LINEAR PROGRAMMING TECHNIQUES IN PASCAL

See Page 14

Also in this Issue

An Introduction to CP/M* BY JAKE EPSTEIN .................................. Page 28
NORTH STAR Topics BY RANDY REITZ ........................................ Page 10
Addressing The Cursor BY LARRY STEIN ..................................... Page 34
S-100 Bus — New Versus Old BY SOL LIBES ................................ Page 39

and more

Complete Table of Contents on Page 3
Suddenly, S-100 microcomputer systems can easily handle 100 million bytes. Because Morrow Designs™ now offers the first 26 megabyte hard disk memory for S-100 systems—the DISCUS M26™ Hard Disk System.

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

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

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

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

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

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

*CP/M is a trademark of Digital Research.
Volume 1 Number 2

March/April 1980

IN THIS ISSUE

North Star Topics ........................................... 10
by Randy Reitz

Linear Programming Techniques in Pascal .............. 14
by William Yarnall

Introduction to CP/M*, Part 2 ............................ 28
by Jake Epstein

Addressing the Cursor ..................................... 34
by Larry Stein

S-100 Bus — New Versus Old ............................. 39
by Sol Libes

Product Review: CGS-808 Intelligent Color Graphics Board .......................... 43
by Jon Bondy

Tarbell Disk Controller Mods
Hardware Mods by George Lyons ......................... 46
Improved BIOS by Marty Nichols ......................... 49

DEPARTMENTS

Editor's Page ................................................ 4
S-100 MICROSYSTEMS News ............................... 6
Letters to the editor ........................................ 8
Club Directory .......................................... 12
Advertiser Index ....................................... 55
Subscription Information ......................... 57
New Products ......................................... 58

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.

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

STAFF

Sol Libes
publisher/editor

Russell Gorr
executive editor

Dennis Thovson
technical editor

Jacob Epstein
CP/M* editor

Jon Bondy
Pascal editor

Don Libes
assistant editor

Lennie Libes
subscriptions/office manager

Susan Libes
subscriptions/office manager

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.

*S/MK Digital Research

Copyright © 1980 by Libes, Inc.
All rights reserved. Reproduction prohibited without permission.
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-99 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 (9600, 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 have any other system with this processor independence. Many of these microprocessors could not 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 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-I 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/PO. Box 91,
Ithaca, NY 14850
607-257-0190/TWX: 510 235 4346
UCSD REVOCKES 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 information 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 occasional newsletter that is a couple of hundred pages that is a compendium of all the news that is 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.

28000 S-100 CPU CARDS TO BE AVAILABLE

Ithaca Intersystems, Ithaca, N.Y., (formerly Ithaca Audio) will soon introduce a 28000 CPU board for S-100 based systems. It will meet IEEE S-100 specs and contain an operating system in ROM. It will also have a version of their Pascal/Z software which can compile to either 280 or 28000 code. Therefore any present 280 Pascal/Z software can be immediately recompiled to the new 28000.

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 Electronics, 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 280 CPU card for S-100 systems. Don Tarbell said that he feels that the 280, 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 CP/M User Group will add eight new disks to the CP/M 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 prestigious 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.

**IMSIA LIVES ANEW**
IMSIAI, one of the pioneers in the microcomputer field is alive and functioning as the the IMSIAI COMPUTER Division of Fischer-Freitas Corp., San Leandro CA. FFC was the outfit handling IMSIAI's production and warranty service at the time that IMSIAI went into bankruptcy, last year. Actually, the new IC division is staffed with all former IMSIAI 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 IMSIAI stock and the rights to use the IMSIAI and IMDOS trademarks. The IC division is manufacturing and selling the full line of IMSIAI Hardware products. They are presently selling copies of the IMDOS operating system which was purchased from IMSIAI. 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: IMSIAI 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.
Hector M. Smith
Fountain Valley CA

Dear Sol:
I'm enclosing a check for $14.00 for a two year subscription, as mentioned in 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

--- CONTINUED ON PAGE 55 ---

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
Software for most popular 8080/Z80 computer disk systems including CP/M, BASIC, KIM, COBOL, and many others. Descriptions for copying into CIS COBOL programs.

**CP/M**

Version 2

- Requires 20K RAM.
- MDBS, DRMS, and ZSO versions.
- When new items, records, or sets are needed, ordering is sorted. FIFO, LIFO, next or prior. One to many set relationship supported.
- Read, write, protect, and directly compiled by CIS COBOL.
- Programs directly compiled by CIS COBOL are used in standard CIS COBOL-ANSI '74 COBOL standard and ZSO CPU.

**BASIC**

- 8080 or Z80 Macro Assembler. Intel and compatible assembler. Linking loader and relocatable code.
- CALL, COPY, SEARCH, 2-dimensional arrays, common utilities plus full documentation.
- **CP/M** version also available.

**COBOL**

- Level 1: ANSI standard COBOL, plus extended COBOL (85/89).
- Level 2: IBM standard COBOL, plus extended COBOL (85/89).
- Level 3:anked COBOL, plus extended COBOL (85/89).
- Micro Focus: COBOL compiler and run time with 20K RAM or 30K assembly language.
- Standard COBOL with extended COBOL (85/89) and static and register class specifications.

**MACRO**

- 8080 Macro Assembler. Inter and Z80 Macro Assembler.
- 8080 Macro Assembler with option for 68000 and 68020.

**FORTAN**

- Level 1: ANSI standard FORTRAN, plus extended FORTRAN (85/89).
- Level 2: ANSI standard FORTRAN, plus extended FORTRAN (85/89).
- Z80 FORTRAN, Level 1, 2, and 3.

**INVENTORY SYSTEM**

- Captures stock levels, costs, and in/out of stocks. Captures data on various fields. Captures data on various types of transactions. Captures data on various types of transactions. Requires CBASIC-2.$1250/$25.

**ANALYST**


**SUPERGRIP**

- Sort, merge, extract utility as above.

**SUPERGRIP**

- A tool for manipulating data. Data is fed into a report writer. Data is fed into a report writer. Data is fed into a report writer. Requires CBASIC-2.$1250/$25.

**EODS SYSTEMS**

- Kiosk System, Sequential Search, and Printer System.

** kèk System**

- Kiosk System, Sequential Search, and Printer System.

**TEXT SYSTEM**

- Sorting, merging, extracting utility as above.

**PROFIT SYSTEM**

- Accounting program with option for 68000 and 68020.

**KISS**

- Keyed Index Sequential Search.

**SUPER-STRIP**

- Sort, merge, extract utility as above.

**OVERCOM**

- Frozen, and defrosting utility.

**TAS**

- Transaction processing system. Supports multiple users. Supports multiple users. Requires ZSO Monitor Debugger to break and examine programs.

**WHATSIT?**

- A trademark of Computer Hardware.

**SCANNED DISK EXTENDED COBOL**

- Basic language with extended COBOL (85/89).

**DOS**

- IBM operating system with extended COBOL (85/89).

**CBS**

- Customized data entry and reporting. Captures data on various fields. Captures data on various types of transactions. Captures data on various types of transactions. Requires CBASIC-2.$1250/$25.

**POSTMASTER**

- A comprehensive package for managing and processing transactions. Features include keyboard recognition and label printing. Requires CBASIC-2.$1250/$25.

**DATAS**

- Interactive database system using Structured Query Language (SQL) and Structured Query Language (SQL).

**MACRO**

- 8080 Macro Assembler. Intel and compatible assembler. Linking loader and relocatable code.

**API**

- IBM API, IBM API, IBM API, IBM API, IBM API.

**SUPER-SORT**

- Sort, merge, extract utility as above.

**MASSAGE**

- A full text search tool.

**WORD-STAR/MAIL-MERGE**

- As above with option for mail-merge.

**SUPER-STRIP**

- Sort, merge, extract utility as above.

**MACRO**

- 8080 Macro Assembler. Intel and compatible assembler. Linking loader and relocatable code.

**API**

- IBM API, IBM API, IBM API, IBM API, IBM API.

**SCANNED DISK EXTENDED COBOL**

- Basic language with extended COBOL (85/89).

**DOS**

- IBM operating system with extended COBOL (85/89).

**CBS**

- Customized data entry and reporting. Captures data on various fields. Captures data on various types of transactions. Captures data on various types of transactions. Requires CBASIC-2.$1250/$25.

**POSTMASTER**

- A comprehensive package for managing and processing transactions. Features include keyboard recognition and label printing. Requires CBASIC-2.$1250/$25.

**DATAS**

- Interactive database system using Structured Query Language (SQL) and Structured Query Language (SQL).

**MACRO**

- 8080 Macro Assembler. Intel and compatible assembler. Linking loader and relocatable code.

**API**

- IBM API, IBM API, IBM API, IBM API, IBM API.

**SUPER-SORT**

- Sort, merge, extract utility as above.

**MASSAGE**

- A full text search tool.

**WORD-STAR/MAIL-MERGE**

- As above with option for mail-merge.

**MACRO**

- 8080 Macro Assembler. Intel and compatible assembler. Linking loader and relocatable code.

**API**

- IBM API, IBM API, IBM API, IBM API, IBM API.
Welcome to what I hope will be a regular feature in S-IOO 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 restricted 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:

```assembly
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 CKCR LF ;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
```

S-100 MICROSYSTEMS
NEW! TPM for TRS-80 Model II
NEW! System/6 Package
Computer Design Labs

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

— Carl Galletti and Roger Amidon, owners.

Software with Manual/Manual Alone

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

for TRS-80® CP/M (Model I or II) for 8" or 10" CP/M (soft sectored single density) for 5½" CP/M (soft sectored single density) for 5½" North Star CP/M (single density) for 5½" North Star CP/M (double density)

BASIC I

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

BASIC II

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

BUSINESS BASIC

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

ZEDIT

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

ZTEL

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

ZTOP

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

MACRO I

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

MACRO II

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

LINKER

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

DEBGI

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

DEBGI!

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

ZAPPLE

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

ZAPPLE II

8060 version of Zapple. $34.95/$15.

NEW! TPM now available for TRS-80 Model II

TPM*

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

SYSTEM MONITOR BOARD (SMB II)

A complete I/O board for S-100 systems. 2 serial ports, 2 parallel ports, 1500/2400 baud cassette tape interface, sockets for 2 of RAM, 3-2708/2717 EPROM's or ROM, jump on reset circuitry. Basic board $49.95/$20.

