM. Explicit representation of one to one, one to many, many to many, and many to one SET relationships. Supports multiple owner and multiple record types within SETs. HDBS files are fully compatible.

☐ **MDBS-DRS** — MDBS with Dynamic Restructuring System option which allows altering MDBS data bases when new ITEMs, RECORDs, or SETs are needed without changing existing data.

NOTE

HDBS-Z80 version .... $250/$35
MDBS-Z80 version .... $750/$35
MDBS-DRS-Z80 version .... $850/$35
8080 Version available at $75. extra.

Z80 version requires 20K RAM. 8080 version requires 24K RAM. (Memory requirements are additional to CP/M and application program.)

When ordering HDBS or MDBS please specify if the version required is for 1) Microsoft L80 i.e. FORTRAN-80, COBOL-80, BASIC COMPILER, 2) MBASIC 4. XX, or 3) BASIC-80 5.0.

Prices and specifications subject to change without notice.

Lower prices!

EIDOS SYSTEMS

☐ **KISS** — Keyed Index Sequential Search. Offers com- Ⓛ plete Multi-Keyed Index Sequential and Direct Access file management. Includes built-in utility functions for 16 or 32 bit arithmetic, string/integer conversion and string compare. Delivered as a relocatable linkable module in Microsoft format for use with FORTRAN-80 or COBOL-80, etc. .... $335/$23

☐ **KBASIC** — Microsoft Disk Extended BASIC with all Ⓛ KISS facilities, integrated by implementation of nine additional commands in language. Package includes KISS. REL as described above, and a sample mail list program .... $585/$45
To licensed users of Microsoft BASIC-80 (MBASIC) .... $435/$45

All Micropro prices are discounted!

MICROPRO

☐ **SUPER-SORT I** — Sort, merge, extract utility as abso- Ⓛ lute executable program or linkable module in Micro- Ⓜ soft format. Sorts fixed or variable records with data in binary, BCD, Packed Decimal, EBCDIC, ASCII, floating, fixed point, exponential, field justified, etc. Even variable number of fields per record! . $225/$25

☐ **SUPER-SORT II** — Above available as absolute pro- Ⓛ Ⓜ gram only .... $175/$25

☐ **SUPER-SORT III** — As II without SELECT/EXCLUDE Ⓛ Ⓜ .... $125/$25

☐ **WORD-STAR** — Menu driven visual word processing Ⓛ system for use with standard terminals. Text format- Ⓜ ting performed on screen. Facilities for text paginate, page number, justify, center and underscore. User can print one document while simultaneously editing a second. Edit facilities include global search and replace, Read/Write to other text files, block move, etc. Requires CRT terminal with addressable cursor positioning .... $445/$40

☐ **WORD-STAR/MAIL-MERGE** — As above with option Ⓛ for production mailing of personalized documents Ⓜ with mail list from Datastar or NAD .... $575/$25

☐ **WORD-STAR Customization Notes** — For sophisticated users who do not have one of the many standard terminal or printer configurations in the distribution version of WORD-STAR .... NA/$100

☐ **WORD-MASTER** Text Editor — In one mode has super- Ⓛ set of CP/M's ED commands including global search- Ⓜ ing and replacing, forwards and backwards in file in video mode, provides full screen editor for users with serial addressable-cursor terminal .... $125/$25

☐ **DATASTAR** — Professional forms control entry and Ⓛ display system for key-to-disk data capture. Menu Ⓜ driven with built-in learning aids. Input field verification by length, mask, attribute (i.e. uppercase, lowercase, numeric, auto dup., etc.). Built-in arithmetic capabilities using keyed data, constants and derived values. Visual feedback for ease of forms design. Files compatible with all CP/M-MP/M supported languages. Requires 32K CP/M .... $350/$35

☐ **CBASIC-2** Disk Extended BASIC — Non-interactive Ⓜ BASIC with pseudo-code compiler and runtime interpreter. Supports full file control, chaining, integer and extended precision variables, etc. .... $109/$15

☐ **PASCAL/M** — Compiler generates P code from ex- Ⓘ tended language, implementation of standard PAS- Ⓛ CAL. Supports overlay structure through additional procedure calls and the SEGMENT procedure type. Provides convenient string handling capability with the added variable type STRING. Untyped files allow memory image I/O. Requires 56K CP/M .... $350/$30

☐ **PASCAL/Z** — Z80 native code PASCAL compiler. Pro- Ⓘ duces optimized, ROMable re-entrant code. All interfacing to CP/M is through the support library. The package includes compiler, companion macro-assembler and source for the library. Requires 56K and Z80 CPU.
Version 2 includes all of Jensen/Wirth except variant records .... $275/$25
Version 3 Upgrade with variant records and strings expected 3/80 .... $395/$25

New Version New Features

☐ **PASCAL/MT** — Subset of standard PASCAL. Gener- Ⓜ ates ROMable 8080 machine code. Symbolic debug- Ⓧ ger included. Supports interrupt procedures, CP/M file I/O and assembly language interface. Real variables can be BCD, software floating point, or AMD 9511 hardware floating point. Version 3 includes Sets, Enumeration and Record data types. Manual explains BASIC to PASCAL conversion. Source for the run time package requires MAC (See under Digital Research). Requires 32K .... $250/$30

STRUCTURED SYSTEMS GROUP

☐ **GENERAL LEDGER** — Interactive and flexible system †† providing proof and report outputs. Customization of COA created interactively. Multiple branch accounting centers. Extensive checking performed at data entry for proof, COA correctness, etc. Journal entries may be batched prior to posting. Closing procedure automatically backs up input files. Now includes Statement of Changes in Financial Position. Requires CBASIC-2 .... $1250/$25

☐ **ACCOUNTS RECEIVABLE** — Open item system with †† output for internal aged reports and customer-oriented statement and billing purposes. On-Line Enquiry permits information for Customer Service and Credit departments. Interface to General Ledger provided if both systems used. Requires CBASIC-2. .... $1250/$25

☐ **ACCOUNTS PAYABLE** — Provides aged statements †† of accounts by vendor with check writing for selected invoices. Can be used alone or with General Ledger and/or with NAD. Requires CBASIC-2 .... $1250/$25

☐ **PAYROLL** — Flexible payroll system handles weekly, †† bi-weekly, semi-monthly and monthly payroll periods. Tips, bonuses, re-imbursements, advances, sick pay, vacation pay, and compensation time are all part of the payroll records. Prints government required periodic reports and will post to multiple SSG general ledger accounts. Requires CBASIC-2 .... $1250/$25

☐ **INVENTORY CONTROL SYSTEM** — Performs control †† functions of adding and depleting stock items, adding new items and deleting old items. Tracks quantity of items on hand, on order and back-ordered. Optional hard copy audit trail is available. Reports include Master Item List, Stock Activity, Stock Valuation and Re-order List. Requires CBASIC-2 $1250/$25

☐ **ANALYST** — Customized data entry and reporting sys- †† tem. User specifies up to 75 data items per record. Interactive data entry, retrieval, and update facility makes information management easy. Sophisticated report generator provides customized reports using selected records with multiple level break-points for summarization. Requires CBASIC-2 .... $250/$15

☐ **LETTERIGHT** — Program to create, edit and type letters or other documents. Has facilities to enter, display, delete and move text, with good video screen presentation. Designed to integrate with NAD for form letter mailings. Requires CBASIC-2 .. $200/$25

☐ **NAD** Name and Address selection system — interactive mail list creation and maintenance program with output as full reports with reference data or restricted information for mail labels. Transfer system for extraction and transfer of selected records to create new files. Requires CBASIC-2 .... $100/$20

☐ **QSORT** — Fast sort/merge program for files with fixed record length, variable field length information. Up to five ascending or descending keys. Full back-up of input files created .... $100/$20

GRAHAM-DORIAN SOFTWARE SYSTEMS

☐ **GENERAL LEDGER** — An on-line system; no batch- Ⓛ ing is required. Entries to other GRAHAM-DORIAN Ⓜ accounting packages are automatically posted. User †† establishes customized C.O.A. Provides transaction register, record of journal entries, trial balances and monthly closings. Keeps 14 month history and provides comparison of current year with previous year. Requires CBASIC-2. Supplied in source ... $995/$35

☐ **ACCOUNTS PAYABLE** — Maintains vendor list and Ⓛ check register. Performs cash flow analysis. Flexible Ⓜ — writes checks to specific vendor for certain in- †† voices or can make partial payments. Automatically posts to GRAHAM-DORIAN general ledger or runs as stand alone system. Requires CBASIC-2. Supplied in source .... $995/$35

☐ **ACCOUNTS RECEIVABLE** — Creates trial balance re- Ⓛ ports, prepares statements, ages accounts and rec- Ⓜ ords invoices. Provides complete information describ- †† ing customer payment activity. Receipts can be posted to different ledger accounts. Entries automatically update GRAHAM-DORIAN general ledger or runs as stand alone system. Requires CBASIC-2. Supplied in source .... $995/$35

☐ **PAYROLL SYSTEM** — Maintains employee master file. Ⓛ Computes payroll withholding for FICA, Federal and Ⓜ State taxes. Prints payroll register, checks, quarterly †† reports and W-2 forms. Can generate ad hoc reports and employee form letters with mail labels. Requires CBASIC-2. Supplied in source .... $590/$35

☐ **INVENTORY SYSTEM** — Captures stock levels, costs, Ⓛ sources, sales, ages, turnover, markup, etc. Trans- Ⓜ action information may be entered for reporting by †† salesman, type of sale, date of sale, etc. Reports available both for accounting and decision making. Requires CBASIC-2. Supplied in source ... $590/$35

☐ **JOB COSTING** — Designed for general contractors. Ⓛ To be used interactively with other GRAHAM-DORIAN Ⓜ accounting packages for tracking and analysing ex- †† penses. User establishes customized cost categories and job phases. Permits comparison of actual versus estimated costs. Automatically updates GRAHAM-DORIAN general ledger or runs as stand alone system. Requires CBASIC-2. Supplied in source $995/$35

☐ **APARTMENT MANAGEMENT SYSTEM** — Financial Ⓛ management system for receipts and security de- Ⓜ posits of apartment projects. Captures data on va- †† cancies, revenues, etc. for annual trend analysis. Daily report shows late rents, vacancy notices, vacancies, income lost through vacancies, etc. Requires CBASIC-2. Supplied in source .... $590/$35

☐ **CASH REGISTER** — Maintains files on daily sales. Ⓛ Files data by sales person and item. Tracks sales, Ⓜ over-rings, refunds, payouts and total net deposits. †† Requires CBASIC-2. Supplied in source ... $590/$35

☐ **tiny C** — Interactive interpretive system for teaching Ⓧ structured programming techniques. Manual includes full source listings .... $75/$40

New Features

☐ **BDS C COMPILER** — Supports most major features of Ⓜ language including Structures, Arrays, Pointers, re- Ⓘ cursive function evaluation, linking loader and library. Floating point function library included. Lacks data initialization and static and register class specifiers. Documentation includes "THE C PROGRAMMING LANGUAGE" by Kernighan and Ritchie. .... $125/$15

☐ **WHITESMITHS C COMPILER** — The ultimate in sys- Ⓛ tems software tools. Produces faster code than Pas- Ⓧ cal with more extensive facilities. Conforms to the full UNIX*** Version 7 C language, described by Kernighan and Ritchie, and makes available over 75 functions for performing I/O, string manipulation and storage allocation. Linkable to Microsoft REL files. Requires 60K CP/M .... $630/$30

☐ **POLYVUE/80** — Full screen editor for any CRT with Ⓜ XY cursor positioning. Includes vertical and horizontal scrolling, interactive search and replace, automatic text wrap around for word processing, operations for manipulating blocks of text, and comprehensive 70 page manual .... $135/$15

☐ **POLYTEXT/80** — Text formatter for word processing Ⓜ applications. Justifies and paginates source text files. Will generate form letters with custom fields and conditional processing. Support for Daisy Wheel printers includes variable pitch justification and motion optimization .... $85/$15

☐ **ALGOL-60** — Powerful block-structured language compiler featuring economical run time dynamic allocation of memory. Very compact (24K total RAM) system implementing almost all Algol 60 report features plus many powerful extensions including string handling direct disk address I/O etc. Requires Z80 CPU .... $199/$20

☐ **Z80 DEVELOPMENT PACKAGE** — Consists of: (1) disk Ⓜ file line editor, with global inter and intra-line facilities; (2) Z80 relocating assembler, Zilog/Mostek mnemonics, conditional assembly and cross reference table capabilities; (3) linking loader producing absolute Intel hex disk file .... $95/$20

☐ **ZDT** — Z80 Monitor Debugger to break and examine Ⓜ registers with standard Zilog/Mostek mnemonic disassembly displays. $35 when ordered with Z80 Development Package .... $50/$10

☐ **DISTEL** — Disk based disassembler to Intel 8080 or TDL/Xitan Z80 source code, listing and cross reference files. Intel or TDL/Xitan pseudo ops optional. Runs on 8080 .... $65/$10

☐ **DISILOG** — As DISTEL to Zilog/Mostek mnemonic Ⓜ files. Runs on Z80 only .... $65/$10

☐ **XASM-68** — Non-macro cross-assembler with nested conditionals and full range of pseudo operations. Assembles from standard Motorola MC6800 mnemonics to Intel hex .... $200/$25

☐ **XASM-65** — As XASM-68 for MOS Technology MCS-6500 series mnemonics .... $200/$25

☐ **TEXTWRITER III** — Text formatter to justify and pagi- Ⓜ nate letters and other documents. Special features include insertion of text during execution from other disk files or console, permitting recipe documents to be created from linked fragments on other files. Has facilities for sorted index, table of contents and footnote insertions. Ideal for contracts, manuals, etc. Now compatible with Electric Pencil***** prepared files .... $125/$20

New Features

☐ **POSTMASTER** — A comprehensive package for mail Ⓜ list maintenance that is completely menu driven. Features include keyed record extraction and label production. A form letter program is included which provides neat letters on single sheet or continuous forms. Compatible with NAD files. Requires CBASIC-2 .... $150/$15

☐ **WHATSIT?****** Interactive data-base system using associative tags to retrieve information by subject. Hashing and random access used for fast response. Requires CBASIC-2 .... $125/$25

☐ **XYBASIC** Interactive Process Control BASIC — Full disk BASIC features plus unique commands to handle bytes, rotate and shift, and to test and set bits. Available in Integer, Extended and ROMable versions.
Integer Disk or Integer ROMable .... $295/$25
Extended Disk or Extended ROMable .... $395/$25

☐ **SMAL/80** Structured Macro Assembled Language — Package of powerful general purpose text macro processor and SMAL structured language compiler. SMAL is an assembler language with IF-THEN-ELSE, LOOP-REPEAT-WHILE, DO-END, BEGIN-END constructs .... $75/$15

Lower Price!

☐ **SELECTOR III-C2** — Data Base Processor to create †† and maintain multi Key data bases. Prints formatted Ⓜ sorted reports with numerical summaries or mailing labels. Comes with sample applications, including Sales Activity, Inventory, Payables, Receivables, Check Register, and Client/Patient Appointments, etc. Requires CBASIC-2. Supplied in source ... $295/$20

☐ **GLECTOR** — General Ledger option to SELECTOR III-C2. Interactive system provides for customized COA. Unique chart of transaction types insure proper double entry bookkeeping. Generates balance sheets, P&L statements and journals. Two year record allows for statement of changes in financial position report. Supplied in source. Requires SELECTOR III-C2, CBASIC-2 and 52K system .... $250/$25

☐ **CPM/374X** — Has full range of functions to create or re-name an IBM 3741 volume, display directory information and edit the data set contents. Provides full file transfer facilities between 3741 volume data sets and CP/M files .... $195/$10

☐ **BASIC UTILITY DISK** — Consists of: (1) CRUNCH-14 Ⓜ — Compacting utility to reduce the size and increase the speed of programs in Microsoft BASIC and TRS-80 BASIC. (2) DPFUN — Double precision subroutines for computing nineteen transcendental functions including square root, natural log, log base 10, sin, arc sin, hyperbolic sin, hyperbolic arc sin, etc. Furnished in source on diskette and documentation ... $50/$35

☐ **THE STRING BIT** — FORTRAN character string han- Ⓜ dling. Routines to find, fill, pack, move, separate, concatenate and compare character strings. This package completely eliminates the problems associated with character string handling in FORTRAN. Supplied with source .... $45/$15

☐ **BSTAM** — Utility to link one computer to another also Ⓜ equipped with BSTAM. Allows file transfers at full data speed (no conversion to hex), with CRC block control check for very reliable error detection and automatic retry. We use it! It's great! Full wildcard expansions to send *.COM, etc. 9600 baud with wire. 300 baud with phone connection. Both ends need one. Standard and Ⓜ versions can talk to one another. Compatible TRSDOS version also available $150/$5

★ ★ ★ ★ ★ ★

SUNDRIES & NOTIONS

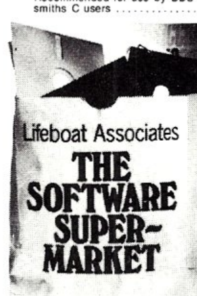
☐ **HEAD CLEANING DISKETTE** — Cleans the drive Read/Write head in 30 seconds. Diskette absorbs loose oxide particles, fingerprints, and other foreign particles that might hinder the performance of the drive head. Lasts at least 3 months with daily use. Specify 5" or 8" .... $20 ea./$45 for 3

☐ **FLIPPY DISK KIT** — Template and instructions to modify single sided 5¼" diskettes for use of second side in single sided drives $12.50

☐ **FLOPPY SAVER** Protection for center holes of 5¼" floppy disks. Only 1 needed per diskette. Kit contains centering post, pressure tool, tough 7-mil mylar reinforcing rings. Installation tools and rings for 25 diskettes $14.95
Re-orders of rings only $7.95

☐ **PASCAL USER MANUAL AND REPORT** — By Jensen and Wirth. The standard textbook on the language. Recommended for use by Pascal/Z, Pascal/M and Pascal/MT users $8

☐ **THE C PROGRAMMING LANGUAGE** — By Kernighan and Ritchie. The standard textbook on the language. Recommended for use by BDS C, tiny C, and Whitesmiths C users $12



Orders must specify disk systems and formats: e.g. North Star single, double or quad density, IBM single or 2D 256, Altair, Helios II, Micropolis Mod I or II, 5¼" soft sector (Micro iCOM SD Systems Dynabyte), etc.

Prices F O B New York. Shipping, handling and C O D charges extra.

Manual cost applicable against price of subsequent software purchase.

The sale of each proprietary software package conveys a license for use on one system only.

*CP/M is a trademark of Digital Research.
**Z80 is a trademark of Zilog, Inc.
***UNIX is a trademark of Bell Laboratories.
****WHATSIT? is a trademark of Computer Headware.
*****Electric Pencil is a trademark of Michael Shrayer Software.
★TRS-80 is a trademark of Tandy Corp.

†CP/M for Heath, TRS-80 Model I and PolyMorphic 8813 are modified and must use specially compiled versions of system and applications software.
††Recommended system configuration consists of 48K CP/M, 2 full size disk drives, 24 x 80 CRT and 132 column printer.

Ⓜ Modified version available for use with CP/M as implemented on Heath and TRS-80 Model I computers.

Ⓛ User license agreement for this product must be signed and returned to Lifeboat Associates before shipment may be made.

Ⓘ Ⓧ This product Includes/eXcludes the language manual recommended in Sundries and Notions above.

VISA · master charge

Shopping List No.10

Everything on Shopping List #10 runs on 64K TRS-80 Model II

Software for most popular 8080/Z80 computer disk systems including ***NORTH STAR, iCOM, MICROPOLIS, DYNABYTE DB8/2 & DB8/4, EXIDY SORCERER, SD SYSTEMS, ALTAIR, VECTOR MZ, MECA, 8" IBM, HEATH H17 & H89, HELIOS, IMSAI VDP42 & 44, REX, NYLAC, INTERTEC, VISTA V80 and V200, TRS-80 MODEL I and MODEL II, ALTOS, OHIO SCIENTIFIC, DIGI-LOG** and **IMS 5000** formats.*

™The Software Supermarket is a trademark of Lifeboat Associates

TM

Lifeboat Associates,
2248 Broadway, N.Y., N.Y. 10024
(212) 580-0082 Telex: 220501

NORTH ★ STAR TOPICS

by
Randy Reitz
26 Maple St.
Chatham Township, N.J. 07928

The first of a regular column by and for users of North Star Disk Systems.

Welcome to what I hope will be a regular feature in S-100 MICROSYSTEMS. With some support from North Star users I hope to bring you some interesting topics each month. Usually I will draw from my experience with the North Star MDS I started using in July '78, but I will welcome any contributions.

Some topics I can write about include Tiny-c, CP/M, several N* utility programs such as MARYELLN and finally N*'s version of UCSD Pascal. I will write about these topics as long as I have something to share. Of course I would like to hear about what is interesting to other N* users so let me hear from you.

I will get started this month with my recent experience with N*DOS release 5.1S. I picked up 5.1S (the "S" is for single density) in October 1979, but I didn't find the time to install it until recently. I also picked up a copy of the new SOFT-DOC System Software Manual that N* sells for $7.50. So far I can say that the Manual is well worth it, but 5.1S is not much different from release 4. Of course if you are lucky enough to have a double density MDS, you will need release 5 to get on the air. Double density is no doubt the reason for release 5.

So far, the major difference I have found in using 5.1S is the DOS is generally more "chatty" with you. Since the four DOS commands that require a buffer to operate were "kicked out" of release 5 (these commands are CF, CD, DT & CO - which was "kicked out" in release 4), I suppose N* used the extra space to put in more text to print out. For example, the hard disk error message is a bit more civilized (the type of error is displayed now) and when the system comes up a sign-on message is printed. This sign-on message caused me some trouble since I use the "auto" command feature at bootstrap time doubled.

As I was preparing this patch, I noticed that N* fixed a problem with the directory list routine. The check for the number of lines to print before pausing for the RETURN key has been restrictd to the console (device 0) only. In release 4, the pause would occur no matter what output device was selected for the directory listing.

This patch also goes in the directory printing routine at the point where one directory entry has finished printing and the directory maintenance routine is about to be entered to check for more entries to print:

```
        ORG   2597H
        CALLPRTBLK ;blank after type
        MOV   A,M     ;get file type
        INX   H       ;start of usr area
        CPI   1       ;is file type ABS?
        JNZ   BLK8    ;8 blks, CRLF?
        DS    7       ;skip 7 bytes
        JMP   CKCRLF  ;4 blks, CRLF?
RETURN  DS    1       ;is needed-return here
                      ; from patch
```

The remainder of the code can go in any convenient spot such as DOS I/O area.

```
*PATCH FOR DOUBLE COLUMN FILE DIR LISTING
BLK4    MVI   C,4     ;initialize counter
        CALL  PRTBLK  ;output one blank
        DCR   C
        JNZ   BLK4+2  ;output 4 blanks
        RET
BLK8    CALL  BLK4    ;output 8 blanks
CKCRLF  CALL  BLK4    ;after each user area
        LDA   DIRTG   ;check column toggle
        XRI   1       ;to see if CRLF is
        STA   DIRTG   ;needed
        CZ    DOSCRLF ;do CRLF
        JMP   RETURN  ;and return from PATCH
```

```
*CHECK IF A CRLF IS NEEDED AT END OF DIR
RSTDIR POP  H        ;get adr of dir toggle
       PUSH PSW      ;save flags
       XRA  A        ;prepare to zap toggle
       CMP  M        ;check first if zero
       MOV  M,A      ;see if another CRLF
       CNZ  DOSCRLF  ;is needed
       POP  PSW      ;restore flags
       LHLD ADRAVAL  ;set HL->next disk adr
       RET
```

This patch makes three references to N*DOS routines:

```
PRTBLK   EQU  26B4H  ;print one blank
DOSCRLF  EQU  26C1H  ;output CR and LF
ADRAVAL  EQU  28A2H  ;next avail disk adr
```

Also, the byte containing the number of console lines to display should be to load my spooling programs, a part of which is the console output driver. Needless to say that the console output drive had better be in memory to receive the N*DOS greetings. This "feature" of release 5 caused much shifting of my I/O drivers.

