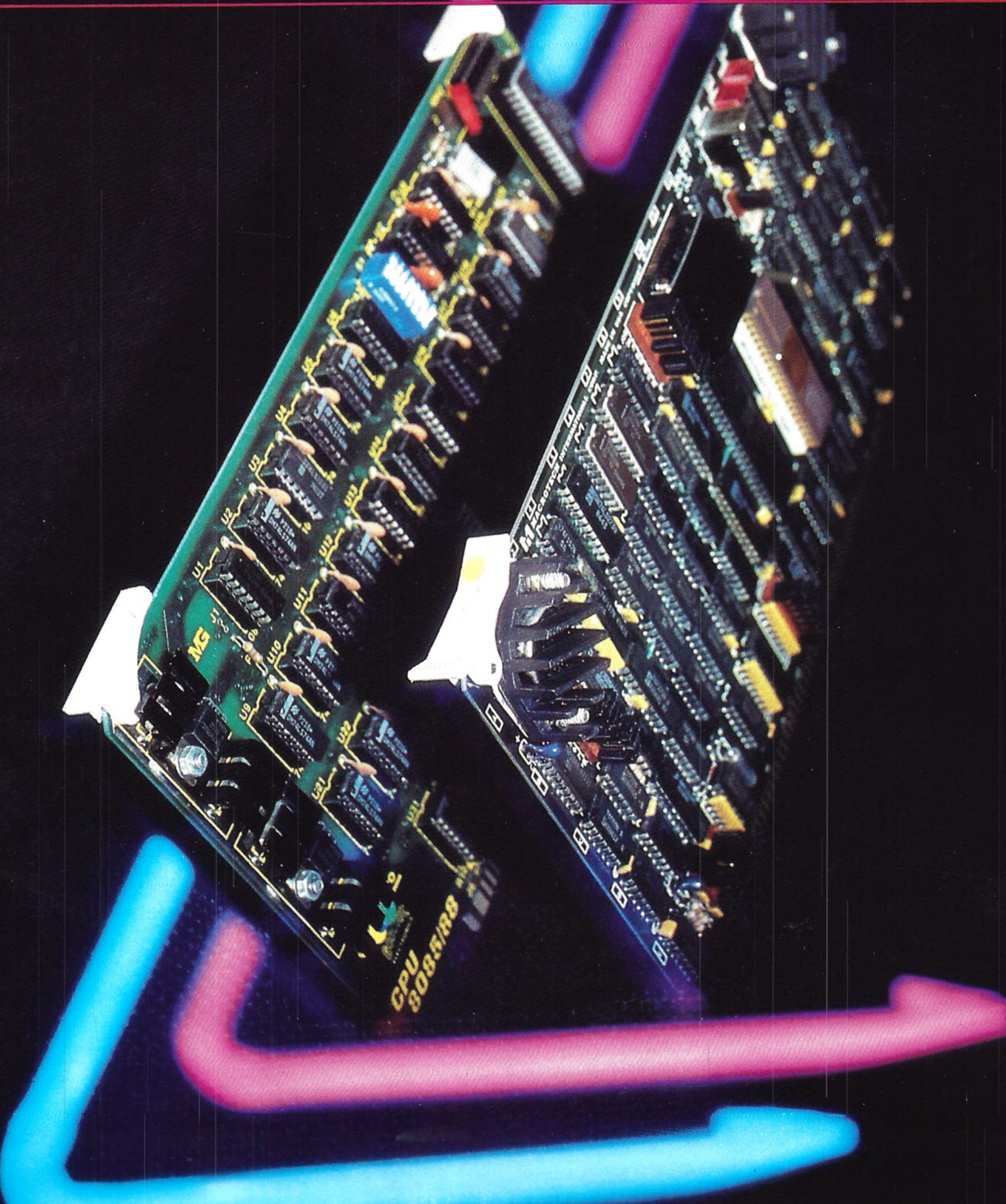
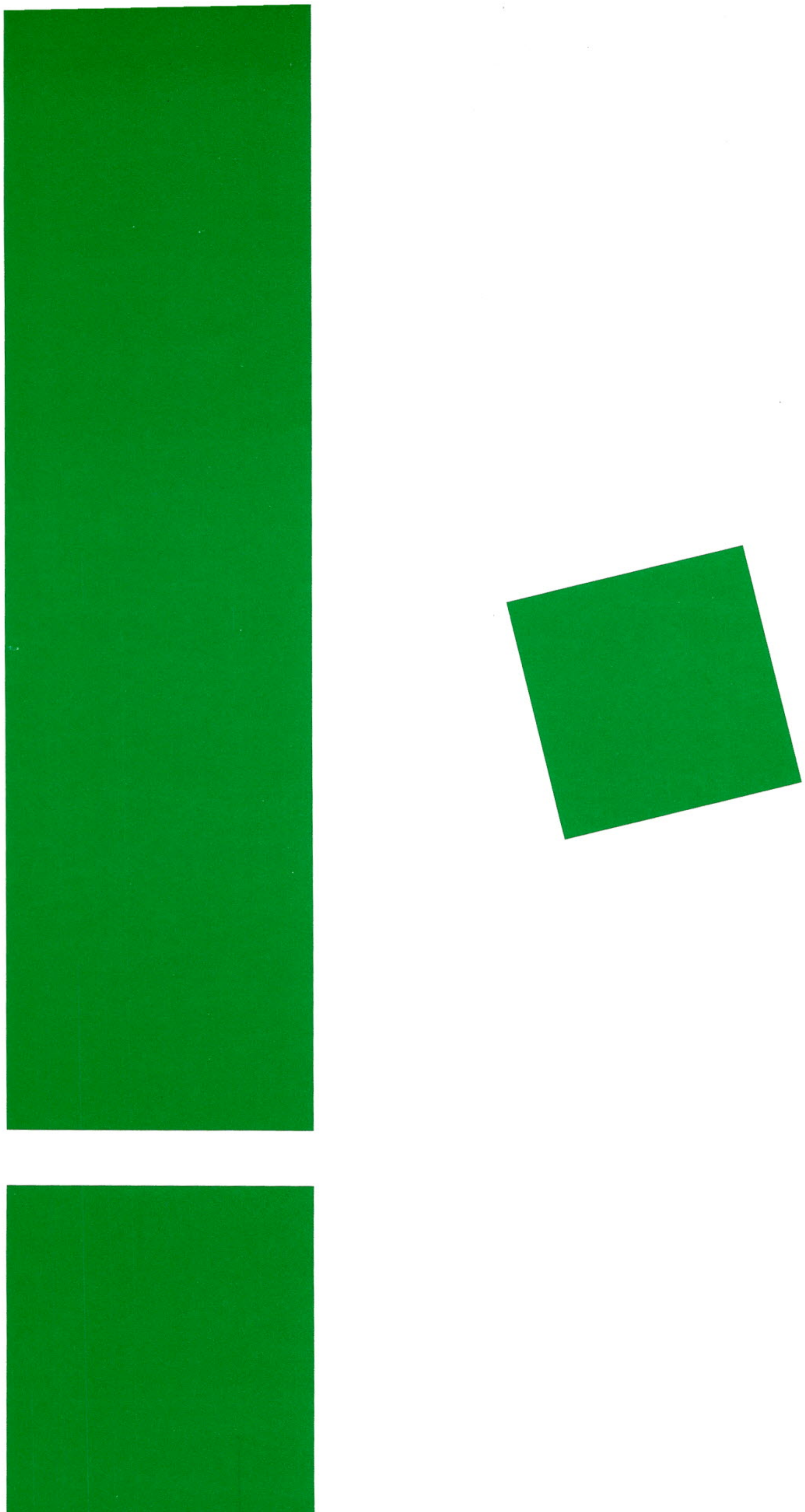
$4.90

VOL. 1, NO. 2 • FALL 1985

S-100 journal





S-100 journal

VOL. 1 NO. 2 FALL 1985

Editor
Jay Vilhena

Associate Editor
Robert Petersen

Business Manager
Linda Pereira

Advertising
Jay Vilhena

Technical Consultant
Steve Renshaw

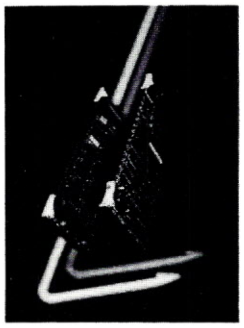
Art Director
Joslyn L. Johanna

S-100 Journal is published quarterly by Octoplus Corporation, 2426 Wade Avenue, Raleigh, NC 27607.
Copyright © 1985 by Octoplus Corporation. All rights reserved.
Subscriptions: U.S.A.: $14.00 for 1 year, $25.00 for 2 years. Canada, England, and Mexico: add $3.00 per year. Other countries: add $6.00 per year. Foreign payments must be in U.S. currency. Send orders and payments to S-100 Journal, P.O. Box 12881, Raleigh, NC 27605.
Advertising: Contact S-100 Journal Advertising, P.O. Box 12881, Raleigh, NC 27605.
POSTMASTER: Send address changes to S-100 Journal, P.O. Box 12881, Raleigh, NC 27605.
Application to Mail at Second-Class Postage Rates is pending at Raleigh, NC.

outline

OUR COVER

The Macrotech MI-286 and the CompuPro CPU 8085/88 offer both 8- and 16-bit processing capabilities. See pages 10 and 14 for more about these CPU boards.



S-100 INTERNATIONAL TRADE ASSOCIATION

While mythomaniacs with ulterior motives narrated the tales of S-100 demise, our systems have evolved to outstanding levels of performance that very few micros, and not all minis, can match. In addition, S-100 machines continue to offer the versatility and expandability that is their most cherished attribute.

Given all these advantages, it is often difficult to understand the factors that allot S-100 systems such a small share of the microcomputer market. It is easy (and certainly satisfying) to accuse or harass the dominant systems for their bullheaded takeover of the market, but to us belongs the ultimate blame. The S-100 industry has flatly failed to communicate to the vast majority of potential computer buyers what we are about and why an S-100 system is the better investment. We remain shy, and one way or another justify our fear of competing head-on with the alphas in the pack.

S-100 suppliers compete in the market at two levels. At one level, which I will call level A, is the competition among these suppliers themselves for a share of the *S-100* market. This is a historical type of competition whose importance has long vanished. At the other level, I'll call it level B, is a more general competition for a share of the *microcomputer* market. This is where the action is. It determines how S-100 companies will perform in the years ahead.

Many S-100 companies continue to direct their efforts to level A. They either lack the resources to enter the larger picture or fail to develop the proper means of doing so. Other companies have seen the rainbow pot and slosh awkwardly at level B, succeeding only because the pot is so immense that a few stray coins will make a pauper rich.

A third group has become aware of a third alternative, level X.

S-100 companies lack market identity. We watch puzzled as the producers of mediocre systems, often priced thousands of dollars above ours, gorge themselves in the core of the pot while our high-performance systems gather the crumbs. It is the game of the name. The microcomputer consumer, like scores of other consumers, buys by name, often understanding little of the substance that goes with the name. S-100 companies have understood this. They have stretched their resources to buy a name. Some became engulfed in long quests for the ultimate name. Others went under. It hasn't worked.

Level X, like level B, belongs in the mainstream of microcomputer competition. It seeks to take over a significant portion of the core of the pot. And it has the resources to do it. Level X has a market name. The name is simple, respected, and seasoned. Level X has the financial resources to promote the name in grand scale. The name is **S-100.**

This fall at COMDEX, an event of historical importance took place. Thirty-five representatives of S-100 and S-100-related companies met to discuss level X. Level X is the cooperative promotion of the S-100 bus. While individual companies may lack enough name recognition to significantly penetrate the market, this need not be so when resources are combined. At this level, advertising could be accomplished in a scale that would place the S-100 bus in the mainstream of the microcomputer and supermicro worlds. The key resides in promoting the S-100 bus itself as a symbol of quality, performance, modularity, wide support, infinite upgrading possibilities, and nonobsolescence. These are certainly the qualities that the user/consumer looks for. The S-100 bus already offers these qualities. Now we need to advertise them.

At the COMDEX meeting, the S-100 INTERNATIONAL TRADE ASSOCIATION (SITA) was created. A committee was formed to direct the early stages of activity, seek charter memberships, and organize a new meeting of all the charter members. This meeting, to take place early this spring, will decide organizational details and discuss specific S-100 promotional strategies.

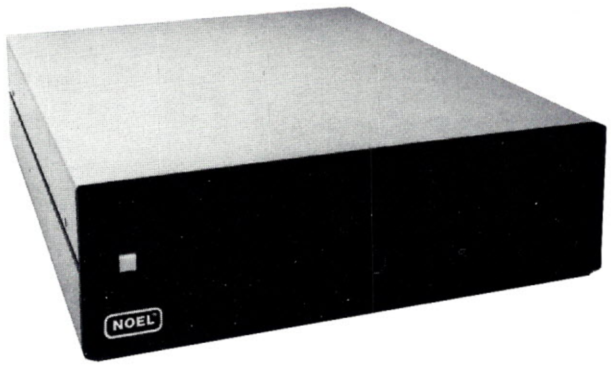
S-100 Journal salutes and endorses the creation of SITA. Over a dozen S-100 manufacturers have already joined, and we urge all other S-100 companies to join this association and take part in the spring meeting. For more information write to SITA, P.O. Box 28270, Raleigh, NC 27611.

If you are an S-100 user, write to the manufacturers that you buy from and urge them to join and support SITA. You'll be doing your part to guarantee that latest-technology S-100 products continue to be introduced.

And keep an open eye, for great days lie ahead . . .

Jay Vilhena

The NOEL™ An 8 User System



This powerful system based on the Motorola 68000 offers **eight RS-232C** ports, 512K bytes of Dynamic RAM, **20 megabytes of hard disk** storage with fast and reliable voice coil seek, 1.2 megabytes of floppy disk storage, programmable realtime clock, time of day clock with battery back up, and **encryption for software protection.** In spite of its compactness this system offers **expandability** in several respects. Within the same chassis RAM memory may be expanded to 2 megabytes and disk storage to 86 megabytes. If more disk storage is desired the SCSI bus is available to go to other disk and tape storage subsystems.

Software is also NOEL's forte. It can run the well known Alpha Micro AMOSL operating system, the fast and efficient Mirage multiuser multitasking operating system from England, the realtime O/S-9 operating system, CP/M-68K and the FORTH environment. The compilers available include APL, several versions of extended Basic, Lattice C, Whitesmith's C, Fortran 77, and Pascal. Applications range the gamut from the very general like accounting packages, spread sheets and word processors to the very specific such as a marine distributor software package, laboratory pathology system, and veterinary office management system.

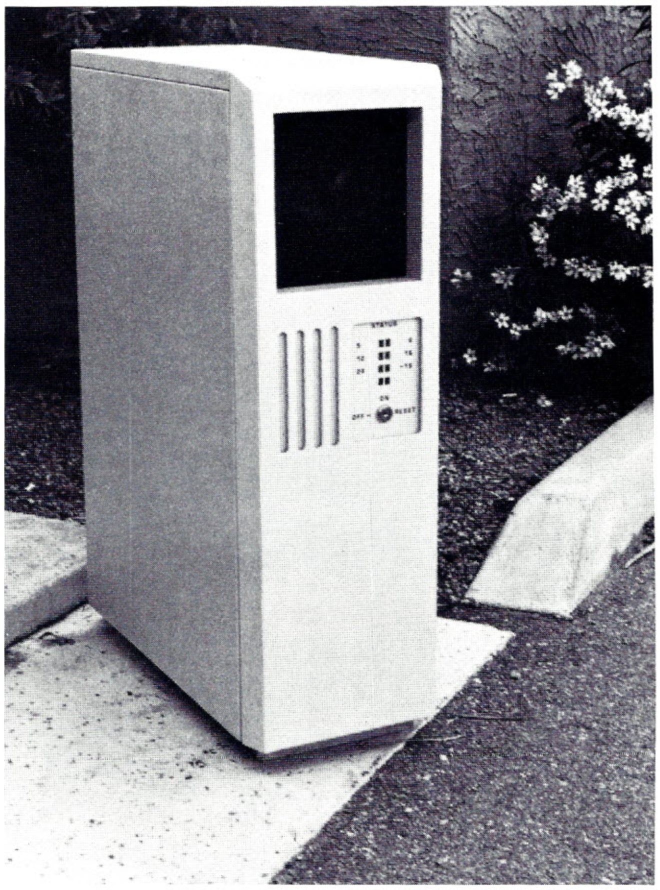
It Grows and Grows And Grows

In-chassis upgrade options include:	**List Prices**
NOEL	$4,999
• 68010 processor	$99
• 68020 processor (with The Powerhouse Board ™) • 68881 floating point unit (with The Powerhouse Board ™)	$1,199
• 86 Megabyte of hard disk storage (28ms avg seek)	$1,499
• second 1.2 Megabyte floppy disk drive	$249
• 20 Megabyte streamer tape unit	$999
• 60 Megabyte streamer tape unit	$2,499
• 2 Megabytes of RAM memory	$1,199
Out-of-chassis upgrade options include:	
• 16 additional card slots	
• 16 Megabytes of RAM memory	
• 170 Megabyte of hard disk storage (18ms avg seek) (any number of units)	

Call or write for our free catalog:

Inner Access Corporation

P.O. Box 888 • Belmont, CA 94002 • (415) 591-8295 • Telex 494-3275 INNACC



Para Dynamics Introduces Vertical Enclosure

An elegant 50-amp tower enclosure, Model 5820-S, is now available for S-100 systems from Para Dynamics. The 5820-S features a 20-slot, 696-standard motherboard, adjustable termination, front panel LEDs to indicate bus and drive voltages, and 3-position key switch.

This enclosure is specially suited for larger multi-user systems, providing plenty of power for up to 20 S-100 boards and a wide assortment of 8″ and 5¼″ disk drives. An advanced cooling design forces air between the boards. The 5820-S lists for $1495.

For additional information contact Para Dynamics, 7895 E. Acoma Dr., Scottsdale, AZ 85260. Telephone (602) 991-1600.

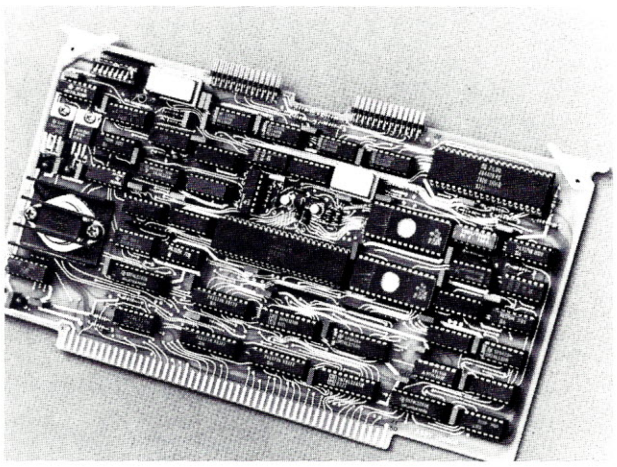
Z8001 Board Available from Way Engineering

The Super Z-10 is an IEEE 696 S-100 board that utilizes the Z8001 16-bit processor. The Z-10 operates at 10 MHz and addresses 16 megabytes of RAM. Additional on-board functions include two serial ports, 32K of phantom EPROM, and up to 3 selectable wait states.

The Super Z-10 operates with standard S-100 static RAM or, for greater clock speeds, with a special piggy-back 512K dynamic RAM board. Combinations of the two types of RAM are also supported.

The monitor EPROM boots 79 FIG FORTH upon power-on or reset. Way Engineering also offers CP/M-8000 with BIOS for the Super Z-10. The BIOS is written in C and assembler and can be modified by the user.

The Super Z-10 costs $995 in single quantity. The CP/M-8000 is $350 and includes the C language. Contact Way Engineering Inc., 2011 Tulip Tree Lane, La Canada - Flintridge, CA 91011. Telephone (213) 245-1480.

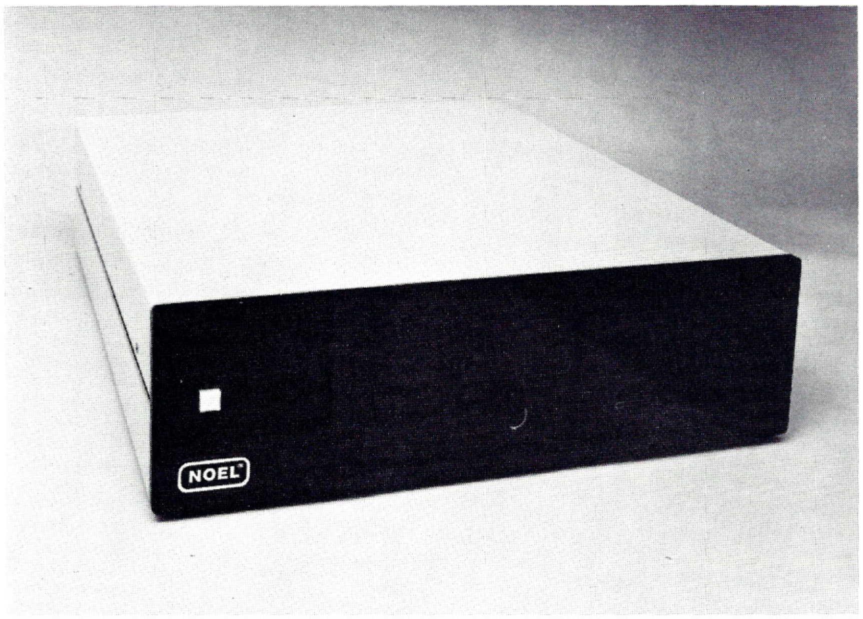


NEW AFFORDABLE 8-USER SYSTEM FROM INNER ACCESS

Inner Access Corporation has announced the NOEL, a multiuser system that runs several operating systems, including AMOS, Mirage, O/S-9, CP/M-68K, and FORTH. A wide range of compilers and applications packages are available for this system.

The NOEL is 68000 based and offers eight serial ports, 512K bytes of Dynamic RAM, 20-megabyte hard disk, 1.2 megabytes of floppy storage, time-of-day clock, and a large array of upgrade options.

This compact system runs at 8 MHz and meets IEEE-696 specifications. It sells for $4999. Contact Inner Access Corporation, PO Box 888, Belmont, CA 94002. Telephone (415) 591-8295.



L/F TECHNOLOGIES OFFERS CONCURRENT DOS

L/F Technologies (formerly IMS International) now offers its 1600 series of S-100 systems with Concurrent DOS, the popular multiuser, multitasking operating system from Digital Research. The 1600 systems with Concurrent DOS, featuring RAM-disk, 1 megabyte of RAM, and 24-meg Winchester, start at $6495.

L/F Technologies has also introduced a back-up power source, the L/F-Power, specially configured for the company's hardware products. L/F-Power supplies up to 1 hour of DC voltage directly to the system, eliminating the process of DC to AC reconversion used in external back-up systems.

A related product, the L/F-Power-II, offers a similar integral back-up power supply for L/F Technologies' terminals.

For more information, contact L/F Technologies, 2800 Lockheed Way, Carson City, NV 89701.

The New Products section is compiled from press releases and other information supplied by manufacturers. Only press releases on S-100 products and peripherals are considered for publication. High-quality photographs are welcome. Press releases should be dated and should include prices. Send to New Products, S-100 Journal, P.O. Box 12881, Raleigh, NC 27605.

editor interface

For those coming in now, Editor Interface is the column where I answer questions and publish letters concerning (mainly) the contents and orientation of the magazine. We like to receive your comments and suggestions for improving S-100 Journal. Send letters to Editor Interface, S-100 Journal, P.O. Box 12881, Raleigh, NC 27605. Please print or type your letters.

Our magazine is growing at a fast pace. With your support, we will be around for a long long time, and, best of all, so will S-100 systems. We will continue to improve S-100 Journal to make it your second best investment. Second only to your S-100 machine, of course.

But let's get on with the mail:

First Issue Comments

Hope next issues look as good as the first!

R. V. Peringer
Orange, California

Glad to know that you liked it. We will keep S-100 Journal looking good. Besides esthetics, we are also concerned about increasing the amount and quality of the information provided. • *Jay*

Finally! A place to stand against the blue fungus. For sheer spite, you should send a copy of that first editorial to IBM "for informational purposes only" of course.

S. Shumaker
Vacaville, California

We've heard that a few people do not share your enthusiasm about our magazine. But, we know who they are. And, with a little imagination, we can also figure out where they live. • *Jay*

In Search of S-100 Info

I received the first issue of the S-100 Journal with interest and pleasure. You invited reader comments; here are some.

I would like to see some articles or be referred to some literature on computer design with the IEEE 696 buss. Sol Libes and Mark Garetz' book "Interfacing to S-100/IEEE 696 Microcomputers" is excellent, but when one searches for the books they suggest as foundation, one finds they are out of print. Search as I may, I have found very little published on S-100 computer design. I have never seen "The S-100 Bus Handbook" in a B. Dalton bookstore. I will specifically request it from them, and I will write to Micro/Systems, using the address in the article on the 68000.

I have always been a little puzzled why the S-100 buss is not used more widely, at least by old circuit design engineers like myself who have lately gotten into micros. It seems to offer one an opportunity to concentrate on the computer circuits that are of interest without having to build up all the supporting equipment. I have designed and built all the power supplies I ever want to! Perhaps the lack of S-100 technical literature is part of the reason for the "... lost ground in the press," as you say. Good documentation still is a problem in a major part of the microcomputer business.

Because I am more familiar with the 8085 series and newer Intel chips, I hope you will have some future articles feature them.

I look forward to your future issues!

Eliot C. Payson
Littleton, Colorado

I expect that S-100 Journal will be able to provide a lot of needed information about the S-100 bus and S-100 products. In the future, we may even consider publishing some S-100 books, if we locate interested authors.

I checked with the local B. Dalton, and they still stock the "S-100 Bus Handbook." Sol Libes is trying to reprint "Interfacing to S-100/IEEE 696 Microcomputers." If he succeeds, we'll make it available through S-100 Journal.

No doubt that the Intel line will be addressed often in our pages. • *Jay*

MS-DOS, UNIX, BIOS and A/D

I was very glad to get my copy of the "Journal" — because most, if not all, microcomputer magazines seem to be very much oriented toward the IBM-PC. Well, I own an S-100 system, and I'd like to keep it as long as possible. I'm happy with its speed — mass data — RAM storage, etc.

I usually don't make a habit of expressing my personal views to others, unless otherwise asked to do so. However, statements were made in the first issue of the Journal which have prompted me to address the following comments/questions to you. I hope you accept the following remarks as being constructive in nature; they are not meant to be a criticism.

1. MS-DOS: "General purpose MS-DOS/PC-DOS software will not be accepted." Comment: I have several Microsoft DOS compatible programs that I cannot be without. For example, my Fortran Compiler, and my 1/2-inch, 9-track tape utilities. All of these programs — as well as other utilities — are all generic MS-DOS programs. Does the statement on page 2 mean that those programs would not be advertised or reviewed in the Journal??

2. I have rack-mounted my system — in a "T-bar" 19-inch cabinet — and at present I'm trying to assemble an A/D subsystem [. . .] for the purpose of collecting and digitally analyzing

geologic-geophysical data. If these topics are of interest to you, please let me know.

3. I am not an electrical/digital engineer, but I would like to have access to information on how to write a BIOS for my 16-bit 8086 system (Seattle) — needless to say the BIOS is not available in source. But, an article in the S-100 Journal which would describe how one could write a "typical" BIOS routine would be very helpful. [. . .] I've got several books on assembler language, but they do not address the overall mechanics of this fundamental piece of software. Even something as simple as a block diagram, with some minimal text, would be a big help!