ROM FOR SMB II

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

PAYROLL (source code only)

The Osborne package. Requires C Basic 2. 5" disks $124.95 (manual not included) 8" disks $199.95 (manual included) Manual $20.00

ACCOUNTS PAYABLE/RECEIVABLE (source code only)

By Osborne. Requires C Basic 2. 5" disks $124.95 (manual not included) 8" disks $199.95 (manual included) Manual $20.00

GENERAL LEDGER (source code only)

By Osborne. Requires C Basic 2. 5" disks $199.95 (manual not included) 8" disks $299.95 (manual not included) Manual $20.00

C BASIC 2

Required for Osborne software. $99.95/$20.

SYSTEM/8

TPM with utilities, Basic Interpreter, Basic E compiler, Macro I assembler, Debug I debugger, and ZEDIT text editor.

Above purchased separately costs $339.75

Special introductory offer. Only $179.75 with coupon!

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

1. Name of Product (e.g. Macro I)
2. Media (e.g. 8" CP/M)
3. Price and method of payment (e.g. C.O.D. include credit card info. if applicable.
4. Name, Address and Phone number.
5. For TPM orders only: Indicate if for TRS-80, Tarbell, Xitan DDDC, SD Sales (5½" or 8"), North Star (single or double density) or Digital (Micro) Systems.
6. N.J. residents add 5% sales tax.

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

For information and tech queries call

609-599-2146

For phone orders ONLY call toll free

1-800-327-9191

Ext. 676

(Except Florida)

OEMS

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

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

DEALER INQUIRIES INVITED.

Computer Design Labs
342 Columbus Avenue
Trenton, N.J. 08629
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 number 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.

S-100 MICROSYSTEMS
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, 80, CP/M, N*, Pascal User Groups
Box 106, Church St. Station, NYC, NY 01007
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
(N-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 August 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 in
$14.00 per year* U.S.
Other Foreign: 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

NEW!

6809 SINGLE-BOARD COMPUTER

S-100 bus

- IEEE S-100 Proposed Standard
- 2K RAM
- 4K/8K/16K ROM
- PIA, ACIA Ports
- add MON: 6839 Monitor Available
- PC Board & Manual Presently Available

ALL PCB BOARDS FROM ADS ARE SOLDER MASKED, WITH GOLD CONTACTS, & PARTS LAYOUT SILK SCREENED ON BOARD. Add 50c postage & handling per item. Ill. residents add sales tax.

Sound Effects... Sound Effects...!!!

ADD "SPACESHIP" SOUNDS, PHASERS,
GUNSHOTS, TRAINS, MUSIC, SIRENS, ETC.
UNDER SOFTWARE CONTROL!!!

- Soundboards Use GIAY-3 6810 L.C. s to Generate
Programmable Sound Effects.
- On Board Audio Amp: Breadboard Area With +5 & GND.
- Noise Sources: Envelope Generators & I/O Ports

PCB & Manual: $39.95 (NM), $34.95 (NM II)

!!!!!!!!ATTENTION APPLE II USERS!!!!!
Assembled and Tested NM II Units Now Available!!!

Call or Write for Details.

ACKERMAN DIGITAL SYSTEMS, INC., 110 N. YORK ROAD, SUITE 208, ELMHURST, ILLINOIS 60126 (312) 530-8992

S-100 MICROSYSTEMS

13
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 occurred 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.

   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

\[ X_1 + 2X_2 + 3X_3 - X_4 \]  

subject to

\[
\begin{align*}
X_1 + 2X_2 + 3X_3 &= 15 \\
2X_1 + X_2 + 5X_3 &= 20 \\
X_1 + 2X_2 + X_3 + X_4 &= 10
\end{align*}
\]  

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

\[-X_1 - 2X_2 - 3X_3 + X_4\]  

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:

\[
\begin{bmatrix}
1.0 & 2.0 & 3.0 & 0.0 \\
2.0 & 1.0 & 5.0 & 0.0 \\
1.0 & 2.0 & 1.0 & 1.0
\end{bmatrix}
\]

It can be seen that a column in this array corresponds to the coefficient of a variable (col 1 = X₁, 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 "UNBOUND 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**


**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.

**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

S-100 MICROSYSTEMS
LISTING 1

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

ENTER FILE NAME ---> LINEARP.TEXT

1: PROGRAM LINEARP;
2: (* MINIMIZE A COST FUNCTION SUBJECT TO CONSTRAINTS.  
3: MAXIMIZE NEGATIVE OF 'PROFIT' FUNCTION.  *)
4:
5: CONST
6: MAXROW = 32;
7: MAXCOL = 64;
8: TYPE
9: ROW = ARRAY [1..MAXROW] OF REAL;
10: COL = ARRAY [1..MAXCOL] OF REAL;
11: FREC = RECORD
12: CASE TAG: INTEGER OF
13: 0: (NAME:STRING{6}; NUM1, NUM2:INTEGER);
14: 1: (HEADER:STRING{64});
15: 2: (NAME:STRING{6}; RINDEX:INTEGER; RHS:REAL);
16: 3: (NAME:STRING{6}; CINDEX:INTEGER; OBJ:REAL);
17: 4: (RNAME:STRING{6}; RINDEX:INTEGER; RHS:REAL);
18: 5: (CNAME:STRING{6}; CINDEX:INTEGER; OBJ:REAL);
19: 6: (R, S: INTEGER; T: REAL);
20: 99: ()
21: END;
22:
23: VAR
24: U : ARRAY [1..MAXROW, 1..MAXROW] OF REAL;
25: A BAR : ARRAY [1..MAXROW, 1..MAXCOL] OF REAL;
26: X, XIK : ARRAY [1..MAXROW] OF REAL;
27: LIST : ARRAY [1..MAXROW] OF INTEGER;
28: ROWNAME : ARRAY [1..MAXROW] OF STRING{6};
29: COLNAME : ARRAY [1..MAXCOL] OF STRING{6};
30: FILEID : FILE OF FREC;
31: FILNAM : STRING;
32: F : FREC;
33: HEADING : STRING{64};
34: HDRFLAG : BOOLEAN;
35: M, N, MP, M1 : INTEGER;
36: PNAME : STRING{6};
37: RESULT : INTEGER;
38:
39: PROCEDURE PRINTH;
40: BEGIN
41:  WRITELN(' '); 
42:  WRITELN(' ' PROG. NAME = ', PNAME);
43:  WRITELN(' ' NO. ROWS = ', M:6);
44:  WRITELN(' ' NO. COLS = ', N:6);
45:  WRITELN(' '); (* PRINTH *)
46:  END;
PROCEDURE PRINTB;
BEGIN
WRITELN('BAD FILE FORMAT');
END;

PROCEDURE PRINTC(R : ROW; C : COL);
VAR
I : INTEGER;
BEGIN
WRITELN('INITIAL DATA');
WRITELN('OBJECTIVE VECTOR');
WRITELN('RHS VECTOR');
FOR I:=1 TO N DO WRITELN(COLNAME[I], C[I], ABAR[I]);
WRITELN('ABAR(M+1), ABAR(M+2)');
FOR I:=1 TO N DO WRITELN(COLNAME[I], ABAR[I], ABAR[M], ABAR[MP]);
END;

PROCEDURE PRINTD;
VAR
I, J : INTEGER;
BEGIN
WRITELN('ABAR ARRAY');
FOR J:=1 TO N DO
FOR I:=1 TO M DO
WRITELN(I, J, ROWNAME[I], COLNAME[J], ABAR[I], ABAR[M], ABAR[MP]);
END;

PROCEDURE PRINTX;
VAR
I : INTEGER;
S : STRING[6];
BEGIN
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;
IF NOT 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;
PRINTB;
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 *)
157: GET(FILEID)
158: END; (* WHILE *)
159: IF FIRSTIN THEN
160: BEGIN
161: PRINTC(B,C);
162: FOR J:=1 TO N DO ABAR[M1,J]:=C[J];
163: FOR I:=1 TO M DO
164: IF B[I] < 0.0 THEN
165: BEGIN
166: B[I]:=-B[I];
167: FOR J:=1 TO N DO ABAR[I,J]:=-ABAR[I,J]
168: END;
169: FOR J:=1 TO N DO
170: BEGIN
171: SUM:=0.0;
172: FOR I:=1 TO M DO SUM:=SUM-ABAR[I,J];
173: ABAR[M,P,J]:=SUM
174: END;
175: B[M1]:=0.0;
176: SUM:=0.0;
177: FOR I:=1 TO M DO SUM:=SUM-B[I];
178: B[M,P]:=SUM;
179: FOR I:=1 TO MP DO
180: BEGIN
181: XI[I]:=B[I];
182: LIST[I]:=N+I;
183: FOR J:=1 TO MP DO U[I,J]:=0.0
184: END;
185: FOR I:=1 TO MP DO U[I,1]:=1.0;
186: PRINTD;
187: ROWNAME[M1]:="M+1 ";
188: ROWNAME[M,P]:="M+2 ";
189: PRINTX
190: END; (* IF FIRSTIN *)
191: WRITELN(" ");
192: END; (* INITIAL *)
193: PROCEDURE PHASE1; FORWARD;
194: PROCEDURE EXIT1(X:INTEGER);
195: BEGIN
196: RESULT:=1; (* NORMAL EXIT *)
197: WRITELN(" END OF PHASE 1 FOR ",PNAME," AFTER ",X," ITERATIONS");
198: PRINTX;
199: EXIT(PHASE1)
200: END; (* EXIT1 *)
201: PROCEDURE EXIT2(X:INTEGER);
202: BEGIN
203: RESULT:=2; (* ERROR EXIT *)
204: WRITELN(" ERROR IN ITERATION ",X);
205: PRINTX;
206: EXIT(PHASE1)
207: END; (* EXIT2 *)
PROCEDURE EXIT1<X:INTEGER>;
BEGIN
RESULT:=0; (* NO FEASIBLE SOLUTION *)
WRITELN(' NO FEASIBLE SOLUTION AFTER X ITERATIONS');
PRINTX;
EXIT(PHASE1)
END; (* EXIT1 *)