My major disappointment with release 5 is that N* did not take the opportunity to enhance the "LI" directory listing with double column output. The "PRESS RETURN TO CONTINUE" message for CRT consoles is nice, but I think a double column directory listing is better. I have had a double column directory patch for some time so this "feature" is not new for N*DOS (for example, Dr. Dobb's Journal had a letter for such a patch in the October, '79 issue); but I have never been completely satisfied with their performance. For example, the Dr. Dobb's patch will output spaces after each directory entry so that two entries will fit exactly on one line of your CRT. The CRT terminal is expected to provide an automatic CR and LF. Hence there is a version for 64 and 80 column CRTs. The double column patch I was using relied upon the fact that the directory maintenance routine stores the numbr of the directory entry being processed in the B-reg while the entry is being printed. My trick was to check for the B-reg to be odd before doing the CRLF. This worked except when the directory contained blank entries.

Here is the ultimate, perfect and completely fulfilling double column directory listing patch for version 5.1S (I will be happy to help figure out addresses for other versions). The only way to do this right is to allocate a flag to indicate which side of the CRT (or hard copy device) the directory entry is currently being displayed. The patch requires 43 bytes which can usually be found in the DOS I/O area. Here is the patch that goes at the end of the directory maintenance routine just before returning to DOS or the program calling entry DLOOK:

```
        ORG  243CH   ;Check if a CRLF
        CALL RSTDIR  ; is required.
DIRTG   DB   0
```

S-100 MICROSYSTEM CLUB DIRECTORY

S-100 MICROSYSTEMS will list in each issue, a directory of clubs with the S-100, CP/M, PASCAL, etc User Groups. If you are a member of such a club, and it is not listed here, then have your club representative register your group with us. In return we will send a free one-year subscription for your club's library.

Amateur Computer Group of New Jersey, Inc.
8080, Z80, CP/M, N*, Pascal User Groups
1776 Raritan Rd, Scotch Plains, NJ 07076
Contact: Sol Libes
Tel: (201) 277-2063

New York Amateur Computer Club, Inc.
S-100, N*, CP/M User Groups
Box 106, Church St. Station, NYC, NY 01007
Contact: (N*) Michael Dubno (212) 549-7359
(S-100) Brian Glasser (212) 674-1185

Northwest Computer Society
CP/M, Pascal, N* User Groups
Box 4193, Seattle, Wa 98104
Contact: (N*) Roy Gillette (206) 523-2866
Recorded Message (206) 284-6109

New England Computer Society
CP/M, N* User Groups
Box 198, Bedford, MA 01730
Contact: Dave Mitton (617) 493-3154

Long Island Computer Association
8080 User Group
3788 Windsor Drive
Bethpage NY 11714
Contact: 8080 UG Agugie Schwab (516)374-4168

Rochester Area Microcomputer Society
North Star User Group
Box 90808
Rochester NY 14609
Contact: N* UG Erwin Rahn 473-3184
Bob Konally 671-4131

- CONTINUED ON PAGE 55 -

Notice

Subscription Rates to Increase

Effective May 1st, the subscription rates of S-100 MICROSYSTEMS will be increased. The low introductory price, which represented a loss to us, will end. The subscription cost will then be as follows:

One Year (six issues)	$ 9.50
Two Years (twelve issues)	17.00
Three Years (eighteen issues)	23.50

Canada: add $3.00 per year	Paid
Far East, Australia, etc: add $14.00 per year*	in U.S.
OtherForeign:add $12.00 per year*	Funds

* sent air mail.

WOULD YOU LIKE TO BE AN S-100 MICROSYSTEMS DEALER?

S-100 MICROSYSTEMS is the S-100 magazine. In fact, there is no other competition. Our current dealers report that having S-100 MICROSYSTEMS magazine available for sale in their stores helps sell S-100 based computer systems.

We require a minimum order of only 10 copies per month and offer standard trade discounts. Further, we will take back any unsold saleable copies within six months.

For more information call or write:

S-100 MICROSYSTEMS
Box 1192
Mountainside NJ 07092
201-277-2063

LINEAR PROGRAMMING TECHNIQUES IN PASCAL

by
W. M. Yarnall
19 Angus Lane
Warren, N.J. 07060

This is the first part of a two-part article on Linear Programming in PASCAL. Linear Programming provides a means for solving problems with numerous constraints that often make solutions non-obvious.

INTRODUCTION

Linear Programming is defined as a technique for finding the optimum solution to a set of linear equations which are subject to constraints. It was first devised in 1947 under sponsorship of the U.S. Air Force and used to solve many problems in military planning.

Since that time, the main applications have been in the area of industrial planning.

Several algorithms have been developed for the solution of Linear Programming problems. Probably the most successful for general problems is the Revised Simplex method, first developed by George B. Dantzig (Stanford). The first successful solution of a Linear Programming problem on a high-speed electronic computer occured in January, 1962, on the National Bureau of Standards SEAC machine. Since that time, the simplex algorithm and its variations have been coded for most of the large general purpose computers in the US and England.

APPLICATIONS

Linear Programming problems are concerned with the efficient use or allocations of limited resources to meet desired objectives. There are four generally recognized classes of such problems:

1. The Product Mix Problems.
   A typical example is that of the manufacturer who must determine what combination of his available resource will enable him to manufacture his products in a way which not only satisfies his production schedule but also maximizes his profit.
2. The Transportation Problem.
   This is the example of a situation where material (of possibly different types) must be shipped from several sources to several destinations; the different routes from each of the sources to each of the destinations have differing costs. The problem is to satisfy the demands at each destination from the available stocks at a minimum cost for transportation.
3. The Diet Problem.
   This problem is also called by Lowe the "activity analysis" problem. Here we are given the nutritional content of a variety of foods (and their costs). We must find the mix which will satisfy the minimum daily requirement for each of the nutrients at a minimum cost.
4. Gaming Problems.
   This class of problems is one in which it is desired to maximize the payoff of a game (or contest) in which the variables are the strategies each player may use. Only the subclass of "Zero-Sum Two-Person" games have been solved so far.

Each of these problems have in common that:

* A multiplicity of choice exists in the solution,
* The choices are bounded by constraints, and
* Some objective must be optimized (maximized or minimized).

In this two-part article, an example of each of the classes will be taken up, and the steps in the formulation will be shown. The wide range of problems to which this technique can be applied is suggested by the recommended reading.

THE PROGRAM

The Revised Simplex Algorithm has been coded for use on small computers, and is shown in Listing 1. It has been implemented in UCSD Pascal, a language chosen for its wide acceptance and installation on small systems. It requires, as a minimum, 48K of RAM and a single floppy drive. The author uses a dual single-density North Star drive system with the North Star implementation of the UCSD Pascal.

First, the program is keyed in, using the Pascal editor. Then the program is compiled and linked to the library functions needed.

Before the program can be run, a data file must be build; part 2 of this article will provide a data file editor program (also in UCSD Pascal), and several example problems).

An Example Problem

In order to solve a problem using Linear Programming techniques, it must first be stated in a "standard form". It usually takes some analysis to get a problem into this standard form (see the bibliography for suggested reading in this area). Our example will be presented in the standard form.

Problem:

Maximize

$$X_1+2X_2+3X_3-X_4 \qquad (1)$$

subject to

$$\left.\begin{aligned} X_1+2X_2+3X_3 &= 15 \\ 2X_1+X_2+5X_3 &= 20 \\ X_1+2X_2+X_3+X_4 &= 10 \end{aligned}\right\} \qquad (2)$$

Here, equation (1) states an objective to be maximized (maybe a profit?), and equations (2) represent the constraints.

Since the program of Listing 1 is coded to minimize the objective function, we convert equation (1) to:

minimize

$$-X_1-2X_2-3X_3+X_4 \qquad (1a)$$

subject to the constraints of equation (2).

This demonstrates that any objective function can be converted from a minimum to a maximum (and vice versa) by a simple operation. Our standard form is the minimizing function (1a).

The representation of equations (2) used in the program is the matrix form. That is, the coefficients of the X's are kept in a two-dimensional array, called "ABAR" in the program. When this data is fed into the program (via reading a data file), the ABAR matrix will contain:

$$\text{ABAR} = \begin{vmatrix} 1.0 & 2.0 & 3.0 & 0.0 \\ 2.0 & 1.0 & 5.0 & 0.0 \\ 1.0 & 2.0 & 1.0 & 1.0 \end{vmatrix}$$

It can be seen that a column in this array corresponds to the coefficient of a variable (col 1 = X_1, etc), and the row corresponds to a constraint (there are 3).

The right-hand-side (RHS) of equation (2) is kept in an array, as are the coefficients of the objective vector.

After the data file is read in, the program echoes it out, and goes through several iterations, until it finds the required solution.

Listing 2 shows the data file, listed by the "LIST" function of the file editor. Each record (there are 22) is shown with a record sequence number. The file uses record variants; the first data is the "TAG". Tag 0 specifies

* an alphanumeric name for the problem (TEST),
* the number of rows (3) in the ABAR matrix, and
* the number of columns (4) (variables) in the problem.

This record must be the first in the file, since the use of the arrays is controlled by its data.

Record 1 has a tag of 1. It is optional and provides up to 64 characters to define the problem to which this data applies.

Records 2 through 20 can be in any order. It is best, however, to organize the data somewhat, so that it may be proof-read more easily. Here, records 2-4 have a tag of 2, which defines the right-hand-side (RHS) data, along with a row name and row index. Records 4, 9, 13 and 17 define column names, column indices and the coefficients of the objective vector (array). They have a tag of 4.

The rest of records 2-20 have a tag of 6, and define the ABAR coefficients, along with the row and column indices.

Records 18 and 19 are redundant, since the ABAR matrix is filled with zeros before the data is read in; they are included here only for completeness.

Record 21, with a tag of 99, is not necessary; it is, however, best to have it included to provide assurance that you have built and read the file correctly and completely.

Listing 3 show the results of a run with this data. The program first asks for the data file name; the initial data is listed. Note that row ABAR(M+1) is the objective function data.

The program then enters Phase 1, in which it finds a "basic feasible solution" which satisfies all the constraints. In this problem it goes through three iterations and lists the values of the X's at each stage. When a basic feasible solution is reached, Phase 2 is started. In this phase, the final optimization is done. The final results are printed:

$X_1 = 2.5$
$X_2 = 2.5$
$X_3 = 2.5$

The pointer to the variable is shown in the third column of the "LIST & X ARRAYS" printout; the last column shows the value of the variable. In this listing, the row labelled "M+1" shows, in the value columns, the negative of the cost; since we originally multiplied by -1, it also directly shows the profit. Note that none of X_4 was used in finding the optimum. It was in the problem at the start of phase 2, but the "better way" was found.

PROGRAM ERRORS

Several types of errors in data are tested for and reported (if found).

1. When the data file is read in, if the first record does not have a tag of 0, the output "BAD FILE FORMAT" is output. The run is aborted.

2. In Phase 2, the output "ERROR IN ITERATION N" indicates that a division by 0 was detected. The data has an error in it.

3. In Phase 1, the output "NO FEASIBLE SOLUTION AFTER N ITERATIONS" indicates that the constraints, as supplied, are inconsistent.

4. In Phase 2, the output "UNBOUNDED SOLUTION" indicates that the problem as posed, has a minimum at Negative Infinity. This problem cannot be solved as posed.

There is a possibility that, during Phase 2, the program may "cycle" - go through iterations in which multiple solutions are found having the same minimum. This generally means that the problem is over-constrained or has redundant constraints.

MORE TO COME

Part 2 of this article will provide an editor program to list and/or build or modify data files, and several problems as examples of the use of the Linear Programming Technique.

Suggested Reading

1. GASS, S.I.: "Linear Programming - Methods and Applications", McGraw-Hill, 1958.

2. SAATY, T.L.: "Mathematical Methods of Operations Research", McGraw-Hill, 1959.

3. HILLIER, F.S. & LIEBERMAN, G.J.: "Operations Research", Holden-Day, Inc. 1974.

NOTICE: SOFTWARE ON DISK AVAILABLE

The linear programming software listed in this article is available on 5-1/4 in. North Star disk for $20. This covers the cost of disk, mailing and handling. Order it directly from Bill Yarnall - address at beginning of article.

LISTING 1

The Linear Programming Algorithm. (Note: line numbers are not part of the program.)

```
ENTER FILE NAME ---> LINEARP.TEXT

PROGRAM LINEARP;
 (* MINIMIZE A COST FUNCTION SUBJECT TO CONSTRAINTS.
      MAXIMIZE NEGATIVE OF 'PROFIT' FUNCTION.    *)
CONST
 MAXROW = 32;
 MAXCOL = 64;
TYPE
 ROW = ARRAY [1..MAXROW] OF REAL;
 COL = ARRAY [1..MAXCOL] OF REAL;
 FREC = RECORD
          CASE TAG:INTEGER OF
              0: (NAME:STRING[6]; NUM1,NUM2:INTEGER);
              1: (HEADER:STRING[64]);
              2: (RNAME:STRING[6]; RINDEX:INTEGER; RHS:REAL);
              4: (CNAME:STRING[6]; CINDEX:INTEGER; OBJ:REAL);
              6: (R,S:INTEGER; T:REAL);
             99: ()
         END;
VAR
 U : ARRAY [1..MAXROW,1..MAXROW] OF REAL;
 ABAR : ARRAY [1..MAXROW,1..MAXCOL] OF REAL;
 X,XIK : ARRAY [1..MAXROW] OF REAL;
 LIST : ARRAY [1..MAXROW] OF INTEGER;
 ROWNAME : ARRAY [1..MAXROW] OF STRING[6];
 COLNAME : ARRAY [1..MAXCOL] OF STRING[6];
 FILEID : FILE OF FREC;
 FILNAM : STRING;
 F : FREC;
 HEADING : STRING[64];
 HDRFLAG : BOOLEAN;
 M,N,MP,M1 : INTEGER;
 PNAME : STRING[6];
 RESULT : INTEGER;
PROCEDURE PRINTH;
 BEGIN
  WRITELN(' ');
  WRITELN(' PROG. NAME = ',PNAME);
  WRITELN(' NO. ROWS   = ',M:6);
  WRITELN(' NO. COLS   = ',N:6);
  WRITELN(' ')
 END;    (* PRINTH *)
```

```

PROCEDURE PRINTB;
 BEGIN
  WRITELN(' ');
  WRITELN(' BAD FILE FORMAT');
  WRITELN(' ')
 END;    (* PRINTB *)

PROCEDURE PRINTC(B : ROW; C : COL);
 VAR
  I : INTEGER;

 BEGIN
  WRITELN(' ');
  WRITELN('      INITIAL DATA');
  WRITELN(' ');
  WRITELN(' OBJECTIVE VECTOR');
  WRITELN(' ');
  FOR I:=1 TO N DO WRITELN(COLNAME[I]:8,C[I]:14:8);
  WRITELN(' ');
  WRITELN(' RHS VECTOR');
  WRITELN(' ');
  FOR I:=1 TO M DO WRITELN(ROWNAME[I]:8,B[I]:14:8);
  WRITELN(' ')
 END;    (* PRINTC *)

PROCEDURE PRINTD;
 VAR
  I,J : INTEGER;

 BEGIN
  WRITELN(' ');
  WRITELN(' ABAR ARRAY');
  WRITELN(' ');
  FOR J:=1 TO N DO
   FOR I:=1 TO M DO
    WRITELN(I:6,J:6,ROWNAME[I]:8,COLNAME[J]:8,ABAR[I,J]:14:8);
  WRITELN(' ');
  WRITELN(' ABAR(M+1), ABAR(M+2)');
  WRITELN(' ');
  FOR I:=1 TO N DO
   WRITELN(COLNAME[I]:8,ABAR[M1,I]:14:8,ABAR[MP,I]:14:8);
  WRITELN(' ')
 END;    (* PRINTD *)

PROCEDURE PRINTX;
 VAR
  I : INTEGER;
  S : STRING[6];

 BEGIN
  WRITELN(' ');
  WRITELN(' LIST & X ARRAYS');
  WRITELN(' ');
  FOR I:=1 TO MP DO
```

```
   BEGIN
    S:='
    IF (LIST[I]<=N) THEN S:=COLNAME[LIST[I]];
    IF (I>M) THEN S:=ROWNAME[I];
    WRITELN(I:8,S:8,LIST[I]:7,X[I]:18:8);
   END;
   WRITELN(' ')
  END;    (* PRINTX *)

PROCEDURE INITIAL;
 VAR
  I,J : INTEGER;
  SUM : REAL;
  FIRSTIN : BOOLEAN;
  B : ARRAY [1..MAXROW] OF REAL;
  C : ARRAY [1..MAXCOL] OF REAL;

 BEGIN
  FOR I:=1 TO MAXROW DO
   FOR J:=1 TO MAXCOL DO ABAR[I,J]:=0.0;
  FIRSTIN:=FALSE;
  IF NOT EOF(FILEID) THEN F:=FILEID^;
  IF F.TAG = 0
  THEN
   BEGIN
    FIRSTIN:=TRUE;
    PNAME:=F.NAME;
    M:=F.NUM1;
    N:=F.NUM2;
    MP:=M+2;
    M1:=M+1;
    PRINTH
   END
  ELSE BEGIN PRINTB; RESULT:=2 END;
  GET(FILEID);
  WHILE (FIRSTIN) AND (NOT EOF(FILEID)) DO
  BEGIN
   F:=FILEID^;
   WITH F DO
   CASE TAG OF
     1: BEGIN    (* HEADER *)
         HEADING:=HEADER;
         HDRFLAG:=TRUE
        END;
     2: BEGIN    (* ROWNAME & RHS *)
         ROWNAME[RINDEX]:=RNAME;
         B[RINDEX]:=RHS
        END;
     4: BEGIN    (* COLNAME & OBJ *)
         COLNAME[CINDEX]:=CNAME;
         C[CINDEX]:=OBJ
        END;
     6: ABAR[R,S]:=T;
    99: ;
   END;    (* CASE OF TAG *)
```

```
    GET(FILEID)
   END;    (* WHILE *)
   IF FIRSTIN THEN
   BEGIN
    PRINTC(B,C);
    FOR J:=1 TO N DO ABAR[M1,J]:=C[J];
    FOR I:=1 TO M DO
     IF B[I] < 0.0 THEN
     BEGIN
      B[I]:=-B[I];
      FOR J:=1 TO N DO ABAR[I,J]:=-ABAR[I,J]
     END;
    FOR J:=1 TO N DO
    BEGIN
     SUM:=0.0;
     FOR I:=1 TO M DO SUM:=SUM-ABAR[I,J];
     ABAR[MP,J]:=SUM
    END;
    B[M1]:=0.0;
    SUM:=0.0;
    FOR I:=1 TO M DO SUM:=SUM-B[I];
    B[MP]:=SUM;
    FOR I:=1 TO MP DO
    BEGIN
     X[I]:=B[I];
     LIST[I]:=N+I;
     FOR J:=1 TO MP DO U[I,J]:=0.0
    END;
    FOR I:=1 TO MP DO U[I,I]:=1.0;
    PRINTD;
    ROWNAME[M1]:='M+1    ';
    ROWNAME[MP]:='M+2    ';
    PRINTX
   END;    (* IF FIRSTIN *)
   WRITELN(' ');
  END;    (* INITIAL *)

PROCEDURE PHASE1;   FORWARD;

PROCEDURE EXIT1(X:INTEGER);
  BEGIN
   RESULT:=1;    (* NORMAL EXIT *)
   WRITELN(' END OF PHASE 1 FOR ',PNAME,' AFTER ',X,
           ' ITERATIONS');
   PRINTX;
   EXIT(PHASE1)
  END;    (* EXIT1 *)

PROCEDURE EXIT2(X:INTEGER);
  BEGIN
   RESULT:=2;    (* ERROR EXIT *)
   WRITELN(' ERROR IN ITERATION ',X);
   PRINTX;
   EXIT(PHASE1)
  END;    (* EXIT2 *)
```

```
PROCEDURE EXIT3(X:INTEGER);
 BEGIN
  RESULT:=3;    (* NO FEASIBLE SOLUTION *)
  WRITELN(' NO FEASIBLE SOLUTION AFTER ',X,' ITERATIONS');
  PRINTX;
  EXIT(PHASE1)
 END;    (* EXIT3 *)

PROCEDURE PHASE1;
 CONST
  TOL = 1.00E-6;

 VAR
  ITER,I,J,L,KSAVE : INTEGER;
  SUM,TEMP,THETA,Z : REAL;
  XL,XLK : REAL;
  DEL,V,W : ARRAY [1..MAXROW] OF REAL;
  TEST : BOOLEAN;

 BEGIN
  WRITELN(' START PHASE 1');
  WRITELN(' ');
  ITER:=0;
  WHILE TRUE DO      (* LOOP HERE *)
  BEGIN
   IF(ABS(X[MP]) < TOL) THEN EXIT1(ITER);
   IF(X[MP] > TOL) THEN EXIT2(ITER);
   ITER:=ITER+1;
   FOR J:=1 TO N DO
   BEGIN
    SUM:=0.0;
    FOR I:=1 TO MP DO
     SUM:=SUM+U[MP,I]*ABAR[I,J];
    DEL[J]:=SUM
   END;
   TEST:=TRUE;
   FOR J:=1 TO N DO IF(DEL[J]<0.0) THEN TEST:=FALSE;
   IF TEST THEN EXIT3(ITER);
   TEMP:=1.00E+36;
   KSAVE:=0;
   FOR J:=1 TO N DO
    IF(DEL[J]<TEMP) THEN
    BEGIN
     TEMP:=DEL[J];
     KSAVE:=J
    END;
   FOR I:=1 TO MP DO
   BEGIN
    SUM:=0.0;
    FOR J:=1 TO MP DO SUM:=SUM+U[I,J]*ABAR[J,KSAVE];
    XIK[I]:=SUM
   END;
   THETA:=1.00E+36;
   L:=0;
   FOR I:=1 TO M DO
```

```
      IF(XIK[I]>0.0) THEN
      BEGIN
       Z:=X[I]/XIK[I];
       IF(Z=THETA) AND (LIST[I]>N)
       THEN L:=I
       ELSE
       IF(Z<THETA) THEN
       BEGIN
        THETA:=Z;
        L:=I
       END
      END;
     IF(L=0) THEN EXIT2(ITER);
     LIST[L]:=KSAVE;
     FOR I:=1 TO MP DO
     BEGIN
      V[I]:=XIK[I]/XIK[L];
      W[I]:=U[L,I]
     END;
     XL:=X[L];
     XLK:=XIK[L];
     FOR I:=1 TO MP DO
     BEGIN
      Z:=THETA;
      IF(LIST[I] <> KSAVE) THEN Z:=X[I]-XL*V[I];
      X[I]:=Z;
      FOR J:=1 TO M DO
      BEGIN
       Z:=W[J]/XLK;
       IF(I<>L) THEN Z:=U[I,J]-W[J]*V[I];
       U[I,J]:=Z
      END
     END;
     WRITELN(' ITERATION ',ITER,' OF ',PNAME);
     PRINTX
    END    (* WHILE *)
  END;   (* PHASE1 *)

PROCEDURE PHASE2;  FORWARD;

PROCEDURE EXIT4(X:INTEGER);
 BEGIN
  RESULT:=1;     (* NORMAL EXIT *)
  WRITELN(' END OF PHASE 2 FOR ',PNAME,' AFTER ',X,' ITERATIONS');
  PRINTX;
  EXIT(PHASE2)
 END;

PROCEDURE EXIT5(X:INTEGER);
 BEGIN
  RESULT:=2;     (* UNBOUNDED SOLUTION *)
  WRITELN(' UNBOUNDED SOLUTION FOR ',PNAME);
  PRINTX;
  EXIT(PHASE2)
 END;

