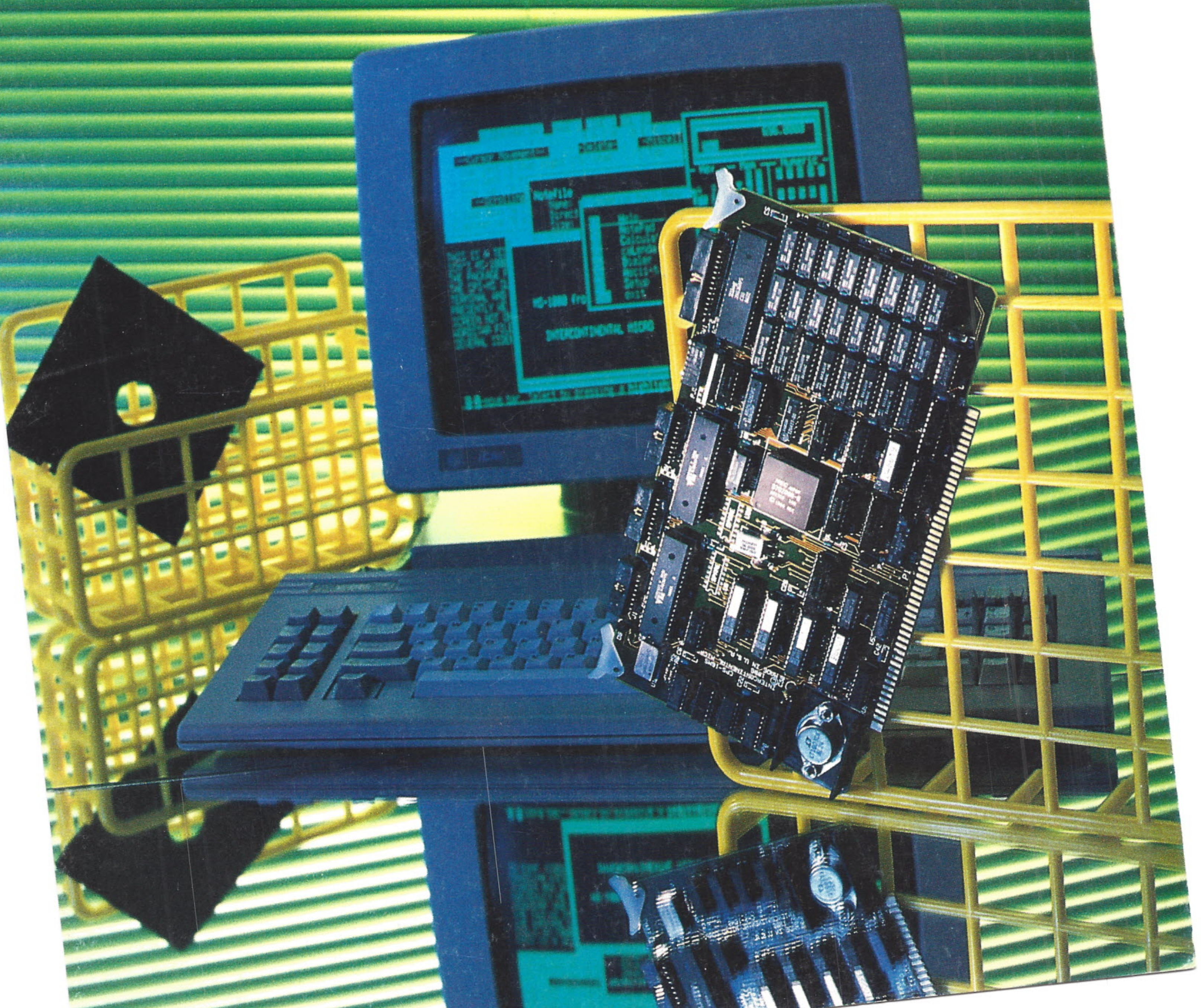
VOL. 1, NO. 6 • FALL 1987

$4.90

S-100 journal



Bus Specific Networks

DOS-IT-YOURSELF

At last, you can build PC-DOS compatible S-100 bus systems — only with ICM's new CPS-MS Slave Cards.

Only InterContinental Microsystems puts PC-DOS and MS-DOS capabilities on S-100. New compatibility with no LAN hookups. Now simply add the new CPS-MS Slave Card and a PC terminal to your S-100 bus system, and you've added a PC. For every CPS-MS card and terminal you add, you get an additional PC — with full connectivity under TurboDOS. Our MS/1000 CPS-MS package, including ICM's exclusive MS/1000 software, saves space, installs easier, reduces noise, reduces heat, reduces maintenance and repair costs, is easier to integrate and performs twice as fast as an IBM PC or XT.

You'll love the "low-cost expansion/add-on performance ratio" of ICM's new CPS-MS cards. They come as a total IBM-compatible package complete with 16-bit processor, M/STER PC-emulating terminal with ICM's monochrome graphics, PC-DOS, TurboDOS/PC, and ICM's exclusive MS/1000 software.

Call ICM for more information and reseller pricing.

IBM PC-AT, IBM PC-XT, IBM PC and PC-DOS, are registered trademarks of International Business Machines Corp. MS-DOS is a registered trademark of Microsoft Corp. TurboDOS/PC is a registered trademark of Software 2000 Corp.

PC, ARCNET, AND S-100 APPLICATIONS

ICM
TOMORROW'S BREAKTHROUGHS TODAY
InterContinental Microsystems
4015 Leaverton Court, Anaheim, CA 92807
Phone: (714) 630-0964
Telex: 821375 SUPPORT UD
Easylink: 62562040 BBS (714) 632-8750

S-100 journal

VOL. 1 NO. 6 FALL 1987

Editor and Designer
Jay Vilhena

Associate Editor
Robert Petersen

Technical Editor
Don Pannell

Circulation
Linda Vilhena

Advertising
Jay Vilhena

Production Assistants
John Petersen
Jonathan Swann

Contributing Artists
A. Pereira
G. N. Pereira

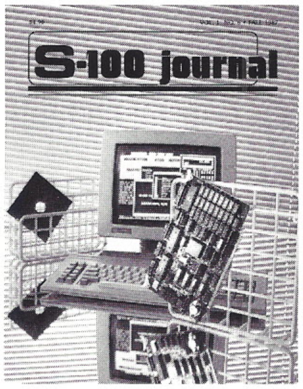
ISSN 0892-6190

S-100 Journal is published quarterly by Octoplus Corporation, 1275 N. University Ave., Unit 7, Provo, UT 84604. Telephone (801) 373-0696. Copyright©1987 by Octoplus Corporation. All rights reserved.
Subscriptions: U.S.A.: $14.00 for 1 year, $25.00 for 2 years. Canada and Mexico: add $3.00 per year. Other countries: add $6.00 per year. All foreign payments must be in U.S. currency. Send orders and payments to S-100 Journal, P.O. Box 1914, Orem, UT 84057.
Advertising: Contact Jay Vilhena at one of above addresses or call (801) 373-0696.
Postmaster: Send address changes to S-100 Journal, P.O. Box 1914, Orem, UT 84057.

outline

OUR COVER

ICM's MS-1000 package adds a PC-DOS workstation to an existing S-100 system. The package includes an S-100 slave board, terminal, and software. See page 85.



What you should know about the S-100 market & Advanced Digital... BEFORE you buy;

"With the new wave of outstanding S-100 products hitting the market, S-100 manufacturers continue to assert their dedication to superior quality and innovative design. For many years ahead, they will equip thousands of small businesses and professionals with the most versatile and upgradable microsystems in existence."

—Jay Vilhena, Editor
S-100 Journal

"At Advanced Digital Corporation our business philosophy can be stated in one word - "Pride." Pride in the quality and performance of the products we design and manufacture...Pride in the dedication, experience and performance of our employees...and Pride in our achievements and contributions to the S-100 marketplace."

—Hossein Asadi, President
Advanced Digital Corporation

1. Attractive Features. Attractive Prices.

Advanced Digital Corporation offers the most complete line of S-100 products with the latest in the state-of-the-art microcomputer technology. All of ADC's S-100 products are powerful high speed 8-bit and 16-bit single board computers. Your S-100 system can grow from ordinary to awesome with ADC's superb family of S-100 boards quickly, easily and cost-effectively.

2. Financial Stability

Founded in 1980, Advanced Digital Corporation is one of the most financially sound, privately held computer manufacturers. The company has consistently shown a profit in every quarter since 1980. It is the intention of ADC to continue as the industry leader and remain in the forefront of technology assuring you of the finest product and service for years to come.

3. Quality Assurance

Rigorous testing of each and every product by ADC's highly skilled technicians assures customers of trouble-free products. ADC prides itself with the strictest quality control in the industry. No company has higher quality control standards, or a better success rate than ADC's Every product is backed by a one year warranty.

4. Customer Service; 24 Hour Repair.

Highly-skilled technicians are assigned to customer support full time. In most cases, a problem can be solved over the phone in a few minutes. If not, repairs can be completed within 24 hours from receipt of product. Centralized repair service is available through ADC's Service Center.

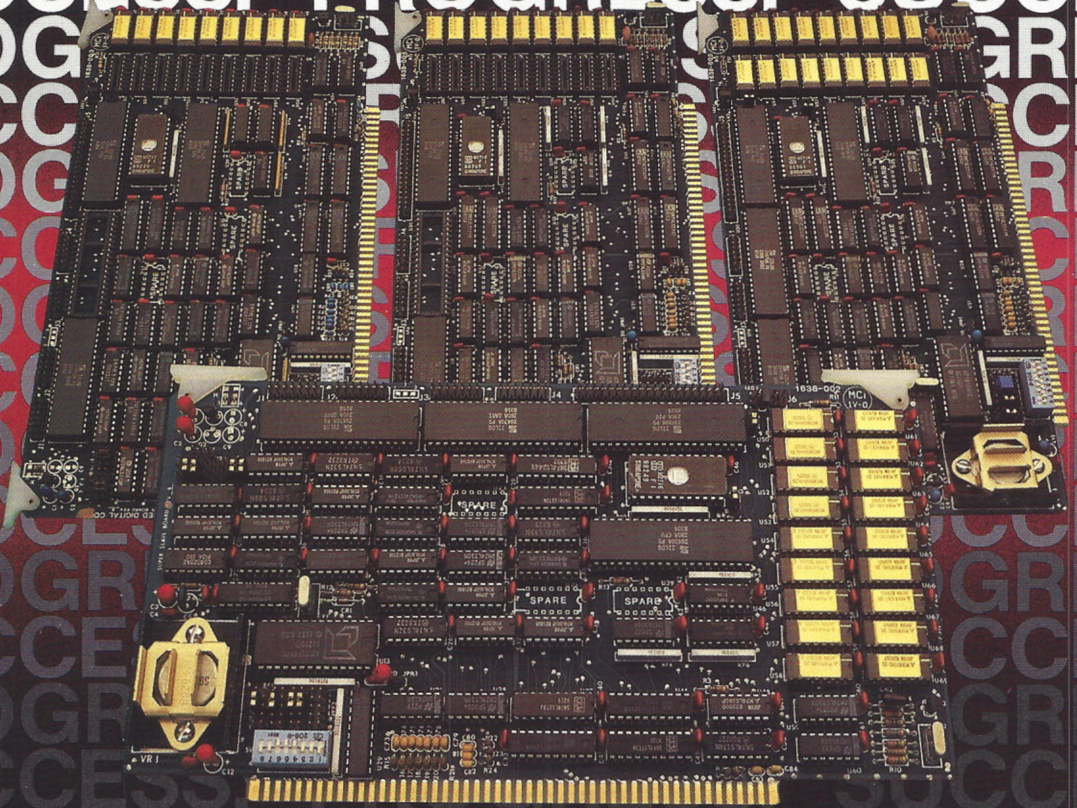
5. Market Penetration; Over 12,000 units installed.

With an installed base of over 12,000 units worldwide, ADC provides service to thousands of businesses and individual owners. By developing a superior line of S-100 products, ADC has become a major force in the industry. Customers include: US Navy, NASA, Harris Systems, Aerospace Corp.

6. Resources... Technical Expertise, Advanced Engineering.

ADC has assembled a team of highly experienced, industry renown specialists in engineering, manufacturing, marketing, sales and administration. ADC is a company of pioneers and innovators that have led the way and set the S-100 standards.

PROGRESS. SUCCESS. PROGRESS.
SUCCESS. PROGRESS. SUCCESS.



Founded in 1980, by a team of highly skilled and experienced engineers, Advanced Digital Corporation has grown to become a leading supplier of Single-Board Computers, Multi-User, Multi-Processor computer systems and memory subsystems for the S-100 Bus and PC Compatible Products marketplace. We're a company that has made tremendous growth without sacrificing the major factor that led to it. Quality...quality in our product, in our people, and in the way we do business.

Leading our success in the S-100 market, is a legacy of products that represent the needs of the S-100 market. Products such as Super System II, Super 16, Super 186 and Multi-Slave have made ADC the success it is today.

SUCCESS STORIES:

• SUPER SYSTEM II...A fully integrated system that runs under CP/M 2.2® for single-user applications or TurboDOS® for multi-user installations for both 8 and or 16-Bit including networking to PC's.

• SUPER 16...The cost effective 16-Bit 8 MHz Multi-User Single Board Slave processor.

• SUPER 186... The first high speed 8 MHz 16-Bit S-100 Single Board Computer.

• MULTI-SLAVE...A three user board contains three independent computers at 8 MHz and runs the TurboDOS operating system.

Progress. Success. It's the Story of our Company.

The success of ADC in the S-100 market is also due to the following line of quality products: • Super Star • Super Quad • Super Six • Super Slave • Master 9.

At ADC we're an aggressive company, a progressive company, an innovative company. To be successful, we know that each product we develop must be totally unique to the market.

Our progress and success is with products that have gained the highest ratings and the respect of the entire S-100 industry. Our success is based on superior technology, unique features, and the most cost competitive prices in the market. When you're looking for progress, look to ADC, because it's the story of our company.

ADC
ADVANCED DIGITAL CORPORATION

ADVANCED DIGITAL CORPORATION
5432 Production Drive • Huntington Beach, CA 92649
(714) 891-4004 • (800) 251-1801 • Telex 183210 ADVANCED HTBH

ADVANCED DIGITAL U.K. LTD.
27 Princes Street, Hanover Square • London W1R8NQ-United Kingdom
(01) 409-0077 • (01) 409-3351 • TLX 265840 FINEST

© TurbosDOS is a trademark of Software 2000. ® CP/M 2.2 is a trademark of Digital Research Corporation.

SEE US AT FALL/COMDEX 86 • BOOTH 1480

EARN YOUR WINGS AND FLY!

It is a common plot in science fiction. At some point in time and space, a soft spot or an oeil-de-boeuf appears in the fabric of the universe through which one can escape into new worlds and dimensions.

The microcomputer industry now staggers in such a plot. Billions of dollars of research, development, and production lay wasted at the dead-end as IBM obsoletes the PC bus overnight. 'Come ye, oh faithful souls, discard thy earlier possessions and follow thy mighty leader for new extravagant ways of wasting your money are at hand.' While the industry vaccilates at the change of tides, the opportunity is here for the S-100 bus to sneak in and conquer a wider base. We certainly have the excuse, the performance, and the modularity. Now, how about building the resources?

We have warned you before. Outside our realm, buy and discard is the name of the game. The S-100 bus remains unique in its arrogant defiance of the word obsolescence. Yes! We can match the performance of IBM's new systems, all the way up to the $11K Model 80, with our traditional board-swap trick. And it's not going to require 'no' $11,000 either. Would you believe $5,000? In fact, to let you in on a secret, except for a few fancy graphics, there is very little in IBM's new top-of-the-line Model 80 (which, by the way, is not even available yet) that does not already exist in the S-100 bus. Wow! They are finally able to address 16 Megabytes of memory. So, friends, how many years is it now since the S-100 bus has been able to address 16 Megs? Where is the 4-Gigabyte address space of their 386 system? Would you also believe that there are now, already built, S-100 boards that can directly address 4 Gigabytes?

Then, how are we going to get our traditional S-100 companies to produce and bring to market a plethora of S-100 products to show the world what our systems can do? Will they see the window in the fabric and come through? Some, no doubt, will continue to produce outstanding S-100 products. But many have been diddling, still trying to sell 5-year old products, wasting time and engineering resources with piddly PC bus products, or becoming distracted by the VMEbus market. For those who come, we will run along their side. We'll continue to open the pages of *S-100 Journal* to their ads, we'll continue to discuss and review their products, and **WE** (the S-100 users and integrators) will continue to buy from them because we refuse to trade our S-100 systems for the inferior junk that dominates the microcomputer industry.

But we want more action. We want boards to interface to today's state-of-the-art peripherals. We want 5- and 10-Megabyte memory boards. We want 50-MHz 32-bit CPU boards. We want top-quality attractive tower enclosures. We want super-high resolution graphics. We want new boards and software drivers that do not require an engineering degree to integrate into our systems. We want full compatibility with current software standards and beyond. We want more, many more, slave CPU boards, and we want them to allow us to run many — not just two — operating systems in the same box, at the same time, at the flick of a software switch. There is no system in the world, from IBM or anyone else, that could win an argument with a 386-based S-100 bus running under C-DOS and containing a 286 slave running OS/2, a 68020 slave running UNIX or d/OS, a Z280 slave running the Z-System, and a graphics board, all conveniently accessible from the same keyboards. It is all possible with the S-100 bus.

Should the action not come, we are not helpless. As you look through this issue, you will find several ads from companies that a year ago you never knew existed. As older S-100 companies leave the market, new ones are arriving to replace them. They are offering new vigor to our old reliable bus. But we need more companies, more clever engineers designing tomorrow's S-100 boards in a corner of their basements, more dealers willing to make larger profits selling mature, professional S-100 systems and components, more repair shops that accept many different brands of S-100 boards.

If you have the ability, go design new S-100 products and start companies. *S-100 Journal* has grown into a powerful instrument through which we can communicate and let new products be known. More than 10,000 readers consult our magazine, and still we are growing. *S-100 Journal* will assist you in letting our readers know that your products exist. We'll get you into the market with no marketing investment. We'll review your S-100 products, help you design ads for free, and offer you extended credit in advertisements that you place.

Go design and produce those boards. We are ready to buy them because we are a special group of intelligent users. We look for quality, wish to maintain our computing freedom, and refuse to be the peons that succumb to the leader's whims.

Jay Vilhena

The Peak of Performance

SCALE THE HEIGHTS OF PRODUCTIVITY

Sure, you've proven that in your hands a computer is a productive tool. But if you haven't teamed up with a SemiDisk you have heights yet to climb!

IT'S NO MERE RAMDISK

SemiDisk has been leading the way for Disk Emulators since their inception. If you've seen RAMdisks you know what it's like to load programs in an instant, and read or write files without delay. Unlike alternatives, the SemiDisk offers up to 8 megabytes of instant-access storage while leaving your computer's main memory free for what it does best - computing!

KEEP A GRIP ON DATA

Go ahead, turn off your computer. Take a vacation. With the battery backup option, your valuable data will be there in the morning even if you aren't. You'll sleep better knowing not even a 5 hour blackout will sabotage your files.

NEW LOWER SEMIDISK PRICES THAT WON'T SNOW YOU UNDER

	512K	2Mbyte
IBM PC, XT, AT	$495	$995
Epson QX-10	$595	$995
S-100, SemiDisk II	$799	$1295
S-100, SemiDisk I	$595	——
TRS-80 II, 12, 16	$695	$1295
Battery Backup Unit	$130	$130

Software drivers available for CP/M 80, MS-DOS, ZDOS, TurboDOS, and VALDOCS 2.

SEMIDISK

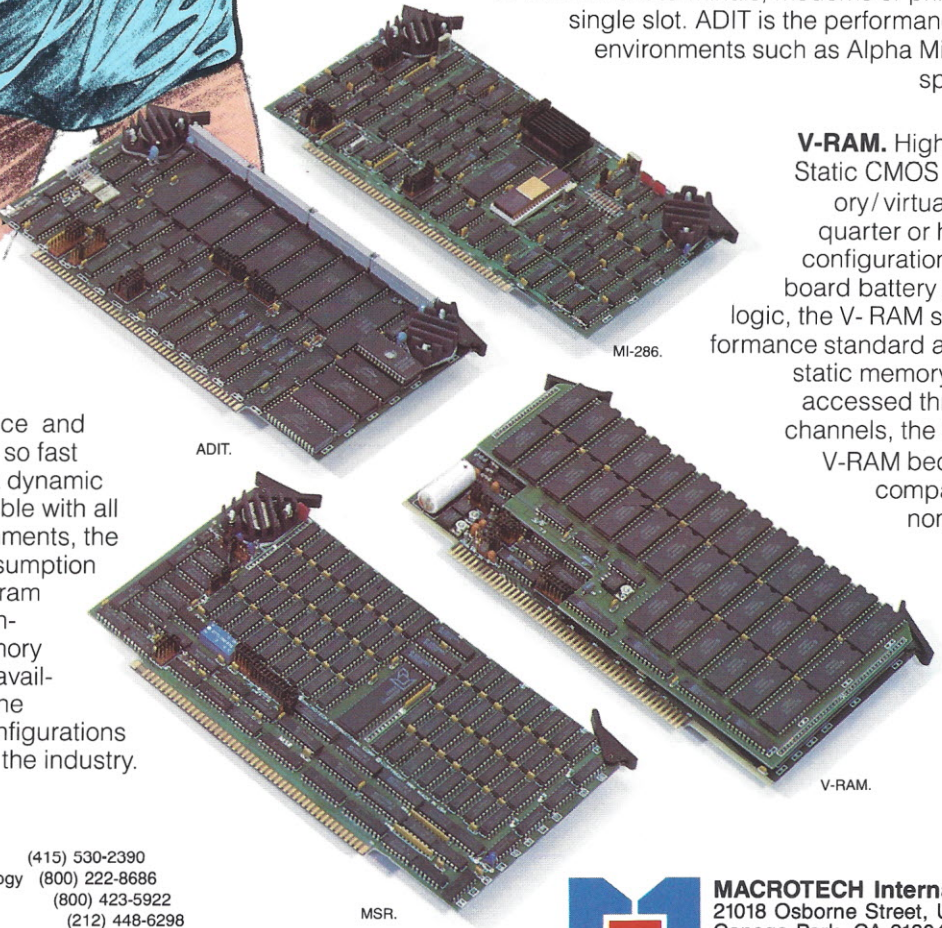
SemiDisk Systems, Inc.
P.O. Box GG, Beaverton, Oregon 97075
503-626-3104

Call 503-646-5510 for CBBS/NW, and 503-649-8327 for CBBS/Aloha, all SemiDisk equipped computer bulletin boards, (300/1200/2400 baud) SemiDisk, SemiSpool trademarks of SemiDisk Systems.

MACROTECH—STILL THE S-100 PERFORMANCE PACESETTER

MI-286. Our 80286/Z80H Dual CPU Board is at least twice as fast as Compupro's 8085/88 and it's a direct replacement. The MI-286 has already become the standard by which other 80286 based systems are measured. Ask us for a complimentary Benchmark Report.

ADIT. There's nothing else like it on the market. It's an Intelligent I/O Board with its own real time firmware that lets you control up to 16 different terminals, modems or printers all from a single slot. ADIT is the performance standard in environments such as Alpha Micro where I/O speed is critical.



V-RAM. High performance Static CMOS system memory/virtual disk in either quarter or half megabyte configurations. With its on-board battery and power-fail logic, the V-RAM sets a new performance standard at conventional static memory prices. When accessed through I/O port channels, the half megabyte V-RAM becomes M Drive compatible with true non-volatile solid-state disk capability.

MSR. High performance and reliability in a memory so fast you won't believe it's a dynamic ram product. Compatible with all popular S-100 environments, the MSR's low power consumption and 120 nanosecond ram devices set a new standard for dynamic memory products. The MSR is available in quarter, half, one and two megabyte configurations at the lowest prices in the industry.

Dealers:

Vision Computers	(415) 530-2390
Compliance Computer Technology	(800) 222-8686
S-100	(800) 423-5922
John D. Owens & Associates	(212) 448-6298
In England; Fulcrum (Europe) Ltd.	(0621) 828763

Macrotech dealers also include most Compupro Systems Centers, Heathkit Electronic Centers and Alpha Micro Dealers.

MACROTECH International Corp.
21018 Osborne Street, Unit #5
Canoga Park, CA 91304
(800) 824-3181 • in Calif. (818) 700-1501
Telex: 9109970653

editor interface

Looks like even with the increase of pages in this issue, this time I will have little more than one page for publishing and answering letters.

Please continue sending in your comments. They are very helpful. Send your letters to Editor, S-100 Journal, PO Box 1914, Orem, UT 84057. Thanks.

It makes me very happy to see a publication which supports the S-100 bus. I would like to make a few suggestions which may help:

1. An advertiser index in the back of the magazine which lists all the current advertisers alphabetically and the page their ad is on.
2. Add, at the end of the article, a box with the name, address, and price of any vendor and product discussed in columns or articles.
3. Maintain a readership separation between computer users and technical people. A user wants to know more about available programs and devices to better utilize his machine, but may not be interested in knowing what is inside the box.

Glen A. Spooner
Orange, California

Advertiser index: You got it! See page 90.

Name and address box: I admit that we do not follow the rule 100%, but as you can see in several articles in this issue, we try to do as you say whenever it is appropriate.

Readership: We are making an effort to offer a good balance of user-oriented and more technical and integrator-oriented articles. The two user-oriented columns introduced in this issue are a step in that direction.

Thanks for your letter. •Jay

I am letting my subscription drop because I am not particularly interested in bus systems at this time and I subscribe to too many magazines.

I do have some comments, however, which may be of interest.

I am one of those people who buy IBM (or equivalent) PCs. My background in this matter is that I was quite active in the CP/M world, and I dislike and distrust IBM in the extreme. Why then buy IBM or clones? The answer is simple — they are cheap, the software and plug-in card availability is tremendous, and they are universally accepted.

If you can't get broad-based commercial support, the S-100 bus is dead, dead, dead. Which maybe is as it should be. RIP S-100!

Ted Parker
Camarillo, California

Opinions such as those you express have been widely publicized since the early 80's. In fact, around 1983 they appeared in practically every computer industry magazine. Claims about the death of the S-100 bus have been around for over 7 years, most likely originally spread by those who had a financial interest in selling other types of systems. It is now 1987. Take a look at the nearly-100 pages of this magazine. Does it look like the magazine of a dead bus?

IBM-PCs, Toyotas, and Datsuns will certainly sell the most, but there will always be a market for those who prefer Mercedes. •Jay

I'm approving of your magazine in the only significant way possible — I'm enclosing my check.

I am not a 'member' of the S-100 user community — yet. And I am puzzled why the S-100 bus, which was, as I understand, very nearly **the** standard at the time of IBM's introduction of the PC, is so poorly known, and why it seems to be doomed to stay that way. As a scientist turned programmer, and finding myself in the position to be specifying hardware for my customers, it is extremely important for me to know my options. I feel that your publication is an important tool, but I am quite dismayed over the lack of basic books on the subject of S-100 systems.

You may appreciate the grim realities of this story: Today I was visiting with a colleague who requires real-time A-to-D for data collection, mainly to ask him if he knew of any competent S-100 people in town. (He didn't.) He opened a cabinet and showed me some boards, and went on for a bit about what excellent devices they were. The reason why they were in the cabinet was that they are S-100 cards, and the standard in his lab is now, for a variety of reasons, the PC/AT bus. It is certainly hard to argue against PCs when the town is flooded with clone houses, and one can't even find a decent *book* on S-100.

I look forward to receiving your publication, and as I grope my way towards some decisions concerning S-100 equipment, you will probably hear from me again.

James W. Albert
San Antonio, Texas

Your predicament is not unique. Today's knowledgeable microcomputer buyer faces a tough decision. On one hand you have the loud fanfare of an industry gone awry and whose attributes are good marketing, popularity, and quick obsolescence — supplier-advantage. On the other hand, you have systems like the S-100 bus that, although being poorly known, feature stability, modularity, upgradability, and non-obsolescence — user-advantage.

In my opinion, today's computer industry has nothing to do with computers: it's a marketing stage. Business magazines disguised as computer magazines proliferate, and true computers are little mentioned, so your

(continued on page 90)

INTRODUCTION TO CONCURRENT DOS

Thomas Bartkus

Multiuser OS *is a regular column that discusses multiuser operating systems currently running on S-100 microcomputers. The column discusses such operating systems as Concurrent DOS, TurboDOS, UNIX, d/OS, AMOS/L, OS-9, THEOS, etc.*

Each issue features one or more operating systems in articles written by one of a group of columnists who are experts on the particular OS they write about.

Readers are invited to submit questions they may have about any of the operating systems discussed. Answers to questions will be incorporated into future issues as space permits. Please send any questions or comments about this column to Multiuser OS, *S-100 Journal, PO Box 1914, Orem, UT 84057. Letters should be typewritten, and each letter must deal only with one OS in order to facilitate forwarding to the proper columnist.*

We still have available a few columnist positions for Multiuser OS. *If you have a solid knowledge about a multiuser operating system, you have access to a version of that OS running on an S-100, and you would like to become a columnist, please contact S-100 Journal.*

In this issue, the operating system featured is Concurrent DOS. Thomas Bartkus is a programmer and technical consultant. He specializes in microcomputer applications for industrial process control, chemistry, and engineering. Thomas can be contacted directly through CompuServe [76010,2045].

It is often said (with considerable justification) that the CP/M operating system launched the microcomputer revolution. The CP/M designation stands for 'Control Program/Microcomputers' and was introduced in 1975 by its inventor, Gary Kildal. It was the first Disk Operating System (DOS) available for the then-infant microcomputers. Designed to execute on an 8-bit CPU chip, the Intel 8080, these first CP/M systems had a 100-pin bus structure called 'S-100.' The S-100 bus became a sophisticated standard for developers of microcomputer hardware. In the same manner, the original CP/M became a standard for developers of software.

Gary Kildal used his invention to found Digital Research, Inc. and proceeds from his original DOS still contribute handsomely to that companies revenues. Time marches on however, and although CP/M and its variant operating systems are still standard in the ever-vigorous 8-bit world, 16-bit processors are now a practical reality for our S-100 computer systems.

Both hardware and software had to be upgraded accordingly. Digital Research's response was CP/M-86. Designed to run under the Intel 8088/8086 line of CPUs, this operating system provided an upgrade that was familiar to anyone who ever sat at the console of a CP/M machine. It was, in fact, an 8-bit operating system reworked to fit the 16-bit world. Like MS-DOS, its rival from Microsoft, it is a single-user, single-tasking operating system.

ENTER CONCURRENT DOS

In 1983, Digital Research introduced its multitasking operating system under the name of Concurrent CP/M. This was a brand new operating system designed to take maximum advantage of the 16-bit Intel chips. Digital Research apparently has been having difficulty deciding what to call

ABOUT MULTITASKING

An operating system's ability to perform many tasks at the same time is called *multitasking*. This advantage is used in several ways. The most obvious is to hook several consoles to the system and have the operating system service the users at different terminals. This is a multiuser system. Less obvious is the ability of a single user to initiate multiple tasks that run simultaneously at a single console. This is single-user multitasking. A multiuser system may or may not allow a user to perform single-user multitasking at any individual console. There are Concurrent DOS implementations that encompass each or all of these situations. In this article, I use multitasking in the largest sense, i.e., it means multiuser, single-user, or the combination of both.

their new OS and, as a result, there is some confusion over the nomenclature. The following have all been used to refer to the same operating system:

Concurrent CP/M
Concurrent CP/M-86
CCP/M
Concurrent PC DOS (IBM PC implementations only)
Concurrent DOS
CDOS
Concurrent DOS 86

Digital Research now seems determined to banish the CP/M designation from the name. It is an unfortunate marketing reality that CP/M now seems to impart a false impression of 'old' and 'obsolete' to a computer product. From here on, I will just use the name Concurrent DOS or just Concurrent for short.

WHAT CONCURRENT DOS LOOKS LIKE

Concurrent DOS essentially comes from DRI with a dual user interface. One is identical to CP/M-86 and to the latest version of the 8-bit CP/M (CP/M Plus). The other is the user interface of Microsoft Corporation's MS-DOS operating system. The two systems were never very different since the roots of MS-DOS lie firmly in soil created by CP/M. MS-DOS, however, was on the scene first when the 16-bit 8088/8086 Intel CPU became available. Software vendors, encouraged by a flood of new microcomputers using MS-DOS and MS-DOS-like operating systems, rushed to market with programs written to that operating system's specifications. Faced with a smaller software base, the late-coming CP/M-86 was never to gain a large following in the 16-bit Intel world. Microsoft continued to develop MS-DOS, taking many UNIX features such as tree-structured directories and I/O redirection, and the two operating systems continued to diverge.

In an attempt to overcome the popularity and software handicaps, Digital Research has incorporated increasing MS-DOS compatibility into Concurrent DOS with each new revision. Concurrent will execute programs written for CP/M-86 (.CMD files) or for MS-DOS (.COM and .EXE files). Advocates of CP/M may PIP files from drive to drive while MS-DOS supporters can COPY them. Concurrent handles it all with equal facility. Whether Concurrent looks like CP/M or MS-DOS depends upon the preference of the person at the console.

DOING IT CONCURRENTLY

While both MS-DOS and CP/M-86 are adaptations of older operating systems to the newer 16-bit Intel chips, Concurrent DOS is a completely new design. Concurrent DOS is a true multitasking system designed from the ground up, taking advantage of the new capabilities these CPU chips were designed for. As already mentioned, Concurrent runs programs written to both MS-DOS and CP/M-86 environments. Indeed, Concurrent is uniquely adapted to provide multitask capability to programs that were never designed for multitasking environments.

The most obvious change from single-tasking CP/M-86 is the addition of a control-key sequence to switch virtual consoles. Virtual consoles may be described by the following analogy: An operator sits on a swivel chair surrounded by four or more computer terminals. At the first terminal, he initiates a modem upload

SYSTEM FUNCTIONS

■ Console attach and detach facilities. Programs need not hold the console resource unnecessarily. If a program has no need for or is finished with screen and keyboard I/O, it can simply release the console for other work. Provisions for a program to regain access to the computer terminal are available if required.

■ A queue system for interprocess communications. This is a means for passing messages and data between programs. It can be used to synchronize execution between programs and prevent collisions when programs compete for system resources.

■ A priority system that allows the programmer to control how much of the CPU a program may get.

■ Shared file support with both file and record locking. Several users or programs can read or manipulate the same file simultaneously. The programmer determines the degree of access to a file under simultaneous use.

■ Shared code support to conserve memory. A program can (and should) be written so that multiple copies can execute with only one copy of the program code loaded into memory. This is an important consideration with multiuser systems in general and with the limited address space of the 8088/8086 CPU's in particular.

■ A delay function to effectively suspend program execution when a delay is needed. It eliminates the need to tie up data processing resources to do NOPs or other useless operations merely to kill time.

■ The ability for a program to spawn or initiate another program that will execute concurrently. If, for example, a program must drive a printer or plotter, it need not wait on these slow devices, it can simply spin off the plot routine as a separate process and continue on its merry way.

Table 1. *Special functions available under Concurrent DOS.*

COMMAND FILE FORMATS

Concurrent DOS command files consist of two parts. These are the *base page* followed by the program code. The base page contains data that the OS needs to successfully execute the program code, such as the location in memory where to load the program, how to initialize the CPU segment registers, and how much stack space is needed. Both MS-DOS and Concurrent DOS follow this basic structure with the first 256 bytes in the command file making up the base page. However, the actual data and how it is presented differs between the two systems. In the case of Concurrent, the file extension .CMD indicates to the operating system that it is a CP/M-style command file so it may load and execute accordingly. In the case of MS-DOS, two different command file structures apply. One is the small-model single-code segment structure indicated by the .COM file extension. The other is the large-model multisegment structure indicated by the .EXE extension.

While Concurrent DOS recognizes all three command file types, MS-DOS programs often give insufficient information for efficient execution under multitasking. A common example would be a program that tries to seize the entire memory space of the computer even though only a fraction of this memory is really needed for successful execution. This is quite acceptable in a single-tasking environment. Such needless extravagant behavior, however, would prevent the person at the next terminal of a multiuser system from having any computer memory to work with. The good news are that Concurrent has the ability to restrict the amount of memory these programs see, enforcing a more economic use of computer resources under multitasking.

of a lengthy file. Rather than stare at the screen for this 20-minute process to complete, he simply swivels around to the next console and starts a tape backup of last week's accounts. Still not ready for his coffee break, he swivels around to the next free console and starts typing a report on a word processor. At any time, he can simply turn around to a ready console or look at the output of a program in progress. Virtual consoles implement this analogy on a single terminal. You can switch in and out of virtual consoles and the display will change just as if you were sitting on that swivel chair. It is like having several computer terminals at your disposal. Typically there are four virtual consoles, but there may be more in any particular implementation.

All ordinary I/O calls properly made to either the MS-DOS or CP/M-86 operating systems work under Concurrent. In addition, Concurrent DOS offers a rich set of operating system functions to enhance multitasking. These functions facilitate multiuser applications and promote efficient use of the CPU and other system resources under multitasking. Some of the capabilities provided are described in Table 1.

THE FILE SYSTEM

Concurrent supports two file systems. One is the familiar user area system as in CP/M-86. Here each drive is divided into sixteen (0-15) logical user areas, each with its own directory. Date and time stamping are an option. Also, multilevel password protection is available at both the file and the drive level. The other file system is the tree-structured directory system similar to MS-DOS. Date and time stamping are mandatory here, and there is no password protection.

Hard drives may be formatted either way or they may contain two partitions with one being defined as CP/M media and the other as tree-structured DOS media. Interestingly enough, Concurrent doesn't care about the mix of formats on the system. The type of formatting on a floppy drive is automatically recognized as soon as it is accessed. Programs of any type (.COM, .EXE, .CMD) can be stored, retrieved, and executed from either format. Data files, likewise, don't care what type of media they are on. The only caveats here are that the tree-structured directories are unavailable on CP/M media and the password protection is unavailable on DOS media.

File attributes of *read-only*, *archive*, and *system* are supported regardless of media type. *Read-only*, of course, forbids the alteration or deletion of a file so marked. The *archive* attribute acts as a flag to detect file modification. The user will set the archive attribute. If the file is subsequently modified or edited, the archive flag is automatically reset. This makes it easy to detect modified files for backup purposes. The *system* attribute hides the file or prevents it from being shown in a normal directory listing unless specifically requested by the user. Also, a system file on the system drive will always be found regardless of the directory the user is logged into. The user has the ability to decide what drive or directory is defined as the system disk and may change this at will.

COMMANDS AND UTILITIES

This is rather difficult to describe because it can vary widely between different implementations. Nevertheless, a few items ought to be discussed. The built-in system commands are usually few in number; the strategy is to implement most commands as external programs. Thus, many seemingly integral commands such as 'DIR' are often (not always) implemented as an external DIR.CMD file. This has the advantage of making the system easier to customize and the disadvantage of poor response when using slower media such as floppy disks. Those interested may buy the customization tools that DRI provides and thus may implement any program as a Resident System Process (RSP). This makes the program a part of the OS without the need for an external command file.

Included are commands to manipulate files and maneuver through both tree-structured directories and CP/M-86-style user areas. Also, there are general multitasking commands to display and abort background tasks. Special attention is given to commands that allow MS-DOS and CP/M-86 programs to execute concurrently. These allow the user to compensate for the lack of essential base-page information needed for the multitasking environment.

An excellent on-line HELP facility is always included. Text describing all system and transient commands with attendant examples and options is included with every implementation of Concurrent. It comes with a description of all commands and programs existent on the system. The user may edit a HELP.HLP file to include information on programs he or she adds. Anyone on the system simply types *HELP {item}* and a description with examples will appear on the screen.

Also always included is a printer manager that allows many programs (or users) to utilize a single printer by spooling each print job and sending it to the printer as it becomes available. And, of course, all systems come with utilities to format and maintain disk storage.

Programs that you may or may not get, depending on the peculiarities of your particular hardware, are mentioned briefly here:

A *text editor*, ED or DR EDIX.

FILE MANAGER. A point-and-shoot system for manipulating files.

A *windowing* system. It allows the user to divide a single screen into separate windows, permitting simultaneous viewing of several virtual consoles on a single screen.

CARDFILE, a simple *data base* program for storing and retrieving names, addresses, and phone numbers. Provides a nice demonstration of programming with the window system.

A *menu* system. Actually a customizeable, menu-driven, shell around the OS. Very useful for preparing applications for more casual computer users.

FOR PROGRAMMERS AND SYSTEM INTEGRATORS

One of the nice things about the old days is that when one purchased CP/M for his system, he received a complete (if somewhat primitive) set of programming tools that enabled him to enhance, customize, and program his system. These usually consisted of an assembler, a debugger, and the source code to the BIOS. Many a brave soul actually wrote his own BIOS from a skeleton provided by Digital Research in order to implement CP/M on systems that were often homemade. The complexity of newer hardware and software is such that few would attempt this today. As a consequence, one now has to make a separate purchase of these formerly free programming tools. The software interface to a system is now called the XIOS for 'Extended Input/Output System,' and this is what a computer manufacturer will write in order to implement Concurrent DOS on his hardware. When you buy a *Programmers Pack* with your system, you will get an assembler and debugger along with the following goodies:

1. GENCMD.CMD — creates a command file from the Intel format hexadecimal file created by the assembler.
2. GENCCPM — generates the operating system as a CCPM.SYS file. Allows the programmer a high degree of system customization.
3. XIOS source code — this is the software interface to the hardware and will normally be written by the computer manufacturer. The programmer can modify this and further customize Concurrent to his system.
4. Complete documentation of the Concurrent DOS operating system at the programmers level.

WHAT IS IT WITH ALL THIS MS-DOS COMPATIBILITY ANYWAY?

Whatever stand one takes in the battle of the operating systems, it has to be said that MS-DOS is **the** operating system for Intel 8088/8086/80286-based micro computers. I have neither the space nor, perhaps, the daring to deal with the reasons for this here. Suffice it to say that most (not all) new program development for Intel CPU- based computers is targeted for MS-DOS and its variants. Digital Research quite understandably takes the position that it needs this software base in order to gain acceptance for its operating system.

There are drawbacks, however. One is that Concurrent DOS revisions are always chasing after MS-DOS revisions. As MS-DOS adds/changes features and functions, Concurrent is in the rather unenviable position of always playing catch up. Alas, despite Digital's best efforts, compatibility with MS-DOS will never be perfect. Fortunately, software vendors typically write to be downward compatible with earlier MS-DOS versions and, therefore, don't often use the latest features that Concurrent may lack.

All this points to another problem, the perceived lack of Concurrent's own identity. Instead of using the rich multitasking environment that Concurrent offers, most vendors

EXECUTING MS-DOS PROGRAMS

How does Concurrent DOS execute .COM and .EXE programs? To be sure, Concurrent DOS is an entirely different operating system. As such, it has a very different I/O structure and different protocols for calling operating system functions. When the name of a .COM or .EXE program is called for execution, an 'MS-DOS emulator' is automatically used to intercept MS-DOS system calls. These are then translated into Concurrent system calls. The happy result is that a mix of all programs may execute concurrently on the system.

produce software designed around the presumption that multitasking for microcomputers has yet to be invented. Such software often makes extravagant and unnecessary use of system resources, rendering them bad neighbors in multiuser/multitask environments. Concurrent does have a remarkable ability to compensate for such lack of forethought. The process is not automatic, however, and the system integrator or user often has to perform considerable experimentation and adjustment to get these programs to work properly.

WHY CONCURRENT DOS?

I trust that readers of this column are well acquainted with the benefits that multitasking provides. Multitasking on high-performance Intel 80X86-based S-100 systems is here right now, and this should be recommendation enough. Concurrent also fits comfortably into the networking world with DR-NET from Digital Research. Other S-100 computers using Concurrent DOS have been integrated with ARCNET, TurboDOS and other networking systems.

People who use Concurrent have two very effective sources of technical help. One is CONUG the Concurrent Users Group that puts out a fine newsletter (*Resource*) and maintains a bulletin board system. It is for users of Concurrent, regardless of the machines they use. The other is the Special Interest Group (SIG) on CompuServe Information Services, sponsored by Digital Research, Inc. Here you will find free and enthusiastic help offered by other technically knowledgeable users as well as Digital Research experts.

CONCURRENT AND ME

When the need to upgrade came, I was using my Z80-based S-100 system to develop applications for small companies. I think that many people will readily perceive that much of the pressure to adopt the Intel 80X86 microprocessors was due to the overwhelming adoption of the IBM-PC in the business community. I had to target these machines with software that I wrote. It was clear that I could not do this with my trusty Z80. Like many people in this predicament, I was quite unimpressed with these new machines because they offered no performance improvements over my old hardware. This made the expenditure of hard cash to change over rather hard to accept.

My solution, of course, was to exercise the S-100 board-swap option. I replaced my Z80 board with an 80186 system from a manufacturer that offered Concurrent DOS. Not only did I gain a real performance boost, but I can develop and port my software to any machine using the Intel chips. By using Concurrent DOS, I get the obvious benefits of multitasking for a superb program development system. I also get the very practical advantage of being able to write, compile, and debug programs for the unenlightened PC world. Of course, my 'PC' is still a high-performance S-100 system. I find that to be the biggest advantage of all. ■

SOME S-100 COMPANIES THAT CONFIGURE CONCURRENT DOS TO THEIR COMPUTERS

L/F Technologies
2800 Lockheed Way
Carson City, NV 89701
(702) 883-7611

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

Macrotech International Corp.
21018 Osborne Street, Unit #5
Canoga Park, CA 91304
(800) 824-3181 / (818) 700-1501

Tarbell Electronics
1082 E. Artesia Blvd., Suite C
Long Beach, CA 90805
(213) 422-7081

Viasyn Corporation (CompuPro)
26538 Danti Court
Hayward, CA 94545-3999
(415) 786-0909

TECHNICAL HELP FOR CONCURRENT DOS USERS

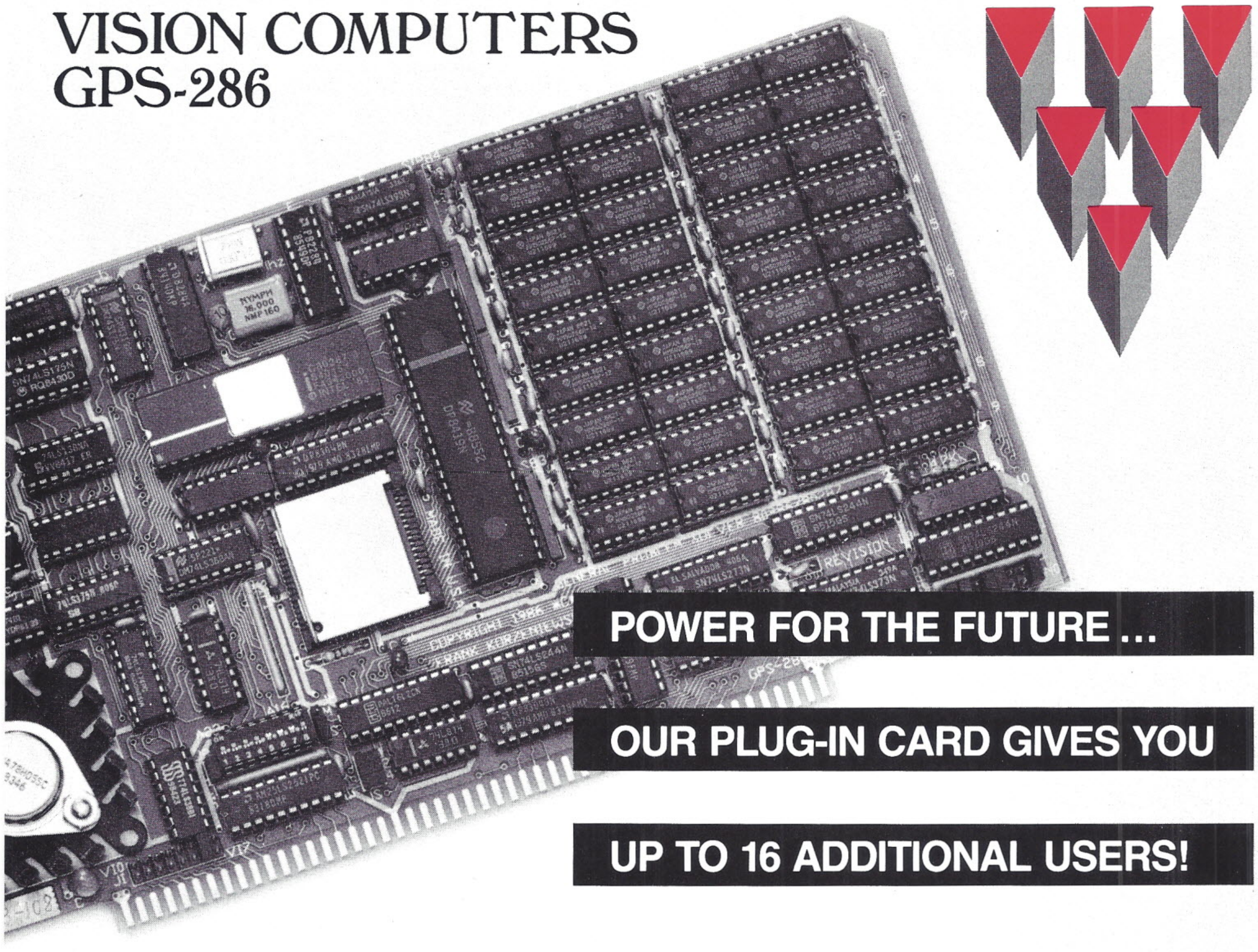
CONUG
PO Box 734
Marina, CA 93933
(408) 384-5575 (Modem)
(408) 384-6797 (Voice)

CONUG was established to support the growing community of users of Digital Research's Concurrent line of operating systems. Membership in CONUG is $25 per year in the US and Canada, and $35 elsewhere. CONUG maintains a software library on the CONUG BBS that contains many Concurrent utilities and programming examples. A list of the files in the library is available upon request.

CompuServe
5000 Arlington Center Blvd.
Columbus, OH 43220
(800) 848-8199

Digital Research, Inc. hosts a special interest group on CompuServe (GO DRI).
Thomas Bartkus, the author of this article, can be contacted through CompuServe number 76010,2045.

VISION COMPUTERS GPS-286



POWER FOR THE FUTURE ...

OUR PLUG-IN CARD GIVES YOU

UP TO 16 ADDITIONAL USERS!

Mainframe Power!

Presenting our GPS-286 package which can quickly turn your CompuPro S-100 computer system into a million dollar mainframe. With a 8 MHZ 80286 and a megabyte of RAM, the GPS-286 opens up a whole new world of business computing that is unconstrained by CPU speed or memory capacity. It keeps you working instead of waiting!

Why You Need GPS-286

The GPS-286 was designed for more users and faster performance. A single user can take advantage of a megabyte of RAM along with a dedicated, no-wait, 10 MHZ 80286, or up to 16 users can share. The GPS-286 satellite is a master at efficient resource allocation, and adapts instantly to your changing system requirements.
All users connected to the system share floppies, hard disks, and solid state disks, as well as printers, and modems — without the overhead of a network. This means ultra high performance and reliability at a rock bottom price.

Fits Your Applications

Just imagine 15 users running WordStar or NewWord, or 11 users using Medical Manager or MC Accounting software, or how about 6 Dataflex applications. All at the same time. Need more applications? Choose from any of the hundreds of other Concurrent DOS applications. Add a second or third GPS-286 and you've got 12 or 18 Dataflex users! Since you can have up to eight cards in a system, the skys the limit. That means lots of applications for a lot of users.

Easy Upgrade

You can install the GPS-286 in ten minutes. No new operating system, no new commands and no complications of a network. Both the CompuPro and Gifford operating systems are supported, including external networks to other CompuPro based systems and IBM PCs, XTs, ATs and compatibles.