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');
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];
X[I, J]:=SUM
END;
THETA:=1.00E+36;
L:=0;
FOR I:=1 TO M DO
268: IF(XIK[I]>0.0) THEN
269: BEGIN
270:   Z:=X[I]/XIK[I];
271: IF(Z=THETA) AND (LIST[I])N)
272: THEN L:=I
273: ELSE
274: IF(Z<THETA) THEN
275: BEGIN
276:   THETA:=Z;
277:   L:=I
278: END
279: END;
280: IF(L=0) THEN EXIT2(ITER);
281: LIST[L] := KSAVE;
282: FOR I:=1 TO MP DO
283: BEGIN
284:   V[I] := XIK[I]/XIK[L];
285:   WIJ := U(L, I)
286: END;
287: XL := X[L];
288: XLK := XIK[L];
289: FOR I:=1 TO MP DO
290: BEGIN
291:   Z := THETA;
292: IF(LIST[I] <> KSAVE) THEN Z := X[I]-XL*V[I];
293: X[I] := Z;
294: FOR J:=1 TO M DO
295: BEGIN
296:   Z := WJ/J/XLK;
297: IF(CU(J) THEN Z := UC(I, J)-WJ/J*V[I];
298:   UC(I, J) := Z
299: END
300: END;
301: WRITELN(' ITERATION ', ITER, ' OF ', PNAME);
302: PRINTX
303: END (* WHILE *)
304: END; (* PHASE1 *)
305: PROCEDURE PHASE2; FORWARD;
306: PROCEDURE EXIT4(X:INTEGER);
307: BEGIN
308: RESULT:=1; (* NORMAL EXIT *)
309: WRITELN(' END OF PHASE 2 FOR ', PNAME, ' AFTER ', X, ' ITERATIONS');
310: PRINTX;
311: EXIT(PHASE2)
312: END;
313: PROCEDURE EXIT5(X:INTEGER);
314: BEGIN
315: RESULT:=2; (* UNBOUNDED SOLUTION *)
316: WRITELN(' UNBOUNDED SOLUTION FOR ', PNAME);
317: PRINTX;
318: EXIT(PHASE2)
319: END;
320: 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[I,M1,J]+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];
XI[K]:=SUM;
END;
TEST:=TRUE;
FOR I:=1 TO MP DO IF(XI[K]>0.0) THEN TEST:=FALSE;
IF TEST THEN EXIT5(ITER);
THETA:=1.0E+36;
L:=0;
FOR I:=1 TO M DO IF(XI[K]>0.0) THEN BEGIN
Z:=XI[I]/XI[K];
IF(Z < THETA) THEN BEGIN
THETA:=Z;
L:=I
END
END
END
LISTING 2

A sample data file named "LINTEST.DATA".

EDIT: LIST, B<UILD, M<ODIFY, Q<UIT [1 0] L

LIST WHAT FILE? LINTEST.DATA

STARTING AT WHAT RECORD? 0
0: TEST 3
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 -1.0000
6: 6 ROW 1 COL 1 1.0000
7: 6 ROW 2 COL 1 2.0000
8: 6 ROW 3 COL 1 1.0000
9: 4 COL2 2 -2.0000
10: 6 ROW 1 COL 1 2.0000
11: 6 ROW 2 COL 2 1.0000
12: 6 ROW 3 COL 2 2.0000
13: 4 COL3 3 -3.0000
14: 6 ROW 1 COL 3 3.0000
15: 6 ROW 2 COL 3 5.0000
16: 6 ROW 3 COL 3 1.0000
17: 4 COL4 4 1.0000
18: 6 ROW 1 COL 4 0.0000
19: 6 ROW 2 COL 4 0.0000
20: 6 ROW 3 COL 4 1.0000
21: 99 LOGICAL EOF

EDIT: LIST, B<UILD, M<ODIFY, Q<UIT [1, 0] Q

---

**THE MM-103 DATA MODEM AND COMMUNICATIONS ADAPTER**
S-100 bus compatible

<table>
<thead>
<tr>
<th><strong>FCC APPROVED</strong></th>
<th><strong>HIGH QUALITY</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Both the modem and telephone system interface are FCC approved, accomplishing all the required protective functions with a miniaturized, proprietary protective coupler.</td>
<td>-50 dBm sensitivity. Auto answer. Auto originate. Auto dialer with computer-controlled dial rate. 61 to 300 baud (anywhere over the long-distance telephone network), rate selection under computer control. Flexible, software-controlled, maskable interrupt system.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>WARRANTY</strong></th>
<th><strong>ASSEMBLED &amp; TESTED</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>One year limited warranty. Ten-day unconditional return privilege. Minimal cost, 24-hour exchange policy for units not in warranty.</td>
<td>Not a kit! (FCC registration prohibits kits)</td>
</tr>
</tbody>
</table>

**LOW PRICE—$359.95—** For Modem AND Coupler plus shipping & handling

---

Potomac Micro-Magic, Inc.

Write for brochure:
First Lincolnia Bldg., Suite B1
4810 Beauregard St.
Alexandria, Va. 22312

Call for further information:
VOICE: (703) 750-3727
MODEM: (703) 750-0930 (300 baud)
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

<table>
<thead>
<tr>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
<th>COL4</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.00000</td>
<td>-2.00000</td>
<td>-3.00000</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

RHS VECTOR

<table>
<thead>
<tr>
<th>ROW1</th>
<th>ROW2</th>
<th>ROW3</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.0000</td>
<td>20.0000</td>
<td>10.0000</td>
</tr>
</tbody>
</table>

ABAR ARRAY

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL</th>
<th>ABAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1.00000</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2.00000</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3.00000</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1.00000</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2.00000</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3.00000</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3.00000</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4.00000</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5.00000</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3.00000</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4.00000</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5.00000</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
<th>COL4</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.00000</td>
<td>-2.00000</td>
<td>-3.00000</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

LIST & X ARRAYS

<table>
<thead>
<tr>
<th>ROW</th>
<th>COL</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>15.0000</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>20.0000</td>
</tr>
</tbody>
</table>
### COMING IN FUTURE ISSUES of S-100 MICROSYSTEMS

Linear Programming Techniques In Pascal (Part 2)
Introduction To CP/M (Part 3)
North Star Topics
DynaTrace - An 8080 Emulator
A Pascal Monitor Program
Spooling For the North Star
Cursor Addressing For Memory Mapped Video Displays
Utilities For The Sol Computer
More Hardware Mods For the Tarbell Disk Controller
Cursor Addressing For CRT Terminals (Part 2)
Handling Look-up Tables

**PLUS ...**

- news...views...product reviews...software
catalog ...letters to the editor...new products.....club
directory.....CBBS systems around the country.

  don't miss S-100 MICROSYSTEMS.....the magazine for the serious microcomputer user.

---

<table>
<thead>
<tr>
<th>ROW</th>
<th>M+1</th>
<th>M+2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>7</td>
<td>10.0000</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>0.00000</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>-45.0000</td>
</tr>
</tbody>
</table>

**START PHASE 1**

**ITERATION 1 OF TEST**

**LIST & X ARRAYS**

<table>
<thead>
<tr>
<th>ROW</th>
<th>M+1</th>
<th>M+2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>3.0000</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4.00000</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>6.00000</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>12.0000</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>-9.00000</td>
</tr>
</tbody>
</table>

**ITERATION 2 OF TEST**

**LIST & X ARRAYS**

<table>
<thead>
<tr>
<th>ROW</th>
<th>M+1</th>
<th>M+2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2.14286</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>3.57143</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>2.14286</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>15.0000</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>-2.14286</td>
</tr>
</tbody>
</table>

- CONTINUED ON PAGE 33 -
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/write 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 to the injurious effects of abrasion. This is not to say that abrasion 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/write 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/write 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:
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 hardware 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 alphanumerical characters and CONTROL functions. The lowercase alphabet is accepted, but it is converted to uppercase 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 lfeed 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 anywhere 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 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 are used 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 separate 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:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BINARY</td>
<td>Used for storage of MEMORY IMAGES of programs in machine code.</td>
</tr>
<tr>
<td>ASCII</td>
<td>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)</td>
</tr>
<tr>
<td>HEX FORMAT</td>
<td>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.</td>
</tr>
<tr>
<td>RELOCATABLE</td>
<td>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.</td>
</tr>
</tbody>
</table>

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 1011 0101

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 occurred. 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 file 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 function. Below is a list of possible primary names:

<table>
<thead>
<tr>
<th>PRIMARY NAME</th>
<th>REPRESENTATION IN FCB</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEXTEDIT</td>
<td>TEXTEDIT</td>
</tr>
<tr>
<td>BASIC</td>
<td>BASIC</td>
</tr>
<tr>
<td>FASCA785</td>
<td>FASCA78</td>
</tr>
<tr>
<td>F-80</td>
<td>F-80</td>
</tr>
<tr>
<td>1234</td>
<td>1234</td>
</tr>
</tbody>
</table>

The following names are not allowed.

<table>
<thead>
<tr>
<th>PRIMARY NAME</th>
<th>REASON</th>
</tr>
</thead>
<tbody>
<tr>
<td>LETTER?D</td>
<td>? IS RESERVED</td>
</tr>
<tr>
<td>JACOB E</td>
<td>SPACE IN NAME</td>
</tr>
<tr>
<td>REN</td>
<td>REN IS A CPM FUNCTION</td>
</tr>
</tbody>
</table>

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 RESEARCH 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:

<table>
<thead>
<tr>
<th>NAME</th>
<th>DATA TYPE</th>
<th>USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COM</td>
<td>BINARY</td>
<td>PROGRAMS THAT CAN BE EXECUTED</td>
</tr>
<tr>
<td>HEX</td>
<td>HEX FORMAT</td>
<td>OBJECT OF ASSEMBLERS OR COMPILERS</td>
</tr>
<tr>
<td>ASM</td>
<td>ASCII</td>
<td>SOURCE CODE FOR ASSEMBLERS</td>
</tr>
<tr>
<td>MAC</td>
<td>ASCII</td>
<td>SOURCE CODE FOR MACRO-ASSEMBLERS</td>
</tr>
<tr>
<td>BAS</td>
<td>ASCII</td>
<td>SOURCE CODE FOR BASIC COMPILERS</td>
</tr>
<tr>
<td>FOR</td>
<td>ASCII</td>
<td>SOURCE CODE FOR FORTRAN COMPILERS</td>
</tr>
<tr>
<td>PRN</td>
<td>ASCII</td>
<td>PRINTOUTS OF TEXT FORMATTERS, COMPILERS, ASSEMBLERS, ETC.</td>
</tr>
<tr>
<td>SSS</td>
<td>ASCII</td>
<td>SOURCE FOR SUBMIT UTILITY</td>
</tr>
<tr>
<td>LIB</td>
<td>ASCII</td>
<td>TEMPORARY FILE MAY BE ANY FORMAT</td>
</tr>
<tr>
<td>TEX</td>
<td>ASCII</td>
<td>LIBRARY FILES</td>
</tr>
<tr>
<td>DOC</td>
<td>ASCII</td>
<td>ASCII FOR TEXT-FORMATTERS</td>
</tr>
<tr>
<td>MSG</td>
<td>ASCII</td>
<td>MESSAGES OR DOCUMENTATION</td>
</tr>
<tr>
<td>TXT</td>
<td>ASCII</td>
<td>SAME AS DOC</td>
</tr>
<tr>
<td>REL</td>
<td>RELOCATABLE</td>
<td>OBJECT OF RELOCATING ASSEMBLERS-</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>FILE NAME</th>
<th>FCB REPRESENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASIC.COM</td>
<td>BASIC...COM</td>
</tr>
<tr>
<td>FORTRANB.DOC</td>
<td>FORTRANBDOC</td>
</tr>
<tr>
<td>LETTER.1</td>
<td>LETTER...1...</td>
</tr>
</tbody>
</table>

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

005D42 41 53 49 43 20 20 20 43 4F 40 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 understanding of this concept.