```

```
PROCEDURE PHASE2;
 CONST
  TOL = -1.0E-6;

 VAR
  I,J,L,ITER,KSAVE : INTEGER;
  SUM,TEMP,THETA,Z : REAL;
  XL,XLK : REAL;
  DEL,V,W : ARRAY [1..MAXROW] OF REAL;
  TEST : BOOLEAN;

 BEGIN
  ITER:=0;
  WRITELN(' START PHASE 2');
  WRITELN(' ');
  WHILE TRUE DO     (* LOOP HERE *)
  BEGIN
   FOR J:=1 TO N DO
   BEGIN
    SUM:=0.0;
    FOR I:=1 TO MP DO SUM:=SUM+U[M1,I]*ABAR[I,J];
    DEL[J]:=SUM
   END;
   TEST:=TRUE;
   FOR J:=1 TO N DO IF(DEL[J]<TOL) THEN TEST:=FALSE;
   IF TEST THEN EXIT4(ITER);
   ITER:=ITER+1;
   TEMP:=1.0E+36;
   KSAVE:=0;
   FOR J:=1 TO N DO
    IF(DEL[J]<TEMP) THEN
    BEGIN
     TEMP:=DEL[J];
     KSAVE:=J
    END;
   FOR I:=1 TO MP DO
   BEGIN
    SUM:=0.0;
    FOR J:=1 TO MP DO SUM:=SUM+U[I,J]*ABAR[J,KSAVE];
    XIK[I]:=SUM
   END;
   TEST:=TRUE;
   FOR I:=1 TO MP DO IF(XIK[I]>0.0) THEN TEST:=FALSE;
   IF TEST THEN EXIT5(ITER);
   THETA:=1.0E+36;
   L:=0;
   FOR I:=1 TO M DO
    IF(XIK[I]>0.0) THEN
    BEGIN
     Z:=X[I]/XIK[I];
     IF(Z < THETA) THEN
     BEGIN
      THETA:=Z;
      L:=I
     END
    END;
```

```
   LIST[L]:=KSAVE;
   FOR I:=1 TO MP DO
   BEGIN
    V[I]:=XIK[I]/XIK[L];
    W[I]:=U[L,I]
   END;
   XL:=X[L];
   XLK:=XIK[L];
   FOR I:=1 TO MP DO
   BEGIN
    Z:=THETA;
    IF(LIST[I]<>KSAVE) THEN Z:=X[I]-XL*V[I];
    X[I]:=Z;
    FOR J:=1 TO M DO
    BEGIN
     Z:=W[J]/XLK;
     IF (I<>L) THEN Z:=U[I,J]-W[J]*V[I];
     U[I,J]:=Z
    END
   END;
   WRITELN(' ITERATION ',ITER,' OF ',PNAME);
   PRINTX
  END    (* WHILE *)
 END;    (* PHASE2 *)

BEGIN   (* MAIN *)
 WRITE(' ENTER DATA FILE NAME ---> ');
 READ(FILNAM);
 RESET(FILEID,FILNAM);
 WRITELN(' ');
 INITIAL;
 IF (RESULT <> 2) THEN PHASE1;
 IF (RESULT = 1) THEN PHASE2;
 IF HDRFLAG THEN WRITELN(' ',HEADING);
 WRITELN(' ');
 WRITELN(' ')
END.
```

LISTING 2

A sample data file named "LINTEST.DATA".

```
EDIT: L(IST, B(UILD, M(ODIFY, Q(UIT [1.0] L

LIST WHAT FILE? LINTEST.DATA

STARTING AT WHAT RECORD? 0
```

```
 0:   0  TEST          3      4
 1:   1 TEST OF LINEAR OPT., (GASS P.95)
 2:   2  ROW1          1    15.0000
 3:   2  ROW2          2    20.0000
 4:   2 -ROW3          3    10.0000
 5:   4  COL1          1    -1.00000
 6:   6 ROW  1 COL  1      1.00000
 7:   6 ROW  2 COL  1      2.00000
 8:   6 ROW  3 COL  1      1.00000
 9:   4  COL2          2    -2.00000
10:   6 ROW  1 COL  2      2.00000
11:   6 ROW  2 COL  2      1.00000
12:   6 ROW  3 COL  2      2.00000
13:   4  COL3          3    -3.00000
14:   6 ROW  1 COL  3      3.00000
15:   6 ROW  2 COL  3      5.00000
16:   6 ROW  3 COL  3      1.00000
17:   4  COL4          4     1.00000
18:   6 ROW  1 COL  4      0.00000
19:   6 ROW  2 COL  4      0.00000
20:   6 ROW  3 COL  4      1.00000
21:  99 LOGICAL EOF

EDIT: L(IST, B(UILD, M(ODIFY, Q(UIT [1.0] Q
```

LISTING 3

A sample run, using the data file "LINTEST.DATA".

```
ENTER DATA FILE NAME ---> LINTEST.DATA


PROG. NAME = TEST
NO. ROWS   =      3
NO. COLS   =      4


     INITIAL DATA

OBJECTIVE VECTOR

 COL1      -1.00000
 COL2      -2.00000
 COL3      -3.00000
 COL4       1.00000

RHS VECTOR

 ROW1      15.0000
 ROW2      20.0000
 ROW3      10.0000


ABAR ARRAY

     1      1  ROW1    COL1        1.00000
     2      1  ROW2    COL1        2.00000
     3      1  ROW3    COL1        1.00000
     1      2  ROW1    COL2        2.00000
     2      2  ROW2    COL2        1.00000
     3      2  ROW3    COL2        2.00000
     1      3  ROW1    COL3        3.00000
     2      3  ROW2    COL3        5.00000
     3      3  ROW3    COL3        1.00000
     1      4  ROW1    COL4        0.00000
     2      4  ROW2    COL4        0.00000
     3      4  ROW3    COL4        1.00000

ABAR(M+1), ABAR(M+2)

 COL1      -1.00000        -4.00000
 COL2      -2.00000        -5.00000
 COL3      -3.00000        -9.00000
 COL4       1.00000        -1.00000


LIST & X ARRAYS

       1  ROW1          5   15.0000
       2  ROW2          6   20.0000
```

```
  3  ROW3      7   10.0000
  4  M+1       8    0.00000
  5  M+2       9  -45.0000
```

```
START PHASE 1

ITERATION 1 OF TEST

LIST & X ARRAYS

  1  ROW1      5    3.00000
  2  ROW2      3    4.00000
  3  ROW3      7    6.00000
  4  M+1       8   12.0000
  5  M+2       9   -9.00000

ITERATION 2 OF TEST

LIST & X ARRAYS

  1  ROW1      2    2.14286
  2  ROW2      3    3.57143
  3  ROW3      7    2.14286
  4  M+1       8   15.0000
  5  M+2       9   -2.14286
```

- CONTINUED ON PAGE 33 -

AN INTRODUCTION TO CP/M*

by
Jake Epstein
Innotronics
Brooks Road
Lincoln, MA 01773

* Reg Tmk Digital Research

PART II
FILE STRUCTURE AND COMMAND SYNTAX

In my last article, I discussed the basic memory map of the CP/M ver. 1.4 operating system plus the dialog that occurs when the system is initially "booted up". In this article I will be discussing the command syntax of CP/M, basic file structure, and FLOPPY DISKETTE mapping.

To begin, I will describe the layout of the FLOPPY DISKETTE so that terms and concepts that I use later will be clearer to those who are not yet familiar with this storage device. Also, for the sake of clarity and to prevent confusion, I will limit the discussion for now to 8 inch, single density diskettes. The diskette is a thin magnetic disk made up off material similar to that used in audio recording tape, and is housed in a square package that gives the disk both protection from dirt and support. When placed in a device known as a DRIVE, the disk rotates 360 times every minute and data is read from and written to the diskette via a read/record head that moves in and out depending on information supplied by host computer or device. What has made the floppy diskette so economically viable is that when actuated, the head comes in contact with the revolving magnetic material thus eliminating the mechanical difficulties associated with hard disks where heads have to be extremely close to the medium but can not touch due the the injurious effects of abrasion. This is not to say that ebrasion is not a problem for diskettes,for they can eventually wear out through normal use and dirt contamination.

Of special interest to CP/M users is diskette layout. The 8 inch variety contains 77 TRACKS which are actually concentric circular areas on the disk. When disks are initially formated, these tracks are laid out, thus read/record head alignment will prove very important for accuracy of data transfer. When a drive is commanded to read or write a certain track, the read/record head moves in or out (also known as SEEK) to find the specific track. In order to calibrate the drive, the head will perform a seek TRACK 0 upon system reset, and thus, all movement of the head can be monitored by software. The track at the outside edge of the diskette is track 0 whereas the innermost track is number 76. Each track is divided into 26 SECTORS thus the total number of sectors on the diskette is 77 * 26 or 2002. In order to locate specific sectors, the first sector of each track, sector 01, is indicated via a hardware indicator which senses a hole in the diskette, the INDEX. Other than TRACK 00, and the INDEX, the type of diskette used by CP/M, SOFT-SECTORED FORMAT, has no other hardware indicators of sector and track location, and thus alignment of the read head coupled with data encoded on the disk during FORMATTING aid in sector location.

There is a type of diskette that has a series of 32 holes to indicate sector location. These HARD SECTORED FORMAT diskettes are incompatible with CP/M and should be avoided even though they can be made useable through formatting tricks. Each sector of a properly formated diskette contains both areas for data (as stored and retrieved by the user) and areas for identification and error checking. It is beyond the scope of this article to discuss formatting , but investigation of the references appended to this text will provide information on this subject. For now, all one needs to know is that each sector contains the track number, sector number, an area for the storage of 128 data characters (BYTES) and a CRC (CYCLIC REDUNDANCY CHECK) location to aid in detecting errors. Thus the available storage on each disk is:

77 TRKS * 26 SECTRS/TRK * 128 BYTES/SECTR

= 256 256 BYTES.

For the uninitiated, a byte is a standard data size of the computer industry which is 8 bits long and represents 256 different BINARY or base 2 numbers. These numbers as stored on diskette can either represent numeric data, special codes such as machine instructions, or alphanumeric characters in the form of the 7 bit ASCII code. Because diskettes are organized as discrete data structures, the 2002 sectors, special characters and/or identification headings are not needed within the data to aid in its exchange. Thus in contrast to sequential storage systems such as MAGTAPE or PAPER TAPE, the disk operating system can handle large streams of data without needing to check for beginnings and/or endings. The disadvantage of disk systems is that the minimum number of data bytes that can be read or written at any one time is dependent on sector size. In contrast, serial systems, such as MAGTAPE, can read/write one character at a time even though this is rarely done. A result of disk storage is that there will be times when storage space will be wasted, for the amount of data stored may not use up an entire sector. An advantage is that having discrete structure allows for random access to files and/or parts of files. In other words, to access a file on a tape, one has to read the entire tape to find the desired data or start of data whereas in disk systems, data can be seen as a series of physical locations. Finally, the term used for the data stored in a sector (128 bytes) is RECORD; thus a record is 128 bytes in CP/M. Also, the term FILE is used to describe a set of data. In other words, the binary data that comprises a program such as a text-editor would be stored on diskette as a file, or, the text to this article could be stored as a file.

At this point I shall change the subject and discuss COMMAND SYNTAX. First of all, when the entire operating system is in main memory, communication is usually accomplished via a CONSOLE device such as a CRT terminal or video-keyboard interface. Through some hardware and software manipulation, however, other devices such as modem boards for communication over telephone lines or card readers for BATCH style processing can be used. The subsystem of CP/M that directs and processes this dialog is the CCP (CONSOLE COMMAND PROCESSOR), and it does so by forming an interface between the user implemented INPUT-OUTPUT routines found in BIOS (BASIC I/O SYSTEM), and file handling routines found in BDOS, (BASIC DISK OPERATING SYSTEM). In a future article, I will discuss BIOS handler modification for different hadware configurations, and user access of BDOS routines for generation of hardware independent programs that can be run on any CP/M or equivalent system.

Conversation when CP/M is in the COMMAND MODE (communicates via CCP with console device) occurs via uppercase alphanumeric characters and CONTROL functions. The lowercase alphabet is accepted, but it is converted to upper case before processing or storage, and thus user programs that leave lowercase characters in areas that are used by CCP, i.e. file names, can cause difficulties later. Also important to note is that all text is buffered until a carriage return function is received at which time a line-feed is sent to the console and then the command text is interpreted and executed. When a character is typed at the keyboard, it is read and then transmitted or ECHOED back to the screen or printer of the console. A BUFFER is an area of temporary storage that can be found any where information needs to be held, and the area of system memory in the CP/M memory map reserved for both console and disk file buffering is location 080H to 0FFH. Thus a text string of 128 characters can be entered before the buffer overflows causing an error to occur. When this happens, the entire text as typed in will be automatically sent to the console followed by a "?", and when the CCP does not understand a command due to a mistake in syntax or other error, the same type of echo of text occurs.

The following list of special characters are reserved and are only used in certain situations:

< > . , : = ? [ ]

The functions implemented via the CCP are almost all file handling in nature, whereas other functions such as memory modify or single step program execution are provided for in utility programs that run under CP/M. Several of these are provided by DIGITAL RESEARCH on the original distribution diskette with the CP/M operating system, while others are available as seperate software packages. The standard CP/M utilities will be discussed in future articles.

As discussed above, since floppy diskettes are organized as a series of discrete records, then special characters such as SOH (START OF HEADING) or FS (FILE SEPARATOR) and/or identification headings that are required to monitor sequential or continuous data streams are not required. However, in certain file types, CP/M utilizes procedures and conventions used in other systems. Below is a table of four type of files used by CP/M:

BINARY	Used for storage of MEMORY IMAGES of programs in machine code.
ASCII	Used for text or source programs. An EOF (END OF FILE) separator is used after last character. In CP/M this is a control-z (01AH)
HEX FORMAT	Byte values of 8 bits are converted to two hexadecimal values each of which represents one 4 bit nibble. Each nibble value is stored as an ASCII character.
RELOCATABLE	Used by certain assemblers or compilers. This is a special coding of machine programs that can be made to run at any memory location by using linking utility programs.

In the above file types, binary is the most compact and can include any kind of data. What is meant by memory image is that the file is a copy of a block of data as it appeared in computer memory thus giving the user the ability to replicate a memory state at a future time after the computer has been shut off or the memory changed. The reason why an EOF is used with ASCII files is that this gives a method for retrieving an exact copy of a stored file as to length, for without this feature, the file would have to include all of the data that was unused from the last sector as explained above. Hex format is a very versatile data type in that software generated checksums can be incorporated into the file giving a means for error checking and correction. This however, causes considerable software overhead and requires a great deal more storage space than other data types. Because Hex format can be confusing, below is an example of data as represented by different data types and hex format.

165 Decimal = 1010 0101 Binary = A5 Hex =

0100 0001 Binary = 41 Hex = A ASCII
0011 0101 Binary = 35 Hex = 5 ASCII

In the above the hexadecimal equivalent of the number, A5, is stored as ASCII codes so immediately it should be apparent that hex files will be at least twice as long as binary files. Checksums are generated in different ways, but usually all the data bytes in a BLOCK (a sub unit of a file usually associated with large storage devices like mag tape) are added together and then an adjusted number is stored in a non-data area of the file. When files are read and the stored checksum is different than the one generated, then an error has occured. Some software systems have the means to correct errors. At present, CP/M can only detect errors by using hardware generated CRC (CYCLIC REDUNDANCY CHECKS) which are generated in a similar manner to checksums, but error correction is not available. Thus, using file types that use checksums can prove useful for increasing reliability of disk storage. It should be mentioned that actually, CRC errors are detected by routines in BIOS, so that different disk controllers handle the errors differently.

CP/M uses a special file called the DIRECTORY to store pointers to file locations on the diskette. This directory files located in 16 sectors on track 02. After a great deal of searching, I found that the following list of sectors contain the directory file on track 02:

sectors 1, 7, 13, 19, 25, 5, 11, 17, 23, 3
9, 15, 21, 2, 8, 14 (decimal)

The reason that the sectors are not in order, i.e. 1, 2, 3 etc., is that SKEWING is used to make reading and writing as efficient as possible. When two sectors are close together, hardware and software may not have time to identify and read/write the second one after doing so with the first before it slips by. In this situation, the system has to wait for one full revolution of the diskette for the second sector to come around again. The skewing for CP/M 1.4 is 6 but in other systems, it may be different causing incompatibility problems. In CP/M 2.0, this skewing can be modified by the user because this aspect of the system is handle in BIOS as opposed to version 1.4 where it is handled in BDOS. In a future article I will discuss this and other enhancements found in version 2.0. A word of warning to CP/M 1.4 users. If you have a CP/M system for 5.25 in disks and wish to add 8 inch disks, you will have problems because of sector skewing and track size. The same is true for 8 inch users that want to add the smaller drives. CP/M 2.0 eliminates this problem.

The data structure that stores information about each file on the disk is the FCB (FILE CONTROL BLOCK). Since I will be dedicating a lengthy discussion to the FCB in a future article on BDOS function calls, I will only describe a few concepts at this time. Each FCB has an area of 11 bytes in length that contains a PRIMARY and a SECONDARY name. The primary name can be any combination of up to 8 characters except for those that are reserved as mentioned above. Also, the primary name may be less than 8 characters, but when it is stored in the FCB, each empty position after the last character will be filled with ASCII "space" (20H). If a primary name is entered that is greater than 8 bytes, then it will be truncated upon storage. Names that are exactly the same as CCP function commands should not be used, for when files are accessed in the LOAD and EXECUTE function, the CCP will generate an error message ("?") because it will expect a command fucntion. Below is a list of possible primary names:

PRIMARY NAME	REPRESENTAION IN FCB
TEXTEDIT	TEXTEDIT
BASIC	BASIC___
PASCAL785	PASCAL78
F-80	F-80____
1234	1234____

The following names are not allowed.

PRIMARY NAME	REASON
LETTER?D	? IS RESERVED
JACOB E	SPACE IN NAME
REN	REN IS A CPM FUNCTION

Secondary names are used to indicate certain types of files, and thus the CCP and/or utility programs can determine the data type of the file. For example, as I shall explain in next month's article, a file with the secondary name of COM can be LOADED into memory and then EXECUTED as a program. DIGITAL REASEARCH has reserved several secondary names for use in the operating system, but as software becomes more available and diverse, new reserved secondary names are added to the list. Of course, the user can use any secondary name that he/she desires, but if he/she uses a reserved name, then the file should fit the criteria of that file type. Below is a listing of the most used reserved secondary names:

NAME	DATA TYPE	USE
COM	BINARY	PROGRAMS THAT CAN BE EXECUTED
HEX	HEX FORMAT	OBJECT OF ASSEMBLERS OR COMPILERS
ASM	ASCII	SOURCE CODE FOR ASSEMBLERS
MAC	ASCII	SOURCE CODE FOR MACRO-ASSEMBLERS
BAS	ASCII	SOURCE CODE FOR BASIC COMPILERS
FOR	ASCII	SOURCE CODE FOR FORTRAN COMPILERS
PRN	ASCII	PRINTOUTS OF TEXT FORMATORS, COMPILERS, ASSEMBLERS,ETC.
SUB	ASCII	SOURCE FOR SUBMIT UTILITY
$$$	-	TEMPORARY FILE MAY BE ANY FORMAT
LIB	ASCII	LIBRARY FILES
TEX	ASCII	ASCII FOR TEXT-FORMATORS
DOC	ASCII	MESSAGES OR DOCUMENTATION
MSG	ASCII	SAME AS DOC
TXT	ASCII	SAME AS DOC
REL	RELOCAT-ABLE	OBJECT OF RELOCATING ASSEMBLERS-SOURCE TO RELOCATING LINKERS

There are several that I left out, but in future articles I will try to include as many as I know of. In naming files, the primary and secondary names are written together but separated by a ".". This delimiter is not found in the FCB but represents the position between the eighth and ninth characters in the name block. Here are a few examples. Please remember that "_" represents space or ASCII 20H:

FILE NAME	FCB REPRESENTATION
BASIC.COM	BASIC___COM
FORTRAN8.DOC	FORTRAN8DOC
LETTER.1	LETTER__1__

A binary dump of a FCB name block with ascii equivalents would be:

005D42 41 53 49 43 20 20 20 434F 4D BASIC COM

where 005D is a location in system memory where the file name usually occurs and 42 41 etc. are the hexadecimal equivalents of ascii codes.

An area that can cause a great deal of confusion is ambiguous verses unambiguous file names. These terms refer to the ability of the CCP to work with files with similar but not identical names. Two special characters are used: "?" and "*" also known as "wild card". Ambiguous file names use these characters whereas unambiguous do not have them present. Also, file names as found in FCB's are unambiguous. Although I will get into more detail in next month's article when I discuss CCP command functions, the following example and explanation will hopefully give a better uderstanding of this concept.

```
ASM?.COM can describe - ASM1.COM or
                        ASMZ.COM or
                        ASMA.COM

JANE.??? CAN DESCRIBE - JANE.COM or
                        JANE.123 or
                        JANE.TEX
```

The wild card "*" character is used to replace an entire primary name, secondary name or both.

```
*.*     CAN DESCRIBE - FORTRAN.COM or
                       LETTER.TEX  or
                       TEST.DAT
```

These two characters are used mainly in listing out the names of files on a disk. For example, using the name *.COM with the DIR command of CCP would list all COM files.