4. One important change seems to be taking place in the computer industry in general — UNIX. UNIX is being ported to almost all micro - mini mainframe systems [. . .], including some S-100 systems! I honestly hope you'll be able to provide the reader with info on who in the S-100 industry will implement UNIX on our bus. For example, I know that there is a very real possibility of getting public domain source code in C, for UNIX — look alikes — work alikes! Here's an area where the Journal could be very helpful: information on how to develop a kernel for a "typical" S-100 system?? For the 8086-80286-6800-3200 CPU's!??

Tony Price
Burbank, California

Thanks for writing. I will tackle your points one at a time:

MS-DOS: You are correct! We will neither accept ads on nor review MS-DOS software. It's not that we have any feud with MS-DOS, but, because of the illogical popularity of the IBM PC, MS-DOS occupies an extraordinary percentage (sometimes close to 100%) of the space of all major computer publications. With all that info available, there is really no need for us to use our space on MS-DOS. But, I do think that S-100 systems should be able to run MS-DOS software (or anything else for that matter), and S-100 Journal will publish information that deals with implementing or running MS-DOS software in our micros.

Analog to Digital: It appears that a lot of people are using, or planning to use, their S-100 systems to collect analog data. And, yes, we are interested! If you succeed in your application, we would be eager to hear about it, and we would even consider publishing an article describing the process in detail.

BIOS: I am certain that S-100 Journal will address BIOS topics many times. In this issue, you will find an article by Howard Spindel that gives detailed information about a (specialized) CP/M BIOS.

UNIX: There are in fact a number of UNIXes (or close variants) running on S-100 systems. For some names, see the multiuser column that Gary Feierbach inaugurates in this issue. • *Jay*

BE-82 VIDEO BOARD FOR THE S-100 BUS

FEATURES:
- Bit-mapped characters and graphics, 640 by 330 pixel display
- Full Greek set in EPROM, 7 by 9 character, 8 by 11 window
- Characters placed on any raster line, 30 char-line display
- Load a user program in BE-82 RAM for special commands
- Create graphics on one screen while viewing another
- 660 raster lines available for a single drawing
- Lines drawn at 8800 pixels/s, pixels set at 800/s
- Effective baud rate to 10,500 in terminal mode

S-100 BUS HACKERS:
- Full documentation, user can customize to special needs
- Purchase items singly, as a group, or choose additions later
- Bare board plus hardware/software documentation ...... $135 ☐
- Programmed EPROMS (three 2716s) .... 30 ☐
- Chip set plus hardware components .... 125 ☐
- Assembly and testing .... 100 ☐

HARDWARE:
- Amdek 310A Video Monitor and Key Tronic keyboard recommended
- S-100 bus system or ZENITH Z-100 Low Profile Computer
- BE-82 VIDEO BOARD acts as a slave device at all times on bus
- Board operation is independent of host processor clock speed

SCIENTISTS AND ENGINEERS:
- DB-WRITE Technical Word Processor .... $100 ☐

Write your papers using a display that shows exactly what is printed. Operates under CP/M 80 by Digital Research and requires a 64k system. The BE-82 VIDEO BOARD is required to use DB-WRITE. Full documentation and all files provided. Use CP/M to customize keyboard layout by changing jump table, modify printer driver, etc. Printer driver included for the NEC SPINWRITER and the Tech Math/Times Roman thimble. Example of display screen and printout:

$$f(\lambda) = \int_{-\infty}^{\infty} e^{-\beta(\lambda^2+\xi^2)}\left[\frac{\partial^2\rho(\xi)}{\partial\xi^2} + H_0(\xi)\sin^2(\lambda-\xi)\right]d\xi \qquad \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix}$$

Position char's $0\;{}^{2}\;{}_{4}\;{}^{6}\;{}_{8}\;{}^{10}\;{}_{8}\;{}^{6}\;{}_{4}\;{}^{2}\;0$ from line

$\alpha - \beta - \gamma - \delta - \varepsilon - \eta - \lambda - \mu - \xi - \pi - \rho - \sigma - \tau - \omega$

Name ______________________________

Address ____________________________ ☐

City _______________________________ Additional Information

State ____________________ ZIP________

D BAGANOFF SYSTEMS, 3141 DAVID COURT, PALO ALTO, CA 94303

INTRODUCTION TO THE S-100 BUS A LITTLE HISTORY

Don Pannell

696 Bus is our new regular column that concentrates on the hardware aspects of the IEEE-696 bus (i.e., the S-100 bus) and answers questions that readers might have about the IEEE standard.

S-100 Journal is pleased to have Don Pannell as our 696-bus columnist. Don is an S-100 hardware enthusiast and is coauthor of the IEEE-696 standard. He bought his first S-100 system (an Altair 8800b kit) in January 1978 and has since designed and built most of his present components. These include a terminal, EPROM programmer, serial/parallel I/O cards, two designs of DMA floppy-disk controller, and a 68000 coprocessor card. (Don markets the 68000 coprocessor through his Peak Electronics company.)

If you have questions about the S-100 bus and IEEE standard, write to Don Pannell, P.O. Box 700112, San Jose, CA 95170-0112. Your questions can range from architectural concerns to how to interface a specific device or function with the bus. Don will incorporate answers to the most common questions in future issues.

In this first column, he gives us a little history background and an overview of the differences between the original S-100 bus and the IEEE-696 bus.

The S-100 bus first appeared as a computer kit by MITS Corporation, in the January 1975 issue of "Popular Electronics" magazine. The kit was based on the Intel 8080 and was called the Altair 8800. It contained a front panel with all the proper lights and switches, an 8-amp unregulated power supply, and a CPU card. The main feature of the Altair was its expandability. The CPU card plugged into a 100-pin edge connector. This sent the CPU card's signals down a bus and out to as many as 15 other cards plugged into the same chassis.

The kit was just what the hobbyist community wanted. MITS, although already known for its calculator kits published in "Popular Electronics," was not ready for the response it received for the Altair kit. Part of the big response was due to the low price of the kit. At the time, Intel was asking $350 just for the 8080 chip. MITS advertised the complete kit for $395. Within one week MITS had received 200 orders. Unfortunately, the kit article had been published before the Altair was ready. It took MITS another five months before their first product was ready for shipping.

The quick design cycle was part of the reason why some aspects of the original bus were not very well thought out. Timing specifications were nonexistent, and future upgrades had not even been considered. Despite this great rush, MITS did several things right. They defined an expandable bus that was easy to interface with, supported DMA (direct memory access), and allowed any card to be plugged into any slot. This accomplishment was partially due to Intel's design of the 8080 CPU. MITS simply brought ALL the 8080's signals to the bus. Anyone that has ever wondered where the unusual S-100 pin assignment came from has only to look at the original Altair 8080 CPU card. An unknown printed circuit board designer defined the pin assignment to make his or her job easier. All the traces on both sides of the original MITS card come straight down from the buffer chips to the bus connector. It's a beautiful card, with very few vias (holes in the board).

THE BUS DESIGN A MIXED BLESSING

The fact that the S-100 bus was a direct extension of the 8080 CPU proved to be a mixed blessing. Initially, it allowed many other vendors to quickly jump on the S-100 bandwagon by designing cards that basically followed the 8080's timing specifications. Without this multivendor backing, the S-100 bus would not have gone very far. In its first two years, the bus became so popular that in 1977 over 60,000

(continued on page 20)

EARTH LAUNCHES NEW STARS

EARTH COMPUTERS launches two of the "Hottest" new stars in the S-100 Universe. Both the **TURBOMASTER 8**™ and **TURBOSLAVE I**™ are Star performers, featuring high speed Z-80H CPUs.

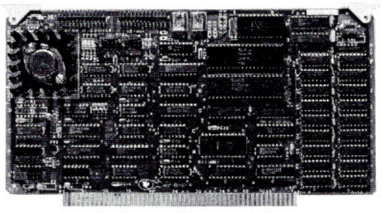
EARTH's new stars are fully compatible with the Multi-user TurboDOS operating system, and will operate in most S-100 systems, including pre-IEEE 696 systems.

TURBOMASTER 8

This outstanding new 8-Bit Single Board Computer offers features that are out of this world:

- On-board ST-506 Winchester Controller
- TurboDOS, CP/M, MP/M compatible
- 5-¼″ and 8″ Floppy Controller
- Up to 256KB of memory
- 8 MHz, Z-80H CPU
- 2 Serial ports
- 1 Parallel port

TURBOSLAVE I

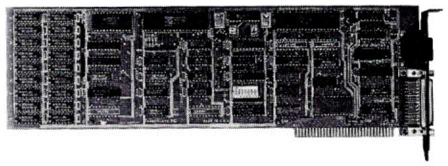


The perfect companion to the TURBOMASTER 8 or other 8/16-Bit Master processor. This high speed slave utilizes an 8 MHz Z-80H CPU and offers extensive on-board diagnostics... an industry exclusive.

- No paddle boards
- S-100, IEEE 696 compatible
- 128KB of RAM
- 2 RS-232 ports, 50–38.4K Baud
- FIFO communications
- Data transfers to 1 MB

EARTH COMPUTERS also manufacturers a growing line of PC compatible stellar performers such as:

- **TURBOSLAVE PC**™—an 8 MHz



Z-80 single board slave processor that runs CP/M applications on a PC and is compatible with the TurboDOS multi-user operating system.

- **TURBOACCEL 286**™—a high performance 80286 accelerator that boosts PC performance up to five times.
- **EARTHNET PC**™ and **EARTHNET S-100,** the low cost, ARCNET-compatible way to tie PC and S-100 systems together.

To put these stars to work for you, call or write EARTH COMPUTERS. **BE SURE TO ASK ABOUT HOW YOU CAN WIN A FREE Z-80 CO-PROCESSOR BOARD.**

EARTH COMPUTERS

"Building Blocks For The Super Micro"

P.O. Box 8067, Fountain Valley, CA 92728 • TELEX: 910 997 6120 EARTH FV • PHONE: (714)964-5784

CP/M-86 ON OLDER S-100 COMPUTERS USING A DUAL-PROCESSOR BOARD

Howard Spindel

The explosive growth rate of the computer industry can make owning a computer very frustrating. Six months after you lay out the big money for the latest system, another company comes along with something better (sometimes for less money — that really hurts!). One of the ways to protect yourself from obsolescense is to buy a computer which can be upgraded in a modular fashion. Five years ago, I decided to purchase a computer which could be upgraded. I chose an S-100 system running the popular 8-bit CP/M operating system with the intent of supporting a more advanced 16-bit operating system in the future. The key to upgrading my computer to run newer 16-bit software was the CompuPro CPU 8085/88, a dual-processor board from Viasyn Corporation. (Also see the box on page 13 for an update on using the Macrotech MI-286 dual-processor board.)

The CompuPro Dual CPU (i.e., the CPU 8085/88) is an S-100 processor board which contains both an 8-bit 8085 microprocessor and a 16-bit 8088 microprocessor. This board gives S-100 computers the flexibility of simultaneously running older 8-bit software and newer 16-bit software. Some articles have appeared describing how older S-100 computers can be upgraded to use the CompuPro CPU 8085/88 as a processor board [1] [2] [3]. These articles have done a fine job of presenting the **hardware** pitfalls that can occur when replacing an older S-100 processor board with the Dual CPU.

*This article will show you a relatively easy way to upgrade the **software** of your S-100 system so that you can run CP/M-86. You will be able to switch between CP/M-80 and CP/M-86 with a simple command.*

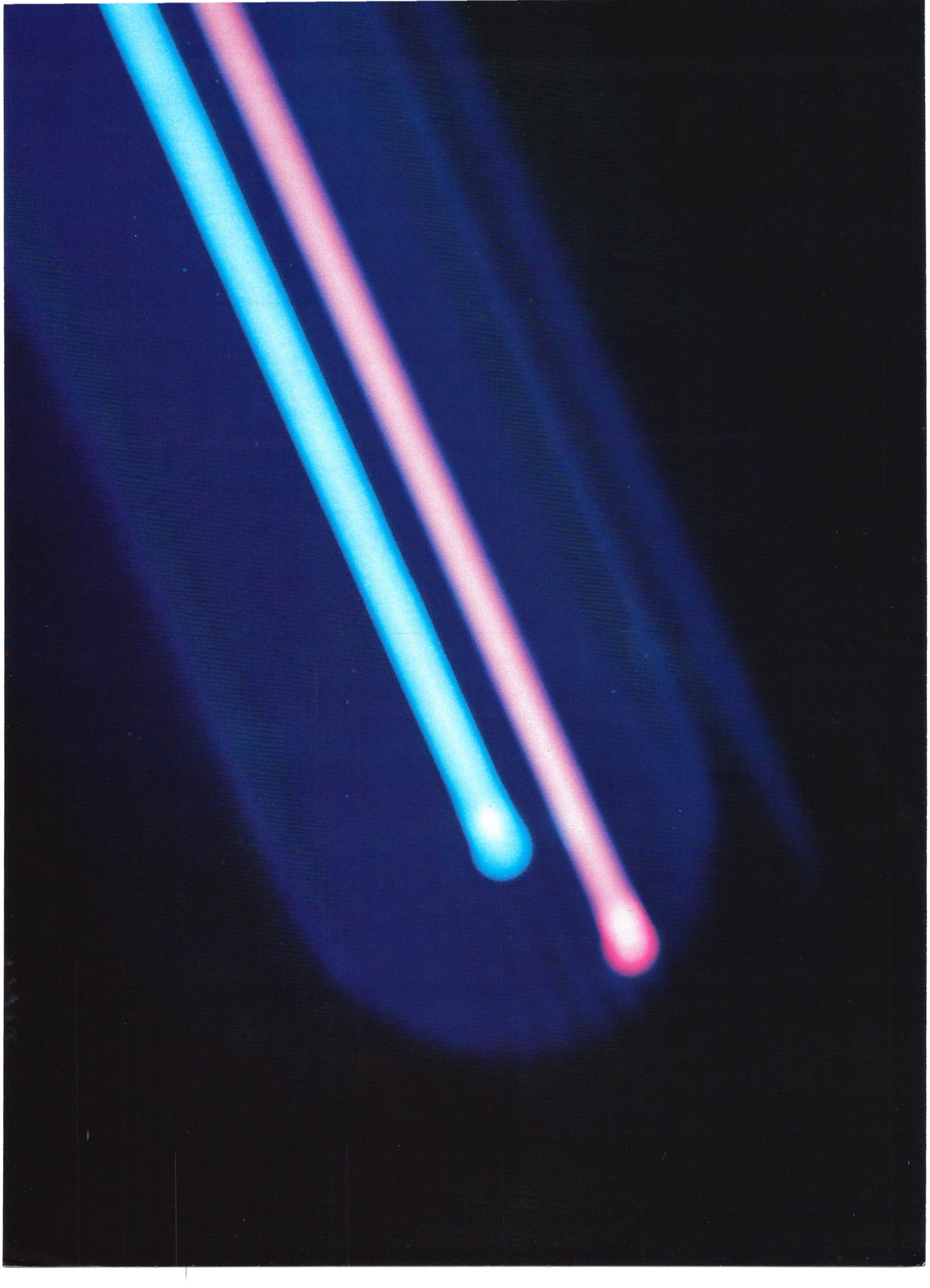
Let's assume that you have successfully upgraded the hardware of your system, and that you are now running CP/M on the 8085 microprocessor of your Dual CPU. The battle is only half won. Now you have a computer that includes a powerful 16-bit 8088 microprocessor, but this CPU never gets used because you don't have any software to run on it. It would be great to run CP/M-86 (a 16-bit version of CP/M) on the 8088 because CP/M and CP/M-86 have compatible disk formats. This would give you an easy way to share files between 8-bit and 16-bit programs.

This article will show you a relatively easy way to upgrade the **software** of your S-100 system so that you can run CP/M-86. You will be able to switch between CP/M and CP/M-86 with a simple command. If you are not familiar with the basic structure of CP/M, including the terms CCP (Console Command Processor), BDOS (Basic Disk Operating System), and especially the BIOS (Basic Input Output System) you may first want to refer to the glossary included with this article.

PAINFUL UPGRADE METHODS

There are two obvious and difficult methods for bringing up CP/M-86:

1. Rewrite all of your BIOS code in 8086 assembler, and build a CP/M-86 system from scratch. If you're like



me, you have already spent many long hours customizing your BIOS to get it just right. The thought of going through that all over again is distressing. (All that time with the computer will take some explaining to your spouse or significant other.) Even if you were successful with this approach, every time you made a BIOS change from then on, you would have to do it in two places (CP/M and CP/M-86).

2. Purchase a customized version of CP/M-86 from Viasyn. This would mean buying additional hardware. In addition to the Dual CPU, Viasyn's implementation of CP/M-86 requires the CompuPro Disk 1 (or Disk 1A) floppy disk controller and a CompuPro support board like the Interfacer or System Support 1. This can get expensive, easily doubling the investment in the Dual CPU.

These were the alternatives that I faced and rejected. I will show you another way to bring up CP/M-86 using the standard Digital Research distribution version of CP/M-86 and your current hardware. Your hardware can be almost anything as long as you've upgraded the processor board to the Dual CPU, and you have CP/M running on the 8085. For the curious, the author's hardware configuration is listed in Table 1.

CP/M-86 THE EASY WAY

The basic approach to an easy upgrade to CP/M-86 is simple in philosophy. Why not use the 8085 and its existing BIOS for CP/M as an I/O (input/output) processor for the CP/M-86 running on the 8088? After all, a BIOS for CP/M and a BIOS for CP/M-86 provide almost exactly the same functions. There are a few minor entry points in the CP/M-86 BIOS which don't exist in CP/M, but they are simple routines, easy to implement on the 8088. All the difficult device driver code, most notably the disk drivers and disk sector deblocking code, is already available in 8085 code. What's more, anytime a change needs to be made to the BIOS for CP/M, that change is also instantly made for CP/M-86 since both versions of CP/M are running exactly the same code when interfacing to BIOS devices.

The Dual CPU makes this approach possible by making it easy to switch back and forth between the 8085 and the 8088 microprocessors. All that is necessary is an input instruction to an I/O port (by default this port is OFD hex). The microprocessors can be configured so that each time one is switched into activity, it picks up exactly where it left off. The microprocessors do not have any way of communicating with each other except to share an area of memory. This led me to the notion of a BIOS task block. A BIOS task block is a special area of memory into which the 8088 can write a request for the 8085 to perform a BIOS function. When the 8085 is finished doing the BIOS function, the results are written back into the BIOS task block and read by the 8088. Note that the Dual CPU does not allow both microprocessors to run simultaneously, so there are no problems associated with synchronizing two independent microprocessors.

Digital Research did something very smart (in my opinion at least) which considerably simplifies the creation of BIOS task blocks. It provided CP/M with a consistent 8085 register allocation method for BIOS calls. Characters to be output are always placed in register C. Input characters always come back in register A. 16-Bit input quantities are always passed in BC and DE, and 16-bit output quantities appear in HL. Not only is this style of register allocation also provided in CP/M-86, but a constant one-to-one mapping of the 8088 registers to the 8085 registers is maintained. This mapping is detailed

Author's hardware configuration at the time that this project was first completed:

Installed in CompuPro Enclosure II:
- CompuPro Dual CPU
- CompuPro RAM 17 (64K static RAM)
- Morrow DJ2D floppy controller
- Morrow Mult IO general purpose board
- Fulcrum VIO-X2 video board
- D.C. Hayes MicroModem
- ADS Promblaster

Installed in separate cabinets:
- 4 Shugart 801 floppy drives

Author's current hardware configuration:

Installed in CompuPro Enclosure II:
- CompuPro Dual CPU
- CompuPro RAM 17 (64K static RAM)
- Digital Research 64K static RAM
- CompuPro Disk 1 floppy controller
- Morrow Mult IO general purpose board
- CompuPro System Support 1
- Fulcrum VIO-X2 video board
- Digital Research LS-100 256K RAM disk boards (2)
- D.C. Hayes MicroModem
- ADS Promblaster

Installed in separate cabinets:
- 4 Shugart 801 floppy drives

Table 1. *The Author's Hardware Configuration. Almost any hardware configuration can be used to run CP/M-86 with the programs described in this article. The programs can be updated with very little effort as the underlying hardware changes.*

in Table 2 on page 16.

To implement the BIOS task blocks, the 8088 writes to memory the values that all 8085 registers are to assume before executing a BIOS function. The mapping of registers remains consistent regardless of the BIOS function being executed. Hence, the BIOS task block can look identical for all BIOS functions; the values of 8085 registers are passed without regards to what parameters the registers might contain.

The following summarizes how the 8085 and 8088 communicate whenever the 8085 is to provide the 8088 with any BIOS function:

1. The 8088 BIOS receives a request from CP/M-86.
2. The 8088 creates a BIOS task

(continued on page 16)

The Macrotech MI-286 Dual Processor Board

Although this article was originally written with the CompuPro 8085/88 Dual Processor in mind, through the good will of S-100 Journal and Macrotech I recently had the opportunity to test my program for a few hours with the Macrotech MI-286 Dual Processor. (The Macrotech MI-286 uses an Intel 80286 and a Zilog Z80 combination and is advertised as a direct plug replacement for the CompuPro Dual Processor.)

Since the MI-286 manual said that it came preconfigured as a CompuPro replacement, I pulled out my CompuPro board, put the Macrotech board in, and hit the power button. I was gratified to see the system boot CP/M 2.2 as usual. Next I tried running the BOOT86 program to fire up CP/M-86. My system crashed. I hit the reset button and tried it again, and this time CP/M-86 booted up and ran! A few more tries showed me that the system was a little flaky. I had been afraid of this because my memory boards are older 8-bit-only boards and I thought they might not be fast enough for the 80286. However, Macrotech had the foresight to also lend me the V-RAM, a 512K-byte static RAM board. I flipped through the V-RAM manual, configured the V-RAM board for 512K of system memory at address zero, yanked out my RAM boards, and put in the V-RAM. Power on, run BOOT86, and I've got a reliably running system!

I spent a little bit of time playing around with CP/M-86 and marvelling at the extra speed of the Macrotech board. The MI-286 has LEDs which show which processor is active. It was really fun to try different things and watch the relative brightness of the Z80 and 80286 LEDs shift back and forth.

I then ran BOOT80 to switch back to CP/M and decided to run BOOT86 again to make sure that the restart vector worked. Oops, crashed again. A couple of retries convinced me that this was a software failure. A little more thinking and the answer came to me — I'd been bitten by the fetch-ahead queue again. The 80286 has a much larger fetch-ahead queue than an 8088, and the 80286 was reading the restart vector before the Z80 could write it. The fix is simple and is left "as an exercise for the reader." For the time being, I decided to just live with having to reset the computer every time before booting CP/M-86.

About this time, I decided to read the MI-286 manual more thoroughly, and discovered that I could enable memory wait states. This sounded like just the thing I needed to use my older memory boards. A quick jumper change and a couple of board swaps later I had a reliable system using my older memory boards.

I also wanted to get some idea of how much faster the MI-286 was than the 8085/88. For a simple test, I put both CBIOS86.A86 and ASM86.CMD on my Digital Research RAM disk and assembled the BIOS. I got the following results (hand timed with a stopwatch):

MI-286 with Macrotech 16-bit memory	15 seconds
MI-286 with my 8-bit memory (1 wait state)	28 seconds
MI-286 with my 8-bit memory (0 wait states)	21 seconds
CompuPro CPU 8085/88	27 seconds

The above table shows that the system RAM has an enormous effect upon the system performance. To use the MI-286 effectively, you should plan on using it with 16-bit memory boards.

I also became curious about how much performance degradation was incurred by using the Z80 to drive the BIOS routines as opposed to having true CP/M-86 BIOS routines. Recently, I broke down and bought and installed CP/M-816 from Viasyn. I tried booting CP/M-816 using the MI-286 with the Macrotech memory board and it worked perfectly. I then tried the above timing test and found that the assembly took 14.5 seconds. This is only slightly faster than the CBIOS86 approach and may in fact be within the error limits of hand timing.

Dear Macrotech,
Your boards were wonderful to use and very fast. I truly hated to ship them back to S-100 Journal. Since they are technically used equipment now, would you consider selling them to me inexpensively? Pretty please?

Howard Spindel

MACROTECH MI-286 AND COMPUPRO CPU 8085/88

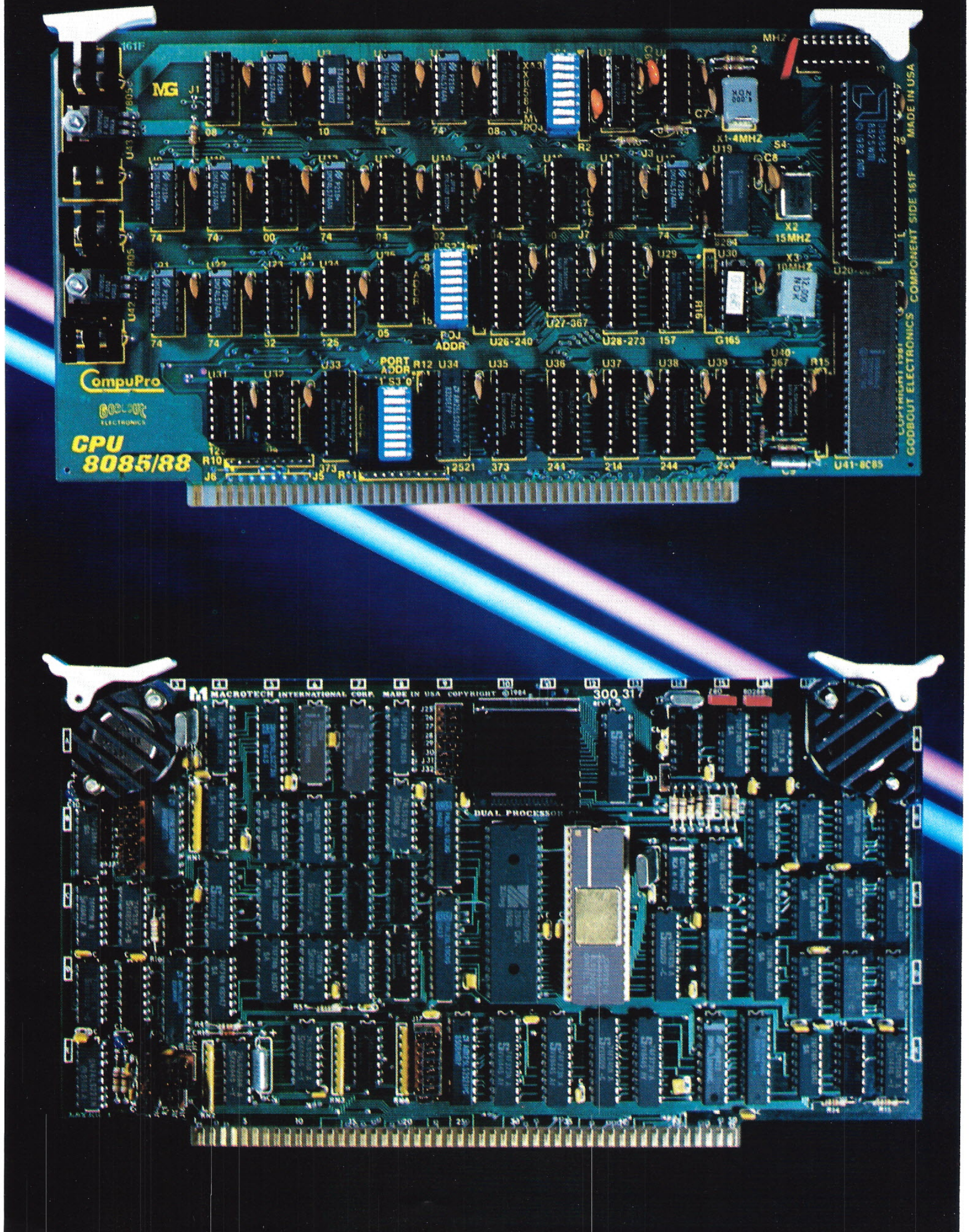
Jay Vilhena

	CompuPro	Macrotech
CPUs	16-bit: 8088 8-bit: 8085	16-bit: 80286 8-bit: Z80H
Math coprocessor	8087 (as a piggy-back board from Hudson Assoc.)	80287 optional
Maximum Memory Addressable	16 megabytes	16 megabytes
Data Bus	8 bits	8 bits (restricted) or 16 bits
Processor active on power-up	8085	Z80
8-bit speed	2 or 6 MHz	2 or 8 MHz
16-bit speed	10 MHz	6 MHz
I/O wait states	0 or 1 - switch selectable	Z80: 0 or 1 - switch selectable 286: up to 3 - switch selectable
MWRITE generator	switch selectable	switch selectable
Cost	$350	$1095
Manufacturer	CompuPro 26538 Danti Court Hayward, CA 94545-3999 800-842-7961 800-842-7962 (California)	Macrotech 9551 Irondale Ave. Chatsworth, CA 91311 800-824-3181 818-700-1501 (California)

Since so much of this issue is dedicated to dual-processor boards, I decided to throw in this small article with an at-a-glance table showing you the main features of the CompuPro CPU 8085/88 and the Macrotech MI-286.