ASM1.COM can describe - ASM1.COM or ASM2.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 - PORTRAN.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 listed above, but block numbers 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 phenomenon 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/secr = 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 256 144 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

This leaves 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 16384 bytes in length as calculated above, then total storage would appear to be 64 * 16384 or 1048576 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 246784 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 give 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:


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

WRITE FOR S-100 MICROSYSTEMS

We are looking for articles on serious microcomputer applications, reviews of software and hardware, and tutorials on S-100 based systems. Writing for S-100 MICROSYSTEMS is probably easier than you think. Remember, S-100 MICROSYSTEMS is devoted to the serious microcomputer user. Since most of the other microcomputer magazines cater to beginners and game-oriented users, an S-100 article will only appeal to a small fraction of the readers and may easily get lost between games, etc. junk. Therefore an article you write for S100-MICROSYSTEMS will get to the attention of the right people.
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.

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.

<table>
<thead>
<tr>
<th>CHARACTER</th>
<th>ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3 18</td>
</tr>
<tr>
<td>b</td>
<td>6 3</td>
</tr>
<tr>
<td>c</td>
<td>18 72</td>
</tr>
<tr>
<td>d</td>
<td>9 45</td>
</tr>
</tbody>
</table>

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,
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. 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.

<table>
<thead>
<tr>
<th>characters: 1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hazeltine 1500</td>
<td>126</td>
<td>17</td>
<td>column</td>
<td>row</td>
<td></td>
</tr>
<tr>
<td>Lear Siegler ADM-3A</td>
<td>27</td>
<td>61</td>
<td>row</td>
<td>column</td>
<td></td>
</tr>
<tr>
<td>SOL terminal computer</td>
<td>27</td>
<td>2</td>
<td>row</td>
<td>27</td>
<td>1</td>
</tr>
</tbody>
</table>

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 addressing, 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  
100 REM  
110 REM  
120 CLEAR 2000  
130 DIM AS(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 - LARRY STEINER ADM-3A"  
160 PRINT PRINT "2 - BASEL LEW'S 1500"  
170 PRINT PRINT "3 - SOL TERMINAL COMPUTER"  
180 PRINT PRINT "ENTER THE NUMBER OF THE ONE YOU ARE USING"  
190 S=INPUT$(1) :WIDTH 80 : GOTO 290  
200 IF S="1" THEN AM=1 :WIDTH 80 : GOTO 290  
210 IF S="2" THEN AM=2:WIDTH 80:GOTO 290  
220 IF S="3" THEN AM=3:WIDTH 80:GOTO 290  
230 GOTO 140  
240 REM  
250 REM  
260 REM * BEGINNING OF PROGRAM - TITLE  
270 REM  
280 REM  
290 REM  
300 Y=0:X=0 :GOSUB 2160  
310 Y=0:X=0 :GOSUB 2160  
320 Y=0:X=0 :GOSUB 2160  
330 FOR I=1 TO 1600:PRINT 
340 IF I=0 THEN 460  
350 PRINT "ENTER THE CHARACTER YOU WANT TO USE FOR BACKSPACE"  
360 GOSUB 2650:BS=IN:PRINT BS  
370 PRINT "ENTER THE CHARACTER YOU WANT TO USE FOR FORWARD SPACE"  
380 GOSUB 2650:FS=IN:PRINT FS  
390 PRINT "ENTER THE CHARACTER YOU WANT TO USE FOR INSERTING"  
400 GOSUB 2650:IT=IN:PRINT IT  
410 PRINT "ENTER THE CHARACTER YOU WANT TO USE FOR DELETING"  
420 GOSUB 2650:DE=IN:PRINT DE  
430 GOSUB 2650:FS=IN:PRINT FS  
440 GOSUB 2650:BS=IN:PRINT BS  
450 GOSUB 2650:IT=IN:PRINT IT  
460 GOSUB 2650:DE=IN:PRINT DE  
470 S=INPUT$(1):PRINT S  
480 IF S="Y" OR S="y" THEN 3100  
490 PRINT "DO YOU WANT TO USE A PREVIOUSLY SAVED LABEL (YIN) "  
500 S=INPUT$(1):PRINT S  
510 IF S="Y" OR S="y" THEN 2750  
520 REM  
530 REM  
540 REM  
550 REM  
560 REM  
570 REM  
580 REM  
590 REM  
600 REM ** DISPLAY LEFT SIDE OF SCREEN **  
610 REM  
620 REM  
630 GOSUB 2160  
640 PRINT  
650 FOR N=1 TO 16  
660 Y=N:X=0:GOSUB 2160  
670 N=STR$(N):IF LEN(N)=2 THEN NS=" ""NS  
680 PRINT "LINE "NS#:"""TAB(MD+14):"""  
690 NEXT N  
700 REM  
710 REM  
720 REM  
730 REM  
740 REM  
750 REM  
760 Y=N:X=12:GOSUB 2160  
770 PRINT AS(N)  
780 NEXT N  
790 FOR N=1 TO LN  
800 1=0  
810 IF AS(N)=" " THEN 820  
820 FOR M=1 TO LN  
830 IF I=0 THEN 850  
840 Y=N:X=0:GOSUB 2160  
850 GOSUB 2650  
860 IF S="Y" OR S="y" THEN 3100  
870 IF N=LS THEN 1600 : REM MOVE CURSOR TO THE LEFT  
880 IF N=LS THEN 1600 : REM MOVE CURSOR TO THE RIGHT  
890 IF N=LS THEN 1600 : REM DELETE CHARACTER  
900 IF N=LS THEN 1600 : REM INSERT CHARACTER  
910 1=0  
920 N=LN(2,L,1):=IN$  
930 PRINT  
940 IF S="Y" OR S="y" THEN 3100  
950 NEXT N  
960 AS(N)=LS  
970 PRINT AS(N)  
980 PRINT N  
990 RETURN  

1650 PRINT "DO YOU WANT TO PRINT MORE LABELS (Y/N)?" ;
1660 IF IN=INP(28) THEN 1670
1670 IF IN="Y" OR IN="Y" THEN 1700
1680 STOP
1690 GOTO 1680
1700 PRINT "DO YOU WANT TO PRINT THE SAME LABEL (Y/N)?" ;
1710 IF IN=INP(28) THEN 1720
1720 IF IN="Y" OR IN="Y" THEN 350
1730 GOSUB 2570
1740 GOTO 1680
1750 REM *
1760 REM ROUTINE TO MOVE CURSOR RIGHT AND LEFT
1770 REM *
1780 REM *
1790 REM *
1800 IF IN=INP(28) THEN 1810
1810 IF IN="S" OR IN="S" THEN 1700
1820 IF IN="S" OR IN="S" THEN 1700
1830 GOTO 1680
1840 GOTO 1680
1850 REM *
1860 REM ROUTINE TO HANDLE CURSOR AT END OF FIELD
1870 REM *
1880 REM *
1890 REM *
1900 GOSUB 2650
1910 IF IN=INP(28) THEN 1920
1920 IF IN="S" OR IN="S" THEN 1930
1930 GOTO 1900
1940 REM *
1950 REM *
1960 REM *
1970 REM *
1980 REM *
1990 HIDE(IS,M,M+1) = "MID(S,M,M+1)" ;
2000 PRINT HIDE(IS,M,M+1) ;
2010 GOTO 1680
2020 REM *
2030 REM *
2040 REM *
2050 REM *
2060 REM *
2070 L1 = HIDE(IS,M,M+1) ;
2080 PRINT HIDE(IS,M,M+1) ;
2090 GOTO 1680
2100 GOTO 1680
2110 REM *
2120 REM *
2130 REM *
2140 REM *
2150 REM *
2160 REM *
2170 REM *
2180 REM *
2190 REM *
2200 REM *
2210 REM *
2220 IF X+10 THEN PRINT CHR$(26) ; RETURN
2230 PRINT CHR$(27) + CHR$(61) + CHR$(31+Y) + CHR$(31+X) + "RETURN"
2240 REM *
2250 REM *
2260 REM *
2270 REM *
2280 REM *
2290 REM *
2300 IF X+10 THEN PRINT CHR$(126) + CHR$(17) ;
2310 REM *
2320 REM *
2330 REM *
2340 REM *
2350 REM *
2360 IF X+10 THEN PRINT CHR$(126) + CHR$(17) + "RETURN"
2370 IF X+10 THEN PRINT CHR$(126) + CHR$(17) + "RETURN"
2380 REM *
2390 REM *
2400 REM *
2410 REM *
2420 REM *
2430 REM *
2440 REM *
2450 REM *