The final aspect of CP/M file structure that I wish to discuss this month is sector allocation and file size. Each file control block has space for 16 data bytes. This list is referred to as the DISK ALLOCATION MAP in the CP/M documentation, and it is used by BDOS to find the ordered list of sectors comprising the file pointed to by the FCB. After a great deal of analysis, I discovered that each position in this table represents a block of 8 sectors. Block numbers 00 and 01 contain the directory as I listed above, but blocks 02 and up point to data files. BDOS also uses another byte in the FCB that keeps track of the total number of records (sectors) in each file in the event that not every sector in a block is used. For example, if a file needs only 3 sectors, then the other 5 in the block pointed to by the disk allocation map are unused. This phenomemon is the same as that discussed above in reference to unused sector space. By comparing actual file size with total available space, CP/M has a means of managing disk space for small file lengths. Since there are 16 positions in the disk allocation map, then the following figures can be calculated for the maximum storage capacity for one file pointed to by one FCB:

16 blks * 8 sectrs/blk * 128 bytes/sectr = 16 384 bytes

When a FCB becomes full, the byte containing the number of records becomes equal to 80H (128 decimal) and then BDOS creates (during write functions) or searches (during read functions) for a new FCB with the same file name as before. This new FCB is called an extension and BDOS is able to create or read up to 15 extensions. Thus in CP/M ver. 1.4, files can be created that are 16 * 16 384 or 262 144 bytes in length. Of course as calculated above, this is impossible because there are only 256 256 bytes of storage on a single density disk. The total amount of storage available for data is calculated below:

```
 256 256 bytes/disk
-  6 656 bytes/tracks 0 and 1 (system tracks)
-  1 024 bytes/directory file
-----------------------------------------------
 248 576 bytes
```

Since each block in a disk access table points to 8 sectors, then this total length in bytes is 1024 (128 x 8). When files are shorter than 1024 bytes or the last block of a file is not full, then this unused space will be wasted. If however, there were 64 files each a maximum of 16 384 bytes in length as calculated above, then total storage would appear to be 64 * 16 384 or 1048 576 bytes. (The reason why I chose the number 64 is that the total length of a FCB as found in the directory is 32 bytes, thus each sector can contain 128/32 or 4 FCBs. 16 sectors in the directory * 4 gives 64.) This is of course quite a bit more than the disk can store. Actually, the total storage space is determined by the fact that CP/M 1.4 supports 243 blocks, and since blocks 00 and 01 are used by the directory, the maximum storage on a disk when there is no unused space is 241 * 1024 or 246 784 bytes. Finally, the CCP command STAT will give the unused space on a disk. This is given in 1000 byte increments thus a disk with no files in the directory will appear to have 241K (K=1000) instead of 246K and the size of individual files will given to the nearest 1000 above the actual size. For example, STAT will give the size of a file of 256 bytes (2 sectors) as 1K, (the full block).

In conclusion, I have included a great deal of information that may or may not prove useful at this time to all users of CP/M, but hopefully, it will help you to expand your knowledge of file structure and management. In next month's article, I will get into more practical matters. I hope to cover CCP command functions, list utilities and there basic functions, and space permitting, begin to discuss BIOS functions and I/O handler modifications.

References:

"CP/M 2.0 User's Guide", (Set of 7 Manuals), Digital Research, Pacific Grove, Ca.

Edelson, Roger: "Super Chip FD1771"," Interface Age, vol. 1, nos. 11, 12, vol. 2, no. 1., Oct-Dec 1976.

"Series 400 Floppy Disk Drive Operation & Maintenance Manual", Innotronics, Lincoln, Ma.

Horowitz, E. and Sahni, S.: "Fundamentals of Data Structures", Computer Science Press, Inc., 1977, 564 pp.

Wiederhold, Gio: "Database Design", McGraw Hill, Inc., 1977, 658 pp.

FD1771 A/B - 01: "Floppy Disk Formatter/Controller Data Sheet", Western Digital, Newport Beach, Ca., Mar 77.

LINEAR PROGRAMMING

Continued from page 27

```
ITERATION 3 OF TEST

LIST & X ARRAYS

        1  ROW1        2     2.14286
        2  ROW2        3     3.57143
        3  ROW3        4     2.14286
        4  M+1         8    12.8571
        5  M+2         9    -0.00000048

END OF PHASE 1 FOR TEST   AFTER 3 ITERATIONS

LIST & X ARRAYS

        1  ROW1        2     2.14286
        2  ROW2        3     3.57143
        3  ROW3        4     2.14286
        4  M+1         8    12.8571
        5  M+2         9    -0.00000048

START PHASE 2

ITERATION 1 OF TEST

LIST & X ARRAYS

        1  ROW1        2     2.50000
        2  ROW2        3     2.50000
        3  ROW3        1     2.50000
        4  M+1         8    15.0000
        5  M+2         9     0.00000012

END OF PHASE 2 FOR TEST   AFTER 1 ITERATIONS

LIST & X ARRAYS

        1  ROW1        2     2.50000
        2  ROW2        3     2.50000
        3  ROW3        1     2.50000
        4  M+1         8    15.0000
        5  M+2         9     0.00000012

TEST OF LINEAR OPT., (GASS P.95)
```

ADDRESSING THE CURSOR

by
Larry Stein
Computer Mart of New Jersey
497 Lincoln Highway
Iselin N.J. 08830

How to get more professional looking video I/O via software control in your BASIC programs. This is the first part of a two-part article.

After you have gotten over the hurdle of writing your first program, you should be able to understand that the difference between good programs and great programs is not what they do, but how easily they allow you to do it!!!

One of the things you can do to add quality to your program is to use the video screen to its fullest capabilities. This is possible if you own either a terminal with addressable cursor (e.g. Hazeltine 1500, ADM3A,etc.),or a computer with a memory-mapped video screen, such as a SOL,VDM, TRS-80, PET or APPLE.

If your computer falls into the above categories, making the video screen look very professional and easy to use is a simple matter. This is done with a technique called "direct cursor addressing".

To understand direct cursor addressing, first draw a picture of your video screen on graph paper, allowing as many boxes horizontally and vertically as your screen contains. For example, if you own a 24 by 80 terminal, make sure your graph paper screen has 80 boxes across and 24 boxes down. Number the upper-left hand box 1,1 and the lower-right hand box 80,24. The upper-right hand box will therefore be 80,1 and the lower-left hand box will be 1,24. These numbers represent the column and row on your video screen. The 80 squares across the top are numbered 1 through 80 and the squares down the left are numbered 1 through 24. Therefore, any box on the video screen can be identified by giving its coordinates: first the column (horizontal position) then the row (vertical position); i.e. when the cursor is at the 15th column across and the 9th row down, the cursor is at (15,9).

In the illustration at the top of the next page the screen columns are numbered from 1 to 80 and the video screen rows are numbered from 1 to 24 corresponding to an 80 by 24 terminal screen. Also, the following characters in the above layout are at the designated addresses.

CHARACTER	ADDRESS row	column
a	3	18
b	6	3
c	18	72
d	9	45

Now we are ready to put the direct cursor addressing scheme to work in our program. I will use the BASIC language for all examples and the Hazeltine 1500, Lear Siegler ADM-3A and SOL terminal computer for all the examples. If you have another terminal or computer, refer to the manual for the direct cursor addressing commands. Usually, in order to move the cursor around the screen, C O L U M N S

```
             11111111112222222222333333333344444444445555555555666666666677777777778
    12345678901234567890123456789012345678901234567890123456789012345678901234567890
   1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   2XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   3XXXXXXXXXXXXXXXXXaXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   4XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   5XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   6XXbXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   7XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
R  8XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   9XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXdXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  10XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
O 11XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  12XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  13XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
W 14XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  15XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  16XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
S 17XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  18XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXcXXXXXXXX
  19XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  20XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  21XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  22XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  23XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  24XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

you will supply some special characters and then the column and row of the location where you want the cursor to be placed.

The Hazeltine 1500 terminal requires the receipt of a lead-in character followed by an escape followed by the column and row. The Lear Siegler ADM-3A requires the receipt of a lead-in character, followed by an equal sign followed by the column and row. The SOL terminal computer requires the receipt of a lead-in character followed by the character 2 followed by the column followed by another lead-in character followed by the character 1 and then the row.

The following table tells what decimal characters have to be sent to the 3 different terminals to place the cursor at the designated row and column.

characters:	1st	2nd	3rd	4th	5th	6th
Hazeltine 1500	126	17	column	row		
Lear Siegler ADM-3A	27	61	row	column		
SOL terminal computer	27	2	row	27	1	column

One additional factor in determining the row and column has to be considered. The terminals described here reference the screen relative to zero, which means that the upper left hand position is 0,0. The 1500 and the ADM-3A use a special formula to determine what character is output for the row and column to be addressed. Rather than go over the formula here, if you own either of these terminals, refer to the manual, or look at the sample program there to see the technique used.

Now in order to use what we have learned about direct cursor adressing, let's look at a sample program written in Microsoft BASIC version 4.51. This program includes routines for the previously mentioned terminals and asks which one is being used. The program then sets a software switch to use the proper routine for that terminal.

In this program, the variable X is used to designate the column (X-coordinate) and the variable Y is used to designate the row (Y-coordinate) on the screen. If we want to position the cursor at screen location column 15 and row 7 we would set X=15 and Y=7 and then GOSUB to the cursor subroutine. The cursor subroutine will position the cursor at (15,7), leave it there and return to the BASIC program where we will display something.

One special function is reserved for clearing the screen. If we set X=0 and Y=0 this will tell the cursor subroutine to clear the screen.

In the example program, line XXXX sets X and Y to 0 and then does a GOSUB to XXXX to clear the screen. The next statement in the program sets Y to 11 and X to 14 and does a GOSUB to the cursor routine. On returning, the BASIC program prints the program title which will appear on the screen at column 14 and row 11.

Following the rest of the program, you can see that on all screens which require data input, the headings of the fields to be filled in are first printed on the screen and then the cursor is placed next to each field name for data input. This allows the user to see all of the fields required before entering any data. Seeing the questions before having to supply the answers is always of benefit!! If you have any doubts as to the value of this kind of screen for data input purposes, just type in the sample program and run it.

The non-computer data entry operator will find it invaluable to use this kind of screen and there is no limit to the aids your program can provide. One of these operator aids is showing the field delimiters for the data being entered on the screen as in statement XXXX. Another valuable aid is to re-display the data that was just typed in, edited into the correct format. Just re-displaying the data will help when the operator has deleted any character in the field and BASIC has displayed the deleted characters surrounded by the familiar \ \. When you re-display the data, only the data will be printed on the screen as in statement XXXX.

I hope that this example and exercise will enable you to use your own imagination to make more use of the hardware you have at your fingertips. This is by no means a complete expose of all of the things that can be done with your screen on your microcomputer, but it will start you on your way away from the thinking that you have a teletype machine or other hardcopy device that can only scroll up the screen.

In the next issue of S-100 MICROSYSTEMS I will discuss the following programs in detail.

If your computer has a video screen,

USE IT!!!!!

```
10 REM   ********** PROGRAM NAME "LABELS"  11/6/79 **************
20 REM   *********************************************************
30 REM   *                                                       *
40 REM   *        PROGRAM FOR DISKETTE LABEL PREPARATION         *
50 REM   *                                                       *
60 REM   *********************************************************
70 REM   *********************************************************
80 REM   *                                                       *
90 REM   *    INITIALIZATION ROUTINE FOR SPECIFIC TERMINALS      *
100 REM  *                                                       *
110 REM  *********************************************************
120 CLEAR 2000
130 DIM A$(2) : REM DUMMY DIMENSION SO THAT ERASE WILL WORK LATER
140 PRINT:PRINT "THIS PROGRAM IS DESIGNED FOR ANY OF THE FOLLOWING:"
150 PRINT:PRINT "1 - LEAR SIEGLER ADM-3A"
160 PRINT:PRINT "2 - HAZELTINE 1500"
170 PRINT:PRINT "3 - SOL TERMINAL COMPUTER"
180 PRINT:PRINT "ENTER THE NUMBER OF THE ONE YOU ARE USING ";
190 Z$=INPUT$(1):PRINT Z$
200 IF Z$="1" THEN AM=1:WIDTH 80:GOTO 290
210 IF Z$="2" THEN AM=2:WIDTH 80:GOTO 290
220 IF Z$="3" THEN AM=3:WIDTH 64:GOTO 290
230 GOTO 140
240 REM  *********************************************************
250 REM  *                                                       *
260 REM  *             BEGINNING OF PROGRAM - TITLE              *
270 REM  *                                                       *
280 REM  *********************************************************
290 Y=0:X=0:GOSUB 2160
300 Y=11:X=14:GOSUB 2160
310 PRINT "DISKETTE LABEL PREPARATION PROGRAM - NOVEMBER 6, 1979"
320 Y=15:X=32:GOSUB 2160
330 PRINT "LARRY STEIN"
340 FOR Z=1 TO 1000:NEXT Z : REM SET FOR DELAY OF TITLE ON SCREEN
350 Y=0:X=0:GOSUB 2160
360 IF BG=1 THEN 460
370 PRINT "ENTER THE CHARACTER YOU WANT TO USE FOR BACKSPACE ";
380 GOSUB 2650:BS=IN:PRINT CHR$(BS)
390 PRINT "ENTER THE CHARACTER YOU WANT TO USE FOR FORWARD SPACE ";
400 GOSUB 2650:FS=IN:PRINT CHR$(FS)
410 PRINT "ENTER THE CHARACTER YOU WANT TO USE FOR INSERTING ";
420 GOSUB 2650:IT=IN:PRINT CHR$(IT)
430 PRINT "ENTER THE CHARACTER YOU WANT TO USE FOR DELETING ";
440 GOSUB 2650:DT=IN:PRINT CHR$(DT)
450 BG=1
460 PRINT "DO YOU WANT TO USE A PREVIOUSLY SAVED LABEL (Y/N) ";
470 Z$=INPUT$(1):PRINT Z$
480 IF Z$="Y" OR Z$="y" THEN 3100
490 PRINT "DO YOU WANT STANDARD PRODIGY LABELS (Y/N) ";
500 Z$=INPUT$(1):PRINT Z$
510 IF Z$="Y" OR Z$="y" THEN 2750
520 REM  *********************************************************
530 REM  *                                                       *
540 REM  *                GET LABEL PARAMETER                    *
550 REM  *                                                       *
560 REM  *********************************************************
570 GOSUB 2460
580 REM  *********************************************************
590 REM  *                                                       *
600 REM  *             DISPLAY LEFT SIDE OF SCREEN               *
610 REM  *                                                       *
620 REM  *********************************************************
630 Y=0:X=0:GOSUB 2160
640 PRINT
650 FOR N=1 TO LN
660 Y=N:X=1:GOSUB 2160
670 N$=STR$(N):IF LEN(N$)=2 THEN N$=" "+N$
680 PRINT "LINE ";N$;"  ]";TAB(WD+14);"]"
690 NEXT N
700 REM  *********************************************************
710 REM  *                                                       *
720 REM  *               GET LABEL INFORMATION                   *
730 REM  *                                                       *
740 REM  *********************************************************
750 FOR N=1 TO LN
760 Y=N:X=12:GOSUB 2160
770 PRINT A$(N)
780 NEXT N
790 FOR N=1 TO LN
800 I=1
810 L$=A$(N)
820 FOR M=1 TO WD
830 IF I=0 THEN 850
840 Y=N:X=11+M:GOSUB 2160
850 GOSUB 2650
860 IF IN=13 THEN M=WD:GOTO 950 : REM CARRIAGE RETURN
870 IF IN=BS THEN 1800 : REM MOVE CURSOR TO THE LEFT
880 IF IN=FS THEN 1800 : REM MOVE CURSOR TO THE RIGHT
890 IF IN=DT THEN 1990 : REM DELETE CHARACTER
900 IF IN=IT THEN 2070 : REM INSERT CHARACTER
910 I=0
920 MID$(L$,M,1)=IN$
930 PRINT IN$;
940 IF M=WD THEN 1900
950 NEXT M
960 A$(N)=L$
970 PRINT
980 NEXT N
990 PRINT
1000 PRINT "DO YOU WANT TO MAKE ANY CHANGES (Y/N) ";
1010 Z$=INPUT$(1):PRINT Z$
1020 IF Z$="N" OR Z$="n" THEN 1100
1030 Y=0:X=0:GOSUB 2160
1040 GOTO 630
1050 REM *********************************************************
1060 REM *                                                       *
1070 REM *           ROUTINE TO SAVE LABELS ON DISK              *
1080 REM *                                                       *
1090 REM *********************************************************
1100 PRINT "DO YOU WANT TO SAVE THIS LABEL ON DISK (Y/N) ";
1110 Z$=INPUT$(1):PRINT Z$
1120 IF Z$="N" OR Z$="n" THEN 1390
1130 PRINT "ENTER THE DRIVE ON WHICH LABEL IS TO BE STORED (A,B,C,D) ";
1140 D$=INPUT$(1):PRINT D$
1150 IF D$["A" OR D$|"D" THEN 1130
1160 D$=D$+":"
1170 F$=D$+"*.LAB"
1180 PRINT
1190 FILES F$
1200 PRINT:PRINT
1210 PRINT "ENTER A FILE NAME *** NOT *** IN THE ABOVE LIST"
1220 LINEINPUT "USE FILE NAME ONLY, NO EXTENSION ";Z$
1230 F$=D$+Z$+".LAB"
1240 OPEN "O",1,F$
1250 PRINT#1,WD$+","+LN$
1260 FOR N=1 TO LN
1270 PRINT#1,A$(N)
1280 NEXT N
1290 CLOSE
1300 PRINT
1310 F$=D$+"*.LAB"
1320 FILES F$
1330 PRINT
1340 REM *********************************************************
1350 REM *                                                       *
1360 REM *              ROUTINE TO ALIGN LABELS                  *
1370 REM *                                                       *
1380 REM *********************************************************
1390 PRINT "READY THE LABELS IN THE PRINTER AND PRESS RETURN ";
1400 Z$=INPUT$(1):PRINT
1410 PRINT "DO YOU WANT TO ALIGN THE LABELS (Y/N) ";
1420 Z$=INPUT$(1):PRINT Z$
1430 IF Z$="N" OR Z$="n" THEN 1570
1440 FOR N=1 TO LN
1450 LPRINT STRING$(WD,88)
1460 NEXT N
1470 FOR N=1 TO SK
1480 LPRINT
1490 NEXT N
1500 PRINT "DO YOU NEED MORE ALIGNMENT (Y/N) ";
1510 Z$=INPUT$(1):PRINT Z$:GOTO 1430
1520 REM *********************************************************
1530 REM *                                                       *
1540 REM *              ROUTINE TO PRINT LABELS                  *
1550 REM *                                                       *
1560 REM *********************************************************
1570 FOR M=1 TO NB
1580 FOR N=1 TO LN
1590 LPRINT A$(N)
1600 NEXT N
1610 FOR N=1 TO SK
1620 LPRINT
1630 NEXT N
1640 NEXT M
```