Although 16-bit systems (and the CPU 8085/88 in particular) have been available for several years, many readers have 8-bit-only machines and may now be considering a 16-bit upgrade to run extra software. A dual-processor board is a good choice because all your older 8-bit software can still run. (Another alternative is to buy a 16-bit slave or coprocessor board available from several manufacturers.)

Both the Macrotech and the CompuPro are excellent dual-processor boards and are now essentially bug-free. The MI-286 offers outstanding performance due to a fast Z80 processor and a 16-bit data path for the 80286. The CPU 8085/88 can be obtained at extremely attractive prices. So the choice, as usually, depends on your intentions. If what you want is an economical and dependable upgrade to 16 bits, the 8085/88 is ideal. Since the 8085/88 uses an 8-bit data path for both processors, it will work fine with all your old 8-bit memory. However, if you are after the ultimate speed, want to take advantage of 16-bit memories, or plan to use the board in multiuser environments, you need the MI-286. You may also prefer the MI-286 if your current CPU is a Z80, since some Z80-specific programs may not run with the 8085.


CompuPro
GODBOUT ELECTRONICS
CPU 8085/88
GODBOUT ELECTRONICS COMPONENT SIDE 161F MADE IN USA
MACROTECH INTERNATIONAL CORP. MADE IN USA COPYRIGHT ©1984
300 317
DUAL PROCESSOR

(continued from page 13)
block. The 8088 registers are converted to the 8085 equivalents and written to the task block memory area. The 8088 also writes into the task block memory the function code for the requested BIOS function. The 8088 then issues an input instruction to activate the 8085.

3. The 8085 wakes up and loads all of its registers from the information in the task block memory, and calls the existing CP/M BIOS code for the requested function. When the existing BIOS returns, the 8085 writes all of its registers back to the task block memory area and in turn issues an input instruction that reactivates the 8088.

4. The 8088 reconverts, into their 8088 equivalents, the 8085 registers stored in the task block memory area. It then returns to CP/M-86.

Of course, things are not always as simple as one would like. The above scenario becomes more complicated when disk I/O is involved, as you will see when each BIOS function is examined in individual detail.

Memory Allocation

Memory must be carefully managed when both microprocessors on the Dual CPU are used because both processors can address and alter the same physical memory. It would be very easy (and probably disastrous) for one of the microprocessors to clobber information used by the other. Not only that, but CP/M and CP/M-86 differ significantly in their memory layout. CP/M uses low memory for a number of important fields, such as the BDOS and warm-boot vectors, and the IOBYTE. CP/M itself runs in high memory in the 8085 address space. CP/M-86 is forced by the architecture of the 8088 to use low memory for interrupt vectors. CP/M-86 itself is relocatable — that means it can run anywhere within the address space of the 8088. Typically, CP/M-86 occupies a memory area starting at address 400 (hex), with the BIOS starting at address 2500 (hex).

Somewhere in this messy picture, a secure area must be found for the BIOS task blocks. The memory allocation I used is pictured in Table 3. Low memory does cause some confusion due to conflicting usage by CP/M and CP/M-86. The code that processes the task block interface has to do some saving and restoring of key low-memory areas.

The Software Pieces

In order to implement the BIOS task block approach, I wrote three separate programs. The first program is the CP/M-86 BIOS. This is a special BIOS which runs on the 8088 and creates BIOS task blocks for the 8085 to process. The second program is called the BIOS Task Block Processor, and it runs on the 8085. It loads CP/M-86 into memory and causes the 8088 to start executing CP/M-86. A portion of the second program remains in memory to act as an interface between the CP/M-86 BIOS and the CP/M BIOS. The third program is a very simple program, called the CP/M Reboot Program, which runs on the 8088. It causes your computer to cease running CP/M-86 and resume running CP/M. This allows you to freely alternate between CP/M and CP/M-86 without the need to reset (reboot) your computer.

CP/M (8085)	CP/M-86 (8088)
A	AL
B	CH
C	CL
D	DH
E	DL
H	BH
L	BL

Table 2. *CP/M versus CP/M-86 Register Allocation. Whenever a BIOS call is invoked, processor registers are used to pass the parameters. This table depicts the translation conventions used in CP/M-86. For example, if a CP/M BIOS call used the 8085 register C to pass a parameter, the corresponding CP/M-86 BIOS call will use 8088 register CL for that same parameter.*

The CP/M-86 BIOS

Listing 1 (CBIOS86.A86) is a BIOS for CP/M-86. This code is really only a translator to allow your existing CP/M BIOS to do all the hard work. Let's examine each of the routines in this BIOS. During this discussion I will frequently refer to program labels which you can find in Listing 1.

The beginning of the BIOS contains several equates. A few of them are very noteworthy. The equate called IFAREA is used to locate the BIOS task block in memory (see Table 3). In Listing 1, the BIOS task block is located at D000 (hex). In the installation section of this article, you will learn how to locate the BIOS task block in your system. The definition of the task block format starts at the equate called CODE. The task block contains the CODE which determines the BIOS function the 8085 is to perform, the 8085 registers, the current IOBYTE and CDISK settings of CP/M (remember that CP/M and CP/M-86 conflict in their usage of low memory, so these important low-memory fields must be saved and restored), and a disk buffer (labelled DSKBUF). The disk buffer will be used by the CP/M BIOS whenever a disk read or write is requested by the 8088. The 8088 will then be responsible for moving the disk buffer's contents to wherever CP/M-86 has currently set its DMA address. Since the 8085 is not capable of addressing more than 64K, and the CP/M-86 DMA address can be outside of the bottom 64K of memory, a fixed disk buffer is required in the BIOS task block.

The BIOS INIT routine is very standard. It initializes all interrupt vectors, prints a sign-on message, and enters CP/M-86.

The console status (CONST) routine is the first place where anything unusual happens. This routine sets the CODE byte in the BIOS task block. The codes used are the offsets from the base of the CP/M BIOS jump table to the location of the jump for the requested BIOS service. Console status uses a code of 6 because the console-status jump is the third jump in the BIOS jump table; therefore, it is at offset 6 from the beginning of the table. Table 4 contains a listing of the BIOS functions and their CODES. After setting the code, the console status routine does a jump to the routine that will swap processors to perform the BIOS function.

Most of the BIOS functions are as simple as console status. The functions LISTOUT (list device output), LISTST (list device status), PUNCH (punch device output), READER (reader device input), HOME (cause the disk drive head to seek to track zero), SETTRK (select the track for the next disk operation), and SETSEC (select the sector for the next disk operation) all work the same as console status and will not be further discussed.

In CP/M-86 the IOBYTE was moved from low memory to a location inside the BIOS. Two new BIOS functions, GETIOBF (get IOBYTE) and SETIOBF (set IOBYTE), are provided. In this case, the IOBYTE itself is in the BIOS task block area so that the 8085 can examine it for any changes.

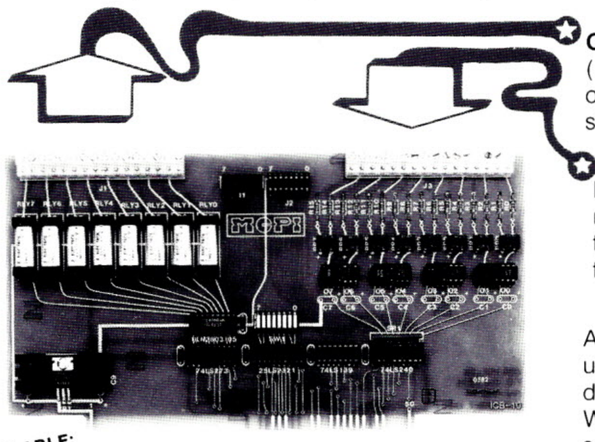
The SELDSK (select disk) routine is the most complicated routine in the CP/M-86 BIOS. Like most of the other BIOS routines, it starts out by setting the CODE byte and swapping to the 8085 to perform the SELDSK routine in the CP/M BIOS. If no errors were detected in the CP/M BIOS, a pointer to a Disk Parameter Header (DPH) is returned. The CP/M-86 BIOS uses the information returned to build a local copy of the DPH for CP/M-86 to use. When the CP/M BIOS returned its pointer to a DPH, it returned an absolute pointer to a location. Within the CP/M-86 BIOS, all references to memory are made relative to the 8088's DS segment register. (If the concept of segment registers is unfamiliar, you might want to read the series of articles on the architecture of the 8086 which have appeared in BYTE [4]. The 8086 and 8088 architectures are identical from a software viewpoint.) The DS register is always set to the base of CP/M-86, in this case 400 (hex). Before the pointer returned from the CP/M BIOS can be used, that pointer must be adjusted by subtracting the bias added by the DS register. This is accomplished by the instruction *sub bx,cpm-offset*. After the pointer is adjusted, the necessary fields are copied to the local DPH, and a pointer to the local DPH is returned to

*****S-100 & STD USERS DISCOVER DIGITAL I/O CONTROL*****

Engineers --- Security Systems Designers --- Clinical Psychologists ---Hackers

ENJOY LOW COST MICROCOMPUTER CONTROL - Professionals and hobbyists have discovered the unlimited possibilities of Digital I/O control. From automated nurseries to controlled energy systems, from instrument monitoring to an automatic cat feeder; these are only a few ways our customers are putting our Digital I/O Control Boards to work.

AUTOMATION and ROBOTICS ARE THE FUTURE. Use MULLEN CONTROLLER BOARDS to extend your hands, eyes, ears, even your feet through sensing and control of "Real World" operations.



OUTPUT: Your computer drives *eight* individual relays (mercury-wetted or dry-reed). These relays can operate almost anything requiring a digital (ON/OFF) state.

INPUT: Electrical signals from the outside world can be monitored by *eight* opto-isolated channels. Sensing these signals then allows conditional control through the OUTPUTS.

All MULLEN products are supported by easy to read user manuals that include programming examples, diagrams of typical connections, and schematics. Write or call for data sheets that have detailed specifications, prices and terms.

AVAILABLE:
S-100 Bus (shown) and STD Bus controllers.
Available soon for IBM PC®

MCPI

MULLEN COMPUTER PRODUCTS, INC. has supplied high quality, products to universities, laboratories and industrial users since 1976. Write to **MCPI** at 2260 American Ave. #1, Hayward, CA 94545 or call (415) 783-2866.

Texas Instruments and Understanding Series are trademarks of Texas Instruments. IBM PC is a registered trademark of IBM.

CP/M-86. One of the fields in the copied DPH (the sector translate table pointer) must also be adjusted for the DS register offset. Incidentally, when I originally wrote this BIOS, I tried not making a local copy of a DPH and just passing to CP/M-86 the adjusted pointer to the DPH in the CP/M BIOS. This did not work properly, and I was not able to find out why.

The SETDMA (set DMA address) and SETDMAB (set DMA base address) BIOS functions are handled without calling the CP/M BIOS. These two functions save pointers to where disk records are found on disk reads or disk writes.

The READ function calls the CP/M BIOS to read one disk record. Remember that the CP/M BIOS uses a fixed disk buffer which is stored in the BIOS task block area. After calling the CP/M BIOS, the READ function transfers the data from the BIOS task block area to the address specified by the current DMA address. The WRITE function is very similar. Before calling the CP/M BIOS, the WRITE function copies the data from the current DMA address to the BIOS task block area. The CP/M BIOS is then called to do the actual disk write.

The CALLCB is the routine where the CP/M BIOS gets called. First, the register translation is performed according to the mapping shown in Table 2. Second, the IN AL,SWAP instruction shuts down the 8088 and wakes up the 8085. After the 8085 is finished processing the BIOS task block, it reawakens the 8088 which starts executing right where it left off. Lastly, the register translation is performed in reverse, and the routine exits.

The remainder of the CP/M-86 BIOS consists of data areas. The only one of any real interest is the *segtable*. This table defines for CP/M-86 what memory is available in the computer. In Listing 1, this table is set so the area between the end of the CP/M-86 BIOS and the BIOS task block is available for program use (see Table 3). If you have memory at or above address 10000 (hex), you will want to change this table to add the other memory areas. The first byte of the table tells how many entries are in the table. The remaining entries are two words each. The first word is the base paragraph address (physical address divided by sixteen) of the available memory; the second word is the number of paragraphs (16-byte chunks) available in this region. Up to eight noncontiguous memory regions may be defined.

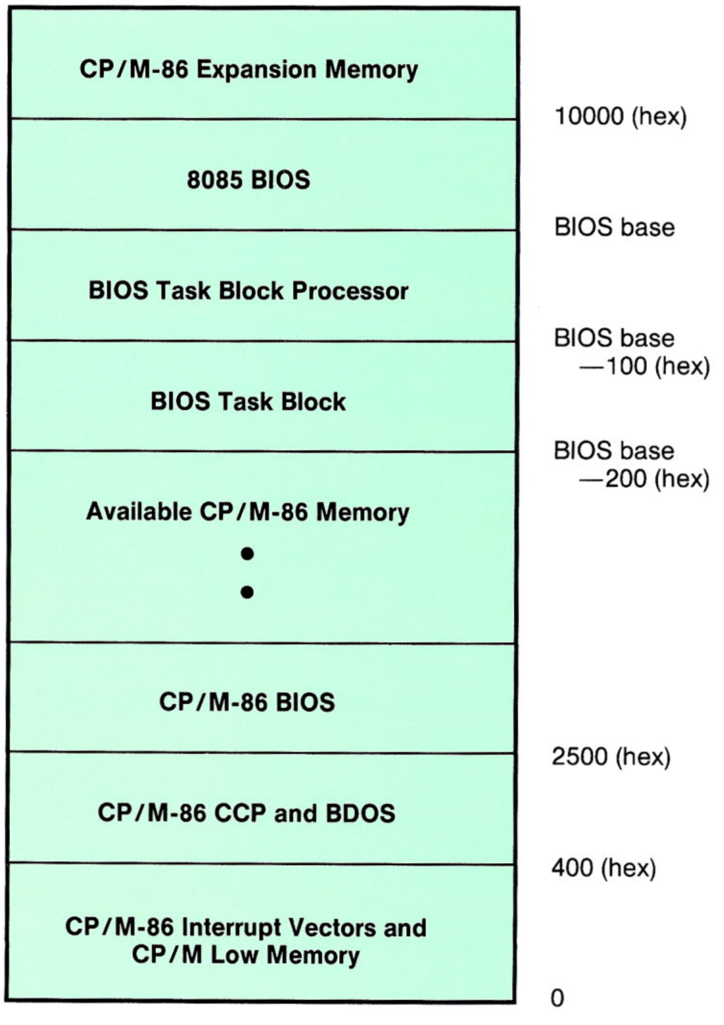


Table 3. *Memory Allocation. This table depicts the memory of the computer as it looks when CP/M-86 is running, and the 8085 is providing BIOS services for the 8088. The area between the top of the CP/M-86 BIOS and the bottom of the BIOS task block is available for running programs. If more than 64K of memory is available, any memory above the bottom 64K (memory above address 10000 hex) can also be used for running programs.*

The BIOS Task Block Processor

Listing 2 (BOOT86.ASM) is a program which runs on the 8085. This program has four responsibilities. Its first job is to load the CP/M-86 system file (a file containing the CP/M-86 CCP, BDOS, and BIOS) into memory. Second, it attempts to provide for the 8088 reset vector. Third, it saves part of the 8085 environment in the task block area so that some important variables are not smashed by CP/M-86. Fourth, it relocates a portion of itself to the BIOS task block area, and functions as the interface between the BIOS task block and the CP/M BIOS. The major sections of Listing 2 are identified by comments, and you may want to refer to each section during the following discussion.

Loading the CP/M-86 system file is straightforward. However, beware that object files under CP/M-86 have, as their first sector, an information sector on how to set segment registers after loading. This sector is of no use and must be discarded. Since the CP/M-86 file must be loaded at 400 (hex), loading is started at 380 (hex) to discard this first sector. The load code terminates at the label EOF.

The section of code that saves some of the important 8085 memory regions simply copies the contents of this memory to secure locations in the task block. The regions copied are those that might be smashed by the 8088 when running CP/M-86. The area at FFF0 is saved as part of providing for an 8088 reset vector.

The next section of code attempts to provide for the 8088 reset vector (the reset vector is an 8088 jump instruction to the address where the 8088 should start executing). This is a tricky problem. A restart vector must also be provided. The reset vector is used the very first time that the 8088 is turned on. The restart vector is used every subsequent time that the 8088 is turned on. The text box on page 28 details the problems of providing for reset and restart vectors on the Dual CPU.

Following the code for the 8088 reset vector, there is a short loop which relocates the remainder of the program to the BIOS task block area. This relocation places the remaining code in a secure area of memory (just after the task block). The relocated code must remain resident during CP/M-86 operation because it processes BIOS requests for the 8088. This code is relocated to the BIOS task block area, rather than simply being loaded at the task block area, because it overlays a portion of the CP/M BDOS.

The definition of the task block area follows. This definition must match the definition provided in the CP/M-86 BIOS. The code that is relocated starts at the label HIMEM. All the jump instructions in this section look funny because of the relocation.

The code that starts at the label HIMEM processes task block requests. The first thing it does is to allow the 8088 to run. CP/M-86 has been loaded into memory, and the reset vectors have been set for the 8088 to begin running at the CBOOT (cold boot) entry point of the CP/M-86 BIOS. After the 8088 gets an initial chance to run, the memory at FFF0 which was altered to provide a reset vector is restored from the values saved in the task block area.

The code starting at the label NXTCMD begins a loop that will continue to execute for as long as CP/M-86 is running. NXTCMD is the entry point for processing task block requests. The CP/M-86 low-memory environment is pushed onto the stack and the CP/M low-memory environment is restored from the values saved in the task block area. A special check is made to see if the task block is requesting a return to CP/M operation (the task code for this special request is FF). If CP/M is to be resumed, the warm-boot vector is rebuilt in low memory, and a jump to 0 warm boots CP/M back into memory. If the task block is requesting a BIOS operation for CP/M-86, the

(continued on page 25)

CODE (hex)	Function
00	Cold Boot
03	Warm Boot
06	Console Status
09	Console Input
0C	Console Output
0F	List Device Output
12	Punch Device Output
15	Reader Device Input
18	Home Disk
1B	Select Disk
1E	Set Disk Track
21	Set Disk Sector
24	Set Disk DMA Address
27	Read Disk
2A	Write Disk
2D	List Device Status
30	Translate Logical Disk Sector to Physical Disk Sector
FF	Terminate CP/M-86 and Resume CP/M Operation

Table 4. *BIOS Functions and BIOS Task Block CODES. This table lists the codes used in the CODE field of the BIOS task blocks. The CODE field is used by the 8088 to inform the 8085 which BIOS function is to be performed. The values for the codes are actually the offset of the requested jump vector from the beginning of the BIOS jump vector table.*

PROGRAM LISTINGS

The listings for the three programs described in this article are published in the S-100 Journal Supplement distributed with this issue.

They are also available on standard IBM format 8" single-sided, single-density diskettes from Howard Spindel, 20877 S.W. Winema Drive, Tualatin, Oregon 97062. There is a handling charge of $30.00 which includes the disk and shipping by United States Postal Service.

(continued from page 8)

systems were sold.

However, when the Z80 CPU was released, great problems emerged. Many vendors tried to simulate the 8080's signals when they connected the Z80 to the S-100 bus. Unfortunately, most of the designers were unaware of how other vendors were using the bus signals and which timing parameters those vendors were using. The result was general incompatibility between the various products.

DMA was another sore point. The bus supported DMA but it tended to work very poorly. During bus exchanges, glitches commonly occurred on the positive true MWRT (memory write) signal. Also, many dynamic memory boards failed to handle refresh cycles properly during DMA.

THE IEEE STANDARD IS INTRODUCED

A small group of S-100 vendors and users got together and, with great effort, generated a rough-draft specification for the bus. The goals were to create a standard that allowed all boards to work together, cleaned up known problem areas of the bus, and expanded the bus to support more/wider memory (and I/O) as well as more processors. In order to attain these goals, many creative ideas went into the rough draft.

The proposed standard was first made public in the July 1979 issue of the "IEEE Computer" magazine. By this time the standard was at draft level D2 (the second draft). This draft is probably the best known. It was also published in JAN/FEB 1980 issue of "S-100 Micro Systems" and in some books on the S-100 bus. This exposure allowed many people to examine and comment on the proposed standard, promoting the design and building of newer, more powerful cards. The proposed standard brought major improvements to the bus (see Table 1). These changes, along with others, allowed the power and flexibility of the S-100 bus to expand. At the same time, it permitted most older S-100 designs to work with the newer designs.

The final meeting of the IEEE-696 standards committee was on June 30th, 1981. By then the draft was at level D4. Some final changes were made, and a copy of the final draft (D5) was mailed to all past and present committee members for a vote. The draft passed the committee, and on June 10, 1982, it was approved by the IEEE Standards Board. The current active standard is officially called "IEEE Standard 696 Interface Devices" and it has the IEEE designation IEEE Std 696-1983. It was first published by the IEEE organization on June 13, 1983.

DRAFT TO FINAL STANDARD — THE CHANGES

Draft D2 succeeded in better defining and vastly improving the functionality and speed of the bus. But it still contained some deficiencies. The major changes between draft D2 and the final standard were in two areas: the 16-bit data bus interface and the temporary master interface.

The draft's 16-bit data bus was labeled as having high- and low-byte significance. This created many problems since there are two ways of storing 16-bit numbers in byte-addressable memory. The 68000-type processors do it by storing the high byte of the word in low memory. The 8080/8086-type processors do it the reverse way. The problem occurs when both of these devices exist on the same bus, attempting to share number data in memory. After spending months on this problem, the committee realized that sharing binary numbers in memory was

Original S-100	IEEE Draft D2
Supported 8-bit data transfers only	Supports 8 and/or 16-bit data transfers
Addressability up to 64K Bytes of RAM	Addressability up to 16 Megabytes of RAM
Addressability up to 256 bytes of I/O	Addressability up to 64K bytes of I/O
Bus clock speed at 2 Mhz	Bus clock speeds to 6 Mhz
Only 1 temporary bus master allowed	Up to 16 temporary bus masters allowed
No timing specifications	AC timing specifications
No DC specifications	DC driver/receiver and terminator specifications

Table 1. *Major differences between IEEE Draft D2 and the Original S-100.*

not the real issue. But, instead, that any given processor should be able to properly read in 8-bit mode whatever it may have written in 16-bit mode, and vice versa. To insure this, the two 8-bit halves of any 16-bit transfer are now labeled as the "odd byte" and the "even byte." The odd byte goes to the odd-byte address while the even byte goes to the even-byte address. Bit significance is no longer implied.

The 16-bit data bus underwent one other major change. Draft D2 defined: "if a 16-bit write to memory is attempted to an 8-bit device, it is recommended that the write be performed in hardware as two 8-bit writes." This would take three memory accesses to perform: one to detect the error, and two more to write the word as two bytes. It was realized that if the proper data byte appeared on the 8-bit DO (Data Out) bus, then the error detection and the writing of the first byte could be combined in the same memory access. Now only two memory accesses are needed. To make this work, the equations for controlling 8/16-bit memories have the active level of address bit A0 reversed from its value in the draft.

The temporary master interface was changed mostly in name. Draft D2 calls the bus arbitration lines DMA0* to DMA3*. DMA implied direct memory access, which in turn implied that a temporary master could only do memory-type accesses. In fact, temporary masters are allowed to perform ALL types of bus cycles, including I/O cycles. It was therefore decided that the arbitration lines should be named TMA0* to TMA3*, which stands for Temporary Master Access of the bus.

One other change, dealing with the temporary master interface, was made to the IEEE-696 specification. An error existed in the example arbitration circuit. If TMA priority device 0 (the lowest priority device) requested and won the bus, then all TMA devices on the bus would have received the arbitration win signal. This error and a timing glitch were both fixed in the example circuit.

The final specification also varies from draft D2 in the following ways:

1. An IDLE bus state was defined and added.
2. The PHANTOM* signal was initially part of the utility bus only. It is now mentioned with the address bus as well, with timing specifications added.
3. POC* (Power On Clear) is better defined. Timing specifications were added.
4. Powerfail specifications were changed.
5. Termination methods of open-collector lines were changed.
6. Some timing parameters were changed and others were added.
7. A 10" double-high board was defined.

CONCLUSION

The S-100 bus has come a long way in its ten years of existence. Today, mainly due to the IEEE specification, it is a powerful and supported bus.

Anyone wishing a copy of the final specification of the IEEE-696 bus can write to the IEEE Computer Society Order Dept., P.O. Box 80542, Worldway Postal Center, Los Angeles, CA 90080. The price is $6.75 for IEEE members, or $7.50 for non-members (plus $2.00 shipping and handling — California residents add sales tax). It can also be obtained from IEEE Service Center, CP department, 445 Hoes Lane, Piscataway, NJ 08854 (New Jersey residents add sales tax).

NEXT TIME

In the next issue, I'll cover in more detail the differences between draft D2 and the standard, and I will give an example of a fast wait state generator.

TERMINALS

ADDS REGENT 200s — These exceptionally smart terminals have separate numeric, cursor control, & function key areas, and support the following software-selectable attributes: Normal, half-intensity, underlined, graphic, blinking, & reverse video, protected and unprotected fields. These attributes may be used together, and can cover all or any part of a screen.

Online/block/page transmission modes. Switch selectable: RS-232/Current loop; 75-19,200 baud; auto scroll on/off; upper only or upper & lower case; normal/reverse video; half/full duplex; even/odd/no parity. $225 each.

MICRODATA PRISM 4700s — Industrial grade RS-232 dumb terminal. Displays upper case only, generates upper/lower case. Numeric keypad, heavy metal housing, detached keyboard w/5' cord, internal fan for hostile environments. Switch selectable: 110-9600 baud; half/full duplex; normal/reverse video; even/odd/no parity. Power cord $5 extra. Excellent for control projects. $150 each.

These terminals have each been cleaned, completely refurbished, and tested prior to shipping. CRTs are all in excellent shape. Buyer pays freight. I guarantee satisfaction, or money (incl. freight) refunded within 30 days of purchase.

Dave Simpson, 151 Warwick, Park Forest, Ill. 60466-1627 or phone 312-747-3107 12-9 pm CDT. Quantity prices available.

