ISSN # 0748-9331

THE COMPUTER JOURNAL®

Programming - Applications - User Support

Issue Number 26

$3.00U.S.

THE COMPUTER JOURNAL
190 Sullivan Crossroad
Columbia Falls, Montana
59912
406-257-9119

Editor/Publisher
Art Carlson

Art Director
Donna Carlson

Production Assistant
Judie Overbeek

Circulation
Donna Carlson

Contributing Editors
C.Thomas Hilton
Donald Howes
Jerry Houston
Bill Kibler
Rick Lehrbaum
Peter Ruber

***Subscription rates**—$14 one year (6 issues), or $24 two years (12 issues) in the U.S., $22 one year in Canada and Mexico, and $24 (surface) for one year in other countries. All funds must be in US dollars on a US bank.*

Send subscriptions, renewals, or address changes to: The Computer Journal, 190 Sullivan Crossroad, Columbis Falls, Montana, 59912, or The Computer Journal, PO Box 1697, Kalispell, MT 59903.

***Bulletin Board**—Our bulletin board will be on line 24 hours a day at 300 and 1200 baud, and the number is (406) 752-1038.*

Address all editorial and advertising inquiries to: The Computer Journal, 190 Sullivan Crossroad, Columbia Falls, MT 59912 phone (406) 257-9119.

Editor's Page

Software vs. Hardware

Which technology is the driving force in the microcomputer industry, software or hardware? It's like asking which came first, the chicken or the egg, because one can not exist without the other.

Way back in ancient history (six or seven years ago) the Apple II® was popular with the hobbyists, but it took Visicalc® to fire up everyone's imagination about what the micros could do for business. Visicalc was the first step in legitimizing the micro and resulted in greatly increased sales for the Apple II.

When the IBM PC® appeared with 16K and one drive, there was very little software available for it. As new software was developed it was apparent that the system needed enhancement, so people added RAM, additional drives, and hard disks — and even better software was developed to use the expanded capabilities. Programs like Lotus 1-2-3® completed the legitimizing of the micro which was started with the Apple II and Visicalc.

The driving force is neither software or hardware, but rather the solutions to problems. A few computer enthusiasts (like many of us) will buy a program or system because it's new and different, but the large volume sales are to fill the needs in the office and the factory. They are not interested in neat ideas, they just want something to solve their problem.

Obviously, the hardware has to exist before the software can run on it, and a study of the successful systems shows that the hardware is usually available on the market for quite some time before the software which assures it's success is fully developed. Astute computer buyers don't buy hardware, they buy the system which runs the software which they have selected.

It is the broad pool of available third party software and peripherals which results in the successful sales of a system, but the system has to be available before the software is written. It's something like pulling yourself up by your bootstraps.

Desktop Publishing—The Next Visicalc?

There is a lot of hype and exagerated claims about desktop publishing, but it has a tremendous potential and may the primary mover for selling micros for the next two or three years.

According to William J. Dorman, president of Dorman Associates, Inc., as reported in the September/October issue of The Typographer, "Desktop Publishing is emerging as a leading application of personal computers for both corporate and commercial market segments." DAI estimates that by 1990 the market for dedicated desktop publishing systems and PC composition and page layout software is expected to reach $651 million. The revenue estimates include dedicated PCs, software, image scanners, laser printers, typesetters, storage devices, and service.

While the term desktop publishing is being used to cover everything from church bulletins to major books, this is a very broad field with many different requirements. Here at TCJ, we have been involved in both professional phototypesetting and microcomputers so it is easy for us to evaluate the systems being presented. For others who have experience in either computers or typesetting, but not both, it is much more difficult — and for those with experience in neither, it is a real bucket of worms.

The battle for desk top publishing (DTP) sales is heating up, and there will be a blood bath because too many people are entering the business without the necessary experience or financial strength (sound familiar?). The potential DTP buyers need professional advice on selecting and implementing hardware and software packages, and they'll also need support in the way of special software for text file filters, disk format conversions, etc., plus hardware support for equipment interfacing and maintenance. This is a ripe field for computer consultants who have the required knowledge for this specialized field.

Although the typesetting requirements are specialized, much of the equipment is standard and normal maintenance and repair is no different than regular computers. Some typesetter manufacturers charge $100 per hour including travel time and in a rural area like ours the owner can figure on paying at least $1,000 for any service call from the manufacturer — even if only to replace a blown fuse or panel lamp. That's if the service person has the right part with them, otherwise it can cost another $1,000 for the second trip to replace a 25 cent item. One owner was faced with paying $950 for a rebuilt replacement 8" drive (standard SA800) plus the $1,000 service call charge. A local computer person just plugged in one of the drives being sold so reasonably now for a savings of about $1,700 dollars, even after making a nice profit.

Many of our readers are capable of providing the software and hardware support needed in this growing field if they have access to the perculiar requirements of the DTP market. TCJ is not the place to talk about points, picas, leading, kerning, letterspacing, and all the fine points of typography; but we can talk about programming, text filters and formatting, and equipment interfacing. We could even run a short series on 'How to Talk to a Typographer' to explain some of the terms they use and put this on our BBS if there is enough interest. Let us know if you would like to see more information on this subject.

Opportunities

The national economy is soft with more and more manufacturing being moved overseas, and the job market in the computer industry is chaotic with lots of layoffs. Colleges and universities are turning out thousands of "Computer Scientists," and everyone with a degree is offering their services as a consultant. The real opportunities are not in the computer industry which makes the systems, but rather in the industries which need to use computers.

What is needed right now, is not pure computer experts with no idea of what's going on in the real world, but people who understand how to use computers to solve problems. Business doesn't buy computers — it buys solutions to problems.

Refined Goals for TCJ

As the industry changes, so must we! And I'm using that as a collective we meaning all of us as a group, not just TCJ. Five years ago we were involved in creating the microcomputer environment — now we have to concentrate on applying micros to problem solving and needs, for both industry and ourselves.

TCJ will provide the information you need with more in-depth articles on how to solve problems with the systems available now, and how to prepare yourself to work with the new systems which are coming. Much of the coverage will be on programming because all of us can write or modify programs while few of us are in a position to manufacture a hardware system, but we will continue to carry the information on both new and existing hardware systems necessary to understand and program them. Many of the upcoming applications for networking, LANs, control, and desk top publishing will require the use of buses, and the article on bus systems in this issue is the beginning of a series on using

(Continued on page 53)

BARGAIN

68000 OPERATING SYSTEM

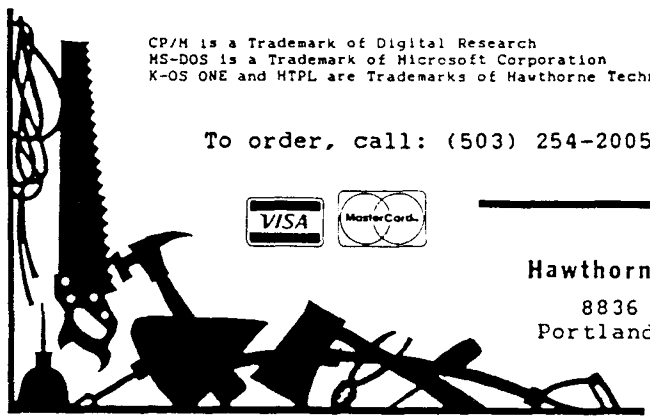
$50

Now you can put K-OS ONE, a 'simply' great operating system, on your 68000. It is simple to use and has all of the features you would expect from an operating system in the class of MS-DOS or CP/M. It's easy to customize because you get source code for the operating system and the command processor.

They are written in HTPL, a powerful, high level, language hybrid. You don't have HTPL? No problem, the HTPL compiler is included. The K-OS ONE operating system is small. You can edit and re-compile the whole operating system in only 64K of RAM.

The package also contains a line editor, a 68000 assembler and a manual covering the whole package.

With K-OS ONE, you can read and write ASCII files on MS-DOS format diskettes. This gives you the tools you will need to port K-OS ONE to your hardware.



CP/M is a Trademark of Digital Research
MS-DOS is a Trademark of Microsoft Corporation
K-OS ONE and HTPL are Trademarks of Hawthorne Technology

To order, call: (503) 254-2005

VISA MasterCard

Hawthorne Technology
8836 SE Stark
Portland, Or 97216

Letters From Our Readers

More on 8-bit versus 16-bit

Advertisements in trade magazines are often profound in many ways. When reading a popular trade journal I came upon a two page advertisement which sported a picture of a large bengal tiger laying next to a small, white kitten. The caption read, "The difference is power." This is a profound concept when applied to the ongoing debate between eight and sixteen bit system users.

For more years than I care to remember I have been an advocate of eight bit systems. I have heard all the arguments for both technologies. Each have their valid points, and each their fiction.

For writing I have always preferred my little AMPRO CP/M system with a rather transient 20 megabyte hard drive, (the hard drive is used on my AMPRO '186 as well). My editor requests, nay demands, text in WordStar format. My library of software is CP/M based. Yet, the Borland Editor Tool Box, which does not run unaltered on the '186 allows me to produce a text editor which makes the CP/M tools I have seem insignificant.

When I annotate listings, or edit text, I am constantly needing to refer to the program source code, or other reference material. The 512K Little Board '186 with a very simple text editor allows me to edit, view, and perform "cut and paste" operations not possible with a 64K system. The ability to work any number of files at once on the screen is a tool I personally need.

Other authors have compared the technology war to automobiles. But, I don't feel they have fully explained their rationale. One would not use a semi-trailer truck to take the kids to school. In the same thought pattern one would not use a Volkswagon for heavy industrial work, though some might try from loyalty to a given technology.

Instead of continuing the debate, an exceptionally foolish argument, over which technology is the best, the fastest, etc., why can't we focus these energies into more productive efforts. For the first time in computing history we, the common people, have a choice as to the amount of computing power we may apply to a given application. How long will we defend that which needs no defense? Instead why don't we use the space in our magazine, and the efforts of warfare to expound upon the proper uses of each technology? In the final analysis "The difference is power." I would hope the readers of TCJ would have the intellect to focus upon the use and selection of power. But, even as I write this I realize this is a lost hope. Man has never been overly adept in the use and selection of power. He uses it to conceal opinion, bigotry, ignorance, and fiscal insecurity. Why should computerists be any different?

T.H.

PS: Should anyone have a PC or CP/M system they are ashamed to admit owning, they may be rid of it by sending it to me in care of TCJ. In this wise they have nothing they may feel the need to defend. I need a 100% PC compatible portabe for research, the rest will be sent to the Multihandicapped Blind Foundation for distribution to people who cannot afford to partake in this exclusive argument.

68000 Hacker

Sure enjoy TCJ, may you prosper!

Yes, a thousand times yes, to your question about interest in a hacker's 68000 system. I'd hoped Atari or Lee F. would ultimately fill the bill, but they aren't so let's go on our own. The EMS system, from all reports, is far from satisfactory and the system presented in The Computer Smyth might have been great as a controller, but not as a general purpose system. I, for one don't know of any other bonafide contenders. I'll start building as soon as I see a schematic/PC board.

Delighted to see you're going to have Clark Calkins as a contributor. I'd like to know a lot more about his dissassembler and technique. His stuff has to be seen to be believed and is the price ever right. Hope you get him into print in a hurry.

Once again, thank you for a good and constantly improving periodical. Your and Dave Thompson's efforts provide considerable comfort and stimulation.

C.H.

Graphing Algorithm Needed

I am looking for an algorithm which will calculate the shortest distance between point A and point Z through n points. Often called a "graphing" problem, and what I'm looking for is a solution to an "undirected" graph — i.e., each existing edge can be travelled in either direction. N.B. Similar to, but not the same as, THE TRAVELLING SALESPERSON. In my application, not every single point is connected to the others.

B.H.

6502 CP/M Clones

In issue #25, D.E. voiced a query as to CP/M clones for the 6502 processor. I have been using an operating system called DOS-65 for the past four years on my ROCKWELL AIM-65. This is VERY CP/M look alike, and I have been quite impressed with it's performance. I obtained the system from:

Cedar Valley Computer Association
P.O. Box 671
Marion, IA 52302

but the operating system was written by:

Microsystem Technology
1363 Nathan Hale Dr.
Phoenixvile, PA 19460

It of course requires some tailoring (like writing your own BIOS), but beyond that, it works fine. It came with an assembler, a debugger, and an editor, all also CP/M utility clones. Most of my first disk based software was written for this operating system, so I have quite a fondness for it!

I don't know if Microsystems Tech is still around, but CVCA should be able to tell you. They, however quit selling the operating system.

As a sidenote, there is also a very good FORTH implementation running around for 6502, written by:

J.W. Hance
1789 Austin Rd.
Miamisburg, OH 45342

I have corresponded with Mr. Hance just this year, so that address should be fairly good.

A.M.

(Continued on page 20)

Bus Systems
Selecting a System Bus
by Art Carlson

When designing a system, which do you choose first, the microprocessor, the operating system, or the bus structure? You hear a lot about the advantages of the different CPUs and operating systems, but very little about bus structures. It's as though the bus just comes along with the CPU and OS, and is not one of your choices.

Whether or not you should concern yourself with the bus depends on your intended use for the system. If it is just for wordprocessing and spreadsheets, base your decision on other parameters — but if it is for real time control or other hardware interfacing intensive applications, the choice of the bus is of primary concern.

Why Bother Learning About Buses?

With the flood of PC clones coming from Taiwan you'll soon be able to buy a system complete with keyboard, drives, display, and software for less than $500. The PC has an open architecture with slots, so all you have to do is to buy cards and throw them into the slots, right? Well, perhaps, because you have access to the bus and you can add serial, parallel, SCSI, IEEE 488, and other ports which enable you to communicate with printers, laser printers, hard drives, modems and other standard office peripherals — but industrial applications have additional requirements and the buses developed for these uses provide many powerful enhancements such as subsystem buses for arbitration and interrupt service.

The appliance computer market for normal office type uses is saturated and offers little opportunity for hardware development (if you need one, just go out and buy it). The real needs in that market are for custom programming and system implementation using standard off the shelf components.

I read a tremendous amount of material, and the current job opportunities are in fields other than personal computers, spreadsheets, wordprocessing, dataprocessing, etc. The activity is in engineering, the laboratory, and the factory; and the microcomputers are often dedicated units without keyboards, drives, or displays, which may be interfaced to a master unit.

The selection of the bus is crucial for these applications (which I'll call non-pc for non-personal computer office applications), and no one bus is best for every application. You have to select the bus which best solves the problems for a specific project. As stated in the October, 1986, IEEE Spectrum, "A modern high-speed bus with a 32-bit data path, designed to support multiple processors, is complete overkill for many applications. A cheaper, slower, 8-bit bus, optimized for a single processor, may be perfectly adequate." and "Some instrumentation buses are designed to connect crates of test and measurement equipment hundreds of feet apart; others function only within a rack of equipment."

Some of the better known buses in current use are: STD Bus, S-100, PC bus, GPIB(IEEE 488), Multibus, Versabus, VME bus, Multibus II, G-64, and G-96. Two years ago I would have included the Apple II bus, but I think that it is dead for now.

Even with all these available, there is still a lot of bus development activity for highly specialized applications. The Electronic Industries Association has been working on the CEBus (Consumer Electronics Bus) including a command and addressing language that would become part of a unified residential control system. Standards are currently being developed for four residential distribution systems: the power line (PLBus), the wired bus (WIBus), a single room infrared remote system (SRBus), and low power RF (RFBus).

Automobiles will include a lot more microprocessors and sensors spread around the vehicle, and the industry is developing a multiplexed bus to minimize the amount of wire. We accept our cars as commonplace, but the automotive environment is quite severe with temperatures from −35 to +140 °F, high humidity, dust, shock, and vibration.

The automotive manufacturing industry, led by General Motors, is also developing MAP (Manufacturing Automation Protocol) which is based on the International Standards Organization's 7-layer OSI (Open Systems Interconnection) model for network communications.

In order to participate in system design, application or implementation, and programming for these new fields, you'll have to be familiar with bus structure and protocol — and they won't all be in military or space applications, they'll be in your home, in your car, and in the factories. I feel very strongly that we must learn about bus structures and protocol or else be left out of future developments.

System Requirements

Every application has its own requirements, and what I need may not be the best answer to your needs — but it is a starting place. My needs are for an experimental system which can be used to investigate a variety of techniques and to evaluate various hardware and software products. While I would like to build a permanent ground temperature and water level monitor, and a sophisticated solar heating evaluation and control system, most of my projects will be of a temporary nature and will be disassembled when completed. I can't afford the high-priced state of the art products, but neither can I afford to waste time with low priced toys.

I want the flexibility of using multiple 8-bit CPUs, 16-bit CPUs, 32-bit CPUs, and microcontrollers with the choice of using their own I/O or the system I/O. One of my primary considerations is that the system does not limit my choice of CPUs or I/O, and that I can easily change from one project to another.

I don't feel that any CPU will ever be fast enough to do everything! Regardless of how fast the CPU is, we'll always want to do it faster. Silicon is cheap and I want to use multi-CPUs rather than multi-user designs (I'm considering the users as devices, not people), although some individual devices may incorporate small multi-tasking programs. I consider the host to be a master which tells slaves what to do, but which does not participate in the actual execution of the task. For example, the host could command that a stepper motor be ramped up to a speed of 2,000 pulses per minute for 3 minutes then ramped down to zero, while monitoring an encoder to be sure that the shaft really was turning, monitoring the torque required of the driven device as an indication of process condition, and monitoring overtravel limit switches; using interrupts to demand immediate response from the master in the event of an out of spec condition. A dozen or more similar operations occurring during the same period would drive any concurrent operating system nuts, and how would a concurrent system with a single CPU keep everything else running if one of the tasks demanded full attention in an emergency situation?