```
1650 PRINT "DO YOU WANT TO PRINT MORE LABELS (Y/N) ";
1660 Z$=INPUT$(1):PRINT Z$
1670 IF Z$="Y" OR Z$="y" THEN 1700
1680 STOP
1690 GOTO 1680
1700 PRINT "DO YOU WANT TO PRINT THE SAME LABEL (Y/N) ";
1710 Z$=INPUT$(1):PRINT Z$
1720 IF Z$="N" OR Z$="n" THEN 350
1730 GOSUB 2570
1740 GOTO 1000
1750 REM *********************************************************
1760 REM *                                                       *
1770 REM *        ROUTINE TO MOVE CURSOR RIGHT AND LEFT          *
1780 REM *                                                       *
1790 REM *********************************************************
1800 I=1
1810 IF IN=BS AND M[|1 THEN M=M-1:GOTO 840
1820 IF IN=FS AND M[|WD THEN M=M+1:GOTO 840
1830 I=0
1840 GOTO 840
1850 REM *********************************************************
1860 REM *                                                       *
1870 REM *       ROUTINE TO HANDLE CURSOR AT END OF FIELD        *
1880 REM *                                                       *
1890 REM *********************************************************
1900 GOSUB 2650
1910 IF IN=BS OR IN=FS THEN I=1:GOTO 840
1920 IF IN$=CHR$(13) THEN 950
1930 GOTO 1900
1940 REM *********************************************************
1950 REM *                                                       *
1960 REM *            ROUTINE TO DELETE A CHARACTER              *
1970 REM *                                                       *
1980 REM *********************************************************
1990 MID$(L$,M,WD-M+1)=MID$(L$,M+1,WD-M)+" "
2000 PRINT MID$(L$,M,WD-M+1)
2010 GOTO 840
2020 REM *********************************************************
2030 REM *                                                       *
2040 REM *            ROUTINE TO INSERT A CHARACTER              *
2050 REM *                                                       *
2060 REM *********************************************************
2070 L1$=MID$(L$,M,WD-M)
2080 MID$(L$,M,WD-M+1)=" "+L1$
2090 PRINT MID$(L$,M,WD-M+1)
2100 GOTO 840
2110 REM *********************************************************
2120 REM *                                                       *
2130 REM *         UNIVERSAL CURSOR POSITIONING ROUTINE          *
2140 REM *                                                       *
2150 REM *********************************************************
2160 ON AM GOTO 2220,2360,2290
2170 REM *********************************************************
2180 REM *                                                       *
2190 REM *        CURSOR POSITIONING FOR ADM-3A TERMINAL         *
2200 REM *                                                       *
2210 REM *********************************************************
2220 IF Y+X=0 THEN PRINT CHR$(26);:RETURN
2230 PRINT CHR$(27)+CHR$(61)+CHR$(31+Y)+CHR$(31+X);:RETURN
2240 REM *********************************************************
2250 REM *                                                       *
2260 REM *   CURSOR POSITIONING FOR THE SOL TERMINAL COMPUTER    *
2270 REM *                                                       *
2280 REM *********************************************************
2290 IF Y+X=0 THEN PRINT CHR$(11);:RETURN
2300 PRINT CHR$(27)+CHR$(2)+CHR$(Y-1)+CHR$(27)+CHR$(1)+CHR$(X-1);:RETURN
2310 REM *********************************************************
2320 REM *                                                       *
2330 REM *    CURSOR POSITIONING FOR HAZELTINE 1500 TERMINAL     *
2340 REM *                                                       *
2350 REM *********************************************************
2360 IF Y+X=0 THEN PRINT CHR$(126)+CHR$(28);:RETURN
2370 IF X[32 THEN X=X+96
2380 Y=Y+96
2390 D=D+1:IF D=5 THEN D=0:PRINT
2400 PRINT CHR$(126)+CHR$(17)+CHR$(X-1)+CHR$(Y-1);:RETURN
2410 REM *********************************************************
2420 REM *                                                       *
2430 REM *           ROUTINE TO DETERMINE LABEL SIZE             *
2440 REM *                                                       *
2450 REM *********************************************************
2460 LINEINPUT "ENTER LABEL WIDTH (IN CHARACTERS) ";WD$
2470 WD=VAL(WD$)
2480 LINEINPUT "ENTER NUMBER OF PRINT LINES PER LABEL ";LN$
2490 LN=VAL(LN$)
2500 ERASE A$
2510 DIM A$(LN)
2520 FOR N=1 TO LN
2530 A$(N)=STRING$(WD,32)
2540 NEXT N
2550 LINEINPUT "ENTER NUMBER OF LINES TO SKIP BETWEEN LABELS ";SK$
2560 SK=VAL(SK$)
2570 LINEINPUT "ENTER TOTAL NUMBER OF LABELS TO BE PRINTED ";NB$
2580 NB=VAL(NB$)
2590 RETURN
2600 REM *********************************************************
2610 REM *                                                       *
2620 REM *       ROUTINE FOR DIRECT INPUT FROM TERMINAL          *
2630 REM *                                                       *
2640 REM *********************************************************
2650 OUT 29,1
2660 WAIT 29,1,0
2670 IN=INP(28)
2680 IN$=CHR$(IN)
2690 RETURN
2700 REM *********************************************************
2710 REM *                                                       *
2720 REM *     ROUTINE TO GENERATE PRODIGY DISKETTE LABELS       *
2730 REM *                                                       *
2740 REM *********************************************************
2750 Y=0:X=0:GOSUB 2160
2760 PRINT "ENTER DISKETTE NUMBER XXXX";STRING$(4,8);
2770 LINEINPUT DS$
2780 PRINT "ENTER UNIT NUMBER     XXXX";STRING$(4,8);
2790 LINEINPUT UN$
2800 PRINT "ENTER DATE (MM/DD/YY) XX/XX/XX";STRING$(8,8);
2810 LINEINPUT DT$
2820 PRINT "ENTER DEALER NAME     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";STRING$(30,8);
2830 LINEINPUT DL$
2840 IF LEN(DL$)|30 THEN PRINT "DEALER NAME TOO LONG !":GOTO 2820
2850 ERASE A$
2860 DIM A$(8)
2870 A$(1)=" P R O D I G Y   S Y S T E M S,  I N C."
2880 A$(2)=" DISKETTE #"+DS$+"   UNIT #"+UN$+"  DATE "+DT$
2890 A$(3)=" "
2900 A$(4)=" DEALER: "+DL$
2910 A$(5)="  MASTER DISKETTE - RETURN IMMEDIATELY"
2920 A$(6)=" "
2930 A$(7)="COPYRIGHT (C) 1979 PRODIGY SYSTEMS, INC."
2940 A$(8)="       ALL WORLDWIDE RIGHTS RESERVED"
2950 LINEINPUT "ENTER TOTAL NUMBER OF LABELS TO BE PRINTED ";NB$
2960 NB=VAL(NB$)
2970 WD$="40"
2980 WD=VAL(WD$)
2990 LN$="8"
3000 LN=VAL(LN$)
3010 SK$="1"
3020 SK=VAL(SK$)
3030 GOTO 1390
3040 STOP
3050 REM *********************************************************
3060 REM *                                                       *
3070 REM *       ROUTINE FOR RETRIEVING LABELS FROM DISK         *
3080 REM *                                                       *
3090 REM *********************************************************
3100 PRINT "ENTER DRIVE ON WHICH LABEL IS STORED (A,B,C,D) ";
3110 Z$=INPUT$(1):PRINT Z$
3120 IF Z$["A" OR Z$|"D" THEN 3100
3130 F$=Z$+":*.LAB"
3140 PRINT
3150 FILES F$
3160 PRINT:PRINT
3170 PRINT "ENTER A FILE NAME FROM THE ABOVE LIST"
3180 LINEINPUT "USE FILE NAME ONLY, NO EXTENSION ";Z$
3190 F$=Z$+".LAB"
3200 OPEN "I",1,F$
3210 INPUT#1,B$,C$
3220 WD$=B$:WD=VAL(WD$)
3230 LN$=C$:LN=VAL(LN$)
3240 ERASE A$
```

Continued on Page 54

THE S-100 BUS— NEW VERSUS OLD

by Sol Libes

A comparison between the old Altair S-100 bus standard and the new IEEE S-100 bus standard.

The IEEE S-100 bus standard is being voted on now and will no doubt be adopted by the time the next issue of |S-100 MICROSYSTEMS| appears in print. The complete proposed standard was reprinted in the previous issue of this magazine.

There are approximately 200,00 "old" S-100 systems presently in operation. Many owners of these systems will be upgrading these systems with new CPU and peripheral boards to keep in step with the changing technology. How compatible will these new "IEEE S-100 compatible" boards be with "old" S-100 mainframes?

The IEEE S-100 standard has defined many bus lines which were not previously defined and redefined some pins. The roblem has been compounded by the fact that many S-100 board manufacturers took liberties with the loosely defined Altair bus and created some of their own pin functions. For example, pin 67 (now defined as PHANTHOM*) was used for as many as eight different functions by various S-100 manufacturers.

First of all lets compare the difference between the features of the old S-100 bus and the new IEEE S-100 bus. They are shown in Table 1.

TABLE 1
S-100 FEATURES

IEEE S-100	ORIGINAL S-100
1. Designed to support more powerful 8-bit (Z80) and 16-bit (e.g. Z8000) mpus	1. Designed specifically for 8080
2. Operating speed up to 10MHz	2. Operating speed - 2 MHz
3. 16 Megabyte direct memory addressing range	3. 65K direct memory addressing range
4. Up to 65K I/O ports	4. Up to 256 I/O ports
5. Up to 16 Masters on bus	5. Single Master operation
6. 8 or 16-bit data transfers	6. Only 8-bit data transfers
7. 10 vectored interrupts plus NMI	7. 8 vectored interrupts
8. Phantom	
9. Three more ground lines	
10. 3 undefined lines	19 undefined lines
11. 5 reserved for future use lines	

The Data Bus

The data bus has been changed so that it can support both 8-bit and 16-bit word transfers. This is shown in Table #2. Thus the user can insert either 8-bit or 16-bit CPU cards into the bus. In fact, the user can operate with both 8-bit and 16-bit processors on the bus via the DMA protocol. For an explanation of the operation of the data bus refer to last months article on the IEEE S-100 standard.

The Address Bus

The address bus has been greatly expanded from 16 direct address lines for 65K of memory and 256 ports for 8-bit processors to 24 direct address lines for up to 16 Megabytes of memory and up to 65K of I/O ports for 16-bit processors. This is shown in Table #3.

The Status Bus

A comparison of the old and new status bus is shown in Table #4. The number of lines has been reduced from 11 to 9 functions. Three status signals have been eliminated, namely STACK, RUN and SS (now used for ERROR*, RFU and NDEF, respectively). One new status signal has been added: sXTRQ*. Further, status signal labels now start with a lower-case "s" letter (except MWRT) while the old S-100 bus used and upper-case "S" lettr. Note also that an active low state is nowwindicated by an "*" compared to the over-bar in the old S-100. This was done because few printers can put an over-bar above a printed character.

TABLE #2
DATA BUS

IEEE S-100		OLD S-100
16 lines		16 lines
8-bit Master	16-bit master	8-bit master only
8-Data In 8-Data Out	16-Bidirectional & 8-Bidirectional or 8-Data In 8-Data Out	8-Data In 8-Data Out

TABLE #3
ADDRESS BUS

IEEE S-100	OLD S-100
16 or 24 lines	16 lines
(A0-A23)	(A0-A159)
16 Megabytes memory directly addressable	65K bytes memory directly addressable
I/O addressing	I/O addressing
up to 65K ports	up to 256 ports
(A0-A7 & A8-A15)	(A0-A7 or A8-A15)

TABLE #4
STATUS BUS

IEEE S-100		Old S-100
9 signals		11 signals
Memory Read	sMEMR	SMEMR
Op-Code Fetch	sM1	SM1
Input	sINP	SIN or INP
Output	sOUT	SOUT
Write Cycle	sWO*	$\overline{SWO}$
Interrupt Acknowledge	sINTA*	$\overline{SINTA}$
Halt Acknowledge	sHLTA	SHLTA
Memory Write	MWRT	MWRT
16-bit data transfer	sXTRQ*	
		deleted
		STACK - stack address on bus
		RUN - CPU in run mode
		SS single step operation

Control Output Bus

The control output bus is shown in Table #5. These signals determine timing and movement of data during a bus cycle. Two lines have been eliminated; PWAIT and PINTE (now defined as RFU). One new signal has been added; pSTVAL (previously this line was used for the Ø1 clock). A lower case "p" is now used instead of the upper case "P" to denote a control line.

Control Input Bus

The control input bus is compared in Table #6. These signals synchronize slave to master operation. Two new lines have been added; NMI* and SIXTN* previously these lines were not defined).

Vectored Interrupt Bus

This bus is compared in Table #7. These signals are used in conjunction with the INT* signal and a vectored interrupt controller circuit. Two new signal lines have been added (using previously undefined lines).

TABLE #5
CONTROL OUTPUT BUS

IEEE S-100		Old S-100
5 lines		6 lines
Indicates start of new bus cycle	pSYNC	PSYNC
Read Strobe	pBIN	PBIN
Write Strobe	pWR*	$\overline{PWR}$
Hold Acknowledge	pHLDA	PHLDA
added		deleted
Address and Status	pSTVAL	PWAIT-CPU in WAIT state
		PINTE-CPM interrupt flip-flop enabled

TABLE #6
CONTROL INPUT BUS

IEEE S-100		Old S-100
6 lines		4 lines
Used by slaves to sync master speed to slave speed	RDY	PRDY
Used by front panel to stop or single step master	XRDY	XRDY
Interrupt request to master	INT*	$\overline{PINT}$
Used by temporary master to request control of bus	HOLD*	$\overline{PHOLD}$
new lines		
Nonmaskable interrupt request	NMI*	
Response to sXTRQ*	SIXTN*	

TABLE #7
VECTORED INTERRUPT BUS

IEEE S-100		Old S-100
10 lines		8 lines
Inputs to an interrupt controller circuit which arbitrates among 8 inputs and asserts INT*. VIO has highest priority	VI0*	$\overline{VI0}$
	VI1*	$\overline{VI1}$
	VI2*	$\overline{VI2}$
	VI3*	$\overline{VI3}$
	VI4*	$\overline{VI4}$
	VI5*	$\overline{VI5}$
	VI6*	$\overline{VI6}$
	VI7*	$\overline{VI7}$
new lines		
Asserted when error occurs	ERROR*	
Asserted when impending power failure occurs	PWR FAIL*	

Utility Bus

The utility lines, compared in Table #9, have reduced in number from 34 to 22. This has occurred primarily through the use of many of the 19 previously "undefined" lines. The Ø1 clock signal was deleted since it was rarely used and has no meaning for most microprocessors which use only one clock signal. The PROT, UNPROTECT and PS functions have fallen into dis-use by present memory board manufacturers. The sense switch disable (SSW-DSB) has similary fallen into dis-use and has been replaced by GND and RFU lines.

On the other hand the ground lines have be increased from 2 to 5 lines to decrease the impedance of the ground circuit. Further, the location of the ground lines affords a small improvement in shielding between lines.

DMA Control Bus

This bus, compared in Table #8, has been greatly expanded from 4 to 8 lines, to allow for multimaster operation. These lines (DMA0 through DMA3) were previously undefined. The DMA control lines are used in conjunction with the HOLD* and pHLDA lines.

TABLE #8
DMA CONTROL BUS

IEEE S-100			Old S-100
8 signals lines			4 signal lines
Address disable	Disable PM signal drivers	ADSB*	$\overline{\text{ADD DSB}}$
Data Out disable		DODSB*	$\overline{\text{DO DSB}}$
Status disable		SDSB*	$\overline{\text{STA DSB}}$
Control disable		CDDSB*	$\overline{\text{C/C DSB}}$
new lines			
arbitrate among up to 16 masters encoded priority requests are asserted & lines contain priority no. of highest requestor		DMA0* DMA1* DMA2* DMA3*	

TABLE #9
UTILITY BUS

IEEE S-100		Old S-100
22 lines		34 lines
Clock signal	CLOCK	CLOCK
Master clock signal	Ø	Ø2
Resets all masters	RESET*	PRESET
Resets all slaves	SLAVE CLR*	EXT CLR
Power-on clear	POC*	$\overline{\text{POC}}$
Overlays slaves	PHANTOM*	
Not defined	NDEF(3 lines)	NDEF(19 lines)
Reserved for future use	RFU(5 lines)	
	+8V(2 lines)	+8V(2 lines)
	+16V	+16V
	-16V	-16V
	GND(5 lines)	GND(2 lines)
		deleted
		Ø1
		PROT
		UNPROT
		PS - protect status
		SSW-DSB

THE CGS-808 INTELLIGENT COLOR GRAPHICS BOARD

by
Jon Bondy
Box 148
Ardmore, Pa. 19003

Although most of the energy which I have applied to my computer system has been towards such mundane things as CRT's and printers, my main interests in having a home computer center around graphics and music. A few high density graphics boards have been available for the S-100 bus for some time (like the Matrox board), but they have been for the most part both inflexible and more expensive, so I devoted time to other projects and waited for the arrival of an inexpensive grahics board. In the fall of 1979, an ad for an Intelligent Color Graphics board appeared in BYTE from a firm of which I had never heard, called Biotech Electronics. They offered a kit for $99 which included the more expensive IC's along with a PC board and instructions. Somehow, this one seemed worth the risk, so I purchased it.

The board, called the CGS-808, uses the Motorola 6847 CRT controller chip to provide a wide range of low and high desnity color graphics, along with alpha-numeric characters in a 16 by 32 character format. It employs an on-board 8085 processor to do the graphics bookkeeping, allowing the main processor in the system to attend to other matters. Up to two on-board 2708 (or 2716) EPROMs are used to store programs which make the CGS-808 an intelligent peripheral device. Unlike some other graphics boards, the video re-fresh memory resides entirely on the CGS-808, making additional purchases of memory boards unnecessary, and leaving the address space of the main processor free for other uses. You can set up the 6847's modes, clear the screen, draw a dot, draw a line, and read or write the graphics refresh memory with simple commands from the central processor using their Firmware Pack I, and their other Firmware packages allow more complex graphics to be performed by the on-board 8085. The 8085 and the 6847 take turns accessing the video refresh memory, so there is no "snow" on the graphics screen while it is in use.

My primary interest in graphics is to do high density line drawings, and the CGS-808 provides enough density to start to do some serious work in its highest density mode with 256 by 192 dots on the screen. At this density, you give up the range of colors which are offered with lower density modes (from 64 by 32 display elements on up), but I intended to use the board with a black and white monitor anyway, so this was not a great loss to me.

The bare board kit arrived quickly and consisted of a set of Hardware and Software manuals, the PC board, and four ICs (8085, 6847, 2708 and 1372). Biotech does not warrantee these ICs because they claim that they may be damaged by static by an inexperienced user. I have never had problems with static so far, and I doubt that this would be as large a problem as Biotech claims. In any event, most other manufactureres will replace ships which prove to be defective, whether they are MOS or not. The only defective chip which I found in the entire process of building my kit was a 74LS74 which was purchased from another vendor. Total parts cost for the board (aside from the kit) was about $150.

The board was as well made as I have seen in Hobby products, with IC locations and types silk-screened over a good solder mask. Complete discussions of board theory (both hardware and software) were contained in the manuals, along with step-by-step assembly instructions and sample programs. The discussions on board use and the 6847 were at times somewhat cryptic, but all of the information is there, and the sample programs were very useful. One surprise here was that their section on debugging the board was four pages long, since many vendors omit that section entirely. Complete schematics for the board were included, but there was no listing of the Firmware EPROM, so that when the board did not work immediately, it was a bit difficult to determine if it was a software or hardware problem. Biotech has informed me that a complete listing of the ROM will be made available at a nominal charge by the time this article is published.

Acquiring the hardware to complete the kit took a bit longer than had been anticipated, with some of the 74LS300 series ICs being the major problem. The Hardware manuals list the parts by schematic resistor, capacitor and IC number first, and then provides a summary for parts ordering, which was very useful. The RAM on the board consists of 2114s, so they were plentiful and reasonably priced.

Before I discuss my debugging problems, let me describe how the main processor communicates with the CGS-808. The CGS-808 reserves four input andfour output ports for communication between the central processor and the on-board processor, and these ports can be placed anywhere in the range of 256 port addresses as long as they start on a port number which is a multiple of four. Of the four ports, only two are actually used by the board, with two unused. It turns out that it is possible to use one of these unused ports for an interesting purpose, which I will get to later. Let's assume that we have set the ports up to start at port-0.

The two ports which the CGS-808 normally uses serve different purposes. Writing to port-0 serves to send both OP codes and parameters to the 8085. The OP codes are numeric values which indicate how the parameters which follow are to be processed, and they consist of 0 (clear screen), 1 (set mode), 2 (plot point), 3 (draw line), 4 (alphanumeric/semigrahic characters), 5 (read screen memory) and 6 (write screen memory). The parameters vary with each of the OP codes as appropriate; for instance, the draw line OP code (3) requires starting X and Y coordinates, ending X and Y coordinates, and a line color.

When the OP code and its associated parameters have been sent to port 0, writing any value at all to port-1 starts the 8085 off executing the requrested operation. the data sent to this port is immaterial, and is ignored.

Reading port-0 provides you with some statusinformation such as whether an invalid OP code has been received, whether the board is so confused by the sequence of data that you should reset it (hardware reset), whether it thinks it is imputting parameters or not, and whether it is ready for a new command or is still executing the previous one. If you are executing a screen memory read, you read the screen data from this port also.

Reading port-1 provides some more status, including whether the most recent OP code or parameter has been accepted by the 8085 yet or not, and whether there is output data in the data register (port-0) during a screen memory read operation.

When the board is first powered up, the 8085 is reset by on-board circuitry and it then sets the board's mode to high density graphics and clears the screen. My board did this when powered up, so I knew that the ROM, 8085, and video chips were functioning properly. Since the screen was an even shade of grey, it seemed likely that the refresh memory ws working properly also, so I was encouraged. Unfortunately, I could not get the proper status bits to show up on port-0, so I suspected that the Biotech ROM was in error; perhaps if I had known the company better, I would not have wasted my time in this particular direction.

At times the video board would spontaneously give me a "light show" for minutes on end, while at other times it would do nothing at all no matter what I did with the ports. After setting up some software loops to write to ports 0 and 1 repeatedly, it became clear on the 'scope that not all of the port writes were setting through to the video board. I isolated it to a flip-flop which was not functioning properly, and the board worked as soon as a new 7474 was installed.

I now came to another problem area; one which I had not expected. A diagonal line drawn across the screen was jagged and uneven. After some experimenting, it turned out that all of the odd pixels (picture elements, or dots) were less wide than the even pixels. Again, I first thought that it was a software error (in the line drawing algorithm) on Biotech's part. I tried to write a bit pattern into the refresh memory to see if the problem would also appear with this simple input. When it continued to produce pixels of uneven size, I assumed that it was a hardware problem. A call to Biotech revealed that sometimes the duty cycle of the color burst oscillator was not exactly 50%, which caused some pixels to be clocked onto the screen more rapidly than others. The former pixels would then be less wide than the latter. Putting a 10K-ohm resistor from pin 3 of the 1372 to ground fixed than problem.

During the debugging phases, the CGS-808 had often thought that it needed a reset because the sequence of OP codes and parameters was not what it had expected (either because of a hardawre problem or my misunderstanding of the way the board worked). Unfortunately, I was debugging the board using my UCSD PASCAL Monitor program (which will appear in the next issue of this magazine), and pressing reset caused me to have to reboot. It seemed that if the software was aware that a board reset was required (see the status bits described for port-0 above), it should also be able to reset the board in software. What was required was a way to generate a low (0 volts) signal on the board on command from the central processor.

I looked for I/O ports from which I could steal a signal, but they are all 'smart' ports which hold the data inside themselves until strobed by the 8085, and so could not be used. There were, however, strobe signals for each of the two ports which the CGS-808 had reserved but not used, and these were normally high but went low when the port was addressed. I connected the strobe for writing to port 2 (U48 pin 5) of the processor's reset circuitry, but it did not work. I then removed the 3.3 microfarad reset capacitor (C22) so that my short strobe would not have to fight the capacitor, and the trick worked. I now can reset the board by writing to port-2.

I did not want to lose the power-on reset and the ability to clear the board from the front panel reset switch, so I installed a diode between the S-100 bus's reset line and the 8085's reset circuitry so that a board reset would not cause a system reset, but a system reset would cause a board reset. Addition of a small capacitor should also restore the power-on reset.

The board's speed is adequate but not high enough to allow something like real time 3-D rotations of complex objects. I asked Biotech if there was a way to increase the speed of the board and they said that if I had purchased fast enough RAMs (about 300nsec) I could replace an RC network which drove the 8085's clock with a 6 MHz or 7MHz crystal. This would increase the board's speed by about a factor of three. In addition, if I was willing to have snow on the screen during processing, I could take pin 1 of U26 (a flip-flop reset pin) and hold it always low, defeating the bus arbitration circuitry which prevents the 8085 and the 6847 from competing for the refresh memory bus. I have not done this yet, but they indicate that approximately a 40% increase in speed could be expected from this modification. If you make this mod, be certain not to ground the other end of the run to U26 pin-1, as it is an output pin on the 6847. Cut the lead before grounding U26 pin-1.

The 6847 has an annoying feature which is that it places a border around the graphics area of the screen. In high density mode, that border can only be white, making for some synch problems on my monitor and making it impossible to see lines drawn along the border in white. There can also be problems with monitor bloom

TARBELL DISK CONTROLLER MODS For Ithaca Audio Z80 CPU

by
George Lyons
280 Henderson St.
Jersey City, N.J.

To use the Tarbell floppy disk contoller in my system some modifications were required for it to work at all and others were desirable for better performance. The attached schematic shows the changes should anyone wish to use them. The diagram is purely schematic, so some circuit disconnections shown actually require more than one trace cut.

1. PWAIT compatibility--With the Ithaca Audio or similar Z80 cpus, PWAIT is not generated exactly like the 8080 and is mostly useless; it cannont be fed back to PRDY as on the Tarbell board, so U57 (diagram lower right) is disconnected and reused below.

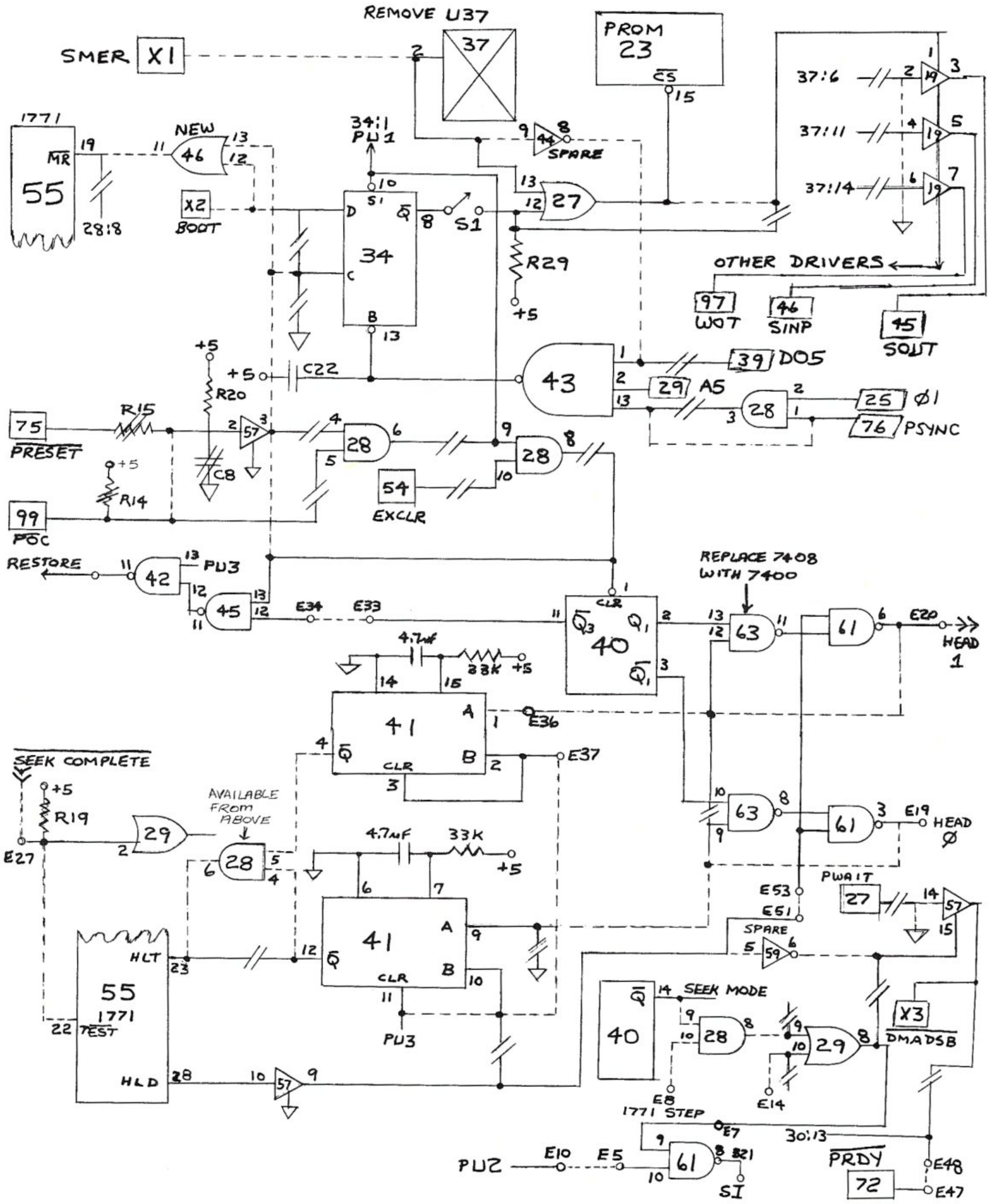
2. Bootstrap PROM in Z80 systems--Z80 cpu cards which do not multiplex status on the data lines like the 8080 cannot use the standard Tarbell bootstrap PROM. One way it can be used is to put a new bus signal $\overline{SMER}$ on an unused bus line "X1" (top left). This signal is the standard line 47, SMER, inverted and not disabled by STATDSB. It allows enabling the PROM and disabling system memory only during memory read cycles, even though SMER itself becomes disabled. On the Ithaca Audio cpu $\overline{SMER}$ can be made by connecting IC18 pin 14 to the bus with driver IC39 pins 9-10, in which case it is conveniently disabled in the HOLD state even though insensitive to STATDSB, to prevent any unwanted reset of the boot flip-flop U34 from a HOLD. Status signal M1 at U43 pin 1 is replaced by reinverted $\overline{SMER}$. PSYNC itself, instead of gated by 01 is used at U43 pin 1 for Z80 cpus synthesizing PSYNC not precisely aligned with 01.

An alternative method eliminating the extra bus signal $\overline{SMER}$ is shown in fig. 2. A 7474 flip-flop is added at socket U58 and used to latch SMEMR on the rising edge of PSYNC, and to reset on the falling edge of PDBIN, "remembering" SMER while it is disabled by STATDSB. $\overline{PHLD}$ gates PSYNC at U28 to clear U58 during DMA operations.

3. Separate Reset and Bootstrap controls--To prevent engaging the bootstrap PROM when hitting RESET, Tarbell provides a dip swithch on the controller. To more conveniently do so, from a keyboard, the boot flip-flop U34 is rewired as shown with input BOOT placed on another unused bus line "X2" (high to engage the PROM). This also allows use of a ROM monitor with its own jump-on-reset circuit (such as on the Ithaca Audio cpu) by using X2 to disable the ROM's jump circuit when high. A panel switch or keyboard key can then select which process is desired upon cpu reset. When a keyboard is interfaced through a MERLIN vide board inverting the active-low EDIT key is convenient.

4. Reset logic at 4mhz--The original circuit pulses the Master Reset pin of the 1771 when starting the bootstrap. I find my 4mhz cpu tends to execute the first bootstrap instruction, IN WAIT, before INTRQ has reset, causing Lost Data errors. This is partly due to MR being intiated on the trailing, rather than the leading edge of $\overline{POC}$. Or gate U46 was therefore added to prevent Master Reset on booting and the reset circuit was simplified to use only $\overline{POC}$ instead of Preset and ExtClr as well. U57 (upper left) provides a buffered $\overline{POC}$ to all functions. Master Reset could also be fixed with a one-shot but the spare one-shots are used below. Two strikes of the RESET key, one with and one without Master Reset are needed to boot in a 4mhz system.

5. Merlin Video Compatibility-- This video board, no longer manufactured, has no on board display memory but uses DMA to refresh the screen--and must be deactivatted during disk transfers. To avoid doing so with software, the 1771 HLD signal is placed on a third spare bus line "X3" (lower right) and jumpered to the PRIORITY-IN pad on the Merlin. The Merlin Priority-In circuit may also be modified to disable immediately instead of waiting until completion of any DMA in progress.


REMOVE U37
SMER
X1
37
PROM
23
ES
1771
MR
55
NEW
46
28:8
X2
BOOT
34:1
PU1
34
S1
27
R29
+5
37:6
37:11
37:14
OTHER DRIVERS
97
WOT
46
SINP
45
SOUT
SPARE
C22
43
39
DO5
29
A5
25
Ø1
76
PSYNC
R15
R20
75
PRESET
C8
R14
99
POC
54
EXCLR
RESTORE
42
PU3
45
E34
E33
REPLACE 7408
WITH 7400
CLR
40
63
61
E20
HEAD
1
4.7uF
33K
41
E36
E37
SEEK COMPLETE
R19
E27
AVAILABLE
FROM
ABOVE
E19
HEAD
Ø
PWAIT
E53
E51
HLT
55
1771
TEST
HLD
SEEK MODE
X3
DMADSB
E8
1771 STEP
E14
E7
30:13
PRDY
72
E48
E47
PU2
E10
E5
SI

- FIGURE 1 -

6. Automatic Head-load Timing--The 1771 inserts a 10ms delay in commands requesting the head to load so that for optimal efficiency the head load command bit must be omitted under software control when the head is already loaded from previous use. This 10ms delay is actually unnecessary because the head settling delay is really provided by the one-shot U41 and the HLT pin on the 1771, and is around 50ms. Use of separate head-loading and non loading commands can be eliminated by activating the 1771 TEST pin; it eliminates the 10ms delay so all commands can be of the head-loading type. However, TEST also speeds up head positioner step pulses to an unusable rate. This is cured by applying SEEKCOMPLETE to the TEST pin (lower left).

7. Dual Head Load--When switching back and forth between two drives the Tarbell board sends the head load signal to only one drive at a time. By rewiring U61 and U63 as shown and replacing U63 by a NAND gate, both heads can remain down simultaneously. Neither head will load until it is selected, but once loaded all remain so until the 1771 HLD signal goes off. The spare one-shot U41 is added to the original HLT circuit to provide settling delay when engaging a second head. Note the reversal of the drive connections to pads E20, E19 resulting from the NAND gate replacement U63.

8. PerSci compatibility--PerSci drives with voice-coil head positioner can be operated in a "fast seek" mode which does not use the 1771 STEP signal but software generated high speed pulses from pad E14 instead. The Tarbell board provides for permanently jumpering the controller in this fast-seek mode or in normal mode. Since the software for positioning the head is different in these two configurations modification was done to allow both types of software to use the board without changing jumpers. The PerSci drive accepts both types of STEP signal in any case. The seek mode is selected by bit Q4 of the function register U40, which also controls the source of the WAIT state generated when reading the WAIT port (FC). In fast seek mode SEEK COMPLETE generates the WAIT state for detecting completion of a fast mult-track seek, while in slow-seek mode the 1771 signals completion of seeks with the usual INTRQ signal. The circuit shown lower right allows either pad E14 or 1771 STEP to drive the step line in nomral mode and disconnects the 1771 STEP signal in fast mode (necessary because a STEP pulse is generated in using the 1771 to set the DIRC signal level). Gates U28 and U29 shown are reused from other mods above where they were fed; alternatively spare AND gate at U36 and the added OR gate at U46 could be used.

Conclusion--Those are all the modifications. One more circuit might be added if the controller were entirely redesigned, but requires half a dozen gates. This would be to provide WAIT stat synchronization on the DATA port rather than on the WAIT port, permitting Z80 block I/O instructions to do disk transfers. It is messy because the 1771 is configured to use the DRQ and INTRQ signals as interrupts, not READY signals, and is not shown.

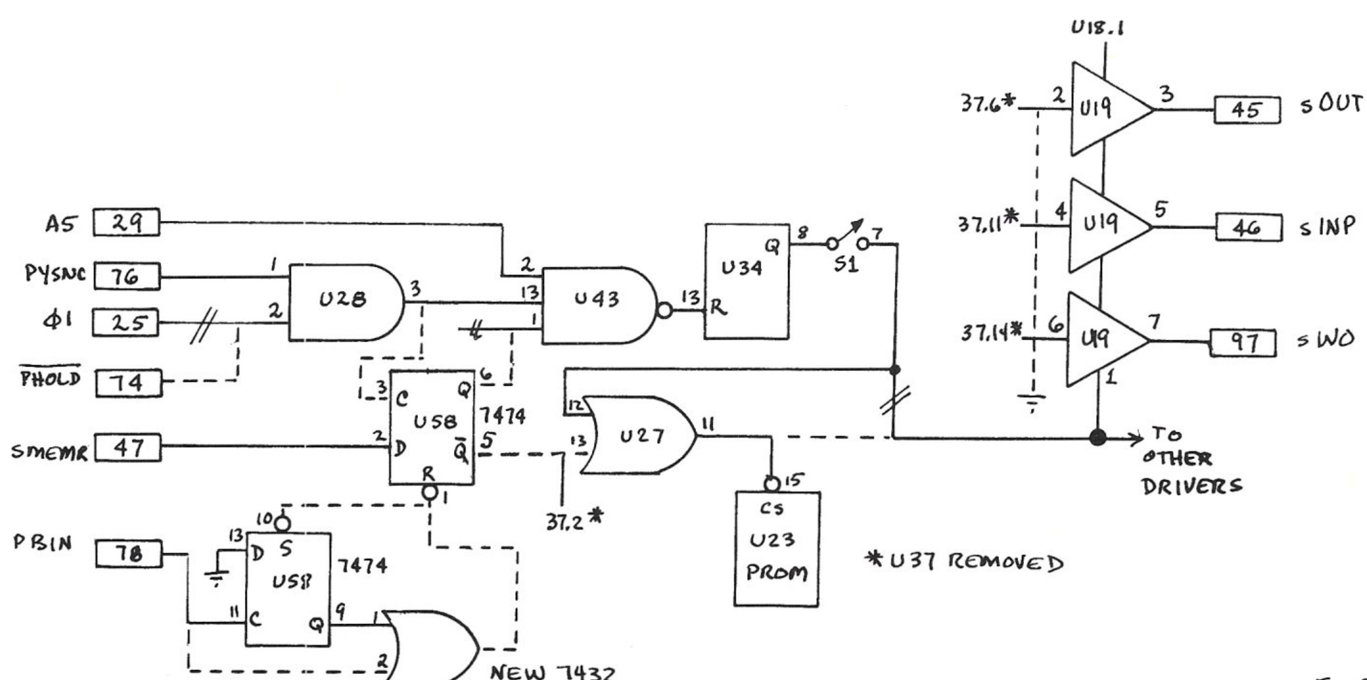


Fig 2

AN IMPROVED CP/M* BIOS FOR TARBELL DISK CONTROLLER

by
Martin Nichols
100 Guy Street
Dover, N.J. 07801

The following program is an improved BIOS (Basic Input/Output System) for CP/M when used with a Tarbell single density disk controller. It offers several improvements over the original BIOS provided with the Tarbell Disk controller. For example, it will support 4 disk drives, compared to 2 previously and provides faster disk I/O operations. Further, it includes a driver for VDM-type CRT displays.

The source code is also extremely well documented allowing for easy modification by the user. Those users wishing the source code on an 8" disk can obtain it by writing to the author. The charge for the disk is $10.

* Reg Tmk Digital Research