2460 LINEINPUT "ENTER LABEL WIDTH (IN CHARACTERS) \* \+M $ F" ;
2470 MD=VAL(M$)
2480 LINEINPUT "ENTER NUMBER OF PRINT LINES PER LABEL \+L $ N" ;
2490 LN=VAL(L$)
2500 ERASE A$
2510 DIM AS(LN)
2520 FOR M=1 TO LN
2530 AS(M) = STRING$(MD,32)
2540 NEXT M
2550 LINEINPUT "ENTER NUMBER OF LABELS TO SKIP BETWEEN LABELS \+K $ F" ;
2560 SK=VAL(K$)
2570 LINEINPUT "ENTER TOTAL NUMBER OF LABELS TO BE PRINTED \*N $ B" ;
2580 NB=VAL(B$)
2590 RETURN
2600 REM *
2610 REM *
2620 REM *
2630 REM *
2640 REM *
2650 OUT 29,1
2660 WAIT 29,1,0
2670 INPUT "ENTER NUMBER OF LABELS TO BE PRINTED \*N $ B" ;
2680 INS-CHR$(14)
2690 RETURN
2700 REM *
2710 REM *
2720 REM *
2730 REM *
2740 REM *
2750 V=O-X;GOTO 2160
2760 PRINT "ENTER DISKETTE NUMBER XXXX" ; STRING$(4,8)
2770 LINEINPUT DS
2780 PRINT "ENTER UNIT NUMBER XXXX" ; STRING$(4,8)
2790 LINEINPUT US
2800 PRINT "ENTER DATE (MM/DD/YY) XX/XX/XX" ; STRING$(8,8)
2810 LINEINPUT DT
2820 PRINT "ENTER DEALER NAME XXXXXXXXXX XXXXXX XXXXXX XXXXXX XXXXXX" ;
2830 LINEINPUT DL
2840 IF LEN(DL$) >10 THEN PRINT "DEALER NAME TOO LONG!" ; GOTO 2820
2850 EXIT AS
2860 DIM AS(LN)
2870 AS(I)=" PRODIGY SYSTEMS, INC." ;
2880 AS(I)=" DISKETTE #"+DS$+
2890 AS(I)=" DATE +DT$
2900 AS(I)=" DEALER NAME "+DL$;
2910 AS(I)=" MASTER DISKETTE - RETURN IMMEDIATELY"
2920 AS(I)=" COPYRIGHT (C) 1979 PRODIGY SYSTEMS, INC." ;
2930 AS(I)=" ALL WORLDWIDE RIGHTS RESERVED"
2940 AS(I)=" PRODIGY DISKETTE LABELS TO BE PRINTED \*+N $ B" ;
2950 ND=VAL(N$)
2960 KOS=W+40
2970 WOL=W+40
2980 KOS=W+40
2990 LNS=0
3000 LNS=0
3010 NBR=LNS
3020 NW=VAL(N$)
3030 GOTO 1390
3040 STOP
3050 REM *
3060 REM *
3070 REM *
3080 REM *
3090 REM *
3100 PRINT "ENTER DRIVE ON WHICH DISKETTE LABELS ARE STORED \+A, B, C, D\ *+D $ F" ;
3110 IF IN=INP(28) THEN 3130
3120 IF IN="A" OR IN="D" THEN 3130
3130 FS=IN$+"\*LAB"
3140 PRINT
3150 FILES FS
3160 PRINT
3170 PRINT "ENTER A FILE NAME FROM THE ABOVE LIST";
3180 LINEINPUT "USE FILE NAME ONLY, NO EXTENSION \*T $ B" ;
3190 FS=IN$+"\*LAB"
3200 OPEN "1",1,FS
3210 INPUT BS,CS
3220 WOS-=5;VAL(WOS)
3230 LN=CS;VAL(LN$)
3240 ERASE AS

Continued on Page 54
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 200,000 "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 problem 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 PHANTOM*) was used for as many as eight different functions by various S-100 manufacturers.

First of all let's 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

<table>
<thead>
<tr>
<th>IEEE S-100</th>
<th>ORIGINAL S-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Designed to support more powerful 8-bit (280) and 16-bit (e.g. 28000) mpus</td>
<td>1. Designed specifically for 8080</td>
</tr>
<tr>
<td>2. Operating speed up to 10MHz</td>
<td>2. Operating speed - 2 MHz</td>
</tr>
<tr>
<td>3. 16 Megabyte direct memory addressing range</td>
<td>3. 65K direct memory addressing range</td>
</tr>
<tr>
<td>4. Up to 65K I/O ports</td>
<td>4. Up to 256 I/O ports</td>
</tr>
<tr>
<td>5. Up to 16 Masters on bus</td>
<td>5. Single Master operation</td>
</tr>
<tr>
<td>6. 8 or 16-bit data transfers</td>
<td>6. Only 8-bit data transfers</td>
</tr>
<tr>
<td>7. 10 vectored interrupts plus NMI</td>
<td>7. 8 vectored interrupts</td>
</tr>
<tr>
<td>8. Phantom</td>
<td>8. Phantom</td>
</tr>
<tr>
<td>9. Three more ground lines</td>
<td>9. Three more ground lines</td>
</tr>
<tr>
<td>10. 3 undefined lines</td>
<td>10. 3 undefined lines</td>
</tr>
<tr>
<td>11. 5 reserved for future use lines</td>
<td>11. 5 reserved for future use lines</td>
</tr>
</tbody>
</table>
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" letter. Note also that an active low state is now indicated 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>
<thead>
<tr>
<th>TABLE #2</th>
<th>DATA BUS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IEEE S-100</strong></td>
<td><strong>OLD S-100</strong></td>
</tr>
<tr>
<td>8-bit Master</td>
<td>16 lines</td>
</tr>
<tr>
<td>8-Data In</td>
<td>16-bit master</td>
</tr>
<tr>
<td>8-Data Out</td>
<td>8-Data In</td>
</tr>
<tr>
<td>8-Bidirectional</td>
<td>&amp;</td>
</tr>
<tr>
<td>8-Bidirectional</td>
<td>or</td>
</tr>
<tr>
<td>8-Data Out</td>
<td>8-Data Out</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE #3</th>
<th>ADDRESS BUS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IEEE S-100</strong></td>
<td><strong>OLD S-100</strong></td>
</tr>
<tr>
<td>16 or 24 lines</td>
<td>16 lines</td>
</tr>
<tr>
<td>(A0-A23)</td>
<td>(A0-A159)</td>
</tr>
<tr>
<td>16 Megabytes memory</td>
<td>65K bytes memory</td>
</tr>
<tr>
<td>directly addressable</td>
<td>directly addressable</td>
</tr>
<tr>
<td>I/O addressing</td>
<td>I/O addressing</td>
</tr>
<tr>
<td>up to 65K ports</td>
<td>up to 256 ports</td>
</tr>
<tr>
<td>(A0-A7 &amp; A8-A15)</td>
<td>(A0-A7 or A8-A15)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE #4</th>
<th>STATUS BUS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IEEE S-100</strong></td>
<td><strong>OLD S-100</strong></td>
</tr>
<tr>
<td>Memory Read</td>
<td>sMEMR</td>
</tr>
<tr>
<td>Op-Code Fetch</td>
<td>sM1</td>
</tr>
<tr>
<td>Input</td>
<td>sINP</td>
</tr>
<tr>
<td>Output</td>
<td>sOUT</td>
</tr>
<tr>
<td>Write Cycle</td>
<td>sWO*</td>
</tr>
<tr>
<td>Interrupt Acknowledge</td>
<td>sINTA*</td>
</tr>
<tr>
<td>Halt Acknowledge</td>
<td>sHLTA</td>
</tr>
<tr>
<td>Memory Write</td>
<td>MWRT</td>
</tr>
<tr>
<td>16-bit data transfer</td>
<td>sXTRQ*</td>
</tr>
<tr>
<td>9 signals</td>
<td>11 signals</td>
</tr>
<tr>
<td>deleted</td>
<td></td>
</tr>
<tr>
<td>STACK - stack address on bus</td>
<td></td>
</tr>
<tr>
<td>RUN - CPU in run mode</td>
<td></td>
</tr>
<tr>
<td>SS - single step operation</td>
<td></td>
</tr>
</tbody>
</table>
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 P1 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

<table>
<thead>
<tr>
<th>IEEE S-100</th>
<th>Old S-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 lines</td>
<td>5 lines</td>
</tr>
<tr>
<td>Indicates start of new bus cycle</td>
<td>PSYNC</td>
</tr>
<tr>
<td>Read Strobe</td>
<td>pBIN</td>
</tr>
<tr>
<td>Write Strobe</td>
<td>pWR</td>
</tr>
<tr>
<td>Hold Acknowledge</td>
<td>pHLDA</td>
</tr>
<tr>
<td>Address and Status</td>
<td>pSTVAL</td>
</tr>
</tbody>
</table>

**TABLE #6**

CONTROL INPUT BUS

<table>
<thead>
<tr>
<th>IEEE S-100</th>
<th>Old S-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 lines</td>
<td>4 lines</td>
</tr>
<tr>
<td>Used by slaves to sync master speed to slave speed</td>
<td>PRDY</td>
</tr>
<tr>
<td>Used by front panel to stop or single step master</td>
<td>XRDY</td>
</tr>
<tr>
<td>Interrupt request to master</td>
<td>INT*</td>
</tr>
<tr>
<td>Used by temporary master to request control of bus</td>
<td>HOLD*</td>
</tr>
<tr>
<td>Nonmaskable Interrupt request NMI*</td>
<td>SIXTN*</td>
</tr>
</tbody>
</table>

**TABLE #7**

VECTORED INTERRUPT BUS

<table>
<thead>
<tr>
<th>IEEE S-100</th>
<th>Old S-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 lines</td>
<td>8 lines</td>
</tr>
<tr>
<td>Inputs to an interrupt controller circuit which arbitrates among 8 inputs and asserts INT*</td>
<td>VI0* VI1</td>
</tr>
<tr>
<td>V10 has highest priority</td>
<td>V15</td>
</tr>
<tr>
<td>Asserted when error occurs</td>
<td>ERROR*</td>
</tr>
<tr>
<td>Asserted when impending power failure occurs</td>
<td>PWR FAIL*</td>
</tr>
</tbody>
</table>
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 Ø 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 similarly fallen into dis-use and has been replaced by GND and RFU lines.