Your Choice

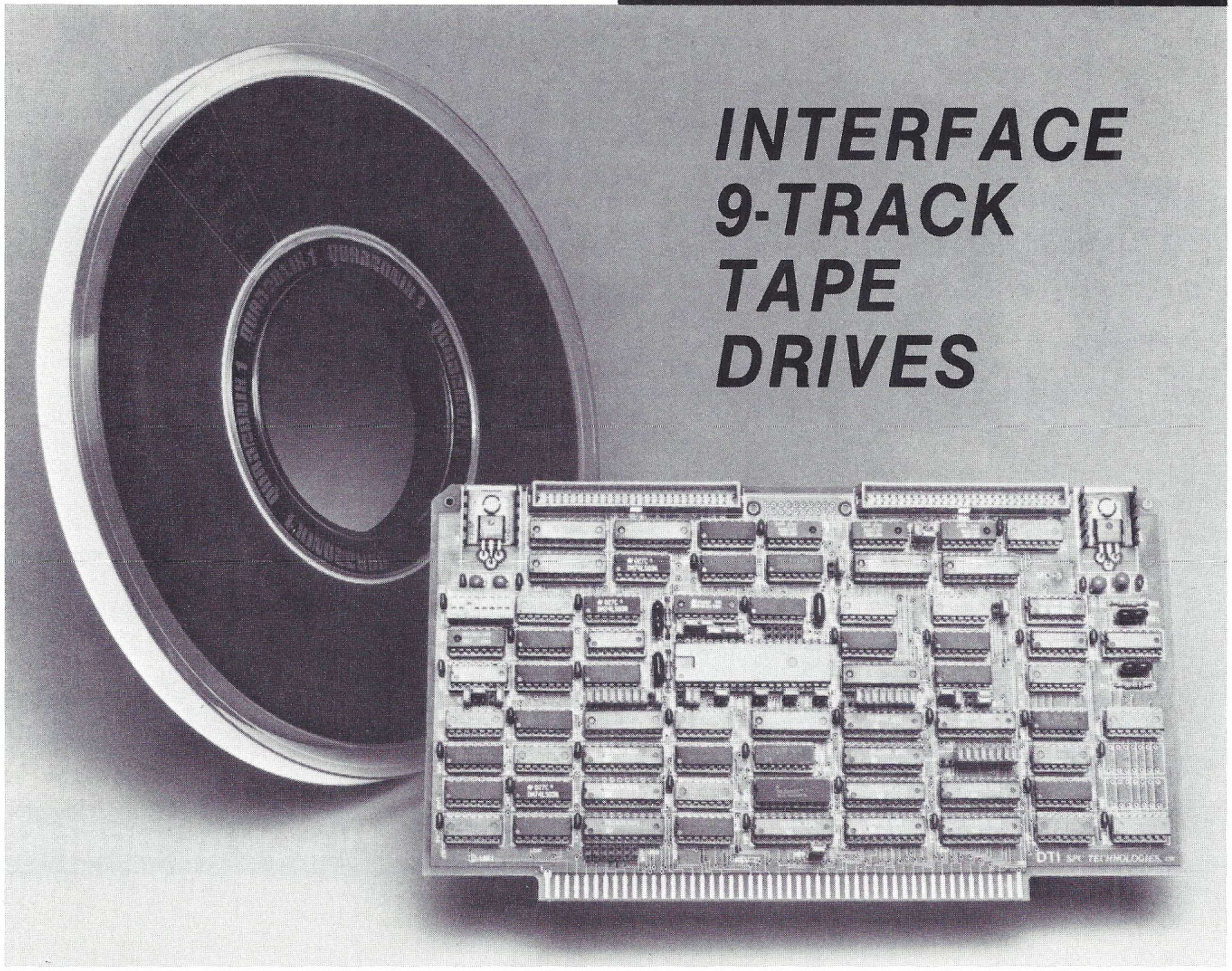
With up to 64 MHZ of 80286 processing power! and 8 megabytes of RAM for user applications! Do you want mainframe performance?

Call Now

Call Vision Computers to find out how you can experience the future today at (415) 530-2390.

Vision Computers
2235 Melvin Road
Oakland, CA 94602
Telex: 910-240-2681

S-100 USERS



With the DTI — DMA Tape-Unit Interface

- Transfers data via DMA up to 200K bytes per second
- Allows full control over all tape drive functions
- Meets IEEE S-100 standard
- Software available

For further information contact:

SPC TECHNOLOGIES, INC.

1425 N. Quincy Street, Arlington, VA 22207
(703) 276-6950

COMPATIBILITY CONCERNS

Don Pannell

Do you have a technical question about the S-100 bus?

Is there an area of the IEEE-696 standard that you do not understand?

Do you need to know how to interface a particular device or function to the bus?

696 Bus *is our regular column that helps clarify questions and concentrates on the hardware aspects of the IEEE-696 bus (i.e., the S-100 bus).*

Don Pannell, our S-100 bus expert, has been using S-100 systems and designing S-100 boards since 1978. Don was a coauthor of the IEEE-696 standard. This is the standard that defines the rules of operation of the S-100 bus. Adherence to the standard permits boards from different companies to work with each other.

If you have a question about the S-100 bus and IEEE standard, please type your question and send to Don Pannell, PO Box 700112, San Jose, CA 95170-0112.

In future columns and as space permits, Don will incorporate answers to questions received from readers.

You may have noticed a lot of older S-100 boards available on the surplus market. They can be found at swap meets, advertised in magazines, and at local surplus parts stores. Most of these boards are very good; they will plug into your system and perform the desired function without problems. There are a number of cards, however, that will give the owner nothing but problems. Most of these 'problem' cards are pre-IEEE-696 (i.e., they were designed before the IEEE specification of the S-100 bus existed), but even some newer cards won't work in everybody's system.

Compatibility problems are not unique to the S-100 bus. All open-architecture systems (S-100, VME, IBM PC & AT, ...) are unable to insure that any given product will work with all other existing products. It is, of course, in the interest of the manufacturer to make a product that is as universal as possible. But, it is impossible to design and test a product for a system that doesn't exist yet. In other words, cards that work in today's systems may not work in tomorrow's.

In this article, I will concentrate on cards that worked in yesterday's S-100 systems but may not work in today's. And what you, the purchaser, can do to better assure that a bargain price on used equipment is in fact a bargain and not a waste of money.

EARLY BOARD DESIGNS

A large majority of compatibility problems are architectural in nature. Basically this means the original engineer designed the board to work in a given system with certain cost and design targets.

For example, when the first S-100 systems appeared on the market (1975) the chief human interface was a RS-232 terminal costing around $2000.00. A lucrative S-100 product of the time was a $500 video board that plugged into the bus. All the purchaser had to add was a keyboard and monitor.

Memory was very expensive back then, so a big S-100 system had 16K to 32K bytes of RAM and the idea of ever filling the 8080's 64K RAM space was seen as unnecessary and extravagant. With these feelings in mind, the engineers of the early S-100 video cards took a 1K or 2K chunk of the 64K address space for their video card. The cards were designed this way for low-cost and high-speed operation.

Many of these cards were sold, and they worked great until memory prices started to drop and software writers began to make programs that required a system to utilize the full 64K as program RAM. Thus, the major architectual problem with these old memory-mapped video cards is the fact that they eat away program RAM space.

Video cards are not the only boards to do this. Some early EPROM programmer cards and EPROM memory cards also take away RAM space. As a secondary problem, cards like these do not work in systems that support more than 64K bytes of RAM. Therefore, these early memory-mapped cards are seldom a useful purchase, so be certain that you have a use for them before spending your money.

THE EVER-EXPANDING RAM SPACE

The original S-100 bus was built around the 8080 CPU chip. Since the chip supported up to 64K of RAM memory, this became the maximum amount of RAM the bus could support. Initially this was more than enough since the biggest RAM cards available were a 1K static and a 4K dynamic. It took sixteen 4K cards to reach the 64K limit.

It wasn't until 16K×1 dynamic RAM chips became available that problems surfaced. Now the 64K bytes of RAM could fit on a single card and companies started to work on ways that would permit more than one 64K RAM card to be used in a system. By doing this, they could sell more cards. The most common approach to add more memory was by allowing 'banks' of RAM to be paged in and out under I/O port control. This was very powerful, but little software supported it.

The Intel 8086 and 8088 CPUs arrived next, and the 64K physical RAM limit was broken. Fortunately, the IEEE working group for the S-100 bus had defined a memory extension to the bus supporting up to 16 Megabytes of RAM. Paged or 'banked' memory cards soon gave way to the new flat or 'full 24-bit address decoded' RAM cards.

The key architectural problem with the older memory cards is their inability to be used in systems that need more than 64K bytes of RAM. Even adding extra memory cards won't help because these cards do not look at or decode the upper 8 memory address lines of the bus (lines A16 to A23). A good RAM card is one that 'supports,' 'fits into,' or is 'addressable anywhere in' the 16-Megabyte address space. A card that supports this feature will have some way to select which range of memory it will fit into (usually by DIP switch). When buying a used RAM card, a quick way to check if it is addressable in the 16-Megabyte space is by looking at the S-100 bus pins. Such a card will have signals connected to the S-100 bus pins 15-17, 59, and 61-64 (the upper 8 address lines). See Figure 1 for the location of the bus pins.

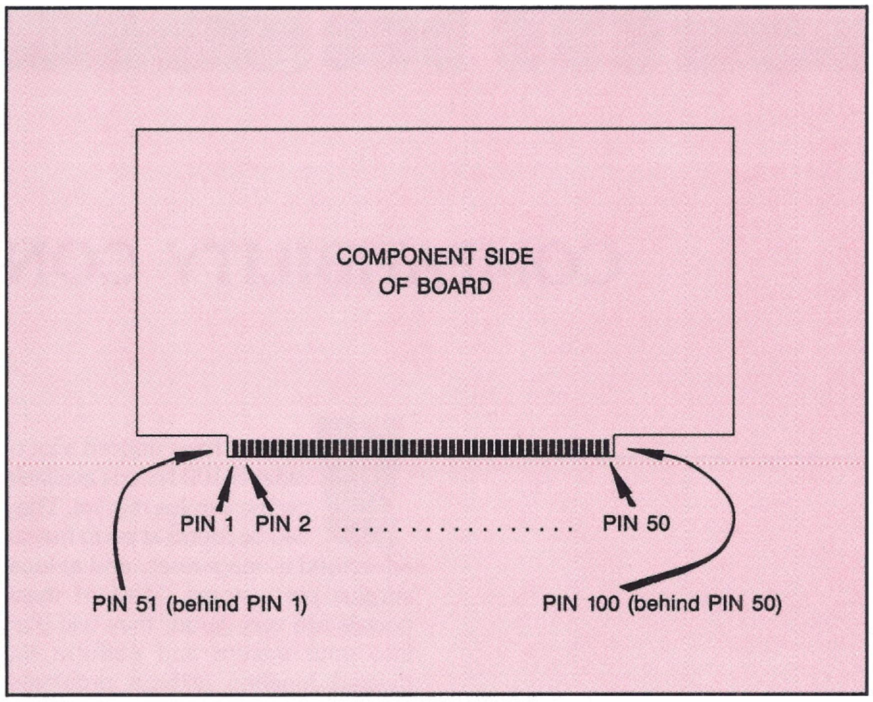


Figure 1. *Location of the S-100 bus pins.*

BOOT UP REQUIREMENTS

All computers need to be instructed on what to do after power-up. The original S-100 systems had front panels so that the first program after a power-up could be entered with switches. Today, EPROMS are used. EPROMs are read-only memory devices that don't forget their program when the power is off.

EPROMs should be architected into the system in such a way that they appear only when needed, i.e., only during boot time. Most cards that contain EPROMs do this by allowing the software to write to an I/O port that disables the EPROM and gets it out of the way. With the EPROM disabled, RAM that may appear at the former EPROM address becomes usable. This approach allows for fast and easy system boot and, at the same time, gives maximum RAM space to the software.

The desirability of overlapping memory was realized very early in the S-100 days. In order to support two memory devices at the same address, a special bus signal was defined. This signal, called PHANTOM* (bus pin 67), is driven by the EPROM card whenever the EPROM is enabled and selected for reading. The RAM card at the same address as the EPROM sees PHANTOM* as an input during these cycles and disables itself. After the EPROM is disabled, it will no longer drive the PHANTOM* signal so the RAM card will be accessed instead.

The IEEE-696 specification requires all memory cards to support PHANTOM*. Some early cards didn't, however. When purchasing a RAM card, check that it supports PHANTOM*. If it does, pin 67 will have a trace connected to it. EPROM cards should support PHANTOM* too. But these cards should drive, rather than receive the signal, and the card should support a disable function through an I/O port.

PROCESSOR DEPENDENCIES

Since the first processor on the S-100 bus was the Intel 8080, the bus ran at 2 MHz and generated 8080 timing. Next was the Zilog Z80. It ran at 4 MHz and didn't generate 8080 timing. Poor or marginally designed cards and motherboards (the bus itself) started to fail at these higher speeds and slightly different timing specifications.

Dynamic RAM cards were the most sensitive to the timing differences between the 8080 and Z80, particularly

S-100 BUS PIN NUMBER	IEEE-696 NAME	ALTAIR NAME / FUNCTION	S-100 BUS PIN NUMBER	IEEE-696 NAME	ALTAIR NAME / FUNCTION
1	+8 V (B)		51	+8 V (B)	
2	+16 V (B)		52	–16 V (B)	
3	XRDY (S)		53	0 V	SSWDSB*/Sense Switch Disable
4	VI0* (S)		54	SLAVE CLR* (B)	
5	VI1* (S)		55	TMA0* (M)	
6	VI2* (S)		56	TMA1* (M)	
7	VI3* (S)		57	TMA2* (M)	
8	VI4* (S)		58	sXTRQ* (M)	
9	VI5* (S)		59	A19	
10	VI6* (S)		60	SIXTN* (S)	
11	VI7* (S)		61	A20 (M)	
12	NMI* (S)		62	A21 (M)	
13	PWRFAIL* (B)		63	A22 (M)	
14	TMA3* (M)		64	A23 (M)	
15	A18 (M)		65	NDEF	
16	A16 (M)		66	NDEF	
17	A17 (M)		67	PHANTOM* (M/S)	
18	SDSB* (M)		68	MWRT (B)	
19	CDSB* (M)		69	RFU	PS* / Protect Status RAM
20	0 V	UNPROT / Unprotect RAM	70	0 V	PROT / Protect RAM
21	NDEF	SS / Single Step CPU	71	RFU	RUN / CPU Run
22	ADSB* (M)		72	RDY (S)	
23	DODSB* (M)		73	INT* (S)	
24	Φ (B)		74	HOLD* (M)	
25	pSTVAL* (M)		75	RESET* (B)	
26	pHLDA (M)		76	pSYNC (M)	
27	RFU	pWAIT / Processor Wait	77	pWR* (M)	
28	RFU	pINTE / Interrupt Enable	78	pDBIN (M)	
29	A5 (M)		79	A0 (M)	
30	A4 (M)		80	A1 (M)	
31	A3 (M)		81	A2 (M)	
32	A15 (M)		82	A6 (M)	
33	A12 (M)		83	A7 (M)	
34	A9 (M)		84	A8 (M)	
35	DO1 (M) / ED1 (M/S)		85	A13 (M)	
36	DO0 (M) / ED0 (M/S)		86	A14 (M)	
37	A10 (M)		87	A11 (M)	
38	DO4 (M) / ED4 (M/S)		88	DO2 (M) / ED2 (M/S)	
39	DO5 (M) / ED5 (M/S)		89	DO3 (M) / ED3 (M/S)	
40	DO6 (M) / ED6 (M/S)		90	DO7 (M) / ED7 (M/S)	
41	DI2 (S) / OD2 (M/S)		91	DI4 (S) / OD4 (M/S)	
42	DI3 (S) / OD3 (M/S)		92	DI5 (S) / OD5 (M/S)	
43	DI7 (S) / OD7 (M/S)		93	DI6 (S) / OD6 (M/S)	
44	sM1 (M)		94	DI1 (S) / OD1 (M/S)	
45	sOUT (M)		95	DI0 (S) / OD1 (M/S)	
46	sINP (M)		96	sINTA (M)	
47	sMEMR (M)		97	sWO* (M)	
48	sHLTA (M)		98	ERROR* (S)	SSTACK / Status Stack
49	CLOCK (B)		99	POC* (B)	
50	0 V		100	0 V	

Table 1. *Current S-100 bus signals. Also original Altair signals that differed in function from the IEEE-696 standard. The (M) in front of a signal name indicates a master-generated signal, (S) a slave-generated signal, and (B) a bus signal.*

where it concerns the timing of memory refresh. To solve this problem, these cards started coming out with jumpers to select the processor (either 8080 or Z80). This was acceptable until the 8088 and other CPUs became available and the previous timing assumptions of when and how to do refresh were no longer valid. The cost of upgrading the CPU in one of these systems also included the cost of a memory card. This is why most memory cards available today are static memory cards. They are generally more tolerant of processor differences.

The trickiest 'processor' of all that gives many old dynamic RAM memory cards problems is any DMA device. DMA stands for Direct Memory Access and is a device that reads or writes RAM without the help of the CPU. Most floppy and hard disk controllers are DMA devices. The controller is given a pointer to an area of memory to read or write and it does the rest. What this means is that while the DMA device is controlling the bus, it must look exactly like an 8080 CPU (or meet the IEEE-696 specifications). While this is not too hard to do, problems arose in the window of time while control of the S-100 bus was being transferred from the CPU to the DMA device, and back again. The IEEE-696 specification fixed this problem by better defining how this transfer was to be executed.

When purchasing a floppy or hard disk controller, it should be a card that supports DMA transfers and does so by following the IEEE-696 specification. An easy way to tell this is by verifying that the card supports TMA (or DMA; in the final revision of the 696 standard, DMA was renamed TMA, Temporary Master Access) bus arbitration. If it does, the card will contain a 4-position DIP switch or header block to select its priority number. An examination of the card's schematics should show logic connected to S-100 bus pins 55, 56, 57, and 14 in order to support this function.

MEMORY BUS WIDTH

The original S-100 bus was 8 bits only. Later, the IEEE-696 standard defined support for 16-bit transfers as well. While the standard requires all 16-bit memory cards to support both 8-bit and 16-bit data transfers, bus masters are not required to do so. This means that you can always buy 16-bit-wide memory cards even if you don't have a 16-bit-wide processor card. In fact, this is a very good idea. That way, when you upgrade to a 16-bit processor, the full potential of the new card will be realized.

Placing a 16-bit-wide processor in a system with 16-bit-wide memory should never create a problem. But this same processor placed in a system with some 8-bit-wide memory may present a problem. According to the IEEE-696 standard, 16-bit-wide processor cards (bus masters) have a choice when attempting 16-bit transfers to 8-bit-wide RAM. They can either 'conduct the requested 16-bit transfer as two byte (8-bit) operations, thus assembling the requested 16-bit word while holding the master in wait state,' or generate an error trap by asserting the ERROR* line. 16-bit bus masters that read 8-bit data by what is known as *byte-serial* are by definition very compatible. There are, however, IEEE-696 16-bit processor cards out there that simply generate the error trap when an 8-bit RAM card is addressed, and thus do not work with 8-bit memory.

Unfortunately, there is no easy way to tell which of the two approaches any given CPU card was designed to. One possible way is to see if the card drives the ERROR* line (bus pin 98). This is an inconclusive test, however. The best approach is to call the vendor or ask someone who might know (or write me a letter).

BUS PIN CHANGES

Table 1 lists the S-100 bus pins and shows those that were modified by the IEEE-696 standard from the original MITS Altair specification. Most of the pin changes removed front panel signals that were not necessary on the bus. Instead, a front panel system could get the extra signals it needed from the desired card(s) with a direct connect cable. These signals include pin 21, *Single Step*; pin 27, *Processor Wait*; pin 28, *Processor Interrupt Enable*; pin 53, *Sense Switch Disable*; and pin 71, *Run*. All but pin 53 cause no compatibility problems since they were redefined as *NDEF*, Not Defined, or as *RFU*, Reserved For Future Use.

Pin 53, Sense Switch Disable, can be a problem. It is a front panel signal used to disable the input data bus drivers so that the front panel could control the CPU on specific cycles. The problem is that pin 53 is now defined as a ground pin. Plugging in just one S-100 card that connects pin 53 to GND into a working front panel system will completely stop the system. No permanent damage will result. The processor just won't be able to read any data from the bus. A fix for these systems is easy enough. Disconnect pin 53 from GND on the problem card. This can be done by cutting a trace or placing a piece of tape on its pin 53.

The opposite situation of an old CPU card in a new chassis is one to watch as well. Processor cards that support pin 53 as Sense Switch Disable will have problems in motherboards that supply GND on pin 53. Again the CPU's data bus will be disabled for all reads. Cutting the trace on the CPU card will solve this one.

Pin 20, *Unprotect*, pin 69, *Protect Status*, and pin 70, *Protect*, were used to write-protect banks of memory. This allowed a programmer who just entered a program via her front panel switches to protect her work from a stack under/overflow bug — which would wipe out all the memory (and all her work). As systems advanced, these functions were no longer needed. The IEEE-696 standard defines the two control pins (Protect and Unprotect) as Grounds and the RAM card output pin (Protect Status) as RFU, Reserved For Future Use. No known compatibility problems exist with these pins.

The last signal, *Status Stack*, pin 98, was a CPU card output signal used to indicate that a stack memory push or pop operation was in progress. This pin was changed to ERROR* which is generally an input to newer CPU cards. The only potential problem is with an external interrupt controller card that supports the ERROR* function being plugged into a system with a CPU card that generates SSTACK on pin 98. Since only 8080 CPU cards output the SSTACK signal, this problem is very rare.

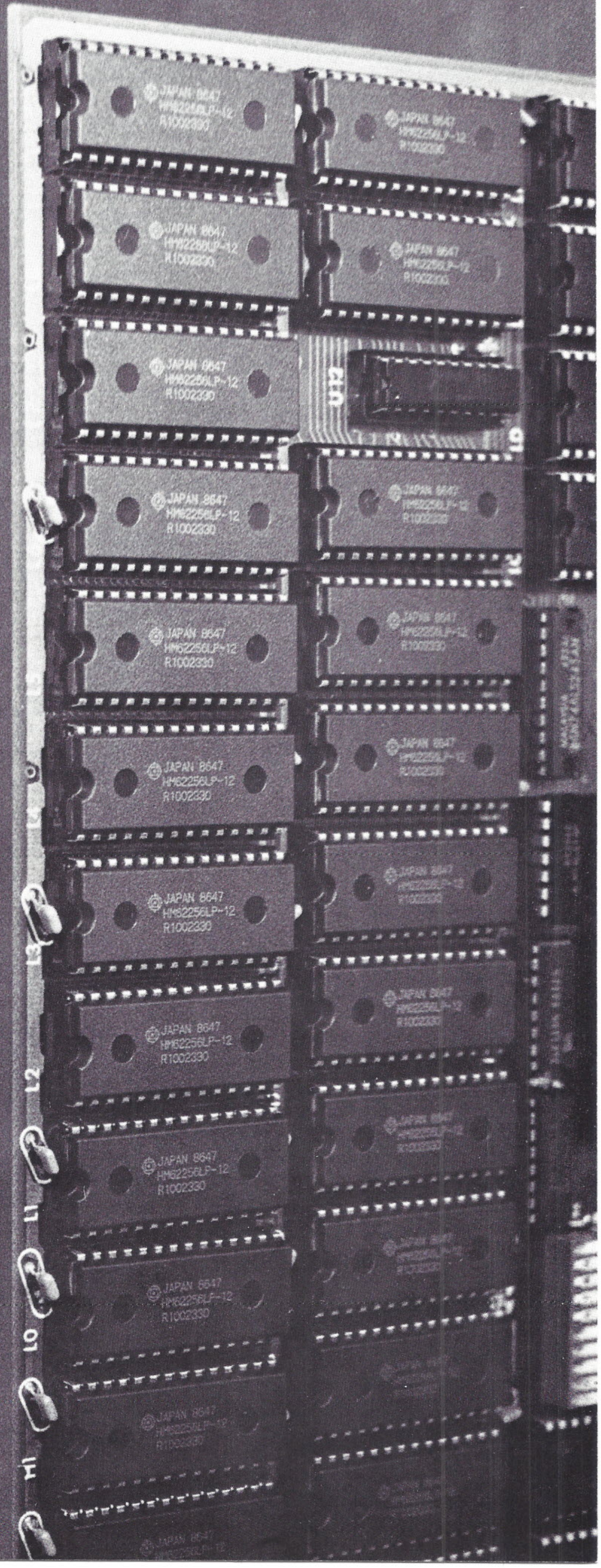
CONCLUSION

Very few old S-100 boards are completely unusable. Many with architectural problems can be reworked and made usable again if you are inclined to make the changes. This brings up a very important point: **Never** buy any 'as is' product unless schematics are available to you. First, you should study the schematics before you buy so you know what you are getting into. Then, if you decide that you need to make some modifications, you will have the necessary information.

I have upgraded many cards for my system. If you have any questions about your S-100 computer, drop me a letter (PO Box 700112, San Jose, CA 95170-0112). Please include a phone number and address. Include specifics about your hardware and your goal or the changes you would like to make. ■

New Megabyte Lower Prices:

TEN STATIC BYTES FOR A MERE CENT



FULLY ASSEMBLED AND TESTED FAST S-100 MEMORY BOARDS

- Up to 1-Megabyte on a single board.
- 10-MHz operation with no wait states.
- 8 and 16-bit operation per IEEE-696.
- One year warranty on parts and labor.

Populated w/ 64K RAM	$350
Populated w/ 256K RAM	$475
Populated w/ 512K RAM	$700
Populated w/ 1 Megabyte	$995

*Please add $5 for shipping and handling. Maryland residents add 5% sales tax.

Guaranteed to work in your system. If a purchased board fails to work, customer has the option of a full refund if board is returned to KepTronics within 15 days of purchase.

KepTronix

P.O. Box 2022 Columbia, MD 21045
(301) 381-4297

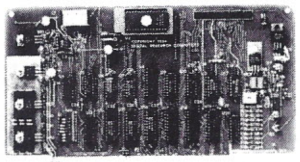
ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A COPY OF THIS WARRANTY IS AVAILABLE FREE, ON REQUEST.

DIGITAL RESEARCH COMPUTERS

(214) 225-2309

S100 EPROM PROGRAMMER

OUR NEWEST DESIGN, FOR FAST EFFICIENT PROGRAMMING OF THE MOST POPULAR EPROM'S ON YOUR S100 MACHINE. COMES WITH MENU DRIVEN SOFTWARE THAT RUNS UNDER CP/M 2.2 (8 INCH). PC BOARD SET CONSISTS OF (S100) MAIN LOGIC BOARD REMOTE PROGRAMMING CARD AND SIX PERSONALITY MINI BOARDS FOR 2716, 2532, 2732, 2732A, 2764, AND 27128. SOLD AS BARE PC BOARD SET ONLY WITH FULL DOC. SOFTWARE FEATURES "FAST" PROGRAMMING ALGORITHM. FOR Z80 BASED SYSTEMS.

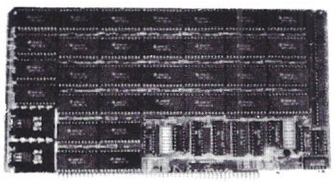


PC BOARD SET, FULL DOCUMENTATION, 8 IN. DISKETTE WITH SOFTWARE.

NEW! $69^{95}

64K S100 STATIC RAM

$99^{00} KIT



LOW POWER!

150 NS ADD $10

BLANK PC BOARD WITH DOCUMENTATION $49.95

SUPPORT ICs + CAPS $17.50

FULL SOCKET SET $14.50

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

ASSEMBLED AND TESTED ADD $50

FEATURES: *PRICE CUT!*

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

128K S100 STATIC RAM/EPROM BOARD

JUST OUT! USES POPULAR 8K X 8 STATIC RAMS (6264) OR 2764 EPROMS. FOR 8 OR 16 BIT DATA TRANSFERS! IEEE 696 STANDARD. LOW POWER. KITS ARE FULLY SOCKETED. FULL DOC AND SCHEMATICS INCLUDED. 24 BIT ADDRESSING.

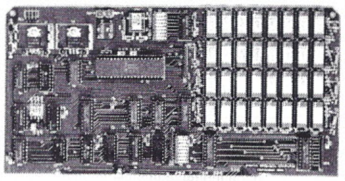
NEW!

$59^{95} BARE PC BOARD — $219^{00} 128K RAM KIT — $139^{00} 128 EPROM KIT

256K S-100 SOLID STATE DISK SIMULATOR!

WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

PRICE CUT!



FEATURES:
- ★ 256K on board, using +5V 64K DRAMS.
- ★ Uses new Intel 8203-1 LSI Memory Controller.
- ★ Requires only 4 Dip Switch Selectable I/O Ports.
- ★ Runs on 8080 or Z80 S100 machines.
- ★ Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
- ★ Provisions for Battery back-up.
- ★ Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
- ★ The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
- ★ Compare our price! You could pay up to 3 times as much for similar boards.

BLANK PCB (WITH CP/M* 2.2 PATCHES AND INSTALL PROGRAM ON DISKETTE) $24^{95} (8203-1 INTEL $29.95)

CLOSE OUT! BLANK PCB ONLY: $24^{95}

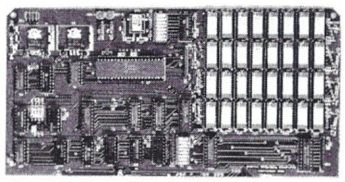
#LS-100

1 MEG. S-100 SOLID STATE DISK SIMULATOR!

WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.

LS 100 II

NEW!



FEATURES:
- ★ 1 Meg. on board, using +5V 256K DRAMS.
- ★ Uses new Intel 8203-1 LSI Memory Controller.
- ★ Requires only 4 Dip Switch Selectable I/O Ports.
- ★ Runs on 8080 or Z80 S100 machines.
- ★ Up to 4 LS-100 boards can be run together for 4 Megs. of On Line Solid State Disk Storage.
- ★ Provisions for Battery back-up.
- ★ Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
- ★ The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
- ★ Compare our price! You could pay up to 3 times as much for similar boards.

BLANK PCB (WITH CP/M* 2.2 PATCHES AND INSTALL PROGRAM ON DISKETTE) $59^{95} (8203 1 INTEL $29.95)

(ADD $50 FOR A&T) $259^{00}

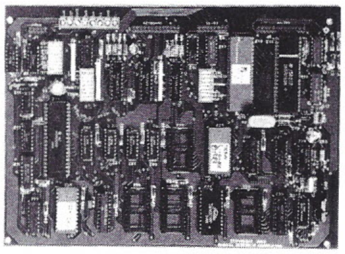
#LS-100 II (FULL 1 M.B. KIT)

1 MEGA BYTE!

ZRT-80 CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

FEATURES:
- • Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
- • RS232 at 16 BAUD Rates from 75 to 19,200.
- • 24 x 80 standard format (60 Hz).
- • Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
- • Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
- • Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
- • 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
- • Composite or Split Video.
- • Any polarity of video or sync.
- • Inverse Video Capability.
- • Small Size: 6.5 x 9 inches.
- • Upper & lower case with descenders.
- • 7 x 9 Character Matrix.
- • Requires Par. ASCII keyboard.



$89^{95} #ZRT-80 — A&T ADD $50

(COMPLETE KIT, 2K VIDEO RAM)

BLANK PCB WITH 2716 CHAR. ROM. 2732 MON. ROM $49^{95}

SOURCE DISKETTE - ADD $10

SET OF 2 CRYSTALS - ADD $7.50

FOR 8 IN. SOURCE DISK (CP/M COMPATIBLE) ADD $10

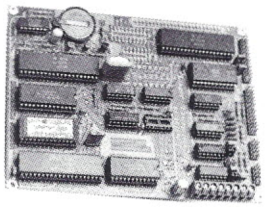
THE NEW 65/9028 VT ANSI VIDEO TERMINAL BOARD!

★ FROM LINGER ENTERPRISES ★

A second generation, low cost, high performance, mini sized, single board for making your own RS232 Video Terminal. This highly versatile board can be used as a stand alone video terminal, or without a keyboard, as a video console. VT100, VT52 Compatible.

FEATURES: — MICRO SIZE!

- ★ Uses the new CRT9128 Video Controller driven by a 6502A CPU
- ★ On-Screen Non-Volatile Configuration
- ★ 10 Terminal Modes: ANSI, H19, ADM-5, WYSE 50, TVI-920, KT-7, HAZ-1500, ADDS 60, QUME-101, and Datapoint 8200
- ★ Supports IBM PC/XT, and Parallel ASCII Keyboards
- ★ Supports standard 15.75 kHz (Horiz.)
- ★ Composite or Split Video (50/60 Hz)
- ★ 25 X 80 Format with Non-Scrolling User Row
- ★ Jump or Smooth Scroll
- ★ RS-232 at 16 Baud Rates from 50 to 19,200
- ★ On Board Printer Port
- ★ Wide and Thin Line Graphics
- ★ Normal and Reverse Screen Attributes
- ★ Cumulative Character Attributes: De-Inten, Reverse, Underline and Blank
- ★ 10 Programmable Function Keys and Answerback message
- ★ 5 X 8 Character Matrix or 7 X 9 for IBM Monitors
- ★ Mini Size: 6.5 X 5 inches
- ★ Low Power: 5VDC @ .7A, ± 12VDC @ 20mA.



$79^{95} FULL KIT

w/100 Page Manual

ADD $40 FOR A&T

OPTIONAL EPROM FOR PC/XT STYLE SERIAL KEYBOARD: $15

SOURCE DISKETTE: PC/XT FORMAT 5¼ IN. $15

Digital Research Computers

P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

TERMS: Add $3.00 postage. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCard. Tex. Res. add 6-1/4% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50 add 85¢ for insurance.

*TM OF DIGITAL RESEARCH INC. (CALIF.)

WE ARE NOT ASSOCIATED WITH DIGITAL RESEARCH INC. (CALIF.) THE SUPPLIERS OF CPM SOFTWARE

BY WAY OF INTRODUCTION

Jack L. Chalker

We are very pleased to welcome Jack L. Chalker as a regular columnist for S-100 Journal. Very prolific (check out his shelf of books at your nearest B. Dalton and you'll see what we mean) and well known, Jack is the author of 34 novels, mostly science fiction, including six best sellers so far. He has been a computer technician, rock concert engineer, audiovideo consumer electronics reviewer, teacher, book and magazine columnist, computer typesetter, Air Force public relations man, book and magazine publisher, and a host of other occupations until he found his niche in writing and stayed there full time. And most important, he is a dedicated S-100 user.

In Buslines, *Jack will give his user's viewpoint, let you know what products work well, poorly, or not at all in his S-100, and offer general commentary on S-100-related matters. The opinions expressed are his own and may or may not reflect the views and positions of S-100 Journal.*

As with the other columns, your comments and questions are welcome. Please send your letters regarding this column to Buslines, S-100 Journal, PO Box 1914, Orem, UT 84057.

OK, now, who's the best-selling professional science fiction writer with a long background in consumer electronics who writes all his books on his CompuPro S-100 computer? No, not *him* — I'm the other one. You'd never imagine *him* engineering outdoor rock concerts in the Sixties, would you? While at the same time co-running a computer typesetting company that, among other things, typeset half the underground newspapers of the period done in the east? And I'm the one who calls the CompuPro the CompuPro and the Toshiba the Toshiba, not Irving and Fred.

All right, that out of the way, I'm happy to be back above ground in computers after a couple of years between magazines and generally limited to the S-100-oriented BBSes. My own computer experience goes back to running the old IBM 1060 that used to be the brains of Johns Hopkins University, handling the graveyard shift for this massive machine complex in a virtual white room that ran Autocoder and took thousands of punch cards. The typesetting company used the original Varityper paper tape input phototypesetter that was down more than up (the repairman actually kept a coffee cup inside the machine along with other things since he was there so much), and then the output went to huge layout tables where complex paste-ups were ultimately turned into enormous negatives for big Harris presses. Desktop publishing it wasn't, but it was close to state of the art in those days.

I can't claim to have had an Altair or even a Radio Shack Model One, but I've been doing much with personal computers since they became affordable. I blundered into UNIX (Xenix, actually) before UNIX was cool by buying one of the early Radio Shack Model 16s, an experience that convinced me that, for the average user, UNIX was a system whose time probably would never come even though its domination was being predicted even then. MicroSoft has been trying to get us all over there ever since, without much success. I had suspected that the ballyhooed OS/2 was going to be a slimmed down single-user Xenix after all, and so far all that I hear about their attempts to get it slimmed down to a mere 500K or so, and the aching slowness of the beta copies, confirms that suspicion. There are many businesses that really need UNIX, or something better than UNIX, but single users and most small businesses aren't among them.

It makes you wonder how Digital Research gets it all into 172-212K, doesn't it? If DRI had better PR and promotion and a less naive front office (you know — they think that just because they have a better product the world will eventually find out about it and switch) MicroSoft would be back in the BASIC business.

My own main system is a CompuPro 816/C, with the Q540 subsystem, 5″ and 8″ drives, CPU 286, PC Video to a Quimax 14″ mono monitor and Enigma 9000 keyboard as Console 0 (and virtuals 1-3), and an Esprit III Color terminal on one of those swinging platforms as Console 4. There's a Meg of memory in it, a PC-PRO replacement ROM on the Disk 1A so I can boot that system if I want to, a half-Meg M-Drive used

S-100 BOARDS

■ **DUPLEX 816:** 8 MHz 80186 & Z80H CPUs, 2 RS232 Serial Ports, 5 & 8-inch Floppy Interface, 8 Kbyte ROM, Memory Management to 16 Mbytes & Interrupt Handler. Runs PC-DOS, CP/M-80, CP/M-86, and Concurrent CP/M-86. CP/M-80 vs 2.2 included. Price: $995.

■ **CPU/IO:** 4 or 6 MHz Z80, 2 RS232 Serial Ports, Memory Management to 1 Mbyte, Programmable Timer, Vectored Interrupt. Price: $316.

■ **DD-ASM:** Double Density Floppy Disk Interface, including boot ROM, Direct Memory Access, CP/M 2.2 operating system. Price: $316.

■ **FDI-ASM:** Single Density Floppy Disk Interface, including boot ROM. Assembled: $260, Kit: $180.

■ **4S/2P:** Four RS-232 Serial and two Parallel ports. Price: $360.

SOFTWARE

■ **DB-3:** The Tarbell Database System, for PC-DOS, CP/M-80, CP/M-86, or Concurrent CP/M-86, Multi-User File Locking, Interactive Program Generator, Up to 19 Files Open at Once, Extensive HELP system, Multi-Field Sort, and Multi-Line Field Type, Price: $95.

■ **CONCURRENT DOS:** Multi-User, Multi-Tasking Operating System for Duplex 816 Above. Has Virtual Terminals, PC-MODE, and Many Utilities, Price: $600.

■ **CP/M-80 vs 2.2:** Industry Standard Operating System for 8080, 8085, or Z80. Price: $75.

■ **INV:** Manufacturing Inventory System, written in dBASE-II. Items may consist of as many levels of other items as desired. Menu includes buying, receiving, packaging, shipping, listing, totaling, and others. Price of source: $80.

■ **SALES:** Sales Order Entry System, written in dBASE-II. Menu includes order entry, change order, list orders, print invoice, and others. Price of source: $80.

■ **TBAS:** Tarbell Disk BASIC Interpreter. Has many extra functions, such as high-speed file search, I/O reassignment, procedures, and alphanumeric labels. Price: $49.

■ **PCPROM-SRC:** Assembly-Language source for our own ROM-BIOS that's used on the Duplex 816 to boot up PC-DOS directly. Price: $150.

TARBELL ELECTRONICS
1082 E. Artesia Blvd., Suite C
Long Beach, CA 90805, (213) 422-7081

as a hard-disk cache, SS1, SPU-Z, and Disk 3, and the system comes up under CompuPro's current CDOS 5.0C-2. There's also a U.S. Robotics modem attached, and Printer 0 is an Apple LaserWriter Plus fed via a MacMaster 1-Megabyte print buffer/controller while Printer 1 goes to a venerable Anderson Jacobson 833 daisy wheel. CDOS (Concurrent DOS) runs partitions A, C, and D; B is reserved for PC-PRO for reasons I'll get to later on. There are five computers in the house but only one is MS-DOS, my Toshiba T-3100 laptop, and that by default.

Some time this year, I intend to upgrade to the new, faster SS2/RAM 24/12.5 MHz 286 system and possibly add PCLink and a bigger disk or a second 540. I'm also waiting with baited breath for one of the four third-party suppliers known to me who are working on a CompuPro-compatible S-100 EGA board, including Lomas and CCT. Programs like Ventura Publishing run on the CompuPro under 5.0 and they are ***fast***, but unless we can get something better than CGA, the screen output where all the work is done is simply too poor to think of doing serious desktop publishing on the CompuPro without PCLink. Possibly the fact that CompuPro is discontinuing the PC Video card (although it's promised to keep supporting it in the operating system for 'the foreseeable future,' whatever that means) will spur more third-party video I/O development.

Naturally, there are those who ask why such compatibility is at all needed or desirable (including Dr. Godbout of CompuPro, who told me that it was next to nil on his priority list). One reason is that CDOS is much better implemented on the S-100 bus as of now than on the PC; the PC version is much slower and must make too many compromises with the PC hardware. Another is that there is no reason why businesses should have to have separate and different computers to do routine letters and reports, some desktop publishing, and the like from the main computer whose customized CDOS software is used for the company database or other major operations. Frankly, most word processors in the CP/M-86/CDOS world are venerable antiques with limited capabilities and zero growth, so compatibility with MS-DOS word processors is a real need.

And, finally, why shouldn't salesmen whose field laptops are using 1-2-3, or some other useful and standardized spreadsheet and database applications, be able to sit down back at the office and just read in those data files (or squirt them via modem) into the main office machine without having, back at that office, to either translate them into a different file format or remember two different spreadsheet and database programs?

Companies like CompuPro have always taken the philosophical view that hardware should drive software — and it's probably true that they *should* but, like King Canute commanding the tide not to come in, it simply isn't a realistic or healthful view of the way things work. Compaq, in fact, is the model of a company that understands the industry. They saw the coming tide and decided to accept it but to do it better, and they grew to number two in the industry even though they had the gall to charge *more* than IBM. The S-100 world is a different world, but the nature of our machines is that we can do not just one thing but ***most*** things better — if the industry wants to. Better, faster hardware, more efficient software and operating systems, and a design that is the antithesis of planned obsolescence. We don't have to junk our old machines and bus and software just because one company dictates we must.

The real attraction of my system, for example, is that I can run just about anything I liked from the old CP/M days, including utilities, and also CDOS, CP/M-86, and MS-DOS programs interchangeably, and I can have the same database and word processor on my Toshiba T-3100 and my CompuPro thereby simplifying everything. And, I must tell you, whenever I'm over in MS-DOS there inevitably comes a time when I want to do something that is rather simple and obvious under CDOS, and then I remember that it just isn't there. Some of those features and functions can be attained for MS-DOS by buying a ton of software add-ons and winding up with 32K of usable memory left after you loaded them, it's true, but even then where's the elegant simplicity of PIP and the convenience of floating drives and a hundred other 'little' things that make work so much easier and more pleasant — without hogging all that memory? And where can I stop worrying about conflicting and confusing memory-hogging resident software to do what I want to do when I want to do it?

Over on the venerable old 'obsolete' S-100 CompuPro, that's where.

At least everybody in the 'mainstream' computer world tells me that the S-100 bus is antique and obsolete, just like that giant IBM of over twenty years ago. Of course, none of them has ever seen or played with an S-100 computer — why bother with obsolete antiques when you can be running a Compaq 386 or drooling over a Model 80? — and all they know about CDOS is that *InfoWorld* told them it wasn't as good as Desqview so why bother? And, of course, if indeed the world should finally go to UNIX, when I want it all I have to do is stick in another processor board and I'm running a very elegant System V without having to throw out my old stuff. There are many fine, debugged UNIX systems on S-100 right now who won't have to reinvent the wheel the way IBM would.

The problem with the S-100 world is the same as DRI's problem and even Sony's problem. That is, beta is better but VHS won the war anyway. When hype and advertising and marketing clout was mobilized by Matsushita and RCA against the beta format in an incredibly efficient way, sales and use of beta machines dropped like a stone — never mind my Superbeta Hi Fi runs rings around my VHS HQ Hi Fi in picture, sound, and stability — to the point where many video rental stores didn't even stock beta, and those that did stocked only a fraction of the VHS titles. Then all the friends and neighbors had VHS so you wanted to be compatible to swap tapes, right? And the old vintage movies came out cheap only on VHS, so....

Likewise, Digital Research has the better operating system — or systems, since FlexOS is even more amazing and already out and mostly debugged for the protected mode 286/386 world while MicroSoft struggles — and DRI has the ones clearly best suited to the majority of small and medium business applications, but MicroSoft and IBM won the hype and advertising and marketing wars, and software trotted mostly to them. And, make no mistake, it is software that drives the computer industry, not hardware (or how many 286 protected mode applications have you seen so far?) Unlike beta, it's not out for the count yet, thanks to us diehards, some incredibly energetic and brilliant apostles like Alex Soya and Bill Spoolhof, and to a stronger European base. But the odds are against it, and only OS/2 may give CDOS its window of opportunity if DRI is smart enough now and clever enough now to see

Now Available from *S-100 Journal:*

IEEE Standard 696 Interface Devices

This is the official booklet published by The Institute of Electrical and Electronic Engineers, Inc. It describes in technical detail the IEEE-696 standard or, in other words, the complete specs for the operation of and interfacing to the S-100 bus. All the bus lines and their relationships are described in detail. (Note: this book is not a tutorial.)

To order this book, please complete and send a copy of this form (or the back issues reply card) with payment to:
S-100 Journal, PO Box 1914, Orem, UT 84057

NAME: ______________________________

____

MAILING ADDRESS: ______________________________

____

Please include prepayment of $7.50 plus $1.00 for shipping and handling. We accept the following forms of payment: ☐ *Check or M.O.* ☐ *American Express* ☐ *VISA* ☐ *MasterCard*

*Card No:*______________ *Expiration:*__________ *Signature:*__________

the opening and crash through.

Likewise, the S-100 world suffers not from any hardware problems or any obsolescence compared to what IBM is foisting off on the world to date, but because the corporations went IBM's way so did the vast bulk of the third-party hardware and software makers. You can get a Bernoulli controller for the PC world, the MAC world, even for Radio Shack Model II/12/16s for heaven's sake — but not one for an S-100 machine. Iomega is convinced that the market out there is too small to make it financially worthwhile. We're just now getting a true SCSI interface. Several plans to market IBM to S-100 bus expansion boxes failed not because they don't work (they do) but because they weren't convinced anybody would buy them. VGA is creeping in and we're still waiting for the first company to give us a compatible EGA even though there's loads of S-100 folks out there panting for a decent AutoCAD environment or desktop publishing.

That's one thing I want to do here. I am always out there looking for good and useful third-party boards and devices for the S-100 world. I have a number of contacts in the industry but I can't know about everything or everybody, so if you have a better board or a better device or a better idea for S-100s, by all means write and tell me so I can tell everyone else. Back when I did this sort of thing for the late *Advanced Computing* magazine, I tried a number of brilliant third-party devices, some of which needed work but others required only user tests and publicity. I guarantee only to call 'em as I see 'em and play no favorites, and I also warn that if there is a fatal bug in either hardware or software I will find it if only by accident, but I'm knowledgeable and I'm fair, and I'm not too proud or vain to go to the experts when I need help, advice, or something explained to me in words of two syllables. Others in this magazine will cover the complex engineering and design areas of all sorts of programs and devices; I'll tell you if it works, how it works, and how useful it is.

This is also true of software. Since software drives the industry and S-100s are not merely development machines but also *business* machines — S-100 machines run everything from automated mining equipment in Pennsylvania coal mines to the 900-number Dial-A-Joke and Dial Porn in LA — then generalized applications for the CDOS environment are essential to its continued health. I would rather not have to worry about being 'compatible' with an inferior computer machine and operating system family. I would prefer to see superior applications in all fields under CDOS making full use of both the operating system and the enormous hardware flexibility of the machines.

One thing you can do is write software companies and third-party S-100 vendors and tell them what you want and assure them that you'll buy these things if they are produced and work as promised. Many companies do listen — a suggested campaign on an S-100-oriented BBS to write MicroPro and urge release of a CDOS WordStar 4 elicited a response that if there was that much interest they would do just that.

But this also calls for help from those who provide our operating system environment. I just ran into a good example of this which I'll pass along to you.

Recently I enabled CompuPro's CDOS 5.0 on my system and it was everything I wanted CDOS to be, only three years later than I hoped and one year later than I expected. There are some bugs — the official bug report lists a few thousand — but also a lot of good things like command line redirection, full path support, added utilities with additional versatility, and it even runs dBase III. Of course, the system keeps insisting it's July when it's May, and if you had drive D as DOS Media under 4.1, you'll find that they relocated the D directory (but not C) in 5.0 so the system can't find anything (back up full first!), but that's minor stuff. I was overjoyed — until I went to print a letter from my word processor, an MS-DOS generic program I love called FinalWord II that was always a very good friend of CDOS and the CompuPro and which is designed not just for IBM PCs but to run on terminals and even drive typesetters. It always printed before, even under 3.1, but now it sent the formatted file to the screen instead of the printer port. Anguish.

I pulled several MS-DOS programs off the shelves, most of which I never use, and stuck them in there, including Dac Easy Base (which now ran), Word Perfect, and WordStar 2000, and some hung up and some just went dark but none printed. I reported this first to my local Systems Center man and then to a number of CompuPro folks out there who earn their livings with the machine, and they all verified the problem. I even called CompuPro tech support on this, and they also were croggled when after 20 minutes of convincing them I wasn't any novice or nerd, they tried it and also failed. Everybody promised to get back to me and nobody did, but some west coast CompuPro programmers began reporting that a number of programs including dBase III and WordStar 2000 now printed. That bothered me more than a total lack of the capability. Some programs printed, others did not. Why?

Now, I can do a fair amount of programming and fool around with all sorts of things, but I'm pretty well stuck with CompuPro's CDOS. I keep PC-PRO on that Drive B, though, so I could still write normally on my machine and print. That's one reason I keep it; another is that I often have to send and receive data from MS-DOS-specific programs that just don't work under CDOS, and it's handy for that as well — and, of course, it verifies that new problems aren't in the hardware but in the software. I find PC-PRO useful but as it turns my $10,000 CompuPro into a fast, decent $1995 AT clone that supports 8″ drives, it's not what I have the machine for.

Impatiently, I wrote a letter to Dr. Godbout about my frustrations and, to my surprise, he called me. The conversation was, however, a bit dull on my side (I keep Dracula's hours and am not at my best when awakened from a sound sleep halfway through said sleep period), and certainly mutually frosty. Basically, I was told that CompuPro never promised compatibility, wasn't interested in users like me any more, and if I didn't like what they gave then I should sell my CompuPro and buy an IBM! Silly me — I thought that when CompuPro's 5.0 User's Guide touted 'PC DOS 2.1 and MS-DOS application support' that indicated that compatibility was indeed a factor. At any rate, for having the effrontery to complain that something that *used* to work *didn't*, I received calls like that from both Dr. Godbout and his tech support people and, later, a letter also stating much the same.

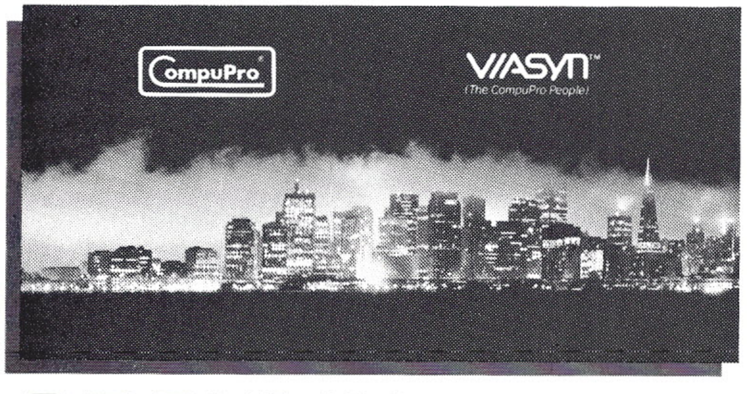
I suspect my letter, which was essentially a bug report (and they have always solicited bug reports) just hit them at a bad time, and I became the object of frustrations having nothing to do with me, but it smacked of killing the messenger bearing bad news. In any event it was bad business and not what the S-100 world needed, assuming I wasn't the only one bitten.

CompuPro has taken the bug seriously, warning people about it on BIX and assuring people that it is indeed a high priority fix, so I assume that this was also a complaint of many of their prime customers. In the meantime, I found not the problem but at least a solution that got everything up and running. The key was that with a few replaced CON files (making 5.0C-2, by the way, the current version you'd get if you ordered now) supplied by Mike Burgett, a lot of programs *would* print — but not others. By trial and error I discovered that all MS-DOS 1.x programs printed, and that the ones for 2.x or above that printed either retained the old MS-DOS 1.x printer calls (such as WordStar 4) or had a two-tiered approach for compatibility in which if the MS-DOS 2.x call wasn't returned they then tried the 1.x call. dBase III and WordStar 2000 do this. When I removed the operating system output from FinalWord II and replaced it with a DOS1Call I/O, it, too, printed as before.

The old default in CDOS was to take whatever DOS call there was and simply give the proper IOTCL return and then route the output to the logged printer no matter what was called for. The PC version still does this. Apparently CompuPro eliminated this default because it either confused the print manager or interfered either with the new SS2 calls or maybe the SPIO calls for the 80186 slaves which are now CompuPro's highest priority. This sort of thing is to be expected when trying networking and large slave processor support under a multiuser operating system when one runs a program that is strictly single user, but apparently it wasn't noticed and therefore no work-around was provided. Fortunately the DOS1 call support was left in, which simply calls device 0, which CDOS translates to Printer 0, and off we go. Much simpler than sending a 1 and expecting a 4 back, as the DOS2 calls have it, but those new calls were also there at IBM's request to make way for an MS-DOS simplified network and avoid confusing it! Ah, me....

It seems an easy fix if you know the code and I expect it will be fixed in future versions, and now there's a work-around, but I am concerned about that over-reaction. Not because it was personal, but because a dominant company in the S-100 field actually went to great pains to tell an end user in no uncertain terms that he was irrelevant to them and should get out of their world. S-100 still has the vitality it has because of the dedication and enthusiasm of some of the best, most creative minds in this business, but we are a tiny minority in the computer business. We must fight and promote or die, but that's the breaks of the game, but all of us, from manufacturers to OEMs to integrators, third-party suppliers, and end users, must be a part of the fight. We might win, lose, or hold our own, but it would be an unconscionable tragedy if S-100 died a suicide.