```
;************************************************************
;
;BASIC INPUT/OUTPUT OPERATING SYSTEM
;FOR USE WITH THE TARBELL CONTROLLER
;
;
;THIS MODULE CONTAINS ALL THE INPUT/OUTPUT
;ROUTINES FOR THE CP/M SYSTEM, INCLUDING
;THE DISK ROUTINES.
;
;THIS VERSION SUPPORTS THE FOUR DRIVE SYSTEM
;OF CP/M 1.4. IT CAN ALSO BE USED WITH 1.3.
;THE ONLY CHANGE NEEDED IF YOU ARE USING 1.3 IS
;TO CHANGE THE BDOS EQUATE FROM "CBASE+3106H"
;TO "CBASE+3206H".
;
;******* THIS BIOS REQUIRES 1K MORE THAN THE 512 BYTES
;ALLOWED BY DIGITAL RESEARCH.  THEREFORE, IF YOU WANT
;TO RUN THIS BIOS YOU WILL HAVE TO GENERATE A CPM THAT
;IS 1K LESS THAN THE AMOUNT OF MEMORY THAT YOU WANT IT
;TO RUN IN (I.E. 23 TO FIT IN 24K, 31 FOR 32K, ETC).
;MSIZE SHOULD AGREE TO THE NUMBER OF K YOU SPECIFIED
;TO MOVCPM OR RELOC (23 FOR 24K SYSTEM).  USE THE
;BIOS FORMULA GIVEN IN DIGITAL RESEARCH MANUAL.
;
;IF YOU WANT VDM AS STD OUTPUT-"PTVDM" SHOULD BE TRUE.
;IF YOU WANT STANDARD OUTPUT ROUTINE-"PTVDM" SHOULD BE FALSE.
;
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;
; "MYSYS" MUST BE FALSE <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;
;
;
;
;THE DISK DRIVER ROUTINES IN THIS BIOS ARE DEPENDENT
;ON EACH OTHER FOR CERTAIN INFORMATION AND TIMINGS.
;MODIFICATIONS TO THE DISK DRIVER ROUTINES SHOULD
;ONLY BE DONE AFTER CAREFUL STUDY.
;
;THIS VERSION CONTAINS SUPPORT FOR FAST READING AND
;WRITING ON A NON-INTERLEAVED BASIS. THAT IS, IT CAN
;READ OR WRITE A COMPLETE TRACK IN ONE REVOLUTION OF
;THE DISK IF THE READS OR WRITES ARE ISSUED IN
;SEQUENTIAL ORDER.
;
;IF YOU HAVE ONLY ONE DISK CHANGE "ONEDSK" TO TRUE.
;THE I/O SYSTEM WILL THEN SIMULATE A MULTIPLE DRIVE SYSTEM.
;
;ERROR MESSAGES ARE PRINTED IN THE FOLLOWING FORMAT:
;
;XAAA TNN SNN Z
;
;
;       WHERE X = (R)READ,(W)WRITE
;       AND AAA = NRY - NOT READY - BIT 7
;                 WPR - WRITE PROTECT - BIT 6
;                 WFT - WRITE FAULT - BIT 5
;                 RNF - RECORD NOT FOUND - BIT 4
;                 CRC - CRC ERROR - BIT 3
;                 LDA - LOST DATA - BIT 2
;                 DRQ - DATA REQUEST - BIT 1
;                 BSY - BUSY - BIT 0
;           TNN = TRACK WHICH CONTAINS ERROR (DECIMAL)
;           SNN = SECTOR WHICH CONTAINS ERROR (DECIMAL)
;             Z = DRIVE ON WHICH ERROR OCCURED
;       THE BIT POSITIONS ARE GIVEN SO THAT THE 1771 MANUAL
;       COULD BE CONSULTED FOR FURTHER EXPLANATION.
;
;---WRITTEN BY M. D. NICHOLS---
;
;************************************************************
```

```
;
;FREQUENTLY CHANGED PARAMETERS
;
TRUE     EQU     0FFFFH           ;USED IN IF STATEMENTS
FALSE    EQU     NOT TRUE         ;USED IN IF STATEMENTS
PTVDM    EQU     TRUE             ;VDM FOR CONOT?
ONEDSK   EQU     FALSE            ;USE IF YOU ONLY HAVE 1 DRIVE
TEST     EQU     TRUE             ;TESTING AIDS
MSIZE    EQU     47               ;MEMORY SIZE
MODIF    EQU     'G'              ;MODIF LEVEL OF DRIVER
SUBMOD   EQU     '4'              ;SUB MODIFICATION LEVEL
MYSYS    EQU     TRUE             ;SPECIAL STUFF FOR MY SYSTEM
;
DBASE    EQU     0F8H             ;BASE FOR CONTROLLER PORTS
DCOM     EQU     DBASE            ;DISK COMMAND PORT
DSTAT    EQU     DBASE            ;DISK STATUS PORT
DTRK     EQU     DBASE+1          ;DISK TRACK PORT
DSECT    EQU     DBASE+2          ;DISK SECTOR PORT
DDATA    EQU     DBASE+3          ;DISK DATA PORT
DWAIT    EQU     DBASE+4          ;DISK WAIT PORT
DCONT    EQU     DBASE+4          ;DISK CONTROL PORT
         ORG     MSIZE*1024-512   ;THIS SETS ADDR OF CBIOS
CBASE    EQU     (MSIZE-16)*1024  ;TEMPORARY CALCULATION
CCP      EQU     CBASE+2900H      ;START OF CP/M SYSTEM
BDOS     EQU     CBASE+3106H      ;START OF BDOS FOR 1.4
CPML     EQU     $-CCP            ;LENGTH OF CP/M W/O CBIOS
NSECTS   EQU     CPML/128         ;# SECT CP/M EXCL. CBIOS
RTCNT    EQU     10               ;RETRY COUNT ON ERRORS
;
;
         JMP     BOOT             ;FROM COLD START LOADER
XWBOOT   JMP     WBOOT            ;FROM WARM START
XCONST   JMP     CONST            ;CHECK CONSOLE STAT
XCONIN   JMP     CONIN            ;GET CONSOLE CHARACTER
XCONOT   JMP     CONOT            ;WRITE CONSOLE CHARACTER
XLIST    JMP     LIST             ;WRITE CHAR ON LIST DEV
XPUNCH   JMP     PUNCH            ;PUNCH CHAR ON PAPER TAPE
XRDR     JMP     READER           ;READ CHAR FROM PAPER TAPE
XHOME    JMP     HOME             ;RESTORE HEAD TO TRK 0
XDSKSEL  JMP     DSKSEL           ;SELECT DISK DRIVE
XSETTRK  JMP     SETTRK           ;SET TRACK NUMBER
XSETSEC  JMP     SETSEC           ;SET SECTOR NUMBER
XSETDMA  JMP     SETDMA           ;SET DMA ADDRESS
XREAD    JMP     READ             ;READ SELECTED SECTOR
XWRITE   JMP     WRITE            ;WRITE SELCECTED SECTOR
;
;ENTER HERE FROM COLD START LOADER
;GIVE SIGN ON MESSAGE
;
BOOT:
         LXI     SP,80H           ;SET STACK POINTER
         XRA     A                ;CLEAR A
         STA     4                ;FORCE DISK A
;
         IF      MYSYS OR PTVDM
         OUT     0C8H             ;RESET VDM
         MVI     C,0CH            ;SET UP FF TO CLEAR SCREEN
         CALL    XCONOT           ;SEND IT OUT
         ENDIF
;
         LXI     H,SMSG           ;PRINT OPENING MESSAGE
         CALL    PMSG             ;DO PRINTOUT
         JMP     GOCPM            ;NOW DO HOUSEKEEPING
;
;WARM BOOT ROUTINE--ACTIVATED WHEN YOU CONTROL C
;
WBOOT:
         LXI     SP,80H           ;SET STACK POINTER
WBOOT1:
         MVI     C,0              ;USED BY DSKSEL AND SETTRK
         CALL    XDSKSEL          ;SELECT DISK 0
         MVI     D,NSECTS         ;# OF SECTORS IN D
         MVI     B,2              ;START WITH SECTOR 2
         LXI     H,CCP            ;GET STARTING ADDR
WBOOT2:
         CALL    SETTRK           ;SELECT TRACK (C)
         PUSH    B                ;SAVE BC
         MOV     C,B              ;PUT SECTOR IN C
         CALL    XSETSEC          ;SET IT UP
         MOV     B,H              ;GET READY FOR SET DMA
         MOV     C,L              ;
         CALL    XSETDMA          ;SET IT UP
         POP     B                ;RESTORE BC
         CALL    XREAD            ;READ RECORD
         JNZ     WBOOTX           ;ERROR ON READ
         DCR     D                ;# OF SECTORS TO READ - 1
         JZ      GOCPM            ;DONE--GO TO CPM
         INR     B                ;POINT TO NEXT SECTOR
         MOV     A,B              ;MOVE IT TO A
         CPI     27               ;END OF TRACK
         JC      WBOOT2           ;NO--CONTINUE READING
         INR     C                ;TRACK NOW EQUAL 1
         MOV     B,C              ;SET SECTOR BACK TO 1
         JMP     WBOOT2           ;READ NEXT TRACK
WBOOTX:
         LXI     H,BTMSG          ;GET ADDR OF ERROR MSG
         CALL    PMSG             ;PRINT IT
         CALL    XCONIN           ;WAIT FOR KB ENTRY
         JMP     WBOOT1           ;DO WARM BOOT AGAIN
;
;THIS ROUTINE IS THE EXIT TO CP/M SYSTEM
;
GOCPM:
         MVI     A,0C3H           ;PUT JUMP TO WARM BOOT
         STA     0                ;AT ADDR ZERO
         LXI     H,XWBOOT         ;GET ADDR OF WARM BOOT ENTRY
         SHLD    1                ;FINISH THE JUMP INSTUCTION
         STA     5                ;PUT JUMP TO BDOS AT 5
         LXI     H,BDOS           ;GET ADDR OF BDOS ENTRY
         SHLD    6                ;FINISH JUMP INSTRUCTION
         STA     38H              ;SET UP INTERRUPT TRAP
         LXI     H,TRAP           ;
         SHLD    39H              ;ARM IT
         LXI     H,80H            ;SET DEFAULT DMA ADDR
         SHLD    DMAADD           ;AND STORE IT
         LDA     4                ;GET PREV DRIVE #
         MOV     C,A              ;SHOULD BE IN C
;
         IF      TEST             ;
         XRA     A                ;CLEAR A
         OUT     0FFH             ;SHOW IN THE FP LIGHTS
         ENDIF
;
         JMP     CCP              ;GO TO CP/M
;
;THIS ROUTINE IS A NULL INTERRUPT TRAP
;FOR NOW, JUST RE-ENABLE INTERRUPTS
;
TRAP:
         DB      0                ;LEAVE ROOM FOR JUMP OR CALL
         DB      0                ;
         DB      0                ;
         EI                       ;TURN INTERRUPTS BACK ON
         RET                      ;GO BACK FROM WHENCE YOU CAME
;
;SELECT DISK GIVEN BY REG C
;
DSKSEL:
         MOV     A,C              ;PUT NEW DISK IN A
         STA     NXTDSK           ;SAVE IT FOR REAL RTN
         RET                      ;
;
;SELECT DISK STORED IN NXTDSK
;
SELDSK:
         PUSH    H                ;
         PUSH    D                ;SAVE REGS
         PUSH    B                ;
         LDA     NXTDSK           ;GET NEW DISK #
         MOV     C,A              ;
         ANI     3                ;LOOK AT 3 LSB'S
         LXI     H,DISKNO         ;GET ADDR OF DISKNO
         CMP     M                ;NEW=OLD
         JZ      SELXIT           ;IF SO,RETURN
;
         IF      ONEDSK
;
         PUSH    B                ;SAVE REGS
         PUSH    A                ;SAVE REGS
         LXI     H,MNTMSG         ;PRINT 'MOUNT' MSG
         CALL    PMSG             ;TELL THEM
         POP     A                ;RETRIEVE REG A
```