Others will say that the new CPUs are so fast and so powerful that they can do all this and still be able to handle the emergency situation, but this is only true (if it is even true at all) if the system is designed so that the CPU has a lot of slack time, and this means that nothing is running as fast as it could run with a CPU's full attention. I prefer to use one CPU for one function, with some possible minor multitasking as part of that function. We can quibble about what consists a function and how finely the job should be divided, but I'd rather just have a flexible expandable system so that the functions can be combined, divided, or regrouped as needed. I believe in reliability thru simplicity with numerous portions each doing their job extremely well instead of one complex device trying to do everything. I'm talking about a control application in automation, robotics, manufacturing, or other significant areas where system reliability and response time are crucial.

Practical Solutions

There is no one right choice when selecting a bus system. You have to consider the requirements, and the resources and time available, and select from what is available. I have decided that, for now, I want to use an existing host system with keyboard, display, drives, printer, operating system, languages, etc., and concentrate on applications instead of the host system. My interests do not currently include color, graphics, or sound, which are the main thrust of the popular micros.

Hal Hardenbergh (DTAK GROUNDED) commented on the "Hacking the 68000" statements in my #25 editorial and said that since the lowest priced and most common system is the PC clone, we should use it as the host and interface to our 68000 board thru a port. This makes a lot of sense because it allows anyone to participate without buying a lot of non-standard equipment.

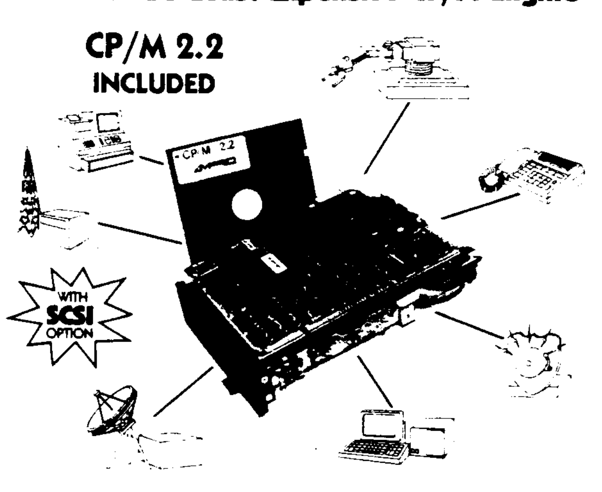
A very important point to consider is whether you are doing something for your own use, if you want to work in cooperation with others, or if you intend to market a commercial product. If you're working by yourself, anything that satisfies you is OK; If you want to work with others, you'll have to use equipment compatible with theirs; if you have a commercial product in mind, you'll have to either supply a complete package or something which works with the systems your customers use — and most of them will have IBM PC clones.

Most of my computer use right now is in the areas of wordprocessing, text file manipulations, data base, and communications, in support of TCJ and Rockland Publishing; but

Little Board™....$249

The World's Least Expensive CP/M Engine

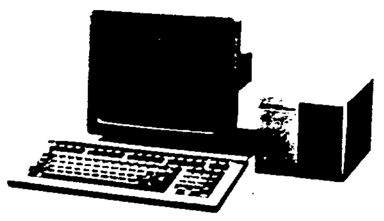
CP/M 2.2 INCLUDED



- 4 MHz Z80A CPU, 64K RAM, Z80A CTC, 4-32K EPROM
- Mini/Micro Floppy Controller (1-4 Drives, Single/Double Density, 1-2 sided 40/80 track)
- 2 RS232C Serial Ports (75-9600 baud & 75-38, 400 baud), 1 Centronics Printer Port
- Power Requirement: -5VDC at .75A; +12VDC at .05A / On board -12V converter
- Only 5.75 x 7.75 inches, mounts directly to a 5-1/4" disk drive
- Comprehensive Software Included:
  - Enhanced CP/M 2.2 operating system with ZCPR3
  - Read/write/format dozens of floppy formats (IBM PC-DOS, KAYPRO, OSBORNE, MORROW...)
  - Menu-based system customization
  - Operator-friendly MENU shell
- OPTIONS:
  - Source Code
  - TurboDOS
  - ZRDOS
  - Hard disk expansion to 60 megabytes
  - SCSI/PLUS™ multi-master I/O expansion bus
  - Local Area Network
  - STD Bus Adapter

BOOKSHELF™ *Series 100*

Fast, Compact, High Quality, Easy-to-use CP/M System



Priced from $895.00
10MB System Only $1645.00

- Ready-to-use professional CP/M computer system
- Works with any RS232C ASCII terminal (not included)
- Network available
- Compact 7.3 x 6.5 x 10.5 inches, 12.5 pounds, all-metal construction
- Powerful and Versatile:
  - Based on Little Board single-board computer
  - One or two 400 or 800 KB floppy drives
  - 10-MB internal hard disk drive option
- Comprehensive Software Included:
  - Enhanced CP/M operating system with ZCPR3
  - Word processing, spreadsheet, relational database, spelling checker, and data encrypt/decrypt (T/MAKER III™)
  - Operator-friendly shells; Menu, Friendly™
  - Read/write and format dozens of floppy formats (IBM PC-DOS, KAYPRO, OSBORNE, MORROW...)
  - Menu-based system customization

DISTRIBUTORS

ARGENTINA: FACTORIAL, S.A., (1) 41-0018, TLX 22408 **BELGIUM:** CENTRE ELECTRONIQUE LEMPEREUR, (041) 23-4541, TLX 42621 **CANADA:** DYNACOMP COMPUTER SYSTEMS LTD., (604) 872-7737 **ENGLAND:** QUANT SYSTEMS, (01) 253-8423, TLX 946240 REF:19003131 **FRANCE:** EGAL+, (1) 502-1800, TLX 620893 **SPAIN:** XENIOS INFORMATICA, 593-0822, TLX 50364 **AUSTRALIA:** ASP MICROCOMPUTERS, (613) 500-0628 **BRAZIL:** CNC-DATA LEADER LTDA., (41) 262-2262, TLX 041-6364 **DENMARK:** DANBIT, (03) 66-20-20, TLX 43558 **FINLAND:** SYMMETRIC OY, (0) 585-322, TLX 121394 **ISRAEL:** ALPHA TERMINALS, LTD., (3) 49-16-95, TLX 341667 **SWEDEN:** AB AKTA, (08) 54-20-20, TLX 13702 **USA:** CONTACT AMPRO COMPUTERS INC., TEL.: (415) 962-0230 TELEX: 4940302

IBM®, IBM Corp.; Z80A™, Zilog, Inc., CP/M®, Digital Research; ZCPR3™ & ZRDOS™, Echelon, Inc.; Turbo DOS®, Software 2000, Inc.; T/MAKER III™, T/Maker Co.

AMPRO
COMPUTERS INCORPORATED

67 East Evelyn Ave. • Mountain View, CA 94041 • (415) 962-0230 • TELEX 4940302

my primary personal interest is in measurement and control. I'm looking for a development system, but the current crop of mass produced personal computers are poor prospects for control system hardware interfacing because they are designed for games and office use.

It would be nice to have all the devices sitting on the same bus for high speed communications but the interface can be accomplished using standard ports, and in fact this will be necessary where the devices are located at a distance from the master. Some of the definitions of "Bus" are: "The physical channel over which electric signals are transferred between the components of a system, along with the protocol rules governing the transfer", "A means of distributing a set of signals so the computer can be interfaced with memory and external devices."

According to these the RS-232C interface can be considered a bus, although we don't normally consider it as such. Some of the topics we'll be evaluating are distributed vs. centralized processing, and selecting from various communications ports and buses. If the host system you choose does not provide for access to the bus you'll have to interface through one of the ports provided.

I started with an Apple][+, and its open architecture with good third party support (I/O and A/D boards are available for under $100), and an abundance of detailed technical books made interfacing easy. The problem is that I don't like Apple's operating system or disk interfacing.

My next choice was S-100/IEEE 696 because it provides 24 address lines (16 Mbytes), 64K I/O ports, 10 vectored interrupt lines, 16 data lines, up to 16 masters on the bus, plus status and control lines. There was a lot of S-100 activity, but frankly, the S-100 bus was more than I was ready to handle. Now, S-100 is considered a dinosaur.

Another possible choice was the IBM PC which has an open system with slots, but I haven't seen enough information on designing and building your own cards — and I've been told that it is a real bitch to interface to.

Most of the other personal computers are designed strictly for games, wordprocessing, spreadsheets, or accounting, and don't have the required I/O capabilities or bus access.

A Proposed System

My first inclination was to use the S-100 bus, because it provides the features I need, and I have two systems. There are several problems. The first problem is that very few of the real world interface cards I need are still being produced, and support for the existing ones will probably be dropped in the near future. If I were to work with the S-100 bus, I'd have to plan on designing and wire wrapping all the boards for the system. The second problem is that I'd have to work on everything alone because not many people are interested in S-100 control applications.

My second choice was the IBM-PC bus. This has the advantages of a large number of users and a lot of software, but most of the units are already overloaded with cards and the operating system developments are not headed in the direction I want to go.

I was looking for a industrial style bus instead of a micro computer style bus, which could be used from various micro computers and still be system independent. I found the answer when I saw the AMPRO SCSI/IOP® which interfaces the STD bus to a SCSI port. This allows people to use any computer with a SCSI interface — adapters are available for Apple II, IBM-PC, S-100, Digital PDP-11, LSI-11, MicroPDP-11, VAX and MicroVAX, AT&T 3B2, VMEbus, Multibus, and others, and almost all new computers will have the SCSI interface built-in. Now different workers can use their choice of host system, and still work with the same SCSI or STD bus peripherals.

I'll be using the AMPRO 8-bit and 16-bit Little Boards for the primary units because they incorporate the SCSI protocol, and their PROTO boards give me buffered address, data, and control lines. These boards are designed for industrial use, and will work with their upcoming 16-bit CMOS board and the future 32-bit board.

I chose the STD bus because the A/D, I/O, Motor Control, CPU, and many other boards are available off-the-shelf at reasonable prices because it is a mature technology which has seen widespread industrial use. While the STD bus was designed for 8-bits, they have revised it using multiplexed lines for 16-bit operation. There's a HD64180 board, and even an 8088 CPU with MS-DOS Ver. 3.1 in ROM.

Using AMPRO's SCSI/PLUS® implementation, I can hang a number of 8-bit and 16-bit (32-bit in the future) boards on the SCSI bus as master/slave units, and use SDT bus cards for the I/O functions. Other workers can use their favorite computer with a SCSI host adapter to control the STD bus and whatever other devices they want to hang on the SCSI interface.

I'm aware that much of the advanced work is now being done on the Intel Multibus II and the Motorola VMEbus plus several new buses, but these 'state-of-the-art' boards are currently out of my price range. A large number of companies are entering this market, and as the prices drop because of the competition we'll be ready for them because they'll interface to SCSI.

For reasonably priced — but still very powerful — projects in automation, robotics, measurement, and control, the combination of AMPRO Little Boards, the SCSI interface, and the STD bus will be very hard to beat.

I'll be working on assembling a system, including C and assembler programming tools, ROMable code, and even small independent units using special microcontroller chips. Some of the things we'll cover include multitasking programming, communications, interfacing, motor and device control, optimum adaptive control, and sensors. We'll be very heavily involved in both hardware and programming. Your input and comments are welcome by mail, phone, or modem (see page one for our modem number). ■

****

Bibliography

IEEE Spectrum, "A Framework for Computer Design", October, 1986, pg 49, W. Kenneth Dawson, etal.

Electronic Engineering Times, January 13, 1986, page 29.

Electronic Component News, December, 1985, page 9.

Electronic Engineering Times, "EIS's Home Electronic Bus", July 23, 1986, page 35.

Using the SB180 bReal Time Clock

by Kenneth C. Turner

Shortly after I read Steve Ciarcia's description of the SB180 in the September and October, 1985 BYTEs, my faithful, but rather marginal home-brew Z-80 system died its final lingering death. Discretion seemed the more economical part of valor at this point, so I sent my money to The Micromint of Vernon, CT, and was soon the pleased possessor of my very own SB180. It's a great machine, and learning to use ZCPR3 has been rather like exploring some great Victorian mansion, with miles of maze-like passageways and hidden rooms.

Naturally, when one has acquired a faster toy, the first thing one does is to race it: how fast is it, really? And there's a real time clock on the Hitachi 64180 CPU chip, so it ought to be easy to do all the benchmarks, right? And so it is, but you have to delve a bit to find out the actual recipes. This article is a cookbook; I will tell all — or, at least all that's necessary to tell what time it is, and how fast it's going. You can use these routines to record time of day on your output, and to optimize your code for speed. I will discuss Assembler approaches, but the emphasis will be on BASIC — in particular, on Microsoft BASIC, since this is probably the most widely used BASIC on CP/M systems.

HD64180 Timer Architecture

Before we get to the recipes, we have to look a bit at the nature of the ingredients — in this case, how the on-chip timer looks to the software. The HD64180 has a number of internal devices, and they are accesed as internal I/O ports. These are treated in much the same way as external I/O ports, but with special I/O instructions: IN0 and OUT0, instead of IN and OUT, for example. All of the on-chip devices have their pre-defined locations in "Internal I/O Port Space." In particular, the Programmable Reload Timer, as Hitachi calls it, occupies internal port locations 0CH to 10H and 14H to 17H. Table I identifies the function of each of these addresses.

All registers are cleared to zero on system reset. Each bit of the control register determines some timer function. These functions are described in Table II. Bits are numbered 0 to 7, with 0 being the least significant bit.

When a timer channel is enabled, that channel's data register counts down from whatever value it contains. One count is subtracted for every 20 cycles of the system clock. When it reaches zero, the Interrupt flag for that channel is set to 1, the data register is reloaded with the number in the reload register for that channel, and the countdown continues from that value. It is important to note that the interrupt flag will stay at one forever after the first countdown to zero, until it is cleared by either a reset or a read of BOTH the control register AND either the high or low byte of that channel's data register. That means that the elapsed time will have an uncertainty of some number times (Reload register value) × 20 clock cycles. (This is 213⅓ msec for the 6.144 Mhz clock of the SB180, if one loads the registers with FFFF hex.) Measures must be taken to avoid this uncertainty if we wish to time longer durations. Usually interrupt subroutines are used to do this bookkeeping, but we will not deal with interrupts in this article, except in passing.

The general procedure for using this sort of timing system is as follows:

1) Define a control word. The details of this process will be discussed in the section on BASIC routines.
2) Load a suitable value into the Reload Register.
3) If desired, write a suitable value into the Data Register. (Note that if you don't do this, the first countdown may be wrong.)
4) Write the control word to the Control Register.
5) Stand back and enjoy the results.

TABLE I Internal Timer Register Addresses

HEX ADDRESS	FUNCTION
ØC	Low Byte, Channel Ø Data Register
ØD	High Byte, Channel Ø Data Register
ØE	Low Byte, Channel Ø Reload Register
ØF	High Byte, Channel Ø Reload Register
1Ø	Timer Control Register, Both Channels
14	Low Byte, Channel 1 Data Register
15	High Byte, Channel 1 Data Register
16	Low Byte, Channel 1 Reload Register
17	High Byte, Channel 1 Reload Register

TABLE II Timer Control Register Bit Functions

BIT NUMBER	FUNCTION
7	Channel 1 Interrupt Flag: This is set to 1 when Channel 1 Data Register decrements to Ø. It is reset to Ø when BOTH the Timer Control Register AND EITHER the High or Low bytes of the Channel 1 Data Register are read. This bit is read only.
6	Exactly like bit 7, but for Channel Ø.
5	Channel 1 Interrupt Enable: If this bit is set to 1, then when bit 7 flips to 1, an interrupt is generated.
4	Exactly like bit 5, but for Channel Ø.
3 and 2	These 2 bits control the multiplexing of an output signal with address line 18. Since this address line is used to select the ROM on the SB18Ø, you'd better leave them at Ø.
1	Channel 1 Enable: When this bit is 1, the Channel 1 Data Register counts down from whatever it happens to contain, at one count for every 2Ø clock cycles.
Ø	Exactly like bit 1, but for Channel Ø.

TABLE III TIME DATA AREA IN BIOS 2.1

ADDRESS (HEX)	DATA		
EAFC	Tenths of seconds	\	Real time clock
EAFD	Seconds	:	Starts when you
EAFE	Minutes	:	reset the system.
EAFF	Hours	/	Counts up.
EBØØ	Seconds	\	Wall Clock. You
EBØ1	Minutes	:	can set this with
EBØ2	Hours	/	the TIME S command.
EBØ3	Tenths of seconds	\	Count down timer.
EBØ4	Tens of seconds	/	Cycles from 255,255.
EBØ5	Motor Time (used by Floppy Disk I/O Routines)		

The Data Register can be read at any time, without stopping the countdown, if you read the Low byte first. This causes the timer to store the High byte in a secret internal place, and give it back to you when you ask for the High byte. Note that if you read the high byte first, two things can happen. First of all, you will get the value that was stored when the Low byte was last read. Secondly, even if this is the first read, the 'real' High byte may have changed (from overflow of the Low byte) by the time the Low byte is read. You have to stop the timer in order to write to the Data Register.