Multi-User Systems



- Word Processing, Accounting, Database Management Software
- Needs Analysis, System Installation, User Training
- Application Software Design in DATAFLEX
- On-site Service and Support

COCHLIN COMPUTER SYSTEMS, Inc.

330 Townsend Street #107, San Francisco, CA 94107
An Authorized CompuPro System Center

(415) 495-5999

68k notes

TEST YOUR MEMORY

Jay Vilhena

This is the new column for users of 68000-based systems. In this column, I will publish anything that might be of interest to readers using or planning to use 68000 S-100 computers — that is any S-100 system using 68000, 68010, 68020, or related CPUs.

Basically, this is neither a technical column nor a news column. Simply a collection of 'stuff' that might be useful. I'll include code for useful programs, tips, and a few reviews of hardware or software that might be of interest. I'll include whatever I can find. Your help is very welcome. If you have a 68K program, a few hints, or other useful information that you would not mind sharing with other 68000 users, we would all appreciate it. Please send it to 68K Notes, *S-100 Journal*, PO Box 1914, Orem, UT 84057, or give me a buzz at 801-373-0696.

I do nearly all of my computer work on a 68000 system running CP/M-68K. Yes, CP/M-68K is at the bottom of the 68000 operating systems hierarchy, but so what? It does nearly everything I want or need (except multitasking which ***everybody*** needs), I like it, it has a reasonably decent mostly-UNIX-compatible C compiler (which comes for *free*, mind you), and it is the ***only*** hardware-independent 68000 operating system that is reasonably priced.

Naturally, on the software side, you'll see here a lot of info for CP/M-68K. However, do not get the idea that this is going to be exclusively a CP/M-68K column because it is not (it's not even an operating systems column, that's a few pages back in the Multiuser OS). I will also include

RAMTEST.S LISTING
PROGRAM WRITTEN BY DON PANNELL

```
*file: ramtest.s            CP/M-68K TPA RAM test program - version 1.1
*last modified:   8/15/85            by: Don Pannell, Peak Electronics
*
start:
   move.w   #9,d0       *d0=code to display a message
   move.l   #startmsg,d1   *d1=address of the message to display
   trap   #2       *display the message
   bra   gettpa      *branch around signon message text
*
startmsg:
   dc.b   cr,lf,'Peak Electronics CP/M-68K TPA RAM test program.'
   dc.b   cr,lf,'Program Version 1.1 - Copyright (C) 1985'
   dc.b   cr,lf,lf,'This program will continuously test all available'
   dc.b   cr,lf,'memory.  Long word accesses are used for fastest'
   dc.b   cr,lf,'program execution.  Type a ^C to end this memory test.'
   dc.b   cr,lf,lf,'Current TPA start address = $'
htpamsg:
   dc.b   cr,lf,'            & end address = $'
ltestmsg:
   dc.b   cr,lf,lf,'The Starting test address is moved higher than the'
   dc.b   cr,lf,'TPA start so that there is room for this program.'
   dc.b   cr,lf,'The ending test address is moved lower than the TPA end'
   dc.b   cr,lf,'address so that there is room for the stack and/or DDT.'
   dc.b   cr,lf,lf,'Starting Test address = $'
htestmsg:
   dc.b   cr,lf,'& Ending Test address = $'
   even         *assure next op is word aligned
*
gettpa:
   move.w   #63,d0       *d0=code to get/set TPA limits
   move.l   #tpab,d1    *d1=address of the TPA parameter block
   clr.w   tpab       *set parameter word to GET TPA limits
   trap   #2       *execute the BDOS get TPA limits
*
   move.l   tpab+2,d2   *d2=low TPA address
   bsr   longout      *display this systems low TPA address
   move.w   #9,d0       *d0=code to display a message
   move.l   #htpamsg,d1   *d1=address of the high tpa msg
   trap   #2       *display the message
   move.l   tpab+6,d2   *d2=high TPA address + 1
   sub.l   #1,d2       *adjust out the + 1
   bsr   longout      *display this systems high TPA address
*
   move.l   #buffer,d4   *d4=1st avail addr for test
   add.l   #3,d4       *add 3 to help in adjusting to long offset
   and.l   #$fffffffc,d4   *adjust start address to long offset
   move.l   d4,a2       *a2=saved low test address
   move.w   #9,d0       *d0=code to display a message
   move.l   #ltestmsg,d1   *d1=address of the low test RAM addr msg
   trap   #2       *display the message
   move.l   d4,d2       *d2=address value to display (low addr)
```

```
    bsr    longout      *display the low addr value
*
    move.l    a7,d3       *d3=the value of the current stack pointer
    sub.l    #$100,d3    *sub value to leave room for stack data
    and.l    #$fffffffc,d3   *adjust end address to long offset
    move.l    d3,a1       *a1=saved high test address
    move.w    #9,d0       *d0=code to display a message
    move.l    #htestmsg,d1    *d1=address of the high test RAM addr msg
    trap   #2      *display the message
    move.l    d3,d2       *d2=address value to display (high addr)
    bsr    longout      *display the high addr value
*
    clr.l    d5       *init the pass counter to zero
    clr.l    d6       *init the error counter to zero
    move.l    #$12345678,d7    *init the data value
*
*********************************************************************
*         main test loop
*    d1.l=action message
*    d4.l=read data
*    d5.l=test pass number
*    d6.l=error counter
*    d7.l=current test data
*    a0.l=current test address
*    a1.l=high test address (not to be changed)
*    a2.l=low test address (not to be changed)
*    a7.l=current stack pointer
*********************************************************************
loopbegin:
    bsr    disppass    *display the pass and error info
    move.l    a2,a0       *a0=first location to test
    move.l    #initpass,d1    *d1=address of this passes msg
initlp:
    move.w    a0,d0       *d0=ls word of next addr to write to
    and.w    #$0fff,d0    *see if a 1000 hex boundary was just crossed
    bne    coninit      *if not continue with the init code
    bsr    dispprog    *display the current progress
coninit:
    move.l    d7,(a0)+    *write normal data to memory
    rol.l    #1,d7      *scramble the normal data
    add.l    #1,d7      *and scramble it some more
    cmp.l    a1,a0      *see if all of memory is initialized
    bls    initlp      *continue if not done
*
pass1:
    move.l    #pass1msg,d1    *d1=address of this pass msg
pass1lp:
    move.w    a0,d0       *d0=ls word of next addr to write to
    and.w    #$0fff,d0    *see if a 1000 hex boundary was just crossed
    bne    conpass1    *if not continue with the pass1 code
    bsr    dispprog    *display the current progress
conpass1:
    sub.l    #1,d7      *do 1st half of data unscramble
    ror.l    #1,d7      *finish with the data unscramble
    move.l    -(a0),d4    *d4=data written to memory
    cmp.l    d4,d7      *see if the data matches
    beq    passlok      *if a match skip the error display
    bsr    disperror    *if no match, display the error
passlok:
    move.l    d7,d4       *insure d4 has correct expected data
    not.l    d4      *invert the expected data value
    move.l    d4,(a0)       *and write the inverted data back
    cmp.l    a2,a0      *see if all of 1st pass is done
    bhi    pass1lp      *continue if not done
*
pass2:
    move.l    #pass2msg,d1    *d1=address of this pass msg
pass2lp:
    move.w    a0,d0       *d0=ls word of next addr to write to
    and.w    #$0fff,d0    *see if a 1000 hex boundary was just crossed
    bne    conpass2    *if not continue with the pass1 code
    bsr    dispprog    *display the current progress
conpass2:
    move.l    (a0),d4       *d4=data written to memory
    not.l    d7      *invert the expected data
    cmp.l    d4,d7      *see if the data matches
    beq    pass2ok      *if a match skip the error display
    bsr    disperror    *if no match, display the error
pass2ok:
    not.l    d7      *reset d7 to normal data value
```

whatever useful goodies I can collect for users of UNIX, OS-9, d/OS, and other 68000 installations.

So, if you are in one of these last groups (or any other S-100 68000 group) and you have something (hints, programs, tricks, anything useful) that you do not mind sharing, please do send it along. To spur your interest, I'll buy a very nice dinner to whoever sends in the best contribution for each issue ***plus*** I'll send you a 68000 T-shirt. (If you are too far away to have dinner with me, then I'll send you $50 so that you can enjoy one on me anyway.) ***And***, even if your contribution is not judged the best but I still publish it, I'll still send you the T-shirt.

Now let's see what nicety I've got for you in this issue. How about a little program for your CP/M-68K system that tests every user memory location in your system as many times as you wish? And, if it finds an error, it blurts out which memory address is the culprit. Nice, heh? That's what I thought too. This contribution comes from Don Pannell, our knowledgeable technical editor. *(Thank's Don! Say, didn't I already buy you a dinner recently? OoooK..., I'll send you the fifty... Oh! Yes, the T-shirt too.)*

To use this program, you do not really need to know much about it, nor about assembly language. Simply type the listing into a file that you name RAMTEST.S, making sure that there are no mistakes. If you do not want to type it in, send $20 to Don Pannell (PO Box 700112, San Jose, CA 95170-0112). He has agreed to send readers a copy of the program on disk (CP/M-68K 8-inch). The $20 are to cover the cost of the disk, postage, and Don's time.

Anyway, after you have the source code file RAMTEST.S, make sure that you have on the current disk drive the files called AS68.68K, AS68SYMB.DAT, and LO68.68K. These are your Assembler and Linker and they came with your CP/M-68K system. (*Note: In your system, the files AS68.68K and LO68.68K may have been renamed AS.68K and LO.68K; if so substitute accordingly below.*) Now type at your console:

AS68 -L RAMTEST.S

Wait until the prompt returns. The file RAMTEST.O has been added to your directory. Now type:

LO68 -R -O RAMTEST.REL RAMTEST.O
which creates the new file RAMTEST.REL. By the way, if you had never before assembled and linked an assembly language program, you have just now done so; that's basically all there is to it. You can find more information and options in your operating system programmer's manual if you wish.

If you order the program from Don, he will include the file RAMTEST.REL in addition to RAMTEST.S, so you don't even need to do the above steps. The file RAMTEST.REL can simply be copied (PIPed) to RAMTEST.68K and *voila* you've got a neat memory tester. Or, better yet, create a more compact and faster-loading RAMTEST.68K by using the utility RELOC.68K (also came with your system) instead of PIP, as follows:

RELOC RAMTEST.REL RAMTEST.68K
The RAMTEST.68K program tests your Transient Program Area, that is essentially all your computer memory not occupied by the operating system itself. The program will tell you what memory range it is testing. If either the read/write operation or a memory location is faulty, the program will display an error.

If every pass of the program detects an error at the same address location(s), you may have a faulty memory chip. While errors at random locations may indicate that the writing and reading of the CPU to and from memory may be flaky. Does your S-100 lack a terminated motherboard, for example? This could be a source of data errors.

All memory locations of your computer will be double-tested within a few seconds. However, for a rock-solid test of your system, Don recommends that you leave the program running overnight. I might also add that if you have several memory boards in your system, you may want to swap their address locations and run the program again so that the physical memory occupied by the operating system also gets tested.

The above information should allow anyone to assemble and operate this program. I've asked Don Pannell to describe in more detail what the program is actually doing. So, if you'll excuse me, I'll pass the

```
   move.l   d7,(a0)+    *and write the normal data back
   rol.l    #1,d7       *scramble the normal data
   add.l    #1,d7       *and scramble it some more
   cmp.l    a1,a0       *see if all of 2nd pass is done
   bls   pass2lp        *continue if not done
*
   add.l    #1,d5       *else, bump the pass counter
   bra   loopbegin    *run the test again w/different data
*
*
initpass:
   dc.b   cr,'Initializing  : $'
pass1msg:
   dc.b   cr,'Testing Pass 1: $'
pass2msg:
   dc.b   cr,'Testing Pass 2: $'
   even          *insure next op is word aligned
*
**********************************************************************
*      subroutine:  crlf
*   This routine is used to display a <cr> followed by a <lf>.
*   input:   nothing
*   output:   nothing
*   effect:   nothing destroyed
**********************************************************************
crlf:
   move.w   d0,-(a7)   *save reg d0
   move.l   d1,-(a7)   *save reg d1
   move.w   #9,d0      *d0=code to display a message
   move.l   #crlfmsg,d1   *d1=address of the cr lf message
   trap   #2      *start a new line
   move.l   (a7)+,d1   *restore d1
   move.w   (a7)+,d0   *restore d0
   rts         *done
*
crlfmsg:
   dc.b   cr,lf,'$'
   even          *insure next op is word aligned
*
**********************************************************************
*      subroutine:  disperror
*   This routine is used to display the address as well as the
* expected and read data whenever an error is detected.  The error
* counter, reg d6.l, is also incremented by 1.
*   input:   d4.l=read data
*      d7.l=expected data
*      a0.l=error address
*   output:   d6.l=d6.l + 1
*   effect:   nothing destroyed
**********************************************************************
disperror:
   move.w   d0,-(a7)   *save reg d0
   move.l   d1,-(a7)   *save reg d1
   move.l   d2,-(a7)   *save reg d2
   move.w   #9,d0      *d0=code to display a message
   move.l   #errmsg,d1   *d1=address of the error message
   trap   #2      *display the message
   move.l   a0,d2      *d2=address of the error
   bsr   longout      *display the error address
   move.w   #9,d0      *d0=code to display a message
   move.l   #expmsg,d1   *d1=address of the expected data message
   trap   #2      *display the message
   move.l   d7,d2      *d2=value of the expected data
   bsr   longout      *display the expected data
   move.w   #9,d0      *d0=code to display a message
   move.l   #readmsg,d1   *d1=address of the read data message
   trap   #2      *display the message
   move.l   d4,d2      *d2=the value of the read data
   bsr   longout      *display the error data
   bsr   crlf      *start a new line
   add.l   #1,d6      *bump the value in the error counter
   move.l   (a7)+,d2   *restore d2
   move.l   (a7)+,d1   *restore d1
   move.w   (a7)+,d0   *restore d0
   rts         *done
*
errmsg:
   dc.b   cr,lf,'===>Error occurred at $'
expmsg:
   dc.b   ' Expected data = $'
```

```
readmsg:
   dc.b    ' Read data = $'
   even          *insure next op is word aligned
*
***********************************************************************
*      subroutine:  disppass
*   This routine is used to display the number of test passes
* executed so far (reg d5) and the number of RAM test errors detected
* so far (reg d6).
*   input:    d5.l=# of completed test passes
*      d6.l=# of detected errors
*   output:   nothing
*   effect:   nothing destroyed
***********************************************************************
disppass:
   move.w    d0,-(a7)    *save reg d0
   move.l    d1,-(a7)    *save reg d1
   move.l    d2,-(a7)    *save reg d2
   move.w    #9,d0       *d0=code to display a message
   move.l    #passmsg,d1    *d1=address of the pass message
   trap   #2       *display the message
   move.l    d5,d2       *d2=the current pass counter
   bsr    longout      *display the pass value
   move.w    #9,d0       *d0=code to display a message
   move.l    #errlmsg,d1    *d1=address of the error message
   trap   #2       *display the error message
   move.l    d6,d2       *d2=the current error counter
   bsr    longout      *display the # of errors
   move.w    #9,d0       *d0=code to display a message
   move.l    #err2msg,d1    *d1=address of the end part of the message
   trap   #2       *display the message
   move.l    (a7)+,d2    *restore d2
   move.l    (a7)+,d1    *restore d1
   move.w    (a7)+,d0    *restore d0
   rts          *done
*
passmsg:
   dc.b    cr,lf,lf,'Completed $'
errlmsg:
   dc.b    'test passes with $'
err2msg:
   dc.b    'RAM errors detected.',cr,lf,'$'
   even          *insure next op is word aligned
*
***********************************************************************
*      subroutine:  dispprog
*   This routine is used to display the progress of the desired
* routine.  If called, the message pointed to be d1.l is displayed
* along with the value found in reg a0.l.  Polling for the ^C key
* for program termination is also performed from this routine.
*   input:    d1.l=addr of message to display
*      a0.l=value to display after the message
*   output:   nothing
*   effect:   nothing destroyed
***********************************************************************
dispprog:
   move.w    d0,-(a7)    *save reg d0
   move.l    d1,-(a7)    *save reg d1
   move.l    d2,-(a7)    *save reg d2
   move.w    #9,d0       *d0=code to display a message
   trap   #2       *display msg pointed to be d1.l
   move.l    a0,d2       *d2=desired value to display
   bsr    longout      *display the value
   move.l    (a7)+,d2    *restore reg d2
   move.l    (a7)+,d1    *restore reg d1
   move.w    (a7)+,d0    *restore reg d0
   rts          *done
*
***********************************************************************
*      subroutine:  longout
*   This routine converts the data found in register d2 to ASCII
* and sends it out to the console.  A single space is appended to the
* end (i.e., 9 bytes of data are sent out).
*   input:    d2.l=value to send
*   output:   ASCII value sent out
*   effect:   nothing destroyed
***********************************************************************
longout:
   move.w    d7,-(a7)    *save register d7
   move.w    #8-1,d7      *init d7 to loop for 8 characters
```

column over to him. See you next time! *(Thanks, Don. My fingers were getting a little tired.)*

(No problem, Jay. Oh... are we on the air...) Hi everyone! Here goes: "RAMTEST.S is a program designed to thoroughly test memory in any CP/M-68K system. It is written in 68000 assembly language, so it is both small and fast.

The program starts out by determining the size of the TPA (Transient Program Area) and exercises all available memory found there. Memory outside the TPA is not tested. After displaying the range of memory that is going to be tested, the program proceeds to test the memory using a pseudorandom data field in a MARCH pattern. Testing progress is displayed along with any error information. After the MARCH test pattern is complete, another one is started but this time a different pseudorandom seed number is used for the test data. The program progresses this way, continuously testing memory until a CONTROL C is typed at the console. A 'good' system should be able to run this program for hours without any errors (each test pass takes only seconds).

The MARCH test pattern starts out each test cycle by initializing memory to a known data pattern. This is usually done from low memory to high memory. Then, starting from high memory and going to low memory, the known data pattern is read, tested, complemented, and written back. The second and final pass reads the complemented data pattern, tests it, complements it again, and writes it back. This last pass goes from low memory to high memory.

The data is modified as it is being tested because this assures that there are no shorts or opens in any of the address lines, and it guarantees that each memory location is separate and unique from all the other memory locations. A memory test that simply fills all memory with zeros, reads them all back, refills memory with ones, and reads them all back, has succeeded in testing only one memory location. All the address lines could be shorted together and this kind of a test would still pass. This is why the MARCH pattern is used instead.

A few notes about CP/M-68K programs that use memory based on the

TPA-size BDOS call: First, the returned information is just that, information. In the program described, the returned data cannot be used for either the low or the high address limits. Instead, the low address limit is at the end of the program that is running (the next unused memory location). The high address limit is the stack pointer (A7) minus some space for the stack. If this high limit is used, then the program can be debugged with DDT because DDT will modify the stack pointer's value to compensate for the space DDT took in memory."
(I need a glass of water. Thanks.)

Z80 Z280 Z8000 HD64180 Z-SYSTEM ZCPR CP/M-2.2

S-100 Journal is planning to add a new regular column tentatively called *Into the 80s* to cover the Zilog and Zilog-like line of CPUs and compatible operating systems. The column will be user-oriented.

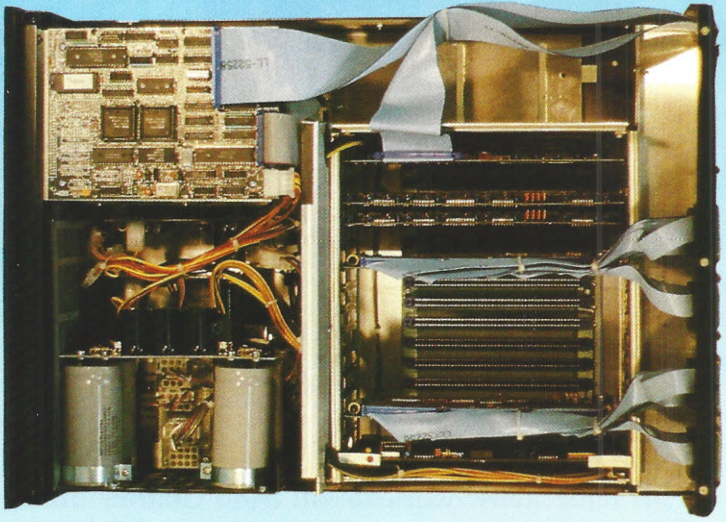
We are looking for one or more regular or semi-regular columnists. If you are interested, please send *S-100 Journal* a brief description of your background and experience with the Z80 line. Or call Jay Vilhena at (801) 373-0696.

The author(s) for this column should have regular access to one or more S-100 Z80-type machines and a solid knowledge of the Z-System and of CP/M-2.2.

```
lolp:
   rol.l   #4,d2       *place MSN to LSN
   bsr   hx2as       *convert and output d0's LSN
   dbra   d7,lolp       *see if done
*
   bsr   blankout   *append a space
   move.w   (a7)+,d7   *restore register d7
   rts         *done
*
**********************************************************************
*      subroutine:  hx2as
*   This routine converts the LSN in d2 to ASCII and sends it
* to the console.
*   input:   LSN of d0=value to convert and send out
*   output:   value converted and sent
*   effect:   nothing destroyed
**********************************************************************
hx2as:
   move.w   d2,-(a7)   *save register d2
   and.b   #$0f,d2      *mask out MSN of d2's LSW
   or.b   #$30,d2      *convert to ASCII (0 to 9)
   cmp.b   #'9',d2      *is result <= 9?
   ble   spfix      *if so, skip the letter fix
   add.b   #$07,d2      *else, convert to ASCII (A to F)
spfix:
   bsr   consoleout   *output d2 to the console
   move.w   (a7)+,d2   *restore register d2
   rts         *done
*
**********************************************************************
*      subroutine:   blankout
*   This routine is used to display a single SPACE character to
* the console.
*   input:   nothing
*   output:   ASCII space sent out
*   effect:   nothing destroyed
**********************************************************************
blankout:
   move.w   d2,-(a7)   *save register d2
   move.b   #' ',d2      *set d2.b to char value to send
   bsr   consoleout   *output the space
   move.w   (a7)+,d2   *restore d2
   rts         *done
*
**********************************************************************
*      subroutine:  consoleout
*   This routine is used to display the byte value found in reg
* d2 to the console.
*   input:   d2.b=character to send to the console
*   output:   character sent
*   effect:   nothing destroyed
**********************************************************************
consoleout:
   move.w   d0,-(a7)   *save register d0
   move.w   d1,-(a7)   *save register d1
   clr.w   d1      *insure MSB of target reg is zero
   move.w   #2,d0      *d0=code to do console output
   move.b   d2,d1      *d1=byte value to output
   trap   #2      *perform the console output
   move.w   (a7)+,d1   *restore register d1
   move.w   (a7)+,d0   *restore register d0
   rts         *done
*
**********************************************************************
*      start of Block Storage Segment
**********************************************************************
   bss         *start the block storage segment
tpab:
   ds   5      *reserve 5 words for the TPA parm block
buffer:
   ds   1      *1st available RAM location for test
*
**********************************************************************
*         EQU's
**********************************************************************
cr   equ   $0d
lf   equ   $0a
esc   equ   $1b
*
   end         *end of ramtest.s
```

— END OF RAMTEST.S LISTING —

NOEL The New Age of PC Cooperation Disk and Printer Server - Timesharing Multitasking - *Prices start at $5800*



The Hardware:

512K to 16Mb RAM
20Mb to 1Gb Disk
800Kb Floppy
8 to 40 RS232 Ports
External SCSI Port
MC68010/20 Processor, 8MHz
High Capacity Cooling
and Power Supply
One Year Warranty

Provides You With:

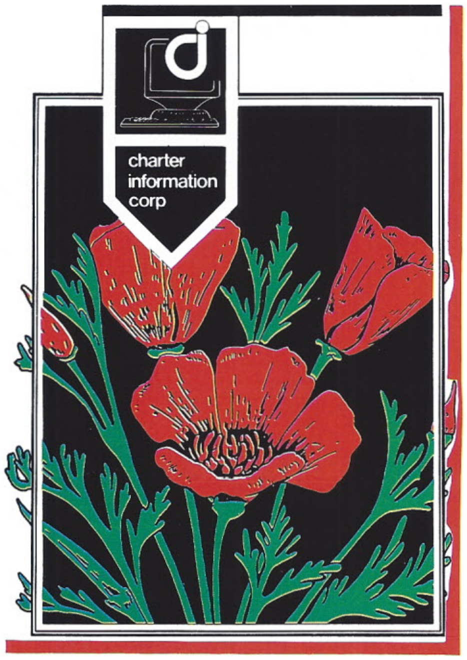
8 to 40 Users with dumb
terminals or PCs
Nearly limitless disk and
RAM capacities
Choice of O/S:
d/OS*, OS-9* and Mirage*
Compilers Available:
Extended Basic, C,
Cobol, Fortran 77,
UCSD Pascal
Application Software:
Extensive Library of
Sophisticated Packages

Powerful Multiuser Systems at PC Prices

Inner Access

1155-A Chess Drive, Suite D, Foster City, CA 94404 • (415) 574-8295 • Telex 494-3275 — INNACC

*Trademarks of d/Soft Ltd., Microware, and Sahara Ltd. respectively.



POPPY
brings compatibility to full flower.

Multi-User • PC-DOS • Turbo-DOS • S-100

Yes, they can co-exist!

And you'll love the way they look!

This remarkable workstation from Charter Information Corp gives you monochrome, color, and EGA capabilities all on one blooming board which fits into your multi-user Turbo-DOS system. In fact, you can fit up to sixteen POPPY boards inside a box for true multi-user capabilities. POPPY uses PC compatible monitors and keyboards which can be up to 200 feet from the central system.

Run all your favorite PC-style programs, including Lotus 1-2-3, Wordstar, Sidekick, Flight Simulator, Jet, and more under PC-DOS or Turbo-DOS on your S-100 system. And then share your peripheral resources among various users.

And if you're a reseller who hasn't made multi-user machines before, give us a call. We'll make the whole system for you in 3 to 10 days. If you missed seeing us at COMDEX, be sure to write or call for our free brochure. We think you'll agree that POPPY will really add to your computing landscape.

You've planted a seed in my mind.

☐ Please send me more information on the systems that you will integrate for me.

☐ Send information on being a dealer for your product.

Name ______________________

Title ______________________

Company ______________________

Address ______________________

City ______________________

State ____________ Zip ________

Telephone (____) ______________

Charter Information Corp

Where good ideas improve.

2421 Rutland
Austin, Texas 78758
(512) 835-1111

ADD OVERLAY TO YOUR 8-BIT BASIC INTERPRETER

Steven L. Salem

For years 8-bit microcomputer users have been restricted by the relatively small memories addressable by 8-bit microprocessors (generally 64K bytes or less). Large mainframe computers usually have some means of overlaying user-written programs, and this makes their large memories seem even larger. Some large applications programs for 8-bit systems use this technique as well. However, programming languages developed for 8-bit microcomputers generally lack this kind of feature. In light of the increasingly large memories that the new breed of 16-bit microsystems can use, is there any hope for the programmer who runs into the classic 'memory full' prolem with his 8-bit BASIC interpreter?

Such an overlay system would allow you to write and run programs many times larger than your computer's memory could accomodate at one time.

Surprisingly, there may be a solution for the BASIC programmer who has some knowledge of assembly language programming. Because of the structure of many BASIC interpreters, it may be relatively straightforward to develop an overlay processor that can be added to your BASIC language interpreter. Such an overlay system would allow you to write and run programs many times larger than your computer's memory could accommodate at one time.

In this article, I will first discuss background information on overlay techniques and interpreted languages. Then I will present a complete overlay processor written specifically for North Star BASIC, with examples of its use. This processor has been developed to maximize the speed of disk operations, by eliminating the need to read the disk directory for each overlay, and to minimize programming effort by automatically calculating the addresses for the overlay segments. It is also suitable as the basis for an overlay system for assembly language programs under the North Star operating system, and, with additional modifications, it can be adapted to other operating systems as well.

OVERLAYING VERSUS CHAINING

Overlaying and chaining are two common techniques that allow programs or sections of programs to reside on disk until they are actually needed in memory. At this point, they are read into the computer's active memory (i.e., RAM), replacing a previous program or segment that is no longer needed. Thus, groups of programs or routines that are too large to fit into memory at one time can be called up from the disk as they are needed. This essentially trades disk space for active memory space. These disk accesses, of course, may greatly slow the execution speed of the program; however, this may be the only reasonable way of executing many large programs.

Aside from these similarities, chaining and overlaying differ in ►

Steve Salem is a Physicist and Engineer. He has been involved with mainframes since 1966 and with microcomputers since 1980. Steve stubbornly refuses to give up any of his three S-100 computers.

significant ways. Chaining is the technique most often found in microcomputer BASICs. To use chaining, a program is loaded into memory and run normally. When the program has finished, or needs to call another program from disk, the new program is loaded into memory and entirely overwrites the original program.

This newer 'chained' program is completely independent of the original program and makes no use of any subroutines contained within it. In some cases, data from the original program can be shared with the new program. In others, such as with North Star BASIC, all the the original data and variables disappear along with the original calling program. In such cases, data can be shared between programs in one of two ways. Data may be written to a disk file by the first program and read back from that file by the second. Figure 1 depicts this approach. Alternatively, the BASIC function POKE (or FILL, in North Star BASIC) may be used to temporarily move data to an area of memory not used by either program, and PEEK (or EXAM) can retrieve the data as needed by the second program. Unfortunately, this approach doubles the memory required to hold the shared data and may negate much or all of the memory-saving benefits obtained by chaining, especially if large arrays are used.

Chaining is a useful way of executing sequential programs, especially when several programs are too large to reside simultaneously in memory. For example, the first program might be a 'preprocessor' of data to be further analyzed by a subsequent program. This first program would read and process an input data file, writing the output to a new disk file. Then it would chain the second program which would read the new data file and continue the processing.

Often, however, where iterative rather than sequential processing is required, chaining is not an efficient approach. One of the key problems is the large amount of disk accessing required for those versions of BASIC in which data cannot be shared between programs. In such a situation, each call of a chained program requires three disk accesses (not including directory reads). That is, data must be written to disk, the next program chained, and the data again read from the disk. If this needs to be done only once, as in a sequential program chain, the disk accessing is generally tolerable. However, in an iterative situation where two or more programs must chain each other a number of times, the disk access time and disk wear may become excessive. The time needed for three disk accesses may also be annoying when chaining is used in a menu-driven application. In this case, the user may sit for several seconds waiting for a response when a menu selection requires chaining to another program. Again, if variables can be shared in memory, this situation calls for only one disk read per chain, and is generally reasonable for both sequential and menu applications.

Overlaying provides a superior alternative for two of the above cases: as a replacement for chaining in the iterative situation, and in BASICs which do not allow shared (or common) data. In the general case, and in the program I will present shortly, overlaying allows specific subroutines or program segments to be loaded into memory right on top of (and thus replacing) segments already in memory. All program variables and data are retained in memory, and the main part (or 'root' segment) of the original calling program is retained as well, free to continue controlling the entire program. Therefore, the main program can continue where it left off but now using the new subroutines that have been overlaid into memory, replacing those routines no longer needed. Figure 2 illustrates how a typical series of overlays would work and can be compared with the example of chaining shown in Figure 1.

FEATURES OF BASIC INTERPRETERS

Let's consider a few features of BASIC language interpreters which will be important when we get to the details of our overlay program. The key concepts concern the line numbers required on each BASIC line and the structure of each program line in memory.

BASIC, unlike just about any other language, requires that you number each program line whether or not your program actually uses these line numbers (e.g., via GOTOs or GOSUBs). Generally, a compiled language such as FORTRAN requires line numbers (or labels) only on lines that must be referenced by other parts of the program. Each of these line numbers would be stored in a table during program compilation and used to locate the memory address of the start of that line in the compiled code. Then, when a FORTRAN GO TO is encountered, the program execution can continue by directly jumping to the beginning of the appropriate line.

A BASIC interpreter does not work this way. Instead, the interpreter must search through the program, line by line, looking at each line number (remember, all lines are numbered), until it finds the right one. This is the key that allows us to write a simple overlay system for such an interpreter. However, we need to understand a little about the way that BASIC lines are stored in memory before we can continue.

Since BASICs generally do not keep table pointers to the start of each line, a good BASIC interpreter needs to store each line in a way that allows quick scanning to find a desired line. Since each and every line must be searched to find the line required by a GOTO or GOSUB, it is undesirably time-consuming to read through each line to find the beginning of the next. Instead, each line begins with a pointer to the beginning of the next line, followed by the line number itself (usually stored as a two-byte hex number, low byte first). If the line number is not the one sought, the previously read pointer allows BASIC to jump over the rest of the line and go directly to the beginning of the next line.

This pointer can be of two types. In North Star BASIC, a single-byte pointer holds the total length of the line. When added to the address of the current line, the address of the next line is thus generated. Some other versions of BASIC actually begin each line with the two-byte address of the next line. In either case, the next line must start precisely at the location indicated, beginning with the new pointer and followed by the line number. Otherwise, BASIC would become hopelessly lost, unable to distinguish pointers and addresses from program statements themselves. Furthermore, in most BASICs each line also concludes with a unique byte which lets BASIC know where the line ends when it is actually interpreting and executing the line. In North Star BASIC this is a carriage return (0D hex); in some other versions it is the null byte (00). Finally, an ENDMARK (01 for North Star, sometimes two null bytes for other versions) follows the last line of a program.

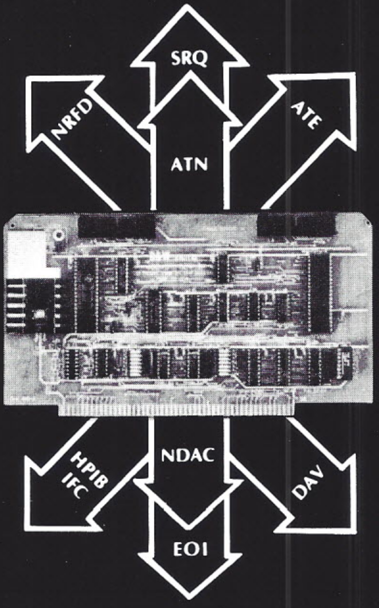
DETAILS OF THE NORTH STAR OVERLAY SYSTEM

Keeping the above discussions in mind, we can now discover the key concept of our overlay system. Since the BASIC interpreter searches blindly each time it needs a particular line, it does not care whether the location of any single line changes from time to time — as long as each line required by a GOTO or GOSUB is there *when it is needed*, and as long as all lines begin *exactly* where required by the pointer from the preceding line. Of course, line numbers must also be in ascending order and cannot be duplicated. Now, by referring to Figure 2, we can see that the process of overlaying merely requires the following:

1. When overlaying a program segment, the program must be executing somewhere other than in the region being overlaid. Generally this will be in the 'root' segment.
2. The new segment being loaded must start with the beginning of a program line (i.e., at the pointer), and it must be placed *exactly* after the end of an existing line in the program, so that the last existing pointer will point to the first line of the new program segment.
3. Since any new BASIC segment will end with an ENDMARK unless explicitly deleted, any parts of the original program that follow can no longer be reached. That is, if the program attempts to continue past the end of the new segment, it will stop upon reaching the new ENDMARK.
4. The new segment *must not* extend past the end of the original program, or the data which follow will be destroyed. Note that any program being loaded may be followed by 'garbage' since an integral number of disk segments will be loaded. This total size must fit within the original program length.
5. The first line number of the new segment must be greater than the last line number of the segment it follows.
6. The new segment may itself call overlays which can replace its tail end and any existing parts of the original program that follow. But remember, the original program end point must extend at least as far as the end of the last overlaid segment. REM statements may be used to pad the end of the original segment as far as necessary to meet this requirement.
7. Finally, note that machine language programs may also be loaded or overlaid into memory in this way, either within the BASIC program area or following the data area. These, of course, can be called by versions of BASIC that allow linkage to machine language routines.

Using the above concepts, we can write a simple overlay processor for versions of BASIC that use a single-byte line pointer containing the length of the current line. The added difficulties of assuring correct placement of addresses for those versions using actual addresses as line pointers will not be covered here. What must be done, however, is to ensure that the program segment being overlaid was orginally created within BASIC at the exact location where it will reside after being overlaid. Alternatively, we could include a short routine to convert these addresses after the new segment has been overlaid. This, in fact, is quite straightforward.

I will illustrate the above concepts by discussing a North Star overlay system I developed and have been using successfully for the past two years. It is an assembly language program (henceforth referred to as *the overlay program* or *the loader program)* consisting of two sections, each called separately from the BASIC program. The first is a 'preprocessor' which sets up a table of disk addresses for each program segment to be loaded from disk later. This is called once, at the beginning of the BASIC program execution. The second routine actually loads each BASIC program segment into memory where and when



THE 488+3

IEEE 488 TO S-100 INTERFACE

S-100 ⟺ 488

- *Controls IEEE 488 (HP1B) Instruments with an S-100 computer*
- *Acts as controller or device*
- *Basic and assembly language drivers supplied*
- *Meets IEEE 696 specification*
- *Industrial quality burned in and tested up to 125K bytes/sec under software control 3 parallel ports (8255-5)*
- *$375*

DWD

D&W DIGITAL, INC.
20655 Hathaway Avenue
Hayward, California 94541
(415) 887-5711

required by the BASIC program.

Now let us look at the overlay program itself. Listing 6 (starting on page 42) is the assembly language code for the complete overlay system (written in 8080 code for the PDS Makro assembler — Allen Ashley, 395 Sierra Madre Villa, Pasadena, CA 91107). Since it is written for North Star BASIC, under the North Star DOS operating system (any double-density or quad version), all disk and directory operations must be changed to function under another operating system. The entry point labeled INIT is the preprocessor, and the table space used to store program disk addresses is called BUFFR, at the very end of the listing. The table space and stack share this area, with the stack growing down from high memory, and the disk addresses starting at the low end. 40 stack bytes are needed, leaving room (in this case) for 21 program segment references, each requiring 11 bytes.

The preprocessor works this way: First the original BASIC program must load the file names of all necessary BASIC program segments into the buffer area. To do this, we set a BASIC variable to the address of this buffer area, and then POKE (or FILL) the file names into the buffer, spaced at intervals of precisely 11 bytes. This space allows 8 bytes maximum for the name, plus a comma and drive number (if needed for files on different drives), and a final carriage return, value 13 (0DH). After all file names have been entered, a 0DH or 00 byte must be entered as the first byte in the next file name location. This signals the preprocessor that no names remain. Note that the assembly program initializes the first byte of the buffer area to 0DH, thus causing the processor to return automatically if it is called erroneously before any names have been entered.

When called, this preprocessor looks up each name in the directory of the appropriate disk, and replaces the name with the disk address and length of the file. It does this by calling the standard DOS routine DLOOK which searches the directory for the selected file name. If a match is found, register HL will be pointing to the disk address of the file. This address is then stored in place of the first two bytes of the program name and is followed by the drive number. HL is then incremented to point to directory byte 13 which contains the length, in blocks, of BASIC programs. Since North Star disk read operations require file lengths in sectors, we divide the number of blocks by two before storing it in the buffer after the drive number. We could have used directory bytes 10-11 which contain the length, in sectors, originally allocated for the file; however, this length may be much greater than byte 13, the space actually filled by the BASIC program. If the program segment is a machine language program, you will first have to set directory byte 13. This can be 'faked' by assigning the file a file type of 1, and giving it an arbitrary (false) execution address whose low byte is the length, in blocks, of the machine program. This byte gets put automatically into byte 13 of the directory entry. You can then change the file to any type desired without affecting byte 13.

Once INIT has been executed, the disk directory will not have to be accessed again, unless some later program segment sets up an overlay of its own. As each segment is loaded, the disk information is read directly from the table and the program is loaded without ever needing to search the directory again. Thus program loading can be extremely fast — especially if sequentially loaded programs are stored physically close to each other on the disk. This is because the drive head will already be near the required track, and no time is wasted stepping out to the directory track and back again to the program tracks. In this way, small segments may load within a second, even with a drive initially off, and within a half-second from a drive already on.

With the table initialized, loading a program (segment) is fairly straightforward. The main BASIC program must pass two pieces of information to the loader program (entry point OVRLY in Listing 6), the location of the disk data for the file and the location in memory at which the new program will be loaded. Since North Star BASIC allows only one parameter to be passed to a machine language program in a CALL statement, the extra parameter must be POKEd into the overlay program before it is called. I have chosen to have the memory address POKEd into location ADRAM (see Listing 6), and the location of the disk data passed via the BASIC call. In this way, if all program segments load into the same RAM area, this address need be POKEd only once, and it will remain unchanged for all subsequent overlays. I will illustrate these POKE and CALL operations with examples later.

The operation of routine OVRLY is now quite straightforward. The disk address, drive number, number of sectors, and RAM address are loaded into registers HL, C, A and DE. Then a single call to the DOS routine DDCOM loads the selected program into the desired RAM area.

INSTALLATION

Before writing BASIC programs using the overlay system, we need a method of loading the overlay program into memory. In an earlier version, I simply POKEd each byte of the machine code from a BASIC routine into a reserved area of RAM. However, there is a simple way of actually attaching the program directly to the end of the BASIC interpreter, so that it is loaded into memory along with the interpreter itself.

The installation process is suggested by the North Star System Software Manual (Rev. 2.1, page O-6). That section discusses how to shorten the length of North Star BASIC by deleting the trig and exponential functions at the end. In our case, we wish to *lengthen* BASIC by adding our processor to the end. Following the North Star documentation, here's how to do it. First, find the end address (ENDBAS) of your BASIC. Version 5.1 ends at 5FC4H for an ORG (origin, or starting address) of 2D00H, version 5.2 ends at 421DH with an ORG of E00H, and version 5.5.0 (for DOS 2.1.1) ends at 4759H with an ORG of 1000H. If you don't know ENDBAS for your version, you can find its value stored in locations ORG+6 and ORG+7 of BASIC, low byte (least significant byte) first. You can either use the Monitor to read the hex values directly or use BASIC's PEEK (EXAM) function to read the decimal values at both locations.

Once ENDBAS is known, assemble

the overlay program code starting at the NEXT address. In Listing 6, I started at ORG 421EH, corresponding to version 5.2 BASIC at E00H. Then, using DOS, load BASIC into RAM at any convenient location, and load the assembled machine code for the overlay loader at the very next location following BASIC. Now, ENDBAS must be changed to reflect the increased length of our new BASIC. For this version, set ENDBAS equal to 43FFH either by using the Monitor or by entering BASIC and using POKE (FILL), as described in the System Software Manual (i.e., use FILL to change locations ORG+6 and ORG+7 to the low and high bytes of ENDBAS, which are now 255 and 67 decimal).

Finally, save the combined BASIC and overlay program back to the BASIC disk file, making sure the file is large enough to contain the extra length of the overlay program. In this case, the 54 blocks required by the original BASIC interpreter were large enough to hold the entire overlay program. In fact, I chose the length of the buffer region at the end to fit precisely on the disk. Note, however, that the buffer and stack regions don't actually have to fit on the disk since they are empty until initialized by a BASIC program. Make certain that the version of BASIC actually saved on the disk file contains the new value of ENDBAS and is followed by the overlay program, and your overlay system is ready to go.

USING THE OVERLAY SYSTEM

I will now show some examples of how to use your newly-installed overlay system and give a few hints for using such a system. BASIC programs using overlay will typically include three sections involving overlay operations. The first will contain addresses, DIM and DEF statements. The second will load program names and initialize the overlay buffer. The third will call the loader routine OVRLY whenever a new segment is to be loaded. Three key requirements determine what must go into the first of these sections:

1. A DIM statement may be executed only once for each array

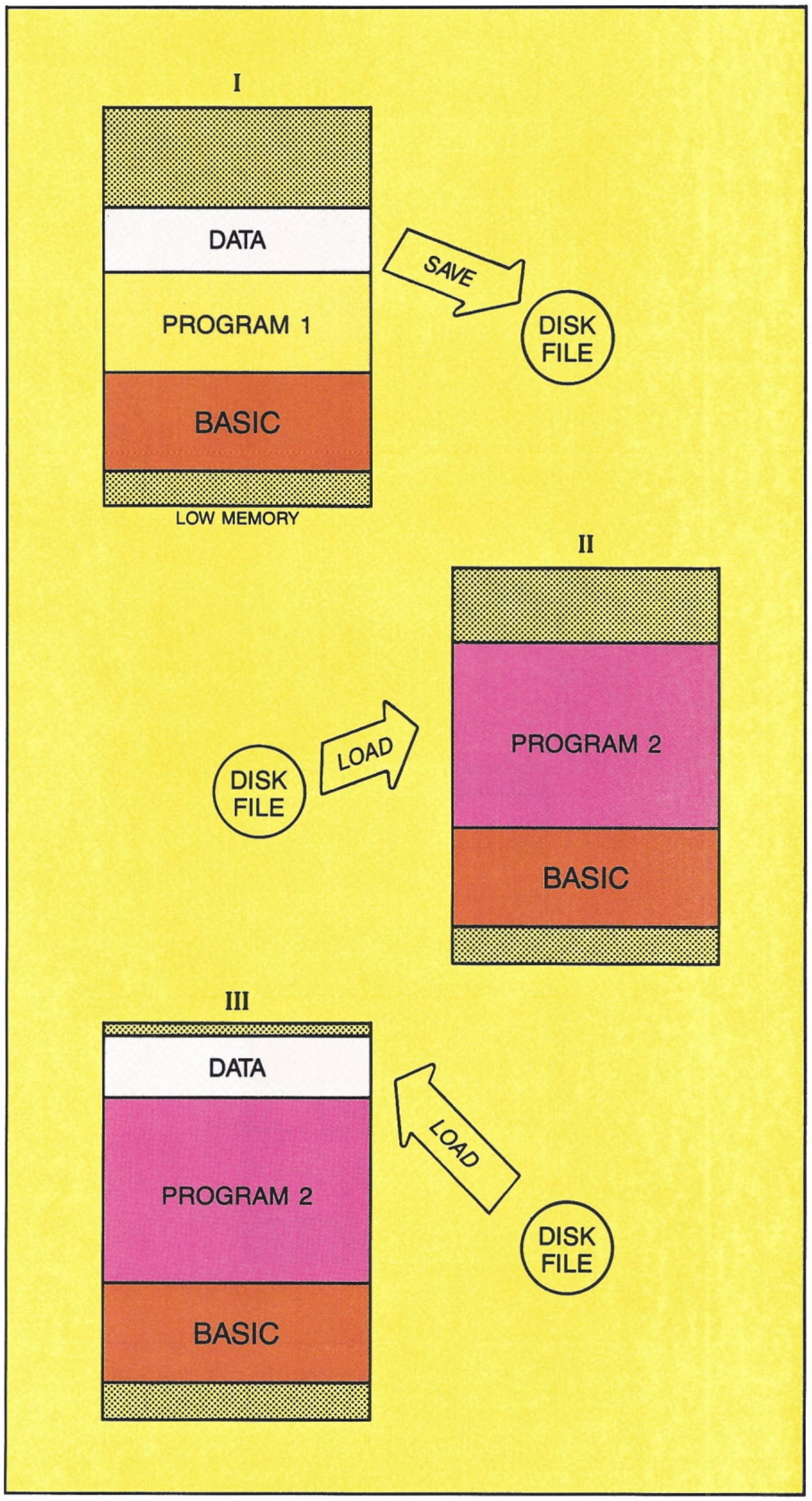


Figure 1. *An Example of Program Chaining. In step I, the original program is finished and ready to chain to Program 2. First, however, Program 1 (optionally) writes any data needed by Program 2 to a disk file. Next, Program 1 CHAINs Program 2 which is then loaded into memory on top of Program 1 and its data. Finally, in step III, Program 2 reads the data file written by Program 1 (if needed) and then continues. Note that in this example the second program is larger than the first, and that the data areas are in different locations.*

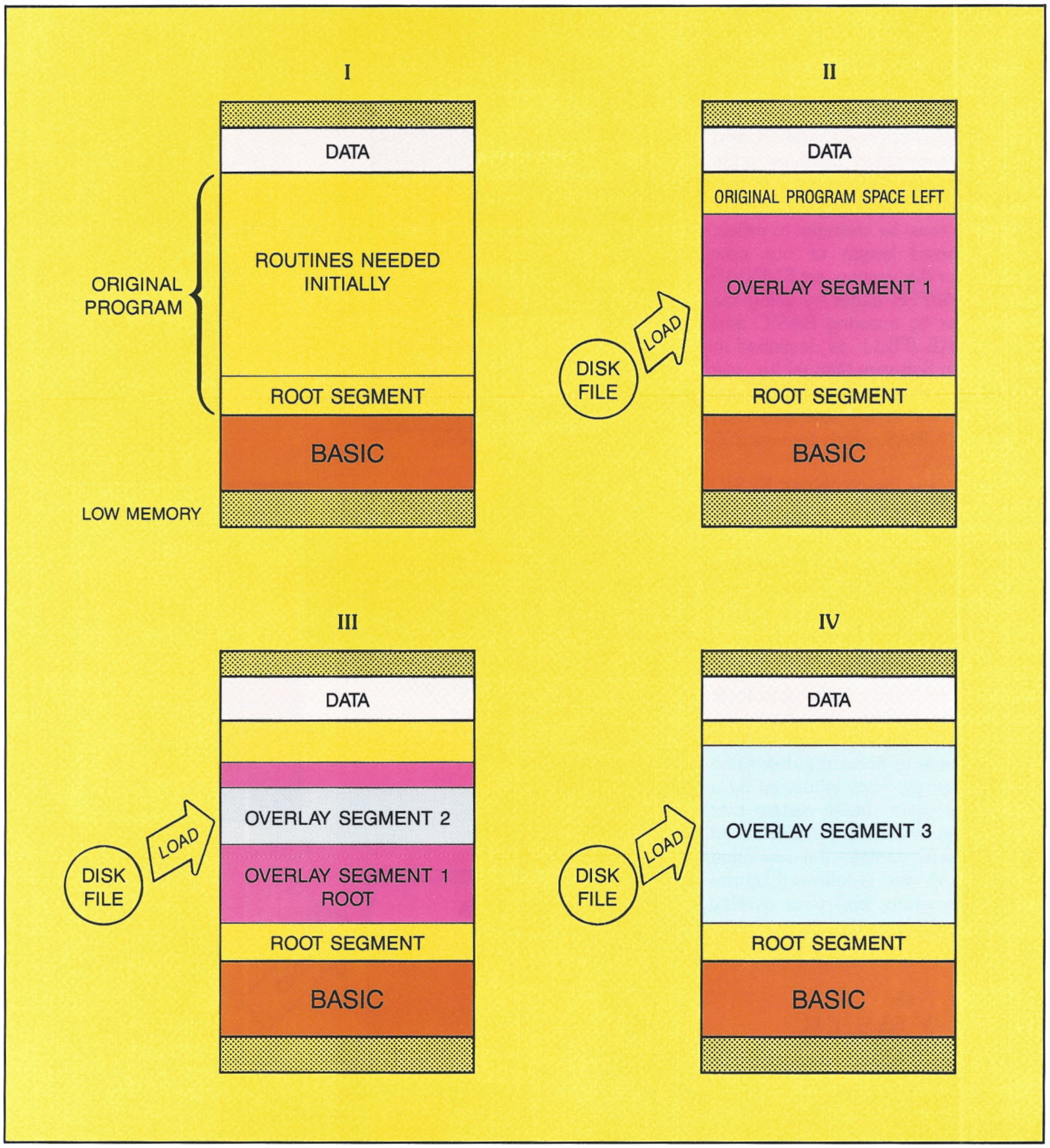


Figure 2. *An Example of Program Overlaying. In this example, the original program consists of a 'root' segment containing the main parts of the program (including all calls to the overlay program), and other routines which are needed only at first. As soon as the program is loaded and begins to run, BASIC begins to store the program variables in the data area immediately following the main program. This area must not be touched. When new routines are needed, overlay Segment 1 is loaded under control of the root segment. The root continues, possibly 'calling' (via GOSUB), or 'going to' the new segment. Then Segment 1 itself is assumed to need new routines, so it (or possibly the root) loads Segment 2. Notice that Segment 1 has not used all the existing program space up to the start of data. Segment 2 could have extended past the end of Segment 1, up to the start of data. When Segments 1 and 2 finish, the root segment then loads additional segments on top. In an iterative situation, this entire procedure could be repeated many times, terminating when a loop counter reached a set value, or when some convergence criterion was satisfied. In this example, Segment 3 begins where Segment 1 did; however, it could be loaded anywhere, as long as alignment requirements discussed in the text are met. In general, these include the restriction that old program lines which follow overlaid sections should not be accessed.*

or string used in a program.

2. Each DEF statement used in the original program or referenced in any segment later overlaid, must have been defined in the original segment.

3. A simple way is needed to determine the address at which each overlay segment is to be loaded.

The DIM restriction is a BASIC standard. In our case, it prohibits us from placing a DIM statement in a segment which may be overlaid and executed more than once. The simple solution is to place such statements at the beginning of the original segment where they will be executed once only.