```
        STA     DISKNO          ;STORE IT FOR LATER
        ADI     'A'             ;MAKE IT ASCII LETTER
        MOV     C,A             ;PUT IN C FOR CONOT
        CALL    XCONOT          ;PRINT IT
        CALL    XCONIN          ;WAIT FOR GO AHEAD
        POP     B               ;RESTORE BC
        XRA     A               ;CLEAR A
        JMP     SELXIT          ;EXIT SELDSK

        ENDIF

        MOV     E,M             ;PUT OLD DISK # IN DE
        MVI     D,0             ;CLEAR D FOR DAD
        LXI     H,TRKTB         ;GET ADDR OF TRACK TABLE
        DAD     D               ;ADD DISK # TO ADDR
        IN      DTRK            ;GET TRACK FROM OLD DRIVE
        MOV     M,A             ;STORE IT IN TRACK TABLE
        MOV     E,C             ;GET NEW DRIVE #
        LXI     H,TRKTB         ;GET ADDR OF TRK TABLE
        DAD     D               ;GET HEAD LOC ON NEW DRIVE
        MOV     A,M             ;AND PUT IN REG A
        STA     HOLD            ;SAVE FOR SEEK ROUTINE
        OUT     DTRK            ;ADJUST 1771
        MOV     A,C             ;GET DISK #
        STA     DISKNO          ;STORE IT FOR LATER USE
        CMA                     ;INVERT BIT FOR LATCH
        ADD     A               ;PUT BITS 0-1 AT 4-5
        ADD     A               ;
        ADD     A               ;
        ADD     A               ;
        ORI     2               ;MAKE LATCH COMMAND
        OUT     DCONT           ;SET LATCH WITH CODE
SELDEL:
        IN      DSTAT           ;UNLOAD HEAD BECAUSE 1771 DOES
        ANI     20H             ;NOT RECOGNIZE DRIVE SWITCH
        JNZ     SELDEL          ;
        XRA     A               ;SAY EVERYTHING OK
SELXIT:
        POP     B               ;RESTORE REGS
        POP     D               ;RESTORE REGS
        POP     H               ;RESTORE REGS
        RET                     ;
;
;SET THE TRACK GIVEN IN REG C
;
SETTRK:
        MOV     A,C             ;TRK WAS IN REG C
        STA     TRK             ;PUT IT WHERE IT CAN BE FOUND
        RET                     ;
;
;SET DISK SECTOR NUMBER
;
SETSEC:
        MOV     A,C             ;GET SECTOR NUMBER
        STA     SECT            ;PUT AT SECTOR # ADDR
        RET                     ;
;
;SET DISK DMA ADDRESS
;
SETDMA:
        MOV     H,B             ;MOV BC TO HL
        MOV     L,C             ;
        SHLD    DMAADD          ;SAVE IT
        RET                     ;
;
;READ A SECTOR AT SECT, SEEK THE NEEDED TRACK
;USE STARTING ADDRESS AT DMAADD
;
READ:
        CALL    SEEK            ;MOVE HEAD TO TRACK
        MVI     A,RTCNT         ;GET RETRY CNT
RRETRY:
        STA     ERCNT           ;SAVE ERROR CNT
        LDA     SECT            ;GET SECTOR NUMBER
        LHLD    DMAADD          ;GET STARTING ADDRESS
READ1:
        OUT     DSECT           ;SET SECTOR IN 1771
        CALL    RDYCK           ;SEE IF DRIVE READY
        DI                      ;DO NOT ALLOW INTERRUPTS
        IN      DSTAT           ;GET STATUS
        ANI     20H             ;CHECK IF HEAD LOADED
        MVI     A,88H           ;SETUP READ W/O HEAD LOAD
        JNZ     READE           ;IF LOADED - THEN DO IT
        ORI     4               ;ELSE FORCE HEAD LOAD
READE:
        OUT     DCOM            ;SEND COMMAND TO 1771
RLOOP:
        CALL    FREAD           ;READ A SECTOR
RDDONE:
        EI                      ;ALLOW INTERRUPTS AGAIN
        IN      DSTAT           ;READ DISK STATUS
        ANI     9DH             ;LOOK AT ERROR BITS
        RZ                      ;RETURN IF NONE
CHECK:
        STA     ERRS            ;SAVE ERROR BYTE

;
        IF      TEST            ;
        CMA                     ;FP INVERTED LOGIC
        OUT     0FFH            ;IN THE LIMELIGHT
        ENDIF                   ;
;
        CALL    ERCHK           ;CHECK FOR SEEK ERROR
        LDA     ERCNT           ;GET ERROR CNT
        DCR     A               ;DECREMENT COUNT
        JNZ     RRETRY          ;TRY TO READ AGAIN
        MVI     A,'R'           ;SHOW READ ERROR
        JMP     ERRMSG          ;TELL SOMEONE
FREAD:
        IN      DWAIT           ;WAIT FOR DRQ OR INTRQ
        ORA     A               ;SET FLAGS
        RP                      ;DONE IF INTRQ
        IN      DDATA           ;READ A BYTE FROM DISK
        MOV     M,A             ;PUT BYTE IN MEMORY
        INX     H               ;INCR MEMORY POINTER
        JMP     FREAD           ;KEEP READING
;
;WRITE THE SECTOR AT SECT--LOAD HEAD FIRST
;USE STARTING ADDRESS AT DMAADD
;
WRITE:
        CALL    SEEK            ;GET ON RIGHT TRACK
        MVI     A,RTCNT         ;GET RETRY COUNT
WRETRY:
        STA     ERCNT           ;SAVE ERROR CNT
        LDA     SECT            ;GET SECTOR #
        LHLD    DMAADD          ;GET STARTING ADDR
WRITE1:
        OUT     DSECT           ;SET SECT INTO 1771
        CALL    RDYCK           ;SEE IF DRIVE READY
        DI                      ;DO NOT ALLOW INTERRUPTS
        IN      DSTAT           ;GET DISK STATUS
        ANI     20H             ;CHECK FOR HEAD LOADED
        MVI     A,0A8H          ;SETUP WRITE W/O HEAD LOAD
        JNZ     WRITEN          ;IF LOADED THEN DO IT
        ORI     4               ;FORCE WRITE WITH HEAD LOAD
WRITEN:
        OUT     DCOM            ;ISSUE COMMAND
WLOOP:
        CALL    FWRITE          ;WRITE A SECTOR
WDONE:
        EI                      ;ALLOW INTERRUPTS AGAIN
        IN      DSTAT           ;READ DISK STATUS
        ANI     0FDH            ;MASK NON-ERROR BITS
        RZ                      ;RETURN IF NO ERRORS
        STA     ERRS            ;SAVE ERROR FLAGS

        IF      TEST            ;
        CMA                     ;INVERT THEM
        OUT     0FFH            ;PUT THEM ON FP LEDS
        ENDIF                   ;

        CALL    ERCHK           ;CHK/CORRECT SEEK ERR
        LDA     ERCNT           ;GET ERROR CNT
        DCR     A               ;DECREMENT COUNT
        JNZ     WRETRY          ;TRY WRITE AGAIN
        MVI     A,'W'           ;SHOW WRITE ERROR
        JMP     ERRMSG          ;DO ERROR MESSAGES
FWRITE:
        IN      DWAIT           ;WAIT FOR READY
        ORA     A               ;SET FLAGS
        RP                      ;GET OUT WHEN DONE
```

```
        MOV     A,M             ;GET BYTE FROM MEMORY
        OUT     DDATA           ;WRITE ON DISK
        INX     H               ;POINT TO NEXT BYTE
        JMP     FWRITE          ;KEEP WRITING
;
;
;READ OR WRITE ERROR DETECTED--HANDLE NO REC FOUND
;CONDITION ELSE NORMAL RETRY LOOP
;
ERCHK:
        LDA     ERRS            ;GET ERROR BYTE
        ANI     10H             ;MASK FOR NRF
        RZ                      ;NOT NRF FAULT
;
;CHECK TO SEE IF ON CORRECT TRACK-
;CORRECT IF NECESSARY
;
CHKSK:
        MVI     A,0C4H          ;SET UP READ ADDR
        OUT     DCOM            ;COMMAND TO 1771
        DI                      ;DO NOT ALLOW INTERRUPTS
        IN      DWAIT           ;WAIT FOR 1ST DRQ (TRK)
        IN      DDATA           ;READ THE TRACK ADDR
        EI                      ;ALLOW INTERRUPTS AGAIN
        PUSH    PSW             ;SAVE TRACK
        CALL    SWAIT           ;WAIT FOR OPERATION TO FINISH
        POP     PSW             ;GET TRACK BACK
        STA     HOLD            ;USE IT TO SET UP SEEK RTN
        OUT     DTRK            ;UPDATE TRACK REGISTER
;
;TRACK DESIRED HAS ALREADY BEEN STORED BY SETTRK
;
SEEK:
        CALL    SELDSK          ;WILL DO NOTHING IF FROM CHKSK
        LDA     TRK             ;GET WHERE WE ARE GOING TO
        PUSH    B               ;SAVE BC
        MOV     B,A             ;SAVE TRK IN B
        LDA     HOLD            ;GET PREV TRACK
        CMP     B               ;COMPARE TO CURRENT
        JZ      FORINT          ;SAME SO FORCE INTERRUPT
        MOV     A,B             ;PUT CURRENT IN A
        STA     HOLD            ;SAVE FOR NEXT TIME
        POP     B               ;RESTORE BC
        CPI     0               ;SEE IF TRK = 0

        JZ      HOMRTN          ;DO HOME ROUTINE INSTEAD
        OUT     DDATA           ;GIVE DESIRED TRK TO 1771
        CALL    SWAIT           ;WAIT TILL NOT BUSY
        MVI     A,1AH           ;SEEK-10MS-
        CALL    SCMND           ;ISSUE COMMAND
        CALL    SLOOP           ;GIVE TIME FOR HEAD TO SETTLE
        RET                     ;
;
SCMND:
        OUT     DCOM            ;ISSUE COMMAND
SWAIT:
        CALL    ZIP             ;WE NEED AT LEAST 12 US
        CALL    ZIP             ;AGAIN, FOR 4 MHZ
SBUSY:
        IN      DSTAT           ;WAIT FOR NOT BUSY, IE INTRQ
        RRC                     ;SET CARRY IF BUSY
        JC      SBUSY           ;STILL BUSY
ZIP:
        RET                     ;INTRQ RESET BY READING STATUS
;
FORINT:
        MVI     A,0D0H          ;CLEAR ANY PENDING COMMAND
        OUT     DCOM            ;AND FORCE TYPE 1 STATUS
        POP     B               ;CLEAN UP
        RET                     ;GO BACK
;
SLOOP:
        PUSH    H               ;GIVE TIME FOR HEAD TO SETTLE
        LXI     H,9*256         ;NEED ABOUT 10 MS
SLOOP1:
        DCR     L               ;
        JNZ     SLOOP1          ;
        DCR     H               ;
        JNZ     SLOOP1          ;
        POP     H               ;
        RET                     ;
```

```
;HOME ROUTINE-SET TRK TO ZERO AND SEEK
;WILL PICK UP DURING READ OR WRITE
;
HOME:
;
        XRA     A               ;CLEAR A
        STA     TRK             ;PUT WHERE IT CAN BE FOUND
        RET                     ;
;
;THIS ROUTINE ONLY CALLED FROM SEEK
;TO PERFORM ACTUAL HOME
;
HOMRTN:
        MVI     A,0D0H          ;RESET ANY PENDING COMMAND
        OUT     DCOM            ;ISSUE COMMAND
        CALL    SWAIT           ;WAIT FOR NOT BUSY
        MVI     A,0AH           ;10 MS SEEK RATE--HOME
        CALL    SCMND           ;ISSUE COMMAND-WAIT FOR INTRQ

        CALL    SLOOP           ;FOR HEAD SETTLING
        RET                     ;GO BACK
;
;CHECK FOR DRIVE READY-IF NOT TELL OPERATOR
;AND WAIT FOR CR
;
RDYCK:
        IN      DSTAT           ;GET DISK STATUS
        ANI     80H             ;MASK FOR READY
        RZ                      ;OK WE ARE READY
        PUSH    H               ;SAVE REGS
        PUSH    B               ;SAVE REGS
RDY1:
        LXI     H,NRDYMS        ;POINT TO MESSAGE
        CALL    PMSG            ;PRINT IT OUT
        LDA     DISKNO          ;GET CURRENT DRIVE
        ADI     'A'             ;CHANGE TO ASCII
        MOV     C,A             ;SET UP TO PRINT
        CALL    XCONOT          ;PRINT IT
        CALL    XCONIN          ;GET KEYBD CHAR
        LXI     H,CRLF          ;SET UP CR AND LF
        CALL    PMSG            ;PRINT IT
        POP     B               ;RESTORE REGS
        POP     H               ;RESTORE REGS
        RET                     ;
;
;
;PRINT MESSAGE ROUTINE
;
PMSG:
        MOV     A,M             ;GET BYTE TO PRINT
        ORA     A               ;SEE IF BINARY ZERO
        RZ                      ;YES, WE ARE DONE
        MOV     C,A             ;PASS IT IN C
        CALL    XCONOT          ;PRINT A CHAR
        INX     H               ;POINT TO NEXT BYTE
        JMP     PMSG            ;STAY IN LOOP TILL DONE
;
;ERROR MESSAGE ROUTINES
;
ERRMSG:
        PUSH    H               ;SAVE HL
        PUSH    B               ;SAVE BC
        MOV     B,A             ;SAVE TYPE INDICATOR
        LXI     H,CRLF          ;DO CRLF
        CALL    PMSG            ;PRINT IT
        MOV     C,B             ;GET TYPE INDIC
        CALL    XCONOT          ;PRINT IT
        LDA     ERRS            ;GET ERROR BYTE
        LXI     H,TYPERR-3      ;POINT TO ERROR TABLE
LOCERR:
        INX     H               ;POINT TO NEXT ENTRY
        INX     H               ;IN ERROR TYPE TABLE
        INX     H               ;
        RLC                     ;SHIFT BIT INTO CARRY

        JNC     LOCERR          ;NOT IN ERROR-KEEP LOOKING
        MVI     B,3             ;SET B = 3
TYPRNT:
        MOV     C,M             ;GET FIRST LETTER
        CALL    XCONOT          ;PRINT IT
        INX     H               ;POIN T TO NEXT LETTER
```

```
        DCR     B               ;DECREMENT COUNTER
        JNZ     TYPRNT          ;KEEP PRINTING LETTER
        CALL    BLK             ;PRINT A BLANK
        MVI     C,'T'           ;PRINT A T FOR TRACK
        LDA     TRK             ;GET TRACK WE ARE ON
        CALL    ERRPRT          ;SHOW THEM LOCATION
        MVI     C,'S'           ;PRINT AN S FOR SECTOR
        LDA     SECT            ;GET SECT WE ARE ON
        CALL    ERRPRT          ;AND SHOW IT
        LDA     DISKNO          ;GET CURRENT DRIVE
        ADI     'A'             ;MAKE IT ASCII
        MOV     C,A             ;SET UP FOR PRINT
        CALL    XCONOT          ;PRINT IT
        POP     B               ;RESTORE B
        POP     H               ;RESTORE HL
        MVI     A,1             ;SIGNAL ERROR
        ORA     A               ;SET FLAGS
        RET                     ;GO HOME
ERRPRT:
        PUSH    PSW             ;SAVE NUMBER
        CALL    XCONOT          ;PRINT LETTER IN C
        POP     PSW             ;GET NUMBER BACK
DECPRT:
        MVI     C,'0'-1         ;SET UP C FOR 10'S DIGIT
DECPR1:
        INR     C               ;EXTRACT MS DIGIT
        SUI     10              ;BY REPETITIVE SUBTRACTION
        JP      DECPR1          ;
        ADI     10+'0'          ;RESTORE TO POSITIVE
        MOV     B,A             ;SAVE LS DIGIT
        CALL    XCONOT          ;LIST TENS DIGIT
        MOV     C,B             ;PUT UNITS DIGIT IN C
        CALL    XCONOT          ;PRINT A BLANK
BLK:
        MVI     C,' '           ;PRINT A BLANK
        JMP     XCONOT          ;PRINT IT AND RETURN
;
;CBIOS MESSAGES
;
SMSG:   DB      0DH,0AH,'MDN CP/M R1.V4 /'
        DB      MODIF,SUBMOD,'/',0DH,0AH,0
BTMSG:  DB      0DH,0AH,'BOOT FAILURE',0
CRLF:   DB      0DH,0AH,0
MNTMSG: DB      0DH,0AH,'MOUNT ',0
NRDYMS  DB      0DH,0AH,'NOT READY-DRIVE ',0
TYPERR: DB      'NRYWPRWFTRNFCRCLDADRQBSY'
;
;CHECK CONSOLE INPUT STATUS
;NOT READY (A)=0, READY (A)=FF

;
CONST:
;
        IF      MYSYS           ;
        IN      2               ;READ CONSOLE STATUS
        ANI     80H             ;LOOK AT BIT 7
        ENDIF                   ;
;
        IF      NOT MYSYS       ;
        IN      0               ;
        ANI     1               ;LOOK AT BIT 0
        ENDIF                   ;
;
        MVI     A,0             ;SET A=0
        RNZ                     ;RETURN WITH NOT READY
        CMA                     ;READY SO A=FF
        RET                     ;
;
;READ A CHARACTER FROM CONSOLE
;
CONIN:
;
        IF      MYSYS           ;
        IN      2               ;READ CONSOLE STAT
        ANI     80H             ;LOOK AT BIT 7
        JNZ     CONIN           ;KEEP WAITING
        IN      0               ;GET DATA BYTE
        ANI     7FH             ;TURN PARITY OFF
        RET                     ;
        ENDIF                   ;
;
        IF      NOT MYSYS       ;
        IN      0               ;READ CONSOLE STAT
        ANI     1               ;LOOK AT BIT 0
        JNZ     CONIN           ;NOT RDY-KEEP WAITING
        IN      1               ;READ DATA BYTE
        ANI     7FH             ;TURN OFF PARITY
        RET                     ;
        ENDIF                   ;
;
;
;WRITE CHARACTER IN (C) TO CONSOLE DEVICE
;
CONOT:
;
        IF      NOT PTVDM
        IN      0               ;GET STATUS
        ANI     80H             ;LOOK AT BIT 7
        JNZ     CONOT           ;NOT READY-KEEP WAITING
        MOV     A,C             ;CHAR IN C TO A
        OUT     1               ;PRINT IT
        RET                     ;
        ENDIF                   ;
;
        IF      PTVDM           ;
;
;
VDBASE  EQU     0CC00H          ;SCREEN MEMORY AREA
VDBAS2  EQU     VDBASE/256      ;MSB OF VDBASE
;
;
        PUSH    H               ;SAVE HL
        LHLD    VDMP            ;GET CURRENT CURSOR LOC
        CALL    VDM             ;DO OUTPUT
        SHLD    VDMP            ;SAVE CURSOR LOCATION
        POP     H               ;RESTORE HL
        RET                     ;
;
VDM:
        MOV     A,C             ;PUT CHAR IN A
        ANI     7FH             ;TURN OFF PARITY
        CPI     7FH             ;CHECK FOR RUBOUT
        JZ      EXIT            ;DO NOTHING IF RUBOUT
        MOV     A,M             ;GET CURSOR
        ANI     7FH             ;TURN IT OFF
        MOV     M,A             ;PUT BACK ON SCREEN
        MOV     A,C             ;GET CHAR IN A
        CPI     ' '             ;CHECK FOR CONTROL CHAR
        JC      CTLCHR          ;IF YES, DO SOMETHING SPECIAL
        MOV     M,A             ;PUT ON SCREEN
        INX     H               ;POINT TO NEXT LOCATION
        MOV     A,H             ;GET SCREEN ADDR MSB
        CPI     VDBAS2+4        ;CHECK FOR END OF SCREEN
        CZ      SCRL            ;IF YES, THEN SCROLL
        JMP     EXIT            ;GET OUT OF HERE
SCRL:
        LXI     H,VDBASE        ;POINT TO START OF SCREEN
        PUSH    B               ;SAVE BC
        LXI     B,VDBASE+64     ;PUT START + 1 LINE IN B
SCRL1:
        LDAX    B               ;GET BYTE FROM SCREEN
        MOV     M,A             ;AND PUT IN NEW POSITION
        INX     B               ;INCREMENT POINTERS
        INX     H               ;
        MOV     A,B             ;GET ADDR MSB
        CPI     VDBAS2+4        ;CHECK FOR END OF SCREEN
        JNZ     SCRL1           ;IF NOT, THEN KEEP DOING
        POP     B               ;RESTORE BC
        PUSH    H               ;SAVE THIS ADDR
SCRL2:
        MVI     M,' '           ;PUT BLANK ON LAST LINE
        INX     H               ;POINT TO NEXT LOC
        MOV     A,L             ;GET ADDR LSB
        ANI     3FH             ;CHECK END OF LINE
        JNZ     SCRL2           ;NOT YET
        POP     H               ;POINT TO BEGIN OF LINE
        RET                     ;WE ARE DONE
CTLCHR:
        CPI     08H             ;IS IT BS
        JZ      DELETE          ;YES
```