How the Micromint BIOS handles the Timer

The BIOS supplied by MicroMint, Inc., for the SB180 uses Channel 0 to provide an interrupt driven real-time clock service for the system. A ten byte area at the top of the BIOS routines is kept updated for system time. Table III contains a map of this area for my system, which is the 2.1 version of MicroMint BIOS. If you have another version, you may have to use a different base address than EAFC (hexidecimal base) in your routines. I will tell you how to find the correct one below. You can find out what version of BIOS you have by looking at the file BIOS.Z80 on the source files disk in your distribution material.

In any case, the first four bytes of the timer area contain hours, minutes, seconds, and tenths of seconds from the time you last pushed the RESET button. The next three bytes contain hours, minutes and seconds, and are settable with the TIME S command. (The file TIME.COM is in area 15 of the system distribution disk.) The next two bytes are a countdown timer, which continuously recycles from 255,255 (decimal). The low order byte is in units of tenths of seconds, and the high order byte is in units of tens of seconds. The last byte is the motor timer, and is used by the floppy disk I/O routines. I won't say anything more about that one.

Access to the timer area may be gained in a number of ways. Under normal circumstances, the page address of BIOS will be found in the third byte of memory. Put that in the H register, put 36(hex) in the L register, and you have the address of the jump to the timer routine. A call to that address will return the address of the timer data area in the HL register. If that sounds a bit confusing, see Figure 1 for some Assembler Code which does what I've just described.

I apologize for this ugly stuff — it will work, and it's understandable, but it is, of course, a mortal sin to write self-modifying code.

If you want an absolute address, which is more convenient for BASIC programs, life gets a little more complicated. Here are a couple of recipes for getting an absolute address for the timer data area:

1) Assemble the file BIOS.Z80 with the output listing option. Then add the address of the bottom of the BIOS to the address of RTC (at the very end of the listing). The address of the bottom of BIOS is the value in bytes 1 and 2 of memory −3.

2) Use ZDMH or DDT or another debugger, and prowl around as follows:

You type: D0000 000F

The debugger responds:

```
0000 C3 03 DA 80 70 C3 00 BA FF 40 40 FF FF 40 40 FF
....P....@@..@@.
```

The first 3 bytes (after the 4 digit address1) are a jump instruction to DA03, the warm boot jump in BIOS. We add 33H to that to get DA36H. So next we enter: DDA30 DA3F

It says:

```
DA30 C3 87 DE C3 97 DB C3 93 DB 00 00 00 00 00 00 00 ..............
```

This is the top part of the BIOS jump vector. We want the jump at DA36, which is to DB93. So next we say: DDB90 DB9F

We then see:

```
DB90 00 04 00 21 FC EA C9 3A 5A DA B7 C8 21 40 DA 01
...!...:Z...!@..
```

The code at DB93 is what we want; it says :

```
    LD  HL,0EAFCH
    RET
```

The address EAFC (hex) is our goal. It is the absolute address of the timer data area.

Using the Real Time Clock From a BASIC Program

Now that we've done our homework, we can move along to reading the timer from a BASIC program. Once we've set the time with the TIME S command (before we enter BASIC), we can read it as shown in Figure 2.

Here we have simply copied the timer data area into a set of integer variables and printed them out. With this, we have access to the time of day, the elapsed time since we last reset the machine, and an interval timer in tenths of a second. Of course, for many timing problems, tenth of a second accuracy is not enough. We now turn to a more accurate technique.

We will use the unused timer channel, channel 1, to obtain the ultimate precision the timer permits. This is a time step of twenty system clock cycles. For the 6.144 MHz SB180, that amounts to a 'tick' of 3.25520833 microseconds. 1536 of these ticks equal 5

Figure 1

```
                    (ZAS Assembler Mnemonics)

            LD    H,(02)             ; Get BIOS page
            LD    L,36H              ; This is the offset
                                     ; HL now contains the address of the
                                     ; jump to the timer routine
            LD    (JUMP+1),HL        ; store in the call instruction
     JUMP:  CALL  JUMP               ; On return, HL will point to the timer
                                     ; data area.
```

Figure 2

```
100 REM  BASIC SEGMENT TO READ THE TIMER DATA AREA
110 DIM B%(8)
120 REM B(3-0) = REALTIME CLOCK HR,MIN,SEC,TENTHS
130 REM B(6-4) = WALL CLOCK HR,MIN,SEC
140 REM B(8-7) = COUNTDOWN TIMER IN TENS OF SECONDS
150 REM             AND TENTHS OF SECONDS
160 TIMER% = &HEAFC          ' (this address valid for BIOS 2.1)
170 FOR I = 0 TO 8
180 B%(I) = PEEK(TIMER%)
190 TIMER% = TIMER% + 1
200 NEXT I
210 PRINT B%(3);B%(2);B%(1);B%(0)
220 PRINT B%(6);B%(5);B%(4)
230 PRINT B%(8);B%(7)
240 END
```

milliseconds. (One second is 307200 ticks.) We will use a very simple hand assembled machine language routine to do this from our BASIC program.

First we will put FFFF in the data and reload registers, to avoid any funny business with 2 zeros in a row appearing in the countdown. (This would happen if the reload register were left in its zero initial state. After counting down to zero, another zero would be loaded into the register, and the next tick would give FFFF (hex).) We then establish a suitable control word to enable the channel 0 timer, but without any interrupts. Then we are ready to time things. We read the data register, run the code we want to time, read the data register again and subtract. The difference (before minus after) is the time the code we are timing took to run, in units of 3.26etc. microseconds.

Next we make up the control word. The warm boot routine writes 00010001 (binary) to the timer control word. That enables channel 0 and the channel 0 interrupt. We wish to enable channel 1, but not the channel 1 interrupt. Therefore we simply set bit 1 to 1 and write 00010011 (binary) as our control word. The code we need for timer initialization is shown in Figure 3.

In order to read the timer data, we need to understand a bit of how our BASIC handles passed parameters. This is different in each BASIC implementation, and I will only discuss the Microsoft BASIC USRn routines. When we make the call I% = USRn(J%), a call is made to the location set by the DEF USRn = nnnn command, and the HL register points to the low byte of I%. This makes our task very easy. The code is shown in Figure 4.

Now all we have to do is find a place to put our code and put it there. If we had a lot of code, we would want to put it above the basic area, but for just a little, like this, we can adopt a simpler procedure. Microsoft BASIC has a built-in function called VARPTR, which returns the address of a variable. We shall use this to store our machine language routines in an integer array. Because BASIC moves it's arrays around as new variables are defined, we must locate our actual address each time we wish to use the timer. In the following code, the fast timer is used to time the code that reads the system timer data area. In my system that time is slightly under 50 msec. The overhead required by the fast timer code itself (measured by deleting lines 240-280) is 5.3 msec. See Figure 5.

Figure 3

```
3E FF       LD     A,0FFH          ; FOR RELOAD
ED 39 14    OUT0   (014H),A        ; INITIALIZE THE DATA REGISTER
ED 39 15    OUT0   (015H),A        ; BOTH BYTES
ED 39 16    OUT0   (016H),A        ; INITIALIZE THE RELOAD REGISTER
ED 39 17    OUT0   (017H),A        ; BOTH BYTES
3E 13       LD     A,013H          ; THIS IS THE CONTROL BYTE
ED 39 10    OUT0   (010H),A        ; STICK IT IN TO ENABLE THE CHANNEL
C9          RET                    ; AND RETURN
```

Figure 4

```
ED 38 14    IN0    A,(014H)        ; PUT LOW DATA BYTE IN A
77          LD     (HL),A          ; PUT A WHERE HL POINTS TO
23          INC    HL              ; NOW HL POINTS TO HIGH BYTE
ED 38 15    IN0    A,(015H)        ; PUT HIGH DATA BYTE IN A
77          LD     (HL),A          ; PUT A WHERE HL POINTS TO
2B          DEC    HL              ; RESTORE HL SO IT POINTS WHERE
C9          RET                    ; IT STARTED, AND WE ARE DONE
```

Figure 5

```
100 DATA &HFF3E,&H39ED,&HED14,&H1539,&H39ED      :REM Machine language for
110 DATA &HED16,&H1739,&H133E,&H39ED,&HC910      :REM timer routines.  Note
120 DATA &H38ED,&H7714,&HED23,&H1538             :REM byte reversal to make
130 DATA &H2B77,&H00C9                           :REM the order correct.

140 DIM A%(15)                                   ' Integer array for code
150 DIM B%(8)                                    ' Integer array for timer data

160 FOR I = 0 TO 15                              ' Read the routines
170 READ A%(I)                                   ' into the array.
180 NEXT I

190 DEF USR0=VARPTR(A%(0))                       ' Initialize routine address.
200 I = USR0(I)                                  ' Initialize the fast timer.
```

Listing 5 continued

```
210 J% = 0                                      ' DEFINE THIS FIRST!
220 DEF USR1=VARPTR(A%(10))                     ' Read timer routine address.
230 J% = USR1(I%)                               ' Read the fast timer.

240 START% = &HEAFC                             ' Address of timer area.
250 FOR I = 0 TO 8                              ' Copy timer data into
260 B%(I) = PEEK(START%)                        ' data array.
270 START%= START% + 1                          ' Update timer data address.
280 NEXT I

290 K% = 0                                      ' DEFINE THIS FIRST!
300 DEF USR1=VARPTR(A%(10))                     ' Read timer routine address.
310 K% = USR1(I%)                               ' Read timer again.

320 PRINT (J% - K%)/307200.                     ' Elapsed time in seconds.
330 REM B(3-0) = REALTIME CLOCK HR,MIN,SEC,TENTHS
340 REM B(6-4) = WALL CLOCK HR,MIN,SEC
350 REM B(8-7) = COUNTDOWN TIMER IN TENS OF SECONDS
360 REM AND TENTHS OF SECONDS
370 PRINT B%(3);B%(2);B%(1);B%(0)               ' Print out Timer data.
380 PRINT B%(6);B%(5);B%(4)
390 PRINT B%(8);B%(7)
400 END
```

The routines sketched above should provide easy access to the on-chip timer of the HD64180. I hope you enjoy using them. ■

EVERYTHING YOU NEED... $279.00

Now it's easy to program the Heath-Zenith HERO-1® Robot with an Apple® II. HERO® Macros for the S-C Software 6800 Cross Assembler program in Heath's Robot Interpreter Language with easily remembered mnemonics. The HERO® Macros come with 30 pages of documentation.

[For example, the line: 1130 > MVWRIM GRIP, OPEN, 60, FAST instructs the HERO® to open his gripper 60 units at fast speed. Motor position is expressed in base-10.]

Transfer to HERO® with ROBI... an affordable interface for the robotics experimenter... is simple.

- **ROBI is a complete package. No additional hardware required for Apple® or HERO®.**
- **ROBI installs quickly in an Apple® II, II+, or IIe. Once installed, no hardware changes are needed. Within minutes, you will be programming HERO®.**
- **With ROBI and the Cross Assembler, the programmer uses Apple®'s memory to write the program, and HERO®'s memory to run the program.**
- **Not "copy protected," archival copies may be made as needed.**
- **ROBI offers expansion potential.**

VISA and MasterCard accepted.

The Cross Assembler with HERO® Macros sells for $100.00; the ROBI Interface sells for $199.00. Both as a package—$279.00.

To order, or for more information, call (303) 670-6137.

BERSEARCH
Information Services

26160 Edelweiss Circle
Evergreen, Colorado 80439

APPLE® is a trademark of Apple Computer. HERO® is a trademark of Heath Electronics

The SCSI Interface
Software for the SCSI Adapter
by Rick Lehrbaum, Vice President Engineering, AMPRO

Introduction
Assuming you have built (or purchased) the SCSI Adapter described in issue #25, you probably would like to make it run. That's the purpose of this issue's SCSI Interface article.

We'll look at software for the SCSI Adapter from two perspectives:

(1) Simple bidirectional I/O, using the adapter as a parallel port.
(2) A SCSI compatible I/O driver.

This article won't cover how to create a complete CP/M (or Z-System) "BIOS" — sorry, that is beyond the scope of this series on SCSI! What we will cover is how to operate the 5380 SCSI controller, for both SCSI and non-SCSI applications.

Inside the NCR 5380
The first requirement in using the SCSI Adapter, in either SCSI or non-SCSI applications, is to know how to use the 5380 IC to manipulate the SCSI interface. The 5380 device is really quite simple to use, and has a number of interesting features, including:

- 17 bidirectional I/O lines with 48 mA current sink.
- Support for automatic handshaking.
- Support for SCSI bus arbitration.
- Compatibility with the ANSC SCSI specification.

The 5380 has 16 internal registers, normally addressed on the SCSI Adapter as shown in Figure 1. These registers are directly read or written by the Z80 CPU using the chip select to the 5380 (pin 21) under programmed I/O control.

Note that the addresses shown in Figure 1 are based on the default board base address of 20H. You can easily reconfigure the board to a different base address by changing the connections between the address decoder outputs (U5A and U5B) and the chip select logic (U6).

What follows is a brief description of the function of each of the 5380's internal registers. Remember that the I/O addresses indicated are based on the default board base address of 20H. In the 5380 register description, neither DMA nor interrupts will be discussed. If you wish to implement either of those functions, obtain and study a copy of the 5380 Design Manual, available from either NCR or AMPRO (for $10).

One other comment, before we look at the registers, is that all of the SCSI bus signals are "active low," which means that their logic levels are opposite to the contents of the corresponding bits in the 5380 registers.

Current SCSI Data Register (20H): | Reading this 5380 register allows you to observe the state of the eight SCSI interface data lines. All you have to do is read the I/O port at 20H, and the value obtained represents the state of the SCSI bus data lines, DB0 through DB7, except that the bus lines are inverted relative to the contents of this register. This register's bits are assigned as follows:

Bit 7	6	5	4	3	2	1	Bit 0
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Output Data Register (20H): Writing to this register in the 5380 sets the state of the SCSI bus data lines (DB0 through DB7), providing that the "Assert Data Bus" bit of the Initiator Command Register is set. If you write to this register when the Assert Data Bus bit is not set, the register will hold your data but not assert it on the SCSI bus until the Assert Data Bus bit (in the Initiator Command Register) is set at a later time. The Output Data Register data bits are assigned as follows:

Bit 7	6	5	4	3	2	1	Bit 0
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Input Data Register (26H): Reading this register returns the latched — not current — state of the SCSI data lines. Data is latched either during a DMA Target Receive operation when ACK (pin 14) goes active, or during a DMA Initiator Receive when REQ (pin 20) goes active. The DMA Mode bit in the Mode Register (22H) must be set before data can be latched in this register. This register may also be read under DMA control using the 5380's DMA control lines. The contents of this register are:

Bit 7	6	5	4	3	2	1	Bit 0
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Initiator Command Register (21H): This is a read/write register which is primarily used to control the 5380's SCSI bus interface when the chip is in the Initiator role. Most functions are also available in the Target role. Two of the bits of this register have different uses when the register is read or written, so two charts are given. These are as follows:

Read Usage:

Bit 7	6	5	4	3	2	1	Bit 0
Assert RST	Arb in Prog	Lost Arb	Assert ACK	Assert BSY	Assert SEL	Assert ATN	Assert Data

Write Usage:

Bit 7	6	5	4	3	2	1	Bit 0
Assert RST	Test Mode	Diff En	Assert ACK	Assert BSY	Assert SEL	Assert ATN	Assert Data

Readable Registers	Address
Current SCSI Data	20H
Initiator Command Register	21H
Mode Register	22H
Target Command Register	23H
Current SCSI Bus Status	24H
Bus and Status Register	25H
Input Data Register	26H
Reset Parity/Interrupt	27H

Writable Registers	Address
Output Data Register	20H
Initiator Command Register	21H
Mode Register	22H
Target Command Register	23H
Select Enable Register	24H
Start DMA Send	25H
Start DMA Target Receive	26H
Start DMA Initiator Receive	27H

Figure 1

As you have probably guessed, this register allows you to control the state of the RST, ACK, BSY, SEL, and ATN bus signals, and also to control whether the 5380 places its data on the SCSI bus or not. Notice that bits 6 and 5 differ according to whether you are reading or writing this register. (Refer to the 5380 Design Manual for details on the use of these bits.)

Here are two restrictions in using these bits to control the SCSI bus:

(1) The 5380 must be in Initiator Mode (Mode Register, bit 6) to be able to set the SCSI control bits ACK and ATN active on the SCSI bus.

(2) If the 5380 is in Initiator Mode (Mode Register, bit 6), then the data bus will not be asserted by the Assert Data Bus bit (Bit 0) unless the SCSI bus I/O signal is false (output from Initiator) and the SCSI bus control signals C/D, I/O, and MSG all match the contents of the Assert bits in the Target Command Register.

Mode Register (22H): This register contains many control signals governing operation of the 5380. It allows you to place the chip in either Initiator or Target mode, and provides control over DMA and arbitration functions, parity, etc.

Bit 7	6	5	4	3	2	1	Bit 0
Block Mode DMA	Target Mode	Enable Parity Check	Enable Parity Int	Enable EOP Int	Monitor BSY	DMA Mode	Arbi- trate

We won't go into the use of the bits regarding DMA, parity, and interrupts, as these are not required for basic operation of the SCSI interface. For our purposes, Bit 6 is the most interesting bit of this register, as it determines whether the 5380 is in its Target Mode or if it is in Initiator Mode. Bit 0 starts the chip arbitrating if you use that option.