The second restriction results from the operation of the North Star interpreter when a program is first run. At that time, the entire program is scanned, and the addresses of all function definitions are stored for later references to those functions. Thus, if a function is not yet defined when the first segment is loaded and run, it cannot be later loaded and referenced. To resolve this problem, we merely make sure that all DEF statements are placed in the original segment. Note that any DEF statement can be overlaid by later segments that do not need the function. The DEF statement can be overlaid again when needed, *as long as* it is loaded again at precisely the same address it occupied originally. Note also that we could actually change the function before replacing it at its original starting address.

The third requirement can be met using one of two general methods to determine the load address for each segment to be overlaid. You can either manually locate the specific address where the overlays will go and store it within the program (and manually change this address any time a modification in the program moves the load address), or you can have the program calculate the appropriate address automatically. Obviously, the second approach is simpler in most cases, but either approach will require some sleuthing to discover how to do it with a specific BASIC interpreter. Following are some specific details for North Star BASIC, Versions 5.1, 5.2, and 5.5.0 which should be generally valid for most other BASICs.

With the first approach, start by defining a variable in the root segment to hold the address, and set it equal to some arbitrary value (simply to hold the space necessary in your program to store the actual value later). Be sure to give this variable enough digits to hold the value when you know it (i.e., 5 digits). When you have developed your root segment and know where in the program the overlay segments will go, save the program and then delete all lines following the end of the root segment. Now, if you can locate the 01 byte which marks the end of this program in RAM, you will have the precise starting address of your overlay segment. Once you have this address, retrieve your complete program from disk (i.e., the version you saved before deleting the final lines), and enter the address in place of the dummy value you used before. But *make certain* you do not add or subtract any digits or spaces when doing this, or you will change the length of your program and thus move the location of the endpoint you just found.

You can find the location of the 01 ENDMARK in two ways. First, by leaving BASIC you can use the Monitor to search through the program lines in RAM, looking for the combination 0DH-01H that marks the end of the last program line with its trailing ENDMARK. However, since each change in your program requires you to find the new end address again, you need a faster way to do this. Fortunately, BASIC stores this address at all times, and you can read it without ever leaving BASIC by using EXAM as soon as you know where to look. Table 1 gives several key addresses for three common versions of North Star BASIC.

KEY ADDRESSES	BASIC VERSION		
	5.1	5.2	5.5.0
ORG	2D00H (11520)	E00H (3584)	1000H (4096)
LINESTRT	59EEH (23022)	3BF7H (15351)	4133H (16691)
ENDPROG	5BDEH (23518)	3E37H (15927)	4373H (17267)
BEGPROG	5BE0H (23520)	3E39H (15929)	4375H (17269)
ENDBAS	5FC4H (24516)	421DH (16925)	4759H (18265)

Table 1. *North Star BASIC key addresses in hexadecimal (and decimal). ORG and ENDBAS are the beginning and end of BASIC. The other three locations each represent the first byte of a two-byte location holding an address pointer.*

In Table 1, ENDPROG is the location which holds the address of the 01 ENDMARK at the end of the current program in memory, and BEGPROG holds the address of the start of the program. ENDBAS is the actual address of the end of each version of BASIC. For each of these versions, ENDPROG is exactly 998 decimal bytes (3E6H) before ENDBAS. If you are not sure of ENDPROG for your version, here's how to find it. First, load BASIC and run it. Don't load any program, just exit (BYE) after executing GO BASIC from DOS. This will leave BASIC with both ENDPROG and BEGPROG pointing to the byte following ENDBAS. (However, if you just load BASIC and don't actually run it, these locations will be empty). Then use the Monitor to search for locations within the BASIC interpreter which contain the value ENDBAS+1, low byte before high byte. You should find two consecutive 2-byte locations with this value. The first will be ENDPROG. Write down this address and convert it to decimal — you will need it later. How is this address used to find the end of your program? When you have a BASIC program loaded, use the decimal equivalent of ENDPROG and follow this example for version 5.2:

```
A = 15927
PRINT EXAM (A) + 256*EXAM(A+1)
```

Enter these statements in the direct mode so the result will print imme-

BASIC PROGRAM LISTINGS DISCUSSED IN TEXT

LISTING 1

```
100 FOR I = S TO E
110 A = EXAM(I)
120 B = EXAM(I)
130 C = EXAM(I+1)
140 IF A <> B THEN PRINT I, A, B, C
150 NEXT
160 END
```

LISTING 2

```
100 REM - ROOT SEGMENT  (ADDRESSES FOR BASIC 5.2)
110 DIM A$(10), F(100), G(100)  \ REM - ALL DIMS HERE
120 KO = 16983                  \ REM - INIT ENTRY POINT
130 LO = 16938                  \ REM - OVRLY ENTRY POINT
140 PO = 17127                  \ REM - ADDRESS ADRAM
150 XO = 15351                  \ REM - LINESTRT LOC
160 ZO = 17133                  \ REM - BUFFR START
170 DEF FNDO(DO)                \ REM - DELAY FUNCTION
180 FOR IO=1 TO DO\NEXT IO\RETURN IO
190 FNEND
195 GOSUB 900                   \ REM - LOAD PROG NAMES
```

LISTING 3

```
 900 REM - LOAD PROGRAM NAMES
 910 A = ZO              \ REM - FIRST SPOT IN BUFFER
 920 RESTORE 1100
 930 READ A$             \ REM - READ EACH NAME
 940 IF A$="END" THEN 1020
 950 Z=LEN(A$)
 960 FOR J = 1 TO Z      \ REM - STORE EACH LETTER
 970 FILL A+J-1, ASC(A$(J,J))
 980 NEXT J
 990 FILL A+Z,13         \ REM - ODH AFTER EACH NAME
1000 A=A+11              \ REM - NEXT SPOT IN BUFFER
1010 GOTO 930            \ REM - LOOP OVER ALL NAMES
1020 FILL A,13           \ REM - ODH IN FINAL LOCATION
1030 Q=CALL(KO,ZO)       \ REM - INITIALIZE BUFFER
1040 IF Q=O THEN RETURN \ REM - IF OK THEN DONE
1050 PRINT "DISK ERROR: HIT ANY KEY TO CONTINUE"
1060 A$=INCHAR$(O)       \ REM - WAIT TO HIT KEY
1070 RETURN
1100 DATA   "HELP","SORT","PROG1,2","PROG2,2","END"
```

LISTING 4

```
600 A1 = -1
610 IF A$(1,1) = "H" THEN A1 = 0
620 IF A$(1,1) = "S" THEN A1 = 1
630 IF A$(1,1) = "1" THEN A1 = 2
640 IF A$(1,1) = "2" THEN A1 = 3
650 IF A1 = -1 THEN 200 \ REM - RETRY MENU ON ERROR
660 GOSUB 898           \ REM - GET OVERLAY ADDRESS YO
670 Y2 = INT(YO/256)    \ REM - ADDRESS HIGH BYTE
680 Y1 = YO - 256*Y2    \ REM - ADDRESS LOW BYTE
690 FILL PO, Y1         \ REM - LOAD LOW BYTE
700 FILL PO+1, Y2       \ REM - LOAD HIGH BYTE
710 Q = CALL (LO, ZO+11*A1)  \ REM - CALL OVRLY
720 IF Q = O THEN 770   \ REM - CONTINUE IF OK
730 PRINT "DISK ERROR IN OVERLAY ATTEMPT"
740 PRINT "HIT ANY KEY TO CONTINUE"
750 A$ = INCHAR$(O)     \ REM - WAIT TO HIT KEY
760 GOTO 200            \ REM - RETURN TO MENU
770 GOSUB 2000          \ REM - CALL NEW SEGMENT
780 GOTO 200            \ REM - RETURN TO MENU
```

LISTING 5

```
600 FILL PO, Y1          / REM - LOAD LOW BYTE
610 FILL PO+1, Y2        / REM - LOAD HIGH BYTE
620 Q = CALL (LO, ZO)    / REM - LOAD PROG 1
630 GOSUB 2000           / REM - EXECUTE PROG 1
640 Q = CALL (LO, ZO + 11)  / REM - LOAD PROG 2
650 GOSUB 2000           / REM - EXECUTE PROG 2
660 Q = CALL (LO, ZO + 22)  / REM - LOAD PROG 3
670 GOSUB 2000           / REM - EXECUTE PROG 3
```

diately on your terminal. You will then have the decimal address of the 01 ENDMARK that follows your program, and you can enter it in place of the dummy address you originally used in your program.

In the second approach, the BASIC program locates its own overlay address by using the various pointers that any BASIC interpreter must use to keep track of its location within a program. However, since these locations are not generally published in users' manuals, we will have to find them, figure out how they work, and then decide how we can use them.

We would like to find a location within our BASIC interpreter that points to some part of the currently executing line of a BASIC program. Then we could insert an EXAM statement at the beginning of a program segment to find the RAM address of that exact line. The short BASIC program in Listing 1 will find any address within your BASIC interpreter whose contents change from line to line.

S and E are the starting and ending addresses of your BASIC interpreter. The results will be the addresses (I) of all locations within the interpreter whose contents change from line 110 to 120. You may find as many as a dozen addresses for some BASICs, and typically seven with versions of North Star BASIC. Most of these are useless. They hold characters read from the line, perhaps intermediate variables, etc. But some of the locations hold the low byte of a two-byte address. In these cases, EXAM(I+1) then gives the high byte, and EXAM(I)+256*EXAM(I+1) gives the decimal value of the address.

We wish to find each location (I) which is part of an address that points to the current BASIC line. The value printed out for C will help. If variable I is actually the address we want, then the value C=EXAM(I+1), the high byte of that address, should be within one or two of the high byte of the address ENDBAS. For each likely value of I, compute the hex addresses given by A+256*C and B+256*C, then exit from BASIC while leaving your test program intact in memory. Now use your Monitor (or DDT, or whatever debugger you have handy) and look at the BASIC program lines in RAM to see what part of each line these addresses point to. If this would destroy the BASIC program in memory, then save the entire BASIC interpreter and program to a disk file, and reload it into memory at a location which will not be overwritten.

Remember the previous comments about the memory image of BASIC lines. If you can't make sense out of the lines in memory, go back and add REM statements between lines, then repeat the process. What you are looking for is some value of I which gives a pointer to a specific location on each line. You may find some values that point to the EXAM statement, and some that point to the byte following the end parenthesis. You should also find one address that consistently points either to the leading byte of the line, the line number, or the first byte of the statement itself. For North Star BASIC, the location I have called LINESTRT in Table 1 points to the first byte of each statement. Thus, for line 110, EXAM(LINESTRT) points to A, and for line 120 it points to B.

Once you have located a value of I which holds a useful pointer, here's how to use it. Consider the line:

```
1000 Y0 = EXAM(I) + 256*EXAM(I+1)
```

For the appropriate value of I, the resulting decimal address points to a specific point on the line. We must now subtract a fixed number to find the first byte of the line. For North Star BASIC, we must subtract 3 plus the number of spaces we've inserted between the line number and 'Y0.' This is because the line begins with the length byte, the two-byte line number, and some number of spaces (if any) before the actual beginning of the statement. If you always use a single space after the line number, then the line:

```
1000 Y0 = EXAM(I) + 256*EXAM(I+1) - 4
```

returns a value for Y0 which is the address of the first byte of that program line in memory *at that exact moment.* This is just what we need for our overlay system — a line that always knows its current RAM location, even if it is moved around by overlays.

How can you use such a line? First store the value you've found for I (this value is called X0 in the examples below). Then insert the above line, with a suitable line number, after the last line of the program segment you wish to remain in memory. That is, put this new line where the overlay segment will begin. If you follow this line with a RETURN, you can use a GOSUB to find the RAM address of the line, and then use this address when you call OVRLY to load the next segment.

Some Examples

Now let's see how the first few lines of a BASIC program might look, using the above concepts. See Listing 2.

In this example, I first dimensioned all arrays and strings for the entire program. Then I placed all addresses required by the later BASIC routines that call the overlay routines. K0 is the decimal address of the assembly routine INIT (4257H, in this case), and L0 is the address of the assembly routine OVERLY (422AH). P0 is location ADRAM, near the end of the assembly routines, which will hold the starting address of the overlay region. X0 is location LINESTRT, which our program will use to find the overlay address needed to POKE into location P0. Finally, Z0 is the address of the start of the assembly routine buffer (BUFFR), which will first hold the names of the BASIC files to be looked up in the directory, and which will thereafter hold the disk addresses for these programs. Function FNDO is a simple delay function illustrating the possible placement of functions in the root segment. Finally, line 195 calls the subroutine which loads the program names and initializes BUFFR.

Listing 3 illustrates one way to load program names and initialize the buffer. This subroutine loads each program name, letter by letter, into the buffer, and then calls INIT to initialize the buffer with the disk addresses. It first sets variable A to the starting address of the buffer (location Z0), then it increments variable J and adds it to A, to point to each successive location that will store a character of the program name. The string A$ holds the program name, and it is split up, character by character, using the North Star function A$(J,J), which picks out the Jth character of the string. Each character, from J=1 to the end of the name, is POKEd into successive locations, addressed A+J−1, using the FILL function. At the end of each name, a carriage return is stored in the buffer (statement 990) and address A is moved up 11 spaces to the beginning of the location for the next name (statement 1000). This routine loops over all names listed in the DATA statement(s) until the name 'END' is reached. You can use a FOR loop or any other way to enter a list of names. Be sure to jump to a statement such as 1020 at the end, to place the required 0DH marker in the first space of the program-name position following the last program name.

Now, statement 1030 CALLs routine INIT, which starts at location K0, and requires the address of the first program name, here stored as variable Z0. I did not imbed this address in the assembly routine because this method allows succeeding segments to set up their own overlay program names using other areas of the buffer. Such later segments would have their own BASIC routines. These routines would load program names into other areas of the buffer and would CALL INIT using the same K0 but a different value for Z0.

The value of Q returned by INIT is 0 if all program names were found in the appropriate disk directories, and 1 if some names were not found. Any program name not found will be printed out by INIT, which will also set a program length of 0 for that program so that any attempt to load that program with routine OVRLY will cause nothing to happen. In this particular example, the BASIC program will simply continue if no errors occurred, and it will pause and wait for you to hit a key before continuing if an error did occur. This gives you a chance to read the error messages if, for example, the next section of your program would erase the screen before continuing. Notice the program names I have assumed in the DATA statement. Often I make use of HELP screens in programs. For large programs, these help screens take up too much space in memory, but they are perfect examples of routines which can be overlaid when needed. Note also that two segments, called PROG1 and PROG2, are located on drive 2. These might be routines under development on another disk, to be called up by an already existing driver program loaded in drive 1. Just make certain that enough space is taken up in memory by the original program, so that these new segments fit without destroying the variable storage area which follows.

At last, we are ready to see how such programs can be loaded and used by the overlay system. The routine in Listing 4 is representative of menu-driven applications, and it uses the four program names of the previous example.

This example assumes that a menu has been displayed by Section 200 of the program, and that the user has entered a selection stored in string A$. Lines 600-650 set variable A1 to indicate which of four legal entries was made (e.g., 'HELP,' 'SORT,' etc.), or whether an illegal name was selected (A1= −1). If an existing program was selected, then line 660 gets the overlay load address Y0, lines 670-680 convert it into two single bytes, low byte and high byte, and lines 690-700 load them into location ADRAM in the assembly program. Line 660 assumes that we've inserted two lines just before line 900:

```
898 Y0 = EXAM(X0)+256*EXAM(X0+1)−4
899 RETURN
```

Ultimately, the overlay segments will begin where line 898 is, replacing it and all lines following. Just make sure that any program lines just before line 898 either end with a RETURN or skip around these two lines.

After the load address has been found and loaded into the assembly program, line 710 calls routine OVRLY, at address L0, and points to the disk information stored for the specific program chosen, at address Z0 + 11 * A1. For example, disk data for program 'HELP' (A1=0) is stored at Z0, the start of BUFFR, and data for program 'SORT' is at Z0 + 11. If the program loads correctly, then line 720 causes the program to continue by calling the new routine (GOSUB 2000), and then returning to the menu when done. If an error was detected, due to either an illegal program name (line 650) or a disk error (lines 730-760), the menu routine is reentered without executing the GOSUB.

The next example (Listing 5) is typical of programs that process data sequentially (or iteratively) and need to overlay a specific sequence of routines. In this example, each routine is loaded and executed in turn. I've assumed that all new segments start with line 2000, although they could start with different line numbers. I've also omitted any error checking that you might wish to perform.

As you might imagine, there are many helpful tricks in using such a system. For example, I've put Section 900, the BASIC routine which loads the program names, after the routine which calls OVRLY (Section 600). This allows us to load another overlay segment over Section 900 after we've executed this section, while saving Section 600 to do subsequent program loading. There are also many ways of loading and executing assembly language routines, or even routines compiled by other high-level languages. Also, remember that any segments loaded towards the end of the BASIC program are free to use any of the routines at the beginning of the program, for example, the delay function (line 170) or other sections referenced by GOSUBs or GOTOs. You can even develop a menu-driven routine which will allow you to call up and run just about any BASIC program on a disk, subject to such restrictions as not repeating DIM statements.

Once you have installed the overlay program at the end of your BASIC interpreter, you may find yourself, as I have, storing this new version of BASIC on all your disks. As soon as you develop a few of your own routines to access the overlay routines, you may never again need to fear the dreaded 'memory full' message.

LISTING 6 — PROGRAM OVERLAY — VS 2.0

8080 ASSEMBLY LANGUAGE LISTING OF OVERLAY LOADER AND INITIALIZATION ROUTINES
PROGRAM DEVELOPED BY STEVEN L. SALEM — COPYRIGHT©1987 BY STEVEN L. SALEM

```
0000                 ;
0000                         ORG     0421EH
421E                 ;
0100                 DOS     EQU     0100H
421E  C32A42                 JMP     OVRLY
4221                 ;
4221  C30D01         DCOUT   JMP     DOS+0DH
4224  C31C01         DLOOK   JMP     DOS+1CH
4227  C32201         DDCOM   JMP     DOS+22H
422A                 ;
422A                 ;   ENTRY POINT TO LOAD BASIC OR MACHINE PROGRAM FROM DISK
422A                 ;   POINTER TO PROGRAM DISK ADDRESS MUST BE
422A                 ;     PASSED IN REGISTER DE (VIA BASIC CALL)
422A                 ;
422A  210000         OVRLY   LXI     H,00    ; CLEAR HL
422D  39                     DAD     SP      ; LOAD BASIC SP INTO HL
422E  22E942                 SHLD    BASSP   ; SAVE BASIC SP
4231  310044                 LXI     SP,STACK ; SET NEW SP
4234  EB                     XCHG    ;         LOAD DE INTO HL
```

```
4235  5E              MOV   E,M     ; SET DE=DISK ADDRESS
4236  23              INX   H
4237  56              MOV   D,M
4238  23              INX   H
4239  7E              MOV   A,M     ; LOAD DRIVE NUMBER
423A  C680            ADI   80H     ; DOUBLE DENSITY FLAG
423C  4F              MOV   C,A     ; STORE DRIVE # + DENSITY BIT
423D  23              INX   H
423E  7E              MOV   A,M     ; LOAD NUMBER OF SECTORS
423F  A7              ANA   A       ; SECTORS=0?
4240  37              STC   ;         SET CARRY
4241  CA4D42          JZ    EXIT1   ; JUMP IF ZERO, WITH CARRY SET
4244  2AE742          LHLD  ADRAM   ; RAM ADDRESS FOR PROGRAM OVERLAY
4247  EB              XCHG  ;         SWITCH RAM AND DISK ADDRESSES
4248  0601            MVI   B,1     ; DISK READ FLAG
424A  CD2742          CALL  DDCOM   ; LOAD PROGRAM FROM DISK
424D  2AE942   EXIT1  LHLD  BASSP   ; RETRIEVE BASIC STACK POINTER
4250  F9              SPHL  ;         RESTORE TO STACK
4251  210000          LXI   H,00    ; CLEAR HL
4254  D0              RNC   ;         RETURN IF NO ERROR
4255  2C              INR   L       ; IF ERROR, SET FLAG
4256  C9              RET   ;         RETURN WITH FLAG IN HL
4257           ;
4257           ;   **** ENTRY POINT TO INITIALIZE PROGRAM PARAMETERS ****
4257           ;
4257  210000   INIT   LXI   H,00    ; CLEAR HL
425A  39              DAD   SP      ; LOAD BASIC SP INTO HL
425B  22E942          SHLD  BASSP   ; SAVE BASIC SP
425E  310044          LXI   SP,STACK ; SET NEW SP
4261  EB              XCHG  ;         LOAD DE INTO HL
4262  AF              XRA   A       ; SET A=0
4263  32EB42          STA   ERFLG   ; CLEAR ERROR FLAG
4266           ;
4266           ;  LOOP OVER ALL PROGRAM NAMES
4266           ;
4266  7E       LOOP   MOV   A,M     ; 1ST CHAR. OF PROGNAME
4267  FE0D            CPI   0DH     ; <CR>?
4269  CAAE42          JZ    EXIT    ; DONE IF <CR>
426C  A7              ANA   A
426D  CAAE42          JZ    EXIT    ; DONE IF 00
4270  E5              PUSH  H       ; SAVE PROGNAME POINTER
4271  3E01            MVI   A,1     ; SET DEFAULT DRIVE NUMBER
4273  CD2442          CALL  DLOOK   ; LOOK FOR PROGNAME MATCH
4276  DA9642          JC    FAIL    ; SET ERROR FLAG IF NO MATCH
4279           ;
4279           ;   - ELSE HL POINTS TO DISK ADDRESS (DIR BYTE 8)
4279           ;
4279  4E              MOV   C,M     ; LOAD DISK ADDRESS INTO BC
427A  23              INX   H
427B  46              MOV   B,M
427C  EB              XCHG  ;         SAVE ADDR POINTER IN DE
427D  E1              POP   H       ; RETRIEVE PROGNAME POINTER
427E  71              MOV   M,C     ; STORE DISK ADDRESS
427F  23              INX   H
4280  70              MOV   M,B
4281  23              INX   H
4282  77              MOV   M,A     ; DRIVE NUMBER OF PROGRAM
4283  23              INX   H
4284  E5              PUSH  H       ; SAVE PROGNAME LOC+3
4285  EB              XCHG  ;         RETRIEVE DISK ADDR POINTER
4286  110400          LXI   D,04
4289  19              DAD   D       ; HL=HL+4: POINT TO DIR BYTE 13
428A  7E              MOV   A,M     ; NUMBER OF BLOCKS
428B  3C              INR   A
428C  1F              RAR   ;         A=A/2 (SECTORS)
428D  E1              POP   H ;         RETRIEVE PROGNAME LOC
```

```
428E  77        RENTR  MOV   M,A     ; STORE NUMBER OF SECTORS
428F  110800           LXI   D,08
4292  19               DAD   D       ; ADVANCE TO NEXT PROGNAME
4293  C36642           JMP   LOOP    ; LOOP OVER PROGNAMES
4296                ;
4296                ;  IF NO MATCH TO PROGNAME, PRINT ERROR MESSAGE
4296                ;    AND STORE 0 AS PROGLENGTH
4296                ;
4296  21D742    FAIL   LXI   H,TEXT
4299  CDB642           CALL  OTTXT   ; PRINT TEXT
429C  E1               POP   H       ; RETRIEVE PROGNAME POINTER
429D  E5               PUSH  H
429E  CDB642           CALL  OTTXT   ; PRINT PROGNAME
42A1  3E01             MVI   A,1
42A3  32EB42           STA   ERFLG   ; SET ERROR FLAG
42A6  E1               POP   H       ; RETRIEVE PROGNAME POINTER
42A7  23               INX   H
42A8  23               INX   H
42A9  23               INX   H
42AA  AF               XRA   A       ; A=0: PROGLENGTH=0
42AB  C38E42           JMP   RENTR   ; CONTINUE IN MAIN LOOP
42AE                ;
42AE                ;  PROGRAM DONE.  RESTORE BASIC SP AND CHECK ERROR FLAG
42AE                ;
42AE  2AE942    EXIT   LHLD  BASSP   ; RETRIEVE BASIC SP
42B1  F9               SPHL  ;         RESTORE TO STACK
42B2  2AEB42           LHLD  ERFLG   ; LOAD 2 BYTE ERROR FLAG
42B5  C9               RET   ;         RETURN WITH FLAG IN HL
42B6                ;
42B6                ;  ** TEXT OUTPUT ROUTINE
42B6                ;      HL POINTS TO TEXT
42B6                ;      00 MARKS END OF TEXT
42B6                ;      0D ENDS TEXT WITH CR/LF
42B6                ;
42B6  7E        OTTXT  MOV   A,M     ; LOAD CHARACTER
42B7  FE00             CPI   00H     ; SEARCH FOR END OF LINE
42B9  C8               RZ    ;         RETURN IF END
42BA  FE0D             CPI   0DH     ; SEARCH FOR <CR>
42BC  CAC942           JZ    CRLF    ; WRITE CR-LF IF <CR> FOUND
42BF  47               MOV   B,A     ; ELSE LOAD CHARACTER
42C0  3E00             MVI   A,0     ; TERMINAL = DEVICE #0
42C2  CD2142           CALL  DCOUT   ; WRITE CHARACTER
42C5  23               INX   H       ; POINT TO NEXT CHARACTER
42C6  C3B642           JMP   OTTXT   ; LOOP OVER TEXT
42C9                ;
42C9                ;  OUTPUT CARRIAGE RETURN - LINE FEED
42C9                ;
42C9  060D      CRLF   MVI   B,0DH   ; <CR>
42CB  3E00             MVI   A,00H   ; TERMINAL = 0
42CD  CD2142           CALL  DCOUT   ; WRITE CR
42D0  060A             MVI   B,0AH   ; <LF>
42D2  3E00             MVI   A,00H
42D4  C32142           JMP   DCOUT   ; WRITE LF
42D7                ;
42D7  20204D49  TEXT   DB    '  MISSING PROG ',0
      5353494E
      47205052
      4F472000
42E7                ;
42E7  0000      ADRAM  DW    00      ; RAM ADDRESS FOR PROGRAM
42E9  0000      BASSP  DW    00      ; TEMP LOC FOR BASIC STACK PTR
42EB  0000      ERFLG  DW    00      ; ERROR FLAG
42ED                ;
42ED  0D        BUFFR  DB    0DH     ; PROG NAME BUFFER AREA
42EE  1100             DS    274     ; SPACE FOR 21 NAMES + STACK
4400            STACK  END   ;         STACK BEGINS HERE
```

— END OF OVERLAY LISTING —

Finally! A 68000 CROMIX™ SPREADSHEET

```
                      MAY      JUN
ROYALTIES       0    0    140    328    422    369
COMMISSIONS     0    0    675   1577   2029   1778
              ----  ----  ----  -----  -----  -----
COST OF SALES:  0    0   2090   4883   6284   5505

GROSS PROFIT    0    0   5372  12624  15973  13926
GROSS PROFIT %  0    0     72     72     71     70
```

NEW SOFTWARE!

68Kalc™ is the 68000 Cromix spreadsheet you've needed for so long.

68Kalc runs directly on the 68000 in your Cromemco computer, taking full advantage of its size and speed. 68Kalc runs on any terminal supported by Cromix termcaps, so you can use it with most popular terminals.

68Kalc is easy to learn. Its commands are a subset of Lotus 1-2-3's. Its command keystrokes mimic 1-2-3's. And help is instantly available inside the spreadsheet.

Order your first copy of 68Kalc now. Try it for thirty days. If you can imagine getting your job done without it after that, return it for a full refund.

ORDER NOW! Your first copy of 68Kalc is just $495.00 (U.S.). To order fill out the response card or call toll-free **1-800-227-3094** and have your VISA, MasterCard or company P.O. ready.

Inquiries and orders can also be addressed to: Warecraft, 501 North 36th Street, #138, Seattle, Washington 98103 USA, (206) 527-5403

Phone Orders 1-800-227-3094

Warecraft

Dealer Inquiries Invited.

BUILD AN EEPROM BOARD

Brian Smithgall

EEPROM stands for Electrically Erasable Programmable Read Only Memory. That's quite a mouthful to say, but the concept is simple. It's an easy-to-use memory device that retains its contents even without power applied. EEPROMs have many uses in a microprocessor system. They, like EPROMs, offer an ideal way to record data semipermanently, with complete recall capability, but with the added advantage that the stored data can be easily altered by the computer.

Whether you are an adventurous hobbyist or a sophisticated system designer, you most likely have one or more uses for EEPROMs. You may, for example, have a special system application that needs to keep track of operation parameters, phone numbers, passwords, etc. Or, if you know how to change the operating system, the EEPROM could store special setup conditions (in lieu of switches) or keep track of the time and date when the system was last accessed. If you are using a PROM-based system without disk drives, a board such as the one described in this article can add a great deal of flexibility by providing field-alterable software. I'm using mine to store parameters about the system that are used by the run-time programs.

The circuits presented in this article can be used to either build a stand-alone EEPROM board or to add the EEPROM function to the clock-board project published in S-100 Journal No. 3.

EEPROMs offer several advantages over EPROMs: you don't need to buy an ultraviolet lamp, deal with special voltages, or plug and unplug the chip to program it. EEPROMs are more robust than battery-backup RAMs (BBRAMs). The latter are often sensitive to power-on conditions and other voltage spikes. The chip used in this project (made by XICOR) operates with the simplicity of standard static RAM. A minimum of external circuitry is required.

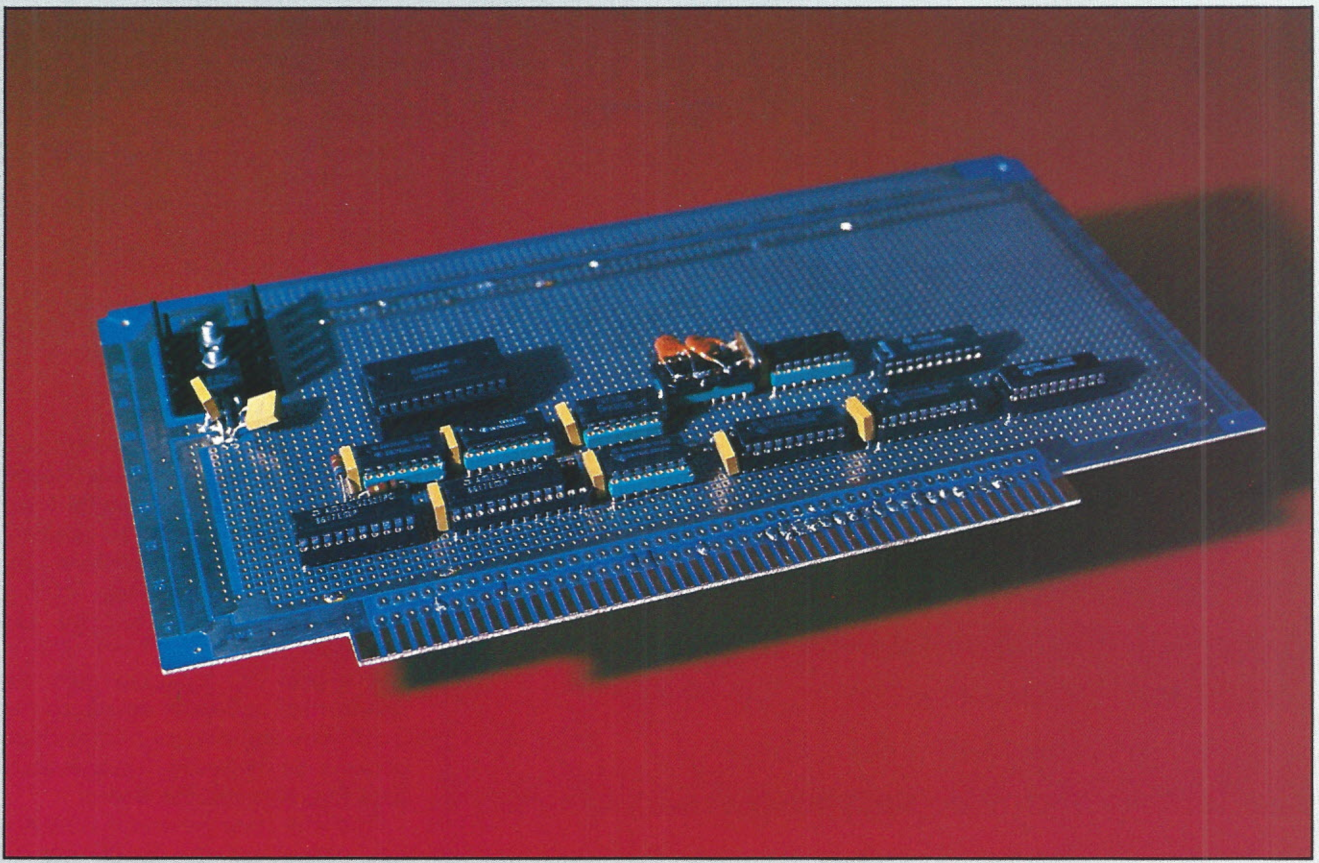
The circuit described will provide 512 bytes of storage. For the more ambitious, a larger 2K EEPROM is also made by XICOR, and it can be used instead with very few modifications to the board described. And, you may of course wish to build a larger capacity board using several of the chips. I'll leave these changes for the interested reader to make.

FROM ROMS TO EEPROMS

The name EEPROM has evolved out of the alphabet soup of electronics through many generations of devices. The mnemonic ROM stands for Read Only Memory, a technology that was available back in the dark ages of microcomputing. They were cheap, easy ways to store programs or data permanently and were usually 'written' by a permanent mask layout during chip design. Often, they could only be afforded by large companies with needs for large quantities.

Along came a way to make them *programmable* by the user, and we gained the PROM. This family of chips could be permanently written to on an individual-byte basis — usually by burning 'fuses' within the ►

Brian Smithgall is a systems designer and is Vice President of Engineering at Systronics, Inc. of Atlanta, Georgia. Brian is also a hiking, camping, and general outdoors enthusiast.



part. The only drawback this time was when the programmer changed his mind or made a mistake: the chip went into the trash and a new one was burned. System designers were often faced with a large development cost on these 'spent' chips.

Then someone clever made a way for these memories to be *erasable*, and we had the EPROM. These chips, usually erased by an ultraviolet light, had a unique look with little windows on the chip top. The window allowed the light from a nearby ultraviolet lamp to shine on the device, and the light waves erased the memory by altering the chip's electrical properties. The only disadvantage now was that the erasing process took time (usually half an hour or so).

Finally, a way was invented for designers to erase the chip electronically, often a byte at a time, and the EEPROM was born.

Important in this chip is the ability to read and write to it almost like to normal RAM, and the data is retained even without power. The main difference in the read/write aspect of the chip is that it is slow compared to normal memory, with a write cycle taking 10 milliseconds per byte.

PREREQUISITES AND A FEW NOTES FOR THIS PROJECT

The circuit for this project utilizes port space for direct mapping. You will need a machine with 16-bit port addressing. The particular circuit presented was chosen because it was an easy modification of the real-time clock circuit published in Vol. 1, No. 3 of *S-100 Journal*. The present article does not require that you have built that project or have that article, but it discusses in the end how to modify that board to add the EEPROM circuit.

This article assumes that you are familiar with hardware to a certain extent. You should be familiar with the basics of electronics, signal flows, how to read a schematic, and the basic logical functions. Additionally, you need access to equipment to solder or wire wrap the board and a logic probe or oscilloscope to trace down problems. *(Editor's note: The clock board article in S-100 Journal No. 3 covers several basics about interfacing to the S-100 bus. The 696 Bus column in general, and in issue No. 5 in particular, is also a good source of basic information for those building circuits for the S-100 bus.)*

In this article, and in general, signals which are described as active low will be followed by an asterisk (for example BDSEL*). If the signal is run through an inverter, it becomes active high (BDSEL). Some data sheets will also represent active low signals by means of a bar over the label.

Each chip has a power and ground connection. A decoupling capacitor is wired across the power and ground lines, near the chip. This capacitor removes high-frequency transients and provides a small burst of power nearby when the chip changes state. An easy way to attach the chip is to use a socket two holes longer than the chip, insert the capacitor, and wire wrap the connectors on the other side.

HOW THE BOARD WORKS

A block diagram of the EEPROM board (with the optional clock) circuit is shown in Figure 1. The 5-sided symbol indicates S-100 bus connections and the direction of signal flow. Busses are multiline signal paths and

Qty.	ID	Part Description
3		14-pin wire wrap sockets
1		16-pin wire wrap socket
6		20-pin wire wrap sockets
1		24-pin wire wrap socket
1	U2	74LS38 (TI) 4 dual-input NAND gates (open collector)
1	U3	74LS04 (TI) 6 inverters
1	U5	74LS165 (TI) 8-bit parallel-in serial-out shift resister
4	U6,7,13,14	74LS244 (TI) 8-line tristate line drivers
2	U9, U10	74LS521 or 74LS688 (TI) octal comparators
1	U11	74LS32 (TI) 4 dual-input OR gates
1	U12	X2804AP-45 (XICOR) 512-byte EEPROM
1		LM340-5 (TI) 5-Volt regulator
1		S-100 (Vector Elect.) prototype board.
1		Heat sink for regulator
1		1000Ω resistor - pullup for open collector to LS chip
1		4700Ω resistor - pullup for open collector to bus
2	C_1, C_2	1μF 25V capacitors for power supply
11	C_D	0.1μF decoupling capacitors (1 for each chip)

Table 1. *List of parts required to build the EEPROM board.*

Qty.	ID	Part Description
4		20-pin wire wrap sockets
1		24-pin wire wrap socket
2	U13, U14	74LS244 (TI) 8-line tristate line drivers
2	U9, U10	74LS521 or 74LS688 (TI) octal comparators
1	U11	74LS32 (TI) 4 dual-input OR gates
1	U12	X2804AP-45 (XICOR) 512-byte EEPROM
3	C_D	0.1μF decoupling capacitors

Table 2. *List of parts required to add the EEPROM function to the clock board.*

are indicated by a slash across the line with the number of wires written near it. The board sections briefly described below are discussed later in more detail. Figures 2 to 8 show details of each section.

The board is functionally divided into six main sections:

1. Address buffer and decoder.
2. Data buffers.
3. Port I/O logic.
4. Wait state generator.
5. EEPROM chip.
6. Power supply.

The address decoder listens to the address generated by the bus master (CPU) and develops signals for other sections when the board is accessed.

The data buffers combine the S-100 Data-In bus and Data-Out bus into a bidirectional bus for the EEPROM chip.

The port I/O logic interprets the special read and write signals on the S-100 bus and properly times these activities on the board. It also enables data flow through the board.

The wait state generator creates a hold signal for the S-100 bus to accomodate a slower chip on a fast bus. This extends the read and write signals to allow the EEPROM to access properly.

At the heart of the board is the EEPROM device itself, which provides its own specialized timing of the internal write procedures.

Finally the power supply delivers a regulated 5 volts to all the chips in order to produce the digital signals.

Some of the signals generated (and shown in the circuit diagrams) apply to the real-time clock project mentioned above. If you have not built that board and do not intend to include the clock chip in this one, ignore the lines that go to that chip.

THE ADDRESS DECODER

Figure 2 shows the address-decoder portion of the circuit. The 16-port address lines are buffered immediately near the bus using chips U13 and U14. This is done for two reasons. Several of the address lines go to both peripheral chips (EEPROM and clock) as well as the address decoders (U9 and U10). The S-100 standard restricts the current draw from the bus, this usually meaning that only one

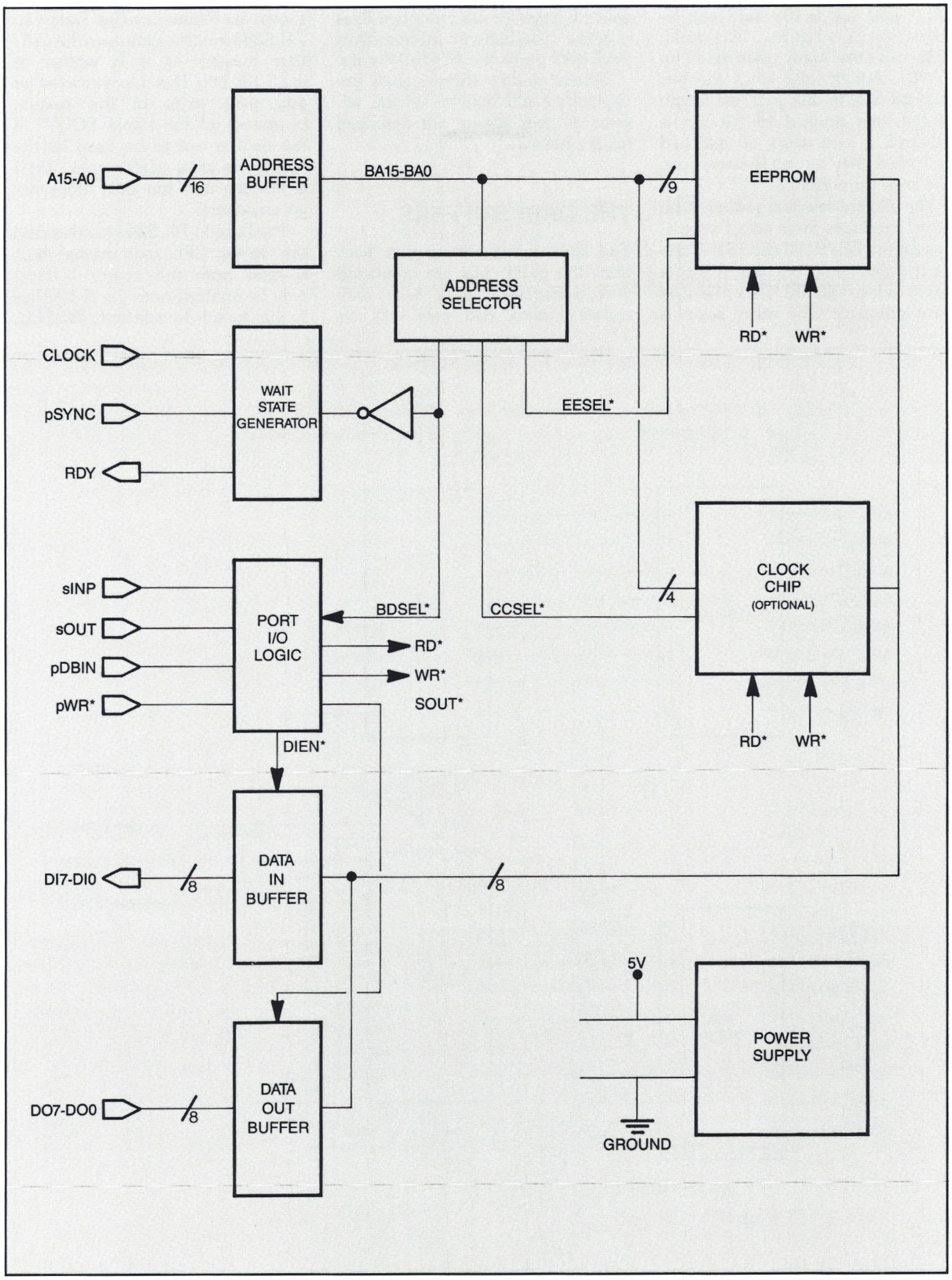


Figure 1. *Block diagram of the EEPROM/clock board.*

74LS part per S-100 card can be connected to a bus line. The buffer chips can drive many parts (often up to 30). Additionally, since the bus lines go only to one chip, the length of the wire hooked to the bus is reduced. If long wires are attached to the bus, they can act like antennae and pick up noise.

The address-selection section of the circuit produces three select signals: one for the EEPROM (EESEL*), one for the clock chip (CCSEL*), and a board select (BDSEL*) for the wait state generator. The board select is active if either of the other two lines is active. The buffered address lines BA8-BA0 go to the EEPROM chip.

Details of how these signals are developed and how to set the address in your system are described further below.

THE DATA BUFFERS

The data input and output lines from the S-100 bus are combined into bidirectional data lines with tristate buffers (U6 and U7), as shown in Figure 3. The Data-Out 74LS244 handles data from the CPU (bus master) as it is written to the S-100 bus. This chip is enabled for any port write in the system by means of the signal SOUT*. If the data is not to be used by the board, the write-select signal (WR*) is inactive and the data 'does not go anywhere.'

The Data-In 74LS244 handles data sent to the CPU (bus master) as it is read from the board. It must only be enabled onto the S-100 bus if the board is selected (BDSEL*

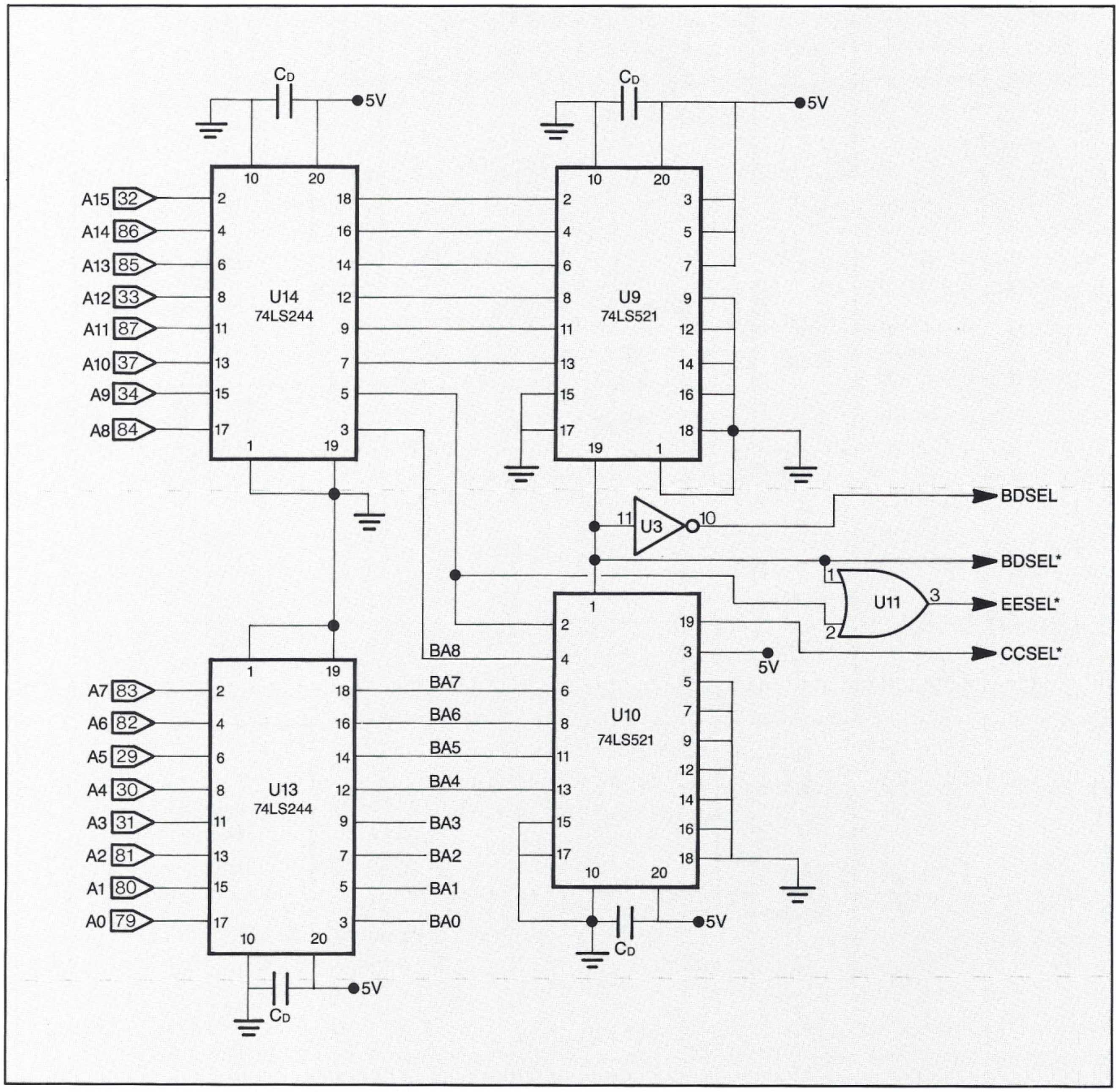


Figure 2. *The circuit for the address decoder. In this diagram, the base address is assumed to start at E000.*

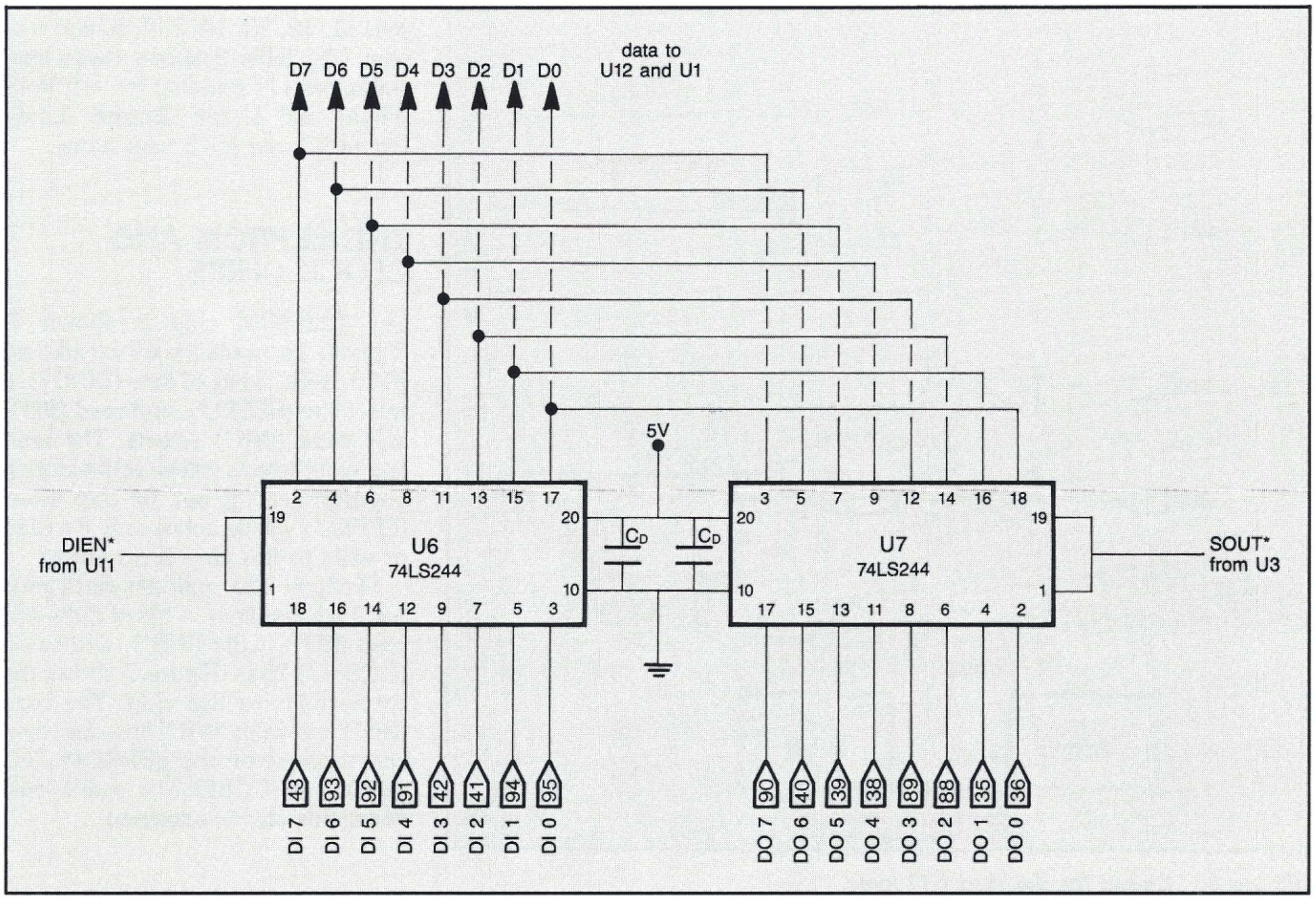


Figure 3. *The data buffers. The S-100 data input and data output lines are combined into bidirectional data lines.*

active) and a read (RD* active) occurs. This is controlled by the signal DIEN*. Note that this board is selected if either the clock chip or the EEPROM is selected.

THE PORT I/O LOGIC

The port I/O logic sections, shown in Figure 4, generate the port-read (RD*) and port-write (WR*) signals as well as control signals for the bidirectional data bus. During a port write, the S-100 bus signals pWR* and sOUT must both be low. These signals will be accompanied by data passing on the S-100 bus from the master CPU to a peripheral board or device. Note that the WR* signal will go active during any port write even if it is not intended for this board. This does not pose a problem since the devices on our board are only enabled if they are addressed (via the SELect lines).

The read strobe (RD*) will go active only when both pDBIN and sINP are high. This also occurs during any port-read transfer to the bus master.

Data enabling from the on-board data bus onto the S-100 DI bus must be carefully controlled to prevent this data from interfering with data flow from other sources. The output buffer, U6, is enabled by DIEN* which is low only when both RD* and the board select (BDSEL*) are low.

The on-board data bus may have data on it from U7 at any time unless there is a bus read in progress. The signal SOUT*, produced by inverting the S-100 bus line sOUT, is sufficient to properly supply data during writes while not interfering with reads.