On the other hand the ground lines have been 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**

<table>
<thead>
<tr>
<th>New S-100</th>
<th>Old S-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 signals lines</td>
<td>4 signal lines</td>
</tr>
<tr>
<td>Address disable</td>
<td>ADSB*</td>
</tr>
<tr>
<td>Data Out disable</td>
<td>DODSB*</td>
</tr>
<tr>
<td>Status disable</td>
<td>SDB*</td>
</tr>
<tr>
<td>Control disable</td>
<td>CDDSB*</td>
</tr>
<tr>
<td>New lines</td>
<td></td>
</tr>
<tr>
<td>arbitrate among up to 16 masters</td>
<td>DMA0*</td>
</tr>
<tr>
<td>encoded priority requests are</td>
<td>DMA1*</td>
</tr>
<tr>
<td>asserted &amp; lines contain priority no. of highest requestor</td>
<td>DMA2*</td>
</tr>
</tbody>
</table>

### Table #9
**Utility Bus**

<table>
<thead>
<tr>
<th>New S-100</th>
<th>Old S-100</th>
</tr>
</thead>
<tbody>
<tr>
<td>22 lines</td>
<td>34 lines</td>
</tr>
<tr>
<td>Clock signal</td>
<td>CLOCK</td>
</tr>
<tr>
<td>Master clock signal</td>
<td>Ø</td>
</tr>
<tr>
<td>Resets all masters</td>
<td>RESET*</td>
</tr>
<tr>
<td>Resets all slaves</td>
<td>SLAVE CLR*</td>
</tr>
<tr>
<td>Power-on clear</td>
<td>POC*</td>
</tr>
<tr>
<td>Overlays slaves</td>
<td>PHANTOM*</td>
</tr>
<tr>
<td>Not defined</td>
<td>NDEF(3 lines)</td>
</tr>
<tr>
<td>Reserved for future use</td>
<td>RFU(5 lines)</td>
</tr>
<tr>
<td></td>
<td>+8V(2 lines)</td>
</tr>
<tr>
<td></td>
<td>+16V</td>
</tr>
<tr>
<td></td>
<td>-16V</td>
</tr>
<tr>
<td></td>
<td>-16V</td>
</tr>
<tr>
<td></td>
<td>GND(5 lines)</td>
</tr>
<tr>
<td></td>
<td>deleted</td>
</tr>
<tr>
<td></td>
<td>Ø1</td>
</tr>
<tr>
<td></td>
<td>PROT</td>
</tr>
<tr>
<td></td>
<td>UNPROT</td>
</tr>
<tr>
<td></td>
<td>PS - protect status</td>
</tr>
<tr>
<td></td>
<td>SSW-DSB</td>
</tr>
</tbody>
</table>

IEEE S-100 MICROSYSTEMS
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 graphics 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 density color graphics, along with alphanumeric 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 refresh 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

---

CATCH THE S-100 INC. BUS!

<table>
<thead>
<tr>
<th>List Price</th>
<th>Our Special Cash Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Godbout 4 MHz Econoram VIIA-24K Unkit</td>
<td>419.00</td>
</tr>
<tr>
<td>S.D. Systems SBC-100 Single Board Computer Kit</td>
<td>295.00</td>
</tr>
<tr>
<td>SSM CB-2 Z-80 CPU Kit</td>
<td>210.00</td>
</tr>
<tr>
<td>Morrow's Thinker Toys Discus II D Dual Density 8'' Disk Drive System</td>
<td>1,149.00</td>
</tr>
<tr>
<td>Anadex DP 8000 Printer w/U/L case</td>
<td>995.00</td>
</tr>
</tbody>
</table>

Subject to Available Quantities • Prices Quoted Include Cash Discounts. Shipping & Insurance Extra. We carry all major lines such as S.D. Systems, Cromemco, Ithaca Intersystems, North Star, Sanyo, ECT, TEI, Godbout, Thinker Toys, Hazeltine, IMC For a special cash price, telephone us.

Bus ....... S-100, inc.
Address ..... 7 White Place
Clark, N.J. 07066
Interface ..... 201-382-1318
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 5847 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 the 74LS93 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 and four 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/semigraphic 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 requested operation, the data sent to this port is immaterial, and is ignored.

Reading port-0 provides you with some status information 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 inputting 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 was 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

S-100 MICROSYSTEMS
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 the 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 hardware 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-O 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 remove 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.

### THE S-100 MICROSYSTEMS SOFTWARE CATALOG

Beginning next month we will start cataloging S-100 system software. If you have a software package you are offering for sale and want a free ad then send us the information in the following form.

Program Name:
Hardware System:
Minimum Memory Size:
Language:
A short description (120 words maximum)
When released:
Price:
What is included with price:
Author:
Where to purchase it:

All information should be included. We reserve the right to edit and/or reject any submission.
To use the Tarbell floppy disk controller 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 cannot 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 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 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 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 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, to reset on the falling edge of PDBIN. "Remembering" SMER while it is disabled by STATDSB, FHLD 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 switch 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 video 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 initiated on the trailing, rather than the leading edge of POC. Or gate U46 was therefore added to prevent Master Reset on booting and the reset circuit was simplified to use only POC instead of Preset and ExtClr as well. U57 (upper left) provides a buffered 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. 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.
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 jumping 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 normal 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.
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
; FREQUENTLY CHANGED PARAMETERS
TRUE EQU OFFFH ; USED IN IF STATEMENTS
FALSE EQU NOT TRUE ; USED IN IF STATEMENTS
PTVM EQU TRUE ; YMP FOR CONOT?
OHDID EQU FALSE ; USE IF ONLY HAVE I DRIVE
TEST EQU TRUE ; TESTING AIDS
MEZZ EQU 47 ; MEMORY SITE
MODIF EQU 'c' ; MODIF LEVEL OF DRIVER
NUMROW EQU 'c' ; NUM MODIFICATION LEVEL
MTSYS EQU TRUE ; SPECIAL STUFF FOR MY SYSTEM

; BASE FOR CONTROLLER PORTS
DBASE EQU GFEM
DCOM EQU DBASE+2
DESTAT EQU DBASE+2+3
DTRK EQU DBASE+2+4
DUCT EQU DBASE+2+5
DDATA EQU DBASE+2+6
DWAIT EQU DBASE+2+7
DCONT EQU DBASE+2+8

; SPECIAL STUFF FOR MY SYSTEM
DBASE+2 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+3 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+4 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+5 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+6 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+7 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+8 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+9 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+10 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+11 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+12 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+13 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+14 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+15 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+16 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+17 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+18 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+19 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+20 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+21 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+22 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+23 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+24 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+25 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+26 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+27 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+28 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+29 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+30 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+31 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+32 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+33 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+34 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+35 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+36 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+37 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+38 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+39 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+40 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+41 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+42 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+43 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+44 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+45 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+46 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+47 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+48 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+49 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+50 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+51 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+52 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+53 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+54 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+55 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+56 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+57 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+58 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+59 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+60 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+61 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+62 ; SPECIAL STUFF FOR MY SYSTEM
DBASE+63 ; SPECIAL STUFF FOR MY SYSTEM

; DISK COMMANDS
CSRC EQU DBASE+33068 ; START OF CP/M SYSTEM
DBCS EQU CSRC+31068 ; START OF BOS FOR 4.1
CMML EQU S-CMP ; LENGTH OF CP/M W/O BIOS
NSEXTS EQU CMML/128 ; # SECT CP/M EXCL. BIOS
NCTOTS EQU 16 ; #RET COUNT ON ERRORS

; FROM COLD START LOADER
JMP BOOT ; FROM COLD START LOADER
XWRD JMP WBOOT ; FROM WARM START
XCONOT JMP CONOT ; CHECK CONSOLE PORT
XCONIN JMP CONIN ; GET CONSOLE CHARACTER
XRCNT JMP CONOT ; WRITE CONSOLE CHARACTER
XLIST JMP LIST ; WRITE CHAR ON LIST DEV
XPRCM JMP PUNCH ; PUNCH CHAR ON PAPER TAPE
XKDR JMP USER ; READ CHAR FROM PAPER TAPE
XROYE JMP RORE ; RESPONSE READY TO BF
XSEKEL JMP DKSREL ; SELECT DISK DRIVE DROK ; SELECT DISK
XTRK JMP TRK ; SET TRACK NUMBER
XTSKSEC JMP TSECT ; SET SECTOR NUMBER
XTSTOT JMP TSTEMA ; SET DMA ADDRESS
XREAD JMP READ ; READ SELECTED SECTOR
XWRITE JMP WRITE ; WRITE SELECTED SECTOR

; ENTER HERE FROM COLD START LOADER
; GIVE SIGN ON MESSAGE
; BOOT:
LXI SP,08H ; SET STACK POINTER
XRA A ; CLEAR A
STA 4 ; FORCE DISK A ; IF MYSYS OR PTVM
GUT OCSH ; RESET VDM
MVI C,0CH ; SET UP IT TO CLEAR SCREEN
CALL XCONOT ; SEND IT OUT
ENDIF

; PRINT OPENING MESSAGE
; JUMP TO MSG
LXI H,MSG ; PRINT OPENING MESSAGE
JMP GDHCP ; DO PRINTOUT
; DO NOW DO HOUSEKEEPING
;
; WARM BOOT ROUTINE--ACTIVATED WHEN YOU CONTROL C
;
; WBOOT:
LXI SP,08H ; SET STACK POINTER
MVI C,0 ; USED BY DESKEL AND SITRK
CALL XDEKSEL ; SELECT DISK 0
MVI D,NSEXTS ; # OF SECTORS IN D
MVI B,2 ; START WITH SECTOR 2
LXI H,CCP ; GET STARTING ADDN
WBOOT: CALL SITRK ; SELECT TRACK (C)
PUSH B ; SAVO SC
MOV C,B ; PUT SECTOR IN C
CALL XDESEC ; READ FROM TRK
MOV B,E ; GET READY FOR SET DMA
MOV C,L ;
STA DISKNO ;STORE IT FOR LATER
ADI A,'A' ;MAKE IT ASCII LETTER
MOV C,A ;PUT IN C FOR CONOT
CALL XCONOT ;PRINT IT
CALL XCONIN ;WAIT FOR GO AHEAD
POP E ;RESPONSE SC
XRA A ;CLEAR A
JMP SEXIT ;EXIT SEXIT