```
        CPI     0DH             ;IS IT CR
        JZ      CRTN            ;YES
        CPI     0AH             ;IS IT LF
        JZ      CLF             ;YES
        CPI     0BH             ;IS IT HOME
        JZ      HOM             ;YES
        CPI     0CH             ;IS IT FF
        JZ      CLEAR           ;YES
        JMP     EXIT            ;NOT WANTED-DO NOTHING
CRTN:
        MOV     A,L             ;GET CURSOR LOC LSB
        ANI     0C0H            ;PUT AT BEGIN OF LINE
        MOV     L,A             ;PUT BACK
        JMP     EXIT            ;GET OUT
DELETE:
        MOV     A,L             ;GET CURSOR LOC LSB
        ANI     63              ;SEE IF BEGIN OF LINE
        JZ      EXIT            ;YES - DO NOTHING
        DCX     H               ;
        MVI     M,' '           ;SET NEW LOC TO SPACE
        JMP     EXIT            ;GET OUT
CLF:
        PUSH    H               ;SAVE REGS FOR SCROL
        PUSH    B               ;
        LXI     B,64            ;PUT LINE LENGTH IN BC
        DAD     B               ;ADD TO CURRENT LOC
        MOV     A,H             ;GET LOC MSB
        CPI     VDBAS2+4        ;IS IT PAST END OF SCREEN
        POP     B               ;RESTORE BC
        JNZ     XTRA            ;HL OK BUT GIGO ON STACK
        CALL    SCRL            ;SCROLL IT
        POP     H               ;RESTORE HL
        JMP     EXIT            ;GET OUT
XTRA:
        XTHL                    ;SWAP HL WITH TOS
        POP     H               ;CLEAN STACK AND CORRECT HL
        JMP     EXIT            ;GET OUT
CLEAR:
        LXI     H,VDBASE        ;GET START OF SCREEN IN HL
        MVI     A,VDBAS2+4      ;PUT END OF SCREEN MSB IN A
CLEAR1:
        MVI     M,' '           ;PUT BLANK IN MEM
        INX     H               ;POINT TO NEXT LOC
        CMP     H               ;CHECK FOR END OF SCREEN
        JNZ     CLEAR1          ;NO, STAY IN LOOP
HOM:
        LXI     H,VDBASE        ;PUT START OF SCREEN IN HL
        JMP     EXIT            ;
EXIT:
        MOV     A,M             ;GET MEMORY BYTE
        ORI     80H             ;TURN ON CURSOR
        MOV     M,A             ;PUT IT BACK
        MOV     A,C             ;MAKE IT LOOK NORMAL
        RET                     ;GOODBYE
;
VDMP    DW      0               ;CURRENT CURSOR LOC
;
        ENDIF                   ;
;
;
;WRITE A CHARACTER ON LISTING DEVICE
;
LIST:
        IF      MYSYS           ;
        MVI     A,0AH           ;CHECK FOR A CR
        CMP     C               ;IF IT IS THEN DO NULL RTN
        MVI     A,25            ;
        JZ      ADDNL           ;
        MVI     A,0CH           ;CHECK FOR FF
        CMP     C               ;DO FF RTN IF YES
        MVI     A,25            ;
        JZ      ADDNL           ;
        MVI     A,0DH           ;
        CMP     C               ;
        MVI     A,02            ;
        JZ      ADDNL           ;
LIST1:
        IN      2               ;CHECK STATUS
        ANI     40H             ;MASK OFF TBE
        JZ      LIST1           ;WAIT FOR TBE
        MOV     A,C             ;GET DATA BYTE
        OUT     03              ;SEND IT OUT
        RET                     ;
ADDNL:
        PUSH    B               ;SAVE BC
        MOV     B,A             ;SAVE IT IN B
ADDNL1:
        CALL    LIST1           ;PRINT CR FIRST
ADDNL3:
        MVI     C,07FH          ;GET NULL CHAR
        DCR     B               ;DECREMENT COUNTER
        JNZ     ADDNL1          ;IF <>0 THEN DO MORE NULLS
ADDNL4:
        POP     B               ;RESTORE B
        MOV     A,C             ;RESTORE A
        RET                     ;RETURN FROM LIST
        ENDIF                   ;
;
;
        IF      NOT MYSYS       ;
        CALL    CONOT           ;ROUTE TO CONSOLE FOR NOW
        RET                     ;
        ENDIF
;
;NORMALLY USED TO PUNCH PAPER TAPE
;CAN BE USED AS BIT BUCKET TO CHECK FILES
;
PUNCH:
        RET                     ;
;

;NORMALLY USED TO READ PAPER TAPE
;NOT IMPLEMENTED BUT CPM REQUIRES EOF
;
READER:
        MVI     A,1AH           ;SET A=CTL-Z (EOF)
        RET                     ;
;
;DATA AREAS FOR CPM
;
TRK     DB      0               ;TRK WANTED
TRKTB   DB      1,1,1,1         ;TRACK TABLE
SECT    DB      0               ;SECTOR #
DMAADD  DW      0               ;DMA ADDRESS
DISKNO  DB      0               ;SELECTED DISK
NXTDSK  DB      0               ;
HOLD    DB      0               ;SAVE AREA FOR SEEK
;
ERCNT:  DB      0               ;ERROR COUNT
ERRS:   DB      0               ;ERROR HOLD AREA
SKCNT:  DB      0               ;SEEK ERROR HOLD
;
;
        END
```

CURSOR

Continued from Page 38

```
3250 DIM A$(LN)
3260 FOR N=1 TO LN
3270 LINEINPUT#1,D$
3280 A$(N)=D$+STRING$(WD-LEN(D$),32)
3290 NEXT N
3300 CLOSE
3310 GOSUB 2550
3320 GOTO 630
3330 REM
3340 REM
3350 REM ***********************************************************
3360 REM *                                                          *
3370 REM *     IN THIS LISTING, "|" MEANS "IS GREATER THAN          *
3380 REM *                      "[" MEANS "IS LESS THAN"            *
3390 REM *                                                          *
3400 REM ***********************************************************
3410 PRINT "HI THERE"
```

LETTERS

Continued from Page 8

use and as a co-conspirator to foment S-100 computing until a better system evolves.

Again, congratulation and good luck.

Edward Lee
Riderwood Md

Dear Sol:

Hope you don't mind the "plug" in our newsletter. I was in Atlantic City last month and picked up one of your flyers at the Computer Store in Linwood. SOCCC is a small club (25-30 members) of mostly S-100 systems, so most of the members will probably subscribe. Next time you're in the southern California area we'd be interested in hearing from you - give me a call or drop a note.

I just got my first issue yesterday - read it from cover-to-cover, and in spite of the error (omission) on page 44/45 (CBBS) I enjoyed the magazine very much. I like the format but most of all, the content - good, worthwhile and useful articles.

If there's anything I (or the club) can do to help, don't hesitate to call or write.

Mel Hengen
South Orange County Computer Club
Fountain Valley, CA

To: Sol Libes

Your first issue of **S-100 MICROSYSTEMS** was very informative and interesting to read. I hope you will be able to keep the good work up. I enjoyed reading the IEEE S-100 information. However, I must confess that I will be studying it for some time to fully understand the bus. Even though I have a SYM-I (6502 based single board computer), I feel that sometime (a Year or two) in the future I would like to build an S-100 system. It would not have to be a 6502 based system (I have corresponded with you on that subject). To spend all that money for a mainframe, etc. I might very well be interested in a 16-bit machine. I am currently investigating the Motorola MC68000. Thus, I am very happy to see the IEEE S-100 enhancements to accommodate 16-bit machines. I am very interested in having a computer system that is both "powerful and has great flexibility". With those thoughts in mind, I am looking forward to more issues of **S-100 MICROSYSTEMS** and I would be interested in articles that cover, or are about the following:

* S-100 MAINFRAMES - good, bad, noise problems and their solutions, power supplies, etc.
* CPU card design principles to meet IEEE standards.
* Operating systems (Good start with the CP/M article).
* IN GENERAL - a good mixture of hardware and software articles (as per Vol 1/ No 1).

George V. Wilder
Lisle, Ill

CLUBS

Continued from Page 12

Cromemco User Systems & Software Pool
Box 784
Palo Alto CA 94302
Tel: (415) 321-5998

Denver Amateur Computer Society
CP/M and Pascal User Groups
1380 South Santa Fe
Denver CO 80223

PROTEUS, The Processor Technology Users Society
1690 Woodside Drive, Suite 219
Redwood City CA 94061
Tel: (415) 368-3331

Space Coast Microcomputer Club
315 Inlet Ave
Merritt Island FLA 32952
Tel: (305) 452-2159

Toronto Regional Assoc. Of Computer Enthusiasts
Box 6922 Station A
Toronto Ontario M5W 1X6
Canada

MAR/APRIL 1980

ADVERTISERS

S-100 NEWS

Continued from Page 7

AMRAD TO PUBLISH CBBS DIRECTORY

The Amateur Radio Research and Development Corporation (AMRAD) -- an amateur radio and computer society with headquarters in the Washington, DC area -- is conducting a survey of computer message systems. AMRAD plans to publish a directory of Computerized Bulletin Board Systems, Apple Bulletin Board Systems, Forum-80's and similar systems in the near future. The directory is to be available to anyone at a nominal charge of $1. AMRAD member and those who contribute first-hand information about existing message systems will receive the directory free of charge.

Individuals connected with existing message systems are asked to send their names and addresses to David W. Borden, Rt 2, Box 233B, Sterling, Virginia 22170, and request a copy of the AMRAD Computer Message System Questionnaire. This questionnaire contains all the elements of information needed for entries in the directory.

IEEE DEVELOPING ASSEMBLY LANGUAGE STANDARD FOR MICROS

The Institute of Electrical and Electronic Engineers is developing a standard for assembly language on microprocessors (IEEE Task P694/D11). It is long overdue and will be of enormous value to all assembly language programmers who are struggling with different microprocessors. The group working on the standard has done some genuinely worthwhile things, such as showing that all the current major chips can be handled nicely by one standard. The problems at present are incredible. For example, on some chips MOV A,B means move the contents of register B to A, while on others it means just the opposite.

Right now, AMD is second sourcing the Zilog Z8000, and would you believe it, they are not using the Zilog mnemonics! Hopefully, the new IEEE standard will cure problems such as this. In another example, Zilog did not use the Intel mnemonics for the Z80's instructions which were a subset of the 8080.

The standard also covers Instruction Names, Address Modes, Operand Sequences, Expression Evaluation, Constants, Lables, Comments and Assembler Directives. The standard does not specify the syntax necessary to support macros or conditional assembly.

The IEEE Computer Society is to be congratulated for its activities in developing computer standards. They are overcoming problems created by companies that all too often purposely create incompatibilities in order to protect their competitive position.

I predict that this standard will meet with the wide adoption that the other IEEE standards (IEEE-488 and IEEE-S-100 bus standards) are meeting with. Incidentally, you can obtain a copy of the Assembly Language Standard draft by sending a self-addressed 10x13 size envelope with 4 stamps on it to Dr. Robert G. Stewart, Chairman Computer Standards Committee, IEEE Computer Society, 1658 Belvoir Drive, Los Altos CA 94022.

Incidentally, the IEEE is also working on several other standards relevant to the microcomputer area. They are: Multibus (Task No. P696.2), Microbus (Task No. P696.3), Futurebus (Task No. P696.4), Floating Point (Task No. P754), High Level Languages (Task No. 755), Pascal (Task No. 770) and Relocatable Object Format (Task No. 695). I will try and report on the progress of these standards in a future S-100 MICROSYSTEMS column.

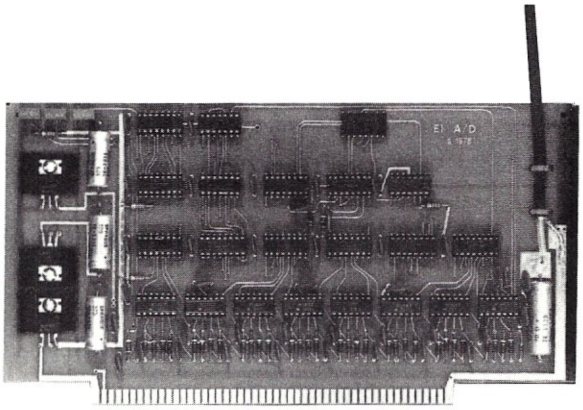
SOL-LIKE COMPUTER TO BE PRODUCED

Lee Felsenstein, the wizard who created the Processor Tech SOL computer, the VDM, the 3P+S and several other products is back with a new computer that is in effect a "SOL-II". (incidentally, before we go any further I must say that contrary to a statement made in a magazine recently, the SOL was not named after me.)

The new computer, whose new name and manufacturer is still a secret will debut in April. It is being manufactured here in the U.S. by a Swedish concern. Physically it will look like a SOL except that it will be housed in a vacuum formed plastic enclosure. It will operate with a TV monitor and have a built-in cassette, printer and spare parallel I/O ports. The entire computer will be on a single board which includes the keyboard and will have as an option, a 4-slot S-100 bus motherboard (IEEE compatible) for plug-ins such as disk controller and memory and I/O expansion.

The computer will have Microsoft BASIC (V5.1) in ROM. They can be removed and the space converted to RAM by changing some jumpers. The computer will have a lot of powerful goodies such as: Real-time clock, interrupt driven keyboard, priority interrupt system, 16K static RAM on the main board, use of cassette port for RS-232 modem I/O, options such as the ACE multitasking scheduler and best of all, the unit will sell for under $1K.

NEW PRODUCTS

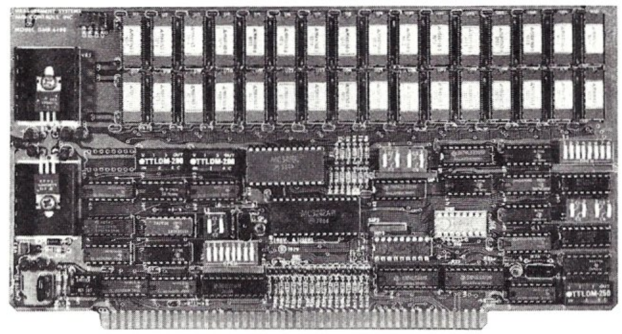


VIDEO DIGITIZER/MONITOR INTERFACE

TECMAR INC. has introduced a Real Time Video Digitizer and Monitor Interface (RT+MI) for S-100 systems, which digitizes video data from TV cameras and uses the digital data to reconstruct a picture on a TV moniotr. It digitizes the picture in 1/60 second and deposits it into memory as a single operation using DMA. It displays pictures in 16 gray levels or black and white combination. It can simultaneously deposit and display allowing constant viewing of picture until desired image is seen on monitor, when it can be frozen on monitor. The last digital image displayed remains in memory and can be displayed at any time thereafter without erasing it from memory. The image can be processed or put on disk for later retrieval. Maximum resolution is 512 x 240 pixels. The complete RT+MI, includes Video A/D(shown above), Video D/A and DMA controller boards occupying 3 S-100 slots and costing $850. For information contact: TEC Mar Inc., 23414 Greenlawn, Cleveland Ohio 44122; tel:(216)382-7599.

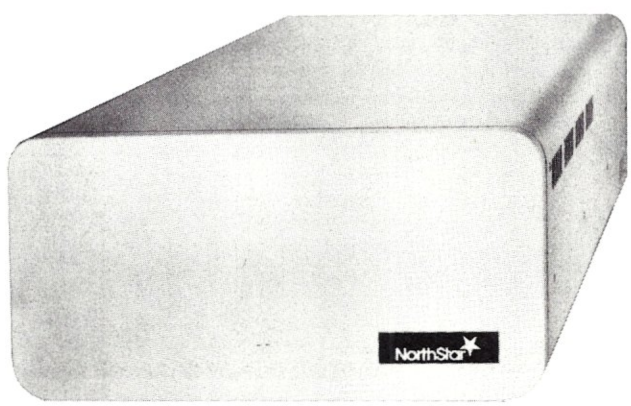
64K RAM BOARD INTRODUCED

Measurement Systems & Controls Inc., has released the DMB-6400, a 64K Bank Switchable Dynamic Memory Board for S-100 systems. Output port addressing is used for bank selection of 4 totally independent 16K banks of memory. Each bank can be turned ON or OFF at system reset and PHANTOM can be used by any of the four banks. Four diagnostic LEDs indicate which banks of memory are on. The board will operate with all 8080, 8085 @ 3 MHz and most Z80A @ 4MHz CPU boards. In addition, it will operate with the Marin Chip M9900 CPU. For further information contact: Measurement Systems & Controls Inc.,867 North Main Street, Orange CA 92667; tel: (714)633-4460.



NORTH STAR INTRODUCES HARD DISK

North Star Computers Inc. has announced a new Winchester-type 18Mbyte enhancement for its Horizon computers and users of North Star floppy disk systems. Up to four hard disks and 2 mini-floppy disks may be accommodated on one system, providing up to 72Mbytes of storage on the hard disks and over 1Mbyte on the floppy drives. Century Data Marksman hard disk drives are utilized with an average access time of 78 msec. Software supplied with the system includes a File Manager, a program for creating backup diskettes, a Command Processor (supporting all the North Star floppy disk DOS and Monitor commands, while adding others) and BASIC interpreter (modified to support hard disk files and run all previous North Star Basic programs with little or no change). For more information contact: North Star Computers, 1440 Fourth Street, Berkeley CA 94710; tel: (415)527-6950.



The Fifth
Trenton Computer Festival
TCF-80

April 19 & 20, 1980

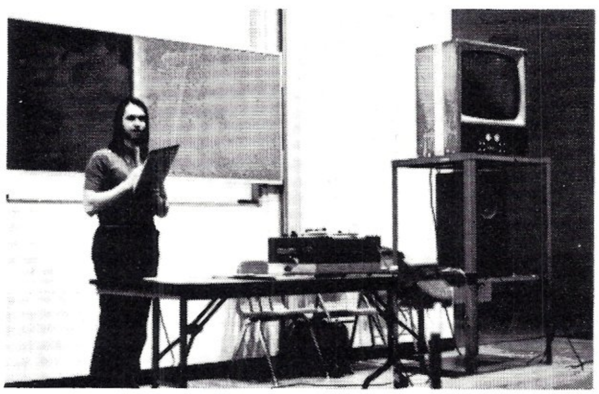
10 AM to 6 PM
SATURDAY, 19th

at TRENTON STATE COLLEGE
Trenton, New Jersey

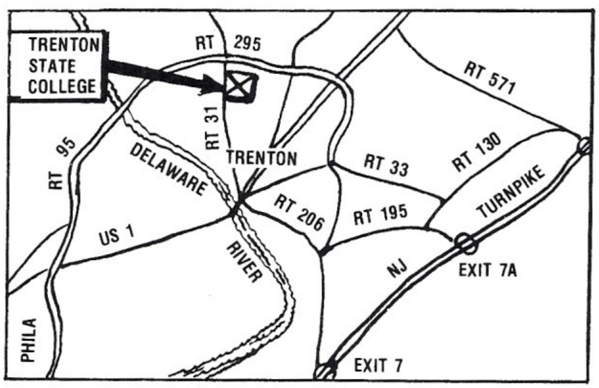
10 AM to 4 PM
SUNDAY, 20th

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

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



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



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

Free Short Courses on Sunday
Hundreds of Door Prizes • Banquet Saturday Night
For additional information call 609-771-2487
Admission $5 - Students $2
Tickets available only at door

Banquet $10. **Avoid disappointment – purchase tickets early on Saturday at door.**

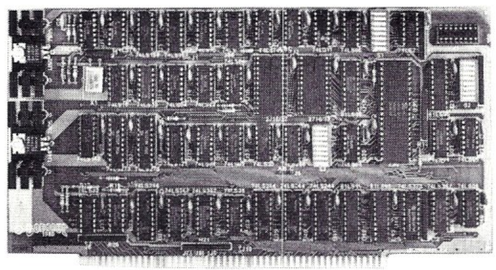
Sponsored by:
Amateur Computer Group of New Jersey
Philadelphia Area Computer Society
Trenton State College Digital Computer Society
Dept. of Engineering Technology, Trenton State College
I.E.E.E., Princeton Section

the FIRST of the New Generation

Our innovative **Z-80A CPU board** is truly the first of a new generation of S-100 bus equipment . . . a generation that's designed to accomodate multi-user setups and other high level industrial, scientific, and commercial applications.

This CPU card contains all the good Z-80 features of other boards, but also features full compliance with the proposed IEEE S-100 bus standards, provision for adding two EROMs for 4K to 8K of on-board ROM (2716 or 2732 – not included with board), power on jump to any of 256 boundaries, on-board fully maskable interrupts at port FE (hex) for interrupt driven systems, 2 or 4 MHz operation, power on clear that generates preset and slave clear according to IEEE specs, selectable automatic wait state insertion for servicing M1* instructions – MRQ* – I/O RQ* – or the on-board ROM (individually or severally selectable), non-maskable interrupt on bus pin 12 as per IEEE specs, and we've also included on-board IEEE compatible extended addressing at port FD (hex).

These advanced features give you the power you need for future expansion, as well as the system flexibility that comes from superior design . . . but perhaps best of all, the price is competitive with boards that do a whole lot less. **The Z-80A CPU board is available for $225 unkit, $295 assembled, and $395 for units qualified under our Certified System Component program.**



Next month: The second new generation CPU board, featuring our 8085/8088 dual processing technique, as well as the Spectrum color graphics board. Please do **not** call for information on these products; send an SASE and we'll send the information to you.

HIGH PERFORMANCE S-100 MOTHERBOARDS

6 slot: $89 unkit, $129 assm

12 slot: $129 unkit, $169 assm **19 slot: $174 unkit, $214 assm**

Ideal for use with the above enclosure. **Unkits** have edge connectors and termination resistors pre-soldered in place for easy assembly. Meets or exceeds IEEE S-100 specs; includes true active termination, grounded Faraday shield between all bus signal lines, and edge connectors for all slots.

LOOKING FOR MEMORY?

Our boards are static, run up to 5 MHz, meet the IEEE S-100 standards, low in power, and include a 1 year warranty. Choose from **unkit** (sockets, bypass caps pre-soldered in place), **assembled**, and boards qualified under our high-reliability **Certified System Component** (CSC) program.

Memory Name	Buss & Notes	Unkit	Assm	CSC
8K Econoram* IIA	S-100	**$169**	**$189**	**$239**
16K Econoram XIV	S-100 (1)	**$299**	**$349**	**$429**
16K Econoram X-16	S-100	**$329**	**$379**	**$479**
16K Econoram XIIIA-16	S-100 (2)	**$349**	**$419**	**$519**
16K Econoram XV-16	H8 (3)	**$339**	**$399**	n/a
16K Econoram IX-16	Dig Grp	**$319**	**$379**	n/a
24K Econoram VIIA-24	S-100	**$449**	**$499**	**$599**
24K Econoram XIIIA-24	S-100 (2)	**$479**	**$539**	**$649**
32K Econoram X-32	S-100	**$599**	**$689**	**$789**
32K Econoram XIIIA-32	S-100 (2)	**$649**	**$729**	**$849**
32K Econoram XV-32	H8 (3)	**$649**	**$749**	n/a
32K Econoram IX-32	Dig Grp	**$599**	**$679**	n/a
32K Econoram XI	SBC/BLC	n/a	n/a	**$1050**

Econoram is a trademark of **Godbout Electronics.**

(1) Extended addressing (24 address lines). Addressable on 4K boundaries.
(2) Compatible with all bank select systems (Cromemco, Alpha Micro, Etc.); addressable on 4K boundaries.
(3) Bank select option for implementing memory systems greater than 64K.

THE GODBOUT COMPUTER BOX

$289 desktop, $329 rack mount

Quiet fan, dual AC outlets and fusehold, heavy-duty line filter, black anodized front panel, and card guide. Ask about our matching power supply.

S-100 2708 EROM BOARD $85 unkit

4 independently addressable 4K blocks. Includes all support chips and manual, but does not include 2708 EROMs.

S-100 ACTIVE TERMINATOR BOARD $34.50 kit

Plugs into older, unterminated motherboards to improve performance.

S-100 MEMORY MANAGER BOARD

$59 unkit, $85 assm, $100 CSC

Adds bank select and extended addressing to older S-100 machines to dramatically increase the available memory space.

2S "Interfacer I" S-100 I/O BOARD

$199 unkit, $249 assm, $324 CSC

Dual RS-232 ports with full handshake. On-board crystal timebase, hardware UARTS, much more.

3P PLUS S "INTERFACER II" I/O BOARD

$199 unkit, $249 assm, $324 CSC

Includes 1 channel of serial I/O (RS-232 with full handshake), along with 3 full duplex parallel ports plus a separate status port.

PASCAL/M™ + MEMORY SPECIAL

PASCAL can give a microcomputer with CP/M more power than many minis. You can buy our totally standard **Wirth PASCAL/M™ 8" diskette**, with manual and Wirth's definitive book on PASCAL, **for $150 with the purchase of any memory board**. Specify Z-80 or 8080/8085 version. PASCAL/M™ available separately for $350.

SEE COMPUPRO PRODUCTS IN PERSON: Many of these products are stocked by finer computer stores world-wide, or write us if there's no dealer in your area.

TERMS: Cal res add tax. Allow 5% for shipping, excess refunded. VISA®/Mastercharge® call our 24 hour order desk at **(415) 562-0636**. COD OK with street address for UPS. Sale prices good through cover month of magazine; other prices are subject to change without notice.

CompuPro™ from GODBOUT ELECTRONICS

Bldg. 725, Oakland Airport, CA 94614 (415) 562-0636