THE WAIT STATE GENERATOR

The wait state circuit is shown in Figure 5. The wait state generator creates a hold-off signal (RDY) to extend the bus cycle and to permit slower devices to meet their timing requirements. The number of wait states required for your system is shown in Table 3. The number of wait states indicated by Table 3 is selected in your circuit with the shift sequence determined by the load values on

PROCESSOR SPEED (MHz)	NUMBER OF WAIT STATES	PINS ON CHIP U5							
		11	12	13	14	3	4	5	6
2	1	H	H	H	H	H	H	H	L
4	2	H	H	H	H	H	H	L	L
6	3	H	H	H	H	H	L	L	L
8	4	H	H	H	H	L	L	L	L

Table 3. *How to select the wait states required for various system speeds.*

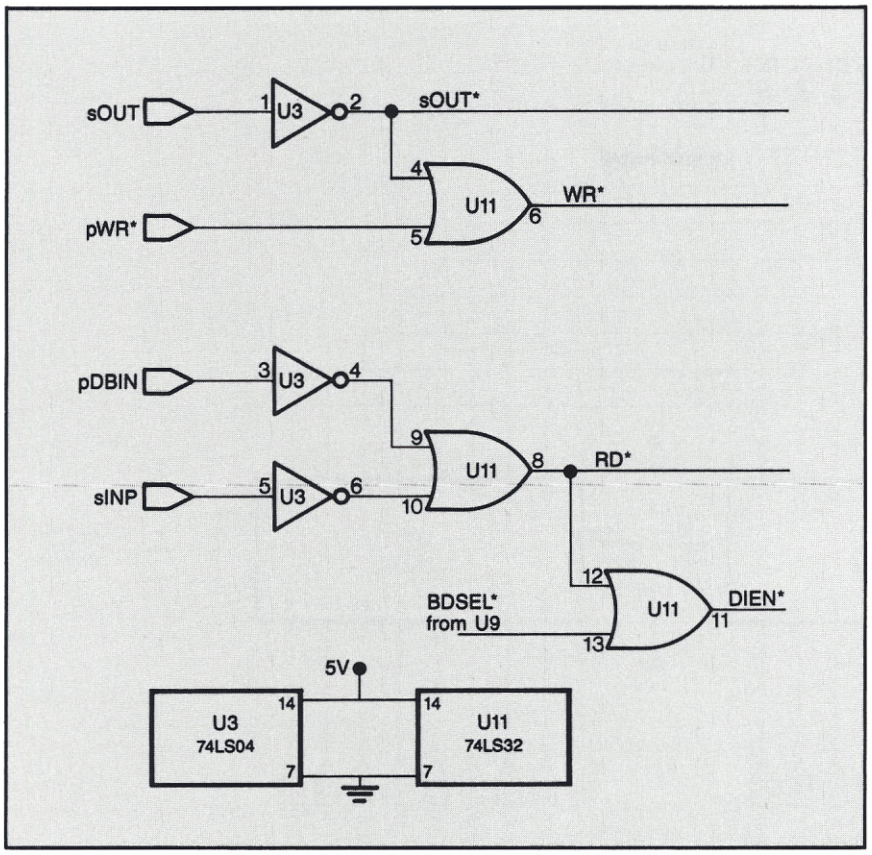


Figure 4. *Circuit for the port I/O logic.*

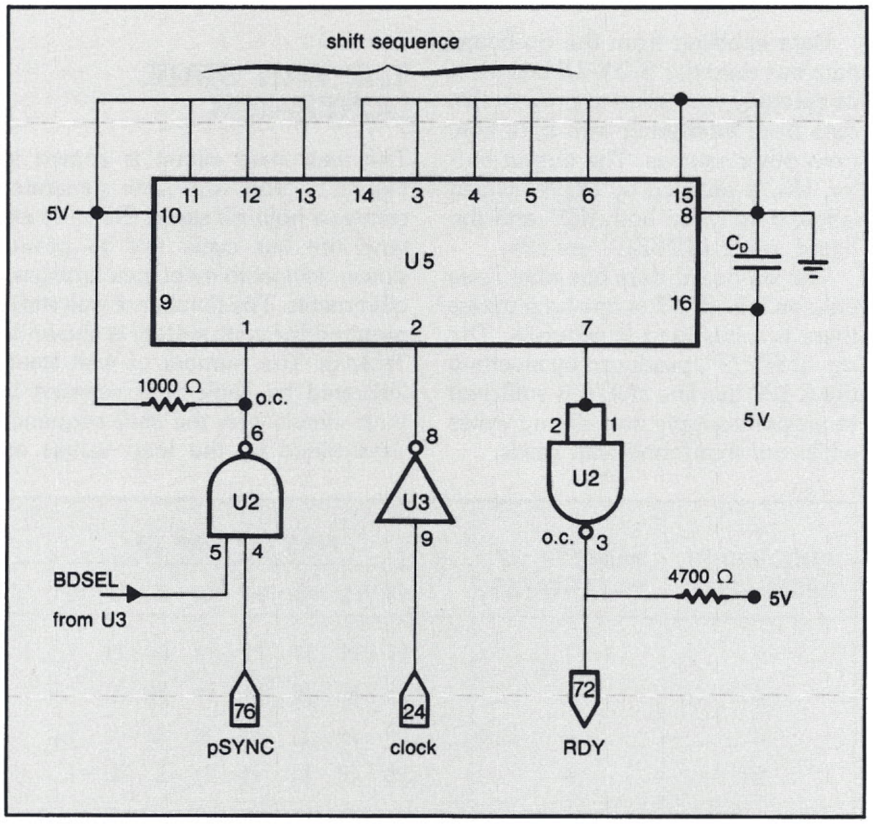


Figure 5. *Circuit for the wait-state generator.*

pins 11, 12, 13, 14, 3, 4, 5, and 6 of chip U5. Table 3 shows these load states, with H standing for +5 Volts (High) and L for Ground (Low). Figure 5 is set for 5 wait states.

THE EEPROM AND CLOCK CHIPS

The EEPROM chip is shown in Figure 6. Its inputs are a 9-bit address (BA0-BA8), 8 bits of data (D0-D7), a select line (EESEL*), and read (RD*) and write (WR*) selects. The read and write selects will be active during any port activity, but the chip select (EESEL*) will be active only if a read or write to this chip is occurring.

Similarly, the optional clock chip has 4 address lines, 4 bits of data, and read (RD*), write (WR*), and select (CCSEL*) lines. Figure 7 shows the connections to this chip. The read (RD*) and write (WR*) lines are identical to those on the EEPROM. The select line (CCSEL*) is active only when this chip is accessed.

THE POWER SUPPLY

Figure 8 shows the power supply circuit for the board. The regulator provides a steady 5 volts to the digital circuit by reducing the S-100 supply voltage from approximately 8 volts. The capacitors C1 and C2 provide some additional filtering and noise immunity.

SETTING THE ADDRESS ON YOUR BOARD

You must locate 1024 unused ports in your system port space. Specifically, this project uses the first 512 ports for the EEPROM and 16 ports in the adjacent 512-port block for the clock chip if so equipped.

The board is accessed through a 16-bit port address (A0-A15). The upper 6 bits (A10-A15) select the base address of the *board*. The base address is a multiple of 1024 (400 hexadecimal) and indicates where in the 64K I/O-port address space our block of 1024 ports should start.

Bit A9 decides whether we are accessing the clock or the EEPROM. A9=0 selects the first 512-port block (the EEPROM). A9=1 selects the second 512-port block (the clock).

If A9=0, then A0-A8 select each 9-bit EEPROM address (0-511).

If A9=1, then A4-A8 select the base address of the *clock* within the second 512-port block. And A0-A3 select each of the 16 ports required for the clock.

This is summarized at the top of Figure 9.

Note that the board consumes 1024 ports starting at the base address. The base address is fixed by connecting wires to the address selection comparators (U9 and U10), or may be made variable using switches. The upper 6 bits of the base address correspond to pins 3, 5, 7, 9, 12, and 14 of chip U9. To set the base address for your system, locate a block of 1024 unused ports, where the base address is divisible by 0400 hex. Assign the upper 6 bits of that address to the pins listed above, where a 1 in the base address bit means that you should connect the corresponding pin to +5 Volts, and a 0 for the bit means that you should connect the pin to Ground. In the circuit shown in Figure 2, the base address is set to E000 hex. This becomes the base address for the EEPROM.

The 16 ports for the clock chip can be positioned anywhere in the upper half of the 1024 ports. This window is similarly selected by corresponding the clock base-address bits to pins 5, 7, 9, 12, and 14 of chip U10. Figure 2 shows the base address set to E200. Figure 9 shows the base address set to E270. It is important that pin 3 of U10 be +5 Volts, and therefore that bit A9 of the clock address is 1, to avoid overlap with the EEPROM. If the clock chip is not used, chip U10 is not needed, and you can disregard the part of the above discussion that concerns the clock base address.

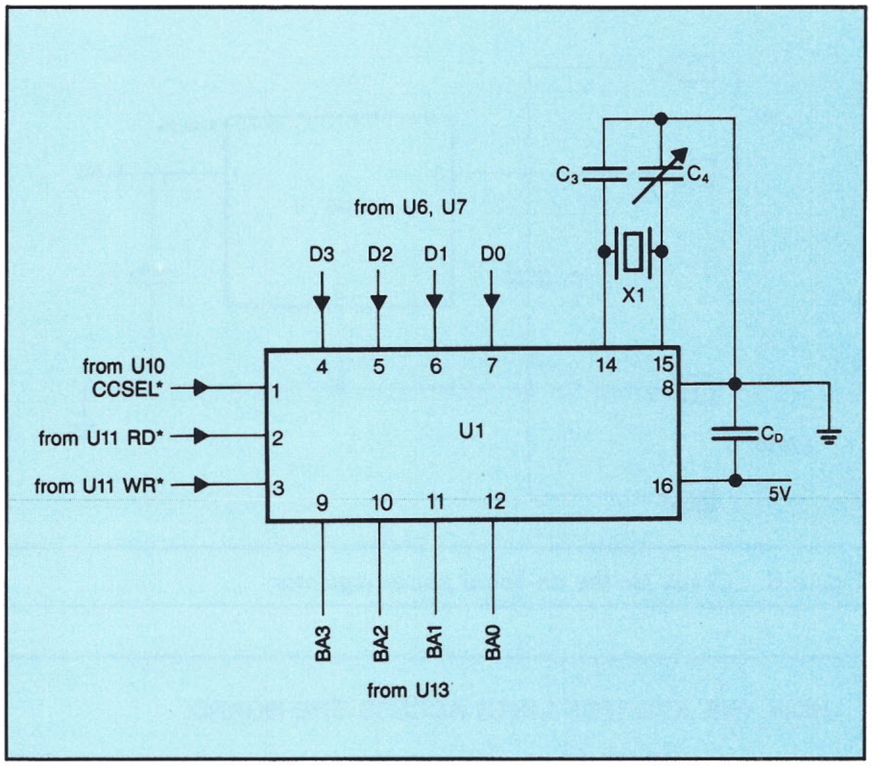


Figure 7. *The clock circuit, shown here for completeness. These parts and circuit were previously discussed in S-100 Journal Number 3.*

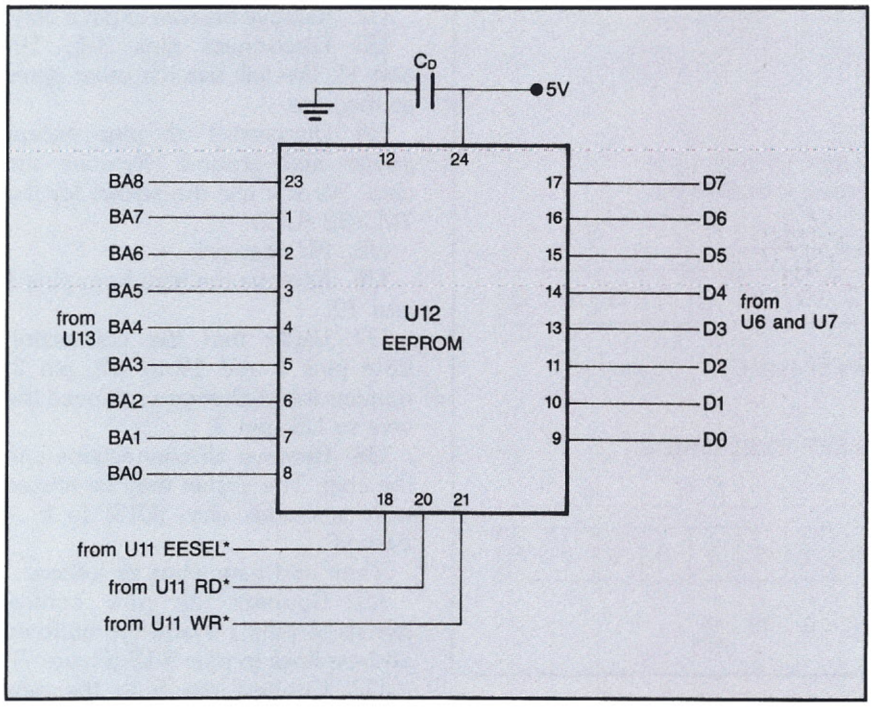


Figure 6. *Circuit showing the connections to the EEPROM chip.*

BUILDING THE CIRCUITS

Table 1, on page 48, lists the parts necessary for the EEPROM project. Note that some of the U numbers are missing. They correspond to parts that were used in the original real-time clock project.

All these parts should be easy to obtain, with the possible exception of the XICOR EEPROM. If you cannot find this chip from your standard electronics supplier, write to Jay Vilhena at *S-100 Journal*. We will try to supply the part at cost plus shipping — the chip costs less than $10.

If you are starting from scratch on this project, plan the chip placement so it reduces wire lengths. Specifically, place the address and data buffers near the bus to reduce wire lengths.

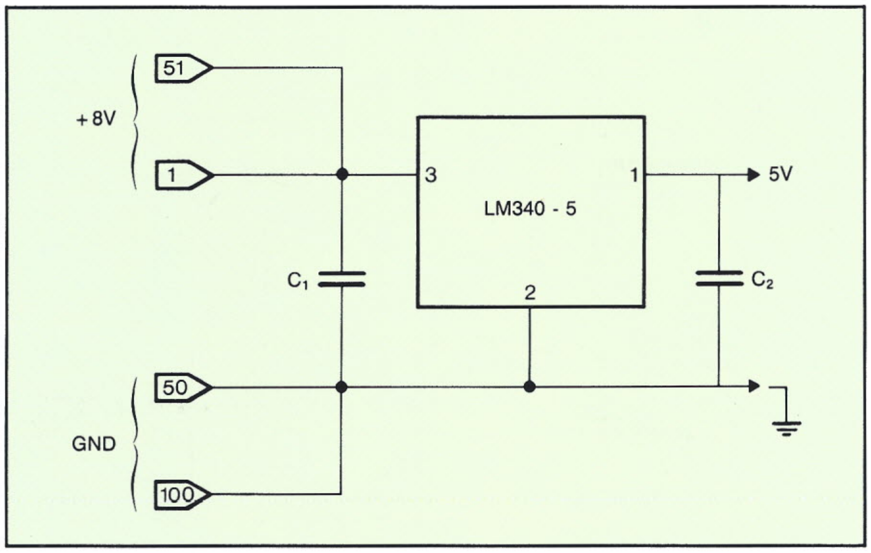


Figure 8. *Circuit for the on-board power regulator.*

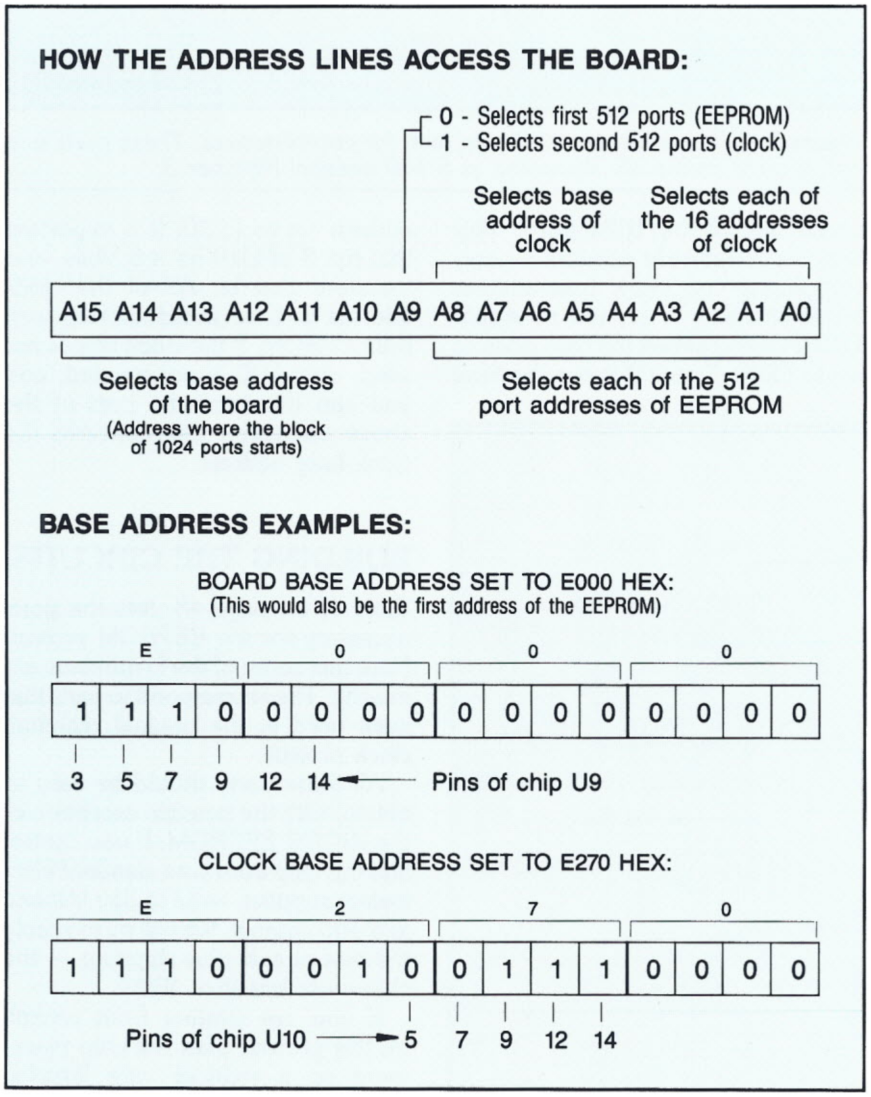

HOW THE ADDRESS LINES ACCESS THE BOARD:
0 - Selects first 512 ports (EEPROM)
1 - Selects second 512 ports (clock)
Selects base address of clock
Selects each of the 16 addresses of clock
A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
Selects base address of the board
(Address where the block of 1024 ports starts)
Selects each of the 512 port addresses of EEPROM
BASE ADDRESS EXAMPLES:
BOARD BASE ADDRESS SET TO E000 HEX:
(This would also be the first address of the EEPROM)
E 0 0 0
1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
3 5 7 9 12 14
Pins of chip U9
CLOCK BASE ADDRESS SET TO E270 HEX:
E 2 7 0
1 1 1 0 0 0 1 0 0 1 1 1 0 0 0 0
Pins of chip U10
5 7 9 12 14

Figure 9. *The board is accessed through 16-bit I/O ports. 1024 ports are used and they can start at any 1024-port boundary within the 65,536-port space.*

Test the circuit without the EEPROM in until you know that everything else is working properly. To test the EEPROM, see the section on software drivers below.

HOW TO MODIFY THE REAL-TIME CLOCK

The real-time clock is described in *S-100 Journal*, Vol. 1 No. 3. It uses the National clock chip MM58174. Due to the similar nature of these projects, a portion of the wiring may be kept the same in upgrading the clock board to include the EEPROM chip.

The wait state generator remains identical. The data bus buffers are also identical, although now we will use all 8 bits of each chip. The port I/O logic has changed to conserve parts, but some portions remain the same.

The circuit described above will support both the clock chip and the EEPROM chip. New control signals have been generated for the clock. The power supply remains unchanged.

To modify the clock circuit in order to add the EEPROM, remove the following wires from the chips indicated:

U1: Remove the wires leading to pins 1-3 and 9-12.

U2. Remove the wire to pin 5 only.

U3: Disconnect pins 3-6, 10, and 11. We will use the other gates as they are.

U4. Disconnect all pins except power and ground. Remove the chip. We will use the socket for the 74LS32 (U11).

U5. No changes.

U6. Remove the wire from pins 1 and 19.

U7. Verify that the connection from pins 1 and 19 to U3, pin 2, remains intact after you removed the wire to U3, pin 3.

U8. Remove all connections and the chip. The socket may be reused with a shorter chip (U13) in it, if desired.

Then add new wires as follows:

U1: Connect the new control signals to pins 1-3 and the buffered address lines to pins 9-12 (Figure 7).

U2: Connect pin 5 to the new select signal, BDSEL*, as described in this article. ►

Z sets you free!

Who we are

Echelon is a unique company, oriented exclusively toward your CP/M-compatible computer. Echelon offers top quality software at extremely low prices; customers are overwhelmed at the amount of software they recieve when buying our products. For example, the Z-Com product comes with approximately 92 utility programs; and our TERM III communications package runs to a full megabyte of files. This is real value for your software dollar.

ZCPR 3.3

Echelon is famous for our operating systems products. ZCPR3, our CP/M enhancement, was written by a software professional who wanted to add features normally found in minicomputer and mainframe operating systems to his home computer. He succeeded wonderfully, and ZCPR3 has become the environment of choice for "power" CP/M-compatible users. Add the fine-tuning and enhancements of the now-available ZCPR 3.3 to the original ZCPR 3.0, and the result is truly flexible modern software technology, surpassing any disk operating system on the market today. Get our catalog for more information - there's four pages of discussion regarding ZCPR3, explaining the benefits available to you by using it.

Z-System

Z-System is Echelon's complete disk operating system, which includes ZCPR3 and ZRDOS. It is a complete 100% compatible replacement for CP/M 2.2. ZRDOS adds even more utility programs, and has the nice feature of no need to warm boot (^C) after changing a disk. Hard disk users can take advantage of ZRDOS "archive" status file handling to make incremental backup fast and easy. Because ZRDOS is written to take full advantage of the Z80, it executes faster than ordinary CP/M and can improve your system's performance by up to 10%.

Installing ZCPR3/Z-System

Echelon offers ZCPR3/Z-System in many different forms. For $49 you get the complete source code to ZCPR3 and the installation files. However, this takes some experience with assembly language programming to get running, as you must perform the installation yourself.

For users who are not qualified in assembly language programming, Echelon offers our "auto-install" products. Z-Com is our 100% complete Z-System which even a monkey can install, because it installs itself. We offer a money-back guarantee if it doesn't install properly on your system. Z-Com includes many interesting utility programs, like UNERASE, MENU, VFILER, and much more.

Echelon also offers "bootable" disks for some CP/M computers, which require absolutely no installation, and are capable of reconfiguration to change ZCPR3's memory requirements. Bootable disks are available for Kaypro Z80 and Morrow MD3 computers.

Z80 Turbo Modula-2

We are proud to offer the finest high-level language programming environment available for CP/M-compatible machines. Our Turbo Modula-2 package was created by a famous language developer, and allows you to create your own programs using the latest technology in computer languages - Modula-2. This package includes full-screen editor, compiler, linker, menu shell, library manager, installation program, module library, the 552 page user's guide, and more. Everything needed to produce useful programs is included.

"Turbo Modula-2 is fast...[Sieve benchmark] runs almost three times as fast as the same program compiled by Turbo Pascal...Turbo Modula-2 is well documented...Turbo's librarian is excellent". - Micro Cornucopia #35

BGii (Backgrounder 2)

BGii adds a new dimension to your Z-System or CP/M 2.2 computer system by creating a "non-concurrent multitasking extension" to your operating system. This means that you can actually have two programs active in your machine, one or both "suspended", and one currently executing. You may then swap back and forth between tasks as you see fit. For example, you can suspend your telecommunications session with a remote computer to compose a message with your full-screen editor. Or suspend your spreadsheet to look up information in your database. This is very handy in an office environment, where constant interruption of your work is to be expected. It's a significant enhancement to Z-System and an enormous enhancement to CP/M.

BGii adds much more than this swap capability. There's a background print spooler, keyboard "macro key" generator, built-in calculator, screen dump, the capability of cutting and pasting text between programs, and a host of other features.

For best results, we recommend BGii be used only on systems with hard disk or RAMdisk.

JetFind

A string search utility is indispensible for people who have built up a large collection of documents. Think of how difficult it could be to find the document to "Mr. Smith" in your collection of 500 files. Unless you have a string search utility, the only option is to examine them manually, one by one.

JetFind is a powerful string search utility which works under any CP/M-compatible operating system. It can search for strings in text files of all sorts - straight ASCII, WordStar, library (.LBR) file members, "squeezed" files, and "crunched" files. JetFind is very smart and very fast, faster than any other string searcher on the market or in the public domain (we know, we tested them).

Software Update Service

We were suprised when sales of our Software Update Service (SUS) subscriptions far exceeded expectations. SUS is intended for our customers who don't have easy access to our Z-Node network of remote access systems. At least nine times per year, we mail a disk of software collected from Z-Node Central to you. This covers non-proprietary programs and files discussed in our Z-NEWS newsletter. You can subscribe for one year, six months, or purchase individual SUS disks.

There's More

We couldn't fit all Echelon has to offer on a single page (you can see how small this typeface is already!). We haven't begun to talk about the many additional software packages and publications we offer. Send in the coupon below and just check the "Requesting Catalog" box for more information.

Item	Name	Price	
1	ZCPR3 Core Installation Package	$49.00	(3 disks)
2	ZCPR3 Utilities Package	$89.00	(10 disks)
5	Z-Com (Auto-Install Complete Z-System)	$119.00	(5 disks) *
6	Z-Com "Bare Minimum"	$69.95	(1 disks)
10	BGii Backgrounder 2	$75.00	(2 disks)
12	PUBLIC ZRDOS Plus (by itself)	$59.50	(1 disk)
13	Kaypro Z-System Bootable Disk	$69.95	(3 disks)
14	Morrow MD3 Z-System Bootable Disk	$69.95	(2 disks)
16	QUICK-TASK Realtime Executive	$249.00	(3 disks)
17	DateStamper file time/date stamping	$49.95	(1 disk)
18	Software Update Service	$85.00	(1 yr sub)
20	ZAS/ZLINK Macro Assembler and Linker	$69.00	(1 disk)
21	ZDM Debugger for 8080/Z80/ HD64180 CPU's	$50.00	(1 disk)
22	Translators for Assembler Source code	$51.00	(1 disk)
23	REVAS3/4 Disassembler	$90.00	(1 disk)
24	Special Items 20 through 23	$169.00	(4 disks)
25	DSD-80 Full Screen Debugger	$129.95	(1 disk)
27	The Libraries.SYSLIB, Z3LIB, and VLIB	$99.00	(8 disks)
28	Graphics and Windows Libraries	$49.00	(1 disk)
29	Special Items 27, 28, and 82	$149.00	(9 disks)
30	Z80 Turbo Modula-2 Language System	$89.95	(1 disk)
40	Input/Output Recorder IOP (I/OR)	$39.95	(1 disk)
41	Background Printer IOP (BPrinter)	$39.95	(1 disk)
44	NuKey Key Redefiner IOP	$39.95	(1 disk)
45	Special Items 40 through 44	$89.95	(3 disks)
60	DISCAT Disk cataloging system	$39.99	(1 disk)
61	TERM3 Communications System	$99.00	(6 disks)
64	Z-Msg Message Handling System	$99.00	(1 disk)
66	JetFind String Search Utility	$49.95	(1 disk)
81	ZCPR3: The Manual bound, 350 pages	$19.95	
82	ZCPR3: The Libraries 310 pages	$29.95	
83	Z-NEWS Newsletter, 1 yr subscription	$24.00	
84	ZCPR3 and IOPs 50 pages	$9.95	
85	ZRDOS Programmer's Manual 35 pages	$8.95	
88	Z-System User's Guide 80 page tutorial	$14.95	

* Includes ZCPR3: The Manual

Echelon, Inc.

885 N. San Antonio Road, Los Altos, CA 94022 USA
415/948-3820 (order line and tech support)
Telex 4931646

NAME ______________________

ADDRESS ______________________

____

TELEPHONE ______________ DISK FORMAT ________

☐ REQUESTING CATALOG

ORDER FORM

Payment to be made by:

☐ Cash
☐ Check
☐ Money Order
☐ UPS COD
☐ Mastercard/Visa:

# ______________________

Exp. Date ______________

California residents add 7% sales tax. Add $4.00 shipping/handling in North America, actual cost elsewhere.

ITEM	PRICE
Subtotal	
Sales Tax	
Shipping/Handling	
Total	

U3: Connect pins 3-6, 10, and 11 as indicated in this article.

U6: Connect pins 1 and 19 to the new enable signal, DIEN*. Also connect pins 2, 4, 6, and 8 as shown in Figure 3.

U7: Connect pins 3, 5, 7, and 9 as shown in Figure 3.

U9, U10, U13, and U14: Add the new address decode section as shown in Figure 2.

U11: Insert the chip in the old U4 socket and connect as described in this article.

U12: All new connections to the EEPROM.

You should then go through all signals in the new schematic to verify that everything is wired correctly before plugging anything in. Check the base address used by your software, as it may be different from before.

SOFTWARE DRIVERS

The listing on this page is a typical program to use the EEPROM. Take care not to write repeatedly to the same address — these chips have a maximum life of about 1/2 million write cycles. That could occur very quickly in a microprocessor system.

The 10-millisecond write-cycle delay is worst case. It may be sped up by monitoring the contents of a memory location. When the write cycle is active, outputs are disabled. If a known data location is read continuously, it will read correctly only after the write cycle is complete, and the outputs are enabled.

Experiment with your system after you make the board.

These routines write only bytes (characters) to the EEPROM. If you wish to write real numbers or strings, for example, you must write them byte by byte. The data constructs vary from language to language but they are usually accessed through address pointers or equivalent superimposed data structures. You'll have to dig into your own system to discover how it represents and accesses data.

Good luck!

EXAMPLE OF A SOFTWARE DRIVER FOR THE EEPROM BOARD

Program written by Brian Smithgall

```
(******************************************************************************)
(*    THIS PASCAL PROGRAM DEMONSTRATES THE USE OF THE XICOR 2804 EEPROM      *)
(*    THE BASE ADDRESS OF THE BOARD IS ASSUMED TO BE E000 hex                 *)
(*    EEPROM IS ADDRESSED FROM E000 hex TO E1FF hex                           *)
(*    THE TEST SHOULD BE RUN BY:                                              *)
(*        1) doing WRITE_TEST, wait until done                                *)
(*        2) doing READ_TEST,  verify no errors                               *)
(*        3) turn power off, wait a few seconds                               *)
(*        4) turn power on, boot up and rerun program                         *)
(*        5) do READ_TEST again, verify memory was held during power out      *)
(******************************************************************************)

PROGRAM DEMO_EEPROM;

procedure HOLD_UNTIL_DONE;      (* worst case delay for write is 10 ms *)
BEGIN
  delay(10);                    (* built-in procedure to delay 10 ms *)
END;

procedure WRITE_EEPROM(ADDRESS,VALUE:integer); (* write the 8-bit VALUE *)
BEGIN                                          (* to ADDRESS 0-511 *)
  port [$E000+ADDRESS]:=VALUE;                 (* write to port *)
  HOLD_UNTIL_DONE;                             (* wait for completion *)
END;

function READ_EEPROM(ADDRESS:integer):integer; (* read location 0..511 *)
BEGIN                                          (* return byte as function *)
  READ_EEPROM:=port[$E000+ADDRESS];            (* read port *)
END;

procedure WRITE_TEST_EEPROM;                   (* write special pattern *)
var I:integer;                                 (* to EEPROM for test *)
BEGIN
  for I:=0 to 255 do                           (* write to address 0..255 *)
    WRITE_EEPROM(I,I);                         (* with 0..255 respectively *)
  for I:=0 to 255 do                           (* write addresses 256..511 *)
    WRITE_EEPROM(I+256,(I xor $FF));           (* with complement of above *)
  writeln('EEPROM WRITTEN TO');                (* tell when done *)
END;

procedure READ_TEST_EEPROM;                    (* test entire EEPROM for *)
var I:integer;                                 (* special pattern *)
BEGIN
  for I:=0 to 255 do                           (* check addresses 0..255 *)
    if READ_EEPROM (I)<>I then writeln('ERROR A ',I);
  for I:=0 to 255 do                           (* check addresses 256..511 *)
    if READ_EEPROM(I+256)<>(I xor $FF)  then writeln('ERROR B ',I);
writeln('TEST DONE');
END;

var C:char;                                    (* temporary variable *)
BEGIN                                          (* main program *)
  repeat
    write('ENTER 1 to READ_TEST or 2 to WRITE_TEST or Q to QUIT: ');
    readln(C);                                 (* read after prompt *)
    if C='1' then READ_TEST_EEPROM             (* do proper test *)
      else if C='2' then WRITE_TEST_EEPROM
        else if C<>'Q' then writeln('UNKNOWN COMMAND');
  until C='Q';                                 (* exit on Q *)
END.
```

CCT®

A PAT MARTINI COMPANY

Compliance Computer Technology, Inc.

CCT Bldg • 6476 Airpark Drive • Prescott, AZ 86301

For information and technical support: 602-776-1188
For ordering and systems quotations: 800-222-8686

CCT will provide FREE technical help to anyone — Call!

- *CCT is one of the oldest and largest S-100 OEM's in the world. Our large user base is comprised mainly of CompuPro and Macrotech based multi-user systems. We have produced thousands of S-100 systems since 1976.*
- *CCT manufactures over 30 disk drive subsystems for S-100 machines, including 3 inch, 5 inch, and 8 inch floppies, and hard disks from 10 to 300 megabytes. These subsystems are designed and manufactured to the industrial quality requirements of our large governmental and large business user base.*
- *CCT sells, services, and supports the complete lines of both Macrotech and CompuPro products. We can provide turn-key packages to upgrade your present system using state-of-the-art hardware and the latest generation operating system releases.*

Limited Special for all S-100 users: The CCT PRINTERFACER 1

The Printerfacer 1 is a self contained standard IEEE-696 S-100 board which is both an output buffer and device interface. The board contains a Z80 processor, serial and parallel I/O ports, operating firmware, and up to 1 megabyte of on-board memory. The Printerfacer 1 accepts output from the system at bus speed, while buffering characters and maintaining interface with a parallel and a serial printer and/or plotter. This speeds up throughput tremendously by relieving the system of slow handshaking protocol. With a simple command the Printerfacer 1 output can be toggled between the parallel and serial devices, thus allowing accessability to both at the same location, at the same time, without any software or operating system changes!

The Printerfacer 1 will work in any S-100 system, in a polled or interrupt driven environment, and also has a special CompuPro emulation mode to appear as any user port of the Interfacer 3 or 4. The board comes standard with 256K and may be expanded to 1 megabyte by simply plugging-in standard 256K bit DRAM chips. The Printerfacer 1 tests memory and self-sizes at each cold start.

CCT will configure and test the board for your specific system requirements, and can also furnish information on addressing as the primary LST device or at another device location. Plug-in lightning output speed and double your device capability.

Limited Time Special: $249.00 plus shipping.

For Concurrent DOS users:
CCT CMX Concurrent DOS operating system

The CCT CMX Concurrent DOS operating system is written for end user flexibility and speed. The CMX 2.1 release supports any floppy and hard disk combination. The CMX XIOS is designed to include drivers, tables, and buffer space for only those devices actually in use in the particular system. The end result: A more compact, efficient, and faster environment in less memory space. Run CP/M 8 bit and 16 bit software, and MS-DOS applications under the most flexible micro operating system today. Upgrades available to CompuPro licensed users.

CCT Implements Tomorrow's Technology Today!™

• Z Best Sellers •

Z80 Turbo Modula-2 (1 disk) $89.95
The best high-level language development system for your Z80-compatible computer. Created by a famous language developer. High performance, with many advanced features; includes editor, compiler, linker, 552 page manual, and more.

Z-COM (5 disks) $119.00
Easy auto-installation complete Z-System for virtually any Z80 computer presently running CP/M 2.2. In minutes you can be running ZCPR3 and ZRDOS on your machine, enjoying the vast benefits. Includes 80+ utility programs and **ZCPR3: The Manual**.

Z-Tools (4 disks) $169.00
A bundle of software tools individually priced at $260 total. Includes the ZAS Macro Assembler, ZDM debuggers, REVAS4 disassembler, and ITOZ/ZTOI source code converters. HD64180 support.

PUBLIC ZRDOS (1 disk) $59.50
If you have acquired ZCPR3 for your Z80-compatible system and want to upgrade to full Z-System, all you need is ZRDOS. ZRDOS features elimination of control-C after disk change, public directories, faster execution than CP/M, archive status for easy backup, and more!

DSD (1 disk) $129.95
The premier debugger for your 8080, Z80, or HD64180 systems. Full screen, with windows for RAM, code listing, registers, and stack. We feature ZCPR3 versions of this professional debugger.

Quick Task (3 disks) $249.00
Z80/HD64180 multitasking realtime executive for embedded computer applications. Full source code, no run time fees, site license for development. Comparable to systems from $2000 to $40,000! Request our free Q-T Demonstration Program.

Z-System OEM inquiries invited.
Visa/Mastercard accepted. Add $4.00 shipping/handling in North America, actual cost elsewhere. Specify disk format.

Echelon, Inc.
885 N. San Antonio Road • Los Altos, CA 94022
415/948-3820 (Order line and tech support) Telex 4931646

Attention CompuPro System Users

Maintenance cleaning and testing of your CompuPro S-100 Computer System.

- *Complete internal and external cleaning of enclosures and cabinets to remove harmful dust and dirt.*
- *Full testing and re-certification of hard disk.*
- *Re-pack all data on hard disk to enhance performance.*
- *Testing and calibration if required of all S-100 boards.*
- *Power supply checkout and voltage adjustments.*

$385.00*

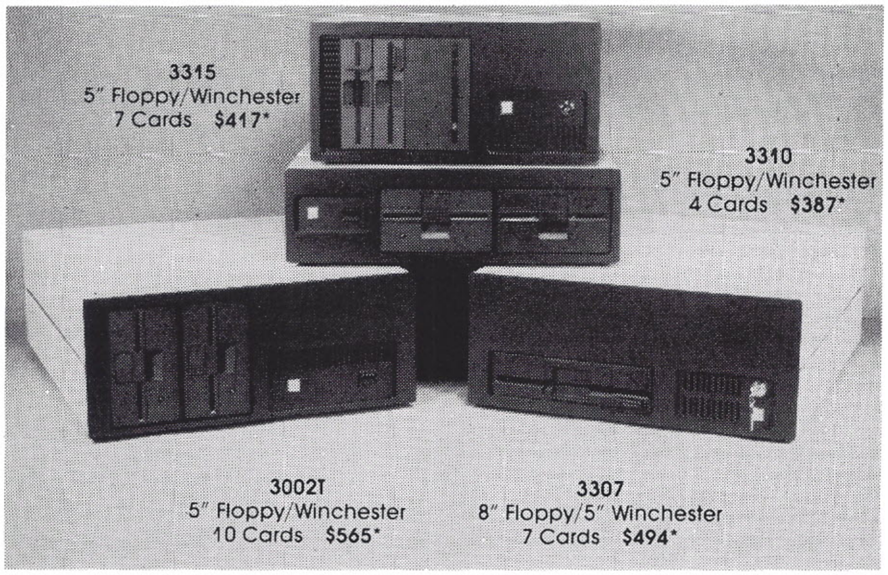
Make your system happy by giving it a tune up! Eliminate those clogged fan filters and that filth on your floppy disk drives. Don't wait until it's too late.

To schedule your appointment
please call Doug @ 415-530-2390

Vision Computers
2235 Melvin Road
Oakland, CA 94602

**price includes pick up and delivery for all Vision customers located within a 30 mile radius of Oakland, California. One day turnaround time. For customers outside the Bay Area add travel time ($60.00 per hour) and any expenses. You may elect to ship your system to and from our Oakland location at your expense.*

S100 MAIN/FRAMES
from $200



Over 50 models available • Desk and Rack
1 Card to 20 Cards • With and without Drives

*1 piece; prices lower in quantity. (Disk drives not included)

Call or write for free catalog

8620 Roosevelt Ave./Visalia, CA 93291 (209) 651-1203
TELEX 5106012830 (INTEGRAND UD)

INTEGRAND RESEARCH CORP.

Single Board Computer MAIN/FRAMES

8 Models from $125*

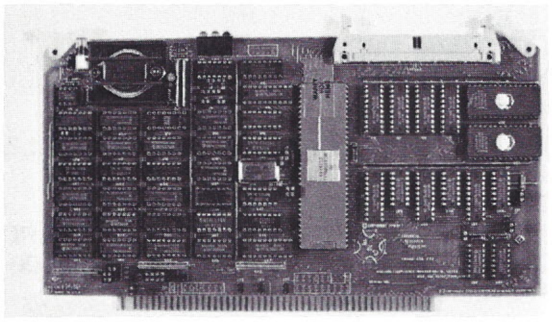
$155* *1 piece, prices lower in quantity
MODEL 3810
Includes power supply & fan
(Disk Drives and Board not included)

INTEGRAND RESEARCH CORP.

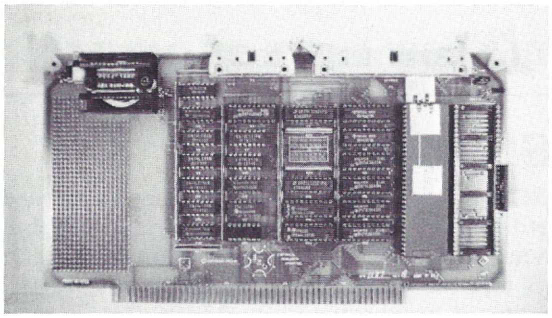
8620 Roosevelt Ave./Visalia, CA 93291
(209) 651-1203
TELEX 5106012830 (INTEGRAND UD)

IEEE-696 S-100 BOARDS

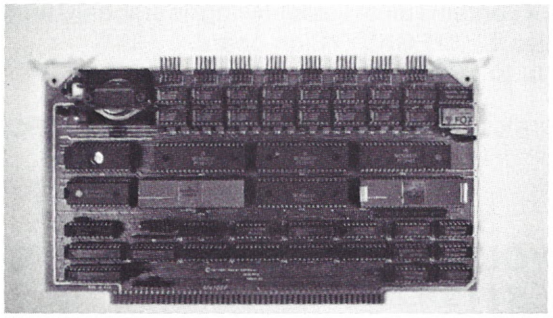
The most efficient 68000 Card Set implementation on the market



★ **ERG68-696 CPU**
68000/68010 MPU
8, 10, 12 MHz clock
ergFORTH ROM kernel and compiler
'SKY-BUS' for system expansion
Full DMA bus transfer timing logic
Auto adjusts to speed of device on bus
User-definable delay generators Price: $785.00



★ **MMU (Memory Management Unit)**
MC68451 MMU
Full user memory protection
Dynamic memory allocation Price: $625.00



★ **ACC8 Intelligent I/O Channel Card**
68008 Coprocessor (8 or 10 MHz)
Dual ported RAM
8 RS-232 serial ports
Double buffered parallel interface
On-board ROM firmware
DUART (68681) Price: $730.00

OEM PRICING ON REQUEST

NuCom CORPORATION
12515 Willows Road N.E. • Kirkland, Washington 98034
206-821-6800

WRITE FOR
S-100 Journal

You may have the information that other readers are looking for. Why not communicate it through an article in S-100 Journal?

We buy articles in a variety of subjects related to S-100 computers and operating systems running on S-100 computers. If you would like more information about writing for us, please request our Author's Guidelines or call Jay Vilhena at (801) 373-0696.

Some of the topics that we look for are:

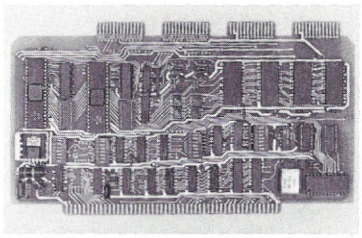
- *Assembling and upgrading S-100 systems.*
- *Hardware and software reviews.*
- *Software drivers for specific S-100 boards.*
- *S-100 application-specific implementations.*
- *Proposals for an S-100 32-bit standard.*
- *Hardware and software projects.*
- *Practical information for S-100 users.*

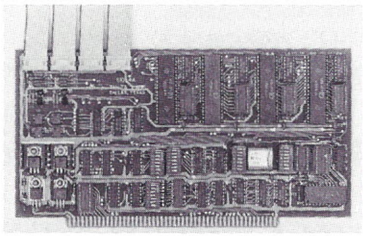
We will be happy to discuss any other topics that you would like to write about and which could be of interest to S-100 users or integrators.

FACTORY CLOSEOUT!

THETA LABS I / O BOARDS.......
While they Last!

PS10 - 2 serial, 2 parallel ports - 6 MHz with full handshaking. 26 pin card-edge connectors.





S10 - 4A 4 - port serial RS 232C current loop on 2 ports - 4 MHz. 14 DIP connector each channel.

Both boards assembled and tested. One year factory guarantee.

While they last..... $125.00

Please add 3.00 for shipping expense.

THETA LABS

P. O. Box 540635, Dallas, Texas 75354 (214) 351-6093

SYSTEMS or BOARDS, ...YOU CHOOSE

INTRODUCING THE LO-MAX, LDP'S NEWEST MULTI-USER SYSTEM.

LO-MAX is **LO**w on price and **MAX**imum on performance. LO-MAX is LDP'S newest system based on the THUNDER/PLUS. LO-MAX has the following advanced features:

- 10 Mhz 80186 Microprocessor.
- 512K bytes of no-waitstate dynamic RAM.
- Supports two users.
- CC-DOS version 4.1 runs both MS-DOS and CP/M-86 programs.
- 7 slot chasis for future expansion.
- 20 Mbyte Harddisk drive.
- SCSI interface for expansion.

OPTION:

- 1 Mbyte of no wait-state dynamic RAM.
- 8087 option.
- 1.2 Mbyte PC/AT compatible 5¼" floppy drive.
- 40 to 140 Mbyte harddisk options.
- Five user support.

S100 BOARDS/SYSTEMS

CPU's:
80186 SBC
80286 8/10MHZ

MEMORY:
STATIC RAM 1MBYTE
DYNAMIC RAM 2MBYTE

INTERFACE:
DUAL HARDDISK/FLOPPYDISK CONTROLLER
8 PORT SERIAL BOARD
SCSI ADAPTER WITH SERIAL PORTS
MULTIFUNCTION SYSTEM SUPPORT BOARD
PC COMPATIBLE GRAPHICS FOR S100 BUS SYSTEMS

SYSTEMS:
2 TO 18 USER SYSTEM

All these features make this system too great a value to pass up. If you need a multiuser system with the performance necessary to support multiple users buy our S100 BUS system, designed for the start to run multiple users.

ALL THESE FEATURES FOR AN UNBELIEVABLY LOW PRICE OF *ONLY $3695.00*

NEW S100 BUS SCSI ADAPTER

Open your system to the world of SCSI peripherals. The LDP-SCSI adapter board enables you to add numerous peripherals that use the industry standard SCSI interface standard to your S100 bus system. Today's high density harddisk drives are offering SCSI interfaces to improve performance and increase the data transfer rates, many tape drives units now support the SCSI standard. In addition to the SCSI interface, the board includes a battery protected CLOCK CALENDAR, and four RS232 ports capable of communicating at baud rates up to 38400 BAUD. Move your system into the future with the LDP-SCSI. ***ONLY $375.00***

THUNDER/PLUS

THUNDER/PLUS is a complete 16 bit S100 computer on one card. It contains all of the following as standard features:

- 512K bytes of no wait-state dynamic RAM expandable to 1 MEGABYTE on board.
- Two RS232 serial ports capable of communicating at up to 19200 BAUD.
- Centronics compatible parallel interface.
- Floppy disk controller capable of controlling a mix of 8", 5¼" and 3½" floppy drives including the PC/AT 1.2 Mbyte high density 5¼ drives.
- 10 MHZ 80186 Microprocessor.
- CONCURRENT-DOS version 4.1.

ALL THE PERFORMANCE AND FEATURES YOU REQUIRE IN A 16 BIT MICRO AT AN *AFFORDABLE $1195.00*

THUNDER/PLUS and LDP-SCSI make an unbeatable 5 user computer system

NEW LOWER STATIC RAM PRICES !!!

MEGA-S-RAM static RAM board available with up to 1 Megabyte of fast static RAM.

16K bytes ***$349*** 256K bytes ***$495*** 1/2 Mbyte ***$995*** 1 Mbyte ***$1749***

LDP

LOMAS DATA PRODUCTS, INC.

Call or write for a complete catalog of our S100 BUS BOARDS AND SYSTEMS.

(617)460-0333

182 Cedar Hill Street, Marlboro, MA 01752

MODIFY YOUR LS-100 RAM-DISK TO ONE MEGABYTE

Mark E. Noneman

In this article, I will discuss a very inexpensive 256K RAM-based Disk Emulator (or RAM-disk) S-100 board. I will then explain a relatively simple modification to this board that will quadruple the available 'disk' memory to 1 Megabyte! This article details methods and techniques used in many electronic designs, so the procedures learned here can also be applied to future electronic projects that you undertake.

LIGHT-SPEED-100 RAM-DISK BOARD

The Light-Speed-100 (LS-100) kit from Digital Research Computers of Duncanville, Texas comes complete with all the discrete components, ICs, sockets, and the Printed Circuit Board (PCB) necessary to complete the project.

The original LS-100 kit had a memory capacity of 256K, and in this article I will show how you can modify it to a full Megabyte. The blank S-100 circuit board for this RAM-disk is still available from Digital Research Computers for $24.95, so even if you do not now own the original board you may still want to build it. For those less adventurous, Digital Research now offers the LS-100 II which is the 1-Megabyte version of the RAM-disk. The price is still a very reasonable $259 for the complete kit, or $309 for the assembled and tested board (including CP/M software drivers). Obviously, if you buy the 1-Megabyte version of the board, you do not need to perform the modification described here, but you may still find some areas of the article to be of interest.

One of the most important features of any kit is the quality of the PCB. This one looks excellent. All the parts are well laid out, especially considering that the board has only two signal layers. The traces of different functions are well separated, and the memory layout appears to follow the recommendations of dynamic RAM manufacturers. This is important since the manufacturers have gone to great pains to determine what two-layer layout and signal routing gives the best noise immunity for their memory products.

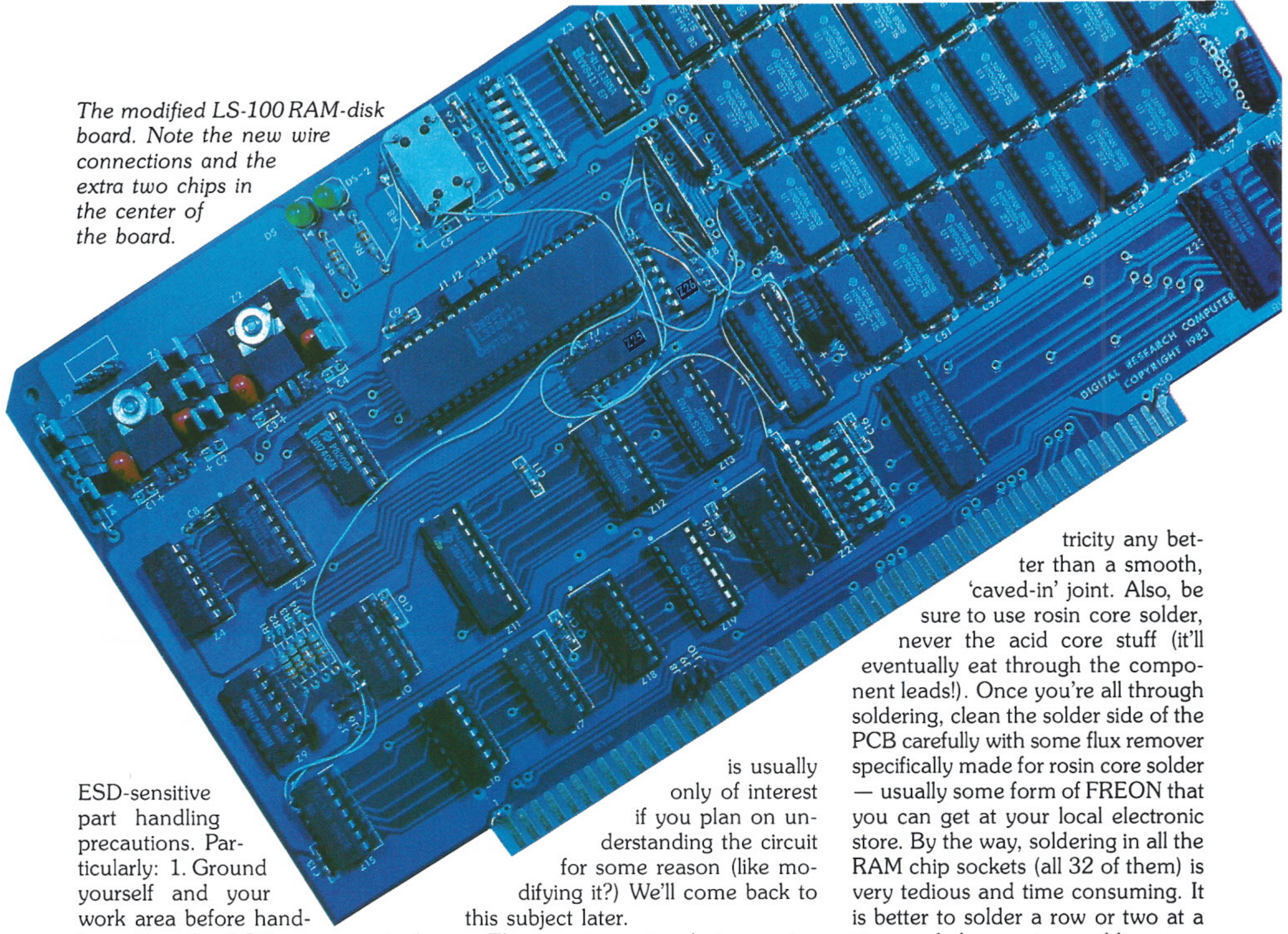
The LS-100 board has a battery-backup option. I haven't tried this feature, but the circuit looks fine from the schematics, and it is definitely a nice feature for those interested. However, you should always save the RAM-disk files to long term media such as a floppy or hard disk.