Target Command Register (23H): This register provides control over the bus phase control bits, REQ, MSG, C/D, and I/O, and is similar in this respect to the Initiator Command Register.

Bit 7	6	5	4	3	2	1	Bit 0
				Assert: REQ	Assert: MSG	Assert: C/D	Assert: I/O

These bits can only be asserted by the 5380 if the "Target Mode" bit in the Mode Register is set. In Initiator mode, these bits have a a different purpose. In Initiator Mode, the states of the Assert MSG, Assert C/D, and Assert I/O bits must match the actual state of the bus (which can be read in the Current SCSI Bus Status Register), for data to be placed on the SCSI bus even if the Assert Data Bus bit of the Initiator Command Register is set. Also, in Initiator Mode, if the Assert MSG, C/D, and I/O bits do match the bus state, then the "Phase Match" bit in the Bus and Status Register will be set.

Current SCSI Bus Status Register (24H): This read-only register allows you to directly read the state of 8 signals on the SCSI bus. The bits are utilized as follows:

Bit 7	6	5	4	3	2	1	Bit 0
RST	BSY	REQ	MSG	C/D	I/O	SEL	DBP

Select Enable Register (24H): This write-only register is used as mask in Target Mode operation to allow the 5380's built-in selection response logic to generate an interrupt. Refer to the 5380 Design Manual for more info.

Bit 7	6	5	4	3	2	1	Bit 0
DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

Bus and Status Register (25H): This read-only register allows you to read two SCSI bus signals — ATN and ACK — which are not included in the Current SCSI Bus Status Register. In addition, six 5380 status flags which are associated with the optional use of interrupts are read through this register. The bits of this register are utilized as follows:

Bit 7	6	5	4	3	2	1	Bit 0
End of DMA	DMA Request	Parity Error	Inter- rupt Request	Phase Match	Busy Error	ATN	ATN

As mentioned above, the use of DMA and interrupts will not be covered in this article. The "Phase Match" bit is handy, in that it shows in a single bit whether the SCSI bus phase matches the settings of the Assert bits (MSG, C/D, and I/O) in the Target Command Register. The Phase Match bit is only meaningful when the 5380 is in its Initiator Mode ("Target Mode" bit = 0).

The Busy Error bit is set if the Monitor Busy bit in the Mode Register has been set and if the SCSI bus BSY signal becomes false. If this occurs, the 5380 output drivers all become disabled.

DMA Registers (25-27H): These are not really registers at all. A write of one of these I/O addresses is used as a trigger to start DMA in one of three DMA modes (Send, Target Receive, or Initiator Receive). Refer to the 5380 Design Manual for more information on the use of DMA.

Reset Parity Interrupt (27H): Like the "DMA Registers," this is not really a register either. A read of this address is used as a trigger to clear a parity error interrupt.

Simple Bidirectional I/O

Have you noticed that the 5380 is really a glorified Parallel I/O device (PIO), with a few special SCSI-related features? So let's use the 5380 as a PIO. In this example, the bus we create will have the following signals:

Signal	Purpose	SCSI Signal Used
data	8 bidirectional I/O lines	DB0-DB7
address	A0-2 (8 ports)	REQ,MSB,C/D
control	Read/Write	I/O
	Data Transfer Strobe	SEL

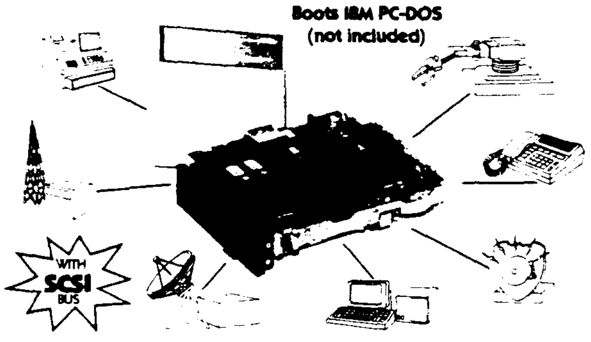
This is a very basic configuration, and is suited for many simple interface applications, for example the direct connection to a UART or similar device. Many other possibilities exist!

SCSI ENGINES

Little Board/186™....$495

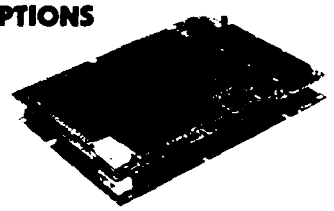
High Performance, Low Cost PC-DOS Engine

Boots IBM PC-DOS (not included)

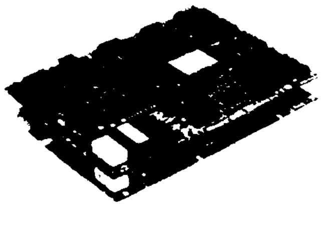


- Three times the COMPUTING POWER of a PC
- Data and File Compatible with IBM PC, runs "MS-DOS generic" programs
- 8 MHz 80186 CPU, DMA, Counter/Timers, 128/512K RAM zero wait states, 16-128K EPROM
- Mini/Micro Floppy Controller (1-4 Drives, Single/Double Density, 1-2 sided, 40/80 track)
- 2 RS232C Serial Ports (50 -38,400 baud), 1 Centronics Printer Port
- Only 5.75 x 7.75 inches, mounts directly to a 5-1/4" disk drive
- Power Requirement: +5VDC at 1.25A; +12VDC at .05A; On board -12V converter
- SCSI/PLUS™ multi-master I/O expansion bus
- Software Included:
  - PC-DOS compatible ROM-BIOS boots DOS 2.x and 3.x
  - Hard Disk support

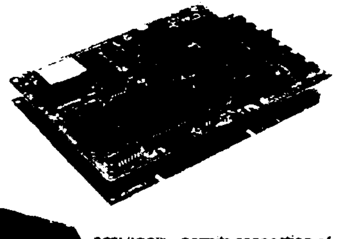
OPTIONS



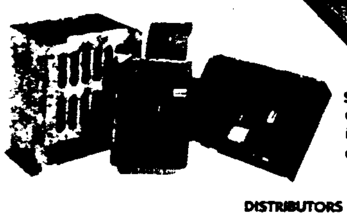
PROJECT BOARD/186™ - adds 25 square inches of wire wrap prototype area with buffered and pre-decoded 80186 bus interface for Little Board/186



EXPANSION/186™ - adds five key options to Little Board/186
- 512K RAM
- 8087 co-processor
- Battery-backed Real Time Clock
- 2 RS232/422 sync/async serial ports
- I/O expansion bus



VIDEO RAM EMULATOR™ -allows use of software that writes to display controller "VIDEO RAM"



SCSI/IOP™ - permits connection of off-the-shelf STD bus industrial I/O interfaces (analog, digital, serial, display, power control, etc.)

DISTRIBUTORS

ARGENTINA: FACTORIAL, S.A., 41-0018 TLX 22408 **AUSTRALIA:** ASP MICROCOMPUTERS, (613) 500-0628, TLX 36587 **BELGIUM:** CENTRE ELECTRONIQUE LEMPEREUR, (041) 23-45-41, TLX 42621 **BRAZIL:** COMPULEADER COMPUTADORES LTDA., (41) 262-1939, TLX 416132 **CANADA:** TRI-M, (604) 438-9012 **DENMARK:** DANBIT, (03) 66 20 20, TLX 43558 **UK:** AMBAR SYSTEMS LTD., 0296 35511, TLX 837427 **FINLAND:** SYMMETRIC OY, 358-0-585-322, TLX 121394 **FRANCE:** EGAL PLUS, (1) 4502-1800, TLX 620893 **ISRAEL:** ALPHA TERMINALS, LTD., (03) 49-16-95, TLX 341667 **SWEDEN:** AB AKTA, (08) 54-20-20, TLX 13702 **USA:** CONTACT AMPRO COMPUTERS INC,.

IBM®, IBM Corp.; 80186®, Intel, Corp,.

AMPRO
COMPUTERS, INCORPORATED

67 East Evelyn Ave, • Mountain View, CA 94041 • (415) 962-0230
TELEX 4940302 • FAX (415) 962-1042

In this example we will use the 5380 in Target Mode, which turns out to allow easier manipulation of the device for non-SCSI bidirectional I/O.

Let us assume you are interfacing to a UART which has ports defined as follows:

```
UART Port A:
00          Read        Status Register
00          Write       Control Register
01          Read        Data Register
01          Write       Data Register

UART Port B:
02          Read        Status Register
02          Write       Control Register
03          Read        Data Register
03          Write       Data Register
```

And assume that the Status Registers of each channel of the UART are used in this manner:

```
  Bit 7      6       5       4       3       2       1      Bit 0
+-------+-------+-------+-------+-------+-------+-------+-------+
:       :       :       :       :       :       :  RDA  :  TBE  :
+-------+-------+-------+-------+-------+-------+-------+-------+
```

Where "RDA" means "Receive Data Available" and TBE means "Transmit Buffer Empty". The other 6 bits have some purpose, of course, but we'll keep this example simple!

Firstly, the following code might be used to setup the 5380 for use in Target mode. It is used to clear all register bits to zero except the Target Mode bit in the Mode Register.

```
INIT5380:       ;Initialize the 5380 to Target mode for ease of use
     MVI A,0
     OUT TCR    ;Clear Target Command Register
     OUT ICR    ;Clear Initiator Command Register
     MVI A,40H
     OUT MR     ;Initialize the Mode Register to Target Mode
     RET
```

Next, if our mission is to read characters from the UART, we could use the following code:

```
GETCHAR:            ;loop until character available
     MVI   A,0   ;address of port A status register
     ORI   08    ;operation will be a read, so REQ = 1
     MVI   A,4   ;set the SEL bit in the Initiator Command Register
     OUT   ICR   ;  to assert the Data Transfer Strobe
     IN    CSD   ;read the data bus to see if the RDA bit is set
     ANI   02
     JZ    GETCHAR    ;loop until data available

     XRA   A     ;clear the Data Transfer Strobe
     OUT   ICR
     MVI   A,1   ;address of port A data register
     ORI   08    ;operation will be a read
     OUT   ICR
     IN    CSD
     RET
```

Isn't this a simple example! But it should be enough to give you the general idea how easy it is to use the 5380 for simple bidirectional I/O. You can probably improve on this in many ways, especially if the device you wish to talk to has a different arrangement of control signals, address modes, etc.

Oh yes, one more comment: the above code wasn't actually tested with a real device, so don't take a hatchet to your hardware if it doesn't work the first time! The routines above were only provided as an illustration. It's your project — you do the programming!

A Sample SCSI Driver

Figure 2 provides a sample SCSI driver. This particular driver is somewhat simplified, and is intended to give you the general idea of how to operate the SCSI interface in general, and the 5380 SCSI IC in specific. Among the features not included are:

- Bus arbitration.
- Disconnect/reselect.
- "Pseudo-dma."
- Interrupts.
- Hangup protection.
- SCSI message system.

Figure 2

```
        * * * * * * * * * * * * * * * * * * * * * *
        *           Sample SCSI Driver             *
        *                                          *
        *     COPYRIGHT (C) 1984,1985,1986         *
        *         AMPRO COMPUTERS, INC.            *
        *          All rights reserved.            *
        * * * * * * * * * * * * * * * * * * * * * *

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                         *
*                     *** NOTICE ***                      *
*                                                         *
*       This sample driver contains an example            *
*       of how to utilize an ncr 5380 scsi controller     *
*       used to communicate with scsi target devices.     *
*                                                         *
*       This driver is provided for non-commercial,       *
*       educational purposes only, and may not be         *
*       used for commercial purposes in full or in        *
*       part without express written license from         *
*       AMPRO Computers, Incorporated.                    *
*                                                         *
*       This driver is provided without warranty of       *
*       any kind, either expressed or implied.            *
*                                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

; ************ MISCELLANEOUS EQUATES AND MASKS ****************
;
ncrbase equ 20h                 ; base address of ncr 5380
;
; 5380 input-only, and input/output registers follow:
;
ncrcsd  equ ncrbase+0           ; current scsi data register
ncricr  equ ncrbase+1           ; initiator command register
ncrmr   equ ncrbase+2           ; mode register
ncrtcr  equ ncrbase+3           ; target command register
ncrcsbs equ ncrbase+4           ; current scsi bus status
ncrbsr  equ ncrbase+5           ; bus & status register
ncridr  equ ncrbase+6           ; input data register
ncrrpi  equ ncrbase+7           ; reset parity/interrupt
ncrdack equ ncrbase+8           ; dack pseudo-dma
;
;
; 5380 output-only registers follow:
;
ncrodr  equ ncrbase+0           ; output data register
ncrser  equ ncrbase+4           ; select enable register
ncrsds  equ ncrbase+5           ; start dma send
ncrsdtr equ ncrbase+6           ; start dma target receive
ncrsdir equ ncrbase+7           ; start dma initiator receive
;
;
```

```
; The following are flag masks for the Current SCSI Bus Status
;   register, at port 24h
;
ncrrst  equ 10000000b
ncrbsy  equ 01000000b
ncrreq  equ 00100000b
ncrmsg  equ 00010000b
ncrcd   equ 00001000b
ncrio   equ 00000100b
ncrsel  equ 00000010b
ncrdbp  equ 00000001b
;
;
; The following is a flag mask for the Bus and Status
;   register, at port 25h
;
ncrphm  equ 00001000b
;
; Storage locations
;
     target    ds   1   ; storage location for target ID
     datptr    ds   2   ; contains address of start of data buffer
     cmdptr    ds   2   ; contains address of start of command buffer
     status    ds   1   ; storage location for storage byte
     message   ds   1   ; storage location for message byte
;
; ****************************************************************


; ************ POWERUP OR RESET 5380 INITIALIZATION ***********
;
; This routine should be used on system reset to clear all
; internal control bits in the 5380
;
ncrinit:
  xra a
  out ncricr
  out ncrmr
  out ncrtcr
  out ncrser
  ret
;
; ****************************************************************


; ************ OPTIONAL SCSI BUS RESET ************************
;
; This routine is used when a SCSI bus reset is required,
; which is not a bad idea to do after system reset
;
reset:
  mvi a,10000000b
  out ncricr
  ;reset pulse of 25 microsec min needed
```

```
    mvi a,20
rstl:
    dcr a
    jnz rstl
  mvi a,00000000b
  out ncricr
delay:
  lxi d,0
rstlop:
  dcx d
  mov a,d
  ora e
  jnz rstlop
    in ncrrpi            ; reset interrupt indicator
    ret
;
; *****************************************************************


; ************  ENTRY POINT FOR SCSI DEVICE ACCESS *************
;
; Begin SCSI access by selecting the target device
;
select:
  xra a                  ; set all Assert bits in Target
  out ncrtcr             ;   Command Register to 0.  This is
                         ;   in preparation for phase logic
  lda target             ; get target ID from memory
  out ncrodr             ;   and put it in data reg
  mvi a,00000101b        ; set Assert Data Bus and SEL bits
  out ncricr
  waitblp:               ; wait for Target to respond with bsy
    in ncrcsbs           ; get current scsi bus status
    ani ncrbsy           ; look at the bsy line
    jz waitblp           ; loop till busy active
  xra a                  ; Target is selected, so clear sel
  out ncricr             ;   and release data bus
  jmp phase              ; then see what the Target will do
;
; *****************************************************************


; ************  MASTER BUS PHASE PROCESSING ROUTINE ************
;
; Now wait for the target device to respond, then see what
; phase it is requesting.
;
phase:
  xra a                  ;reset NCR ctrl registers
  out ncrmr
  out ncricr
  ; The Target will indicate what needs to be done
  ; by the values of MSG, C/D, and I/O when it sets
  ; REQ active.  So, wait for REQ to become active.
```

```
  ; Watch for busy loss and exit if this occurs.  That
  ; means the Target is finished with the transaction.
phl:
  in ncrcsbs             ; check for BSY active
  mov b,a
  ani ncrbsy
  rz                     ; if busy drops, get out
    mov a,b
    ani ncrreq           ; is REQ active?
    jz phl               ; wait for REQ active
      mov a,b            ; update MSG, C/D, I/O in 5380
      ani 00011100b      ; check current phase
      rar
      rar
      ani 0fh
      out ncrtcr
      mov a,b
      ani 00011100b
      mov e,a
      mvi d,0
      lxi h,pjtab        ; get ready to jump to applicable
      dad d              ;    routine
      pchl
;
; This jump table is used to jump to a routine for each
;   possible SCSI bus phase
;
pjtab:
  jmp phase0
  nop
  jmp phase1
  nop
  jmp phase2
  nop
  jmp phase3
  nop
  jmp phase4
  nop
  jmp phase5
  nop
  jmp phase6
  nop
  jmp phase7
;
; For each bus phase, some preparation is done and then
; a jump is made to a read or write routine
;
phase0:         ; data out phase
  lhld datptr   ; set data transfer pointer to data block
  jmp wscsi     ; use scsi write routine
;
phase1:         ; data in phase
  lhld datptr   ; set data transfer pointer to data block
  jmp rscsi     ; use scsi read routine
```

```
;
phase2:          ; command out phase
  lhld cmdptr    ; set data transfer pointer to command block
  jmp wscsi      ; use scsi write routine
;
phase3:          ; status in phase
  lxi h,status   ; set data transfer pointer to status block
  jmp rscsi      ; use scsi read routine
;
phase7:          ; message in phase
  lxi h,message  ; set data transfer pointer to message block
  jmp rscsi      ; use generalized read routine
;
; currently unused phases
phase4:
phase5:
phase6:
  ret
;
; ****************************************************************


; ************* GENERALIZED SCSI WRITE ROUTINE *****************
;
wscsi:
  mvi a,1                ; assert data bus
  out ncricr
wscsil:                  ; wait for req, keep an eye on
                         ;   phase and bsy
  in ncrbsr              ; check for phase mismatch
    ani ncrphm
    jz phase             ; exit when phase changes
  in ncrcsbs             ; check status of bsy
    ani ncrbsy
    jz phase             ; exit if bsy goes away
  in ncrcsbs             ; check for req
    ani ncrreq
    jz wscsil            ; loop until req or phase change
                         ;   or loss of bsy
    ; fetch the data and give it to the 5380
    mov a,m              ; put the data in the output data
    out ncrodr           ;   register
    inx h
    mvi a,00010000b      ; set ack to show target the data
    out ncricr           ;   is available
waitreq:
    ; wait for req to go away
      in ncrcsbs
      ani ncrreq
      jnz waitreq
    xra a                ; when req occurs, clear ack to
    out ncricr           ;   show the target we have the data
    jmp wscsil           ; send more data until phase changes
                         ;   or bsy goes away
;
; ****************************************************************
```

```
; ************* GENERALIZED SCSI READ ROUTINE *****************

;
; generalized scsi read routine
;
rscsi:
  ; wait for req, keep an eye on phase and bsy
  in ncrbsr              ; check for phase mismatch
    ani ncrphm
    jz phase             ; exit when phase changes
  in ncrcsbs             ; check status of bsy
    ani ncrbsy
    jz phase             ; exit if bsy goes away
  in ncrcsbs             ; check for req
    ani ncrreq
    jz rscsi             ; loop until req or phase change
                         ;   or loss of bsy
    ; read the data, store it
    in ncrcsd
    mov m,a
    inx h
    mvi a,00010000b      ; set ack
    out ncricr
    noreq:               ; wait for req to go away
      in ncrcsbs
      ani ncrreq
      jnz noreq
    xra a                ; clear ack
    out ncricr
    jmp rscsi            ; get more data until new phase or
                         ;   or loss of busy
;
; ****************************************************************
```

You would probably want to add some or all of these features in an actual driver. As they say, "These enhancements will be left as exercises for the student!"