HUMOR

Whenever space permits, we will continue to place a cartoon or two in S-100 Journal.

If you have a good idea for a computer-related (preferably S-100) cartoon, please send it to us. It must be an original. It can be already drawn or simply send us the idea. If your cartoon is published, we will enter or extend your subscription to S-100 Journal for 2 years.

SOURCES AND RELATIONS

Gary Feierbach

Multiuser OS *is a new S-100 Journal column that discusses all multiuser operating systems running on S-100 machines.*

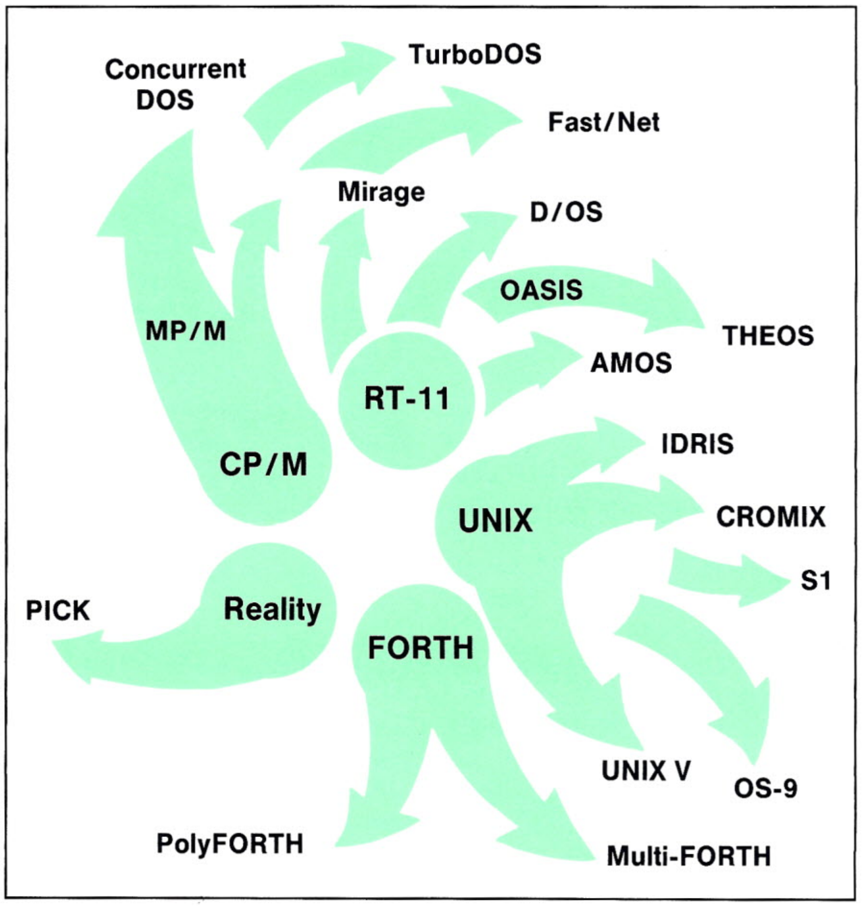
S-100 Journal welcomes Gary Feierbach as our regular columnist for Multiuser OS. Gary has over 20 years of experience as a computer professional and holds B.A. and M.S. degrees in Computer Science and E.E. from the University of California at Berkeley. He has worked in numerous operating-system environments and is actively providing implementations of various multiuser operating systems on S-100 machines. Gary is president of Inner Access Corporation, one of those intrepid companies that is ready to offer new S-100 products forever.

Readers are invited to write to Gary and send questions, tidbits of information, or gossip about any S-100 multiuser operating system. He will try to work them into future columns. Write to Gary Feierbach, P.O. Box 888, Belmont, CA 94002.

This time, Gary brings a little order to the OS families and provides a list of S-100 implementations, complete with addresses. If you have any additions or updates to this list, he invites your input.

How does one inaugurate a column? Do I break a bottle of champagne over the bow of an S-100 system on its way to the loading dock? S-100 systems have traveled far since the Altair days, and along the way some systems did resemble battleships (e.g., the Cromemco Z-2D). Many of these early battleships are still doing useful work for engineers and developers.

The advent of the Apple II and the IBM PC has in large measure gobbled up the personal computer and engineering workstation market. While this went on, S-100 did not stand still. With its revamped IEEE-696 high-bandwidth specification, it invaded minicomputer territory — the multiuser systems. Today, in spite of many industry crosscurrents (the VME, the Multibus II, the Nubus, and a variety of other contenders), the S-100 multiuser arena continues to grow and prosper. It is my opinion that we will continue to see the introduction



O/S	SOFTWARE VENDOR	HARDWARE VENDORS
AMOS	Alpha Micro	Alpha Micro, Inner Access
Concurrent DOS	Digital Research	High Tech, CompuPro, Lomas, Advanced Digital, Macrotech
CROMIX	Cromemco	Cromemco
D/OS	Dravac	Alpha Micro, Dravac, Inner Access
Fast/Net	Newtons Labs	Newtons Labs
Mirage	Sahara Ltd.	Empirical, Inner Access
MP/M	Digital Research	Macrotech, Intercontinental, High Tech
Multi-FORTH	Creative Solutions	Any CP/M system
THEOS	Theos Software Group	Third Coast
OS-9	Microware Systems	
PICK	Pick Systems	Climax
PolyFORTH	FORTH Inc.	Any CP/M system
S1	Multi Solutions	CompuPro (from Multi Solutions)
TurboDOS	Software 2000	Northstar, Intercontinental, Teletek Advanced Digital, JC Systems
UNIX V	AT&T or Unisoft	Cromemco, Empirical, Dual, High Tech, Dynacomp
IDRIS	Whitesmith	Empirical

S-100 multiuser operating systems and respective vendors.

of new S-100 products well beyond the year 2000.

Now we need to take a look at the players in the arena. I am referring to operating systems and not to S-100 manufacturers although many of the operating systems are strongly identified with particular hardware.

The systems that I will describe in the future generally trace their lineage to common parents within a small set of families. One well-known family is UNIX which has evolved and spawned several variants and work-alikes, including UNIX V, CROMIX, IDRIS and S1. Another is RT-11 which inspired AMOS, Mirage, and D/OS. Another, CP/M, evolved into MP/M and eventually into Concurrent DOS. RT-11 can also be considered the inspiration for OASIS which evolved into THEOS. Then, there are the composite systems like TurboDOS and Fast/Net that meld two or more hardware and software systems together, in the latter case Mirage and CP/M. TurboDOS and Fast/Net support multiprocessors; each slave processor is doled out to one or more users, and the bus master handles such tasks as file management and printer spooling.

Less known systems are Reality and Forth. Reality has become PICK. The multiuser versions of FORTH are Poly-FORTH and Multi-FORTH. Finally, there are the real-time systems like OS-9, and various real-time kernels some of which offer multiuser capability.

This will certainly give me plenty to talk about in the next several issues. Next time, I will start with AMOS, Mirage, D/OS and Fast/Net since they have much to offer, and many readers are probably not very familiar with them.

The table on this page shows the above mentioned operating systems correlated to software vendor and the particular S-100 hardware it runs on. ■

ADDRESSES

Advanced Digital Corp.
5432 Production Dr.
Huntington Beach, CA 92649
(714) 891-4004

Alpha Micro
3501 Sunflower St.
Santa Ana, CA 92704
(714) 957-8500

AT&T
600 Mountain Ave.
Murry Hill, NJ 07974
(201) 582-3624

Climax
3605 W. MacArthur, Suite 702
Santa Ana, CA 92704
(714) 557-2398

CompuPro
26538 Danti Ct.
Hayward, CA 94545
(415) 786-0909

Creative Solutions
4801 Randolph Rd.
Rockville, MD 20852
(301) 984-0262

Cromemco
280 Bernado Ave.
Mt. View, CA 94039
(415) 964-7400

Digital Research
60 Garden Ct.
P.O. Box DRI
Monterey, CA 93942
(408) 649-3896

Dravac Ltd
16 Muller Road
Oakland, NJ 07436
(201) 337-8350

Dual Systems Control Corp.
2530 San Pablo Ave.
Berkeley, CA 94702
(415) 549-3854

Dynacomp Computer Systems Ltd.
100-210 W. Broadway
Vancouver, B.C. Canada V5Y 3W2

Empirical Research Group Inc.
1112 S. 344th St., Suite 310
Federal Way, WA 98003
(206) 874-4844

Forth Inc.
2309 Pacific Coast Hwy.
Hermosa Beach, CA 90254
(213) 372-8493

High Technology Electronics
303-305 Portswood Rd.
Southampton, England S02 1LD

Inner Access Corporation
3206 E. Laurel Ck. Rd.
Belmont, CA 94002
(415) 591-8295

Intercontinental Micro Systems
4015 Leaverton Ct.
Anaheim, CA 92807
(714) 630-0964

JC Systems
469 Valley Way
Milpitas, CA 95035
(408) 945-0318

Lomas Data Products
Hopkington Rd.
Westboro, MA 01581
(617) 460-0333

Macrotech International Corporation
9551 Irondale Ave.
Chatsworth, CA 91311
(818) 700-1501

Microware Systems
1866 N.W. 114th St.
Des Moines, IA 50322
(515) 224-1929

MultiSolutions, Inc.
123 Franklin Corner Rd. #207
Lawrenceville, NJ 08648
(609) 896-4100

Newtons Laboratories
P.O. Box 789
111-113 Wandsworth High St.
London, England SW184JB

Northstar Computers
14440 Catalina St.
San Leandro, CA 94577
(415) 357-8500

Pick Systems
1691 Browning
Irvine, CA 92714
(714) 261-7425

Sahara Ltd.
Unit 1F
Tideway Industrial Estate
87 Kirtling St.
London, England SW8 5BP
(01) 627-1733

Software 2000
1127 Hetrick Ave.
Arroyo Grande, CA 93420
(805) 489-1977

Teletek
4600 Pell Drive
Sacramento, CA 95838
(916) 920-4600

Theos Software Corp.
201 Lafayette Cir., Suite 100
Lafayette, CA 94549
(415) 283-4290

Third Coast Technology Inc.
555 Pilgrim Dr., Suite B
Foster City, CA 94404
(415) 570-4641