In summary, the folks at Digital Research Computers have done a great job.

DOCUMENTATION

An important aspect of any kit (or the most important if you have never built a kit before) is the quality of the instructions. If you've done any soldering before and know what a resistor and capacitor look like, the instructions provided with the LS-100 kit should get you through with very little difficulty. However, few practical hints are given. For example, you should be aware of electro-static discharge (ESD). They go to the trouble of shipping all the ESD-sensitive parts in conductive material for a reason: any static charge built up between pins on a device could destroy it. You would think they would warn the kit builder too. If you build this kit, please pay careful attention to ►

Mark Noneman is an Electrical Engineer at TRW, Inc. in the San Diego area. He is a graduate of UCLA School of Electrical Engineering and is experienced in microcomputer and electronic design.



The modified LS-100 RAM-disk board. Note the new wire connections and the extra two chips in the center of the board.

ESD-sensitive part handling precautions. Particularly: 1. Ground yourself and your work area before handling either the PCB or any parts (a cold water pipe is usually a good ground point). 2. Never handle the parts in a carpeted area. 3. Never let someone else touch you or your work area while you're handling ESD-sensitive parts. Remember, just because the parts are installed in the PCB does not mean that they are safe from static discharge. I know many people who have ruined some very expensive parts by poor handling. The memory chips and the 8203 dynamic RAM controller are particularly sensitive.

The manual explains how to setup the board (switches, etc.). This is in general very simple and following the instructions should have you powered up and running in just a few minutes. The only difficult decision to make is between *Normal Ready* and *Advanced Ready*. Just follow the manual and try Advanced Ready. If this doesn't work then switch to Normal Ready. (If this still doesn't work, better start debugging).

The manual also describes the theory of operation for the board. This is usually only of interest if you plan on understanding the circuit for some reason (like modifying it?) We'll come back to this subject later.

There are two minor but annoying problems with the documentation. First of all, the schematic diagram is hand drawn. The boxes and lines are OK, but the handwritten lettering on the signal names and pin numbers can be difficult to read. The second irritation is an inconsistency. The resistor packs for the board are labeled RP1, RP2, and RP3 on the schematic but Z8, Z21, and Z24 on the parts placement diagram and PCB silk-screen legend. For everyone's information, RP1 is really Z21, RP2 is Z8, and RP3 is Z24.

SOME BASIC GUIDELINES

Although the manual gives clear step-by-step insertion instructions, a few suggestions for the uninitiated are in order (those of you who have soldered PCBs before can skip this section). Only use a little solder. Just enough to have a smooth curvature from the solder pad up the component lead. You don't want to gob on the stuff — it won't conduct electricity any better than a smooth, 'caved-in' joint. Also, be sure to use rosin core solder, never the acid core stuff (it'll eventually eat through the component leads!). Once you're all through soldering, clean the solder side of the PCB carefully with some flux remover specifically made for rosin core solder — usually some form of FREON that you can get at your local electronic store. By the way, soldering in all the RAM chip sockets (all 32 of them) is very tedious and time consuming. It is better to solder a row or two at a time and then rest or solder some other sockets on the board. This might help reduce sloppy soldering errors.

When you're through soldering all the discrete parts and sockets, you're ready to install the ICs. Usually, new IC packages have the pins bent outward at about 10 degrees from perpendicular. If you don't have an IC insertion tool and you don't plan to get one, here's what you do before installing the ICs on the sockets: Hold the IC on the edges without the pins (never hold the IC by the pins: static electricity again). Set the row of pins so that they are almost horizontal, points down, on some conductive surface (aluminum foil will work but make sure that both the foil and you are solidly grounded to earth). Now, *very very gently*, rotate the body of the IC so that the entire row of pins gets bent toward perpendicular. Reverse your grip and do the same thing to the other side. The small devices are pretty easily bent. The large ones (like the 8203) take quite a bit of pressure, so be careful. Once the pins are bent under the body of the IC, it's very difficult to straighten them out without breaking them. If you follow this procedure (or use an insertion tool), the chances of bending leads under the body while you're installing the IC into the socket are greatly reduced.

SOFTWARE

There are three major CP/M 2.2 programs that come with the LS-100 board: *DIAG*, *FMAT*, and *INSTALL*. **DIAG** is a software diagnostic program that checks out (rather simply) the operation of the LS-100 board. This is very helpful when you build the board from a kit. This program should indicate whether any parts of the board are not working or, hopefully, that the board seems to be OK.

FMAT is a 'disk' formatting program. Executing this program formats the RAM disk in a way similar to how a floppy disk is formatted (except it only takes a few seconds). The major differences are that a reserved area is formatted with 80H and that no sector header and gap information needs to be written (as is required for floppy or hard disks). The reserved area is used by the INSTALL software to hold a checksum of each sector in the RAM-disk. The data area is filled with E5H just as with a normal floppy or hard disk.

INSTALL is a program which installs the RAM-disk driver just below the CP/M CCP (Console Command Processor). It reassigns certain BIOS and jump assignments so that this installation is almost invisible to calling programs and the RAM-disk can be accessed just like any other disk drive. Unfortunately, you don't get something for nothing. This simple program eats away from the TPA (Transient Program Area) from the top down. This has several unwanted problems, some of which are explained in the manual. Also, problems occur for some compiled programs. For example, when Turbo Pascal compiles a program, it finds the top of the TPA by looking at the BDOS jump address at absolute addresses 6 and 7 and uses this, minus a few hundred bytes, as the base address for the CPU stack. If you use the standard compile (to .COM) options before the RAM-disk is installed and then try to run the

LISTING 1 — PROGRAM Diag1M.PAS

PASCAL PROGRAM TO TEST THE MODIFIED LS-100 BOARD

Program written by Mark E. Noneman — Copyright © 1986 by Mark E. Noneman

```
PROGRAM Diag1M ( INPUT , OUTPUT );

{
   This program tests the LS-100 RAM disk board using pseudorandom
   numbers.  It generates a unique pattern of 2^20 - 1 byte values
   which are written to the disk and then read back to verify that
   the sequence of bytes is the same.  This program takes a very
   long time to run -- over 3 hours on my 4-MHz Z80 machine!
   The assembly language version runs much faster (about 2 minutes).
}

CONST
   MaxMSB = 32;          {count to this minus 1 for MSB register}
   MSBPort = $D1;        {I/O port for MSB register}
   LSBPort = $D2;        {I/O port for LSB register}
   DataPort = $D0;       {I/O port for data}
   BoardMax = 1;         {maximum board number to be entered (7 max) }

TYPE
   ShiftRegister = ARRAY [ 0..19 ] OF BOOLEAN;

VAR
   BoardNum : BYTE;
   Bank : ARRAY [ 0..3 ] OF BYTE;

  PROCEDURE Initialize;

  VAR
    I : INTEGER;

  BEGIN     {Initialize}
    WRITELN;
    WRITELN ( 'Light Speed 100 - One Megabyte Pseudorandom Test Program' );
    WRITELN;
    REPEAT
        WRITE ( 'Enter the board number to test (0-' , BoardMax , ') : ' );
        READLN ( BoardNum )
    UNTIL ( ( BoardNum >= 0 ) AND ( BoardNum <= BoardMax ) );
    BoardNum := BoardNum * 32;
    FOR I := 0 TO 3 DO
         Bank [ I ] := 0
  END;      {Initialize}

  PROCEDURE Clear ( VAR SR : ShiftRegister );    {zero out SR}

  VAR
    I : INTEGER;

  BEGIN     {Clear}
    FOR I := 0 TO 19 DO
        SR [ I ] := FALSE
  END;      {Clear}

  PROCEDURE LFSR ( VAR B : ShiftRegister );
  {  This routine implements a linear-feedback-shift-register
     twenty bits in length of maximal sequency (ie: 2^20-1).
     The formula is: bit 0 = NOT ( bit 16 XOR bit 19 ) while
     bits 1 through 19 are shifted from bits 0 through 18 re-
     spectively.
  }

  VAR
    Temp : BOOLEAN;
    Ptr : INTEGER;

  BEGIN      {LFSR}
    Temp := NOT ( B [ 16 ] XOR B [ 19 ] );
    FOR Ptr := 19 DOWNTO 1 DO
        B [ Ptr ] := B [ Ptr - 1 ];
    B [ 0 ] := Temp
  END;       {LFSR}

  FUNCTION Data ( SR : ShiftRegister ) : BYTE;

  {  Converts the LSByte (BOOLEAN) to binary data. }

  VAR
    B,
    T,
    Ptr : BYTE;
```

program after installing the RAM-disk, you'll get:

Not enough memory
Program Aborted

This happens because INSTALL alters the JMP address. Fortunately, it can be avoided with Turbo Pascal by changing the Options menu to a lower End Address whenever compiling to a .COM file.

By the way, if you hardware reset your computer, the data in the RAM-disk isn't lost. All you do is type INSTALL again to let CP/M know about the RAM-disk, and all your data will still be on that drive — as long as there wasn't any power glitch.

Of course, changing your BIOS will eliminate all of these problems and also negate having to type INSTALL every time you boot up or reset. This method is a little more complex in the short run but much more satisfying once you are done. The instructions in the manual are short but complete, and you shouldn't have any difficulty following their example code. If you have never done any assembly language or BIOS programming before, you'd better get a good book for CP/M beginners.

ONE-MEGABYTE MODIFICATION

The LS-100 is definitely a good product just as it is. But, I wanted to make it better. Sure, 256K of RAM is pretty good, but it's very easy to eat all of that up. How can we make the RAM-disk space bigger? The LS-100 uses 64K-bit dynamic RAM chips. The relatively new 256K-bit dynamic RAM chips come in a compatible package yet increase the number of bits per device by four times. We are going to replace the thirty-two 64K RAMs used in the board with new 256K RAMs. (256K when referring to the chip means 256K *bits*, not bytes.)

Before we go on, however, it's important to understand that the following modification is not approved in any way by Digital Research Computers. In fact, since we are modifying the board to some degree (although we don't have to cut any traces), the warranty is specifically void. I'll try to give you as much information as I can and, if you're careful, this modification should work

```
  FUNCTION Power2 ( X : BYTE ) : INTEGER;

  VAR
    I,
    Temp : INTEGER;

  BEGIN        {Power2}
    Temp := 1;
    FOR I := 1 TO X DO
        Temp := Temp * 2;
    Power2 := Temp
  END;           {Power2}

BEGIN     {Data}
  T := 0;
  FOR Ptr := 0 TO 7 DO
  BEGIN
    IF SR [Ptr] THEN B := 1 ELSE B:= 0;
    T := T + Power2 ( Ptr ) * B
  END;
  Data := T
END;        {Data}

PROCEDURE WriteDisk;

VAR
  Bits : ShiftRegister;
  MSB,
  LSB,
  Byt : BYTE;

 BEGIN    {WriteDisk}
   Clear ( Bits );    { reset SR -> seed =0 }
   FOR MSB := 0 TO ( MaxMSB - 1 ) DO  {fill for each MSB register}
   BEGIN
     PORT [ MSBPort ] := MSB + BoardNum;
     FOR LSB := 0 TO 255 DO          {fill for each LSB register}
     BEGIN
       PORT [ LSBPort ] := LSB;
       FOR Byt := 0 TO 127 DO       {fill every byte location}
       BEGIN
         PORT [ DataPort ] := Data ( Bits );
         LFSR ( Bits )            {shift data pseudo-randomly}
       END
     END
   END
 END;        {WriteDisk}

PROCEDURE ReadDisk;

VAR
  Bits : ShiftRegister;
  MSB,
  LSB,
  Byt,
  D,
  DExpected : BYTE;

   PROCEDURE BadChip ( X , Y : BYTE );

   {   Stores each detected bad bit in the array of banks. }

   BEGIN        {BadChip}
     Bank [ MSB AND 3 ] := ( Bank [ MSB AND 3 ] ) OR ( X XOR Y )
   END;         {BadChip}

 BEGIN        {ReadDisk}
   Clear ( Bits );
   FOR MSB := 0 TO ( MaxMSB - 1 ) DO   {for each MSB register}
   BEGIN
     PORT [ MSBPort ] := MSB + BoardNum;
     FOR LSB := 0 TO 255 DO           {for each LSB register}
     BEGIN
       PORT [ LSBPort ] := LSB;
       FOR Byt := 0 TO 127 DO       {for every byte location}
       BEGIN
         D := PORT [ DataPort ];
         DExpected := Data ( Bits );
         IF D <> DExpected THEN
             BadChip ( D , DExpected );
         LFSR ( Bits )
       END
     END
   END
 END;          {ReadDisk}

PROCEDURE OutResults;

VAR
  I : INTEGER;
```

```
   BEGIN        {OutResults}
     WRITELN ( 'Test Results:' );
     WRITELN;
     WRITELN ('         D0 D1 D2 D3 D4 D5 D6 D7' );
     FOR I := 0 TO 3 DO
     BEGIN
       WRITE ( 'BANK ' , I , '  ' );
       IF (Bank [I] AND 1) = 0 THEN WRITE ( 'G  ' ) ELSE WRITE ( 'B  ' );
       IF (Bank [I] AND 2) = 0 THEN WRITE ( 'G  ' ) ELSE WRITE ( 'B  ' );
       IF (Bank [I] AND 4) = 0 THEN WRITE ( 'G  ' ) ELSE WRITE ( 'B  ' );
       IF (Bank [I] AND 8) = 0 THEN WRITE ( 'G  ' ) ELSE WRITE ( 'B  ' );
       IF (Bank [I] AND 16) = 0 THEN WRITE ( 'G  ' ) ELSE WRITE ( 'B  ' );
       IF (Bank [I] AND 32) = 0 THEN WRITE ( 'G  ' ) ELSE WRITE ( 'B  ' );
       IF (Bank [I] AND 64) = 0 THEN WRITE ( 'G  ' ) ELSE WRITE ( 'B  ' );
       IF (Bank [I] AND 128) = 0 THEN WRITE ( 'G  ' ) ELSE WRITE ( 'B  ' );
       WRITELN
     END;
   WRITELN;
   END;         {OutResults}

BEGIN        {program}
   Initialize;
   WRITE ( 'Writing to RAM disk . . . ' );
   WriteDisk;
   WRITELN ( 'Done.' );
   WRITE ( 'Reading from RAM disk . . . ' );
   ReadDisk;
   WRITELN ( 'Done.' );
   OutResults
END.         {program}
```

LISTING 2 — PROGRAM LSCHK.ASM

ASSEMBLY LANGUAGE PROGRAM TO TEST THE MODIFIED LS-100 BOARD

Program written by Mark E. Noneman — Copyright © 1986 by Mark E. Noneman

```
          title     'Pseudorandom RAM-Disk Test'
          maclist   off
;
;    This program tests the LS-100 RAM disk board from Digital
;    Research Computers.  The current version assumes that a minimum
;    of one megabyte of RAM disk memory exists.  The modified board
;    described or four standard boards will qualify.
;
;    The following macros are used to make the program a little more
;    "user friendly."  If you don't have a macro assembler, the only
;    routine that must be coded in is the print macro.  All the others
;    can be left out.
;
version:  .MACRO   ARG1
          jp   varound
          db   ARG1
varound:  equ  $
          .ENDM

print:    .MACRO   ARG1
          push bc
          push de
          push hl
          ld   de,ARG1
          ld   c,9
          call bdos
          pop  hl
          pop  de
          pop  bc
          .ENDM

abort?:   .MACRO
          push af
          push bc
          push de
          push hl
          ld   c,11
          call bdos
          or   a
          jp   nz,exit
          pop  hl
          pop  de
          pop  bc
          pop  af
          .ENDM
          page
;
; This program is designed around the concept of pseudorandom
; testing.  It generates a unique pattern of 2^20 - 1 byte values.
; These are written to the RAM-disk and then reread and compared
```

just fine. But be forewarned, if the board stops working at any time, either from the modification itself or for any other reason, you're on your own. If you bought the LS-100 in kit form, make certain that the board is assembled and in perfect working order before starting this modification.

Back to the drawing board. To understand the following discussion, it would be helpful to have the schematic diagram of the LS-100 board, the specification sheet for the 8203 dynamic RAM controller, a specification sheet for a 64K-bit dynamic RAM chip (any manufacturer will do), and a specification sheet for a 256K-bit dynamic RAM chip (again, any manufacturer). Also, a cursory understanding of the theory of operation of the board, as given in the manual, will be helpful. I will not explain completely how the board works, only what's important to understand for the modification.

DYNAMIC RAM CHIPS

In order to have a good perception of the modification described in this article, we need to examine the requirements of dynamic RAMs. This is not a tutorial on dynamic RAMs, just a look at the features that concern us directly.

Addressing

First of all, most dynamic RAMs have a multiplexed address scheme. That is, the total address space (say 16 bits) is split into two parts (8 bits each — called the ROW address and COLUMN address) that are applied to the chip one at a time. To indicate which of the two parts, control signals are used; usually called Row Address Strobe (RAS*) and Column Address Strobe (CAS*) — the * means an active low signal. This keeps the package pin count smaller but complicates the interface. The 64K RAMs have 16 address bits. The 256K RAMs have 18 address bits. These two extra bits will be multiplexed (like the other bits) onto one package pin. There are several options as to where the two extra address bits will come from; this will be discussed later.

An 8203 dynamic RAM controller

takes care of the 16 address bits for the 64K RAM but has no provisions for the additional two bits required by the 256K RAMs. We'll have to multiplex them ourselves when implementing our modification.

To *address* a dynamic RAM chip, several steps must be taken to either read or write data. First, the row address signals (the eight LSB address bits) are applied to the RAM chip address lines. Then, after an appropriate setup time, the signal RAS* is set low. After a suitable hold time, the other eight address bits are multiplexed to the RAM chip address lines and, after another setup delay, the CAS* line is set low. See Figure 1 (page 70) for a timing diagram of this process. This applies all sixteen address bits to the RAM chip using only 8 address pins. Then, data can be read out of the RAM address, or data can be applied and a write strobe issued. The 8203 handles all of this timing and multiplexing in one package. Notice that the timing diagram for the 8203 in Figure 1 shows at which clock cycles the RAS* and CAS* lines become active. The clock cycle between these two signals is when the address line multiplexing occurs. This scheme provides about 50 nanoseconds (ns) of hold time for RAS* and the same amount for CAS* setup time; more than adequate.

Refreshing

Dynamic RAM chips are so called because they hold the state of each bit (a one or a zero) by the charge on a capacitor and that charge must be *refreshed* periodically before it leaks away. Static RAMs vary in that each bit is held by a transistor configuration called a latch. This latch does not require refreshing but requires at least twice the integrated circuit area and considerably more power than a dynamic cell. That's why dynamic RAM chips are so popular: low power and less circuit board area. The 8203 handles dynamic RAM refreshing.

To refresh the RAM chip, the 8203 controller waits until there is no read or write request on the bus. Then, it applies the next address to be refreshed (held by an internal 8-bit counter) to the RAM address lines and forces

```
; with the original sequence.  This method completely checks out
; the addressing, refreshing, buffering, and RAM chips of the entire
; board.  Any addressing errors would overwrite previously written
; data and show up as an error.
;
; This program continues to run forever, or until any key is pressed.
; To minimize time impact (to check for a pressed key), the macro
; abort? is only run between major events (write, read, output) and
; between MSB register updates (32 times during writes and reads).
;
; The GetBoard routine inputs a valid MSB register number to indicate
; board number.  It really places these bits in the three most signi-
; ficant bits of the MSB value.  The equate BoardMax sets the maximum
; allowed value.  With the standard modification, only two boards can
; exist and BoardMax equals one.  Change this for different mods.
;
;
; Change the following equates for your system and modification.
;

MSBPort:   equ  0d1h       ;MSB address I/O port
LSBPort:   equ  0d2h       ;LSB address I/O port
DataPort:  equ  0d0h       ;data I/O port
MaxMSB:    equ  31         ;maximum value for MSBPort
MaxBank:   equ  4          ;banks 0-3
BoardMax:  equ  1          ;maximum valid MSBAdd value (7 maximum)
           page

lschk:     version    'LSCHK V1.0 13-May-86  (c) 1986 Mark E. Noneman'

           ld   (oldsp),sp      ;save old stack pointer
           ld   sp,stack        ;get new SP


           xor  a               ;clear a and flags
           ld   hl,BytBuf0      ;set pointer
           ld   b,4             ;set counter
clbuf:     ld   (hl),a          ;write data
           inc  hl              ;next byte
           djnz clbuf           ;for b bytes

           print     headmsg

           call GetBoard        ;get MSB value 3 MSBits (0-7)

wrstart:   print     writemsg

           abort?
           call write           ;fill disk

           abort?
           print     readmsg
           call read            ;read entire disk

           call output          ;write out results

           abort?
           jp   wrstart         ;keep going until keypressed

exit:      print     tailmsg

           ld   sp,(oldsp)      ;restore old stack pointer
           ret                  ;normal return

           page
;
;     This routine gets the number between 0 and BoardMax that will
;     be added to the 3 MSB positions of the MSB Register value for
;     the LS-100 board.
;
GetBoard:  print     BdPrompt

           ld   c,1
           call bdos            ;get result
           cp   '0'             ;must be >=0
           jp   m,GetBoard      ;wrong
           cp   BoardMax+'0'+1  ;must be <=0
           jp   p,GetBoard      ;wrong

           and  07h             ;get to binary
           sla  a               ;
           sla  a               ;
           sla  a               ; multiply by 32
           sla  a               ;
           sla  a               ;
           ld   (MSBAdd),a      ;store final number

           ret
;
;     The write routine writes out all 2^20 bytes of data to the RAM
;       disk.  It writes from the maximum MSB value down to zero with
;       256 "sectors" per MSB and 128 bytes per "sector."  At every new
;       MSB value, the keyboard is checked to see if a key has been
;       pressed.  If so, the program aborts to exit.
;
write:          ld    b,MaxMSB
                xor   a          ;reset A and flags
                ld    c,a
                ld    e,a
                ld    h,a
                ld    l,a

WNewM:          ld    a,b                ;get MSByte
                push  bc                 ;save b
                push  hl                 ;save hl
                ld    hl,MSBAdd          ;get offset value
                add   a,(hl)             ;add to MSByte
                pop   hl                 ;get it back
                out   (MSBPort),a        ;set MSB address byte

WNewL:          ld    a,c                ;get LSByte
                out   (LSBPort),a        ;set LSB address byte
                ld    d,BytPerSect       ;set D counter for byte count

WNewB:          ld    a,e                ;get LSByte
                out   (DataPort),a       ;write data
                call  lfsr               ;shift for new pseudorandom number
                dec   d                  ;count off bytes
                jp    nz,WNewB           ;keep going

                inc   c                  ;next LSByte address
                jp    nz,WNewL           ;keep going

                abort?
                pop   bc
                dec   b                  ;next MSByte (down)
                jp    p,WNewM            ;if >=0, go again

                ret
                page
;
;       The read routine reads all of the 2^20 bytes of data from the RAM-
;       disk.  It reads data in the same order as the write routine.  At
;       each read, the data is checked against what it's supposed to be.
;       If it's OK, continue on but if it's wrong, store the bits that are
;       wrong using the routine BadBits.  At every new MSB value, the
;       keyboard is checked to see if a key has been pressed.  If so, the
;       program aborts to exit.
;
read:           ld    b,MaxMSB
                xor   a                  ;reset A and flags
                ld    c,a
                ld    e,a
                ld    h,a
                ld    l,a

RNewM:          ld    a,b                ;get MSByte
                push  bc                 ;save b
                ld    (MSBVal),a         ;store MSB in memory for BadBit
                push  hl                 ;save hl
                ld    hl,MSBAdd          ;get offset value
                add   a,(hl)             ;add to MSByte
                pop   hl                 ;get it back
                out   (MSBPort),a        ;set MSB address byte

RNewL:          ld    a,c                ;get LSByte
                out   (LSBPort),a        ;set LSB address byte
                ld    d,BytPerSect       ;set D counter for byte count

RNewB:          in    a,(DataPort)       ;read data
                xor   e                  ;=0?
                call  nz,BadBit          ;set bad bit if not

                call  lfsr               ;shift for new pseudorandom number
                dec   d                  ;count off bytes
                jp    nz,RNewB           ;keep going

                inc   c                  ;next LSByte address
                jp    nz,RNewL           ;keep going

                abort?
                pop   bc
                dec   b                  ;next MSByte (down)
                jp    p,RNewM            ;if >=0, go again

                ret
                page
;
;       The lfsr routine creates a linear-feedback-shift-register twenty
;       bits long of maximal sequence.  This is formed be shifting into the
```

the RAS* line to go low (with the write strobe held false, of course). The CAS* line does not go low, so the RAM never completes its address mode but instead it refreshes the row of capacitors associated with the row address. This process is continued whenever no read or write request is made of the 8203.

Since RAM chips of different sizes have a different number of rows associated with them, each type requires a different number of row addresses to be accessed in order to refresh the entire RAM. Also, a certain time limit — the refresh period — exists for data retention to be guaranteed. For most standard 64K RAMs (the notable exception being Texas Instruments), there are 128 refresh rows (address bits A0 through A6) and a 2-millisecond (ms) refresh period. For all standard 256K RAMs, there are 256 refresh rows (address bits A0 through A7) and a 4ms refresh period. The 8203 on the LS-100 is set up to refresh 256 rows with a 4ms period on address its A0 through A7. The 64K RAMs included (unless TI RAMs are used) ignore bit A7 during refresh. The way the LS-100 is designed, the 8203 will handle all RAM refresh for us without any modification as long as standard (256 rows, 4ms refresh) 256K RAMs are used.

DESIGN OF THE MODIFICATION

As mentioned earlier, the 256K RAMs require two extra address bits which we must supply in order to address the entire memory space. Also, the address line multiplexing must follow the same scheme used by the 8203 so that the RAMs will be properly addressed. In order to do this, we need several items. First of all, we need a clock with the same frequency and phase as that of the 8203. Second, we need a signal which indicates when any of the four RAM-bank RAS* lines is true (each of the four 64K banks has its own RAS* line; these *four* RAS* lines are generated from the *two* 8203 RAS lines using external logic), so that the time to multiplex can be known. Third, we need a multiplexer to mux the two additional address lines.

Fourth, a register is required in order to delay the time of the address line multiplexing, so it lines up with that of the 8203. Finally, some means of applying the multiplexed address signal to all 32 RAM chips must be provided. Figure 2 (page 70) shows a general schematic for these modifications. Figure 3 (page 71) and the following discussion explain the modifications in detail.

Regarding the clock, the 8203 uses an internal crystal oscillator and the LS-100 simply provides an external crystal. It might be possible to apply one of the crystal legs to a driving transistor or a buffer with input hysteresis, but the reliability and stability of the clock would be questionable. A better option is to replace the crystal with a crystal oscillator package of the same frequency (20 MHz). This package provides a very stable, TTL output oscillating signal. In order to do this, the XTAL1 input of the 8203 must be tied to VCC and the XTAL2 input tied to the oscillator source. Even though the crystal oscillator package costs a few dollars (should be less than five dollars anywhere), the extra stability and ease of use is well worth it. We are going to have to route a 20-MHz wire across the PCB and every bit helps.

It turns out that the second item, a RAS*-detector signal, won't cost us anything. There is a spare NAND gate (Z10) on the LS-100 board. Whenever either of the two 8203 RAS lines is low, a RAS* operation is taking place. If we NAND these two lines together, the output will be high whenever any of the four RAM-bank RAS* lines is low.

The third item, the multiplexer, needs to be added to the board. We need a somewhat-fast 2:1 multiplexer. Looking in the data books, it looks like the 74LS157 will work fine. Also, this part should be easily available.

The fourth item, the register, also has to be added to the board. A register capable of 20-MHz operation is required. At first it looks like a 74LS74 might work, but we will see later that a 74S74 is actually required.

Finally, Digital Research Computers kindly provided us with a very simple method of connecting the muxed address line to the RAM chips. On the schematic, the resistor in the resistor network (pin 2 of Z8 or RP2) pulls up

```
;    zero bit the exclusive-NOR of bits 16 and 19 and simultaneously
;    shifting all bits toward the MSB.
;
lfsr:      ld    a,l
           and   1           ;save bit 16
           ld    b,a
           ld    a,l
           and   00001000B   ;save bit 19
           rra
           rra
           rra
           xor   b           ;xor them both
           cp    0           ;SEE IF A=0 (COMPARE NOW)
           scf               ;SET CARRY FLAG
           jp    z,shift     ;IF A=0 THEN SKIP CLEAR CARRY
           ccf               ;reset if not equal (X-NOR)
shift:     ld    a,e         ;get LSB
           rla               ;shift
           ld    e,a
           ld    a,h         ;get middle byte
           rla               ;shift
           ld    h,a
           ld    a,l         ;get MSB
           rla               ;shift
           ld    l,a

           ret
           page
;
;    The BadBit routine accepts, in the accumulator, a bit map of bits
;    considered to be bad (xor of read data with good data).  This is
;    OR'ed with the stored map of bad bits (if any) so that a continuous
;    look at bits is kept over the test.  (The bad bit map is NOT reset
;    between runs of the test so that a history of failures can be
;    seen at every print out.
;
BadBit:    push af           ;save a
           exx
           ld    a,(MSBVal)  ;get MSB Value
           and   6           ;get bits 2 and 1
           rrca              ;rotate down (divide by 2)

           ld    hl,BytBuf0-1       ;set hl pointer to before buffer
incptr:    inc   hl          ;next position
           dec   a           ;count to bank
           jp    p,incptr    ;keep going if a >= 0

           pop   af          ;get a back
           or    (hl)        ;directly OR with original
           ld    (hl),a      ;save it

           exx               ;get old reg's back

           ret
           page
;
;    The output routine prints out to the screen (console) the bad
;    bit map created during the read routine (via BadBit).  If the
;    bit is a zero, a "G" is printed; otherwise, a "B" is printed.
;
output:    xor   a                 ;clear a and flags
           ld    (BankN),a         ;clear bank number
           ld    hl,BytBuf0        ;set pointer

           print     outhead         ;write output header

NextBank:  call WriteBank          ;write out "BANK x  "

           ld    a,(hl)            ;get byte
           ld    b,8               ;number of bits

NewBit:    rra                     ;get LSB into carry
           push af                 ;save a
           jp    c,Bad

           print     GoodBit         ;good bit
           jp    ContOut

Bad:       print     BdBit          ;bad bit

ContOut:   pop   af                ;get a back
           djnz NewBit             ;

           print     crlf            ;write cr and lf

           ld    a,(BankN)         ;get bank number
           inc   a                 ;next bank
           cp    MaxBank           ;covered all banks?
           ret   z                 ;return if done
```

```
        ld   (BankN),a       ;put new bank num back
        inc  hl              ;next bank
        jp   NextBank
        page
;
;    The WriteBank routine simply prints (to the console) out the
;    bank number of the current bad bit map we're going to print in
;    the output routine.
;
WriteBank:     push bc
               push de
               push hl

               print     Bank

               ld   a,(BankN)
               add  a,'0'
               ld   e,a
               ld   c,2
               call bdos

               ld   e,' '
               ld   c,2
               call bdos

               pop  hl
               pop  de
               pop  bc

               ret
        page
;
;       Global equates for program LS100Chk.
;
bdos:   equ  0005h       ;bdos function call address
printf: equ  9           ;print function
cr:     equ  13          ;
lf:     equ  10          ;

BytPerSect:    equ  128  ;number of bytes per disk sector

crlf:   db   cr,lf,'$'

headmsg:  db   cr,lf,lf
          db   '                LS-100 Diagnostic Program',cr,lf
          db   '              Test by Pseudorandom Numbers',cr,lf
          db   '$'

BdPrompt: db   cr,lf,lf
          db   'Enter the board number to test (0-'
          db   BoardMax+'0',') : '
          db   '$'

writemsg: db   cr,lf,lf
          db   ' (Strike any key to abort test.)',cr,lf,lf
          db   'Writing pseudorandom data to disk . . .'
          db   '$'

readmsg:  db   ' done.',cr,lf
          db   'Verifying data from disk . . .'
          db   '$'

outhead:  db   ' done.',cr,lf,lf
          db   '         D00 D01 D02 D03 D04 D05 D06 D07',cr,lf
          db   '---------------------------------------',cr,lf
          db   '$'

GoodBit:  db   ' G ','$'
BdBit:    db   ' B ','$'

Bank:     db   'Bank ','$'

tailmsg:  db   cr,lf,lf
          db   ' Program done.',cr,lf
          db   '$'

MSBVal:   db   0         ;This is the current MSB register value
MSBAdd:   db   0         ;This value is added to MSBVal (0-7 max)
BankN:    db   0         ;This is used to temporarily hold the bank
                         ; number for Output routine.
BytBuf0:  db   0         ;This buffer holds the bank bit indicators:
BytBuf1:  db   0         ; a one in a bit indicates a bad chip.
BytBuf2:  db   0
BytBuf3:  db   0

stackbuf: ds   200       ;plenty of stack space
stack:    dw   0
oldsp:    dw   0

          end
```

— END OF LSCHK.ASM LISTING —

pin 1 on all RAM chips. This is because some 64K RAMs have a special method of refreshing that utilizes pin 1; most RAMs don't use this method and their pin 1 is labelled NC. On the LS-100, pin 1 is tied high via the resistor to turn this feature off. Luckily for us, all we need to do is connect our muxed address line to this resistor, at the end that connects to the RAM pin 1 (the A8 address pin on 256K RAMs), and we're all set.

Now, the main question is: Where do the two additional address bits come from? These are the signals marked A18 and A19 in Figure 2. There are many possible solutions, but the one I recommend is to use the two LSB board-select bits. These bits come from Z14 pins 9 and 15 and, in the unmodified LS-100 board, are connected to pins 1 and 2 of Z3. This solution has several advantages. First, it only requires two wires. Second, and perhaps most important, it will simply make this one physical board appear as **four** standard LS-100 logical boards to the software. That's right, virtually no software changes are required with this modification. Only if the driver is installed in your BIOS would a change be required, and that's only the table with the disk parameter information at label DPS(X): (to tell the software that you have 1 Megabyte of RAM-disk).

The only disadvantage with this scheme is that the total possible number of physical boards in a system is reduced to two. And the addressing of the second modified LS-100 board will be a little strange; its address will be four (switch 5) because the software thinks that the first 1-Megabyte physical board is really four logical boards (switches 1 through 4).

The now-unused input pins 1 and 2 of Z3 must be connected to ground for the board to decode addresses correctly. Pins 9 and 15 of Z14 will be connected to the new multiplexer and act as address bits A18 and A19, respectively.

Alternatively, there are ways to use as many as eight boards in a system (such as by connecting pins 5 and 12 of Z14 to pins 1 and 2 of Z3), but all of those options require software changes that will not be covered here.

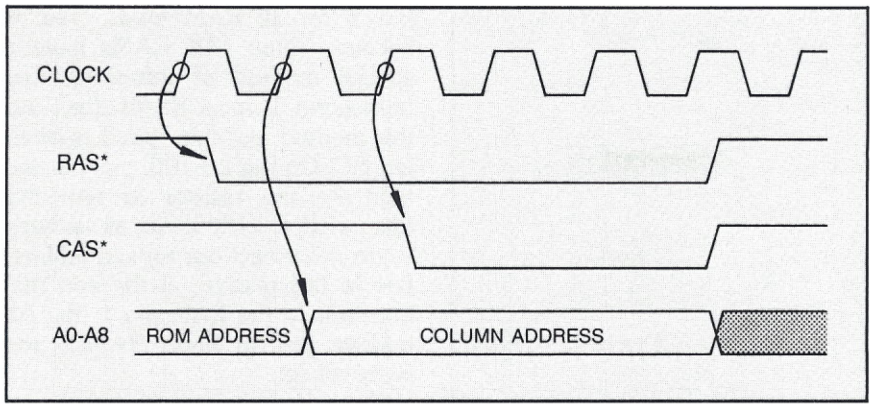


Figure 1. *RAS* and CAS* timing for the 8203.*

TIMING AND LOADING ANALYSIS

Let's analyze the design to see how well it meets timing and loading specifications. An understanding of this section is not required to successfully complete the RAM-disk modification if you follow the method given in this article, but the information is presented here for completeness.

The left side of Table 1 (page 72) shows part of the timing analysis for the modification. Refer to Figure 3 for a complete and final schematic of the modification and as a guide to Table 1. Unfortunately, the 8203 data sheets do not provide complete timing information. Table 1 indicates the 8203 delays necessary to meet timing requirements (and how likely these are). Notice that two problems show up from this analysis: 1) a 74LS74 flip-flop will not get the required setup time, and 2) the setup time for the CAS* signal is hard to count on. In order to resolve these issues, we need to speed up the circuit paths. The easiest way is to replace the 74LS74 flip-flop with a 74S74 (Schottky) flip-flop. This device only requires about 4 ns of setup time and propagation delay of about 10 ns. The right side of Table 1 shows the same timing analysis using the new flip-flop. This change adds an additional 15 ns to the flip-flop setup time and gives us well over 8 ns of CAS* setup time.

This timing analysis shows how a thorough understanding and some simple mathematics can avoid difficult-to-find errors when troubleshooting. Imagine trying to find out why your RAM-disk keeps losing data because of a setup time violation.

Tables 2 and 3 show a complete loading analysis. The notation used in Table 2 is Zx/y where Zx is the component designator (like Z7, the 8203) and y is the pin number. To determine the Total Load Current, add up the I_{IH}s and I_{IL}s for the parts in the Loading column. These are indicated in Table 3. The Capability column is simply the I_{OH} and I_{OL} of the part on the Pin column. Table 2 shows that there are no loading problems for any of the circuits affected by the modification.

The only other loading analysis concern is the power consumption. We are adding circuits and using RAM chips that consume extra power. The power for the new parts plus the additional power required by the 256K RAM chips draws an extra 200 mA of current from the voltage regulators. The unmodified LS-100 board consumes about 600 mA of current. Thus the total current for the modified board is less than 1 ampere. The two voltage regulators on the LS-100 board can supply about 2 amperes of current nominally. Clearly, there are no power loading issues to be concerned about.

IMPLEMENTING THE DESIGN

Figure 3 shows the complete modification with all of the pertinent pin numbers. First of all, add the oscillator package. This should be done all by itself, so you can be sure that the 8203 is getting the proper clock without adding any other variables. Add the oscillator package by removing XY1 (the crystal), R7, R8, and C5 and C7 if they're installed. Mount the package upside down, over the position of the old crystal, with pin 1 toward the top of the

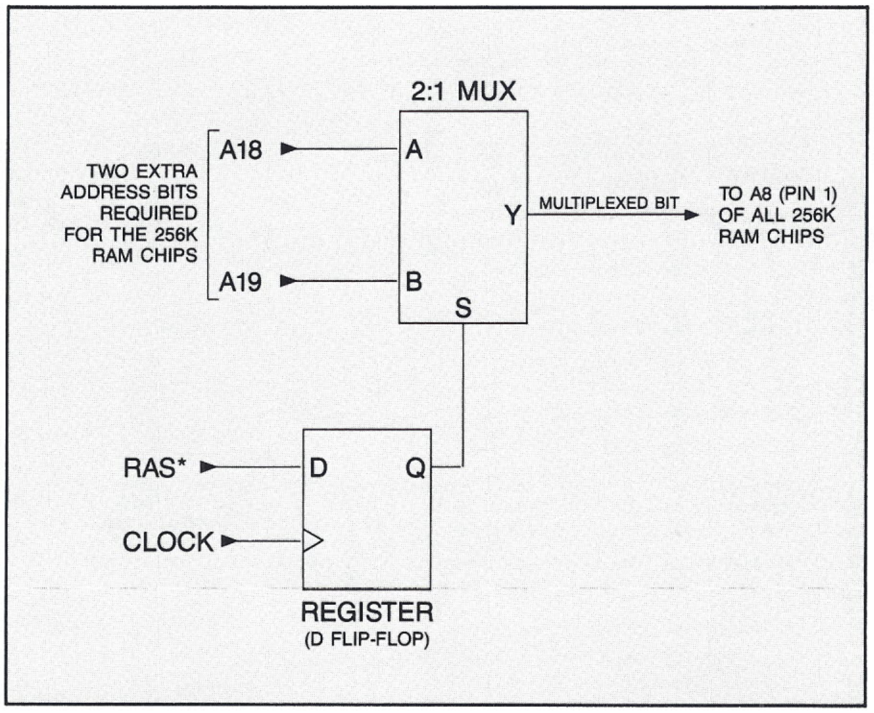

2:1 MUX
TWO EXTRA ADDRESS BITS REQUIRED FOR THE 256K RAM CHIPS
A18
A19
A
B
Y
S
MULTIPLEXED BIT
TO A8 (PIN 1) OF ALL 256K RAM CHIPS
RAS*
CLOCK
D
Q
REGISTER
(D FLIP-FLOP)

Figure 2. *Simplified diagram of the proposed modification. This circuit generates the additional (multiplexed) address bit required by the 256K chip.*

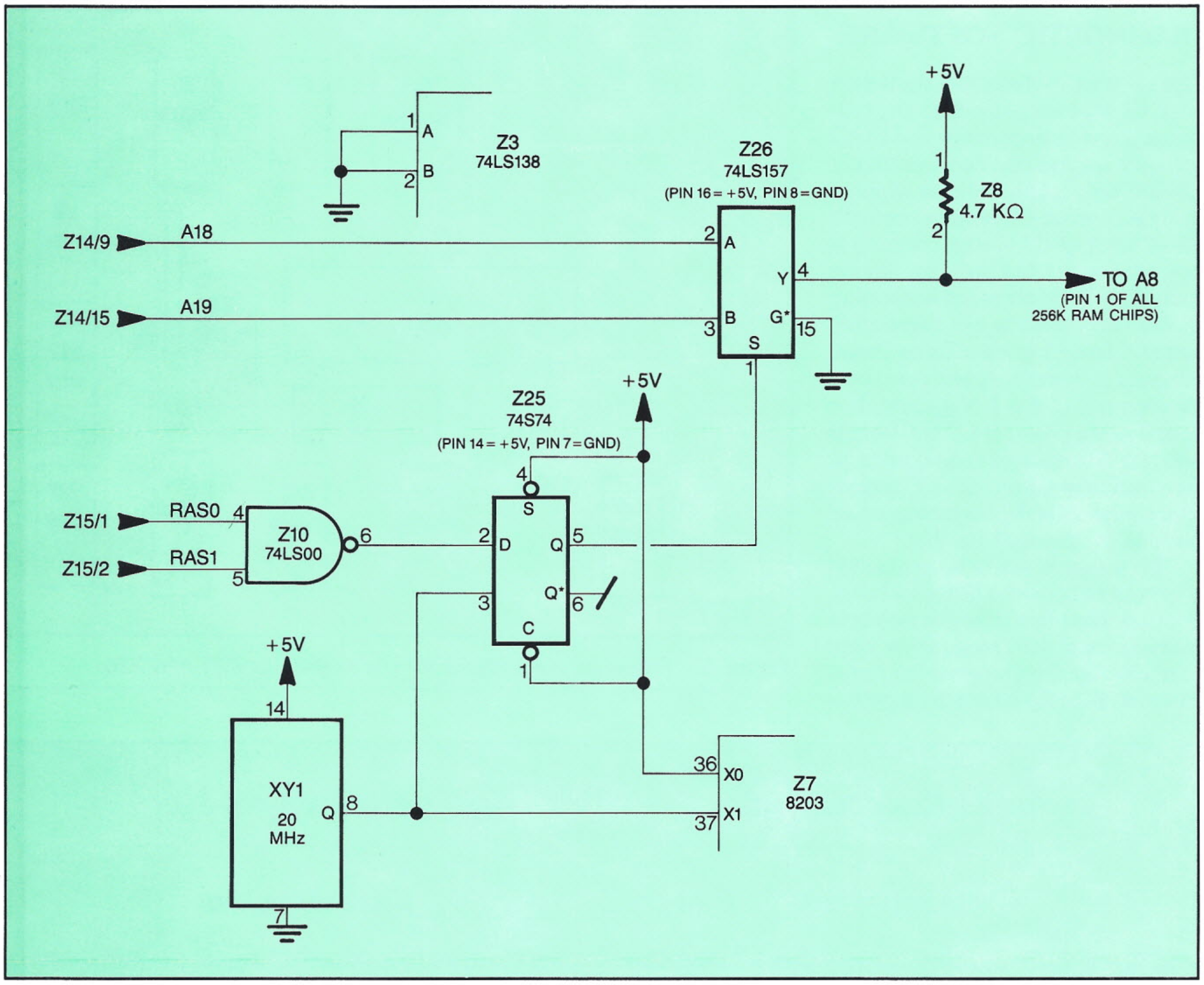


Figure 3. *Detailed schematic of the RAM-disk modification. All pin numbers are indicated here.*

board. Now, wire pin 36 of the 8203 (Z7) to +5V (you can do this on the back of the board). Wire the oscillator to power and ground; pin 14 to +5V and pin 7 to ground. Finally, wire pin 8 of the oscillator to pin 37 of the 8203. These last three connections should be on the component side of the board.

Retest the board, using DIAG, FMAT, and INSTALL, to be sure it operates properly. It should either work fine or not at all. If it doesn't work, you know that something is wrong with the oscillator connections.

Once the oscillator modification works, it's time to add the other parts. The easiest way to add the IC packages is to glue them to the board in a 'dead-bug' position. That is, upside down. Figure 4 shows a suggested parts placement. Use silicon adhesive, epoxy, or even the fast contact glues. The silicon adhesive is nice because it is strong and sets pretty quickly but can be removed if necessary. Cut all of the pins shorter so that they don't stick up like little needles; they are too easily bent and could short together. Cut them to just above the body of the IC (see Figure 5).

I recommend removing Z10, the NAND gate, bending out pins 4, 5, and 6, and cutting them short as well. This avoids cutting traces on the PCB. The same goes for Z14 (the MSB address register), pins 9 and 15. The only connections to be made where neither the pins can be lifted nor traces cut (i.e., the pins must remain in the socket), are to the RAS0 and RAS1 signals. As indicated in Figure 3, solder the wires directly to Z15. Do this by removing Z15 from its socket, carefully solder the wires to pins 1 and 2 up near the body of the IC, and reinsert Z15 into its socket.

Be sure to ground pins 1 and 2 of Z3. You can do this on the back side of the board. Be careful to route all the wires close to the board and with as short a path as is practical. Especially, the wire from the 20-MHz clock and the wire from Z25 to Z26 should be short. Use 30 AWG wire-wrap wire for all the connections.

After completing the connections, replace all the original RAM chips with your new 256K RAMs. Now you're ready to test your modification. Try using DIAG (for boards one through four), FMAT, and INSTALL. Then copy some text files to the RAM-disk. Make them large by concatenating several together into one. Copy to several different file names and try to fill up the disk. Then verify that none of the data is corrupt.

DIAGNOSTIC SOFTWARE

The software routines that started on page 63 will help you debug any problems you may encounter. The DIAG software routine that comes with the LS-100 will roughly indicate whether the board works. However, since each 256K bank of RAM-disk memory is not contiguous but split up among four 256K RAM chips, bit addressing of the two MSBs is not thoroughly tested. Listing 1 gives a Turbo Pascal program to completely test every byte location out of the 1 Megabyte. It is specifically written for LS-100 boards with the 1-Megabyte modification. It won't work for normal LS-100 boards. The constants at the beginning would have to be changed for that.

This program works by generating a unique pattern of 1,048,576 bytes. It writes these to successive disk byte locations and then reads them back to check that the pattern is the same. Since an 8- or 16-bit machine cannot

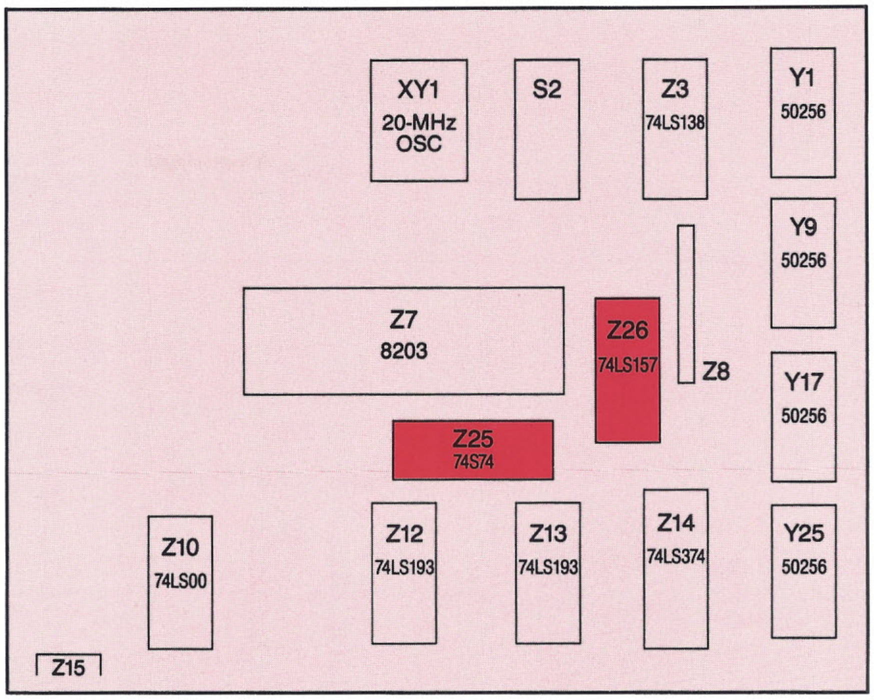


Figure 4. *Proposed parts layout for the new chips Z25 and Z26.*

USING 74LS74 FLIP-FLOP	**USING 74S74 FLIP-FLOP**
Z25 SETUP TIME: $Z25t_{SU}$ = 50ns – Z7/RAS0,RAS1t_{PD}(max) + Z10t_{PD}(max) $Z25t_{SU}$(required) = 22 ns Z10t_{PD} = 16 ns Therefore, Z7/RAS0,RAS1t_{PD} must be less than 12 ns **This is highly unlikely.**	**Z25 SETUP TIME:** $Z25t_{SU}$ = 50ns – Z7/RAS0,RAS1t_{PD}(max) + Z10t_{PD}(max) $Z25t_{SU}$(required) = 3 ns Z10t_{PD} = 16 ns Therefore, Z7/RAS0,RAS1t_{PD} must be less than 31 ns **This delay should give adequate margin.**
CAS* SETUP TIME: CASt_{SU} = 50 ns – Z25t_{PD}(max) + Z26t_{PD}(max) – Z7/CAS*t_{PD}(min) CASt_{SU}(required) = 0 ns Z25t_{PD} = 28 ns Z26t_{PD} = 30 ns Therefore, Z7/CAS*t_{PD} must be greater than 8 ns **Although likely, somewhat too long to count on.**	**CAS* SETUP TIME:** CASt_{SU} = 50 ns – Z25t_{PD}(max) + Z26t_{PD}(max) –Z7/CAS*t_{PD}(min) CASt_{SU}(required) = 0 ns Z25t_{PD} = 10 ns Z26t_{PD} = 30 ns Therefore, CAS* has a minimum of 10 ns of setup time.

Table 1. *Worst-Case Timing Analysis. A 10% additional derating is included in all times.*

easily produce a 20-bit pseudo-random number in a high-order language, the program simulates one via a Boolean array. The lower eight bits of this array are converted to a binary number and serve as the byte value. Unfortunately, this program is **very** slow (it takes hours) but you should only have to run it once. For those of you who prefer faster testing, Listing 2 gives an assembly language version of the same program. This program uses Z80 mnemonics, so you may have to translate to your own processor's assembly language.

COST ESTIMATE

What is the cost of the described modification? The RAM chips are the most expensive parts by far. At the time of this writing, I.C. Express was selling 150ns (Hitachi) RAM chips for $2.75 each. This is the lowest price I've found. Don't buy RAMs faster than 150ns since the extra speed won't be seen by the 8203. Thirty-two of these RAMs will cost under $100. Try not to get Mostek chips since these appear to have well over 50ns of hold time requirements for RAS*, and they might not work! Beside the RAMs, the two discrete chips (the register and multiplexer) cost less than a dollar each. The crystal oscillator package costs less than $5.