The sample SCSI driver is structured as follows:

Miscellaneous Equates and Masks: The 5380 register port addresses, along with bit masks used by the code for testing and setting various flags and conditions.

Powerup or Reset 5380 Initialization: This routine should be used to initialize the 5380 on powerup or reset. It disables all outputs from the chip, so that the SCSI bus is in a free state, in preparation for later usage.

Optional SCSI Bus Reset: This routine can be used if you want to reset the SCSI bus. Use it with some care, as many SCSI disk and tape controllers go through lengthy reset routines when they detect an SCSI bus reset. Some even respond in ways that act like error conditions!

Entry Point for SCSI Device Access: This routine is the main entry point of the SCSI driver.

Master Bus Phase Processing Routine: This is the "heart" — or perhaps more properly the "brains" — of the SCSI driver. This section of the driver operates like a state machine, regulating the operation of SCSI interface.

Generalized SCSI Write Routine: When data is transferred for any reason (Command, Data Out) to a Target it is handled by this routine. Here is where the REQ/ACK handshaking is performed. During the process, the code watches for either a change in bus phase (indicating the current data transfer is complete) or a loss of bus busy (indicating the current SCSI bus operation is complete).

Generalized SCSI Read Routine: Same as write routine, except used for all transfers of data from a Target (Status, Data In, Message In).

How it Works

The SCSI driver is a unique piece of code in that it functions like a software "state machine."

Before you call the driver (at label "select:"), you must:

(1) Create a command parameter block (CPB) somewhere in memory.

(2) Store a pointer to your CPB in location "cmdptr."

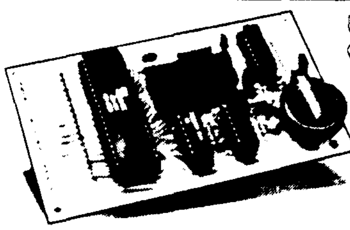
(3) Store a pointer to your data buffer in location "datptr."

(4) If the operation will be a block write, move the block of data to be written into the data buffer pointed to by "datptr."

Typically, the data buffer is 512 bytes, but this depends on the type of SCSI Target device you will be using. The command buffer must be whatever size your command parameter block requires, which is dictated by the command set of the specific SCSI Target device you are going to talk to. It is usually 6 bytes, but can be longer.

Once the driver is called, the SCSI interface is no longer being controlled by your driver, but is actually responding to the signals generated by the SCSI Target itself, as indicated by the state of MSG, C/D, I/O, REQ, and BSY. The listing provided in Figure 2 contains lots of comments which should explain how the 5380 is being used. Read the driver over several times before you give up trying to understand it!

When the driver returns to the calling program, the data passed by the command is located in the data buffer at the location pointed to by "datptr," and the status byte from the command is in the location called "status." You can determine whether the operation was successful or not by examining the status byte. Congratulations — you are now a SCSI user! ■



Ztime-I

CALENDAR/CLOCK

$69 KIT

NOW WITH FILE DATE STAMPING!

- Works with any Z-80 based computer.
- Currently being used in Ampro, Kaypro 2, 4 & 10, Morrow, Northstar, Osborne, Xerox, Zorba and many other computers.
- Piggybacks in Z80 socket.
- Uses National MM58167 clock chip, as featured in May '82 Byte.
- Battery backup keeps time with CPU power off!
- Optional software is available for file date stamping, screen time displays, etc.
- Specify computer type when ordering.
- Packages available:

Fully assembled and tested	$99.
Complete kit	$69.
Bare board and software	$29.
UPS ground shipping	$ 3.

MASTERCARD, VISA, PERSONAL CHECKS, MONEY ORDERS & C.O.D.'s ACCEPTED.

N.Y. STATE RESIDENTS ADD 8% SALES TAX

KCT KENMORE COMPUTER TECHNOLOGIES

P.O. Box 635, Kenmore, New York 14217 (716) 877-0617

AMPRO COMPUTERS INCORPORATED

AUTHORIZED DEALER

BEAR ELECTRONICS

SAVE 10% on AMPRO

INTRODUCTORY SALE THROUGH DECEMBER

ALL AMPRO PRODUCTS READY FOR IMMEDIATE SHIPMENT

SOME EXAMPLES: (Other AMPRO products available at similar savings.)

MODEL	DESCRIPTION	REG. PRICE	DISCOUNT PRICE
1B-1	Little Board SBC	249.00	224.10
1B-2	Little Board/PLUS SBC	289.00	260.10
2A-2	Little Board/186 SBC	495.00	445.50
2A-P	Proto Board	179.00	161.10
2VR	Video RAM Emulator	195.00	175.50
3A-1	STD Bus SCSI I/O Board	119.00	107.10
122A	CP/M System w/2 Drives	995.00	895.50
222	PC-DOS System w/2 Drives	1295.00	1165.50
321	Expandable PC-DOS System	1395.00	1255.50

Please add $3.00 ($8.00 on systems) for UPS shipping & handling. California residents add appropriate sales tax.

VISA MasterCard

CHECKS PO's

Some restrictions apply to credit and large quantity orders. Call for information. Allow 2 - 3 weeks for 122A, 222 or 321 systems.

CALL NOW - CALL COLLECT (415) 376-0125

BEAR ELECTRONICS, P.O. Box 61, Moraga, CA 94556

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; our customers are overwhelmed at the amount of software they receive when buying our products. For example, the Z-Com product comes with approximately 80 utility programs; and our TERM III communications package runs to a full megabyte of files. This is real value for your software dollar.

ZCPR3
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 users.

Multiple Commands per Line
You can easily use multiple commands per line under ZCPR3. Simply separate the individual commands with semicolons. For example, "PIP B:=A:*.TXT;STAT B:*.*" will copy files and then show you the STAT results.

User-Programmed menu systems
ZCPR3 comes with three different menu systems that you can use to create custom menu-driven "front ends" for your computer. This is especially useful for setting up menus for your spouse or co-workers to use the computer, as they never have to see the A> prompt. All they have to do is press a single key to run any single or multiple CP/M programs, and when the task is done, control is automatically returned to the menu (ordinary CP/M menu programs cannot do this).

Extended Command Processing
When you type a command under CP/M, it will only look for the program in the current drive and user area. ZCPR3 gives you more flexibility by additionally searching other disks and user areas when resolving commands. You have full control of this function, called the PATH. This is probably the one element of ZCPR3 that is missed most if you return to "ordinary" CP/M.

Also, ZCPR3 supports the capability of grouping all your commonly used utility programs into a library file (*.LBR). This is great for systems with a small number of directory entries per disk, as the library file only uses one entry. It also has the advantage of reducing disk space requirements for a given set of programs, allowing you to put more programs on a disk. And the programs in the library file are invokable from the command line just like any other program not in the library.

Other Features
There's much more to ZCPR3, like named directories, online help system, etc., but it can't be described on one page. If you would like more information, consider the books shown below.

Z-SYSTEM
Perhaps the only shortcoming of ZCPR3 is that it is not a complete replacement for CP/M. This is what the Z-System does. The Z-System contains ZCPR3 and an additional module, ZRDOS, and is a complete replacement for CP/M. 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 $44 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. 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. At present, only Kaypro computers have this option available.

BOOKS
We sometimes joke around the office that we are really in the business of publishing, not selling software. We have books. Lots of books. We have to have lots of books, considering how powerful our software is and the large quantity of different packages we offer. Here are our best sellers:

ZCPR3: The Manual
This is the "bible" for the ZCPR3 user. An exhaustive technical reference, bound softcover, 350 pages. Contains descriptions of each ZCPR3 utility program, a detailed discussion about the innards of ZCPR3, and a full installation manual for those doing their own installation. You could order it from B. Dalton, but why? Get it from us.

The Z-System User's Guide
For those who are not technically inclined. This is an excellent tutorial-style manual filled with examples of how to use the power of ZCPR3/Z-System most effectively, written by two highly experienced Z users. (One user is a lawyer, the other a writer; this proves that anyone can use Z and benefit from it.)

ZCPR3: The Libraries
The extensive documentation for the libraries of ZCPR3, SYSLIB, Z3LIB, and VLIB. A must for any serious user of these programming tools. Loose-leaf notebook style; easy to work with as it will lay flat on your desk.

THERE'S MORE
We couldn't fit all Echelon has to offer on a single page (you see how small this type is). We haven't begun to talk about the many additional software packages and publications we offer. Send in the order form below and just check the "Requesting Literature" box for more information.

Item	Name	Price
1	ZCPR3 Core Installation Package	$44.00 (3 disks)*
2	ZCPR3 Utilities Package	$89.00 (9 disks)
3	Z3-Dot-Com (Auto-Install ZCPR3)	$99.00 (6 disks)*
4	Z3-Dot-Com "Bare Minimum"	$49.95 (1 disk)
5	Z-Com (Auto-Install Z-System)	$119.00 (7 disks)*
6	Z-Com "Bare Minimum"	$69.95 (2 disks)
12	PUBLIC ZRDOS Plus (by itself)	$59.50 (1 disk)
13	Kaypro Z-System Bootable Disk	$69.95 (3 disks)
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	$150.00 (4 disks)
25	DSD-80 Full Screen Debugger	$129.95 (1 disk)
27	The Libraries. SYSLIB, Z3LIB, and VLIB	$69.00 (8 disks)
28	Graphics and Windows Libraries	$49.00 (1 disk)
29	**Special –** Items 27, 28, and 82	$129.00 (9 disks)
40	Input/Output Recorder IOP (I/OR)	$39.95 (1 disk)
41	Background Printer IOP (BPrinter)	$39.95 (1 disk)
42	Programmable Key IOP (PKey)	$39.95 (1 disk)
43	**Special –** Items 40 through 42	$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)
81	**ZCPR3: The Manual** bound, 350 pages	$19.95
82	**ZCPR3: The Libraries** 310 pages	$29.95
83	Z-NEWS Newsletter, 1 yr subsciption	$24.00
84	**ZCPR3 and IOPs** 50 pages	$9.95
85	**ZRDOS Programmers'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)

NAME ____________________
ADDRESS ____________________
____
TELEPHONE __________ DISK FORMAT __________
☐ REQUESTING LITERATURE

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.

ITEM	PRICE
Subtotal	
Sales Tax	
Shipping/Handling	
Total	

(Continued from page 3)

Response to Letters in #25

In response to D.E.'s letter on CP/M and the 6502, there are many reasons why CP/M has not found a home on the 6502 processors. The first consideration is the fact that most systems using the 6502 microprocessor are low end consumer machines. These machines, owing to their cost, make no effort to adhere to any industry standards other than their own. The mere mention of the 6502 processor places the machine in the, "under $500 market," category.

CP/M s a disk based and disk handling program. What format will the 6502 based system use for its disks? The purpose, other than just to do it, to implement a CP/M compatible system would be to access the wealth of software available for this system. Just for the sake of discussion, let's assume we chose a popular disk format. Most existing 6502 systems use serial, or other odd way of communicating with the disk, and its controller. This is slow, defeating the purpose of disk storage, though it may be faster than a tape storage system. The one nice thing about the 6502 is that it uses direct memory accessing for I/O functions. It would not be difficult to add a disk controller chip. The next thing we need to do is replace the existing system's monitor ROM. A "monitor" is similar to a BIOS. It handles all system input and output. We now have the cost of the disk controller, the new ROM, and can add two disk drives, their enclosure, and power supply. I doubt the existing system could provide enough power to run disk drives. They also require a 12 volt power source. At this point we require more than the original cost of the system to interface with a technology essentially obsolete in the consumer market.

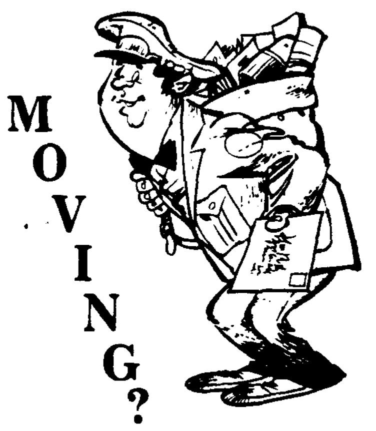
A CP/M type operating system could be written for the 6502. But to what purpose? Where is the market? Where is the interest in such a system? Economically we should think of such a project design as a new design, not an add-on. A consumer system would compete with the giants Commodore and Atari. The system would require a specialized market. The real death of the idea of a 6502 based CP/M compatible system is one of primary economics, not really a technological one. While the 6502 is still the fastest 8 bit processor in a relative setting, the processor would be spending most of its time translating 8080 and Z80 opcodes into something it could understand.

Systems such as the APPLE allow the internal bus to be captured by an external processor. In this case all that is required is to write a BIOS capable of accessing the internal device handling structure. The APPLE systems use single sided disks. Have you tried to locate single sided disk drives lately?

The grass is always greener in the other technologies. I have seen very nice CP/M systems selling for $200, lacking only a terminal to be put on line. The 6502 was king in its day, and now has a limited use. It is in the same condition as CP/M. There is little logic in application of one obsolete technology upon one even more obsolete. There are many sentimental reasons for doing so, but few practical ones. The primary concept is the reality that there just isn't enough interest in these technologies. This is why the various emulators for 6502 machines are not available any more — no one is interested in them on a scale to make the expense and effort profitable.Progress is seldom welcome. It often means we have to abandon old friends. There is much truth in the concept of, "Once you understand something it is obsolete."

Editor's note: There have been CP/M cards for the Apple II on the market, but I'm not sure how long they will continue to be available. BCE (1-800-545-7447) is liquidating the Appli-Card with CP/M 2.2, 6MHz Z80B processor, 70 column dispaly without an 80 column card (or 80 columns with an 80 column card), and including Wordstar Professional for only $79! This requires programs on special Apple CP/M format disks, or programs downloaded from BBSs. I have an adapter card which runs standard CP/M 8 inch drives on my Apple II+, but I no longer use it. Apple IIs used to be one of the largest CP/M markets, but I think that it is dead commercially. It's a good deal for someone who still uses their Apple, so get a card from BCE while they still have some left (I think I'll get one just in case I want to use my Apple for a control project).

As for J.T.'s S-100 Hard Disk problems, the reason the formatting utilities and BIOS will allow you a maximum of eight megabytes per logical drive is based in neither of these areas. CP/M is not capable of handling more than eight megabytes per drive. ZRDOS, and other



Make certain that TCJ follows you to your new address. Send both old and new address along with your expiration number that appears on your mailing label to:

THE COMPUTER JOURNAL
190 Sullivan Crossroad
Columbia Falls, MT 59912

Please allow six weeks notice. Thanks.

Registered Trademarks

It is easy to get in the habit of using company trademarks as generic terms, but these registered trademarks are the property of the respective companies. It is important to acknowledge these trademarks as their property to avoid their losing the rights and the term becoming public property. The following frequently used marks are acknowledged, and we apologize for any we have overlooked.