Unisoft Systems
739 Allston Way
Berkeley, CA 94710
(415) 644-1230

Whitesmith Ltd.
97 Lowell Rd.
Concord, MA 07142
(617) 369-8499

VOL. 1, NO. 2 • FALL 1985

S-100 journal

Supplement

This supplement contains the source code of the programs

CBIOS86.A86
BOOT86.ASM
BOOT80.A86

described in the article

CP/M-86 ON OLDER S-100 COMPUTERS USING A DUAL-PROCESSOR BOARD

Programs by Howard Spindel

```
                ; LISTING 1 - CBIOS86.A86
                ; THIS PROGRAM COPYRIGHT 1983 BY HOWARD SPINDEL.
                ; PERMISSION GRANTED TO REPRODUCE THIS PROGRAM
                ; FOR INDIVIDUAL NON-PROFIT USE ONLY.

                ; This Customized BIOS adapts CP/M-86 to any CP/M hardware
                ; system built around the Godbout Dual CPU processor board.
                ; Except for requiring the Dual CPU, this BIOS is hardware
                ; independent from the target computer.  Hardware independence
                ; is achieved by allowing all I/O operations to be handled
                ; by an existing CP/M BIOS running on the 8085.  This
                ; BIOS builds task blocks describing the requested I/O
                ; operation, places the task blocks in a secure memory area,
                ; wakes up the 8085 to perform the I/O operation, and waits
                ; for the 8085 to reawaken the 8088 so that the results of
                ; the I/O operation can be obtained from the task block.

                ; A couple of useful equates
FFFF            true            equ     -1
0000            false           equ     not true

                ; cpm_offset is the offset of the base of CP/M-86 from absolute
                ; zero (which also defines the setting of the cs register on
                ; entry to CP/M-86).  400h is the standard setting, but CP/M-86
                ; is relocatable and may run anywhere in the 8088 address space
                ; above the interrupt vectors.
0400            cpm_offset      equ     0400h           ;cs register on entry to cpm

                ; Specify the user changeable equates near the front of the
                ; listing so they are easily found.

CC00            IFAREA          equ     0D000H - cpm_offset     ;task block address
00FD            SWAP            equ     0FDH            ;Dual CPU processor swap port
0000            SAVLOWMEM       equ     false           ;Debugging equate

                ; Maximum number of disks in the system.  May be changed for
                ; different hardware configurations if necessary.
0004            maxdisk         equ     4

                ; a couple more useful equates
000D            cr              equ     0dh             ;carriage return
000A            lf              equ     0ah             ;line feed

                ;Equates for a standard Disk Parameter Header (DPH)
0000            xlt             equ     0               ;translate table pointer
0008            dirbuf          equ     8               ;directory buffer pointer
000A            dpb             equ     0AH             ;disk parameter block ptr
000C            csv             equ     0CH             ;checksum vector pointer
000E            alv             equ     0EH             ;allocation vector pointer
```

```
                        ; Equates for the relevant fields of the BIOS task block area.
                        ; This definition must match the definitions present in
                        ; BOOT86.ASM and BOOT80.A86
  CC00                  CODE        EQU     IFAREA          ;Requested BIOS function
  CC01                  AREG        EQU     IFAREA+1        ;8085 A register
  CC02                  CREG        EQU     IFAREA+2        ;8085 C register
  CC03                  BREG        EQU     IFAREA+3        ;8085 B register
  CC04                  EREG        EQU     IFAREA+4        ;8085 E register
  CC05                  DREG        EQU     IFAREA+5        ;8085 D register
  CC06                  LREG        EQU     IFAREA+6        ;8085 L register
  CC07                  HREG        EQU     IFAREA+7        ;8085 H register
  CC08                  IOBYTE      EQU     IFAREA+8        ;I/O redirection byte
  CC09                  CDISK       EQU     IFAREA+9        ;current default disk

  CC80                  DSKBUF      EQU     IFAREA+080H     ;disk dma address for
                                                            ; 8085 BIOS

                        ; Software interrupt vector used for CP/M-86 system calls
  00E0                  bdos_int    equ 224                 ;reserved BDOS interrupt

                        ; Equates for the base addresses of CP/M-8 CCP, BDOS, and BIOS
  2500                  bios_code   equ 2500h
  0000                  ccp_offset  equ 0000h
  0B06                  bdos_ofst   equ 0B06h ;BDOS entry point

                                cseg
                                org     ccpoffset
                        ccp:
                                org     bios_code

                        ; BIOS jump vector table
                        ; This table must start at the beginning of the BIOS.
                        ; All accesses to BIOS functions go through this jump
                        ; table, so the order of the entries in the table may
                        ; not change.

2500 E93C00     253F            jmp INIT        ;cold bootstrap entry
2503 E95400     255A            jmp WBOOT       ;warm bootstrap entry
2506 E98800     2591            jmp CONST       ;console status
2509 E98D00     2599            jmp CONIN       ;console input
250C E99700     25A6            jmp CONOUT      ;console output
250F E99C00     25AE            jmp LISTOUT     ;list device output
2512 E9A900     25BE            jmp PUNCH       ;punch device output
2515 E9AE00     25C6            jmp READER      ;reader device input
2518 E92601     2641            jmp HOME        ;seek track 0 on current disk
251B E9CB00     25E9            jmp SELDSK      ;select a disk
251E E92801     2649            jmp SETTRK      ;set disk track
2521 E92D01     2651            jmp SETSEC      ;set disk sector
2524 E93E01     2665            jmp SETDMA      ;set memory offset for disk I/O
2527 E94901     2673            jmp READ        ;read disk sector (128 bytes)
252A E96401     2691            jmp WRITE       ;write disk sector (128 bytes)
252D E98600     25B6            jmp LISTST      ;list device status
2530 E92601     2659            jmp SECTRAN     ;translate logical to physical disk sector
2533 E93401     266A            jmp SETDMAB     ;set memory base for disk I/O
2536 E93601     266F            jmp GETSEGT     ;return pointer to memory segment table
2539 E99200     25CE            jmp GETIOBF     ;return current I/O redirection byte
253C E99400     25D3            jmp SETIOBF     ;set new I/O redirection byte

                        ; Cold boot initialization entry point
                        ; Initialize segment registers, interrupt vectors, and
                        ; print the sign on message.

                        INIT:
253F 8CC8                       mov     ax,cs           ;set all the segment registers
2541 8ED0                       mov     ss,ax           ; to the base address of CP/M-86
```

```
2543 8ED8                   mov     ds,ax
2545 8EC0                   mov     es,ax
                                                ;set a local stack for initialization
2547 BC9C2A                 mov     sp,offset stkbase

                            IF      SAVLOWMEM   ;check for debugging equate on
                            call    savelow     ;save 8085 environment
                            ENDIF

254A E81000        255D     call    makvec      ;make low ram interrupt vectors

254D BB4127                 mov     bx,offset signon
2550 E88500        25D8     call    pmsg        ;print signon message
                                                ;default to CURRENT DRIVE on coldstart
2553 8A0E09CC               mov     cl,byte ptr .CDISK
2557 E9A6DA        0000     jmp     ccp         ;jump to CP/M-86 CCP

                   ; Warm boot entry point.  Simply jump to the CCP.

255A E9A9DA        0006 WBOOT: jmp  ccp+6       ;jump to CP/M-86 CCP at warm boot entry

                   makvec:
255D FC                     cld                 ;set forward direction

255E 1E                     push    ds
255F 06                     push    es
2560 B80000                 mov     ax,0
2563 8ED8                   mov     ds,ax       ;set up for string move operation
2565 8EC0                   mov     es,ax

                   ;set all interrupt vectors to point to the invalid interrupt trap

                                                ;first set interrupt 0 to invalid
                                                ; interrupt trap
2567 C70600008525           mov     int0_offset,offset int_trap
256D 8C0E0200               mov     int0_segment,CS
2571 BF0400                 mov     di,4
2574 BE0000                 mov     si,0        ;Now copy the interrupt 0 vector
2577 B9FE01                 mov     cx,510      ; to all 256 interrupts
257A F3A5                   rep     movs ax,ax

                   ;set the BDOS interrupt vector to point to the BDOS entry point

257C C7068003060B           mov bdos_offset,bdos_ofst
2582 07                     pop     es
2583 1F                     pop     ds
2584 C3                     ret

                   ; Entry point for invalid interrupt trap.
                   ; This routine will catch any unexpected interrupts.

                   int_trap:
2585 FA                     cli                 ;no more interrupts allowed
2586 8CC8                   mov     ax,cs
2588 8ED8                   mov     ds,ax       ;get BIOS data segment
258A BB7A27                 mov     bx,offset int_trp
258D E84800        25D8     call    pmsg        ;notify user of our problem
2590 F4                     hlt                 ;crash and burn

                   ; Console status routine
                   ; Input: None
                   ; Output: AL = FF -> character ready
                   ;         AL = 00 -> no character ready
```

```
                           CONST:
                                                           ;Get status from the 8085
2591 C60600CC06                   mov     byte ptr .CODE,6
2596 E91701         26B0          jmp     callcb

                           ; Console input routine
                           ; Input: None
                           ; Output: AL = character read

                           CONIN:
2599 E8F5FF         2591          call    const            ;is data available?
259C 74FB           2599          jz      conin            ;if not, wait for some

                                                           ;Ask the 8085 to read the data
259E C60600CC09                   mov     byte ptr .CODE,9
25A3 E90A01         26B0          jmp     callcb

                           ; Console output routine
                           ; Input: CL = character to send
                           ; Output: None

                           CONOUT:
                                                           ;Ask the 8085 to send the data
25A6 C60600CC0C                   mov     byte ptr .CODE,0CH
25AB E90201         26B0          jmp     callcb

                           ; List device output routine
                           ; Input: CL = character to send
                           ; Output: None

                           LISTOUT:
                                                           ;Ask the 8085 to send the data
25AE C60600CC0F                   mov     byte ptr .CODE,0FH
25B3 E9FA00         26B0          jmp     callcb

                           ; List device status routine
                           ; Input: None
                           ; Output: AL = FF -> list device ready
                           ;         AL = 00 -> list device busy

                           LISTST:
                                                           ;Ask the 8085 for the status
25B6 C60600CC2D                   mov     byte ptr .CODE,2DH
25BB E9F200         26B0          jmp     callcb

                           ; Punch device output routine
                           ; Input: CL = character to send
                           ; Output: None

                           PUNCH:
                                                           ;Ask the 8085 to send the data
25BE C60600CC12                   mov     byte ptr .CODE,12H
25C3 E9EA00         26B0          jmp     callcb

                           ; Reader device input routine
                           ; Input: None
                           ; Output: AL = character read

                           READER:
                                                           ;Ask the 8085 to read the data
25C6 C60600CC15                   mov     byte ptr .CODE,15H
25CB E9E200         26B0          jmp     callcb
```

```
                          ; Return the current value of the I/O redirection byte (IOBYTE)
                          ; Input: None
                          ; Output: AL = current IOBYTE

                          GETIOBF:
25CE 8A0608CC                     mov     al,byte ptr .IOBYTE
25D2 C3                           ret

                          ; Set the I/O redirection byte (IOBYTE) to a new value
                          ; Input: CL = new IOBYTE
                          ; Output: None

                          SETIOBF:
25D3 880E08CC                     mov     byte ptr .IOBYTE,cl
25D7 C3                           ret

                          ; Handy little routine which sends a string to the
                          ; console output device.  The string must be terminated
                          ; by a binary zero.
                          ; Input: BX points to string (based on DS)
                          ; Output: None

                          pmsg:
25D8 8A07                         mov     al,[bx]         ;get the next char to send
25DA 84C0                         test    al,al           ;end of the string?
25DC 740A           25E8          jz      return          ;yup, found the end of the string
25DE 8AC8                         mov     cl,al           ;move character to appropriate reg
25E0 53                           push    bx              ;careful, CONOUT smashes BX
25E1 E8C2FF         25A6          call    CONOUT          ;send the character to the console
25E4 5B                           pop     bx
25E5 43                           inc     bx              ;point to next character of message
25E6 EBF0           25D8          jmps    pmsg            ; and do it all again
                          return:
25E8 C3                           ret

                          ; Select a new disk.
                          ; Input: CL = requested disk
                          ; Output: BX = pointer to selected DPH (0000 if error)

25E9 BB0000               SELDSK: mov     bx,0000         ;prepare bad return
25EC 80F904                       cmp     cl,maxdisk      ;good select?
25EF 7328           2619          jae     badsel          ;nope
                                                          ;ask the 8085 to select the disk
25F1 C60600CC1B                   mov     byte ptr .CODE,1BH
25F6 E8B700         26B0          call    callcb
25F9 0BDB                         or      bx,bx           ;bad select?
25FB 741C           2619          jz      badsel          ;yes
25FD E81C00         261C          call    findph          ;get pointer to DPH in DI
2600 81EB0004                     sub     bx,cpm_offset   ;correct for seg reg bias
2604 8B07                         mov     ax,[bx]+xlt     ;get translation tbl ptr
2606 8905                         mov     [di]+xlt,ax     ;put in local table
2608 8B470A                       mov     ax,[bx]+dpb     ;get dpb ptr
260B 89450A                       mov     [di]+dpb,ax     ;put in local table
                                                          ;correct some fields for seg reg bias
260E 812D0004                     sub     [di]+xlt,cpm_offset
2612 816D0A0004                   sub     [di]+dpb,cpm_offset
2617 8BDF                         mov     bx,di           ;return result to CP/M-86
2619 0BDB                 badsel: or      bx,bx           ;set condition codes for CPM
261B C3                           ret
```

```
                          ; Subroutine to get a pointer to the correct local copy of
                          ; a Disk Parameter Header (DPH).
                          ; Input: BX = pointer to DPH on 8085
                          ; Output: DI points to local DPH

                          findph:
261C 51                          push    cx
261D B90400                      mov     cx,maxdisk
2620 BF3127                      mov     di,OFFSET dphtbl          ;point to base of table
2623 391D                 find1: cmp     word ptr [di],bx          ;is this the correct DPH?
2625 7414          263B          je      gotit                     ;yes
2627 833D00                      cmp     word ptr [di],0000        ;empty slot?
262A 740D          2639          je      empty                     ;yes
262C 83C704                      add     di,4                      ;point to next entry
262F E2F2          2623          loop    find1
2631 BB9227                      mov     bx,OFFSET nodph           ;send error message
2634 E8A1FF        25D8          call    pmsg
2637 FA                          CLI                               ;die
2638 F4                          HLT
2639 891D                 empty: mov     word ptr [di],bx          ;set up a new entry
263B 47                   gotit: inc     di                        ;point to DPH pointer
263C 47                          inc     di
263D 8B3D                        mov     di,word ptr [di]          ;return a DPH pointer
263F 59                          pop     cx
2640 C3                          ret

                          ; Seek to track 0 on the current disk
                          ; Input: None
                          ; Output: None

                          HOME:
                                                          ;ask the 8085 to home the disk
2641 C60600CC18                  mov     byte ptr .CODE,18H
2646 E96700        26B0          jmp     callcb

                          ; Seek to specified track on current disk
                          ; Input: CX = requested track
                          ; Output: None

                          SETTRK:
                                                          ;ask the 8085 to seek
2649 C60600CC1E                  mov     byte ptr .CODE,1EH
264E E95F00        26B0          jmp     callcb

                          ; Select sector on current disk
                          ; Input: CX = requested sector
                          ; Output: None

                          SETSEC:
                                                          ;ask the 8085 to select a sector
2651 C60600CC21                  mov     byte ptr .CODE,21H
2656 E95700        26B0          jmp     callcb

                          ; Translate a logical sector to a physical sector
                          ; Input: CX = logical sector
                          ;         DX points to translate table
                          ; Output: BX = physical sector

                          SECTRAN:
                                                          ;ask the 8085 for a translation
2659 C60600CC30                  mov     byte ptr .CODE,30H
265E 81C20004                    add     dx,cpm_offset  ;correct for seg reg bias
                                                          ; before calling 8085 CBIOS
2662 E94B00        26B0          jmp     callcb
```

```
                               ; Set the memory offset to be used on next disk I/O
                               ; Input: CX = requested offset
                               ; Output: None

                               SETDMA:
2665 890EED26                          mov     dma_adr,cx
2669 C3                                ret

                               ; Set the memory base to be used on next disk I/O
                               ; Input: CX = requested base address
                               ; Output: None

                               SETDMAB:
266A 890EEF26                          mov     dma_seg,cx
266E C3                                ret

                               ; Return the address of the memory segment table.
                               ; Input: None
                               ; Output: BX points to the memory segment table

                               GETSEGT:
266F BBAB27                            mov     bx,offset seg_table
2672 C3                                ret

                               ; Read a sector from the disk.  Use the currently selected
                               ; disk, the currently requested track, the currently
                               ; requested sector, and place the data read at the
                               ; current DMA address.
                               ; Input: None
                               ; Output: AL = 0 -> no errors occurred
                               ;         AL = 1 -> non-recoverable error occurred

                               READ:
                                                               ;ask the 8085 for a 128 byte sector
2673 C60600CC27                        mov     byte ptr .CODE,27H
2678 E83500          26B0              call    callcb
267B BE80CC                            mov     si,DSKBUF       ;copy from task block to dma address
267E 06                                push    es
267F 8E06EF26                          mov     es,dma_seg      ;a block move is nice and quick
2683 8B3EED26                          mov     di,dma_adr
2687 B94000                            mov     cx,64           ;copy 128 bytes (64 words)
268A FC                                cld
268B F3A5                              rep     movs ax,ax
268D 07                                pop     es
268E 0AC0                              or      al,al           ;set status for CP/M-86
2690 C3                                ret

                               ; Write a sector to the disk.  Use the currently selected
                               ; disk, the currently requested track, the currently
                               ; requested sector, and place the data read at the
                               ; current DMA address.
                               ; Input: None
                               ; Output: AL = 0 -> no errors occurred
                               ;         AL = 1 -> non-recoverable error occurred

                               WRITE:
                                                               ;ask the 8085 to write 128 bytes
2691 C60600CC2A                        mov     byte ptr .CODE,2AH
2696 BF80CC                            mov     di,DSKBUF       ;copy from dma address to task block
2699 8B36ED26                          mov     si,dma_adr
269D 1E                                push    ds              ;set up for string move
269E 06                                push    es
269F 1E                                push    ds
26A0 8E1EEF26                          mov     ds,dma_seg
26A4 07                                pop     es
```

```
26A5 B94000                 mov     cx,64           ;copy 128 bytes (64 words)
26A8 FC                     cld
26A9 F3A5                   rep     movs ax,ax      ;string moves are nice and quick
26AB 07                     pop     es
26AC 1F                     pop     ds
26AD E90000        26B0     jmp     callcb          ;let the 8085 write the data

                    ; Call the 8085 BIOS using the interface area.
                    ; The interface area is set up with all the desired values
                    ; that the 8085 registers are to assume when the 8085 wakes
                    ; up.  The one-to-one mapping provided by Digital Research
                    ; between 8085 BIOS registers and 8088 BIOS registers allows
                    ; this routine to be common for all BIOS function calls.

                    CALLCB:
                                                    ;set the 8085 pseudo-registers
26B0 880601CC               mov     byte ptr .AREG,al
26B4 882E03CC               mov     byte ptr .BREG,ch
26B8 880E02CC               mov     byte ptr .CREG,cl
26BC 883605CC               mov     byte ptr .DREG,dh
26C0 881604CC               mov     byte ptr .EREG,dl
26C4 883E07CC               mov     byte ptr .HREG,bh
26C8 881E06CC               mov     byte ptr .LREG,bl

                            IF      SAVLOWMEM       ;check debugging equate
                            call    savelow86       ;save 8088 environment
                            call    restorelow      ;restore 8085 environment
                            ENDIF

26CC E4FD                   IN      AL,SWAP         ;swap processors so the 8085 does
                                                    ; the dirty work

                            IF      SAVLOWMEM       ;check debugging equate
                            call    savelow         ;save 8085 environment
                            call    restorelow86    ;restore 8088 environment
                            ENDIF
                                                    ;Rebuild the 8088 registers based
                                                    ; on the information returned by
                                                    ; the 8085 in the task block
26CE 8A0601CC               mov     al,byte ptr .AREG
26D2 8A2E03CC               mov     ch,byte ptr .BREG
26D6 8A0E02CC               mov     cl,byte ptr .CREG
26DA 8A3605CC               mov     dh,byte ptr .DREG
26DE 8A1604CC               mov     dl,byte ptr .EREG
26E2 8A3E07CC               mov     bh,byte ptr .HREG
26E6 8A1E06CC               mov     bl,byte ptr .LREG
26EA 0AC0                   or      al,al           ;set status for CP/M-86
26EC C3                     ret

                    ; These routines are only present if saving and restoring
                    ; low memory is necessary.  Saving and restoring memory
                    ; is useful in debugging the initial installation of these
                    ; programs if the 8085 BIOS makes use of any low memory
                    ; areas.

                            IF      SAVLOWMEM

                    ; save 8085 low memory

                    savelow:
                                                    ;point to 8085 save area
                            mov     di,offset savarea
                            jmps    saveit
```

```
; save 8088 low memory

savelow86:
                                ;point to 8088 save area
        mov     di,offset savarea86

; Save 256 bytes of memory
; Input: DI points to storage area where low memory can be stored
; Output: None

saveit:
        push    ds
        push    es
        push    ds
        pop     es
        mov     si,0            ;Start at absolute zero
        mov     ds,si
        cld
        mov     cx,128          ;save 256 bytes (128 words) of low mem
        rep     movs ax,ax      ;a string move is quick and easy
        pop     es
        pop     ds
        ret


; restore 8085 low memory

restorelow:
                                ;point to 8085 save area
        mov     si,offset savarea
        jmps    restoreit

; restore 8088 low memory

restorelow86:
                                ;point to 8088 save area
        mov     si,offset savarea86

; restore 256 bytes of low memory
; Input: SI points to memory area containing saved bytes
; Output: None

restoreit:
        push    es
        mov     di,0            ;restore absolute 0
        mov     es,di
        cld
        mov     cx,128          ;restore 256 bytes (128 words) low mem
        rep     movs ax,ax      ;string it along
        pop     es
        ret
        ENDIF

; end of debugging code


; Data Areas

26ED                    data_offset     equ offset $

        dseg
        org     data_offset     ;contiguous with code segment
```

```
                            ; storage for the current dma address

26ED 0000                   dma_adr dw      0               ;dma offset (from DS)
26EF 0000                   dma_seg dw      0               ;dma segment base

                            ; local copies of the disk parameter headers
                            ; storage is allocated for four disk drives - may be altered
                            ; for different system configurations

26F1 0000                   dph0    dw      0       ;local copy of Disk Parameter Header Block 0
26F3 000000000000                   dw      0,0,0
26F9 B027                           dw      dirbf
26FB 0000                           dw      0
26FD 5C29                           dw      csv0
26FF 3028                           dw      alv0

2701 0000                   dph1    dw      0       ;local copy of Disk Parameter Header Block 1
2703 000000000000                   dw      0,0,0
2709 B027                           dw      dirbf
270B 0000                           dw      0
270D 9C29                           dw      csv1
270F 7B28                           dw      alv1

2711 0000                   dph2    dw      0       ;local copy of Disk Parameter Header Block 2
2713 000000000000                   dw      0,0,0
2719 B027                           dw      dirbf
271B 0000                           dw      0
271D DC29                           dw      csv2
271F C628                           dw      alv2

2721 0000                   dph3    dw      0       ;local copy of Disk Parameter Header Block 3
2723 000000000000                   dw      0,0,0
2729 B027                           dw      dirbf
272B 0000                           dw      0
272D 1C2A                           dw      csv3
272F 1129                           dw      alv3

                            ; Table to remember which dph is which.
                            ; This table is used by the findph routine

2731 0000                   dphtbl  dw      0
2733 F126                           dw      OFFSET dph0
2735 0000                           dw      0
2737 0127                           dw      OFFSET dph1
2739 0000                           dw      0
273B 1127                           dw      OFFSET dph2
273D 0000                           dw      0
273F 2127                           dw      OFFSET dph3

                            ; Signon message printed at powerup
                            ; May be changed to anything desired

2741 0D0A0D0A               signon  db      cr,lf,cr,lf
2745 43502F4D2D38                   db      'CP/M-86 Version 1.1',cr,lf
     362056657273
     696F6E20312E
     310D0A
275A 0D0A0D0A                       db      cr,lf,cr,lf
275E 53797374656D                   db      'System Generated 10/07/82'
     2047656E6572
     617465642031
     302F30372F38
     32
2777 0D0A00                         db      cr,lf,0
```

```
                           ; Interrupt trap fatal error message
                           ; May be changed to anything desired

277A 0D0A                  int_trp db      cr,lf
277C 496E74657272                  db      'Interrupt Trap Halt'
     757074205472
     61702048616C
     74
278F 0D0A                          db      cr,lf
2791 00                            db      00

                           ; Internal failure due to lack of DPH space message
                           ; May be changed to anything desired

2792 0D0A                  nodph   db      cr,lf
2794 4E6F20726F6F                  db      'No room in DPH table'
     6D20696E2044
     504820746162
     6C65
27A8 0D0A                          db      cr,lf
27AA 00                            db      00

                           ; System Memory Segment Table

27AB 01                    segtable db     1               ;1 segment
27AC EA02                          dw      tpa_seg         ;1st seg starts after BIOS
27AE 160A                          dw      tpa_len         ; and extends to IFAREA


                           ; Miscellaneous uninitialized data areas.

27B0                       dirbf   rs      128             ;Used by local dph copies
2830                       alv0    rs      75              ;Used by local dph copies
287B                       alv1    rs      75              ;Used by local dph copies
28C6                       alv2    rs      75              ;Used by local dph copies
2911                       alv3    rs      75              ;Used by local dph copies
295C                       csv0    rs      64              ;Used by local dph copies
299C                       csv1    rs      64              ;Used by local dph copies
29DC                       csv2    rs      64              ;Used by local dph copies
2A1C                       csv3    rs      64              ;Used by local dph copies

                                   IF      SAVLOWMEM       ;Debugging memory save areas
                           savarea rs      256
                           savarea86 rs    256
                                   ENDIF

2A5C                       loc_stk rw      32              ;local stack for initialization
  2A9C                     stkbase equ     offset $

  2A9C                     lastoff equ     offset $

                           ; The following two equates are used to determine the amount
                           ; of free memory usable by CP/M-86 in the first 64k.  This
                           ; information is used to build the first entry in the memory
                           ; segment table.  If you have more than 64k of RAM in your
                           ; computer add more entries to the memory segment table.

                           ; Calculate the paragraph address of the first paragraph following
                           ; the end of this BIOS.
  02EA                     tpa_seg equ     (lastoff+cpm_offset+15) / 16
```

```
                          ; Calculate the number of paragraphs free between the end of this
                          ; BIOS and the beginning of the task block.
  0A16                    tpa_len equ     (IFAREA+cpm_offset)/16 - tpa_seg

 2A9C 00                          db 0                    ;fill last address for GENCMD

                          ; Dummy data section for establishing interrupt vectors

  0000                            dseg    0       ;absolute low memory
                                  org     0       ;point to interrupt vector 0
 0000                     int0_offset     rw      1
 0002                     int0_segment    rw      1
                          ;       pad to the BDOS interrupt vector (INT 224)
 0004                             rw      2*(bdos_int-1)

 0380                     bdos_offset     rw      1
 0382                     bdos_segment    rw      1
                                  END

END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR: 12%
```

```
                        ; LISTING 2 - BOOT86.ASM
                        ; THIS PROGRAM COPYRIGHT 1983 BY HOWARD SPINDEL.
                        ; PERMISSION GRANTED TO REPRODUCE THIS PROGRAM
                        ; FOR INDIVIDUAL NON-PROFIT USE ONLY.

                        ; This program has four tasks to accomplish.  It is
                        ; responsible for loading a CP/M-86 system file into
                        ; memory.  It provides for an 8088 reset or restart
                        ; vector.  It saves the CP/M low core environment in
                        ; the BIOS task block area.  A portion of this program
                        ; is relocated into high memory where it remains
                        ; permanently resident while CP/M-86 is running so it
                        ; can function as the interface between the BIOS task
                        ; block requests and the CP/M BIOS, thereby providing
                        ; BIOS services for CP/M-86.  All of the code in this
                        ; program runs on the 8085.

                        ; Specify the user changeable equates near the front
                        ; of the listing so they are easily found.

D000 =                  IFAREA  EQU     0D000H                  ;address of the task block
00FD =                  SWAP    EQU     0FDH                    ;CPU swap port

                        ; Miscellaneous Equates
0005 =                  BDOS    EQU     5                       ;BDOS jump address
005C =                  FCB     EQU     5CH                     ;Address of file control
                                                                ; block at program invocation
0003 =                  IOBYTE  EQU     3                       ;Address of CP/M iobyte
0004 =                  CDISK   EQU     4                       ;Address of CP/M current disk
D100 =                  FWAREA  EQU     IFAREA+100H             ;Address where BIOS task block
                                                                ; processor code is relocated

0100                            ORG     100H

                        ; This section of code attempts to open the CP/M-86 system
                        ; file specified by the user.  If the open is successful, the
                        ; CP/M-86 system file is read into memory at address 400 (hex).
                        ; Remember that all CP/M-86 executable files contain an extra
                        ; 128 bytes on the front (used for segment register initialization
                        ; information).  Therefore, if the CP/M-86 system file is read
                        ; into memory starting at address 380 (hex) the first executable
                        ; code will wind up at 400 (hex) as desired.

0100 310A02                     LXI     SP,STKBAS1              ;Set up a stack
0103 AF                         XRA     A                       ;Clean up the default FCB
0104 326800                     STA     FCB+12
0107 326A00                     STA     FCB+14
010A 327C00                     STA     FCB+32
```

```
010D 0E0F               MVI     C,0FH                ;Open user specified file
010F 115C00             LXI     D,FCB
0112 CD0500             CALL    BDOS
0115 FEFF               CPI     0FFH                 ;File opened successfully?
0117 C23601             JNZ     OK                   ;Jump if yes

011A 0E09               MVI     C,9                  ;File open failed
011C 112501             LXI     D,EMSG               ;Use BDOS call to print
011F CD0500             CALL    BDOS                 ; failure message
0122 C30000             JMP     0                    ;Warm boot to abort

0125 46494C4520EMSG     DB      'FILE NOT FOUND',13,10,'$'

0136 218003     OK:     LXI     H,380H               ;Start reading file at 380 (hex)
                                                     ; to discard first sector
0139 E5         LOOP:   PUSH    H                    ;Save current memory address
013A 0E1A               MVI     C,1AH                ;Set BDOS dma address to
013C EB                 XCHG                         ; current memory address
013D CD0500             CALL    BDOS

0140 0E14               MVI     C,14H                ;Read a sector of the CP/M-86
0142 115C00             LXI     D,FCB                ; system file at the current
0145 CD0500             CALL    BDOS                 ; memory address
0148 E1                 POP     H                    ;Restore current memory address
0149 B7                 ORA     A                    ;Did we hit end of file on read?
014A C25401             JNZ     EOF                  ;Jump if yes

014D 118000             LXI     D,128                ;Not end of file, so add 128 to
0150 19                 DAD     D                    ; current memory address and
0151 C33901             JMP     LOOP                 ; go back to read some more

0154 0E0D       EOF:    MVI     C,0DH                ;CP/M-86 all read in, so reset
0156 CD0500             CALL    BDOS                 ; disks
0159 0E1A               MVI     C,1AH                ;Set the dma address to the disk
015B 1180D0             LXI     D,DSKBUF             ; buffer in the task block
015E CD0500             CALL    BDOS

                ; This section of code saves the memory at FFF0 in the BIOS task block
                ; so that it can be used for an 8088 restart vector.  This is not
                ; useful if a ROM exists at FFF0.

0161 110CD0             LXI     D,SFFF0              ;Point to the task block
0164 21F0FF             LXI     H,0FFF0H             ;Point to FFF0
0167 0E05               MVI     C,5                  ;Save 5 bytes for an 8088
                                                     ; long jump
0169 7E         LOOP2:  MOV     A,M                  ;Fetch a byte from FFFx
016A 12                 STAX    D                    ;Store it in the task block
016B 23                 INX     H                    ;Increment the pointers
016C 13                 INX     D
016D 0D                 DCR     C                    ;5 bytes done?
016E C26901             JNZ     LOOP2                ;Jump if no

                ; This section of code saves the CP/M iobyte and cdisk fields from
                ; low memory in the BIOS task block.  The low memory area is changed
                ; when CP/M-86 runs, and must be rebuilt every time the CP/M BIOS
                ; is called.  A pointer to the CP/M BIOS is also calculated and
                ; saved in the task block for future use.

0171 3A0300             LDA     IOBYTE               ;Get the iobyte
0174 3208D0             STA     IOBYTE2              ;Store iobyte in task block
0177 3A0400             LDA     CDISK                ;Get the current disk
017A 3209D0             STA     CDISK2               ;Store cdisk in task block

017D 2A0100             LHLD    01                   ;Get a pointer to the BIOS
                                                     ; warm boot
```

```
0180 2B                 DCX    H                    ;Warm boot is second vector,
0181 2B                 DCX    H                    ; so three decrements points
0182 2B                 DCX    H                    ; to beginning of BIOS
0183 220AD0             SHLD   CBIOS                ;Save pointer in task block

               ; This section of code provides for 8088 reset and restart vectors.
               ; FFF0 is set up in case it is RAM.  A reset vector is stored at
               ; address 0.  A restart vector is stored at address 10.  In all
               ; cases, the 8088 is to begin execution by jumping to the first
               ; instruction of the CP/M-86 BIOS (the cold boot entry point at
               ; address 40:2500).

0186 21F0FF             LXI    H,0FFF0H             ;Jump goes at FFF0
0189 110000             LXI    D,0                  ;Jump goes at 0
018C 011000             LXI    B,10H                ;Jump goes at 10 too

018F 3EEA               MVI    A,0EAH               ;Op code for 8088 long jump
0191 77                 MOV    M,A                  ;Store opcode at all three
0192 12                 STAX   D                    ; places
0193 02                 STAX   B
0194 23                 INX    H                    ;Increment all three pointers
0195 13                 INX    D
0196 03                 INX    B
0197 3E00               MVI    A,00                 ;Store low byte of 8088 offset
0199 77                 MOV    M,A                  ; at all three places
019A 12                 STAX   D
019B 02                 STAX   B
019C 23                 INX    H                    ;Increment all three pointers
019D 13                 INX    D
019E 03                 INX    B
019F 3E25               MVI    A,25H                ;Store high byte of 8088 offset
01A1 77                 MOV    M,A                  ; at all three places
01A2 12                 STAX   D
01A3 02                 STAX   B
01A4 23                 INX    H                    ;Increment all three pointers
01A5 13                 INX    D
01A6 03                 INX    B
01A7 3E40               MVI    A,40H                ;Store low byte of 8088 segment
01A9 77                 MOV    M,A                  ; at all three places
01AA 12                 STAX   D
01AB 02                 STAX   B
01AC 23                 INX    H                    ;Increment all three pointers
01AD 13                 INX    D
01AE 03                 INX    B
01AF 3E00               MVI    A,0                  ;Store high byte of 8088 segment
01B1 77                 MOV    M,A                  ; at all three places
01B2 12                 STAX   D
01B3 02                 STAX   B

               ; This section of code relocates the permanently resident portion
               ; of this program to high memory where it will remain and function
               ; as the interface between the BIOS task blocks created by CP/M-86
               ; and the CP/M BIOS.

01B4 2100D1             LXI    H,FWAREA             ;Point where code is going
01B7 110060             LXI    D,HIMEM              ;Point where code is now
01BA 0E80               MVI    C,TOP-HIMEM          ;Get number of bytes to move
01BC 1A        LOOPR:   LDAX   D                    ;Fetch a byte of code
01BD 77                 MOV    M,A                  ;Store it in high memory
01BE 13                 INX    D                    ;Increment both pointers
01BF 23                 INX    H
01C0 0D                 DCR    C                    ;Done moving code?
01C1 C2BC01             JNZ    LOOPR                ;Jump if no
```

```
01C4 317FD0             LXI     SP,STKBAS             ;Reset the stack to the stack
                                                      ; area allocated in the task
                                                      ; block
01C7 C300D1             JMP     FWAREA                ;Start executing the task
                                                      ; block processor

01CA           STACK    DS      64                    ;Allocate space for initial
020A =         STKBAS1 EQU      $                     ; stack

               ; Define the BIOS task block area.
               ; These definitions must match the definitions in BOOT80.A86
               ; and CBIOS86.A86.
               ; Note that these register saving areas are in a very specific
               ; order.  They must be stored low byte first because sixteen bit
               ; loads and stores are done.
               ; DO NOT CHANGE THE ORDER OF THE REGISTER SAVING AREAS.

D000 =         CODE     EQU     IFAREA                ;1 BYTE: BIOS task code
D001 =         AREG     EQU     IFAREA+1              ;1 BYTE: 8085 A register
D002 =         CREG     EQU     IFAREA+2              ;1 BYTE: 8085 C register
D003 =         BREG     EQU     IFAREA+3              ;1 BYTE: 8085 B register
D004 =         EREG     EQU     IFAREA+4              ;1 BYTE: 8085 E register
D005 =         DREG     EQU     IFAREA+5              ;1 BYTE: 8085 D register
D006 =         LREG     EQU     IFAREA+6              ;1 BYTE: 8085 L register
D007 =         HREG     EQU     IFAREA+7              ;1 BYTE: 8085 H register

D008 =         IOBYTE2 EQU      IFAREA+8              ;1 BYTE: iobyte save area
D009 =         CDISK2   EQU     IFAREA+9              ;1 BYTE: cdisk save area
D00A =         CBIOS    EQU     IFAREA+0AH            ;2 BYTES: pointer to BIOS
D00C =         SFFF0    EQU     IFAREA+0CH            ;5 BYTES: FFF0 save area
D07F =         STKBAS   EQU     IFAREA+07FH           ;Leave enough room for a stack

D080 =         DSKBUF   EQU     IFAREA+080H           ;128 BYTES: disk dma address

               ; End of task block interface definition.

               ; This portion of the code is relocated to high memory where it
               ; will remain resident while CP/M-86 is running to act as an
               ; interface between CP/M-86 task blocks and the CP/M BIOS.  All
               ; of the jump instructions in this area look funny because of
               ; the relocation.

               ; Origin this code to an arbitrary place which is expected to
               ; be higher than the end of the CP/M-86 BIOS, lower than the
               ; BIOS task block area, and low enough so that this program loads
               ; reasonably quickly under CP/M.
               ; This code is relocated because it overlays a portion of the
               ; CP/M BDOS.

6000                    ORG     06000H
6000 DBFD      HIMEM:   IN      SWAP                  ;Let the 8088 run

               ; Restore the memory at FFF0 from the bytes saved in the BIOS task
               ; block.  This is not useful if FFF0 is ROM.

6002 210CD0             LXI     H,SFFF0               ;Point to save area in task block
6005 11F0FF             LXI     D,0FFF0H              ;Point to FFF0
6008 0E05               MVI     C,5                   ;Restore 5 bytes
600A 7E        LOOP3:   MOV     A,M                   ;Fetch byte from task block
600B 12                 STAX    D                     ;Store at FFFx
600C 23                 INX     H                     ;Increment both pointers
600D 13                 INX     D
600E 0D                 DCR     C                     ;Done with 5 bytes?
600F C20AD1             JNZ     FWAREA+(LOOP3-HIMEM)  ;Jump if no
```

```
               ; The remainder of the code in this program is a loop which processes
               ; BIOS task blocks as requested by the 8088.

6012 3A0300    NXTCMD: LDA      IOBYTE                  ;Save the current CP/M-86 low
6015 F5                PUSH     PSW                     ; core environment on the stack
6016 3A0400            LDA      CDISK
6019 F5                PUSH     PSW
601A 3A08D0            LDA      IOBYTE2                 ;Restore the CP/M low core
601D 320300            STA      IOBYTE                  ; environment from bytes saved
6020 3A09D0            LDA      CDISK2                  ; in the task block
6023 320400            STA      CDISK

6026 2A0AD0            LHLD     CBIOS                   ;Point to the CP/M BIOS start
6029 3A00D0            LDA      CODE                    ;Get task block code
602C FEFF              CPI      OFFH                    ;Time to go back to CP/M?
602E C23FD1            JNZ      FWAREA+(STAY86-HIMEM)   ;Jump if not
6031 23                INX      H                       ;Three increments to the BIOS
6032 23                INX      H                       ; start makes a warm boot
6033 23                INX      H                       ; pointer
6034 220100            SHLD     01                      ;Put warm boot vector back at 1
6037 3EC3              MVI      A,(JMP)                 ;Put 8085 jump opcode back at 0
6039 320000            STA      00
603C C30000            JMP      00                      ;Warm boot CP/M

603F 5F        STAY86: MOV      E,A                     ;Add task block code to base of
6040 1600              MVI      D,0                     ; BIOS creating a pointer to the
6042 19                DAD      D                       ; requested routine
6043 1158D1            LXI      D,FWAREA+(RETPT-HIMEM)  ;Push a return address
6046 D5                PUSH     D
6047 E5                PUSH     H                       ;Push pointer to requested routine

6048 2A02D0            LHLD     CREG                    ;Load up the 8085 registers from
604B 44                MOV      B,H                     ; the values stored in the task
604C 4D                MOV      C,L                     ; block
604D 2A04D0            LHLD     EREG
6050 EB                XCHG
6051 2A06D0            LHLD     LREG
6054 3A01D0            LDA      AREG
6057 C9                RET                              ;This is really a funny call to
                                                        ; the requested BIOS routine

               ; When the CP/M BIOS is done with the current request, it will return
               ; here.

6058 3201D0    RETPT:  STA      AREG                    ;Save the current value of all
605B 2206D0            SHLD     LREG                    ; 8085 registers in the
605E EB                XCHG                             ; appropriate task block place
605F 2204D0            SHLD     EREG
6062 60                MOV      H,B
6063 69                MOV      L,C
6064 2202D0            SHLD     CREG

               ; The BIOS might have updated the low core variables, so better save
               ; new copies in the task block.

6067 3A0400            LDA      CDISK                   ;Save possible new CDISK
606A 3209D0            STA      CDISK2
606D 3A0300            LDA      IOBYTE                  ;Save possible new IOBYTE
6070 3208D0            STA      IOBYTE2
6073 F1                POP      PSW                     ;Restore the CP/M-86 low core
6074 320400            STA      CDISK                   ; environment from the values
6077 F1                POP      PSW                     ; save earlier on the stack
6078 320300            STA      IOBYTE
```

```
607B DBFD                IN     SWAP                    ;Let the 8088 run again
607D C312D1              JMP    FWAREA+(NXTCMD-HIMEM)   ;Jump back to decode next task
                                                        ; block request

               TOP:                                     ;End of relocated code

6080           END
```

```
                        ; LISTING 3 - BOOT80.A86
                        ; THIS PROGRAM COPYRIGHT 1983 BY HOWARD SPINDEL.
                        ; PERMISSION GRANTED TO REPRODUCE THIS PROGRAM
                        ; FOR INDIVIDUAL NON-PROFIT USE ONLY.

                        ; This program warm boots CP/M from CP/M-86.
                        ; A BIOS task block is built using the special request
                        ; code of FF (hex) which signifies to the BIOS task
                        ; block processor that a warm boot of CP/M is desired.
                        ; Since this is the last program that the 8088 will be
                        ; running until CP/M-86 is rebooted (or a compatible
                        ; 8088 application is rebooted) this program is
                        ; required to leave the 8088 executing at absolute
                        ; address 10 (hex).  The 8085 can then provide for
                        ; an 8088 restart vector by writing an 8088 jump
                        ; instruction at address 10 before turning the 8088
                        ; on.

                        ; Specify the user changeable equates near the front of
                        ; the listing so they are easily found.

 D000                   IFAREA  EQU     0D000H                  ;address of the task block
 00FD                   SWAP    EQU     0FDH                    ;CPU swap port

                        ; Definition of relevant portions of the BIOS task block area.
                        ; This definition must match the definitions present
                        ; in the CBIOS86 and the BOOT86 files.

 D000                   CODE    EQU     IFAREA+0
 D009                   CDISK   EQU     IFAREA+9

                        ; Set up equates for the addresses in low memory
                        ; where the swap instruction will be built.

 0005                   opcode  EQU     5
 0006                   operand EQU     6
 0007                   noparea EQU     7

 0100                   tpa     EQU     100H

                                cseg
                                org     tpa

                        ; Do a little cleanup of the current environment.

0100 B119                       mov     cl,19H                  ;get current disk
0102 CDE0                       int     224
0104 50                         push    ax                      ;save current disk
0105 B10D                       mov     cl,0DH                  ;reset disk system
0107 CDE0                       int     224                     ; to make sure all is clean
0109 58                         pop     ax                      ;restore current disk
```

```
                         ; Set up the ds segment register so that memory can
                         ; be accessed from absolute zero.

010A FA                            CLI
010B B90000                        mov     cx,0000                 ;must get absolute 0 relative
010E 8ED9                          mov     ds,cx

                         ; Set up the required fields in the BIOS task block.

0110 880609D0                      mov     byte ptr .CDISK,al      ;save the current disk in
                                                                   ; the task block area so
                                                                   ; it is preserved across swap
0114 C60600D0FF                    mov     byte ptr .CODE,0FFH     ;special reboot flag

                         ; Build the swap instruction at absolute location 5 and
                         ; follow it with nine no ops to leave the 8088 executing
                         ; at address 10 (hex).  Nine no ops seems to be enough to
                         ; guarantee that the 8088 fetch ahead queue is full.

0119 C6060500E4                    mov     byte ptr .opcode,0E4H   ;put in al,SWAP instruct. at 5
011E C6060600FD                    mov     byte ptr .operand,SWAP
0123 B90900                        mov     cx,9                    ;put in 9 no ops to take care
0126 BE0700                        mov     si,noparea              ; of 8088 fetch ahead queue
0129 C60490              loopr:    mov     byte ptr [si],90H
012C 46                            inc     si
012D E2FA           0129           loop    loopr

                         ; Now do a long jump to absolute address 5 to execute the
                         ; swap instruction and the no ops.  The assembler doesn't
                         ; seem to want to build the instruction for us so we will
                         ; just do it with some db's.

012F EA                            db      0EAH                    ;kludge long jump to 0:5
0130 0500                          dw      opcode                  ; offset
0132 0000                          dw      0000H                   ; segment

                         ; we will never return, so program ends here!

                         END


END OF ASSEMBLY.  NUMBER OF ERRORS:   0.  USE FACTOR:  1%
```

ABOUT THE AUTHOR

Howard Spindel is a Senior Software Designer with Corporate Data Sciences, Inc. located in Santa Clara, California. Prior employment has included positions at Concept Technologies, Inc., Tektronix, Inc. and Burroughs Corp. He received a B.A. in Computer Science from the University of California at Berkeley in 1975. His special areas of interest are operating systems, with emphasis on real time multitasking executives for microprocessors, and graphics for microprocessor-based systems.