Therefore, for a total of roughly $100, you can quadruple the 256K RAM-disk capacity of the LS-100. Of course, this doesn't include your time, but only you can put a dollar value on that! ■

LS-100 KIT:	Digital Research Computers PO Box 381450 Duncanville, TX 75138 (214) 225-2309
256K RAMS:	IC Express 15358 Valley Blvd. City of Industry, CA 91746 (800) 882-8181 (CA) (800) 892-8889 (outside CA)
	Microprocessors Unltd, Inc. 24000 S. Peoria Avenue Beggs, OK 74421 (918) 267-4961
ALL PARTS:	JDR Microdevices 110 Knowles Drive Los Gatos, CA 95030 (800) 538-5000

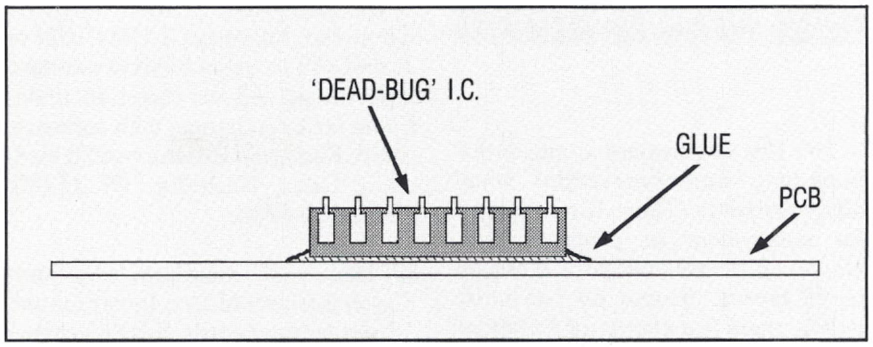


Figure 5. *IC package in a 'dead-bug' position.*

PIN ($I_{OH,L}$)	LOADING ($I_{IH,L}$)	TOTAL (mA)	CAPABILITY (mA)
Z7/21 High	Z15/2 + Z4/13 + Z4/10 + Z10/5	0.17	1
Z7/21 Low		6.4	10
Z7/22 High	Z15/1 + Z4/1 + Z4/4 + Z10/4	0.17	1
Z7/22 Low		6.4	10
Z10/6 High	Z25/2	0.05	0.4
Z10/6 Low		2	8
Z14/9 High	Z26/2	0.02	2.6
Z14/9 Low		0.4	24
Z14/15 High	Z26/3	0.02	2.6
Z14/15 Low		0.4	24
Z25/5 High	Z26/1	0.04	1
Z25/5 Low		0.8	20
Z26/4 High	(-Z8/2) + (Y1 thru Y31)/1	0.32	0.9
Z26/4 Low	(NOTE: Z8 sources current with Z26)	0.32	8
XY1/8 High	Z7/37 + Z25/3	0.14	0.4
XY1/8 Low		6	16

Table 2. *Loading Analysis. Problems only exist if the capability is less than the total load. Thus, the circuits affected by the modification show no problems.*

	Z4,Z15 74S32	Z7 8203	Z8 4.7K	Z10 74LS00	Z14 74LS374	Z25 74S74	Z26 74LS157	Yx 50250	XY1 20MHz
I_{OH}	-1	-1	-.5	-.4	-2.6	-1	-.4	N/A	-.4
I_{OL}	20	10	30	8	24	20	8	N/A	16

I_{IH}	.05	.04	N/A	.02	.02	.05(2)
						.1(3)
							.04(1)
							.02
								.01	N/A

I_{IL}	-2	-2	N/A	-.4	-.4	-2(2) -4(3)	-.8(1) -.4	-.01	N/A

Table 3. *Component Currents (in mA). Pin numbers are shown in parenthesis.*

bits

The Bits department is mostly for publishing ***non-commercial*** *small advertisements. There is* ***no charge*** *for subscribers to place an ad. However, please limit your message to* ***50 words.*** *If your ad has more words, there is a charge of $1.00 for each word over 50.*

Commercial *ads are also acceptable for this section at the rate of $20 for up to 50 words plus $1 for each word over 50.*

You may take advantage of this service to sell or buy used S-100 hardware, trade personal programs, etc. Send ads and prepayment (if applicable) to S-100 Journal, BITS, PO Box 1914, Orem, UT 84057.

WANTED

Need documentation for SD Systems VDB-8024 Video Board, Jade MPC-4 Quad SIO, and Measurement Systems and Controls (Systems Group) CPC-2810 Quad SIO with Dual PIO. Trying to repair and refit an Action Computer Enterprises DPC-280 but need Rev. N/C schematic for the DPC-100. Trade or Sell? Tim (714) 679-9217.

Wanted: Information and/or schematics on S-100 SCSI(?) board Type 1A manufactured by Sci Corporation circa 1983. B. Springer, 5219 65th Ave West, Tacoma, WA 98467.

FOR SALE

USED S-100 BOARDS: Teletek System Master SBC, $300. Teletek SBC-1, $250. Teletek FDC-1 SBC, $250. Teletek 64K RAM, $200. Seals 8K Static, $75. Seals 32K Static, $100. OTHER USED ITEMS: Ampex D80/D81 smart terminal, $250. Lear Seigler ADM-3 terminal, $125. Alloy Eng DZ-80B tape interface $100. Claude Hill (615) 331-4743 evenings.

SPECIAL!!! New 9 Track Magtape Controller. Interfaces S-100 CP/M or TurboDOS to ½-inch ANSI standard tape drives. Allows direct to mainframe file interchange, with software: $950. Redwood Research, 820 Redwood Drive, Nashville, TN 37220. (615) 331-4743.

(2) New S-100 Mainframes by Internation Instrumentation Incorporated. 12-slot motherboards. S-100 Switcher power supply, with output for floppy or hard drives. Units come with desktop enclosures and rack-mount front panels with drive cutouts. Schematic documentation included. Units have never been used. $500.00 for both. Contact: Ira Goodberg (213) 650-6327.

S-100 Boards, 8K Memory $18 each, 64K Static RAM $50, 8080 CPU $14, Video Boards $15, Disk Controller $15, I/O $10, Motherboard $10. All in E.C. with full documentation. Prices do not include shipping. G. Seweryniak, 12 Kathy Dr, Poguoson, VA 23662.

Selling one of my S-100 systems: CompuPro Z-80, Disk 1, Interfacer 4, RAM 16 64K, Televideo 910+ terminal, two Mitsubishi ½ ht 8-inch drives (#2894-63), CCS 220 mainframe (12 slots), WordStar, Turbo Pascal + others. Cost $3000; asking $1595. David DuPuy (703)463-6793.

CompuPro Interfacer 3: 8-port serial card with vectored interrupts & modem control. BRAND NEW: with all cables, documentation, original warranty card. Originally $600. Asking $300 — must sell — best offer. Call or Write: Michael C. Greespon, 2124 Kittredge #117, Berkeley, CA 94704. (415) 849-2182.

S-100 Computer: VECTOR-1 mainframe (18 slots, 18 amps), TELERAY 100 smart ANSI terminal (amber w/smooth scrolling), 2 SIEMENS DD drives, 64K SRAM, 2/4/6-MHz Z-80, 504K RAMdisk, 9 parallel — 3 serial ports, sound/real-time clock. 8 boards total. Full docs/schematics, software and more. $400 negotiable. (919) 596-9101 evenings.

CompuPro System 816/E-CSC, includes the CPU68K 10MHz, Disk 1 Controller, System Support 1, Interfacer 4, 2-RAM21, 2-RAM22, 2-M-Drive/H 512K, System Desk Enclosure, Floppy Subsystem, Z29 Display Terminal, Datasouth 180 Printer, documentation, and CP/M-68K. $2500. Douglas Smith (408) 758-0292.

UCI S-100 RAM board populated with 256Kb which can be expanded to 2Mb. With manual. $350 or best offer. Call (408) 429-1036.

TELETEK Z80A TurboDOS 4-user (SBC-1 128K), custom 12-slot frame, Xebec S-1410A, Miniscribe 4020, NEC FD-1165, $1700. ICM TurboDOS 4-user (CPS-Q6A) 10 serial ports, OMTI 5300, RO252 10MB, NEC FD-1165, Teac FD55F & Fujitsu M2551A, $4000. 2 Okidata ML83A, $200. ea. 4 ADDS VP1A, $100. ea. 2 WY-50 green, $200. ea. 3 TELETEK SBC-1/128, $300. ea. Paul Callaway, P.O. Box 2123, Winter Park, FL 32790. (305) 260-8004.

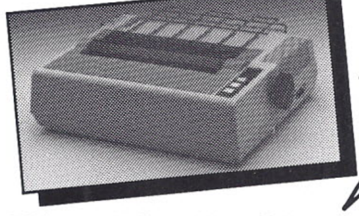
MS-DOS S-100 System: Lomas Thunder 186 & Color Magic, 2 disk drives, Integrand 3315 Mainframe, with CCDOS 4.1 and comm pgm. Will boot MS-DOS. $1,000 + shipping. With WICO trackball keyboard $1,150. CompuPro Interfacer 4 w/ cables $200. Gary Van Cott, PO Box 1879, Grafton, VA 23692. (804) 898-3680.

Seattle Computer Products 8086 CPU/128K memory. 1 serial port. 2 parallel ports. Tarbell MD2022 controller. Dual Morrow 8″ disk drives. Single and dual cabinets for drives. TEI 22-slot microframe. Soroc terminal. Over $4000 inested —make offer. W. L. Overton (704) 376-2512.

Rugged 10-slot commercial chassis with Z-80, dual 8-inch floppies, complete system. Extensive word processing, accounting, and commercial software. Must go — best offer received by 12/31/87. Send SASE for details. Winer Mobile Observatory, 10912 Broad Green Terrace, Potomac, Maryland 20854. (301) 983-9442 evenings.

THE BEST BUYS IN COMPUTER EQUIPMENT AND SUPPLIES

PRINTERS



LOW PRICE $169 NEW

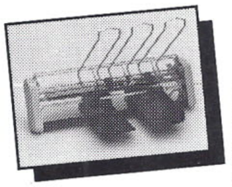
CSS-5152 Graphics Printers

- ■ 100% IBM 5152 Compatible
- ■ IBM Parallel Interface, 80 cps
- ■ IBM PC Graphics Compatible
- ■ Easy Installation – Just Plug In
- ■ Order Cable No. 5132 for IBM – $10.00
- ■ **Discount on 2 or more**

SPECIAL $199

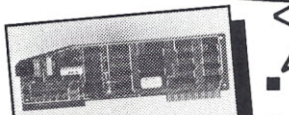
TeleVideo 970 Terminals

- ■ The Terminal Bargain of the Year
- ■ VT100, VT52, ANSI 3.64 Compatible
- ■ In Original Factory Packaging
- ■ Manual Included



PRICED TO MOVE $49.95

Qume Sprit 11 Series Single-Bin Sheet Feeders

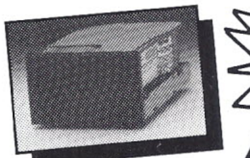


LIMITED OFFER $19.95

- ■ Apple II, II+, IIE Compatible
- ■ Hayes Compatible
- ■ Manual Included

APPLE INTERNAL MODEM

HARD DISK DRIVES & CONTROLLERS



GREAT $1495 PRICE

MAXTOR XT2140 140 Megabyte

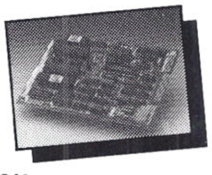
■ 11 Heads ■ 1224 Cylinders ■ 38 ms Avg. Access
"We carry a large assortment of other disk drives."

Adaptec Hard Disk Controllers

PC-XT Controller ST506/412	$85
2070 PC/XT RLL	$109
3530 **SCSI** to Tape QIC 36	$99
4000 **SCSI** to ST506/412	$109
4000A **SCSI** to ST506/412	$129
4070 **SCSI** to ST506/412 RLL	$129
4520 **SCSI** to ESDI	$139
5500 **SCSI** to ST506/412	$255
5580 **SCSI** to SMD	$450
Manuals	$8 each

Cables Available — Ask for Pricing

Other Hard Disk Controllers



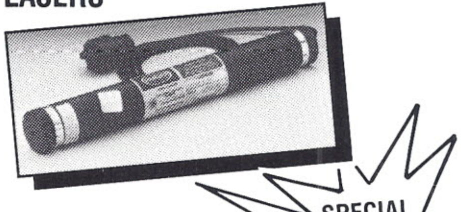
ONLY $79

Shugart 1610-1/3/4

- ■ 3 Firmware Sets (DTC-XEBEC-SCSI)
- ■ 5¼″ Form Factor
- ■ Controls 2 Standard 5¼″ Hard Drives
- ■ SASI/SCSI
- ■ Manual – $8 extra

OMTI 20L/C SASI Controller	$99
XEBEC S1410 SASI Controller	$109
WD 1002-SHD (Xebec Clone) SASI Controller	$109
DTC 510A SASI Controller	$120
DTC 700 (Shugart 1408-I) Controller	$150
DTC 1404/1404D Controller	$125

LASERS



SPECIAL $49.95

Used HeNe Laser Tubes

- ■ ≈ 2–5mW (Off Spec.)
- ■ 1800–2000 Vdc
- ■ Glass Tube
- ■ Sample Power Supply Schematic Included

Power Supply .... Only $49.95

- ■ Modular Metal Design

★★ SPECIALS ★★

110 Vac Motors (Record Player Type)	10 for $7
Sleeper Motors 9.2Vdc 0.24A/ϕ 38Ω 400S/R	$6.95
Alignment Diskette Shugart SA145 (96 tpi Single Sided)	$7
Diablo 630 Bi-directional Tractors	$75
CSS 3522 12 Mb Hard Disk 3½ In. ½ Height	$149
Corvus Omninet Transporter Card (IBM-PC)	$89

"It's Like a Swap Meet Every Day"

CALL FOR SHIPPING CHARGES

We have a large invventory of New & Used Peripherals, (Terminals, Monitors, Printers, etc.), and PCB's for many systems, (S100, Multibus, STD Bus, Q Bus, etc.). If you have a requirement give us a call, we may have it.

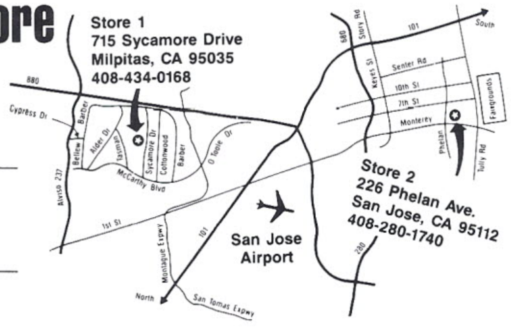
Computer Surplus Store

Two Locations

715 Sycamore Drive,
Milpitas, CA 95035
408-434-0168
FAX 408-434-0931

226 Phelan Avenue
San Jose, CA 95112
408-280-1740
FAX 408-280-0518

Phone Orders
408-434-1060



MAIL ORDER INSTRUCTIONS

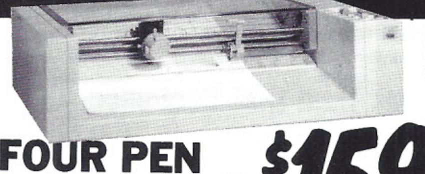
408-434-1060

Mail orders accepted on most items. Payments can be made via UPS COd, Master Card, Visa, or Prepay (Check or M.O. U.S. Dollars Only. No Cash). To eliminate back-orders please call prior to ordering so that we can reserve your items. Calif. residents add 7% sales tax. Minimum credit card or C.O.D. order $15.

***Quantities subject to stock on hand. Minimum Order $5.00.**

California Digital

17700 Figueroa Street ● Carson, California 90248



FOUR PEN COLOR PLOTTER $159

The manufacturer has asked us not to publish their name. But this four color plotter was produced by one of the Worlds largest makers of computers.
The 410 color plotter will connect to the **serial port** of virtually any microcomputer. Simple ASCII commands direct one of the four color pens to draw circles, arcs or ellipses on paper or transparency material up to 11 by 17 inches. Enlargements or reductions are achieved through elaborate firmware. Pen travel is four inches per second with .004" pen resolution. Standard pens are available in an assortment of 32 different colors and widths.
The ideal plotter for architecture, CAD engineering or graphic design. **At $595 it was a great buy, at $159 its a steal.** Support kits for specific computers.

OTHER PLOTTERS AVAILABLE:
Hewlett Packard ● Houston Instr. ● Roland ● Sweet P ● CalComp

Free Interface Kit

~~$2495~~ $759

80 Character Daisy Wheel Printer

These Fujitsu Daisy Max 830 were manufactured for Motorola's Computer Division. The purchase order was canceled and Fujitsu was forced to liquidate these 80 character per second daisy wheel printers at "fire sale" prices.
Features: bullet proof construction, serial RS-232 interface, Diablo 630 wheels and commands, programmable line spacing in increments of 1/96" and column spacing of 1/120". The printer is also capable of underscoring, bold overprint, shadow print, centers and justifies along with vector plotting.
Factory suggested price of the Daisy Max 830 was $2495, California Digital is offering this liquidated special at only $759 and including FREE a serial card kit with cable. Also available: tractor and sheet feeders.

COLOR GRAPHICS TERMINAL

~~$3995~~

$659

The Lear Siegler 7107 color graphics terminal is both DEC (VT-100 / VT-51) and Tektronix (4010/4014) compatible.
The terminal offers a palette of 4,096 colors with a resolution of 640 by 480 pixels on a 13 inch non-glare black screen.
Horizontal pan and image zoom, to 16 times original size, are standard features of this color graphics terminal.
The 7107 is the ideal terminal for CAD/CAM applications.

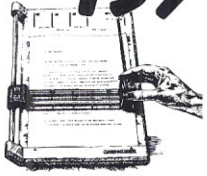
PRINTERS

IBM Thermal printer, 80 col., serial	$39	Panasonic P1592 15" 180 cps.	349
Fujitsu D830 80 cps., daisy wheel serial	759	Star Gemini NX10, 120 cps 10" NLQ.	239
Silver Reed EXP-800 136 col., 40 cps.	729	Star Gemini NX15, 120 cps 15" NLQ.	359
Juki 6300 daisy wheel 40 cps.	819	Toshiba 321, 24 wire head 216 cps; NLQ.	519
NEC 8850 Spinwriter daisy wheel 55 cps.	1159	Toshiba 341, 24 wire head 216 cps. 15".	629
NEC P6/660P 10" 216 cps. NLQ.	469	Printronics P300 dot matrix, 300 LPM	3995
NEC P7/760P 15" 216 cps. NLQ.	659	Printronics P600 dot matrix, 600 LPM	5795
Epson FX86E NLQ 10" 240 char/sec.	399	Dataproducts B600/3 band ptr, 600 LPM	6985
Epson FX286E NLQ 15" 240 char/sec.	539	LASER PRINTERS	
Epson EX1000 300 cps, NLQ/54 cps.	599	Apple Laser Writer Plus; Postscript	4295
Citzen MSP-10 160 cps. 10"	299	QMS PS/800 2 meg. Postscript, 8 pgs.	4295
Citzen MSP-15 160 cps. 15"	449	Hewlett Packard Series II, laser printer	1795
Okidata 182 80 column, par'l.	239	Texas Instr. OmniLaser 2108 P/S, 8 pgs.	4295
Okidata 192+ 136 column, par'l.	345	Texas Inst. OmniLaser 2115 P/S, 15 pgs	5595
Okidata292 136 column, parellel.	489	Quadram Quad/Laser, 5 Megabytes	2995
Panasonic P1080/I, 10" 120 cps.	229	AST Turbolaser, 8 pages/minute.	3195
Panasonic P1091/I, 10" 160 cps.	299		

NOW YOUR COMPUTER CAN READ!!

Omni-Reader... the first optical character reader **designed and priced for the small computer**

$159



COPY:
·Manuscripts ·Contracts ·Articles
·Forms ·Invoices

APPLICATIONS:
·Mailing Lists ·Editing ·Data Base Management ·Transfering information between incompatible systems

Uses a standard RS-232 serial port hookup to interface eaisily with your computer.

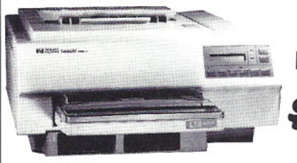
20/20 Bernoulli Box

~~$3495~~

$1595

California Digital has purchased these 20/20 Bernoulli systems from Iomega. The units needed some minor alignment and had to be sent back to Iomega. These have to sold as as reconditioned... but for all practical purpose they are new and come with a **one year Iomega factory warranty.**
The 20/20 Bernoulli Box features removable cartridges, and delivers reliability, expandability, transportability, security and speed in one versatile subsystem. It lets you transfer megabytes of information safely and swiftly for primary or backup storage. Or combine several software programs onto a single cartridge for easy switching from one to another.
Reliable... the Box has incredible resistance to shock and vibration completely eliminating the possility of head crash.
security essential? Don't lock up your system... just lock up the cartridges.

20+20 Subsystem... **$1595;** Non bootable controller... **$159;**
Bootable controller... **$189;** 20 Megabyte Cartridges... **$69.**



HEWLETT PACKARD

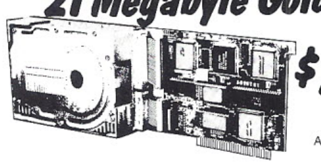
LASER JET II

$1795

Standard features of the new Hewlett Packard Series II Laser printer: 512 K byte memory expandable to 4 Megabytes, both serial and parallel ports. List price $2495 California Digital price $1795. Memory upgrade, type fonts and toner cartridges available.

Other Laser Printers available:
Texas Instruments ● Okidata ● Apple ● Ricoh
Qume ● AST Research ● QMS ● Xerox ● Canon
Quadram ● Centronics ● CIE C.Itoh ● DataProducts

21 Megabyte Gold Card



$459

Also available 31 Megabytes at only $559.

The fastest, lowest powered, longest warrantied, most durable, highest capacity, most reliable, lowest priced HARD-DISK-ON-A CARD available in the world today.

• 65ms Access Time - *Fastest Available* • Automatic Head Unloading - *Protects Heads and Media* • 2K Sector Buffer - *Increases System Throughput* • High Reliability: 28,000 hr. MTBF *No one else even close* • 15.5 Watt Power Consumption - *Lowest Available* • 2 Year Warranty *Longest Available*

$35

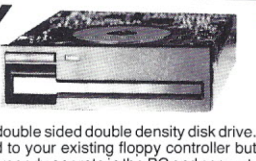
We have them again... The Remex 480 was a sell out two years ago but we located an additional 10,000 units and are offering these 2/3 height IBM/PC compatible 360K/Byte drives at only $35 each at quantity two.

115

The Panasonic 475 is jumper selectable between 96 TPI format and 1.2 Megabyte as used in the IBM/AT computer.
Suggested price of the 475 is $189 but California Digital located a computer manufacturer that had to liquidate stock making available 1000 of these choice drives. Quantity two $115 each.

	One	Two	Ten
REMEX 480 ⅔ height for PC	39	35	call
MITSUBISHI 4851 360K/Byte	89	85	82
MITSUBISHI 4853 96 TPI.	99	95	92
MITSUBISHI new 501 half ht.	129	119	109
MITSUBISHI 504A AT comp.	169	159	155
TEAC FD55BV half height	109	99	89
TEAC FD55FV 96 TPI, half ht.	119	109	105
TEAC FD55GF for IBM AT	169	159	155
PANASONIC 455 Half Height	109	99	89
PANASONIC 475 1.2 Meg./96	119	115	109
Switching power supply			49
Dual enclosure for 5¼" drives			59

SONY $139



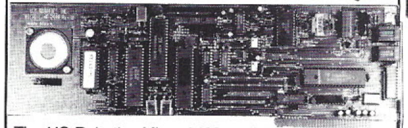
The Sony 53W is a 3½ inch double sided double density disk drive. The drive can be connected to your existing floppy controller but will require MS/DOS 3.2 to properly operate in the PC and compatibles. The 53W is simiiar to the drive being used in the PC portable.

Smarteam 2400 Baud

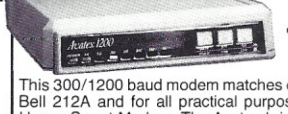
$289

The Smarteam 2400 offers all the features of the Hayes Smart Modem 2400 for a fraction of the price. Now is your opportunity to purchase a 2400 baud modem for only $289.
Also available: The Smarteam 1200 at only $139

U.S. Robotics 2400 Internal $189



The US Robotics Micro 2400 modem is one hundred percent Hayes compatible, auto dial, auto answer, auto everthing. A super value at only $189.



$99

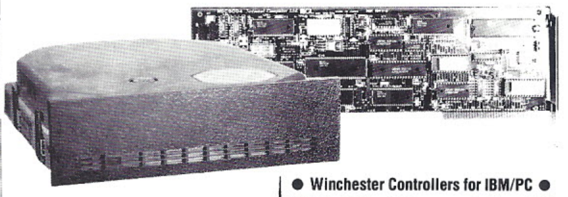
This 300/1200 baud modem matches design specs of Bell 212A and for all practical purpose those of the Hayes Smart Modem. The Avatex brings you reliable data transmission for only $99.

OTHER MODEMS AVAILABLE:
Hayes • Universal Data • Fujitsu
Prometheus • Anchor • US Robotics

Seagate

$459

30 MEGABYTE WINCHESTER HARD DISK KIT



Five Inch Winchester Disk Drives

	each	two+
SEAGATE 225 20 Meg. ½ Ht.	319	299
SEAGATE 238 30 Meg. RLL	369	359
SEAGATE 4026 26 M. 35mS.	659	629
SEAGATE 4051 51 M. 35mS.	759	739
FUJITSU 2242 55 M. 35mS.	1399	1329
FUJITSU 2243 86 M. 35mS.	1895	1819
RODIME RO-202E 27 Meg.	659	629
RODIME RO-203E 40 Meg.	695	659
RODIME RO-204E 53 Meg.	995	959
CONTROL DATA 94155-86 M	1829	1779
MAXTOR XT1140 140 Meg.	2595	2529
TOSHIBA MK56 70 M. 30mS.	1789	1729
TANDON 502 10 Meg.	419	379

● Winchester Controllers for IBM/PC ●	
XEBEC 1220 with floppy controller	229
DTC 5150CX	139
OMTI 5520 half card	129
OMTI 5527 RLL controller	159
ADAPTEC 2070 RLL controller	179
ADAPTEC 2010A	159
WESTERN DIGITAL WD/1002WX2	119
● SCSI/SASI Winchester Controllers ●	
XEBEC 1410A 5¼" foot print	219
WESTERN DIGITAL 1002-05E 5¼"	289
OMTI 20L	119
● Winchester Accessories ●	
Installation Kit with manual	10
Winchester enclosure and supply	139
Dual 20/34 cable set	25
Switching power supply	49

VISA

master charge
THE INTERBANK CARD

Shipping: First five pounds $3.00, each additional pound $.50.
Foreign orders: 10% shipping, excess will be refunded.
California residents add 6½% sales tax. ● COD's discouraged.
Open accounts extended to state supported educational institutions and companies with a strong "Dun & Bradstreet" rating.

TeleFax
(213) 217-1951

TOLL FREE ORDER LINE
(800) 421-5041
TECHNICAL & CALIFORNIA
(213) 217-0500

DATAPLOTTER PRINTER GRAPHICS

David L. DuPuy

Graphics are an important aspect of microcomputing these days, and there are many graphics packages available. For microcomputers with memory-mapped video, graphics software must be tailored to match the hardware characteristics of a particular computer. But there are many S-100 microcomputer systems in use with serial I/O terminals and no screen graphics at all, and any graphics on these systems must use a dot-matrix printer or some other graphics device instead of the screen.

DataPlotter, from Lark Software, fulfills this need and has particular appeal for a printer-graphics package: it is inexpensive, easy to use, requires no installation, has an excellent manual, and is versatile and reliable. DataPlotter is available from Lark Software (address at the end) for $69 for the line and scatter graph software, or $69 for the pie chart and bar graph software, or $99 for both packages (plus $3 shipping charge). DataPlotter is available for CP/M-80, CP/M-86, and MS-DOS. This article describes my experiences in using the CP/M version of DataPlotter for the past year or so, with details on plotting procedures, limitations, and results.

FEATURES AND GRAPHING PROCEDURES

DataPlotter arrived neatly packaged with a good quality diskette. The software is unprotected. No installation was required since the software comes configured for a specific printer. DataPlotter is available for a wide range of printers, as listed in the order brochure. I have used my MX-80 version with an Epson MX-80/Graftrax, an Epson LX-80, and a Citizen MSP-20 printer, and all three produced nice results. The convenience of no installation hassles is welcome.

The basic procedure for reading a data file and then graphing it is as follows:

```
PREPLOT NAME.DAT NAME.PLT
PLOT NAME.PLT
```

where PREPLOT is the Lark software which prepares the plotting file, NAME.DAT is the data file to be graphed, NAME.PLT is the file produced by PREPLOT, and PLOT is the actual plotting software (from Lark). PREPLOT reads the data file and then asks a number of questions regarding the format of the graph to be plotted. For example, should it be a *scatter* plot or a *line* plot? How large should the final graph be? What titles or legends do you want? What type of symbols do you want and how large should they be? These graphing parameters and several others are obtained from the user interactively in a very easy and efficient manner. The graphing neophyte will suffer no intimidation, and the experienced user will not find the interactive exchange insulting. Furthermore, for repetitive graphs with similar graphing parameters, these answers to PREPLOT can be saved in another file, so you do not have to answer them more than once and successive graphs can be plotted using the same parameters but different data files. This feature is essential for routine graphing, and it has been implemented in a very successful manner by Lark Software.

One limitation of DataPlotter is that decimal numbers cannot be plotted directly. Fortunately, there is a simple means around this problem: a special utility called TRANSF (for TRANSForm) is provided to read the data files with decimals and then transform them to integers, with ►

David DuPuy is a Professor of Astronomy at the Virginia Military Institute. His special interests are in variable stars and electronic instrumentation for Astronomy. David also enjoys hiking, skiing, and golfing.

numbers you supply as multipliers. For example, if you wish to plot a data file with numbers such as 0.023, 0.741, 1.238, ..., you could supply a multiplier of 1000, and the integers actually plotted would be 23, 741, 1238, ... The labels can then be changed to read 0.0 to 1.4, for example, so the final graph *appears* to be plotting decimals. In many cases, the disk data file could simply be stored as integers, bypassing this extra step of using TRANSF. In any case, with the TRANSF utility available, I have not found this 'integer only' limitation to be a nuisance.

The data to be plotted must be in a disk file produced by an editor, or some other program in BASIC, FORTRAN, Pascal, etc. No particular format is required for the data, and this is a very convenient aspect of using DataPlotter. Furthermore, the data may be in columns (e.g., a spreadsheet) or in 'free form' with any number of data points on any of the lines in the data file, and spaces or commas may be used to separate the data points. There are three formats: (a) the x and y values may alternate; (b) the x values may be listed first, followed by the y values; or (c) the y values only may be given, for graphs with uniformly-spaced x values. This means that data files assembled for other purposes, or for formatted output to accompany a text, may be used directly by DataPlotter without additional formatting. Up to 300 numbers can be plotted from one file in the CP/M-80 version, and up to 2000 in the CP/M-86 and MS-DOS versions.

Scaling the data for plotting may be accomplished automatically or manually. The instruction manual recommends using automatic scaling for 'quick look' graphs, then manual scaling for the final copy. That is a good approach since the automatic scaling often leaves a larger than expected margin on the right hand side. I am somewhat finicky about the tick mark values and the overall layout of the graph, and I found I could always get the layout I wanted with manual scaling.

One characteristic of DataPlotter I especially like is the labelling. Compared to other graphics packages I have used, the legends and titles are neat and easy to read. Both upper and lower case are available for labels, a feature sometimes not found on other graphics packages. Labels are positioned by specifying (in inches) the distance above the x-axis, and either the distance from the y-axis or that the label be centered. For any of the labels, there is a choice of two font sizes: the smaller (default) font is the regular printer text mode, and it yields a nice solid-looking label. A larger-character font is also available for any of the labels. An extremely useful feature of the labelling is that the numbers for each tick mark may be replaced with any numbers or text you wish. That feature is important if decimals are to be plotted, since the graph really plots integers. With a multiplier of 1000 (in TRANSF), the tick marks of 10, 20, ... would be replaced with 0.01, 0.02, ... This feature has other important uses not possible on other graphics packages: the numbers by the tick marks may be replaced by text. For example, the months of the year may be shown on the x-axis by each tick mark (an example is shown in Figure 1 and discussed below). All in all, the labelling capabilities in this software are one of the things that make this an excellent software package for the price.

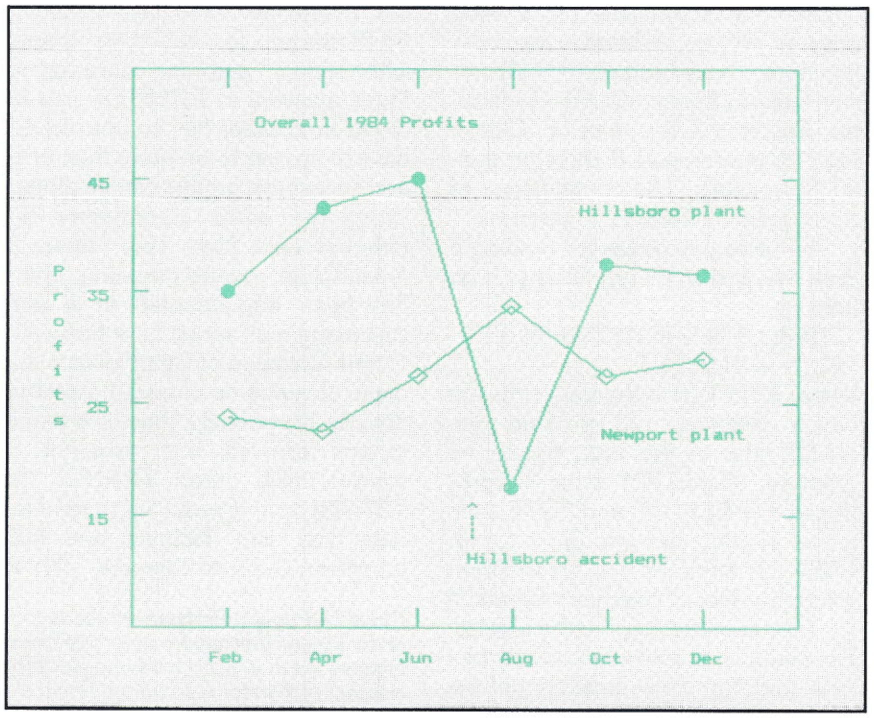


Figure 1. *An example of a line graph, showing two sets of data. The user chooses size and type of symbol. Additional labels may be added anywhere on the graph.*

SAMPLE LINE AND SCATTER GRAPHS

The purpose of this section is to illustrate line and scatter graphs produced by DataPlotter. Note that the orientation of the graphs cannot be changed: the graphs are upside up as they come out of the printer. That is very useful if you want to combine text and graphs on an 8½ × 11 page, but it also means that the x-axis cannot be longer than about 6 to 7 inches (I presume a longer x-axis is possible with a 132-column printer). A simple program could be used to reverse the axes and the vertical symmetry of the data if a longer x-axis were required. For most graphs, the horizontal orientation across the 8½ dimension is adequate.

Figure 1 illustrates several features of DataPlotter. The data file consisted of 12 pairs of x, y coordinates, created with a text editor. The other labels on the graph were added while running the PREPLOT software. One nice feature is that PREPLOT asks the user how many of the remaining n numbers in the data

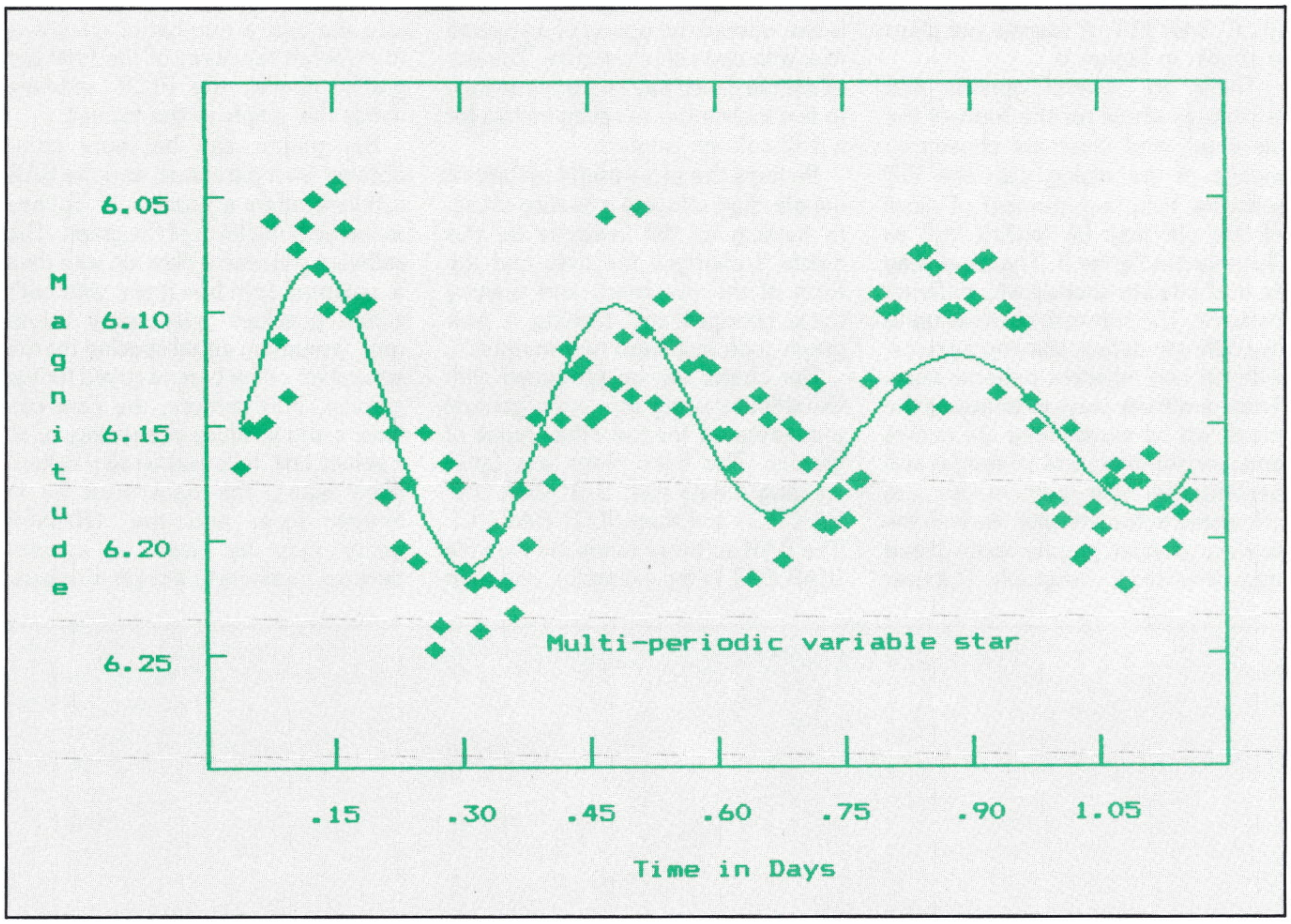


Figure 2. *An example of a combined line graph and a scatter graph. The solid line represents a theoretical fit to the data points. The experimental data points (solid diamonds) and the points making up the line were contained in two data files configured with identical scaling. The two plot files were concatenated before plotting, using the PIP command.*

file go together. In this case, the 12 data pairs were divided into 6 pairs, and then into a final 6 pairs. Hence, any number of separate data groups can be in one data file. The user has control over the choice of symbols (open circles, closed circles, open squares, etc.), as well as the size of the symbols for each data group within the data file. If the PREPLOT answers are saved in an 'answer file' (a .ANS file), changes can subsequently be made after seeing the graph by simply editing that part of the .ANS file, and plotting it again. Note that the numbers on the x-axis (1.0 for Feb, etc.) have been replaced by text (Feb, Apr, etc.). Any keyboard character (such as the arrow by 'Hillsboro accident') can be placed anywhere on the graph. This graph was prepared with 'manual scaling' to produce tick marks exactly where desired.

Figure 2 illustrates the use of DataPlotter with basic x, y data graphed in a scatter plot. The graph also shows a solid line based on a model to fit the data. This type of graph requires a little more effort to put together. First the data points are changed to integers using TRANSF, and then a 'plot file' is produced using PREPLOT. Then the data representing the line is TRANSFormed using the same multipliers, and a simpler, but similar, plot file is produced from these data, using the same PREPLOT parameters. The two resulting .PLT files are merged using PIP, e.g., PIP FINAL.PLT = DATA1.PLT, DATA2.PLT, and the combined graph is produced with PLOT FINAL.PLT. In graphing solid lines, it is easy to exceed the limit of 150 data pairs (for CP/M-80). This can be overcome by either (a) plotting two (or more) data files, and then merging the .PLT files with PIP; or (b) for evenly-spaced data, simply arranging for the data file to contain only the y-values, thereby extending the maximum number of points to 300. As mentioned earlier, the limit for CP/M-86 and MS-DOS is 2000 points.

PIE CHARTS AND BAR GRAPHS

The pie charts are extremely easy to produce. A data file must be produced using a word processor or editor, and the general format for the data file is:

106 Contracts
87 Federal
51 Universities .... etc.

If this data file is stored on disk as BAR.DAT, there are then two steps to produce the pie chart. The command PIE BAR.DAT BAR.PLT reads the data file, asks the user a few questions about the form of the pie chart, and then writes the plot file (BAR.PLT) to disk. The actual plotting on the printer is initiated by the command

PLOT BAR.PLT. A sample pie chart is shown in Figure 3.

There are several options and choices available for the form of the pie chart, and these are chosen by means of the dialog with the PIE software. First, any number of slices of the pie may be 'pulled out' as illustrated in Figure 3. The slices may be unshaded or shaded with different patterns. The software chooses up to five different default shading patterns, with no two adjacent patterns alike. There are three ways to annotate the pie slices: (a) names only; (b) names and raw numbers; and (c) names and percentages (the percentages are calculated automatically). As with the line and scatter graphs, extra labels may be added to the graph. The user is also offered the option of an overall title, with up to 80 characters. The size of the pie chart may be from one inch to five inches (up to seven inches for a 132-column printer).

Perhaps the most useful feature of the pie chart software is its ease of use. In making up the example for this article, I changed the data and the form of the pie chart, and making these changes and plotting a new graph took less than two minutes.

Bar charts can be produced with DataPlotter using the same general procedure as for the other types of graphs. The basic steps are (after creating a data file): BAR BAR.DAT BAR.PLT, and then PLOT BAR.PLT. The BAR software reads the data file (BAR.DAT in my example), and then asks the user a number of questions to establish the form of the final bar graph. Finally, the PLOT software writes the graph to the printer.

Bar graphs can be more complicated than pie charts, and the BAR software offers a number of options to establish the form of the graph. The software will read a data file with data in columns or in free form, with both x and y values given, or y values only (assuming equal spacing for the separation of the bars, required for bar graphs). Furthermore, the data can have x and y values alternating, or all x values first, followed by all y values. Here again, the data must be in integer form and the TRANSF utility may be used to convert decimal numbers to integer numbers.

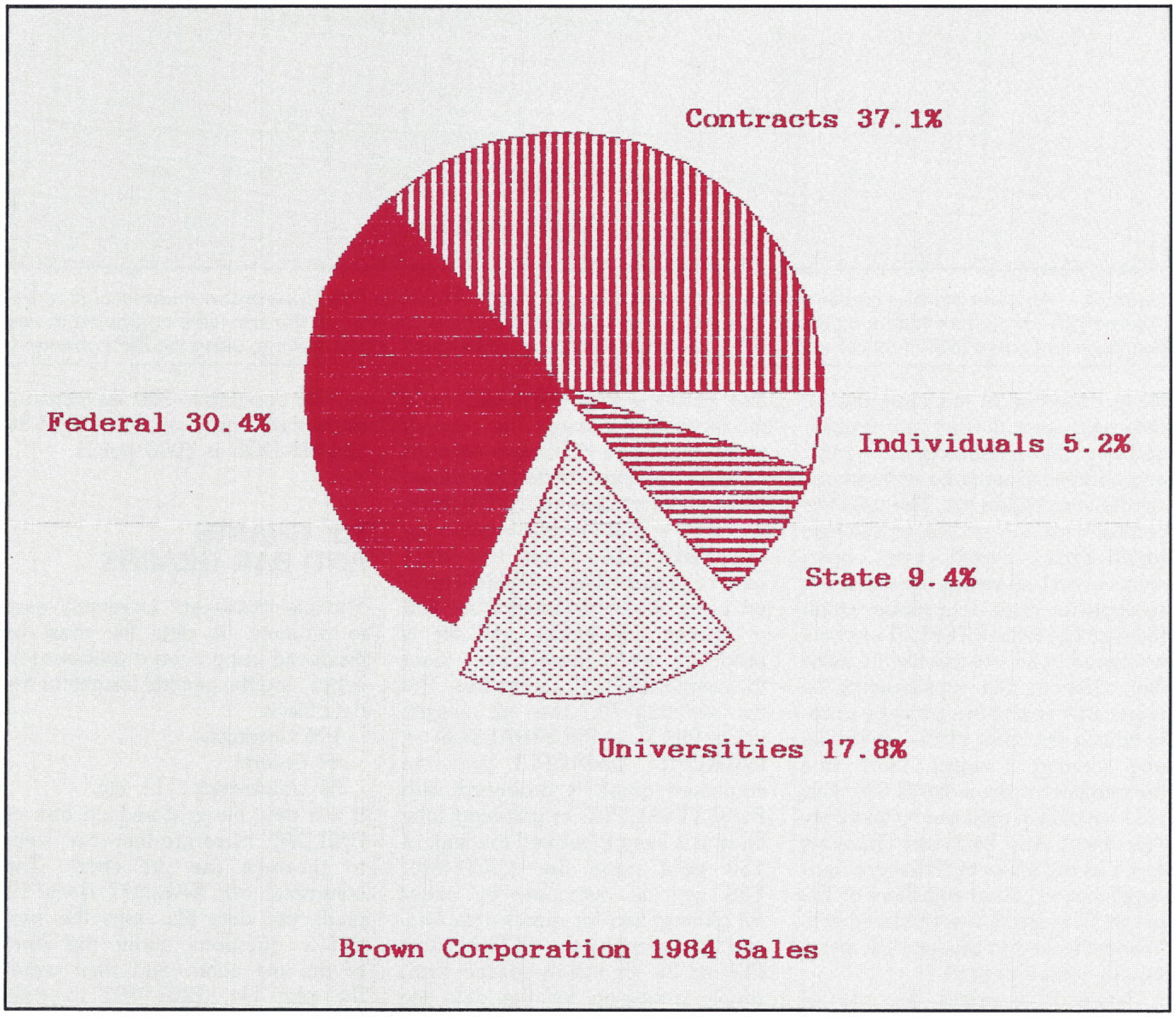


Figure 3. *A sample pie graph.*

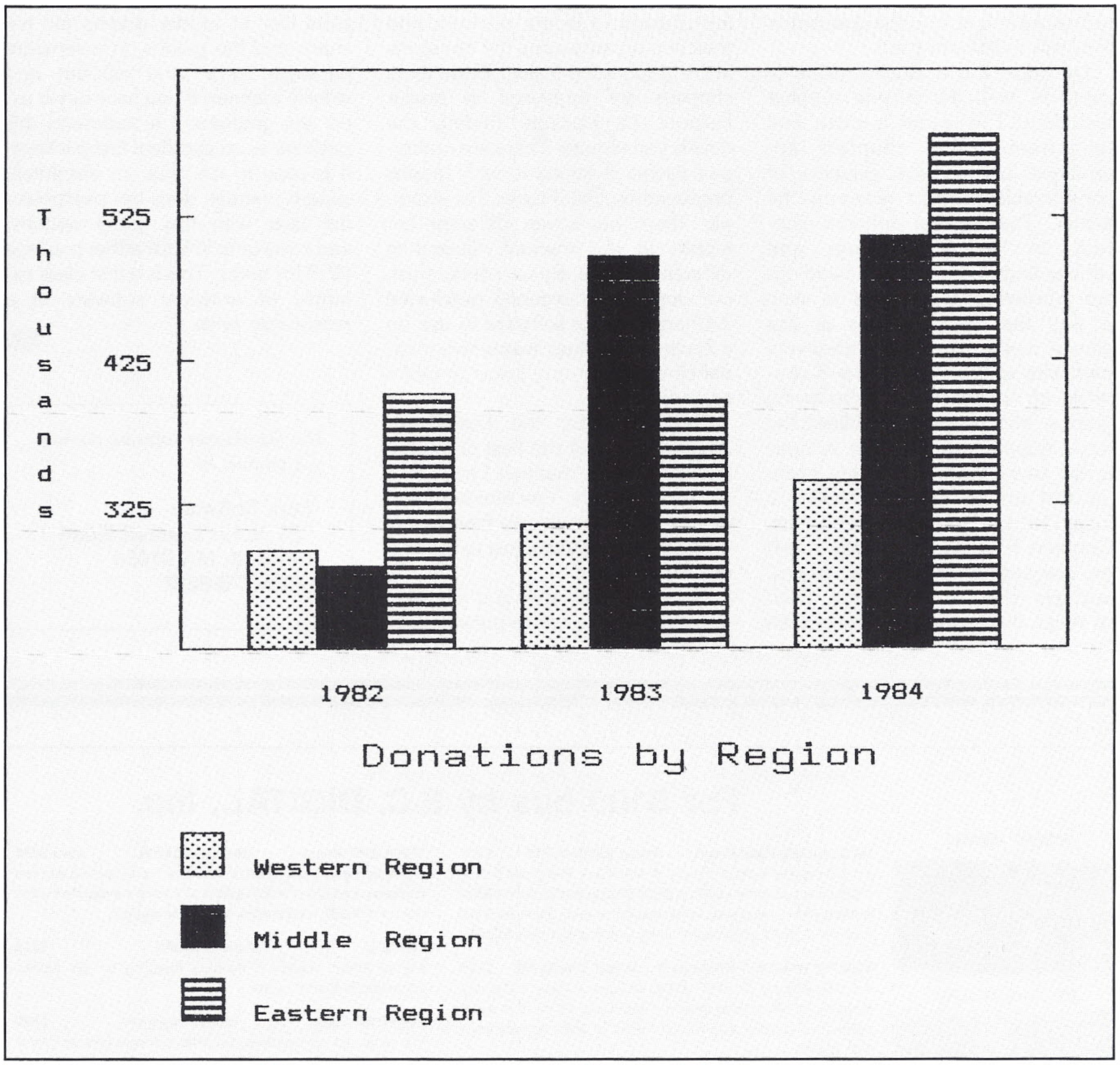


Figure 4. *A sample bar chart, showing bars in groups of three.*

The y values (the height of the bars) may be scaled automatically, or the user can specify the vertical divisions if desired. This permits a very important option: if the y-axis is scaled manually, it is not necessary to start at zero, and differences in bar heights can be shown to advantage. A sample bar graph illustrating this and other options is shown in Figure 4. Here also, text may be substituted for the x and y axis divisions. This feature is often essential for bar graphs because of the ability to annotate the bars with words, such as months of the year. Extra labels can be added to the graph, and the user may specify that a large or small font be used to indicate main title and subtitles.

Finally, there is provision for *grouped* versus *ungrouped* bars. Any number of bars may be shown grouped together, to be compared to other groups. I consider this a rather impressive extra feature for such a modestly priced graphics package. The example in Figure 4 shows groups of three. There are three choices for the shading on the bars: (a) same shading for all bars (mostly useful for ungrouped bars); (b) program selects various shadings automatically; and (c) user selects various shadings.

DOCUMENTATION

The manual accompanying this software arrived neatly bound, along with the disk in a professional-looking package. The entire manual has been printed using Fancy Font, with a plain, very elegant type style and an open layout that is very easy to read. It would be hard to overemphasize how much this contributes to the manual's ease of use. The chapter headings are large and in bold type. Where interactive programs are illustrated, output from the computer is in italics, the user's response is in an easily recognizable font, and the description is in still a different font.

On page 1 is a detailed table of contents, with the version number and date. The layout is clean and uncluttered. The chapters are arranged logically with evidence of considerable thought given to the layout. The manual adheres carefully to logical headings and subheadings. The chapter headings are labelled and arranged in such a way that it is obvious at first glance where to look to answer a particular question. Chapter 3 contains an overview on producing graphs, and chapter 4 outlines the steps required to produce a basic graph (very useful when the package first arrives). Chapter 5 tells you what form the data must take. Chapters 6, 7, and 8 contain details on line/scatter graphs, bar graphs, and pie charts, respectively. Each of these three chapters contains an introduction, a simple example, and a section on answering the questions in the graph setup dialog. These three chapters are organized in similar fashion, to facilitate finding the details you require. There are numerous useful illustrations with graphs produced by DataPlotter. For example, there are seven different bar graphs in the manual, illustrating different options. By way of contrast, our department recently purchased Z-Chart graphics software to use on a Zenith Z-100; their instruction manual contains not one single graphics example!

To summarize, the DataPlotter manual is one of the best organized and easy-to-use manuals I have ever seen for software. I recommend that anyone writing a software manual buy this software package, just to see and use the manual!