Apple II, II +, IIc, IIe, Macintosch, DOS 3.3, ProDOS; Apple Computer Company. CP/M, DDT, ASM, STAT, PIP; Digital Research. MBASIC; Microsoft. Wordstar; MicroPro International Corp. IBM-PC, XT, and AT; IBM Corporation. Z-80, Zilog. MT-BASIC, Softaid, Inc. Turbo Pascal, Borland International.

Where these terms (and others) are used in The Computer Journal they are acknowledged to be the property of the respective companies even if not specifically mentioned in each occurence.

CP/M compatible operating systems follow this convention. They must or they will not be CP/M compatible. There is no way to break the eight megabyte limit without altering the operating system's DOS segment. When CP/M was written there didn't seem to be a need for more than eight megabytes per physical drive. The BIOS defeats this barrier by dividing up the the physical drive into smaller eight megabyte segments. For similar reasons, having to do with a 16 bit representation of disk drives, physical or logical, a CP/M compatible system can theoretically support only 128 megabytes. In reality, however, the practical limit is around 80 megabytes. For each drive, logical or physical, the BIOS requires a disk parameter block and storage area for the drive's directory. This can use up TPA memory very fast. You should have your programmer make sure that additional drive support in the way of BIOS directory storage and disk parameter block space is available in your BIOS. If you have gotten this far on your own, then search all programs for drive limitation checks. In many systems you are limited to four drives, logical or physical, due to the BIOS storage and support concepts we just discussed. These drive checking code segments will have to be defeated before the additional drive support can be accessed.

In answer to P.H., to "port" software from one machine to another is not always an easy task. Even if they have the same operating system every computer is different. In some cases one just connects a cable from one output port to the other system's input port transferring the program as if by modem. This process deals with different disk formats well. In many cases it is the only way to translate different disk formats. Once you have the program written for one machine on the other, minor changes in the code must be done. Some systems use different codes for clearing the screen, moving the cursor and so on. Think of it as an immigration problem. The alien software must be adapted to a new, and bigoted society.

One feat of magic in the computer world are "tristate devices." As might be inferred, this refers to "three states." These three states regard the output lines of a device. Everything in, what I call the "MicroWorld," is connected to a bus structure. A bus structure is a common freeway type of affair where data travels. The first two states are, "on," and "off." These states represent the binary "high," or +5 volt state, and the "low" or near +0 volt states respectively. Now then, if these were the only states a device could have then each device would require its own bus, or there would have to be some other way to coordinate traffic. All roads begin and end, after all, at the CPU. This "other" way implements the "third state." In the third state the outputs of a device are disconnected from the common bus structure. The CPU can then "shut up" a device by setting a pin on the concerned device, allowing another device to speak to it over the common microworld highway system. This concept directly applies to memory devices. Were it not available we would, in theory, write a byte of data to one memory cell, or address, and that value would be written into every cell in the device. One nice thing about the computer world is that any concept, in order to be valid, must be universally applicable. If we can use an additional line to make a single output, device full of outputs, disappear from the bus structure, then we can make groups of devices disappear — magic.

In "banked" memory systems there may be 128K of memory, but only 16 address lines. In these systems two sets of identical memory chips are installed. A ROM may also be installed. Using a separate, nonaddress line, or one of the upper address lines, we can make one set of memory devices "disappear" and another set "appear" in the same address space. This is sort of like multiple realities, and as with the concept of multiple realities, travel can be tricky. Generally what is done is to reserve a portion of memory which is common to all sets of memory devices, called "banks" to serve as a common point while the switching of banks takes place. The schemes for jumping from one reality to another are many and varied. The concept we need to look at is that one set of memory devices is replaced with another in the same memory space, or reality. Each bank may contain different programs and data.

So, banks of memory share the same space, but are considered virtual memory, within the addressing capabilities of the microprocessor. I am not real fond of bank switching. If the system is to support 512K why don't they just add more address lines instead of trying to be cute? Most often it is done to save adding pins to a chip. Heaven forbid that chip technology might violate a standard of any kind, especially that of a two cent socket into which a $12 chip would be placed. Actually, there are many reasons for bank switching beyond expanding the addressing limits of a given microprocessor. Justifying them is another thing. Anyway, virtual memory is always there and totally accessible by the microprocessor, banked memory systems share virtual memory space at different times.

The term "interleave," on a hard disk is the same concept as the term, "skew," on a floppy disk drive. It is a common, though not universal, practice not to place related sectors consecutively upon

FREE CATALOG AND SOFTWARE APPLICATIONS GUIDE

CP/M MSDOS **AFFORDABLE ENGINEERING SOFTWARE** TRSDOS PCDOS

ANALYSIS

XFER $72.95
[illegible]

LOCIPRO $72.95
[illegible]

SPP $72.95
[illegible]

STAP $72.95
[illegible]

CIRCUIT DESIGN

ACTFIL $72.95
[illegible]

ACNAP $72.95
[illegible]

DCNAP $72.95
[illegible]

GRAPHICS

PDP $72.95
[illegible]

PCPLOT $72.95
[illegible]

PLOTPRO $72.95
[illegible]

TEKCALC $72.95
[illegible]

MATHEMATICS

TEKCALC $72.95
[illegible]

MATRIX MAGIC $72.95
[illegible]

COMCALC $72.95
[illegible]

REPORT WRITING

Right Writer $97.95
[illegible]

BV Engineering
Professional Software
2200 Business Way, Suite 207
Riverside, CA 92501 USA
(714) 781-0252

the disk's surface. Instead, consecutive blocks of data, which should be adjacent, are separated by a number of sectors on the disk. This constant number of sectors is a portion of a given system's format. The purpose of skipping a few sectors on the disk is to improve disk drive performance. Whether we are speaking of a hard drive, or a floppy drive, the medium spins beneath the read/write heads at a high rate of speed. If we demanded that each sector of data were laid down in a perfectly consecutive manner disk access to the data may be slowed. If the disk controller were not ready to write the data to disk exactly when the proper place spun around it would have to wait a complete revolution for the exact spot on the media to come around again. By skipping a constant number of sectors an "access window" is created, easing some of these timing requirements. If this access window is large enough to account for disk controller set up times delays waiting for the proper medium position to arrive under the heads can be reduced, if not eliminated. This results in faster reading, and writing of data.

You are quite correct in offering the thought that part of the MS-DOS/PC-DOS operating system resides both on disk and in ROM. Many CP/M systems have their BIOS in ROM. Kaypro is one such system. The AMPRO machines do not have their BIOS in ROM, which is why they are so easy to reconfigure. The "DOS" system has three portions of the operating system on disk, as well as the directories, configured in the same manner as normal files. There is the DOS segment, the input/output system, and the command processor on the disk. The BIOS is in ROM. While the contents of the ROM can be altered to suit your fancy, this is an expensive process. Those systems without the BIOS in ROM may be altered and written back to the disk. Anyone can boot a disk, but few are prepared to "burn" new EPROMs which would also require opening the computer's case and replacing the chip(s). Those system which do not have their BIOS in ROM are the ones preferred by all but the casual user.

Bit mapped graphics may be worked by accessing the special RAM used by the video system. The manner for accessing this section of memory varies with the system. Check your system's technical manual for clues as to the process required with your system. I am using an AMPRO PC-DOS system, which is terminal based, so I cannot currently help you on this score even though I have a great deal of data on the subject. I'd prefer to test out concepts such as these rather than offering an opinion.

I agree with you that a beginner's column, or at least a forum should be had in TCJ. Often the editor has a different viewpoint. Your questions thus far are not in the least rhetorical. Other readers may find information presented here of use, which is why I sat down to write these responses. Secondly, reader participation in a magazine is to the benefit of all readers. You may never know until you ask, or until some other reader asks.

C. Thomas Hilton

Your editor is eager to hear whats on your mind: about the magazine, about what your working on, tips or troubles you may have. Write us: Letters, The Computer Journal, 190 Sullivan, Columbia Falls, MT 59912, or post on our bulletin board (406) 752-1038.

Inside AMPRO Computers

by Peter Ruber

As a departure from my regularly scheduled Ampro column, Publisher Art Carlson has given me carte blanche to write a feature for the Computer Journal about my recent visit to Ampro Computers. I made the 3000-mile journey to Mountain View, California, this past August, because I wanted to meet the people I had talked to so often by telephone for nearly a year.

It was an interesting experience. The atmosphere at Ampro was informal and open. No closed doors or secretaries running interference, or artificial protocol to contend with. I went where I wanted to, talked to whoever had a few free moments. Sometimes I just watched and listened. You often learn more that way.

The principal purpose of this trip was to gather background information on how the company got started, how the SCSI/PLUS® interface bus evolved, how the Little Boards are produced, and where Ampro was heading in the next few years. Were they committed to the LITTLE BOARD concept? Or were they planning boards of different dimensions because the larger architecture required by enhanced 16-bit and 32-bit systems would dwarf the 5.75" × 7.75" footprint of the LITTLE BOARD?

Having worked with so many different computers during the last six years — from tinker toys to IBM compatible systems — I find myself becoming more involved with the Ampro Little Boards than any of my other computers because they are such versatile products. I like the fact that all I/O functions are included on each single board computer and that they have an open architecture to interface most any peripheral I might need in the years ahead, including the burgeoning CD ROM and Optical Storage technology we will see in volume production within the next year or two. My PC system will require a special interface to handle these devices, but my Ampro Little Boards already have that interface.

Most computer manufacturers design their systems with the same planned obsolescence as do the automobile makers in Detroit. The typical life-span of a computer product is less than three years; and during the second year of its life, a new model rolls off the assembly line to ease the withdrawal pains of the older model.

A good measure of this marketing philosophy is brought about by the investment community who pressure computer manufacturers to go for the "big kill" — profits and earnings per share — rather than a more restrained approach of smaller earnings but consistent growth.

"That's where the Japanese have it over us," William I. Dollar, Ampro's president told me. "They plan on surviving over the long haul. When the large American high-tech firms run into problems, or become impatient during a static growth period, they start buying up other companies. And, when they become too large and unwieldly, they spin off their subsidiaries. It's like a game of checkers. It lacks the subtlety of chess. The real losers are the employees, because they're always being shuttled in and out of jobs.

"American ingenuity created the Integrated Circuit and this spawned a vast industry. The Japanese then took our technology, refined it, and started making chips cheaper than we could. There's nothing wrong with being preoccupied with new levels of technology, but if you don't make a concerted effort to refine your manufacturing techniques to lower your production costs, or apply the same ingenuity toward creating new products, the manufacturers in the Orient will."

Bill Dollar seems irritated when he talks about the opportunities American computer manufacturers have overlooked. "We could easily be the world's leader in chip manufacturing. Now most of them are imported, and we cry 'foul'. The same goes for PC board manufacturing. A very large percentage of PC board production is shuffled to Singapore."

"Isn't that where the Little Boards are produced?" I asked.

"Quite true. When we first went into production, early in 1984, our boards were manufactured right here in the Valley. But the firm we were dealing with only specialized in short production runs; and when we outgrew their capabilities, they referred us to a firm in Singapore that they used for large volume orders. This company still produces the pre-production runs of new boards that we use for BETA testing, and which are also sent to some of our high-volume customers for evaluation. After that, everything comes from Singapore. The company in Singapore stamps the boards out like cookie-cutters, and their quality control is so good that it is a rare occasion when we have to reject a board that fails our own testing."

"Is there a chance that you will transfer board production back to the U.S.?"

"That's a very strong possibility. There's a company in Idaho that grew very large, in a relatively short period of time, manufacturing keyboards for the IBM-PC Jr. When IBM dropped this product, their employment slid from over 850 to about 300 in a matter of weeks. Many other firms who had been manufacturing components for various IBM computer products can relate similar horror stories. At any rate, they had a survival instinct, and redesigned their facilities to mass-produce computer boards by using auto-insertion equipment and wave soldering. We've sent them our specs and have received quotes that are cheaper than our factory in Singapore. Our next board may very well be produced in Idaho. It would be to our advantage, too, if something goes wrong. One of our engineers could hop a plane and be there in a couple of hours. Singapore is 9,000 miles away. It's expensive to send someone there. On top of that, we lose a key staff member for a week."

The Little Boards

Ampro's design philosophy appears committed to the LITTLE BOARD concept of having all the features of a desk top system on a board that mounts directly to a floppy or hard disk drive. Similarly, their SCSI/PLUS interface is the key ingredient of compatibility for all their products.

There are now nine LITTLE BOARD microprocessor based and compatible enhancement products. The bulk of these have appeared during the past year. While each new generation of boards grows in sophistication, they will either interface with previous products to expand a user's system (or to enhance his I/O capabilities), or to upgrade that system without having to replace any peripherals.

Two key product categories seem to be Ampro's current stars. The first is the SCSI/IOP. It wasn't designed to just interface with the Little Boards. It can be used with any computer system that requires an intelligent I/O processor. It will add Real-Time capabilities to any computer system having an SCSI interface. It plugs into any STD bus backplane and can control 8-bit STD bus I/O boards such as analog-to-digital converters, video display controllers, speech synthesizers, network interfaces, etc. It effectively replaces STD bus CPU functions.

But its applications don't end there. It will also open up the architecture of such closed computer systems as the MacIntosh and the Amiga to simplify the addition of hard disks and controllers. It contains a 4MHz (or an optional 6MHz) Z80® CPU, 8 byte-wide memory sockets for up to 64K of EPROM/RAM memory. Included is a Z80-family counter/timer controller (CTC) option, a battery backed real time clock, and a battery backed RAM option. It is fully ANSC X3T9.2 SCSI compatible for target, initiator and arbitration capabilities, and will permit up to 8 host computers and SCSI/IOP's to share resources. The price/feature ratio is amazingly low for a product of this kind.

The second, and most important, product is the LITTLE BOARD/186, a complete PC-DOS® computer that has three times the computing power of a PC. It is data and file compatible with an IBM PC® , running most generic MS-DOS® programs. The BIOS has been optimized so that PC-DOS 3.0 or higher is required to boot the system.

It is powered by an 8 MHz 80186® CPU which incorporates all DMA and Counter/Timer functions usually performed by support chips in a PC compatible system. The basic board contains 512K RAM (with zero wait states) and 16K-128K EPROM. A Mini/Micro Floppy Controller will service 1-4 drives, which can be Single/Double Density, 1-2 sided and 40- or 80-track. There are also two RS232C Serial Ports (one of which interfaces to most popular ASCII or ANSI terminals) that can operate up to 38.4K baud and a Centronic Printer port. The SCSI multi-master I/O expansion bus creates a simple interface for hard disk drives and tape units; and with Ampro's implementation of Concurrent-DOS® , multi-tasking and multi-user systems are possible at prices lower than most businesses are accustomed to seeing.

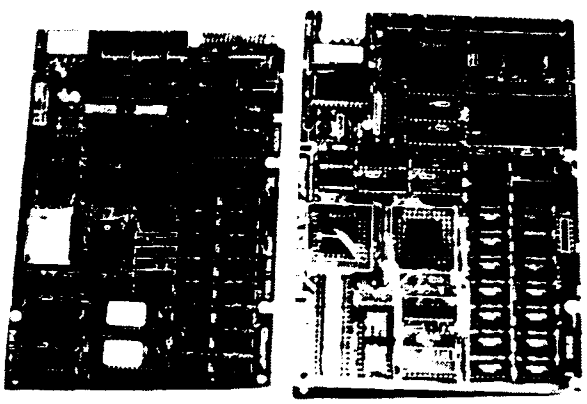
Once the LB/186 was in place, Ampro seemingly went for over-kill. They designed the EXPANSION/186, which sandwiches on top of the LB/186 and provides 5 key expansion options: a buffered I/O bus with 128 I/O locations; two additional RS232/RS422 Sync/Async Serial Ports using the 8530/82530 SCC; an additional 512K bytes of RAM; a battery backed Real Time Clock; and, lastly, the provision for a 10-MHz 8087 Math Coprocessor that offers a 400% speed advantage over an 80287.

Just released, during my visit, was the VIDEO RAM EMULATOR which allows an ASCII terminal to run software which writes directly to the IBM's "Video RAM" rather than using PC-DOS or ROM BIOS function calls. Such software would otherwise require modification for use with the LB/186. It contains on-board "Video RAM", it simulates the IBM's 6845 registers, and emulates the IBM's keyboard port.

The final LB/186 add-on is the PROTO/186, a user-configurable general purpose I/O and memory expansion interface. It provides buffered address, data, and control lines; a large breadboard area where custom circuitry can be added to allow the LB/186 to be used in data acquisition, process control, and SCSI test instrumentation. To simplify the addition of the custom circuitry, the PROTO/186 has 20 bits of latched Address, 16 buffered Data lines, buffered CPU status and control lines, 6 decoded chip selects (using a 16L8 PAL decoder), and 5 programmable inputs for transceiver control.

Ampro also produces completely assembled systems of their Little Boards, complete with floppy, hard disk drives and back-up tape systems, in configurations too numerous to describe in this article.

There is also a growing list of special software that exploits the features of all the Little Boards. Source Code is available for most of it so that system integrators, engineers developing special applications, and knowledgeable users can customize


The Little Board/186 (Left) and the Expansion/186 (Right). The smallest and most powerful PC-DOS computer system available on the market today.

their applications. All boards come with some of the best and most detailed technical manuals available in the industry.