(continued from page 19)
8085 registers are loaded from the information that the 8088 left in the task block area. Then, based on the task block CODE, the correct routine in the CP/M BIOS is called. Since the 8085 instruction set does not provide an indirect call instruction, the BIOS call is performed with a common 8085 trick. The return address is pushed on the stack, followed by the address which is to be called. A return instruction is then executed which actually calls the intended routine. The return instruction at the end of the called routine will pull the return address which was pushed on the stack.

After the CP/M BIOS returns to the task block processor, the 8085 registers are stored in the task block area. The possibly updated CP/M low-memory environment is saved again in the task block area. The CP/M-86 environment is rebuilt in low memory. Then the 8088 is allowed control again so it can use the information which the 8085 built in the task block area. When the 8085 next wakes up, it will begin processing by jumping back to NXTCMD.

The CP/M Reboot Program

Listing three (BOOT80.A86) is a program that runs on the 8088. It will cause CP/M-86 to terminate and CP/M to be warm booted. This program is quite short, but contains an interesting trick. A task block is built (in the task block area) which uses the task code FF to request that the 8085 task block processor warm boot CP/M. The trick is that this program must leave the 8088 executing at a known address so that, if CP/M-86 is restarted, the 8085 can write a restart vector into memory. As mentioned in the text box of page 28, I have used the convention that the 8088 will always start executing again at absolute address 10 (hex). It seems obvious that the way to leave the 8088 executing at address 10 is to put at address 0E a 2-byte IN instruction which swaps processors back to the 8085. However, the 8088 hardware has a feature which is called a *fetch ahead queue* (also called instruction pipeline) for its instructions. This feature allows the 8088 to look ahead in memory and ready some future instructions for execution, during the execution of another instruction. This means that, if an IN instruction were executed at address 0E, the 8088 would already have decoded the instruction at location 10. The 8085 will want to write a different instruction at location 10, but can't because the 8088 won't look at it. The way to solve this problem is to place several NOP instructions between the IN instruction and location 10. That way, the *fetch ahead queue* fills up with the NOP instructions, and the 8088 does not decode the instruction at location 10 until the 8085 has had a chance to write something there. This was one of the trickier problems to figure out when debugging these programs.

SYSTEM REQUIREMENTS FOR INSTALLING THESE PROGRAMS

• *CP/M 2.2 computer system.*
You must have CP/M 2.2 working in your computer using the CompuPro CPU 8085/88 (or Macrotech MI-286 — see the text box on page 13) as a

processor board. The BIOS code must not use any interrupt-driven code because the 8085 will not be available to handle interrupts when the 8088 is in control.

- *CP/M-86 1.1 distribution disk from Digital Research.*
- *CompuPro CPU 8085/88 (or Macro-tech MI-286) Dual-Processor board.*

The CompuPro CPU board must be strapped so that both the 8085 and 8088 reset-on-swap switches (labelled 5RS and 8RS on the circuit board) are OFF, and the extended address clear-on-reset switch (labelled XAC) should be ON.

- *Recommended minimum 48K CP/M System.*

You will rapidly discover that CP/M-86 programs tend to eat up core, and many programs probably will not run without additional memory boards (beyond 64K). If you plan to add memory past 64K, you may need to upgrade your current memory to respond to all 24 bits of the IEEE 696 (S-100) bus.

- *Provision for an 8088 Reset Vector.*

5 bytes of Global RAM Memory (memory which responds to CPU requests without checking the upper 12 S-100 address bits) at address FFF0, or, alternatively, an EPROM installed at address FFFF0 (or FFF0 if FFF0 is global memory) which contains a 5-byte 8088 instruction to execute a far jump to absolute address 0:0. This memory requirement is necessary to provide a restart vector for the 8088 on initial power-up (as discussed in the text box on page 28).

INSTALLATION INSTRUCTIONS

Installation of these programs (once you get them typed in!) is fairly simple. Usually, the only customization that will be necessary for your system is to determine where the task block will reside in memory.

Examine the three source files (BOOT86.ASM, CBIOS86.ASM, and BOOT80.ASM). Near the beginning of each of the files, there is an equate called IFAREA. In the listings, there is a constant 0D000H as part of the calculation of IFAREA. There may be other terms involved in the calculation, but the 0D000H term is what you will need to change in order to make these programs run on your system.

In each of the source files, the term 0D000H should be changed (using your favorite editor) to become the base of your CP/M BIOS minus at least 200 (hex). If you do not know the base of your BIOS, use the DDT program to list (L command) the code at 0 in your system. Take the address of the jump listed at 0, subtract 3, and you have the base of your BIOS. Now subtract 200 (hex), and edit the resulting number into the source files where it now has 0D000H in the IFAREA equate. Note that all three source files must be set exactly the same!

GLOSSARY

This article contains some of the jargon with which all computer articles seem afflicted. To help explain some buzzwords, here is a short glossary.

CP/M (also called **CP/M-80**)
Control Program for Microcomputers. This is a widely used operating system provided by Digital Research. It runs on 8080, Z80, and 8085 microprocessors.

CP/M-86
This is a version of the CP/M operating system which runs on the 8086 family of microprocessors (8086, 8088, 80186, 80286). Data files are stored in the same format as CP/M, allowing disk compatibility between the two operating systems.

CCP
Console Command Processor. This is the first of three parts of the CP/M family of operating systems. The CCP is responsible for processing keyboard input and generating the appropriate calls on the other parts of CP/M.

BDOS
Basic Disk Operating System. This is the second of three parts of the CP/M family of operating systems. The BDOS is primarily responsible for maintaining all of the disk structures (directories and files) and allowing an easy, structured access to the disks. The BDOS will also make appropriate calls to the BIOS.

BIOS
Basic Input Output System. This is the third of three parts of the CP/M family of operating systems. The BIOS contains all the drivers for any hardware devices in a CP/M system. All the machine dependent code in the CP/M operating system is isolated in the BIOS. When CP/M is ported (moved) to a new machine, only the BIOS needs to be rewritten.

DDT
Dynamic Debugging Tool. This is a program supplied with CP/M which allows users to examine memory and interactively debug programs.

STAT
Another program supplied with CP/M which allows the user to configure the current active devices, control some disk parameters, and generally report system status.

Warm Boot
When CP/M (the 8080 version only) is running, the CCP and BDOS may be destroyed by a running program in order to gain more useful

If your dual CPU board is set so that the processor swap port is not the standard value of 0FD (hex), you will also need to change the equate for SWAP found near the front of each source file. No other changes should be necessary to customize these programs for your system. You may want to change some of the messages (the logon banner is a good example) in CBIOS86.A86. If you have more than 64K of memory in your system, you will want to eventually change the *segtable* memory table entries in CBIOS86.A86. *Segtable* entries were briefly discussed above in the description of the CP/M-86 BIOS. The CP/M-86 Operating System Guide, provided by Digital Research with CP/M-86, contains complete information on how to change the *segtable* entries.

memory space for running the program. A special BIOS function, called the Warm Boot, may be called by the program to cause the CCP and BDOS to be reloaded into memory from the disk.

Warm Boot Vector
A special jump stored at address 0 in the 8080 version of CP/M which is used to activate the Warm Boot routine in the BIOS. A program wishing to cause a Warm Boot need only jump to address zero.

IOBYTE
A byte of memory which contains the current active device mappings for CP/M. There are four logical devices in CP/M, the console, the reader, the punch, and the list device. The IOBYTE allows each of the four logical devices to be mapped to one of four physical devices controlled by the BIOS. The STAT program is used to examine or change the setting of the IOBYTE.

CDISK
A byte of memory which contains the current default disk being accessed by CP/M.

DPH
Disk Parameter Header. A table of information about the disk which is stored in the BIOS and used by the BDOS. The disk parameter header contains pointers to a sector translation table, a DPB, and scratchpad areas for use by the BDOS.

DPB
Disk Parameter Block. Another table of information about the disk which is also stored in the BIOS and used by the BDOS. The DPB contains several fields which determine the storage capacity of the disk, how many directory entries the disk can contain (and therefore how many files), and some other information which allows disks of varying capacities and capabilities to be used by CP/M.

Sector Translation Table
A table which is used by CP/M to convert logical disk sector numbers into physical disk sector numbers. The physical sectors of a disk are usually numbered consecutively on each track. In order to minimize rotational delays when accessing disks, it is usually necessary to avoid accessing physical sectors consecutively. Consecutive logical sectors in a file (as maintained by the BDOS) are therefore rarely stored as consecutive physical sectors on the disk. Given a logical sector number, the sector translation table tells how to find the physical sector on the disk.

DMA Address
Disk Memory Address. The address of a 128-byte buffer which is used to contain one disk sector on any disk read or disk write operation.

Assuming that the address equates are now correctly set for your system, it is time to assemble the sources. BOOT86.ASM is compatible with the standard Digital Research 8080 assembler (ASM); CBIOS86.A86 and BOOT80.A86 are compatible with the 8086 assembler distributed on Digital Research's CP/M-86 distribution disks (ASM86). To assemble and link the set of programs, perform the following steps:

```
ASM BOOT86
LOAD BOOT86
ASM86 CBIOS86
PIP CPMX.H86=CPM.H86,CBIOS86.H86
GENCMD CPMX 8080 CODE[A40]
ASM86 BOOT80
GENCMD BOOT80 8080
```

CPM.H86, ASM86.COM, and GENCMD.COM will be found on the CP/M-86 distribution disk. ASM.COM and LOAD.COM will be found on the CP/M distribution disk. Note that the above entire sequence can be performed while running under CP/M because Digital Research thoughtfully provides both CP/M and CP/M-86 versions of ASM86 and GENCMD. It would be inconvenient, to say the least, to require running ASM86 and GENCMD under CP/M-86 to bring up CP/M-86. The filename CPMX in the above sequence is an arbitrary choice; you may call it anything you like. All of the other filenames are fixed.

Lastly, make sure that your system is providing for the restart vector needed by the 8088. Reread now the system requirements section and the text box of page 28 to determine how to provide for the restart vector. If you will be providing an EPROM with an 8088 far jump to 0:0, note that the codes to program into your EPROM (in hexadecimal) are EA,00,00,00,00.

Your system should now be completely configured to run CP/M-86.

OPERATING INSTRUCTIONS

Booting your CP/M-86 system is very simple with the provided programs. Under CP/M, execute the command:

PROVIDING FOR AN 8088 RESET OR RESTART VECTOR

When the 8085 turns on the 8088 (running the BOOT86 program), so the 8088 can start executing the cold boot code in the CP/M-86 BIOS, the 8085 does not know where in memory the 8088 will start executing. If this is the very first time the 8088 is running since a power up or front panel reset, then the BOOT86 program must provide for an 8088 reset vector. If the BOOT86 program has activated the 8088 since the last power up or front panel reset, then the BOOT86 program must provide for an 8088 restart vector. If some other program has activated the 8088 since the last power up or reset, then BOOT86 is really lost — the computer must be reset or power cycled before BOOT86 can work.

The Reset Vector

When an 8088 microprocessor executes its first instruction after a reset, it automatically begins execution at address FFFF0 (hex). This is designed into the 8088 microprocessor chip, and the Dual CPU board provides no way around it. At address FFFF0, the typical thing to find is a jump instruction to the beginning of whatever program the 8088 is to execute. Ideally, there should be some RAM at address FFFF0 and the 8085 should write a jump instruction into that RAM. Unfortunately, address FFFF0 is not within the 64K addressable space of an 8085, so the 8085 can't readily write into any RAM above address FFFF (hex). The CompuPro Dual CPU board does provide a simple bank-switch memory port. This allows the 8085 to latch the upper address bits so that the 8085 can be executing in any one (and only one at a time) of the many 64K banks defined by the S-100 bus. However, the bank-switch memory port does not help solve the problem. If the 8085 sets the bank switch so that it can address the 64K block beginning at F0000 (the block containing the 8088 reset vector), then whatever code the 8085 was executing disappears because it is in a different bank and the 8085 will begin executing meaningless instructions.

The BOOT86 program allows a solution to this problem in either one of two ways. At address 0 (this is a nice accessible location for the 8085) the 8085 will write a jump instruction to wherever it wants the 8088 to begin executing. Then you have to get the 8088 to jump from FFFF0 to 0. The straightforward way is to place a ROM at address FFFF0 containing an 8088 jump instruction. The hex codes for a jump to 0 are EA,00,00,00,00. The alternative way requires an older memory board that ignores the upper address bits of the S-100 bus. This type of board is frequently called a global memory board. A board like this must be placed so that it responds to address FFF0 (hex). Since it ignores the upper address bits, it will also respond to address 1FFF0, 2FFF0, etc., and most importantly FFFF0. In my system I have used this approach with the Disk Jockey 2D board from Morrow. If the global memory board has ROM at FFF0, program that ROM with an 8088 jump to 0. If the global memory board has RAM at FFF0, it will still work. As part of its initialization, the BOOT86 program attempts to write an 8088 reset jump at FFF0. Of course, BOOT86 saves and restores whatever was at FFF0 since that could easily have been an important area to the 8085 BIOS.

Note that if you use the global memory approach, you may have a problem when expanding your system beyond 64K of memory. Since the global memory will respond in every 64K block, other memory boards cannot occupy conflicting addresses in any other 64K blocks. In my case, the solution was easy; the Disk Jockey 2D RAM has a disable port. I added an instruction to the CP/M-86 BIOS to disable the DJ2D upon entry, and reenable it anytime the CP/M BIOS is called (this instruction is not in the listings because it is peculiar to my system only). You may have to consult a hardware engineer to discover the best solution for your system.

The Restart Vector

If the BOOT86 program was previously used and the 8088 had been previously running, the 8088 will not go to address FFFF0 to find its first instruction. Instead, it will start executing wherever it left off the last time it was swapped out. (The CompuPro Dual CPU board must be strapped so that each microprocessor begins executing exactly where it left off after a processor swap — see the installation instructions section of this article.) To handle microprocessor restarts in an orderly fashion, I established a convention that any program I write for the 8088 terminates by leaving the 8088 expecting to find its next instruction at address 10 (hex). Once again, this is a nice easy address for the 8085 to get at. To handle the microprocessor restart, the BOOT86 program also writes an 8088 jump instruction at address 10. This instruction causes the 8088 to start executing at the desired location.

```
BOOT86 CPMX.CMD
```

The CPMX.CMD file is the CP/M-86 system file which you generated using ASM86 and GENCMD. Note that since BOOT86 accepts a filename input, you may command BOOT86 to pick between multiple CP/M-86 versions on a single disk.

After some disk access time, you will see the CP/M-86 sign-on message, followed by a familiar-looking CCP prompt. You are now running CP/M-86. Pat yourself on the back, and play with CP/M-86 for a while.

To return to CP/M-80 at any time, execute (under CP/M-86) the following command:

```
BOOT80
```

This will warm boot your CP/M-80 system back into memory (make sure that a disk with your CP/M system on the system tracks is inserted in drive A:). You may use BOOT86 and BOOT80 to swap back and forth between operating systems as often as you wish. There is no need to reset your computer between operating system swaps.

These programs are set up in a way that preserves the currently logged disk across operating system swaps. This means that if your system is displaying a B> prompt under CP/M, it will also display B> after booting in CP/M-86. The logged disk is again preserved when using BOOT80 to return to CP/M.

The programs will also preserve IOBYTE changes across operating system swaps. If you use STAT to change the IOBYTE assignments while running CP/M-86, you will find that the CP/M IOBYTE also reflects the change you made.

If you are using another package besides CBIOS86 to run your 8088, it will be necessary to reboot (front panel reset) your computer if the other package has been executed since the last reset. This is necessary so that the 8088 will be in a known state at the start of BOOT86 execution. Most likely, it will also be necessary to reset your computer if you want to run another 8088 application after running these programs.

TIPS IF YOU HAVE PROBLEMS

This section is intended to give some additional guidelines in case these programs do not work immediately.

1. Have you ever executed code on your 8088 before? It is possible that your 8088 does not work.

Many CPU 8085/88 boards are shipped with the 8088 set up to run at 8 MHz, and this may be too fast for your memory boards or other system components. This could be your problem. It may be corrected by replacing the 8088 crystal on your processor board with a slower crystal (remember that the 8088 uses a crystal three times the desired operating frequency).

2. Make certain that the IFAREA equates in all of the source files are exactly the same!

3. Be sure the switches on your CPU 8085/88 are set the way that the above system-requirements section specifies.

4. These programs have to maintain a low-memory (0-100 hex) environment for CP/M. For almost all users, this means rebuilding the IOBYTE and CDISK areas after every processor swap between the 8085 and the 8088 (since CP/M is not running, it is not necessary to rebuild the BDOS jump or the BIOS warm-boot jump). If your BIOS for the 8085 is using any of the low memory areas, other than IOBYTE and CDISK, you will probably have problems. To verify if this is the case, edit your CBIOS86.A86 file to change the equate for SAVLOWMEM to "true." This will cause the CP/M-86 BIOS to save and restore all 256 bytes of the low-memory area. Now go through the installation steps again and try running the system. You will notice a substantial reduction in the operating speed of your system. But, if the problems go away, then they are caused by your 8085 BIOS using some of the low-memory areas.

The reduction in operating speed makes saving all 256 bytes an undesirable long-term solution. The best solution is to rewrite the 8085 BIOS, so that it does not use any low-memory areas except the IOBYTE and CDISK. Another possible solution is, first, to determine the addresses that need saving and restoring. Second, edit the BOOT86.ASM file to save and restore those locations in a manner similar to the way that the IOBYTE and CDISK are handled (this requires some skill with 8080 assembly language).

5. Have you correctly provided for an initial starting vector for the 8088 as detailed in the system requirements?

6. If you have created a very large version of your CP/M-86 BIOS, you may need to change the ORG 6000H statement in BOOT86.ASM. BOOT86 assumes that the last address of the CP/M-86 BIOS will be less than 6000H. If it is more than 6000H, then a portion of the BOOT86 program will get clobbered when CP/M-86 is loaded in. If the last address of the CP/M-86 BIOS is greater than 6000H then change the ORG 6000H statement in BOOT86. ASM to be anything greater than the last address of the CP/M-86 BIOS. But, it should still be small enough for your CP/M loader to be willing to load it (small enough so that it loads below the BDOS in your system).

SUGGESTION FOR ENHANCEMENT

You may notice that your newly-running CP/M-86 system is slower than your CP/M system. This is largely due to the CP/M-86 BDOS which runs a lot more slowly than the CP/M BDOS. Swapping microprocessors back and forth to do I/O operations does cause some additional overhead processing which may further slow down CP/M-86. To eliminate some of the swapping overhead, you can begin to rewrite portions of your BIOS in 8088 assembler and to incorporate them directly into the CBIOS86.ASM file. One easy change, likely to have a noticeable effect, is to rewrite your console output driver. Typically the console output driver is a very easy portion of the BIOS to rewrite.

APPLICABILITY TO OTHER MACHINES

While the programs presented here are specifically tailored to the CompuPro Dual CPU, the idea of using a BIOS task block interface should be generally useful with any dual-processor board. For example, the same approach could be taken to bring up CP/M-68K on the Z80/68000 dual-processor board made by Cromemco. This approach might also be useful to someone running a Zenith Z-100 which uses an 8085 and an 8088. Operating systems like MP/M 816 (furnished by CompuPro) use similar techniques in reverse — the 8088 is used as an I/O processor for the 8085. Using the 8088 as an I/O processor has an additional advantage because the 8085 BIOS becomes very small, allowing a larger CP/M transient program area.

CONCLUSION

This article has shown a way to bring up CP/M-86 on older S-100 computers. The method presented can result in considerable savings in cost, time, and effort over alternative methods of getting into 16-bit processing. The running system generated with this method is suitable for long-term use. It is also suitable for use as a bootstrap to other implementations of 16-bit operating systems. The most difficult part of upgrading to a 16-bit operating system — getting the initial system to work — has been greatly simplified. ■

References

1. Bray, David W. Upgrading Older S-100 Computers to the CompuPro Dual Processor. *Microsystems,* Vol. 4 No. 9, September 1983; page 80.
2. Ratoff, Bruce R. The Godbout Dual Processor Board and CP/M-86. *Microsystems,* July/August 1981.
3. Kalish, Richard L. Upgrade Problems and Solutions. *CompuPro Product Users Manuals,* Vol. 2, January/September 1981; page 4.
4. Heywood, Stephen A. The 8086 — An Architecture for the Future. *BYTE,* Vol. 8, No. 6, June 1983; page 450 (part 1). *BYTE,* Vol. 8, No. 7, July 1983; page 299 (part 2). *BYTE,* Vol. 8, No. 8, August 1983; page 404 (part 3).

homebrewing

This column belongs to you, the tinkerer, hacker, DIYer, home integrator, or whatever it is that your S-100 machine files you under. If you are one of the proud (and stubborn) breed of high-tech frankensteinian inclination, here is the chance to show off your gold. If you have built or integrated your S-100 machine, share it with us. Tell us what your ultimate product has become, the boards and OS that you have running, problems that you encountered and how you overcame them.

You may write a letter (2-3 pages max.) suitable for publication — in this case give us some thrills, stay out of the mundane and boring prose. Or, if your word processor regularly laughs at your writing skills, simply tell us in your own words about your cherished system. We'll pick it up from there and do the polished writing for you. Oh! Yeah, don't forget a photograph . . . if you dare. Send to Homebrewing, *S-100 Journal, P.O. Box 12881, Raleigh, NC 27605.*

Today, Burt Hanagami, gives us an account that mirrors well the evolution of S-100 systems in the past 10 years. (Don't let the nostalgia bring you down!) Burt's system is living proof of the great maxim: S-100 systems are upgradable forever.

I was very delighted with your very first issue of the S-100 Journal. Your article on "Assembling a 68000-based S-100 Microcomputer" brought back some old memories.

I bought an IMSAI 8080 kit back in 1976. It took about three 24-hour days to assemble, and the next year-and-a-half to get it working. I was a real hardware novice, and made many many mistakes. I had cold solder joints, bent leads on IC's, the instructions had errors, and errata sheets were confusing. There were a few blown IC's that I had to track down, and some minor surgery (cutting of traces and soldering in wires) of the boards that had to be done. For some reason, I could not get the clock crystal on the CPU board to oscillate, so I ended up kludging up my own clock circuit, and wired it into the system. When I finally had the system up, I had to learn how to use it. I don't know how I survived those trying times, but by the time I was through, I was a well-seasoned computer hacker.

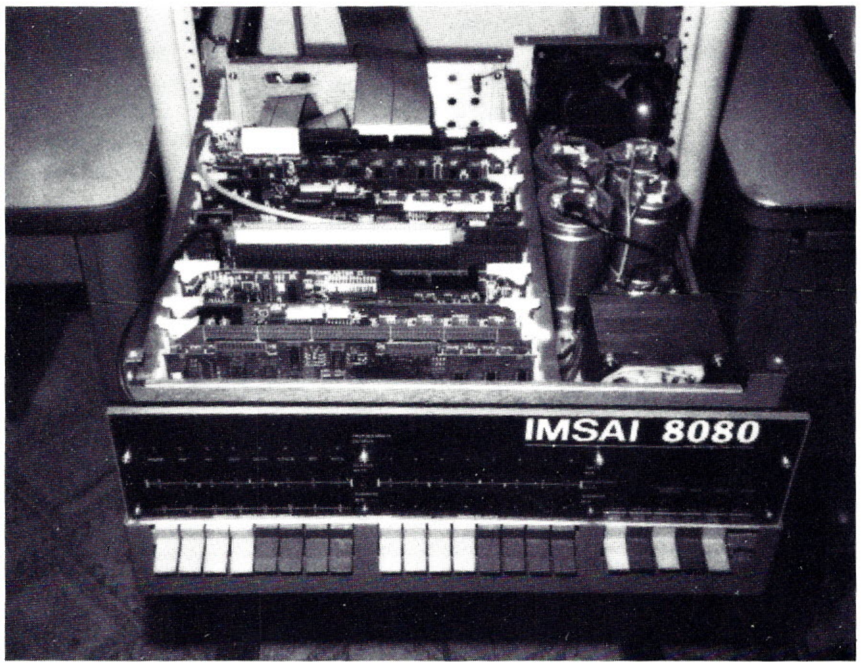
My original system was very primitive. In order to start the system, I had to toggle in a bootstrap loader (a cassette reading program) through the front panel switches. Since I had to try several times before I got it right, I used to keep the system powered up 24 hours a day. I originally wrote programs in machine language (not assembly language, but ones and zeros) by entering the data through the front panel switches. Once the programs were in and working, I could save and load the programs on a regular JC Penny cassette recorder with a Tarbell cassette interface. I could examine and modify the memory (one byte at a time), and run, stop, and single step my program through my front panel switches. I wrote a primitive ping-pong game where I would make the A light flash back and forth on the eight data lights. Another program I worked on would create sounds on a FM radio. My system would radiate so much EMI that nearby radios and television sets would pick up the EMI as whistles, clicks, and whirls. I could create certain noises by executing certain subroutines. I would actually string subroutines together to actually create structured noises. They were too primitive to be called music.