ENDIF

MOV S,N ;PUT OLD DISK # IN DE
MVI D,0 ;CLEAR D FOR DAO
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 L,C ;GET NEW DRIVE #
LXI H,TRKTB ;GET ADDR OF TRACK TABLE
DAD D ;ADD DISK # TO ADDR
MOV A,N ;AND PUT IN REG N
STA HLD ;SAVE FOR SEEK ROUTINE
OUT DTRK ;ALOUD 1771
MOV A,C ;GET DISK BEAU
STA DISKNO ;STORE IT FOR LATER USE
CNA ;INSERT BIT FOR LATCH
ADD A ;PUT BITS 0-1 AT 4-5
ADD A ;
ADD A ;
CRI 2 ;MAKE LATCH COMMAND
OUT DCONT ;SET LATCH WITH CODE

SEXIT:
POP B ;RESTORE REGS
POP D ;RESTORE REGS
POP H ;RESTORE REGS
RET ;

;SET THE TRACK GIVEN IN REG C
;SETRK:
MOV A,C ;THE NAS IN REG C
STA TRA ;PUT IT WHERE IT CAN BE FOUND
RET ;

;SET DISK SECTOR NUMBER
;SECT:
MOV A,C ;GET SECTOR NUMBER
STA SECT ;PUT AT SECTOR # ADDR
RET ;

;SET DISK DMA ADDRESS
;SETDMA:
MOV H,B ;MOV DC TO HL
MOV H,0 ;
SRL DISKDMA ;SAVE IT
RET ;

;READ A SECTOR AT SECT, SEEK THE NEEDED TRACK
;USE STARTING ADDRESS AT DMAAD
;READ:
CALL SEEK ;MOVE HEAD TO TRACK
MVI A,'A' ;SETUP READ W/O READ LOAD
RET ;

;READ:
MVI A,'A' ;READ
JMP LREAD ;IF LOADED - THEN DO IT
ORI $ ;ELSE FORCE READ LOAD

READE:
OUT DCOM ;SEND COMMAND TO 1771

RLOOP:
CALL FREAD ;READ A SECTOR

READONE:
EI ;ALLOW INTERRUPTS AGAIN
IN DSTAT ;READ DISK STATUS
ANI 9BH ;LOOK AT ERROR BITS
HI ;RETURN IF NONE

CHECK:
STAT ERRS ;SAVE ERROR BYTE

;IF TEST=
;FF INVERTED LOGIC
CMA
OUT DCOM ;SET THE LIMELIGHT

RDONE:
CALL ERCHK ;CHECK FOR SEEK ERROR
LDA ECCNT ;SET ERROR CNT
DCM A ;RECEIVE COUNT
JNZ RRETRY ;TRY TO READ AGAIN
MVI A,'A' ;SHOW READ ERROR
JMP ERMSG ;TELL SOMEONE

READ:
IN DWAIT ;WAIT FOR DMA OR INTRO
ORA A ;SET FLAGS
RP ;DON'T IF INTR
IN DRATA ;READ A BYTE FROM DISK
MOV M,A ;PUT BYTE IN MEMORY
INX H ;INCH MEMORY POINTER
JMP FREAD ;KEEP READING

/FRAED:
;WRITE THE SECTOR AT SECT--LOAD READ FIRST
;USE STARTING ADDRESS AT DMAAD
;WRITE:
CALL SEEK ;SET ON RIGHT TRACK
MVI A,'ATCH ;GET RETRY COUNT

WRETRY:
STA ERCTH ;SAVE ERROR CNT
LSA SECT ;GET SECTOR #
LDDL DMAAD ;GET STARTING ADDR

WRITE:
OUT DCONT ;SET SECT WTO 1771
CALL RDVCK ;ISSUE COMMAND
DI ;DO NOT ALLOW INTERRUPTS
IN DSTAT ;READ DISK STATUS
ANI 20H ;SETUP WRITE W/O HEAD LOAD
JNZ WRITED ;WRITE ERROR IF HEAD LOAD
ORI $ ;SET FLAGS
ORI 3 ;FORCE WRITE WITH HEAD LOAD

WRITED:
OUT DCOM ;ISSUE COMMAND

WDONE:
CALL WWRITE ;WRITE A SECTOR

WWRITE:
OUT DSTAT ;ALLOW INTERRUPTS AGAIN
IN DSTAT ;READ DISK STATUS
ANI 0FH ;MASK NON-ERROR BITS
RZ ;RETURN IF NO ERRORS
STA ERRS ;SAVE ERROR FLAGS

IF TEST=
CMA
OUT DCOM ;PUT THEM ON FP LEOS

ENDIF

ERRS:
ANI 9BH ;MAKE ERROR BYTE
ANL ERRS ;MASK NON-ERROR BITS
MVI A,'A' ;SHOW WRITE ERROR
JMP ERMSG ;SEND ERRORS

WRITE:
IN DWAIT ;WAIT FOR INTRO
ORA A ;SET FLAGS
RP ;DON'T IF INTR
IN DRATA ;READ A BYTE FROM DISK
MOV M,A ;PUT BYTE IN MEMORY
INX H ;INCH MEMORY POINTER
JMP FREAD ;KEEP READING

/FREAD:

MOV A, $A000 ; GET BYTE FROM MEMORY
OUT DDATA, $99 ; WRITE ON DISK
INX H ; POINT TO NEXT BYTE
JMP TRK ; KEEP WRITING

; READ OR WRITE ERROR DETECTED--HANDLE NO REC FOUND
; CONDITION ELSE NORMAL RETRY LOOP

ERRCH: LDA ERRS ; GET ERROR BYTE
ANI $03H ; MASK FOR MRS
BEQ $ ; NOT MRS

; CHECK TO SEE IF OR CORRECT TRACK
; CORRECT IF NECESSARY

CHKSK: MVI A, $0100 ; SET UP READ ADDR
OUT DCOM, $99 ; COMMAND TO 1771
EI ; DO NOT ALLOW INTERRUPTS
IN SMNT ; WAIT FOR 1ST DRQ (TRK)
IN DDATA ; READ THE TRACK ADDR
EI ; ALLOW INTERRUPTS AGAIN
PUSH PSW ; SAVE TRACK
CALL SMNT ; WAIT FOR OPERATION TO FINISH
POP PSW ; GET TRACK BACK
STA HOLD ; USE IT TO SET UP SEEK MN
OUT DISK ; UPDATE TRACK REGISTER

; TRACK DESIRED HAS ALREADY BEEN STORED BY SETTRK
; SEEK:
CALL SELSK ; WILL DO NOTHING IF FROM CHKSK
LDA TRK ; GET WHERE WE ARE GOING TO
PUSH B ; SAVE BC
MVI B, $0100 ; GET PREV TRACK
CNP B ; COMPARE TO CURRENT
JSF FORINT ; SOME SO FORCE INTERRUPT
MVI A, $0101 ; GET CURRENT IN A
STA HOLD ; SAVE FOR NEXT TIME
POP B ; RESTORE BC
CPI $00 ; SEE IF TRK = 0
JE HOME ; DO HOME ROUTINE INSTEAD
OUT DDATA ; GIVE DESIRED TRK TO 1771
CALL SMNT ; WAIT TILL NOT BUSY
MVI A, $0000 ; SEE-100K
CALL SCMND ; ISSUE COMMAND
CALL SLOOP ; GIVE TIME FOR READ TO SETTLE
RET ;

SCMND: OUT DCOM ; ISSUE COMMAND
SWAIT: CALL ZIP ; WE NEED AT LEAST 12 US
CALL ZIP ; AGAIN, FOR 4 MS
IN DSTAT ; WAIT FOR NOT BUSY, IF INTRO
INH ; SET CARRY IF BUSY
JC SBUY ; STILL BUSY
ZIP: RET ; INTRO RESET BY READING STATUS

FORINT: MVI A, $0000 ; CLEAR ANY PENDING COMMAND
OUT DCOM ; COMMAND TO 1771
OUT DISK ; AND FORCE TYPE 1 STATUS
POP B ; CLEAN UP
RET ; GO BACK

SLOOP: PUSH H ; GIVE TIME FOR READ TO SETTLE
LXI H, $0256 ; NEED ABOUT 10 MS
SLOOP1: DCR L ;
JNZ SLOOP1 ;
DCR H ;
JNZ SLOOP1 ;
POP H ;
RET ;

; HOME ROUTINE-SET TRK TO 1250 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
;
HOMTN: MVI A, $00DH ; SET UP ANY PENDING COMMAND
OUT DCOM ; COMMAND TO 1771
CALL SMNT ; ISSUE COMMAND
MVI A, $0000 ; 10 MS SEEK RATE--HOME
CALL SCMND ; ISSUE COMMAND-WAIT FOR INTRO
CALL SLOOP ; FOR READ SETTLING
RET ; GO BACK

; CHECK FOR DRIVE READY-IF NOT TELL OPERATOR
; AND WAIT FOR CR
;
ROYCK: IN DSTAT ; SET DISK STATUS
ANI $03H ; MASK FOR READY
BEQ $ ; IF WE ARE READY
PUSH H ; SAVE REGS
POP H ; SAVE REGS

ROY1: LXI H, $2000 ; POINT TO MESSAGE
CALL PRMSG ; PRINT IT OUT
LDA DSTAT ; GET CURRENT DRIVE
ADC A, $0F ; CHANGE TO ASCII
MVI C, $300C ; SET UP TO PRINT
CALL XCONOT ; PRINT IT
CALL XCONIN ; GET KEYBD CHAR
LXI H, $CRLF ; GET UP CR AND LF
CALL PRMSG ; PRINT IT
POP B ; SAVE REGS
POP H ; RESTORE REGS
RET ;

; PRINT MESSAGE ROUTINE
;
PRMSG: MVI A, $A ; GET BYTE TO PRINT
ORA A ; GET IF BINARY ZERO
BEQ $ ; IF WE ARE DONE
MVI C, $A ; PASS IT IN C
CALL XCONOT ; PRINT A CHAR
INX H ; POINT TO NEXT BYTE
MVI A, $300C ; SET UP TO PRINT
CALL XCONIN ; PRINT IT OUT
LXI H, $CRLF ; POINT TO NEXT BYTE
CALL PRMSG ; STAY IN LOOP TILL DONE