The Origin Of The Little Boards

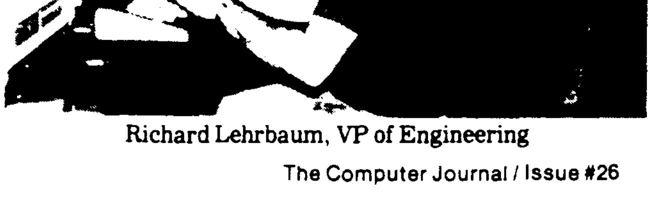
The original product that was eventually to become the first LITTLE BOARD was created by Richard B. Lehrbaum, VP of Engineering. Rick is one of those rare people who can do three things at once without missing a beat.

When I first arrived at Ampro, Rick was working furiously on the final portion of the schematic for the next generation Little Board, about which I will provide a quick over-view later in this article. His free hand was pecking away on a computer creating the support software. In between he fielded numerous telephone calls, conferred with his engineers, ran around following up on their progress, and put the final touches on an innovative new product call the T-BOX, which has the potential to replace every Telex system in the country.

Rick received his Bachelor of Science in Physics from New York University School of Engineering and Science in 1968. From that year until 1978, he taught Physics and Physical Science at Northeast Louisiana University, where he also completed his Master of Science in Physics.

Although he had received a grant from the National Science Foundation and had designed special computers during his academic tenure, he wanted to broaden his experiences in a more creative and competitive environment. The years from 1978 through 1983 were spent working in various engineering capacities at Data General Corp., Advanced Micro Computers, Dynabyte Corp., and Telesensory Systems, where he worked on data acquisition devices, intelligent disk controller boards, multi-user and multi-processor computer systems, disk subsystem design, computer architectural designs, telecommunication systems for the handicapped, and a host of other product categories too numerous to mention.

Since Rick is his own best eloquent spokesman, the following


Richard Lehrbaum, VP of Engineering

conversation was extracted from an hour-long talk he managed to fit into his already busy work schedule.

RL: Did Bill ever tell you where the Little Board came from?

PR: Only briefly, on our way to the office this morning.

RL: It just came as sort of a flash one day in the spring of 1983, while I was working at Telesensory Systems. I had picked up a Vic-20® to fool around with because I was interested in what was developing in the low-cost personal computer market. I saw right away that the Commodore could be used as a terminal, and that it would be real easy to design a compact board with very few components for a low cost CP/M system. I had the idea that the Osborne computers had sold quite a few units, and that users didn't seem to mind jockeying the screen back and forth. I felt that Commodore users also wouldn't mind this window approach, and that all I needed was the right terminal emulation program. I could place my board inside a cheap box and hook it up to the Commodore.

I took my design to Dave Feldman, who was a professional draftsman and head of Telesensory Systems' document control section, and asked him to lay the board out for me and prepare the camera-ready circuit separations. I planned to sell the finished product by mail order out of my garage as sort of a weekend hobby.

By the time we had the working prototype — I think that was in September of '83 — both Dave and I suddenly realized that we had created something much more important than a mere add-on product for the Vic-20. The market was also changing rapidly. The prices of Osbornes and Kaypros had dropped, and the IBM-PC was becoming more and more of an influence. So we decided to make the most of what we had, and make a business quality system we called the "Bookshelf 100." We also decided to attack the industrial and hacker market who wanted to replace their bulky S-100 systems, and make the board available separately for those who wanted to integrate their own system.

What happened then was that we placed ads in the January, '84 issue of NUTS & VOLTS, which was sort of a newsprint type publication for the hacker market; followed by the February issue of MICROSYSTEMS JOURNAL, and the March issue of BYTE. When Microsystems appeared, we could tell where it was being delivered by the phone calls we were getting. It made such a splash — the product was so right for the time, and no one else had done anything remotely like it, offering a complete computer mounted on a disk drive, with CP/M included in the price of the board.

The board had an immediate popularity, but very quickly we were finding out that the imbedded applications market had taken to our product and that we had to get serious about our goals because the hacker market only had a limited appeal.

It was about this time that we met Bill through some mutual business connections, and the three of us seemed to hit it off. Bill had been involved in the sales and marketing end of large computer systems and setting up distribution networks. Dave and I didn't have this kind of expertise, so we incorporated as a full-time business.

As we pursued the computerized applications market, we began to realize that they needed more than just two serial ports, floppy and printer control. There was a limit to the kinds of applications that could be served by these ports. They tend to be keyboard intensive, and without any other ports on it, what can you control? Nothing, unless you start adding other boards onto it.

So, the original Little Board taught us that you needed a way to open the architecture in a way to add additional ports. Some of our customers asked for more memory, but CP/M doesn't recognize more than 64K. Although it's easy to add memory to a CP/M system, it's used mainly as a cache or RAMdisk for storing data that would later be saved to media.

At the same time, the 8-bit processor in our Little Board was ideal for controlling machines, A/D converters and all kinds of laboratory equipment. Suddenly hard disks were becoming cheaper and a lot of people were asking us to provide some kind of interface. Thus, I was presented with a two-fold problem. How do we satisfy the needs of the industrial market to provide an open architecture that would allow them to interface to many different types of equipment? And how do we satisfy the needs of the desktop business user who required a hard disk to store his data. I really didn't want to trade off one for the other.

And then, one day, it hit me. Why not use SCSI (or SASI as it was then being referred to) as a small computer systems interface. Everyone was in the mindset that SASI/SCSI was only a hard disk interface. Most of the world still thinks of it that way. What convinced me otherwise was that SCSI was really a Parallel I/O bus — with bi-directional bits and control lines — so it should be possible to make that into an I/O bus for many different applications; and — oh, by the way — use it as a hard disk interface for typical SASI/SCSI applications.

SCSI/PLUS, at its inception, was two things. On the one hand, it was a philosophy — the architecture; on the other, it was a means for using it for more than just mass storage. That's what the PLUS meant from that perspective.

There was also the flip side of the coin of SCSI/PLUS, which was the idea of expanding the address space of SCSI, because if you planned to add I/O devices to the bus, in addition to hard disk and tape storage devices, you can easily run out of address lines, because SCSI was limited to 8 I/O addresses. This is a pretty small bus. The Z80 has 256 I/O addresses, and people think of that as limited.

SCSI, being an intelligent bus, could have a card on it that could talk to many different devices. So, in fact, 8 isn't as bad as it sounds, because you could have logical units plugged into each of those 8 devices. But I felt it was important to be able to extend that limitation — talk to more than 8 devices, especially when the bus structure allows you to go 18-feet, because it's buffered I/O.

So, I looked at the specs and figured out that it shouldn't really be a problem, and I came up with a scheme to allow 256 addresses on the bus. But looking at the electrical properties, finally limited that to 64.

PR: Have you actually hooked 64 devices to the SCSI/PLUS bus?

RL: Not yet. And once the real implementation is looked at for lots of devices, we may want to limit it to a smaller number. The important thing to remember is what I mentioned earlier. SCSI/PLUS is both an architecture and a philosophy that allows you to think and plan systems in terms of more than 8. I wrote it up as a spec — a proposed extension to the ANSC standard, which would allow a couple of additional bus phases in it.

PR: OK. You have, as you mentioned, created an architecture to expand the ANSC SCSI bus. What is Ampro doing to actually implement the SCSI/PLUS standard beyond 8 devices?

RL: One of the reasons we developed the SCSI/IOP card was to provide a means for accessing external devices other than disks and tapes. It's a target that's under our control. What I think will happen is that the disk and tape guys will not implement SCSI/PLUS unless it's for their own in-house testing. They don't really see, with their limited vision — of disk and tape being the whole story — as to why anyone would want to be able to interface more than 8 devices.

The representative from AT&T — their ANSC committee member — has expressed some interest in our capabilities, because they envision larger systems than the average computer manufacturer, with a larger number of peripherals on SCSI. So, different types of needs have to be placed into the proper perspectives in order for anyone to understand how SCSI, as a mass storage interface, and SCSI/PLUS, as a means to add more than just mass storage relate to real world applications.

PR: Specifically, how do you relate these two concepts?

RL: For the personal computer user, having more than one hard disk and a tape for backup storage is over-kill. For a minicomputer you can imagine getting close to 8 devices. And for something like a super-mini, in an intense computing environment, coupled with the need to control the types of external devices I mentioned earlier, you can see how you may need to have 12 — or 15 — or more devices under your control, each with logical units performing their assigned tasks.

PR: You're referring to real-time clocks, a communications gateway, disk and tapes, floppies, real-time feedback mechanisms, external sensors, burglar alarms, co-processors, and so forth all stuck on SCSI.

RL: Yes. That's precisely the concept we had in mind, and our SCSI/IOP is the ideal product for us to create a mass-device implementation of the SCSI/PLUS bus. We can make the IOP invisible on the bus and not run out of address space. The host computer would be able to talk to SCSI devices — disks and tapes and controllers — and the host will be able to talk to SCSI/PLUS devices. If you picture our 186 PC-DOS board, for instance, as the host, linked to other 186 boards performing multitasking and other computing operations, and our IOP board controlling external devices, you can very easily have 64 devices on the SCSI/PLUS bus — not counting the 8 devices under the SCSI side — without any conflict in address space. SCSI/PLUS devices have their own way of being accessed, and SCSI devices have theirs. So, if the host can talk both of these languages — being bi-lingual — we can have a jumper on our board that can be placed one way to access SCSI devices, and jumpered the other way to talk to only SCSI/PLUS devices. And as long as the host has the drivers to talk to these devices, then our devices don't place a burden on the SCSI address space.

We benefit from the silicon that's been developed, because it only takes one chip — the 5380 — on each device. And that's pretty decent. How many buses do you know that can be established with the kind of protocol support this chip has? So, we've seized on the silicon that's been developed for the purpose of mass storage and have created the first standard Parallel I/O bus for accessing every conceivable I/O device. There aren't any ubiquitous Parallel I/O buses available. The IEEE488 is very limited in ability and very slow. But SCSI/PLUS has the advantage of high speed; it has the advantage of standard silicon; and it has the advantage of flexibility of mixing both mass storage and I/O on the same bus.

PR: Do you get involved to any great extent in providing custom software drivers for your customers?

RL: Sometimes. We prefer that our customers do their own drivers, and that's one of the reasons why we provide source codes. We've actually incorporated a SCSI driver that extends PC-DOS ROM BIOS interrupt structure by having added an interrupt 13 for direct SCSI calls. So customers don't have to worry about the low level protocol, and they can then interface from that through their own programs. Most of the time, we'll only prepare custom drivers for specific hard disk drives. In the SCSI world, even though there is a high degree of standardization, individual devices have their individual characteristics that have to be optimized. We do this kind of work where larger companies may be unwilling to support their customers and usually suggest that their clients hire a consultant.

PR: I know that NCR, who designed the 5380, has now listed Ampro's SCSI/PLUS specifications in the technical literature they provide for this chip. I assume that represents an endorsement for the SCSI/PLUS bus.

RL: Up to a point. I pointed out earlier that we will be bringing up a full implementation of the SCSI/PLUS bus during the next few months through our IOP board. We may find, at that time, that the silicon in the 5380 may not be capable of supporting 64 devices. Maybe it will only support 32, for the sake of a count. We may have to wait for an advanced version of the 5380 with the proper silicon to handle 64 SCSI/PLUS devices.

Another company here in the valley — whose name I shouldn't mention — is thinking of making a 5380 compatible chip, and they're looking at our SCSI/PLUS in order to make the chip special. I think that they, and some other firms, are beginning to realize that the kind of storage and I/O devices we will have available in the near future will benefit or need a universal Parallel I/O bus of this kind. In the past, it used to be quite unruly. Everyone had their own custom interfaces and standards were hard to define. Ultimately everyone capitulated and some standards were created: the Centronics parallel printer interface, the RS232C serial interface, the SASI hard disk interface and the SCSI, which is a superset of the SASI. SCSI/PLUS embraces all these devices, as well as industrial I/O devices, all on one ribbon cable.

I would like to see SCSI/PLUS become the industry standard for all single board computer systems, because it provides for an inexpensive way to open the architecture of otherwise closed systems. It could be on the back of every computer as it will be on every mass storage device. Right now, most single board computers have some kind of expansion connector to interface to I/O boards. But each manufacturer has his own peculiar way of doing this. As such, I/O devices designed for one computer won't necessarily work on any other. What SCSI/PLUS will accomplish, if others follow suit, is to create a standard so that an A/D converter from one manufacturer will work with any single board computer on the market.

PR: I assume, then, that all future Little Boards will have the SCSI/PLUS interface?

RL: Yes. We've had too much of a challenge creating the standard, and we want all of our customers — whether they have an 8-bit Little Board, or our current 16-bit 186 Little Board, or whatever boards we have planned for the future, to be able to still function as part of the systems they have created for their businesses or industrial plants.

I remember, not too long ago, when I called Larry Boucher and told him I had something I though he might be interested in. Larry was the father of the SASI architecture when he was at Shugart. Now he's the Chairman of Adaptec, one of the largest manufacturers of hard disk and tape controllers. He told us to come over. Maybe he had heard of Ampro. Anyway, Bill and I went to see him, and I handed him an overview entitled a "SCSI Opportunity." When he got to the middle of the page, he began to smile. The more he read, the bigger the smile got. Finally, he said, "I like it. Let's do it." He's been real supportive of us ever since. He reviewed the final proposal for SCSI/PLUS, and helped make various industry connections for us.

PR: Aside from Larry Boucher at Adaptec, how much success have you had in generating interest for the SCSI/PLUS architecture?

RL: Surprisingly, a fair amount, especially from those who see the potential of it as a universal I/O bus. But it takes time, and you have to give people the opprtunity to digest the concept. I've attended the ANSC SCSI Forums, and talked about SCSI/PLUS. Typically, it falls on deaf ears at these forums, and committees are conservative by nature. They don't usually react until there is a clamoring of requirements in the market, or until someone shows it will cause a problem if they don't add it. Everyone has their own perspective and dedication to their products — just as we do.

PR: Do you think that, in time, ANSC will adopt the SCSI/PLUS architecture?

RL: That's our hope. Eventually there will be more of a universal need for a standard I/O bus, as more and more external devices fall under computer control. So, it is possible that all or part of our proposal may be adopted. And, as we begin to interface more and more devices on SCSI/PLUS, we may find that we may have to modify our proposed standard. Possibly, we may ultimately discover some aspect to ease the interfacing that we hadn't thought of before. And possibly, someone else might contribute an enhancement. That's yet to be determined.

You have to remember one important aspect from our conversation. SCSI was designed to provide a common interface for mass storage. SCSI/PLUS provides a way to open the architecture of single board computers, so that they can access mass storage devices as well as interface to real world applications by means of the 5380 SCSI Controller, or a future derivative of this silicon.

How The Little Boards Are Produced

Although I was familiar with graphic arts techniques and the photographic process through my years in advertising and publishing, I was unfamiliar with the process that turned a schematic into a fully populated computer board. Thus, when David L. Feldman, VP of Manufacturing, had some time to spare, I cornered him and got him to explain just what was involved.

Over the past 20 years, Dave has worked in various commercial, industrial, and social organizations in capacities ranging from data processing consultant to advertising. He is also an experienced draftsman and printed circuit board designer. He spent the four years prior to helping to start up Ampro working for Telesensory Systems, where he was Manager of Document Control and Engineering Services.

PR: What was your first reaction when Rick approached you to do the circuit board layout of the original Little Board?

DF: I knew he had something unique. I thought it was too sophisticated a product to interface to the Commodore; and the deeper we got into the project and saw how rapidly consumer interest and loyalties were changing, the more convinced we became that we should treat the board as a separate computer.

PR: How did the name Ampro originate?

DF: It was a contraction of Amateur and Professional — the markets we hoped to appeal to. Hence, Ampro. When Bill joined us in the project, we set out in earnest to make a go of it. Some years earlier, when I had a custom photographic processing service, I liked being in business for myself, and I jumped at the chance to try it again with Ampro. I knew we would have a rocky road over the first year or two. And if things clicked, I knew we would be successful and be able to contribute some useful products in the computer marketplace. We had some definite concepts in mind, and being relatively unconstricted by trends, we had the creative freedom to develop the Little Boards through several generations.

PR: Speaking of the Little Boards, would you mind showing me how the design is translated into a finished product?

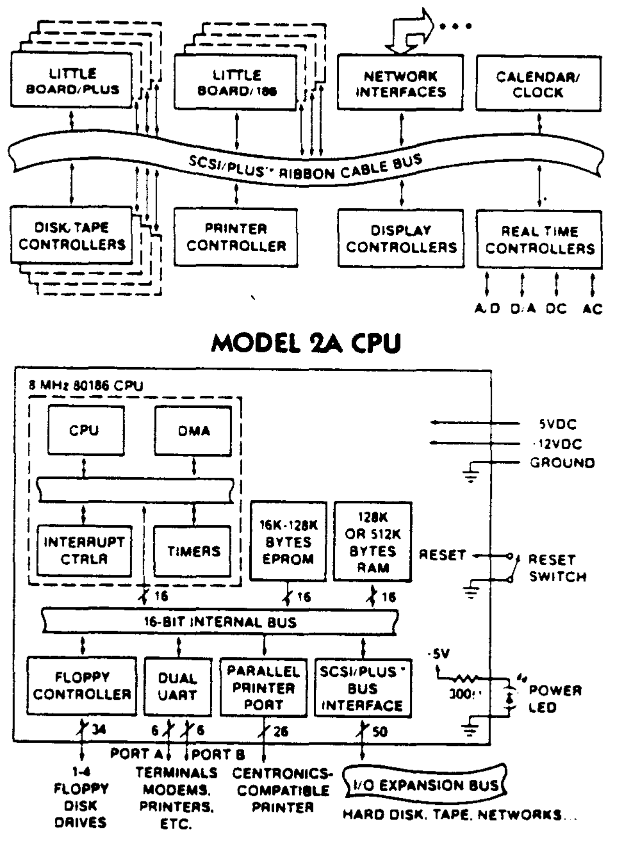
DF: Not at all. When my department receives a finished schematic, we do a chip and component count and calculate how much space we're going to need. Actually, this phase has several developmental stages, in which we trade off components while the board is in the design stages. If a chip count tells us we need 60 square inches to execute the design, but the Little Board is limited to 43 square inches, then something has to give. That was, essentially, the problem we faced with our forthcoming CMOS board, until we came across some custom chips that enabled us to eliminate enough TTL components to make the design fit into the Little Board format.