Later on, I got an IMSAI Video Input/Output (VIO) board, and hooked up a RF modulator, a television set, and a parallel keyboard. Presto, I had a 40×16 character display that I could actually type on. I added a Cromemco byte-saver board with a built-in PROM programmer. I programmed my bootstrap loader into ROMs so I would not have to toggle in the bootstrap loader each time I powered up. I complemented my 4K IMSAI RAM memory by adding two 8K Godbout memory boards. Eventually, I replaced all the RAM boards with two 32K 200ns Artec memory boards. I was running out of slots, so I also added another motherboard section. In the original IMSAIs, the motherboard came in 4-slot sections. After assembling the new 4-slot section, I installed the new section by connecting the sections together with 100 separate wire jumpers. For an operating system, I bought three books from Scelbi Computer Consulting: "8080 Monitor Routines," "An 8080 Editor Program" and, "An 8080 Assembler Program." These books had source listings of the respective programs complete with octal opcodes (most early 8080 systems did everything in octal). The code was also available in punched paper tape that could be read in on a teletype machine. But, because I did not have access to a TTY, or even had a serial port on my machine, I toggled in the whole thing through the front panel switches. Finally, I had to spend a few weeks modifying the software drivers to match my hardware configuration. It was worth it. I was finally free from the bondage of the front panel switches. Earlier, when I first got my system up, I used to get blisters on my finger tips until my callouses thickened, from handling the front panel switches. Now I could boot up instantly, and actually type in my programs in assembly language. I never ran BASIC or any other languages on my S-100 machine until I got CP/M up and running. The Timex computer (from Sinclair) that finally was reduced to about $15.00 in some department stores, did a lot more than my machine. The Apple II and TRS-80 computers weren't out then, so my machine was very impressive.

About five or six years ago, floppy disk drives were getting affordable, CP/M was gaining in popularity, and my machine was obsolete again. I did a complete overhaul of my system. I replaced the original motherboard with a 22-slot Jade ISO-bus, and upgraded my power supply to a 30-Amp. The ISO-bus motherboard had ground traces between each signal trace to reduce cross talk, and the whole board was designed around a specially tuned network mesh that dramatically reduced the EMI output of my machine. I got rid of the bus-termination board to reduce the amount of heat generating from inside my system. I bought a SSM (Solid State Music) IO/4 board (with 2 serial and 2 parallel ports), and added a Lear Siegler ADM 3 computer terminal. The terminal was a major enhancement. I was no longer limited to a 40×16 screen and I no longer had to reserve a section of memory to the video map. I also bought an Integral Data Systems 460 G printer. The 460 G was, at that time, the most advanced printer I could afford.

I researched the S-100 market very carefully, and decided to replace the guts of my system with SD Systems boards: the SBC-200 (Z80 CPU), Expandoram II (64K Memory), and the Versafloppy II (Floppy Disk Controller) boards. I installed two Shugart single-density, single-sided 8″ drives. I originally installed CP/M 1.4, and later upgraded to CP/M 2.2. These enhancements were made over a period of a couple of years. My cash was very limited, and the versatility of the S-100 system enabled these upgrades to be done gradually.

Sometime during this time frame, I obtained a 19.5″ rack-mount cabinet, and started installing everything on drawer slides. Everything except the computer terminal. For the floppy disk drives, I purchased a bare-bones disk cabinet from Jade, and crammed a severely modified power supply from Sunny International. I also managed to squeeze in a fan. I installed a special switch in the front lower right of the cabinet (actually mounted on brackets, under the drives) to shut the disk drive motors on and off to save wear and tear of the floppies. The disk drives and the CPU boxes were mounted on slides for easy access. For the printer, I made a platform out of some aluminum extrusions, and then mounted some heavy-duty slides on the sides. To secure the printer to the platform, I put some long screws through the same holes used for mounting the rubber feet. The box of paper for the printer sits on the bottom of the cabinet behind the power panel. It takes about two seconds to remove the printer from the cabinet, and about five seconds to put it back in. The power panel has master power switch, four separate switched outlets (for the disk drives, CPU, printer, and terminal), and two auxiliary outlets. I have a fuse and a Corcom EMI filter mounted on the master power switch, and a MOV (surge supressing, metal-oxide varesisters) installed on each outlet.



Today, my S-100 system is still alive and well. I've since swapped my Shugarts for a couple of Siemens 100-8 D (single-sided, double-density 8″ disk drives), and added a Ackerman Digital Systems Promblaster II (PROM programmer board that programs everything from a 2708 to a 27256 EPROM). Since banked CP/M 3.0 OS requires a two-bank memory system, I bought a second Expandoram II (64K board). I eventually modified my 64K memory boards into 256K boards. Thus, my system is now running with a half Meg of memory. I am running RAM-disk software for certain applications, and my current copy of CPM 2.2 has ZCPR 1.4 installed. I have a running version of CPM 3.0, but since none of my software requires it, and because of its added overhead, I seldom use it. For troubleshooting purposes, I own a Mullen TB-4 extender board with a built-in logic probe.

The S-100 machine was my first real machine. I literally grew up with it. I am extremely familiar with it, and I'll probably keep it forever.

Burt Hanagami
Ontario, California

CORRECTIONS FOR LOMAS MS-DOS BIOS

Below will be found several bugs uncovered in the MSDOS version of Lomas' BIOS for the floppy disks. These have caused some trouble when using Lomas' 8086, 8072 floppy controller, and I/O boards. The solutions which were implemented will hopefully aid other users who have run into the same problems. Unfortunately, I cannot supply corrected source code due to restrictions imposed by Lomas, which is understandable. Those wishing to follow these procedures will have to get the source code from Lomas and make the changes themselves or get a friend to do the job.

1. When trying to "type ahead" while a floppy is working, a byte or two of the data entered at the terminal is often lost due to the keyboard interrupts being impeded by the floppy controller hogging the buss. Rather than adding an 8089 DMA controller that would eliminate the problem completely, a partial answer has proved quite successful. The driver is altered so that the DTR line connected to the terminal is brought low before the floppy controller begins to transfer data and then brought high again after the transfer is complete. As long as the terminal has a keyboard buffer and a DTR/CTS handshake, this will eliminate nearly all lost data. This same idea can be used with a hard disk, but is unnecessary unless the operator is a very fast typist.

2. Quite often Lomas' BIOS will read a floppy sector incorrectly and fail to report the error, which makes even the driver think that all is OK. This can have serious results whenever you count on the data to be 100% perfect, and that is usually all the time. This was cured by having the driver check both status registers 1 and 2 instead of just register 1. In essence, register 2 from the 8272 is AND'ed with 21H and then OR'ed with the result the driver originally got from the 8272. This value is now used to determine whether all is OK. After about six months use, this seems to report all read errors.

3. There was another problem that occurred when the driver reported a floppy error to MSDOS. MSDOS would then post an error message on the screen and ask the operator if he/she would like to abort, retry, or ignore the trouble. If the operator chose to retry, either MSDOS or the driver would start writing all over the floppy. I don't know if the problem was with the Lomas driver or with MSDOS. Anyway, the problem was solved by having the driver tell MSDOS that no sectors have been transferred, even if MSDOS had called for a transfer of several sectors and some had been transferred before an error occurred.

4. Unfortunately, Revision 2.01, and I assume below, do not have the ability to verify that a sector was written correctly. Opinions on this subject seem to vary, but I consider the MSDOS "VERIFY ON" command a necessity. I want to be sure the backups made from the hard disk are 100% accurate. Fixing this omission requires writing some additional code, but it is well worth the trouble.

In addition to these bugs, I thought I would include a handy bit of information and dispel the belief that a logical disk on MSDOS is limited to 32 meg. Actually a logical disk as large as 256 meg can be used if a 4K sector size is used. To implement sector sizes larger than 512 bytes, you have to change the word at 101H (on MSDOS 2.11) in the MSDOS.SYS file to the maximum sector size to be used. You will then be able to install a driver with a sector size up to and including that value.

I hope this information proves useful to a few readers of the S-100 Journal.

Hul Tytus
Cincinnati, Ohio

The Bug Report department is a way for communication among users and between users and manufacturers. Users can use the column to report bugs and undocumented features (or documented but non-existent features) on hardware and software, and to suggest solutions to bugs. Manufacturers and software publishers are encouraged to send in notices about bugs, updates, and similar information useful to those using their products.

THE TELETEK SBC 86/87

Duane Chinnow
Teletek

Teletek's SBC 86/87 is a 16-bit slave single board computer intended for use on the S-100 bus in a multiuser/multiprocessing system. This slave board gives the system integrator the flexibility of mixing 8-bit economy with 16-bit processing power.

Since the SBC 86/87 is designed as an I/O mapped slave on the S-100 bus, it can be added to any existing S-100 system to expand the processing capability. In the following pages, I will introduce the features of this slave SBC and show how it interfaces to an existing S-100 bus system.

DESIGN PHILOSOPHY

The SBC 86/87 was designed to provide an easy-to-implement 16-bit alternative for system integrators. With this board, the integrator can provide 16-bit, high-speed performance where required in a system, and yet retain the cost-effectiveness of 8-bit SBCs for system functions that do not need the additional capability. With the addition of this slave to an existing 8-bit system, the user can access CP/M-86 application software and the power of the 8087 math coprocessor for numeric intensive applications.

The interface to the S-100 bus was kept as simple as possible to allow this board to work with a variety of S-100 systems. All communications to and from the slave take place through two I/O mapped FIFO buffers. This greatly simplifies the requirements for the bus master.

THE CPU AND NPX

The SBC 86/87 utilizes the Intel 8086 CPU and the companion 8087 math coprocessor, both running at 8MHz. Since the SBC 86/87 operates independently of the S-100 bus, the internal speed can be different from that of the main system CPU.

The 8087 coprocessor adds arithmetic, trigonometric, exponential, and logarithmic instructions to the standard 8086 instruction set. The 8087 will significantly improve the performance of the CPU during numeric intensive operations. The 8087 coprocessor conforms to the proposed IEEE Floating Point Standard.

In addition to the 8087, this board design incorporates several other peripheral-support ICs that increase the capability of the slave SBC. These include an Intel 8208 DRAM controller, the Intel 8256 MUART, and the Signetics 2651 USART.

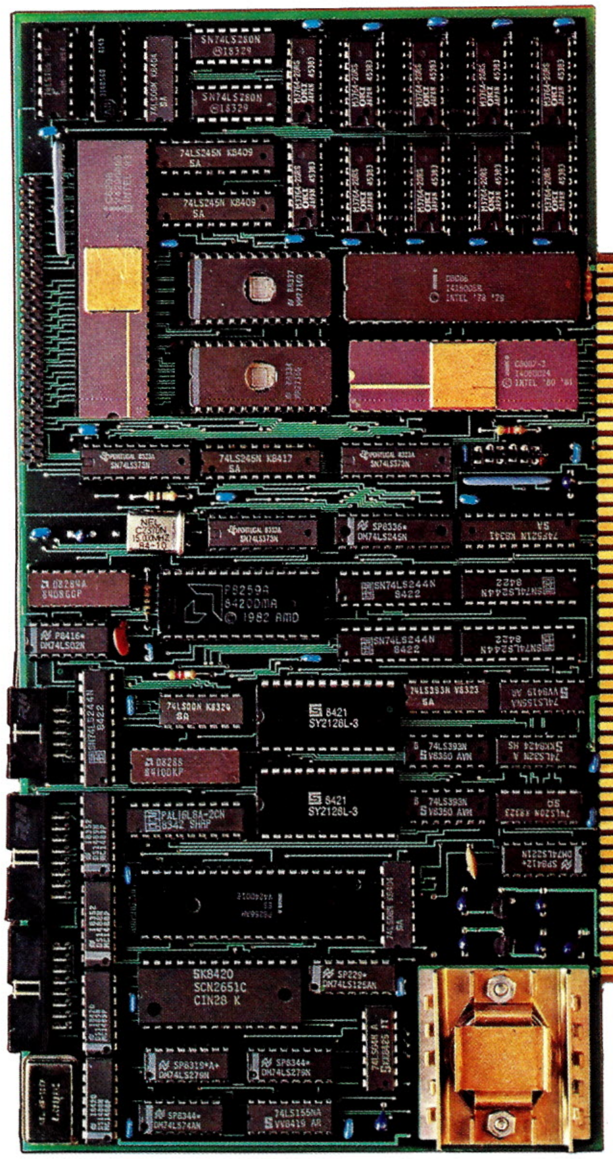
ON-BOARD MEMORY

The standard SBC 86/87 includes 128K bytes of RAM, expandable to 512K by using 256K-bit RAM ICs. The memory layout uses stacked RAM ICs to reduce the physical size of the array. The standard board also provides 4K bytes of EPROM using two 2716s. It is expandable to 64K bytes by using two 27256 EPROMs. Normally, the on-board EPROM contains hardware initialization and system-boot software.

The on-board RAM controller, an Intel 8208, supports either 64K or 256K devices. It generates the necessary signals to address, refresh, and directly drive the memory array. The controller is automatically initialized upon reset by a 74LS165 shift register. The 8208 controller allows operation without wait states when accessing

TELETEK SBC 86/87

USE:	S-100 16-bit slave SBC for use in TurboDOS multi-user/multiprocessing systems.
MANUFACTURER:	TELETEK ENTERPRISES, INC. 4600 Pell Drive Sacramento, CA 95838 (916) 920-4600
FEATURES:	Processor — 8086 16-bit, 8 MHz. Optional 8087 Math Coprocessor. Memory — 128K RAM (expandable to 512K using 256K DRAMs), 4K ROM (expandable to 64K), 4K FIFO. I/O — Two RS-232C serial ports, One Centronics compatible printer port.
SOFTWARE:	TurboDOS
MANUALS:	Technical Reference Manual, 31 pages.
PRICE:	$899 for 128K, w/o 8087 (100 quantity).



RAM memory, while a single wait state is inserted for each access of EPROM.

THE INTEL 8256 MUART

The 8256 is labeled as a Multi-Function Universal Asynchronous Receiver-Transmitter (MUART). As the name implies, this peripheral-support IC offers more than just a serial communications port inside its 40-pin DIP package. The extra functions include 16 bits of parallel I/O, five 8-bit counter/timers, and an eight-level priority interrupt controller.

The asynchronous serial communications port provides one of the two RS-232C compatible serial ports on the SBC 86/87. (The other is generated by the Signetics 2651 USART.) To permit a variety of operating speeds without additional external components, the MUART serial port includes an internal software programmable baud rate generator. This serial port can be programmed by the CPU for different character sizes, parity generation and detection, error detection, and start/stop bit handling. The line drivers and receivers are supplied on board of the SBC 86/87, eliminating the requirement for "paddle boards" on the peripheral cables.

A Centronics-compatible printer port is derived from the two parallel ports on the 8256. One parallel port is responsible for the printer data lines and a portion of the other for the printer control signals. The remaining signals of the second parallel port are used in the interrupt circuit. They also provide optional control lines for the MUART serial interface. The cable line drivers for the Centronics port are also included on board.

The five 8-bit timing channels furnished by the MUART can also be used for event counting. Additionally, four of the channels can be cascaded into two 16-bit counter/timers if desired. The clock source for these circuits comes from the 5.0688 MHz oscillator used with the 2651 USART.

The SBC 86/87 supplies two cascaded eight-level Priority Interrupt Controllers (PICs) to resolve all on-board and bus master interrupts. An Intel 8259A acts as the master PIC. The second PIC is provided by the 8256 MUART.

Table 1 shows the PIC assignments as well as the Non-Maskable Interrupt (NMI) assignment.

8259A PIC

Number	Usage
NMI	Memory Parity Error
0	MUART
1	USART Transmit Buffer Empty
2	USART Receive Data Available
3	Tx INT from master
4	Rx INT from master
5	Aux. INT from master
6	8087 NPX
7	EXPANSION BUS

8256 MUART

Number	Usage
0	Timer 1
1	Timer 2 or Port 1 P17 Interrupt
2	External Interrupt (EXTINT)
3	Timer 3 or Timers 3 & 5
4	Receiver Interrupt
5	Transmitter Interrupt
6	Timer 4 or Timers 2 & 4
7	Timer 5 or Port 2 Handshaking

Table 1. *PIC and NMI assignments.*

THE 2651 USART

The 2651 combines, in a single 28-pin DIP package, the necessary features for a serial interface with a programmable baud rate generator. This allows the designer to conserve valuable board real estate for other functions. The baud rates are derived from an external clock oscillator, and their operation is independent of the MUART serial port. The 2651 serial interface provides full modem control signals. These signals support hardware handshaking protocols of peripheral devices that require them.

S-100 FIFO INTERFACE

All information other than protocol control signals, exchanged between the S-100 bus master and the SBC 86/87, occurs via the dual FIFO memory circuit on board the slave. One FIFO buffer is dedicated to receiving data from the S-100 master. The other is dedicated to sending data to the master. The SBC 86/87 cannot function as the bus master of a system. Therefore, it depends on a master processor to manage all the system data transfers through these I/O mapped buffers.

This method of system communication combines the simplicity of mapping the slaves as ports in the master I/O space with the high performance of using DMA block data transfers between the master memory and the slave FIFO buffer. Unlike a memory-mapped DMA system, where the slave's RAM is mapped into the master's memory space, the slave CPU does not have to be idle during the transfer. While the data transfers are taking place between the master processor and the FIFO, the slave processor is free to perform normal operations. Also, no complicated memory management capability is required of the bus master.

Asynchronous communication is inherent in the design of this type of system. The master processor and any SBC 86/87 slaves in the system operate independently of each other. In fact, the master processor and the slaves can be running at altogether different clock speeds.

MASTER/SLAVE COMMUNICATIONS

The SBC 86/87 appears as a cluster of four I/O ports to the bus master. By accessing these ports, the master can control data transfers between itself and the slave. Each SBC 86/87 has option jumpers that allow it to be addressed on any four-port boundary within the first 256 I/O locations. Table 2 illustrates how each port is used.

STATUS REGISTER

The status register contains three flags that can be set by the slave and read or cleared (by sending interrupts) by the master. This register is normally polled by the master during network communications over the S-100 bus. The diagram in Figure 1 illustrates how the status register is defined.

INTERRUPTS

The bus master can cause three different interrupts to the slave: the Rx INT, the Tx INT, and the Aux INT. Anytime one of these interrupts is sent to the slave by accessing the appropriate I/O port, the associated flag in the status register is cleared.

The Rx INT is used to indicate to the slave that the master has written data to the slave RxFIFO. The Tx INT is used to indicate that the master has read data from the slave TxFIFO. The Aux INT and Aux flag, in the status register, are free for programmer defined functions. The most common usage is for a "slave running" check: the master sends an Aux INT to the slave, and the slave responds by setting the Aux flag in the status register.

RESET

Two levels of reset are available on the SBC 86/87. The first is a system reset; all boards (and therefore all users) in the system are reset simultaneously by activating the RESET* signal (pin 75) on the S-100 bus. The second type of reset is a software reset; the master issues an I/O command to individually reset one user. This allows the master CPU to "wake up" a user that doesn't respond to an inquiry.

SOFTWARE

At the present time, Teletek provides software support for the SBC 86/87 under the TurboDOS operating system. TurboDOS allows a system integrator to assemble a powerful multiuser/multiprocessing system based on Teletek's master, slave, and hard-disk/tape controller boards. The standard TurboDOS implementation of the SBC 86/87 is CP/M-86 compatible. An MS-DOS 1.0 emulator is furnished at no extra cost.

For other applications, Teletek can provide examples of the existing software to aid in the development of new drivers. ■

port0	*input*	=	read from slave status register
	output	=	send Tx INT to slave, reset TxRDY flag to master
port1	*input*	=	read data from slave TxFIFO
	output	=	write data to slave RxFIFO
port2	*input*	=	send Aux INT to slave, reset Aux flag to master
	output	=	reset slave RxFIFO address counters
port3	*input*	=	reset the slave
	output	=	send Rx INT to slave, reset RxRDY flag to master

Table 2. *I/O Port Assignments.*

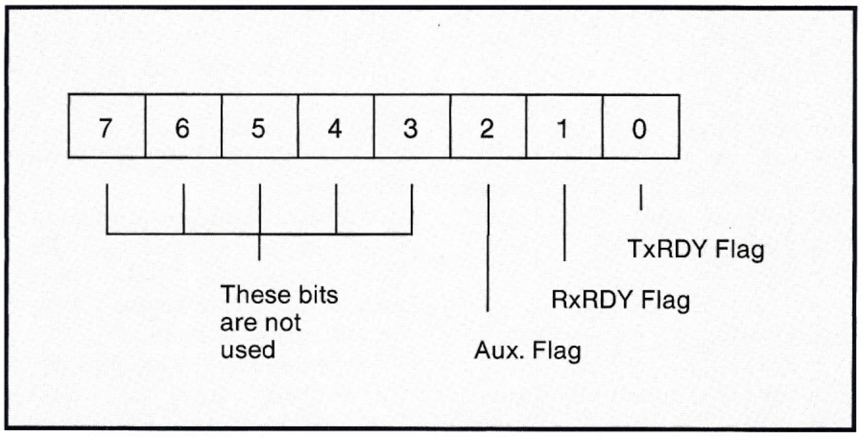


Figure 1. *Slave to Master Status Register Bits.*

THE THUNDER 186 SINGLE BOARD COMPUTER

Brian Smithgall

If you are looking for a powerful yet economical system, consider the Thunder-186 single board computer by Lomas Data Products. The board features the powerful 8-MHz 80186, 256K of Dynamic RAM, two serial ports, and one parallel port. It will control both 8- and 5¼-inch floppy disk drives. The Thunder-186 is available for less than $1000. With it, a full system with two floppy drives and terminal can be built for less than $2000.

PROCESSOR FEATURES

Many of the features of the Thunder-186 are actually functions of the 80186 chip. Foremost, the processor is object code compatible with the 8086/8088, allowing it to run most popular software packages. In addition, the 80186 provides several new instructions, most notably those for block I/O, pushall and popall registers, and some immediate arithmetic.

The real benefit of the 80186 is its inclusion on board of several functions normally requiring peripheral chips. This allows the designer to package more capabilities on a single S-100 board. In the Thunder-186, some of the extra capabilities are an interrupt controller, two DMA controllers, and three timer/counter functions. One of the DMA controllers is used as the floppy disk controller. Two of the timers are used by the serial ports to set the baud rates, and the other is used as the real-time clock by the supplied CCP/M operating system.

Another significant advantage of the 80186 over the 8086 is its hardware computation of the complex Intel addressing scheme, which increases speed by about 20%.

Unfortunately, the external inputs to the timer/counter circuit are excluded from the board. These were left out probably because, in the principal operating mode, the timers are not available (they are tied to the serial ports and the real-time clock).

SUPPORT FEATURES

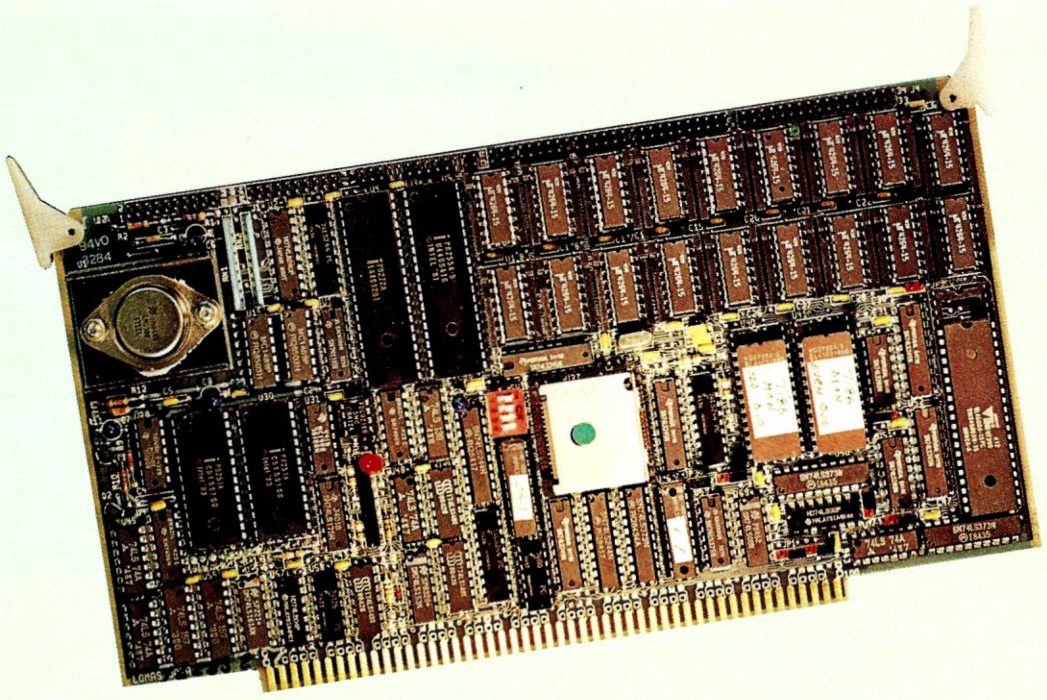
The support hardware on the board provides several useful features. Two serial ports are controlled by two 8251A USARTs. One serial port is used by the system as a terminal. There is also a parallel channel controlled by a 8255A. The output is used as a printer port. The input is used by the monitor at boot time to read the setup switches, and thus it is not generally available.

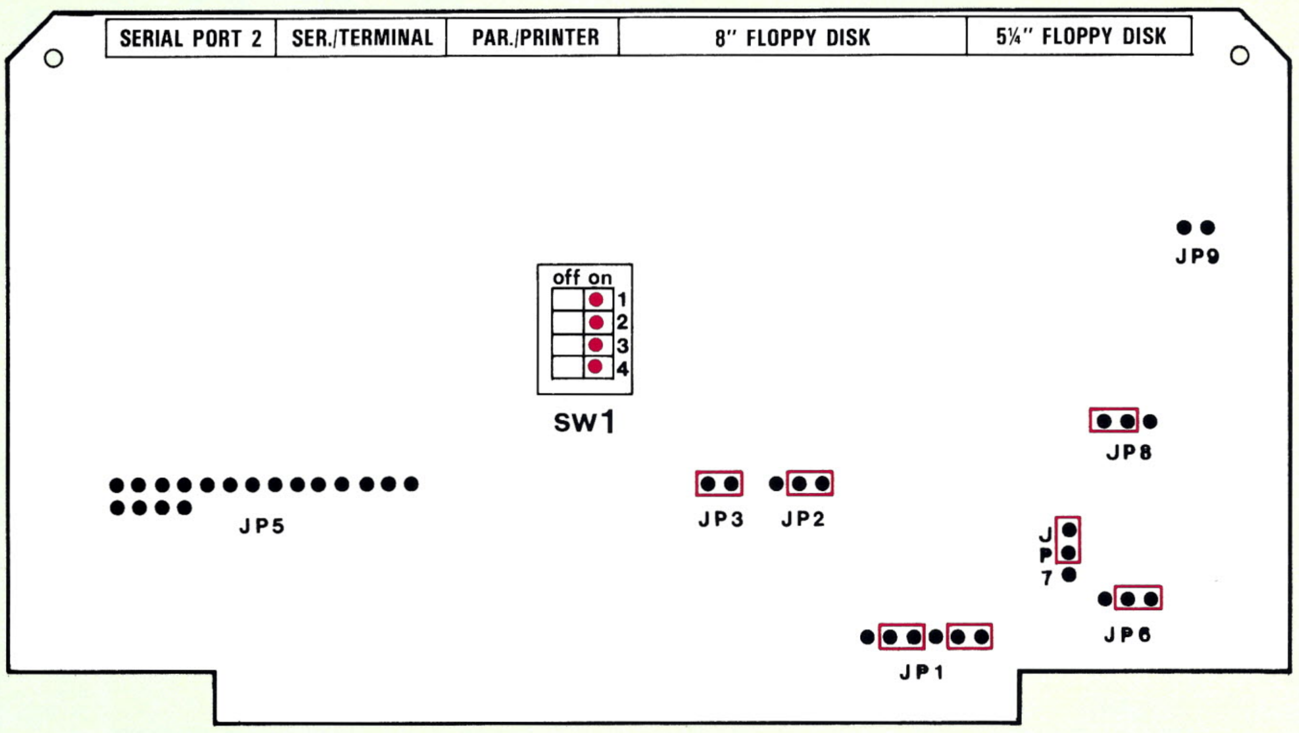
The board supports any combination of two 8- and/or 5¼-inch floppy disk drives. This gives a great deal of flexibility in using existing equipment.