; ERROR MESSAGE ROUTINES
;
ERRMBG: PUSH H ; SAVE HL
POP B ; SAVE BC
MVI B, $0A ; SAVE TYPE INDICATOR
LXI H, $CRLF ; DO CR/LF
CALL PRMSG ; PRINT IT
MVI C, $A ; SET 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 ;
RCL ; SHIFT BIT INTO CARRY
JNC LOCERR ; NOT IN ERROR-KEEP LOOKING
MVI H, $3 ;
CALL XCONOT ; SET B = 3
CALL XCONOT ; PRINT IT
INX H ;
INX H ; POINT TO NEXT LETTER

; Lookup Table

; 0001
; 0010
; 0011
; 0100
; 0101
; 0110
; 0111
; 1000
; 1001
; 1010
; 1011
; 1100
; 1101
; 1110
; 1111

; ACK: 0
; ACK: 1
; BUSY: 0
; BUSY: 1
; ERR: 0
; ERR: 1
; NO ERR: 0
; NO ERR: 1
; NO SIG: 0
; NO SIG: 1
; OK: 0
; OK: 1
; RLY: 0
; RLY: 1
; DECREMENT COUNTER
JNZ TYPRNT ; KEEP PRINTING LETTER
CALL BLK ; PRINT A BLANK
CALL ERPLT ; SHOW THEM LOCATION
LDA TRK ; GET TRACK WE ARE ON
CALL ERPLT ; AND SHOW IT
LDA DISPOS ; GET CURRENT DRIVE
ADI 'A' ; MAKE IT C
MOV C,A, ; SET UP FOR PRINT
CALL XCONOT ; PRINT IT
POP B ; RESTORE B
POP H ; RESTORE ML
MOV A,1 ; SIGNAL ERROR
ORA A ; SET FLAGS
RET ; GO HOME

ERRRT: PUSH PSW ; SAVE NUMBER
CALL XCONOT ; PRINT LETTER IN C
POP PSW ; GET NUMBER BACK
DECRT: MVI C,'0'-1 ; SET UP C FOR 10'S DIGIT
INR C ; EXTRACT ME DIGIT
SUB 10, ; BY REPEETITIVE SUBTRACTION
JP DECRT ; \IF NOT READY-KEEP WAITING
ADI BS'=6' ; RESTORE TO POSITIVE
MOV B,A, ; SAVE LS DIGIT
CALL XCONOT ; LIST TNS 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

;iCBSID MESSAGES
; MSGT: DB ODH,0AH,'MSC CP/M 81.V4 /
; BTMSG: DB ODH,0AH,'BIT failure','
; CRFL: DB ODH,0AH,0
; NRDYMS: DB ODH,0AH,'NOT READY-DRIVE ',0
; TYPEFL: DB 'IBMYPYNNHCPHELADNQRO' ; \CHECK CONSOLE INPUT STATUS
; NOT READY (A)=0, READY (A)+FF

; CONOT:
IF MUSYS:
IN 2 ; READ CONSOLE STATUS
ANI BSH ; LOOK AT BIT ?
ENDIF : ;
 IF NOT MUSYS :
 IN 0 ; LOOK AT BIT 0
 ANI 1 ;
 ENDIF : ;
 MVI A,0 ; SET A=0
RZ ; 
 RET ; ;
; READ A CHARACTER FROM CONSOLE
CONIN: ;
 IF MUSYS:
 IN 2 ; READ CONSOLE STAT
 ANI BSH ; LOOK AT BIT ?
 JNZ CONTIN ; KEEP WAITING
 IN 0 ;
 ANI 7FH ; TURN PARITY OFF
 RET ;
 IF NOT MUSYS:
 IN 0 ; READ CONSOLE STAT
 ANI BSH ; LOOK AT BIT ?
 RET ;
...Continued from Page 38

3370 DIM A$ (LN)
3374 FOR N = 1 TO LN
3378 LINE INPUT H, 0$
3380 A$ (N) = D$ + STRING$ (WD-LEN(D$) , 32)
3384 NEXT N
3388 CLOSE
3392 GOSUB 2550
3396 GOTO 630
3400 REM
3404 IN THIS LISTING, "I" MEANS "IS GREATER THAN"
3408 REM "I" MEANS "IS LESS THAN"
3412 REM
3416 PRINT "HI THERE"

CURSOR
Continued from Page 38

3350 DIM A$ (LN)
3360 FOR N=1 TO LN
3370 LINE INPUT H1,0$
3380 A$ (N) = H1 + STRING$ (LEN(H1) , 0$
3390 NEXT N
3400 REM
3404 IN THIS LISTING, "I" MEANS "IS GREATER THAN"
3408 REM "I" MEANS "IS LESS THAN"
3412 REM
3416 PRINT "HI THERE"
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, IL

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
310 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

Advertiser Page
Ackerman Digital Systems 13
Computer Design Labs 11
Computer Mart of New Jersey 31
Digital Graphic Systems 36
Electronic Control Technology 7
Godbout Electronics 60
Ithaca Intersystems 5
Lifeboat Associates 9
Morrow Designs/Thinker Toys 2
Potomac Micro-Magic 25
S-100 43
S-100 Microsystems 57
Tec Mar 15
Trenton Computer Festival 59
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 No. P694/Dll). 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), Labeled Languages (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.
The only magazine by and for S-100 system users!

Take advantage of this limited time charter subscription offer!

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

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

*Registered Trademark Digital Research

Take advantage of our low charter subscription offer right now! This offer runs out on April 30, 1980.

Send me ___ Issues of

S-100 MICROSYSTEMS

Name ____________________________ Payment enclosed

Address __________________________

City ____________________________ Zip __________

State ____________________________ Zip __________

CPU ____________________________ Disk System __________________________

Clip and mail to: S-100 MICROSYSTEMS

P.O. Box 1192 Mountainside, NJ 07092
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 monitor. 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.

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 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 70 msec. Software supplied with the system includes a File Manager, 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.

S-100 MICROSYSTEMS
The Fifth
Trenton Computer Festival

TCF-80

April 19 & 20, 1980

10 AM to 6 PM
SATURDAY, 19th
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 accommodate 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

<table>
<thead>
<tr>
<th>Memory Name</th>
<th>Bus &amp; Notes</th>
<th>Unit</th>
<th>Assm</th>
<th>CSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>8K Econoram II A</td>
<td>S-100</td>
<td>$169</td>
<td>$189</td>
<td>$239</td>
</tr>
<tr>
<td>16K Econoram X</td>
<td>S-100 (1)</td>
<td>$299</td>
<td>$349</td>
<td>$429</td>
</tr>
<tr>
<td>16K Econoram X-16</td>
<td>S-100 (2)</td>
<td>$329</td>
<td>$379</td>
<td>$479</td>
</tr>
<tr>
<td>16K Econoram XII-A-16</td>
<td>S-100 (2)</td>
<td>$349</td>
<td>$419</td>
<td>$519</td>
</tr>
<tr>
<td>16K Econoram VX-16</td>
<td>H8 (3)</td>
<td>$339</td>
<td>$399</td>
<td>n/a</td>
</tr>
<tr>
<td>16K Econoram IX-16</td>
<td>Dig Grp</td>
<td>$319</td>
<td>$379</td>
<td>n/a</td>
</tr>
<tr>
<td>24K Econoram VBA-24</td>
<td>S-100</td>
<td>$449</td>
<td>$499</td>
<td>$599</td>
</tr>
<tr>
<td>24K Econoram VBA-24</td>
<td>S-100 (2)</td>
<td>$479</td>
<td>$539</td>
<td>$649</td>
</tr>
<tr>
<td>32K Econoram X-32</td>
<td>S-100</td>
<td>$599</td>
<td>$699</td>
<td>$789</td>
</tr>
<tr>
<td>32K Econoram X-32</td>
<td>S-100 (2)</td>
<td>$649</td>
<td>$729</td>
<td>$849</td>
</tr>
<tr>
<td>32K Econoram XV-32</td>
<td>H8 (3)</td>
<td>$649</td>
<td>$749</td>
<td>n/a</td>
</tr>
<tr>
<td>32K Econoram IX-32</td>
<td>Dig Grp</td>
<td>$599</td>
<td>$679</td>
<td>n/a</td>
</tr>
<tr>
<td>32K Econoram X</td>
<td>SBC/BLC</td>
<td>n/a</td>
<td>n/a</td>
<td>$1050</td>
</tr>
</tbody>
</table>

THE GODBOUT COMPUTER BOX

<table>
<thead>
<tr>
<th>Shipment &amp; Storage</th>
<th>Unit</th>
<th>Assm</th>
<th>CSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-100 2708 EROM BOARD</td>
<td>$85 unkit</td>
<td>$85 assm</td>
<td>$100 CSC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shipment &amp; Storage</th>
<th>Unit</th>
<th>Assm</th>
<th>CSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-100 ACTIVE TERMINATOR BOARD</td>
<td>$34.50 kit</td>
<td>$34.50 assm</td>
<td>$45.00 CSC</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shipment &amp; Storage</th>
<th>Unit</th>
<th>Assm</th>
<th>CSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-100 MEMORY MANAGER BOARD</td>
<td>$199 unkit</td>
<td>$249 assm</td>
<td>$324 CSC</td>
</tr>
</tbody>
</table>

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

25 "Interfacer II" S-100 1/O BOARD
$259 unkit, $329 assm, $409 CSC
Dual RS-232 ports with full handshake. On-board crystal timebase, hardware UARTS, much more.

3P PLUS "INTERFACER II" I/O BOARD
$199 unkit, $239 assm, $299 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/M™ can give a microcomputer with CP/M more power than many minis. You can buy our totally standard Wirth PASCAL/M™ II diskette, with manual and Wirth's definitive book on PASCAL, for $150 with the purchase of any memory board. Specify Z-80 or 8080/8088 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.

---

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 ROMs.

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.

25 "Interfacer II" 1/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 "INTERFACER II" I/O BOARD
$199 unkit, $239 assm, $299 CSC
Includes 1 channel of serial 1/O (RS-232 with full handshake), along with 3 full duplex parallel ports plus a separate status port.

PASCAL/M™ + MEMORY SPECIAL

PASCAL/M™ can give a microcomputer with CP/M more power than many minis. You can buy our totally standard Wirth PASCAL/M™ II diskette, with manual and Wirth's definitive book on PASCAL, for $150 with the purchase of any memory board. Specify Z-80 or 8080/8088 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) 362-0096. COD OK with street address for UPS. Sale prices good through cover month of magazine; other prices are subject to change without notice.