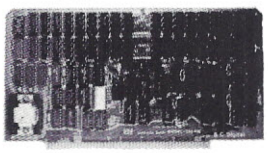
Overall, DataPlotter is a relatively uncomplicated graphics package that plots line or scatter graphs (or pie charts and bar graphs, in a separate package) in a very efficient and reliable manner. If you have never used any graphics, I recommend this package as an excellent first package. It is easy to use, has an absolutely superb manual, does not overpower the user with too many options, and costs only $69 for either package ($99 for both). This is a first-class example of excellent software at a reasonable price.

The DataPlotter software package is a product of:

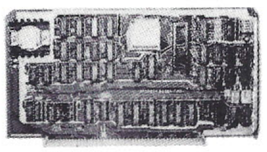
Lark Software
131 North Leverett Road
Leverett, MA 01054
(413) 773-8687

For S100 bus by S.C. DIGITAL, Inc.

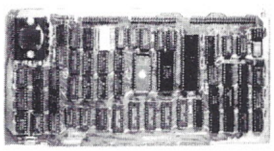
MODEL 256KMA



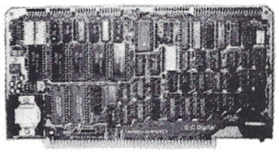
MODEL 80286CPU



MODEL FDC-1A



MODEL SUPPORT-1



1M DYNAMIC RAM Board Model 256KMA-512 $375
• 1 Megabytes • 8/16b data, 24b address • Parity per byte • 175nsec access time • will run Z80/Z8000 to 6 mhz, 8086 68000 to 8mhz without wait states • transparent refresh, unlimited DMA • works in most systems including Compupro • with 512KB.

256K/1M DYNAMIC RAM board Model 256KB-256 $325
• 1 Megabytes • 8b data, 16/24b address • parity • Memory Mapping in 16K blocks, Ideal for Ram Disk, application notes available • addressable in 128K, 192K, or 256K boundaries • with 256KB.

128K/512K STATIC RAM/EPROM Model 128KS $235
• 128K or 512K bytes • 8/16b data, 24b address • bank select, RAM shadowing, etc. • ROM simulation in RAM • Battery backup with powerfail detector/memory disable, battery holder • with 64K RAM.

FLOPPY DISK CONTROLLER board Model FDC-1A $325
• Single or double density, sides, in any combination of up to four 8″, 5¼″ drives • DMA data transfer with cross 64K boundaries, 24b address, DMA arbitration • built in monitor/boot ROM that accomodates two different processors • serial port with hand shakes, up to 19.2Kbaud • 765A software compatible, compatible with Compupro's Disk 1A • with CP/M bios.

8086 CPU Board Model 8086CPU $325
• 8/4 (or 10/5) mhz SW selectable • 8087 interface • provision to run two processors on a bus such as our Z80 CPU • convertible to 10, 12mhz clock • optimized for DRAM boards • specify clock speed.

80286 CPU Board Model 80286CPU from $350
• 8/4mhz switchable • 80827 interface • provision to run two processors on a bus • convertible to 10mhz • separate built in clock for 80287 • optimized for DRAM boards.

Z80 CPU Model Z80CPU $249
• 6 or 8mhz, specify • memory mapping in 16K blocks, addressable four M bytes.

SUPPORT Board Model Support-1 $325
• 4 serial, full handshakes, two with software programmable baud rates • Centronics • SASI interface • Real/interval timers • Calendar-clock with battery backup • expandable interrupt controllers for 8086 or 8080/Z80 • CPU switching circuit to run 2 processors on a bus such as our 8086 or 80286 and Z80 CPU boards.

I/O Interface Board Model 3SPC-N $229
• 3 serial RS-232C with switch selectable baud rates, 110 to 19.2K baud. • 1 parallel.

256K STATIC RAM Board Model 256KS $375
• 256K bytes • 8/16b data, 24b address • can be loaded in 16Kb increments • fast access time: will run in excess of 10mhz when fully loaded with 150nsec chips • Battery backup with power fail detector/memory disable, battery holder • w/256KB, less battery.

Please call for latest prices.
Prices subject to change without notice.

S.C. DIGITAL, INC.
1240 N. HIGHLAND AVE., SUITE 4 • P.O. BOX 906, AURORA, ILLINOIS 60507
PHONE: (312) 897-7749

TRANSACTION PROCESSING SYSTEMS

HIGH PERFORMANCE ● FAST ACCESS ● LOW COST

Action Computer Enterprise, Inc., pioneers in the field of S-100 multiprocessor technology since 1979, brings you:

THE TRANSACTION™

DATABASE MANAGEMENT MACHINE

The transaction processing system that's designed to break price/performance barriers with unmatched micro-based speed and capacity.

- 64 users — local or remote
- 85 Mbyte hard disk — expandable to 340 Mbytes or more
- 60 Mbyte streaming tape and floppy drive
- 80186 Service Processor 8 MHz 1 Mbyte RAM
- 8086 8-user Application Processor 8 MHz 1 Mbyte RAM
- Caching controller option
- 500 transactions per minute in a 64-user environment with a million key database
- MULTIUSER TAS — 4th Generation Application Development Language
- DPC/OS-86 Multiuser Operating System
- Upgradability. Hardware expandable, source available for application customization and maintenance.
- 64-user system for under $50,000!

APPLICATION DEVELOPERS —

MULTIUSER TAS, with its transaction-specific qualities, is an industrial strength Fourth Generation Language that's easy to use, easy to port and dramatically cuts development time. The combination of TRANSACTION and MULTIUSER TAS allows you to move into large configuration/high performance demand areas previously unavailable to micro-based systems.

Action Computer has all the tools to help you develop affordable, powerhouse transaction processing systems.

Call today, at: 1-800-821-6596 • In California call: 818-351-5451

TRANSACTION and **DPC/OS** are trademarks of Action Computer Enterprise, Inc.
MULTIUSER TAS, the version for all **dpc/os** and 16-bit Digital Research compatible operating systems, is licensed by Action Computer Enterprise, Inc. under exclusive agreement from Business Tools, Inc.

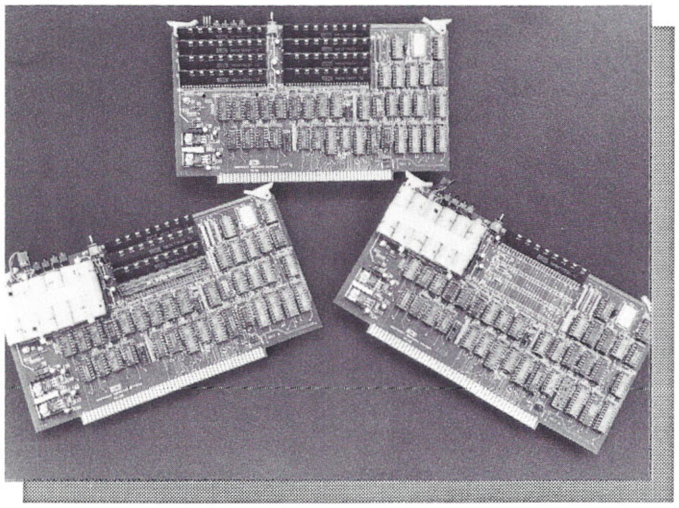
ace Action Computer Enterprise, Incorporated

430 North Halstead Street, Pasadena, CA 91107 USA TLX 279865 ACEI UR

SCS
CORPORATION

Professional S-100...

Southwest Computer Systems Corporation • 4239 North Hall Street • Dallas, Texas • 75219 • (214) 522-1512 ScaanCorp Services Corporation • 4241 North Hall Street • Dallas, Texas • 75219 • (214) 520-7270



RamCard Series I • 256KB to 2MB S-100 DRAM Card

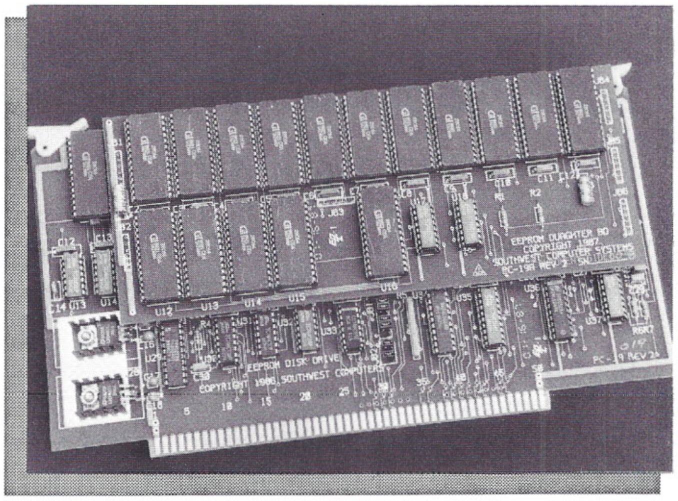
The RamCard Series I is a single-slot S-100 card designed to accept from 256K bytes to 2M bytes of dynamic RAM memory for use as a solid state disk drive emulator. Occupying only four ports in the S-100 address space, up to sixteen RamCards can exist in virtually any combination of population options and logical partitions. An on-board battery for 1M and smaller versions and an optional off-board battery for the 2M version provide complete data protection during power loss in conjunction with fully self-contained refresh circuitry. All these features, backed by SCS's one-year warranty, combine to make a professional RAM drive for the S-100 bus.

- IEEE696.1/D2 S-100 Bus Compatibility
- 2MB per card/16 cards per four S-100 ports
- 1.3MB per second DMA block transfer speed
- On-board refresh interleaved with bus access
- Fully parity checked RAM with host status port
- Battery backup included with on/off-board battery
- Full one-year warranty and 48-hour burn-in test

EROM™ Card Series I • 128K/256K EEPROM Card

The EROM™ Card Series I is a single or dual slot S-100 card designed to accept 128K bytes or 256K bytes (128K on main board, 128K on daughter board) of non-volatile EEPROM memory for use as a disk drive emulator. Occupying only four ports in the S-100 address space, up to eight EROM™ Cards can exist in virtually any combination of board capacities and logical drive partitions. A custom-programmable PAL provides security for stored data and acts as a hardware lock device to protect your valuable software from pirates and hackers. The EROM™ Card, available with 10K or 100K minimum write cycle EEPROMs, is also the ideal candidate for harsh environment applications.

- IEEE696.1/D2 S-100 Bus Compatibility
- 256K per card/8 cards per four S-100 ports
- 10K or 100K write cycle EEPROM storage
- Security PAL for data encryption/hardware lock
- Full one-year warranty and 48-hour burn-in test



List Prices

Substantial OEM Discounts Available • Dealer Inquiries Invited • Prices Subject to Change

RamCard Series I

RamCard 256K with on-board battery	$540
RamCard 512K with on-board battery	624
RamCard 1MB with on-board battery	791
RamCard 2MB with charger for off-board battery	1099

EROM™ Card Series I

EROM Card 128K, 10K write cycles	$791
EROM Card 128K, 100K write cycles	975
EROM Card 256K, 10K write cycles	1378
EROM Card 256K, 100K write cycles	1746

Options

DC to DC Converter for RamCard backup (12v-48v)	$249
Off-board High Capacity NiCad Battery for RamCard	49
Unprogrammed PAL for EROM Security Option	9
Custom Security PAL Programming	call
Card Manual Only (Refundable upon purchase)	25
TurboDOS™ Drivers for RamCard Series I & II	40
TurboDOS™ Drivers for EROM Card Series I & II	40

All orders shipped from Dallas via Airborne or UPS.
Next day delivery is avilable subject to stock on-hand.
Texas residents add 6.125% state and local sales tax.

Begun in 1971 as a high technology consulting firm, Rex F. Gallaway and Associates has since merged with MediCorp to become Southwest Computer Systems Corporation, a leader in the radio paging and telecommunications market. SCSC designed paging systems include small hospital based analog systems and massive parallel processing systems for nationwide and large regional paging operations. These are the systems designed and built around ICM processors and the TurboDOS™ Operating System. But over the course of implementing such systems, many of which involve literally hundreds of S-100 processors and multiple local area networks, SCSC has designed and built a series of custom peripheral devices, S-100 and stand-alone alike. Recently, we saw the need for these products in the rest of the S-100 market; thus, ScaanCorp Services Corporation was created as a company whose sole purpose is to sell, service, and support Southwest's line of S-100, stand alone, and PC products. We know the S-100 market is not dead, but instead reserved for a small group of hobbyists, professionals, and companies who know the advantages of a universal bus-based system. S-100 will always have its niche and the SCSC family of companies will continue to support S-100 well into the future.

TurboDOS is a trademark of Software 2000

Intercontinental Micro Systems

ScaanCorp
(214) 520-7270

Coming Soon: RamCard Series II • 8 MB DRAM in one S-100 slot

ICM'S MS-1000 SUBSYSTEM

Jay Vilhena

If someone asked me what the single most important feature of the S-100 bus is, I would answer 'the ability to run multiple CPU's simultaneously.'

The implications of this concept are enormous. Since the various processors do not have to be identical, the S-100 bus allows many different operating systems to operate concurrently inside the same box, which in turn could produce computer systems capable of running a huge variety of software applications.

It is a curious fact that the potential of such a capability remains for the most part unexplored. It has been extensively implemented in multiuser systems to give each user his or her own processor and dedicated memory (most commonly installed under TurboDOS which explicitly supports such an approach), but little more beyond that. Efforts to run several operating systems simultaneously (or at least alternatively through a console command) have been few, and I have never seen more than two implemented. Some of the earlier approaches included dual-processor boards such as Cromemco's DPU or Macrotech's MI-286 (see *S-100 Journal* No. 2, for example). More recent (and more logical) approaches utilize a slave or coprocessor board to run a second operating system (or software group). Successful examples are CompuPro's SPU-Z to run 8-bit CP/M software under a 16-bit operating system such as CP/M-68K or Concurrent DOS, Peak Electronics' 68K8-CP to add CP/M-68K to a CP/M-2.2 system, and Systems Atlanta's Z80H slave to run CP/M software in Cromemco's UNIX environment (CROMIX).

ICM'S MS-1000 package includes the following:

- *CPS-MS V40 S-100 slave*
- *TurboDOS drivers*
- *MS-1000 Software/utilities*
- *PC-compatible terminal*
- *PC-DOS operating system*
- *TurboDOS/PC*
- *Manuals and personality board with ribbon cable.*

Even more curious is the fact that, after several years of S-100 users crying out for a means of *adding* (**not** *replacing with)* MS-DOS capabilities to their *existing* S-100 systems, no one has — until now — built a slave-single-board IBM-PC-compatible computer that simply plugs into the S-100 bus and interfaces to the existing hardware and software through I/O ports. When companies like Ampro are offering PC-compatible systems on a single $300 board, one legitimately questions why such single-board designs have not been available as S-100 coprocessor or slave boards (particularly since there is a guaranteed buyer base).

ICM'S CPS-MS CARD

InterContinental Microsystems deserves our applause for being one of the first to implement PC-DOS and IBM compatibility on an S-100 slave board, the CPS-MS. Despite its various shortcomings (discussed later), this board — and its associated software — is of extreme relevance because it can be *added* to an existing system without disturbing the hardware already there or the host operating system. This is very important! It is not enough to have an S-100 which is a PC clone; if so, we might as well simply buy some other clone and go home. The important point is that we want to continue using S-100s the way they already are: powerful, versatile, professional-level systems. IBM-PC compatibility is kind of a little extra feature that we all would like to add to our long list of S-100 features, sort of a little extra topping on the sundae. The CPS-MS fits into this spirit splendidly.

MS-1000 PACKAGE AND TurboDOS INTERFACE

Although ICM will sell the board by itself (with TurboDOS drivers to operate as a regular user-dedicated slave board), the important product is the MS-1000 package. It includes everything needed to add an IBM-PC-compatible subsystem to an existing S-100 TurboDOS system. The package comes with the CPS-MS slave board, an IBM-compatible terminal that ICM calls the M/STER (actually a customized Link Technologies terminal), PC-DOS, and software to load PC-DOS into the slave board, to emulate IBM's BIOS, and to communicate with TurboDOS, plus several other utilities.

Simply plug the board into the bus (of an existing S-100 TurboDOS system), connect the board to the terminal through a port adaptor board that also comes in the package, run a few programs to install the drivers, and the PC-DOS subsystem is ready to fly. Typically, once everything is installed, the M/STER comes up under TurboDOS (it can also be made to come up under PC-DOS) at power-up. At this point, you are at a regular TurboDOS console and you can behave as a regular TurboDOS user on the system. To switch to PC-DOS simply type a simple command (*MS-SLAVE),* and the MS-1000 software then loads PC-DOS, performs some housekeeping, and returns with the PC-DOS prompt. You are now ready to run PC-DOS programs.

So far so good, but now you decide that you want to access some of the facilities — main printer, database on the hard disk, program, etc. — of the host operating system (and of course access to these resources is the real advantage of having PC-DOS running as a guest on the S-100 versus having a simple-minded PC clone). You must get back to TurboDOS. For any implementation where more than one operating system is accessible, there must be some type of software interface between the different operating systems. In ICM's MS-1000 package the interface between PC-DOS and TurboDOS is provided by TurboDOS/PC, a program by Software 2000, Inc. (the makers of TurboDOS) which runs transparently under PC-DOS. From PC-DOS simply type *MASTER* and the invisible TurboDOS/PC immediately hooks you up as a direct console to TurboDOS. (There are also other commands provided by TurboDOS/PC which deal with accessing various printers or drives on the system). When you are through, *CONTROL A* brings you back, almost instantly, to PC-DOS.

A FEW CPS-MS DETAILS

The CPS-MS board is basically a monochrome IBM-PC-compatible single-board computer which is implemented as an S-100 slave board.

The board features the NEC V40 chip, which is an 8088-compatible IC that bundles a number of peripheral functions within one IC package. The CPU runs at 8 MHz. A total of 768K of RAM reside on the board, with 640K being for PC-DOS and 128K reserved for BIOS and PC emulation. The board also features a 2-port parallel channel, and three serial ports. One of the serial ports is to connect to the M/STER PC-terminal, and the other two are fully compatible with IBM's COM1 and COM2, thus allowing I/O-oriented PC-DOS software to behave properly. The parallel ports can be connected to frame-mountable mini-boards that ICM calls *personality modules* and which allow the parallel ports to become things like Centronics or SCSI ports.

The interface to the S-100 bus is done through two I/O ports (the addresses are switch selectable). These are 8-bit ports, so a chunk of 16-bit port addresses is used up. Data transfers to and from the CPS-MS are always 8 bits at a time, so the board has the S-100 signal SIXTN* permanently tied high to indicate that it cannot accept 16-bit transfers.

Note that this slave board can be used with virtually any standard S-100 master CPU. ICM has developed drivers which allow the board to be integrated with most TurboDOS systems (8-bit or 16-bit) — however, the TurboDOS version must be 1.43 or above.

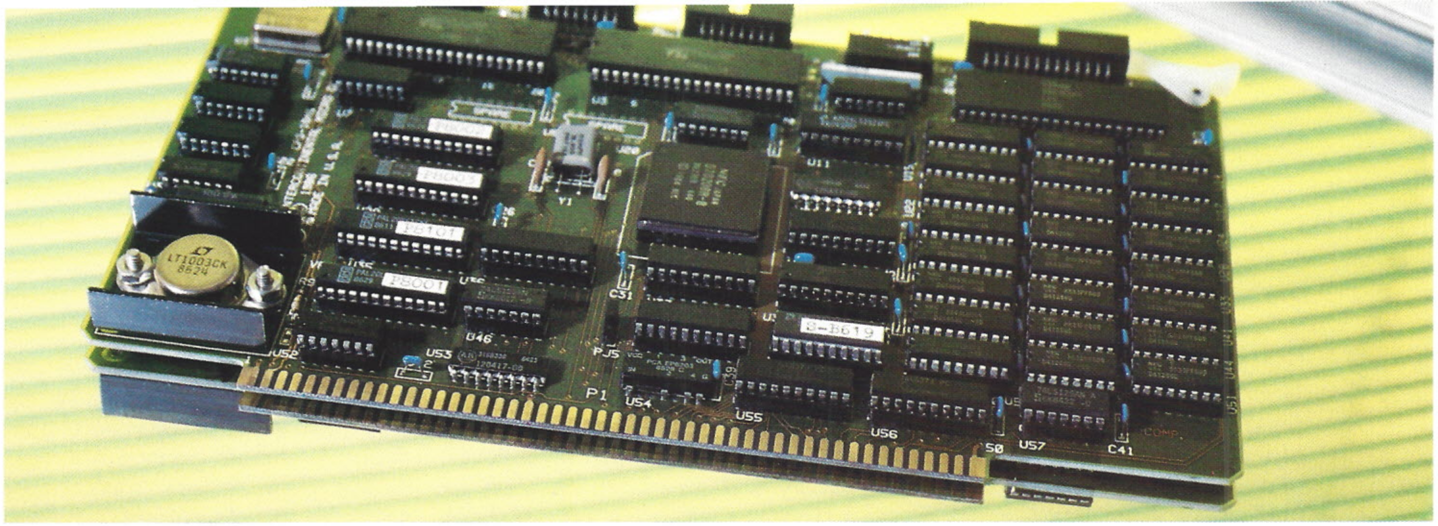
Multiple CPS-MS boards can be installed in a system to provide several users with PC-DOS workstations.

PERFORMANCE

I tested the MS-1000 package with several MS-DOS programs. It ran WordStar, SideKick, Turbo Pascal, and several other programs. It did not run WordPerfect. This is probably a typical rate: out of four or five programs, one won't run. ICM does not claim 100% compatibility and they are very open about letting users know which programs are known to run or not run. Therefore, if you want this package to run a particular program, check with ICM before buying. In addition they've indicated that they probably can 'fix' the package so it will run a specific program. Although it is unlikely that they will try this on an individual basis, they are likely to do it if an OEM buying a quantity of MS-1000 packages or a significant number of users show interest in having a specific application running properly.

Programs executing from memory run several times faster on the MS-1000 subsystem than on an IBM-PC. But slower than on an AT. Disk accesses are noticeably slow since the board must interface with the host operating system to access the drives. However, since hard disks used on multiuser S-100 systems are typically faster than low-cost PC/AT-type hard disks, similar access speeds may be attained. I did not make any comparisons in this area. Furthermore, performance here may be increased by connecting a dedicated drive through an SCSI interface to the onboard parallel channel. Again, I don't know if anyone has done that.

Unfortunately, the MS-1000 software cannot read MS-DOS program files directly. They must first be converted to TurboDOS format using a clever copy utility that comes with the package. Once converted and residing in the directory, the programs can be run indefinitely. The obvious drawback: copy protected programs cannot be converted and therefore are



not accessible to the MS-1000.

The current MS-1000 version supports only character graphics, including IBM's extended character set. A new version already running at ICM will support monochrome bit-mapped graphics utilizing a different terminal.

COST

The CPS-MS board by itself, including TurboDOS drivers, costs $795. With the terminal (required to run PC-DOS) and MS-1000 software, it costs $1395. The whole MS-1000 package, including PC-DOS and TurboDOS/PC, lists for $1595. At first, this may appear somewhat high compared to the price of even a U.S.-made clone. However, notice that you are not buying a clone, you are buying the *ability to add a high degree of MS-DOS compatibility without sacrificing any of your existing S-100 setups, plus adding an extra workstation with full access priviledges to the system/network resources.* The hardware and software required to connect several systems in a LAN would alone cost you almost that much.

Looking at it another way, suppose that you need an extra workstation in your TurboDOS installation anyway. Figure the $795 for an extra slave board plus $500 for an extra terminal, then MS-DOS compatibility is only costing you an additional $300. Of course, the slave at $795 is probably about $250 overpriced, but you can hardly criticize ICM for charging an extra $250 for its uniqueness.

I'm going over this in some detail not to justify ICM's prices or simply to be nice to them, but because we often make the wrong comparisons and forget that our objective, as I mentioned earlier in the article, is not to obtain a clone (we can do that anytime) but to *add* features to a powerful system that we wish to preserve.

DOCUMENTS AND TECHNICAL SUPPORT

ICM provides manuals for the terminal, TurboDOS/PC, and PC-DOS (produced by the original companies), plus ICM-produced technical manuals and user manuals for the MS-1000 software *and* for the CPS-MS slave board. Both the MS-1000 and the CPS-MS *technical* manuals are very complete, clear, relatively well organized, and adequate, although without circuit diagrams. The CPS-MS *user*'s manual starts out very complete and clear, but it soon reads like a chapter of the technical manual. Finally, the MS-1000 *user*'s manual is basically unreadable (I recommend that ICM rewrite this particular portion of the manuals).

Since the MS-1000 user's manual is the part that you need (and read) first — it contains the software installation procedures, it is likely that you'll end up calling ICM to find out how to install the drivers, unless you have a local ICM dealer to do it for you.

Fortunately, ICM technical support is probably one of the best around. The manual explicitly instructs how to call the Technical Staff and the company maintains a 24-hour BBS for interactive and tutorial purposes. The best indication, however, is the fact that I routinely receive letters from readers complaining about service from this or that company, *and I have never received a complaint about ICM.*

WHAT ABOUT THE REST OF US

As you have by now realized, the MS-1000 package was developed to provide a degree of MS-DOS compatibility to **TurboDOS** systems. In this respect it succeeds.

Like nearly all S-100 companies, ICM is married to a single operating system, TurboDOS in this case. Those of us who do not use TurboDOS are still at a loss for an MS-DOS slave. Concurrent DOS users, who have built-in MS-DOS compatibility, don't really care, but that still leaves thousands of CP/M and UNIX folks for example. What to do?

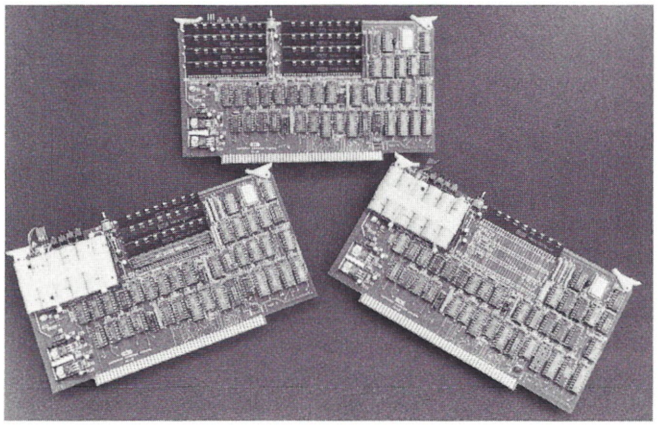
Since the CPS-MS is I/O-mapped and quite independent of the rest of the system, it could be integrated with any other operating system if you write your own drivers, including PC-DOS loader and BIOS. If you are a whiz and do it on your own, be sure to share the results. If you are an S-100 supplier with another operating system and decide to embark on such a project, integrating the CPS-MS into your current S-100 systems for commercial profit, contact ICM and they will provide details and support to help you write the software.

Finally, and most important, ICM has opened the way for MS-DOS (or OS/2) on S-100 dedicated slave processors. The field is wide open, and the rest of us are still waiting. Improve on the idea, and the cash is yours! ■

For more information on these products or the name of your nearest ICM dealer, please contact: ICM, 4015 Leaverton Court, Anaheim, CA 92807. (714) 630-0964.

SCAANCORP RELEASES EEPROM BOARD AND 2-MEGABYTE RAM-DISK

Southwest Computer Systems Corporation has announced that several of the company's S-100 board-level products will be available through its marketing company, ScaanCorp Services Incorporated. The boards, originally designed for use in SCS's telecommunications systems, include a 128K/256K EEPROM board (the EROM Card Series I) and a 256K to 2-Megabyte dynamic RAM-disk (the RamCard Series I).



Up to 16 RamCards can occupy the same I/O-port addresses for a possible 32 Megabytes of RAM-disk capacity. The on-board dynamic refresh circuitry allows RAM refreshing without CPU intervention. On-board battery is standard for the 256K to 1-Megabyte versions of the RamCard and off-board backup battery is available for the 2-Megabyte version.

The EROM is also an I/O-port addressable device, and it allows up to 256K of data to remain on-board permanently or until rewritten by the CPU. Up to eight boards can be combined for a total of 2 Megabytes of nonvolatile memory.

Drivers for these products are available for operation under TurboDOS and CP/M. For more information, contact ScaanCorp Services, 4241 North Hall Street, Dallas, TX 75219. (214) 520-7270. ★

TEN TALENT OFFERS MICROMATION SUPPORT

A full range of user and dealer support services are provided for the Micromation line of computers by Ten Talent Computer Services.

The firm offers new board upgrades, including a new Dual-Z80H slave board running at 8 MHz and an 8088 16-bit slave. Also available are hard disk upgrades up to 80 Megabytes, large-capacity disk-cache and RAM-disk boards and software, a number of peripherals, including printers, terminals, drive and tape subsystems, and even whole systems. The RAM-disk boards are available with up to 2 Megabytes of memory on a single card, and the software supports up to 4 Megabytes of cache memory.

Ten Talent also offers technical support, consultation, and repair services for the Micromation line.

For a list and prices of the available products and services, send an Editorial Feature Reply Card or contact Ten Talent Computer Services, PO Box 687, Fairfax, CA 94930. (415) 258-0799. ★

F-16 FLIGHTS PLANNED ON CROMEMCO S-100 COMPUTERS

Cromemco, Inc., a wholly owned subsidiary of Dynatech Corporation, has been awarded a contract to supply the U. S. Air Force with state-of-the-art 32-bit S-100 systems 220. These supermicros, based on Cromemco's XXU 32-bit S-100 board, will be deployed throughout the world and used as data ground terminals for the F-16 Fighting Falcon multirole aircraft. Virtually all flight plans of the F-16s will be prepared on a System 220.

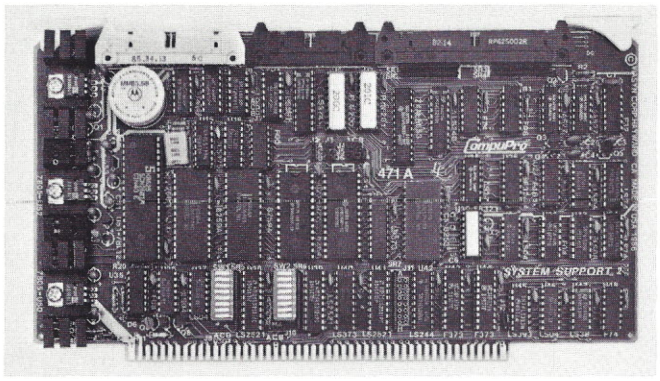
The multiphase contract has been awarded by General Dynamics to Computer Crossroads of America, Inc., a major Cromemco VAR that specializes in government computer support.

For more information on Cromemco's line of S-100 products, please contact or send EF Reply Card to Cromemco, Inc., 280 Bernardo Ave., PO Box 7400, Mountain View, CA 94039. (415) 964-7400. ★

New System Support Card From CompuPro

CompuPro is now shipping an enhanced version of the system support board. Called the System Support 2, this multifunction S-100 board provides two serial ports, two interrupt controllers, three programmable interval timers, a battery-powered real-time clock/calendar, sockets for up to 128K of battery-backed RAM or for EPROM/EEPROM, a Centronics parallel port, and a SCSI port.

The SS2 is designed to run with faster processors (over 10 MHz). The suggested list price is $995.



For additional information, send an Editorial Feature Reply Card or write to CompuPro, 26538 Danti Court, Hayward, CA 94545-3999. (415) 786-0909. ★

Removable Mass Storage Devices

Systems Peripherals Consultants, Inc. offers REMEDY, a removable hard disk drive that occupies a regular 5¼ half-height disk drive slot. The device can be inserted into or removed from the computer even with the power on.

REMEDY is available with up to 50 Megabytes of capacity. It is a completely sealed device that, according to the manufacturer, requires no maintenance. It will work with any S-100 hard disk controller that supports the ST-412 interface. An SCSI version is also available. The company also offers a controller to operate REMEDY with the Zenith Z-100.

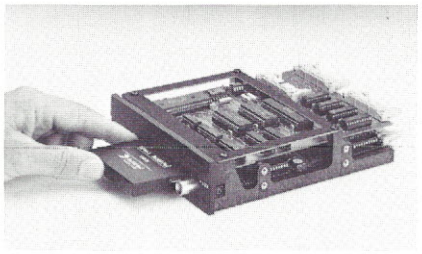
Prices range from $795 to $1595. Contact or send EF Reply Card to SPC. Inc., 9747 Business Park Avenue, San Diego, CA 92131. (619) 693-8611. ★

Bubbl-Tec, a division of PC/M, Inc., provides a solid-state, bubble-memory 5¼-drive replacement primarily for use in areas where mechanical drives are unsuitable, such as real-time process control, scientific instrumentation, and industrial automation.

The product, named the BDH-2 BUBBL-DEK system, consists of a cartridge drive mechanism which accepts removable 3-square-inch cartridges (BUBBL-PACs) each with a capacity of 128K of bubble memory. Optionally, the drive can also accomodate an extra Megabyte of fixed bubble memory. The cost for the basic drive is under $1,000.

The manufacturer claims an access time 8× faster than floppy drives. The BUBBL-PACs come in a permanently-sealed protective case and can be individually write-protected or write-enabled at any time. The drive has a built-in controller that communicates with the computer through a RS-232 or RS-422 port. Alternatively, Bubbl-Tec indicated that it is willing to develop and offer a direct S-100 interface (with S-100 controller) if there is reasonable interest.

If you are interested, contact Bubbl-Tec, 6805 Sierra Court, Dublin, CA 94568. (415) 829-8700. ★



Software

GOLDEN KEY is a software package for the UNIX system which allows VARs or end users to set up a turnkey environment. GOLDEN KEY does not require login or password nor does it allow the user to escape from the shell. When a turnkey terminal is turned on, the program automatically sets required variables and executes the application assigned to that terminal. From Goldbar Computer Systems, 3326 Transit City, IA 51106. (712) 274-7462. ★

A new enhanced version of CALLME, a remote console supervisor, is available for the Concurrent DOS series of operating systems. It includes the GOLIATH electronic mail facility and supports the XMODEM file transfer protocol. CALLME handles access to a computer via modem. From Concurrent Research, Inc., 1592 Hwy A1A, Satellite Beach, FL 32937. (305) 777-7080. ★

Other News

Source code for hundreds of useful public domain C programs is available on disk, at a modest charge, from The C Users' Group. They publish a directory (cost $10) which describes all the programs available. Contact The C Users' Group, Box 97, McPherson, KS 67460. (316) 241-1065. ★

ICM now offers software and LAN cards to connect the Zenith Z-100 to a NOVELL network. Contact ICM, 4015 Leaverton Ct., Anaheim, CA 92807. (714) 630-0964. ★

Productive Data Systems has purchased the Para Dynamics line of S-100 enclosures. Para Dynamics is no longer in business. For information on this line of products contact Productive Data Systems, 303 North Indian Ave., Palm Springs, CA 92262. (619) 323-9896.

Do you have a new product for use with the S-100 bus? Send information to S-100 Journal for free publication in this section (subject to space availability).

The System Analyzer

BY COMPUTER DOCTORS

A compact microprocessor based diagnostic analyzer designed for cost effective maintenance and repair of:

- ☐ **Floppy Disk Drives**
- ☐ **RS-232 Serial Interfaces**
- ☐ **Centronics Interfaces**
- ☐ **Hard Disks**
- ☐ **Printers**
- ☐ **Buffers**
- ☐ **Terminals**
- ☐ **Ports**
- ☐ **Controllers**
- ☐ **Cables**

Using the System Analyzer a technician can clean and align a floppy drive, test a terminal, or configure a cable in only minutes. All without use of any other test equipment.

ALSO AVAILABLE
Complete depot service on:

- ☐ **CompuPro**
- ☐ **Vector Graphics**
- ☐ **Disk Drives**
- ☐ **Other S-100 & related equipment**

Computer Doctors
4810 Calvert Road
College Park, Maryland 20740
(301) 779-5110

INDEX OF ADVERTISERS

Action Computer Enterprise	83	Lugert, S.	92
Advanced Digital Corporation	2-3	Macrotech International Corp.	6
BV Engineering	91	Micro Power Unlimited	92
CCT	57	Microsmith Computer Technology	91
California Digital	76	Nirvonics, Inc.	91
Charter Information Corp	32	NuCom Corporation	59
CO-SA, Computer und Systeme	91	Renard Systems Inc.	92
Cochlin Computer Systems	25, 92	Roy	92
Computer Doctors	90	S.C. Digital, Inc.	82
Computer Surplus Store	75	SCS	91
Crevelt Computer System, Inc.	91	SPC Technologies, Inc.	14
D&W Digital, Inc.	35	S-100 Journal	23, 59
Digital Research Computers	20	ScaanCorp	84
Discover Control Corporation	92	Semidisk	5
Fraser Business Equipment	92	Sparks Micro	92
Earth Computers	C-3	Support Service Associates	92
Echelon, Inc.	55, 58	Tarbell Electronics	22
ICM	C-2	Teletek	C-4
Inner Access	31	Ten Talent Computer Services	92
Integrand Research Corp.	58	Theta Labs	59
KepTronix	19	Vision Computers	13, 58
LPI - Micro Drive Service	92	WareCraft	45
Lomas Data Products, Inc.	60	ZeitlerSystems	91

▶ ED INTERFACE

(continued from page 7)

'puzzle' is not surprising.

For a user, the rewards of owning an S-100 machine are enormous. The price we pay is a situation where demand is greater than supply, i.e., we can't always get what we want.

I hope you decide in favor of the S-100 bus. Happy hunting! •*Jay*

I just finished reading *S-100 Journal* No. 5 from cover to cover and decided I had better renew the magazine for another year plus catch up on the two back issues I missed.

Your current 1-Meg RAM board article just blew me away. You can't easily buy such a board on the commercial market! I do have one slight problem before I order the board which I hope you can help out on. If, after spending $1150 for the full megabyte board, I find out a year down the road that the PAL chip is defective, I can kite the board if I cannot find Kevin Parker or KepTronix. I can understand that Mr. Parker may consider the PAL programming information proprietary, but there should be some allowance for the fact that he may get hit with a Mack truck before my $1150 is fully amortized.

Changing the subject, I think it would considerably enhance the *S-100 Journal* if you could maintain a BBS system. Your readers could keep in touch and discuss the various articles, download software (who has time to type in that barcode listing in issue No. 5?), and possibly the SYSOP could download from Byte and CompuServe S-100 sections and germane issues of wide interest.

Dick Fairbanks
Milwaukee, Wisconsin

I talked to Kevin and he says that he has no problem with releasing the PAL programming information. Simply ask for it when you order the board. He actually had more information about the PAL in the original manuscript, but I felt that it could be left out of the published article for the sake of brevity (it was a very long article as it was) because a reader could always get it from him directly.

I agree that we should have a Bulletin Board. I am open for suggestions from readers about what BBS software to use, how it should operate, etc. Or, if a reader with SYSOP experience and the necessary hardware (S-100 of course) would be interested in an evening job, please contact me, as the physical location of the board (or boards) is irrelevant. •*Jay*

$45 PER ISSUE BUYS THIS SPACE

List your software product here in the S-100 Software directory. All listings must be of software that runs on an S-100 system. Messages must have approx. 30 words or less. We do all typesetting free.

NIRVONICS, INC.
PO Box 5062, Plainfield, NJ 07061
(201) 561-2155

Software for CP/M and MS-DOS. Products include: UNIX-style utilities, database program, form completion tool and an arcade-style action game requiring no graphics hardware. *Orders for custom software taken.*

CO-SA, COMPUTER UND SYSTEME GmbH
Krischerstr. 70, D-4019 Monheim, West-Germany
Phone:02173-396170 Fax: 02173-52071 Telex: 8515836 amph d

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.

ZeitlerSystems
9 Pajaro Way, Salinas, CA 93901

Redirect and Pipe input/output of DIR, FIND, SORT, MORE, ECHO, CLS, WC under Concurrent 4.1. $29.95 includes shipping, CA residents add tax. Specify format: 8″ SSSD, 5″ MS-DOS, inquire about others.

SCS
166 Wright Street, Staten Island, NY 10304
(718) 442-0665

DISMAL: table-driven disassembler you can configure for almost any processor and mnemonics. Preconfigured for Z80, all documentation and source included for total user customization. Specify disc format. $50 prepaid.

CREVELT COMPUTER SYSTEM, INC.
3111 S. Valley View E-103, Las Vegas, NV 89102
TurboDOS SYSTEMS (702) 362-6946

ACCESS CONTROL SYSTEM provides control & security. Remote users have call-back protection. $95.00 retail.
NETWORK COMMUNICATION PACKAGE provides software for dial-up master-to-master networking. $125.00 retail.

Affordable Engineering Software
CALL OR WRITE FOR FREE CATALOG

Circuit Analysis • Root Locus • Thermal Analysis • Plotter Drivers • Graphics • Signal Processing • Filter Design • Report Proofreader • Transfer Function Analysis.

BV Engineering VISA/MC
2200 Business Way Suite 207, Riverside, CA 92501 (714) 781-0252

NO SOURCE CODE?

REL/PAK converts M80 format REL files to Z80 and 8080 MAC files which can be edited and then assembled. All public and external symbols inserted. Free brochure available. $99.95 prepaid.

Microsmith Computer Technology
PO Box 1473, Elkhart, IN 46515
(800) 662-4070 or in Illinois (800) 942-7317

Do You Have a Product, Service, or Software that Is Suitable for the S-100 Bus?

The *S-100 Software* and *S-100 Directory* sections of *S-100 Journal* offer an inexpensive and effective way of marketing your products or services to the S-100 community. The following rates apply to each of the above spaces:

SINGLE INSERTION $45 **FOUR INSERTIONS $160** **EIGHT INSERTIONS $280**

The above rates require prepayment in full for the total ad duration. Note that with multiple insertions the rates are as low as $35 per issue. Send your ad now for the next issue and collect the sales rewards that only a specialized magazine can offer.

s-100 directory

$45 PER ISSUE BUYS THIS SPACE

List your business or product in the S-100 Directory. All listings must be S-100 related. Messages must have 30 words or less. We do all typesetting free. Send listing w/ $45 per issue to S-100 Journal, S-100 Directory, Box 1914, Orem, UT 84057.

COMPUPRO/VIASYN CARDS NEW SURPLUS

USED DYNABYTE SYSTEMS, TAXAN MONITORS, DATASOUTH PRINTERS, MORE

Send SASE for list or call:

Discover Control Corporation

PO Box 2875, Durango, CO 81302 (303) 247-9306

COCHLIN COMPUTER SYSTEMS INC.

330 Townsend St. #107, San Francisco, CA 94107

(415) 495-5999

Authorized CompuPro System Center. On-site service and maintenance, custom software design, multiuser system integration, local area networking, user training and more.

LPI - Micro Drive Service

10020 Fontana, Overland Park, KS 66207

(913) 648-1373

Unique service for the unique Micropolis disk drives found in Vector Graphic System B, Exidy Sorcerer, and other S-100 systems. Factory qualified, fast turnaround reasonable prices.

MICRO POWER UNLIMITED

1716 Erring Rd. #102, Simi Valley, CA 93065

(805) 584-6789

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

Using a MICROMATION Computer System?

Upgrades and full product support: Fast slave processor boards (8 MHz Z80H), hard disks (to 80 MB), RAM-disk w/caching, laser printers, operating system enhancements, technical assistance, repairs and more. Complete systems available.

Ten Talent Computer Services

PO Box 687, Fairfax CA 94930 (415) 258-0799

RENARD SYSTEMS INC.

4248 Ridge Lea Road, Amherst, NY 14226

(716) 833-4758

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.

SUPPORT SERVICE ASSOCIATES

6331 Fairmount Ave, El Cerrito CA 94530

(415) 524-6766

Morrow Designs parts, service, and support for all models and upgrades to hard disk or larger hard disk. Decision Micronix multi-user or CP/M systems, and Tricep Unix System V, plus all Microdecisions.

PMMI S-100 MODEMS Professionally Serviced

Guaranteed 90 days private use. 300/600 Baud.
Paid Orders from stock. No CC.
Order PMMI/103MS, Send $54.45 incl P&H to

Sparks Micro, Box 880, Sparks, MD 21152

(301) 472-4880 *S-100 Specialists Since 1978.*

S-100 Power Upgrade

Dual Processor slave board includes:

Two 10-MHz V-30 processors, 512K memory and 8- & 16-bit TurboDOS capability. Drivers and 8-bit converter available. Only $1295. Quantities limited. Call Roy at (512) 454-9250.

RAM DISK

S-100, 2 MEG, PORT I/O

New, Warranteed, $725.00

S. LUGERT, 439 PECK SLIP STATION, NYC, NY 10272 or call (718) 622-0654

NEW AND USED SURPLUS EQUIPMENT

- ✓ S-100 Boards and Enclosures
- ✓ Disk Drives
- ✓ Software
- ✓ Computers
- ✓ VDTs and Monitors
- ✓ Printers and Hardcopy Terminals

and much, much more!

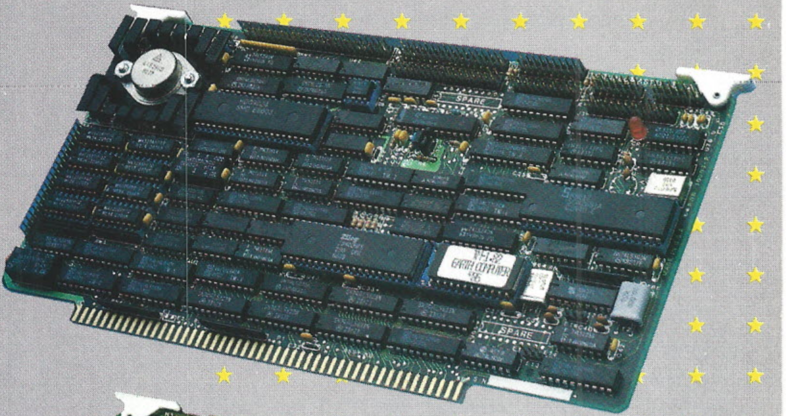
FRASER
business equipment
P.O. Box 7 523 Franklin Street
Reading, PA 19603

Send SASE or call
for more information...

(215) 378 - 0101 Ask for Chuck Ryan!

EARTH LAUNCHES NEW STARS

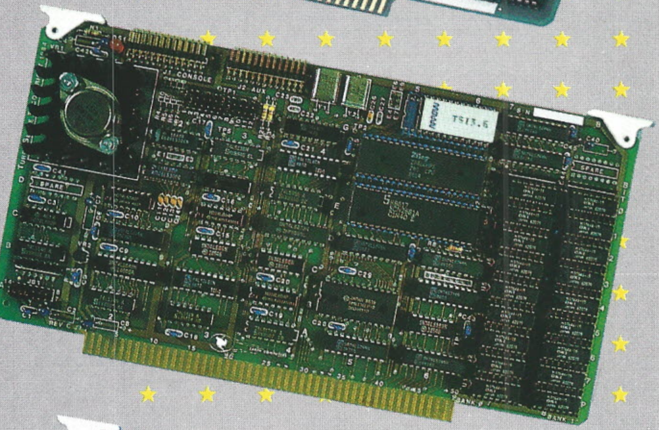
EARTH COMPUTERS launches some of the 'Hottest' new stars in the S-100 Universe. EARTH's new stars are fully compatible with the Multiuser TurboDOS operating system, and will operate in most S-100 systems, including pre-IEEE 696 systems.



TURBOMASTER 8

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

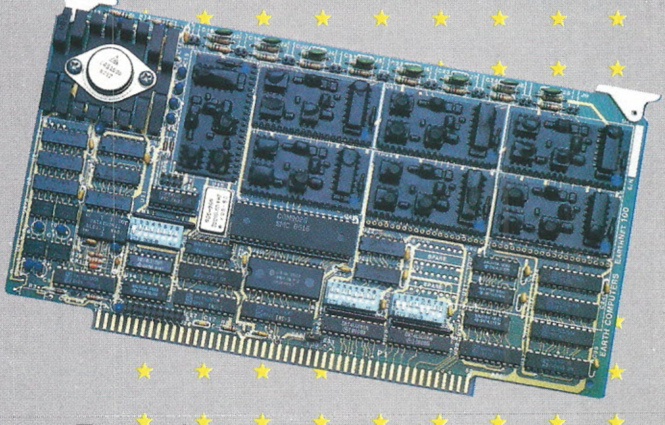
- On-board ST-506 Winchester Controller
- TurboDOS, CP/M, MP/M compatible
- 5¼″ and 8″ Floppy Controller
- Up to 256K of memory
- 8-MHz, Z-80H CPU
- 2 serial ports, 1 Parallel port



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
- 128K of RAM
- 2 RS-232 ports, 50-38.4 KBaud
- FIFO communications
- Data transfers to 1 MByte



EARTHNET S-100 and EARTHNET PC

The low cost, ARCNET-compatible way to tie S-100 systems and PCs.

- Token-passing architecture
- 2.5 Megabit data rate
- 255 nodes per network
- 2000 feet between nodes
- Simple coax wiring
- Optional on-board active hub

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

Teleteknology

One Success After Another.

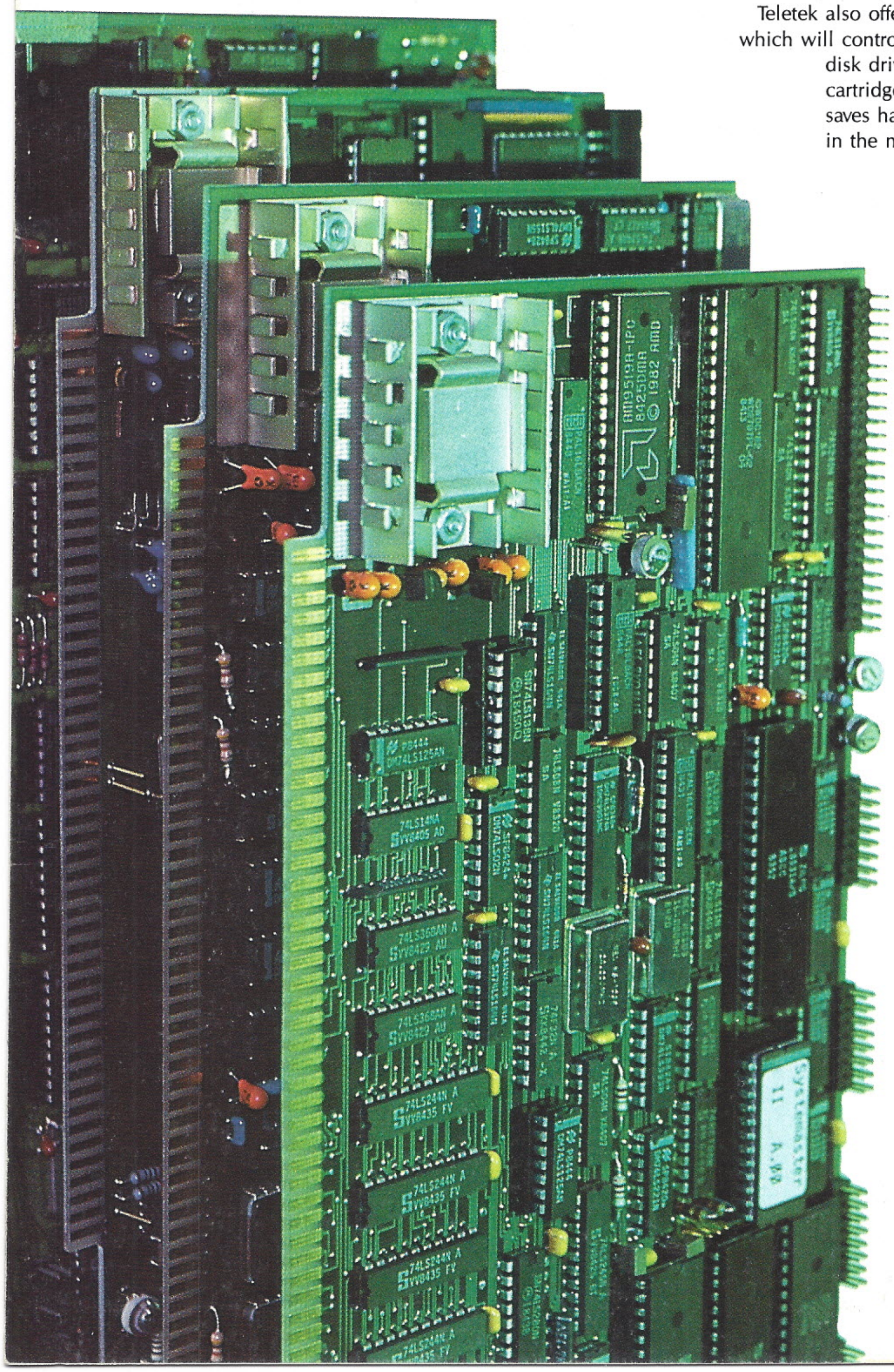
Since 1968 Teletek has been a leader in the design and manufacture of single board computers, controllers, memory boards and interface boards.

Teletek offers a variety of 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, 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 Z80-H CPU speed and up to 128K of on-board dynamic RAM. It is the ideal high-performance system controller/file server for a multiuser, multiprocessing system.

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 multiuser, multi-processing operating system TurboDOS by Software 2000. Using TurboNET, Teletek's new Local Area Network, PC-compatible workstations may share a single network. The benefits are obvious: The cost savings of shared peripherals, almost unlimited system expansion, and the ability to run a myriad of application software written for the MS-DOS and PC-DOS environments.



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

In Brazil:
DANVIC S.A.
176 Rua Brasilio Machado
01203 São Paulo, Brazil
tel: 826-8900 (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 95838
(916) 920-4600
Telex #4991834 Answer back - Teletek