Dynamic memory is used with no wait states at either 128K or 256K bytes. Up to 1M byte of additional memory may be added.

The more sophisticated user may wish to alter the boot EPROMs. These may be enlarged to a whopping 64K if necessary.

The Thunder-186 conforms to the IEEE-696 standard with pins 65 and 66 additionally defined as external DMA requests. Hence, boards added to the system should not use these pins for their own functions. Added boards should also provide a full 20-bit address and 16-bit port decodes, or conflicts might arise in the system. For example, if a graphics board used port 28h but only decoded the lower 8 bits, a conflict would arise with the 80186 port FF28h which masks the interrupt sources.





Thunder-186 switch and jumper settings to use with 9600-baud terminal, 8" drives, auto boot, 8K of EPROM, and no slave boards. Red rectangles indicate installed shunts. Red dots indicate positions of microswitches.

SUBSCRIBE

C F85

PLEASE ENTER MY SUBSCRIPTION TO S-100 JOURNAL.
ENCLOSED IS ☐ $14.00 FOR 1 YEAR. ☐ $25.00 FOR TWO YEARS.

(Canada, Mexico, and England please add $3.00 per year. Other countries please add $6.00. All payments must be in U.S. currency.)

NAME ______________________________

____

ADDRESS ______________________________

____

PLEASE MAIL TO: S-100 JOURNAL, PO Box 12881, Raleigh, NC 27605

The S-100 phantom line is asserted during memory accesses within the Thunder-186 board. This may be important to users with additional boards.

SETUP

Configuring the Thunder-186 is very simple. The principal options are chosen by means of a 4-slide DIP switch. Two switches select one of four baud rates for the terminal. Another switch determines if boot is to be done from 8- or 5¼-inch floppies. The remaining switch allows the user to not boot, but rather to run an on-board program called the monitor. This monitor provides several simple and useful commands. They are discussed below in the monitor section.

Jumper JP1 allows advanced users to install their own boot sequence with up to 64K of EPROMs (2xAMD 27256). Normally this is left as set at the factory.

Jumper JP2 sets the floppy disk drive write precompensation to 125 or 250 nanoseconds. See your disk drive manual to determine the setting for your drives.

Jumper JP3 is for users with slave processors on the same S-100 bus. It provides reset to slaves on master reset.

The interrupt jumpers (JP5) allow the user to connect the eight S-100 interrupt vectors to any of the four 80186 primary interrupt vectors. It may be necessary to alter these to match the requirements of additional boards.

A terminal with normal handshaking is required to boot the system. The floppy drives need not be hooked up to use the monitor (a nice feature). The board should be used with a terminated S-100 motherboard. This is very important if other boards are to be used in the bus. If your bus is not terminated, purchase a small terminator card from someone like Viasyn.

OPERATING SYSTEM

The operating system supplied with the board is the popular Concurrent CP/M-86. It allows multiusers and multitasking. Thus, multiple jobs may be run from the same board by time-slice task swapping. In the "single user/multitasking" environment, a special code, called the lead-in character, is typed to switch virtual terminals. The lead-in character for switching terminals may affect other software (such as my editor), but the manual gives instructions for changing it.

A parallel printer can be connected to one of the headers. Beware that this printer header is displaced one row- so pin one of the header cable needs to be connected to pin three of the connector. Alternatively, the second serial port may be used for the printer. This is done by running a command file after boot or reassembling the operating system.

A routine to do a track-for-track disk copy is not available among the operating system functions. A fairly simple alternative technique is used to copy the system tracks using the monitor.

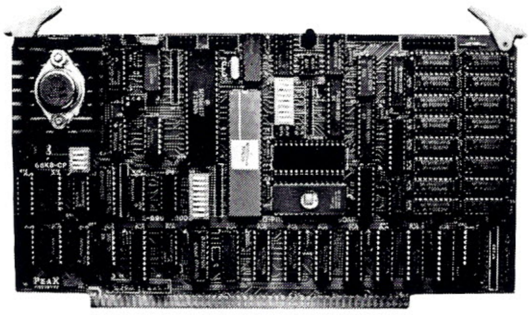
MONITOR

The monitor is a program burned in EPROM which is available at boot time (by setting a DIP switch). It first performs some self-tests (another nice feature) and board setup, and then allows the user to perform simple board level operations. These include hex arithmetic, port I/O, raw disk I/O, examining and changing memory, and running and tracing programs. The monitor is powerful and simple to use. A list of the monitor commands is given in Table 1. Unfortunately, the self-tests at boot clear

MONITOR COMMANDS:

B:	BOOT SYSTEM FROM FLOPPY DISK
C:	CONVERT DECIMAL TO HEX
D:	DUMP MEMORY SECTION (HEX AND ASCII)
E:	ENTER DATA INTO MEMORY MANUALLY
F:	FILL SECTION OF MEMORY WITH VALUE
G:	GO. LOAD REGISTERS, SEGMENTS, IP AND START EXECUTION
H:	HEX ARITHMETIC (+,—,*,/)
I:	INPUT FROM I/O PORT
L:	LOAD DISK SECTORS TO MEMORY
M:	MOVE MEMORY SECTION
O:	OUTPUT DATA TO I/O PORT
R:	DISPLAY REGISTERS, SEGMENTS, AND FLAGS. (SEE GO)
S:	SEARCH MEMORY SECTION FOR DATA STRING
T:	TRACE INSTRUCTION CYCLES. (SINGLE STEP CAPABILITY)
V:	VERIFY MEMORY SECTION WORKS
W:	WRITE MEMORY TO DISK SECTORS

Table 1. *Monitor commands of the Thunder-186. In addition, the on-board monitor sets up peripherals, automatically sizes memory, and performs memory tests.*



68K8-CP

The World's First 68000 Coprocessor

Peak Electronics' 68K8-CP is a high performance processor card designed to function as one of several CPU elements within a multi-processor S-100 system. This card features the MC68008™ (8-bit object code compatible version of the industry renowned 68000), up to 512K bytes of RAM and 128K bytes of EPROM, two serial ports and a parallel printer port.

The true power of the 68K8-CP card is shown with its ability to be plugged into any existing S-100 system running CP/M®-2.2 and to be running CP/M-68K within minutes without any change in existing hardware or software. This card does not replace your current processor. All of the original system's devices (RAM, disks, and other peripherals) are immediately available to the user of CP/M-68K, files can be accessed by whichever operating system is currently active.

Features:
- Does not replace your current CPU card
- IEEE-696-1983, S-100 Compatible
- MC68008 8 or 10 MHz CPU
- 128K bytes of RAM expandable to 512K
- 8K bytes of EPROM expandable to 128K
- No wait state access to on board RAM and EPROM
- Two high speed serial ports (up to 38.4K BAUD)
- 8-bit parallel printer port
- Supports CP/M-68K and Concurrent-68K
- Onboard 16-bit counter-timer
- Includes printer buffer and RAM disk firmware

$995.00, includes CP/M-68K

PEAK electronics
P.O. Box 700112
San Jose, CA 95170-0112
(408) 253-5108

S-100 Journal Vol. 1 No. 1 Available

A limited number of the first issue is still available for $4.90 per copy (we pay postage). To get your copy send this ad (or a note with your name and address) to Back Issues, S-100 Journal, P.O. Box 12881, Raleigh, NC 27605. Prepayment is required to process your order.

NAME ______________________________

ADDRESS ____________________________

____

NOTE: Several subscriptions that arrived during this Fall asked that we start with the first issue. To retroactivate a subscription we would have to incur additional expenses that the low subscription price does not cover. So please use this form instead. Thank you for your cooperation.

memory, so software bugs that jam the system are not easily traced.

DOCUMENTATION

The Thunder-186 suffers from a malady common to many products these days — a lack of good documentation. The manual leaves out some fairly major things, like the function of certain jumpers and the printer cable trick. In defense, however, the company has always been very responsive to questions and helpful in solving specific problems.

CONCLUSIONS

The Thunder-186 is an excellent computer that offers the flexibility of the S-100 bus. That makes it a great choice for both the casual programmer and the sophisticated OEM. The Thunder more than lives up to its claims, and it offers a good deal more versatility than many other boards on the market.

The board is available at a discount from Integrated Microsystems, as well as from other dealers. The unit may also be purchased as part of a system with chassis, floppy drives, etc. Options include a hard disk drive, and more memory.

ADDRESSES

Lomas Data Products
182 Cedar Hill Street
Marlboro, MA 01752
(617) 460-0333

Integrated Microsystems
PO Box 2415
Del Mar, CA 92014
(619) 481-6530

Viasyn Corporation
3506 Breakwater Court
Hayward, CA 94545
(800) VIASYN1 (outside CA)
(800) VIASYN2 (within CA)

IF YOU ARE AN S100 BUS DEVELOPER or SERIOUS USER then let's be frank . . .

There is a wealth of fine S100 processor and support boards,

but WHERE 'S THE BEEF ???

Up until now you had to develop custom Hardware and Software to complete even simple applications.

A NEW DAY HAS COME TO S100...

CSD is making available a complete line of Realtime Intelligent Products to speed up your job.
These products are Preconfigured to plug into your BUS and work.
When needed, there are simple Host commands to reconfigure the board and save the new state.

Write or call for details on feature packed 80186 based products for these realtime applications.

COMMUNICATION   FAULT TOLERANT PROCSSORS

DATA ACQUISITION   AUTOMATED TEST

ARRAY PROCESSING   IMAGE/GRAPHIC PROCESSOR

Source or Custom Software available to our OEM accounts

other fine S100 products at the ABSOLUTE BEST PRICES

COMPUPRO - MACROTECH - ADVANCED DIGITAL

ICD - TELETEK - IMS - LOMAS - EARTH

FULLY INTERGRATED SYSTEMS AVAILABLE   CORPORATE LEASING   VISA / MC / AMEX

COMPUTER SYSTEM DESIGN, LTD

STONY BROOK TECHNOLOGY CENTER PO BOX 2131 SETAUKET, NY 11733

Mon Thru Fri 10AM-6PM and Sat 10AM-2PM E.S.T.   516 - 689 - 3100

reader i/o

This department will be an interactive medium for readers. Everyone is invited to send questions, answers to previous questions, or other comments for publication. Your questions and comments may range from technical to philosophical. Whenever a discussion on a topic starts up, we'll try to keep it going for several issues. Here's a few suggestions to get you started: The future of S-100 computing . . . 32 bits on the S-100 bus . . . Software trends . . . or just plain old technical questions about any S-100 topic.

Send your (typed/printed) letters to Reader I/O, *S-100 Journal, P.O. Box 12881, Raleigh, NC 27605.*

THE S-100 SOFTWARE GUIDE

The *S-100 SOFTWARE GUIDE* that was announced in the first issue of S-100 Journal is not being published this year (1985). We plan to publish it next November (1986).

THE GUIDE is a software advertising medium for publishers of software for S-100 systems. For more information please contact *THE GUIDE,* c/o S-100 Journal, PO Box 12881, Raleigh, NC 27605.

ARTICLES ON SPECIAL APPLICATIONS

S-100 Journal is seeking articles that describe in detail how S-100 systems are being used. We especially need articles that demonstrate the outstanding power of modern S-100 systems and how that power is being applied to perform actual tasks in industry, academia, research, business, etc.
To learn more about writing for S-100 Journal, please request our Author's Guidelines. Or call Jay Vilhena at 919-839-0115 if you would like to discuss potential articles that you are interested in submitting.

FOR SALE

TELETEK SYSTEMASTER S-100 SBC w/ CP/M-2.2, $395; ParaGraphic SuperGraphics board and terminal, software, to 512x576, $295; Sequential Circuits M206 Musicboard (MAX), 6-note polyphonic, 80 voices, non-S100, serial interface, $195; California Digital 256K-1M memory, $150; VT100 83-key keyboard, $150; 80-trk drives, $50. Buckley, Riverview, Durham NH 03824. (603) 868-5006.

FOR SALE

CompuPro 85/88 CPU board plus the following software with manuals. CP/M-80: BASCOM, V-edit, dBase II 2.4, Turbo Pascal. CP/M-86: Pascal MT+86 3.1, Supercalc, V-edit. Upgrading to MC68000 system. Neil Swenson (Atlanta). (404) 255-8007.

FOR SALE

RGB Video-512 Color Graphics S-100 board by I/O Technology. Uses NEC 7220 graphics chip. With cable; software C, Pascal; extra 7220 chip. Works fine. Cost $1040, asking $700 or best offer. John Roberts (404) 449-7776.

This department is for publishing non-commercial small advertisements. There is no charge to place an ad. Readers may take advantage of this service to sell or buy used hardware, trade personal programs, etc. Messages must be kept short. Send ads to S-100 Journal, BITS, PO Box 12881, Raleigh, NC 27605.

S-100 SYSTEMS CAN BE UPGRADED FOREVER!

s-100 directory

The S-100 Directory will appear in the back pages of every issue of S-100 Journal. All services and products being offered through the directory must be related to S-100 systems. Messages must be 50 words or less.

To list your business in the directory, send $80 for one year or $140 for two years, along with your entry, to S-100 Directory, c/o S-100 Journal, P.O. Box 12881, Raleigh, NC 27605.

Users Groups and Bulletin Boards, not identified with any nonS-100 system, are also welcome for listing in the directory. There is no charge for listing nonprofit Groups or Boards.

ARIZONA

S-100 (602) 991-7870
14455 N. 79th. St., Scottsdale, AZ 85260

S-100 stocks most popular brands of S-100 boards, peripheral devices and software. We offer excellent pricing on all items especially drive subsystems.

CALIFORNIA

COCHLIN COMPUTER SYSTEMS INC. (415) 495-5999
330 Townsend St. #107, San Francisco, CA 94107
303 Potrero St. #29-104, Santa Cruz, CA 95060

Authorized CompuPro System Center. We provide on-site service and maintenance, custom software design, multi-user system integration, user training and more.

DATABANK (805) 962-8489
228A West Carrillo, Santa Barbara, CA 93101

SMD 8" Disk Subsystems — 84 — 168 — 300 MBytes with S-100 Controller. Mag Tape — 9 Track Tape. Boards from CompuPro, Macrotech, Dual Systems — all S-100.

MENTZER COMPUTER SYSTEMS (415) 340-9363
1441 Rollins Rd., Burlingame, CA 94010

CompuPro Full Service System Center. Custom S-100 systems, CAD Systems.

MICRO POWER UNLIMITED (805) 584-6789
1716 Erring Rd. #102, Simi Valley, CA 93065

Authorized CompuPro System Center. We provide on-site service and maintenance, custom software design, multi-user system integration, user training and more.

PRO*ACCESS (415) 969-4969
P.O. Box 747, Mountain View, CA 94042

If you don't find the File*works File Processor program the most powerful file backup and manipulation program you have ever used, return it within 30 days for your money back. For CP/M 2.2 or later, including CompuPro 8/16 series. $149. Please specify format.

FLORIDA

JURIS SCRIVENER, INC. (305) 920-7772
2019 B Hollywood Blvd., Hollywood, FL 33020

System Integrator and Vendor. S-100 Granturbo Computers running TURBODOS 8 and 16 bit, will also run CP/M and MS-DOS. Systems for 4 users and up. FMS 80 Data Base Management Software for S-100 computers. Primage printers.

ILLINOIS

SMALL BUSINESS SYSTEMS, INC. (312) 579-3311
1016 E. 31st St., LaGrange Park, IL 60525

CompuPro multi-user systems, networking, peripherals, software. Fast, reliable, and versatile systems for any business computer needs.

s-100 directory

MARYLAND

MICRO COMPUTER COMPANY, INC. (301) 942-5442
101 Wheaton Plaza North, Wheaton, MD 20902

We offer a wide selection in Syntech Data International and CompuPro S100 Bus products.

MICHIGAN

WALDORF ASSOCIATES, LTD. (313) 996-0646
4477 Jackson Road, Ann Arbor, MI 48103

CompuPro Full Service Center specializing in UNIX on S-100 Buss and special configurations for both R&D shops and Universities.

MONTANA

HARRIS-LARSEN & ASSOC. (406) 721-7070
211 North Higgins, Missoula, MT 59801

CompuPro Full Service System Center. System Integrators. S100 Consultants.

NEW YORK

RENARD SYSTEMS INC. (716) 833-4758
4248 Ridge Lea Road, Amherst, NY 14226
Service offices in NYC and Rochester

Full Service CompuPro System Center specializing in word processing automation, AutoCad on S-100, laser printers, very large hard disk drives, and tape backup. Service contracts to entire North East (out of NYC). National distributors for a variety of system enhancing software products, including PrintMan, Phantom Printer, Smart-Op and Speed-Op by Data Base Administrators.

PENNSYLVANIA

FOREFRONT TECHNOLOGY CORPORATION (215) 386-1500
3508 Market Street, 2nd Floor, Philadelphia, PA 19104

CompuPro (Viasyn) System Center Plus! Software emphasized to take full advantage of hardware capabilities.

FRASER BUSINESS EQUIPMENT (215) 378-0101
523 Franklin Street, Reading, PA 19603

AUSTRALIA

BJ ELECTRONIC SYSTEMS (079) 513064
11 Wood St., Mackay 4740, Australia TELEX AA 46044

Fully integrated Fleet Management System with fuel, maintenance, driver, workshop, spare parts control and debtors, creditors, stock control. Payroll system.

CANADA

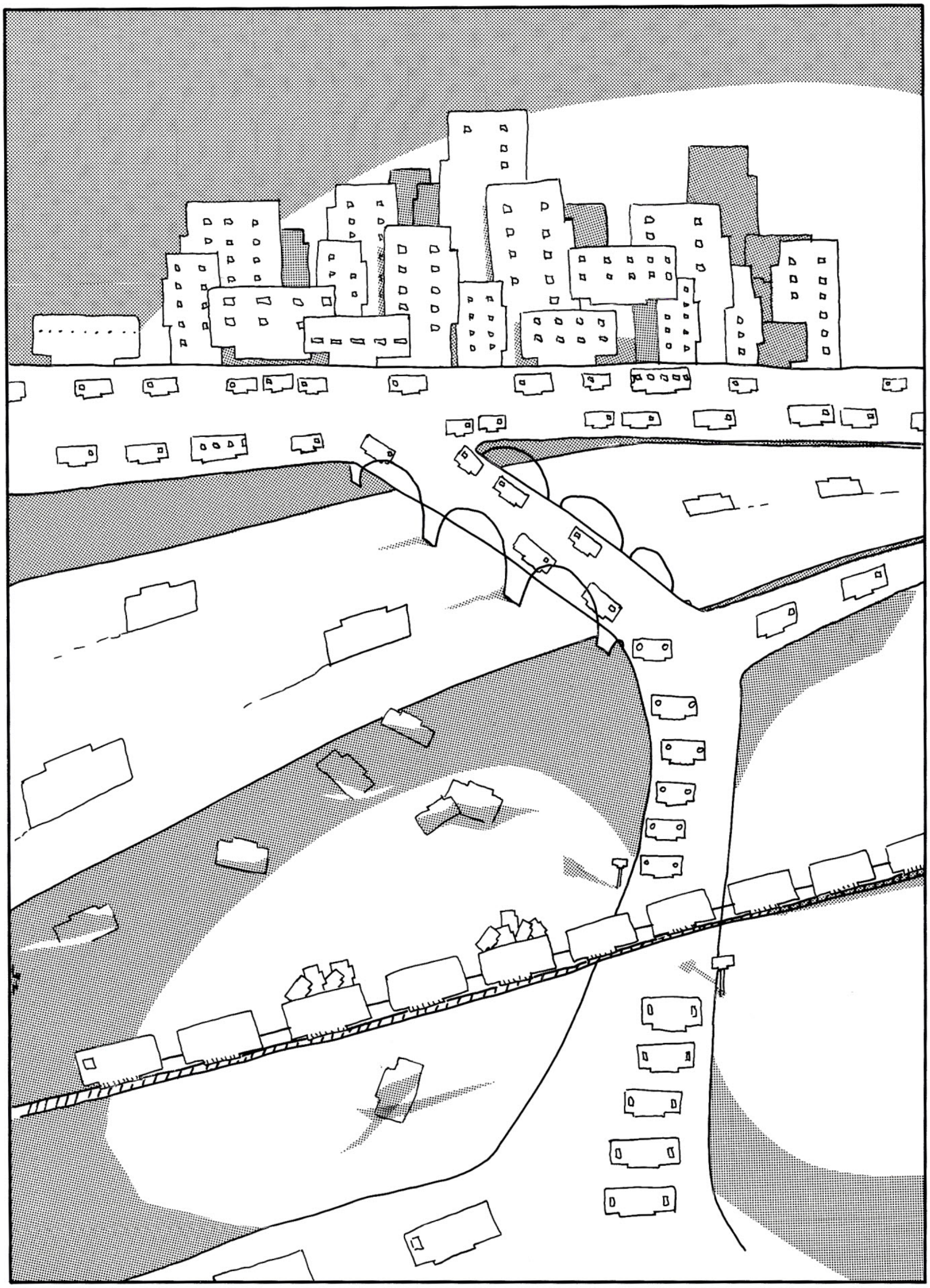
M.A.E. MICROSYSTEMS, INC. (514) 341-1210
8255 Mountain Sights, #175, Montreal, Canada H4P 2B5

S-100 Integrator for Custom Applications. Turbodos Applications 8/16 Bits. Sweda Cash Register interfacing and more.

GERMANY

CO-SY, COMPUTER UND SYSTEME (2173) 52071 or 72
Krischerstr. 70, d-4019 Monheim, West-Germany

We are looking for cooperation with Software manufacturers-authors to adapt and translate Software into German. We specialize in UNIX and C language, MS-DOS and C, and C-Basic.



S-100 Journal accepts articles on a variety of S-100 and S-100-related topics. Typical articles can range from reviews to integration of systems, or to special applications. Articles can be about hardware or about software, but articles about software should still relate to S-100 systems.

To learn more about writing for S-100 Journal, please write to Jay Vilhena, S-100 Journal, PO Box 12881, Raleigh, NC 27605, and request our Author's Guidelines.

Teleteknology
One Success After Another.

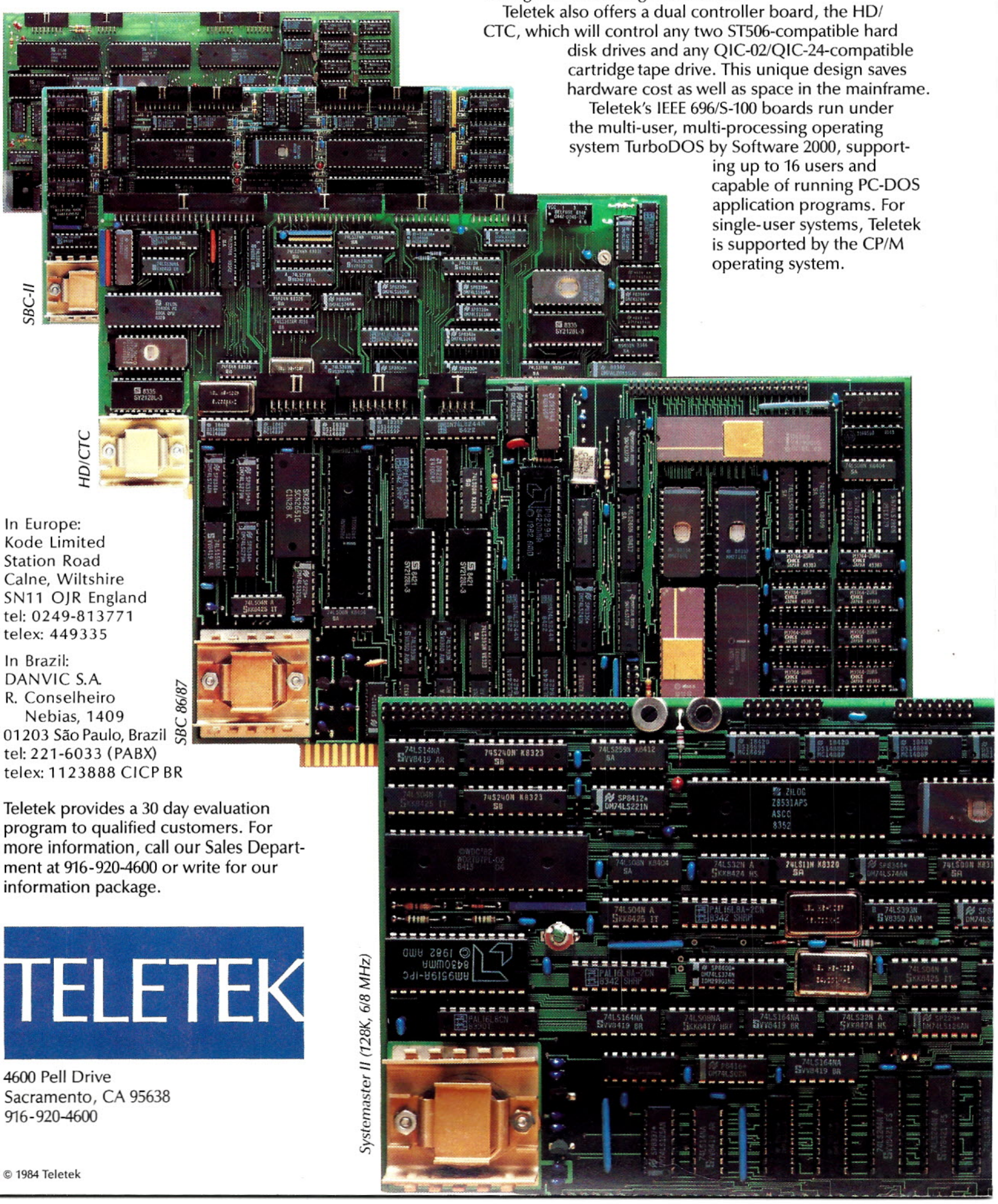
Since 1968 Teletek has been a leader in the design and manufacture of single board computers, controllers, memory boards and interface boards.

Teletek offers five distinct single board computers (SBCs), each with its own unique features, to meet the varied needs of the system integrator. Based on the 8086 16-bit and Z80 8-bit microprocessors, Teletek's SBCs will run at 4, 5, 6, or 8MHz and are available with up to 512K of onboard dynamic RAM. The SBC 86/87 also offers an optional 8087 math coprocessor for numeric intensive applications.

Teletek's Systemaster II provides two RS232C serial ports and two Centronics-compatible parallel ports or may be optionally configured to provide a SCSI interface or an IEEE-488 interface to support many laboratory testing and measuring instruments.

Teletek also offers a dual controller board, the HD/CTC, which will control any two ST506-compatible hard disk drives and any QIC-02/QIC-24-compatible cartridge tape drive. This unique design saves hardware cost as well as space in the mainframe.

Teletek's IEEE 696/S-100 boards run under the multi-user, multi-processing operating system TurboDOS by Software 2000, supporting up to 16 users and capable of running PC-DOS application programs. For single-user systems, Teletek is supported by the CP/M operating system.



In Europe:
Kode Limited
Station Road
Calne, Wiltshire
SN11 OJR England
tel: 0249-813771
telex: 449335

In Brazil:
DANVIC S.A.
R. Conselheiro
Nebias, 1409
01203 São Paulo, Brazil
tel: 221-6033 (PABX)
telex: 1123888 CICP BR

Teletek provides a 30 day evaluation program to qualified customers. For more information, call our Sales Department at 916-920-4600 or write for our information package.

TELETEK

4600 Pell Drive
Sacramento, CA 95638
916-920-4600