Anyway, to get back to your question, we prepare a rough component layout on paper, and then place it on top of a grid over a light table. We place a clear sheet of mylar over this and transfer all the component pads to the sheet from printed patterns. Another clear sheet of mylar is placed over the sheet with the component pads, and the component side of the circuit traces are laid down with blue plastic tape. I like to use blue and red to distinguished between the top and bottom sides of the circuit board.

As you'll notice from these sheets, all the circuit traces on the top layer go from left to right: And from top to bottom on the solder side. That's to facilitate completing a connection that starts at one end on the top, and terminates at the other end on the underside. Plated-through holes are used to transport a circuit trace from one side of the board to the other.

When the circuit traces have been laid down, we prepare another overlay containing all component ID numbers which will later be silk-screened on the board to aid in the board's assembly. Then everything goes to a photographic house which prepares a film of the solder pads with the component side circuit traces, and another one of the pads and the circuit traces of the solder side of the board.

A final photographic image is made to create the solder mask of each side. At one time, this process used to be drawn by hand, but cameras can now do this task much quicker. All I do is specify how many thousands of an inch clearance I want around each pad so that the wave soldering equipment will give us a clean connection at every point. This is done by defocusing the lens which enlarges the image of the pads to my specifications. They use a high-contrast film which, when it is developed, will provide sharp edges around all the pads. They send us a set of negatives and a set of positives. We retain the positives and send the negatives to the fab house.



Functional diagrams illustrating the diversified I/O capabilities of the SCSI/PLUS interface (Top), and the Little Board/186 (Bottom).

PR: I assume that they're the ones who will do the production runs?

DF: No. That phase of production is still weeks away. The fab house will produce a dozen or so custom boards for us for in-house and BETA testing. If there's a circuit connection missing, or the value of a component has to be changed to optimize the performance of the board, you'll want to make these changes now, rather than swallow the cost of several thousand finished Little Boards. So, if changes are necessary, we will simply redo the layout and prepare new films, which will go to our factory in Singapore.

PR: I've always wondered how the holes are drilled into a circuit board....

DF: This is a laborious task that's also prepared by the fab house. We have to provide a layout that indicates the proper hole size for each component. The boards will be stuffed by auto-insertion equipment, so there must be enough tolerance for the leads and pins to go through the holes without bending. The operator places our negative over a light table and looks through a scope that kind of resembles a bomb sight. After recording the outside dimensions of the board, he goes through the painful process of locating each hole, matching up a set of cross-hatches, and pressing a trigger to record that hole and the proper drill bit size.

It's not a job I would want, as there are anywhere from 7-800 component holes, and an equal number of plated-through holes to record. I think there are six different hole sizes on our average Little Board. All this data is recorded on an NC tape, which will later be fed into the drilling machine. A copy of this tape is also sent to our factory in Singapore when we're ready to send them the final negatives for a full production run. It usually takes anywhere from seven to ten days to produce a handful of test boards.

PR: So you now have this data on tape. How does the drilling process work?

DF: This is also an automated process. In actual production runs, the factory manufactures six boards on a sheet of fiber glass. After each board has been etched, these sheets are mounted on a table, and the arm of the drilling machine can hold six bits at the same time. When it starts up, the arm moves to a hopper where it selects six drill bits of the same size (such as an 0.031 or 0.045 size bit) and drills all the holes for that size on all six boards. When the holes of one size have been finished, the arm selects the next bit size. All in all, the drilling of all the holes on the six boards on that sheet of fiber glass takes about twenty minutes. When all the boards in a production run have been drilled, they go to the machine that will do all the component insertion. Then they're wave soldered.

PR: Out of curiosity, what does it cost to produce the bare boards in quantity?

DF: Well, in the case of the CP/M Little Board, the cost is around $7-8.00. The 186 board costs about $22.00.

PR: Why the disparity of cost when the boards are the same size and have approximately the same number of components?

DF: That's because the 186 board is four layers, which involves several additional production steps. I'm glad you asked that question, because laying out a four-layer board is a much more complicated process. We knew right from the start that we couldn't possibly get all the circuit traces on the top and bottom sides of the board. So, the first step in a case like this is that you draw separate layers for the power plane and the ground plane. Then, at those points where you have to make a VCC or ground connection within the circuit, you have to create "tag-ins" that will carry the VCC or ground to the proper pins of your components.

Since you have to be conscious of all the holes that are going to be drilled, you have to use a very thin tape to lay down your circuit traces. Ultimately, when the original negatives are reduced in size, some of these traces are going to be as thin as a human hair. Now each of these layers are etched on thin sheets of plastic material. The etching process is critical. If they're left in the acid too long, some of the fine circuit traces might be totally etched away.

The etched sheets of all the layers are now ready to be assembled. They place a blank fiber glass board in a jig which has pins to hold the board and the individual layers in place. This, too, is critical. First an adhesive (or bonding substance) is placed over the surface of the board. Then the sheet that has the component pads and the component side circuit traces is laid down. Two more bonding layers are placed on this. This is followed by the power plane, another bonding layer, the ground plane, two more bonding layers, and, finally, a sheet of laminate. Then the board is turned over and the solder side is laid down. The assembled sheets then go through a heating press to fuse all the layers together and then through a controlled cooling process so that the boards don't bend. The final steps will add the solder mask and the silk screen legends.

The sheets are then drilled and cut apart to create six individual Little Boards. The edges are routed, as are the mounting slots. Since the bonding process will sometimes tear some of the hair-line circuit traces, we insist that the manufacturer test the continuity of all the traces. And after the boards are assembled, they must perform another test to check that all the passive components are alive. This is done by placing the board on a device that looks like a bed of nails. After the IC's have been added we insist that the manufacturer run a diagnostic check on every board. We've sent him terminals and disk drives for this purpose. If the manufacturer neglects to perform any of these steps — especially the continuity check — and we receive a board that is D.O.A., we will charge back the full assembled cost of the board. This will affect his profit, so he tends to follow our procedures.

PR: Obviously you don't take his word that the boards are fully functional.

DF: No we don't, but you would be surprised to learn that some computer manufacturers only have the factory perform spot checks on the boards that are manufactured. The ones who care about the reliability of their products will test every board. The boards are usually shipped to us in cartons of 50, as they come off the production line, and we throw them on a burn-in rack for 48 hours. Then we place each board in a jig that contains connectors for the serial ports and the floppy drives. We plug in a SCSI cable connected to a hard disk and a printer cable, and run a diagnostic program from the console terminal that reads and writes to every port. If everything checks out, they're placed inside anti-static bags and packed for individual or bulk shipments.

PR: When the manufacturer tests the in-circuit continuity of four-layer boards, what fall-out percentage does he have?

DF: There's about a five per-cent fall-out rate. It used to be much higher. And that figure is dropping steadily all of the time. It's very rare for a standard two-layer board to be tested as thoroughly as a four-layer board, because the production techniques are such that there is almost a zero defect ratio. Six, eight and even ten-layer boards are coming on the market now, and the fall-out rate is much higher. About twenty-five per-cent of all eight-layer boards are defective. An eight-layer Little Board would probably cost us about $45 in bare form. We pay for the fall-out rate.

As you know, we sell a lot of assembled systems in our series 100, 200 and 300 Bookshelf systems, but we don't put them together ourselves. All the cases, cables, and drives that we've ordered come to our office where all the components are given a visual check. They're sent to the company that does our pre-production test boards, because they have an assembly-line set-up for this purpose. When the assembled units are returned, we hook them up for another 48-hour burn-in, during which time a read/write to the disk drives is also performed. If anything breaks down, it's usually the drives, but not very often.

PR: What drives are you using now?

DF: We use Panasonic floppy drives, and Xebec "Owls" and Seagate hard disk drives. Our selection was made after we sent a batch of drives over to Dysan, where they have a laboratory that ingeniously tries to destroy a drive. If the drives survive, then they're worth using. The cheaper models don't last. That's why it's very rare for a Little Board to fail. They're nearly bullet proof. It costs us money to service our products. It also costs our customers time and money if a board fails. They're paying us good money to provide them with a computer board that will work the first time and everytime. You don't retain customers if you send them inferior products.

PR: I'll buy that. However, can you tell me, from your experiences, over the last few years, what usually goes wrong if a board fails?

DF: An IC might go. But, generally, if an IC slipped past the manufacturer's quality control, then it usually dies during the burn-in phase. In assembled systems, a drive might go under occasionally if it's abused. But the single most common problem stems from the users themselves. They fail to follow our very explicit directions that tell them pin 1 on the power connector is +5 volts; pins 2 and 3 are return grounds; and pin 4 is +12 volts. They either reverse the power and ground pins and send the board into cardiac arrest, or they solder the power supply leads directly to the Little Board. We use standard connectors and we identify the part names and the manufacturer's ID number so that the user is guided to the proper part. But sometimes, after spending several hundred dollars for one of our boards, they balk at spending fifty-cents for the right connector — and boom!

David Feldman, VP of Manufacturing

What's Next For The Little Boards?

I finally caught up with William Dollar for an evening of nagging questions. Bill is in a constant state of hyper-drive, whether in or out of the office. I suppose that's only natural for someone who has spent the last 20 years travelling around the country.

After receiving a Marketing Degree from San Francisco State University, in 1963, he joined Sperry-Univac, where he worked himself up to the position of Senior Account Representative for scientific and commercial markets using the UNIVAC 1107 system. Early in 1967, he began an 11-year stint with Control Data Corporation, responsible for sales of minicomputers, large scale systems and scientific software applications to such firms as Lockheed Missiles and Space, SRI International, NASA-AMES and Ford Aerospace. In 1978, he joined Itel Corporation as National Account Manager, where he formulated marketing strategies for sales of large computer systems to Fortune 1000 companies.

In 1980, he co-founded Designet, a firm that provided consultation services to the communications industry. One of his key accounts was the Rolm Corporation, where he conducted data communications seminars. In 1982, he sold his interest in the firm and joined Applied Digital Data Systems. This eventually brought him into contact with Rick Lehrbaum and Dave Feldman.

PR: My questions aren't always tactful. I suppose all three of you put your own money on the line in more ways than one to launch Ampro as a full-fledged operation.

WD: We converted all of our assets to cash and jumped in with both feet. A few close friends bought shares at the beginning, but it was mostly our own funds.

PR: Any regrets?

WD: None. I don't think Dave and Rick have any either. I think all three of us have had more honest satisfaction with Ampro than anything else that we've done over the years. We've all worked for some good companies, but there's nothing like working for yourself. Fortunately, we haven't lost our sense of humor and our pride. This helps to sustain you emotionally over the rough spots.

PR: What about venture capital?

WD: We've had to turn that down on several occasions in the last year, because these groups want controlling interest, and they want to tell you what to create. We have our long-range plans fairly well defined, and it doesn't include markets with which we're unfamiliar. You have to be careful how you choose your bedmates, especially if you're going to be working together. If we do decide to go this route, we prefer it to be on our terms.

PR: What's ahead for Ampro? Are you going to continue producing faster, more powerful Little Boards? Or are you going to spin off products like the T-BOX I saw earlier today?

WD: The answer is yes to both questions. We will continue to produce products based on the Little Board concept. They will take advantage of all the new developments in chip technology. We have to grow in that respect, because we now have a sizable and established customer base that will need more powerful tools in the years ahead. I hope they'll continue to look to Ampro for computing solutions. But we won't be producing microprocessor based boards at the rate that we have been producing them. This was necessary during our formative stages. We had to build a variety of boards to cover most 8-bit and 16-bit computing operations. What you will see from Ampro in the next few years will be an interesting series of packaged applications.

PR: Do you want to elaborate?

WD: Right now, we're just releasing the T-BOX. This unit, which can effectively replace any Telex system in the country for a quarter of the cost — and without the additional monthly charges for having a dedicated line running into your office — will automatically dial a user's Telex service or Electonic Mail Box, log on, and download all messages. The frequency of these calls can be established either through software or onboard firmware. The messages are stored on disk to provide a permanent record. Sending messages is equally convenient. The sender merely types the message on his computer, substitutes his disk for the disk in T-BOX, and the message sequence is handled automatically.

Next, we will have a SCSI Test Package, complete with hardware and software, that will allow any manufacturer of SCSI devices — such as hard disk and tape units — to gang-mount dozens of units at one time in order to perform a predetermined series of diagnostic tests to check out the integrity of those products. There is nothing like it on the market today.

We're also developing Concurrent DOS for multi-user systems in conjunction with our 186 PC-DOS board. One of the Beta test sites will be a drug manufacturer who is trying to eliminate expensive regional seminars in order to explain new products to his field representatives. Usually these seminars are a series of lectures with a question and answer period. Since no one wishes to sound foolish in front of one's contemporaries, no one asks questions, and management is never certain how much of the data they presented is actually retained. We're setting up clusters of four-user systems across the country. When a new product is introduced, data sheets will be distributed to all field personnel. Then they can go to one of the regional centers in their area, as work schedules permit, and have their knowledge on the product re-enforced through a hands-on session with our computer systems. The session concludes with a very detailed series of questions, after which the representative is graded. The participants remain anonymous, so they can learn at their own pace. The system will be used to educate them on all of the company's products, so that the representatives go through a continuing educational process that should enable them to answer questions from doctors and hospitals with greater accuracy and authority. This concept of staff training has application in many fields where there has to be an on-going educational process to stay abreast of new information, concepts, and so forth. This is an age where dispensing information is often a critical and expensive problem. We'd like to offer our solutions on a cost-effective level.

The new Little Board we have coming out in January, 1987, will be a radical departure from anything now available on the market. It uses a new microprocessor and several custom chips. It's an all CMOS board and consumes only 1.5 watts of power. That means you can place it into a sealed environment in Alaska, or in the jungles of South America, and it will function properly. It will run all IBM software. You can attach an IBM keyboard to it — or even a standard ASCII terminal. Did Rick and Dave show it to you?

PR: Yes. I'd like to stick one into my jacket pocket with a battery pack and a keyboard on my lap.

WD: It may come to that, too. Anyway, we now see our market becoming more defined with packaged applications that provide quick and easy solutions.

PR: I know it's been nearly three years since you launched the 8-bit CP/M Little Board. How is that product surviving in the relation to your 16-bit board and the companion boards?

WD: We're selling more of these now than we did in the early days. Not all applications require a 16-bit PC-DOS board. One of our industrial users — a firm in Denver called Robotool — manufactures robotic milling machines. Each robot costs $20,000. It only requires our CP/M board to make it work. The board is mounted in a box with a numerical keypad through which the operator codes in the X-Y-Z coordinates. Then he sets the machine in motion. If he wants the robot to perform a different set of routines, he merely punches in a new set of coordinates. The brains of this $20,000 robot is a $200 computer board.

We've supplied more than 700 CP/M boards to a company in Sweden that makes a point of sale terminal used for off-track betting. The limited scope of the ticket sales doesn't require hundreds of bytes of memory, so the CP/M board is more than adequate. A lot of times a customer doesn't want to talk to you unless you're out to sell him a 16-bit board. All he knows about computers is that the IBM-PC is a 16-bit computer. So, when we deal with the technicians who will ultimately work with our hardware, we try to convince them to trust our judgment on which product gives them be most performance and the lowest price. I think you gain more respect and have a better long-term relationship with a client if you're honest with him. He's coming to you, in a sense, to solve his problems. Treat his needs accordingly.

PR: Do you contract with outside consultants for special projects?

WD: Most of our software in communications, desktop applications, and so forth is either contracted for or licensed. Our Concurrent DOS is a contract project. We will be using a form of Xenix or Unix for our CMOS board. This product will be licensed. We only prepare the driver software that accesses I/O devices and peripherals, and the codes for our BIOS firmware.

PR: What about hardware?

WD: That's all done internally through our engineering department. With one exception. And that was the Video Ram Emulator card for our 186 PC-DOS board. This was an odd situation that I think you'll find interesting. A couple of months after the 186 was released, a young man from Lebanon walked into our offices one day with a wire-wrap card under his arm. He said it could emulate IBM screen graphic calls on an ASCII terminal. This was a product we had talked about doing sometime in the near future. His product was quite ingenius — and, it worked. We offered to buy the rights and pay him a royalty on each board we produced. But he declined our offer. He had dreams of starting his own company, and friends and relatives offered to lend him $20,000 for start up expenses.

Rick told him that he needed ten times that sum just to go into production, but the young man was adamant. So we told him that if he could manufacture the product for us, we would buy it. If he couldn't, then come back and we'll license it. Well, we didn't hear from him for months, until he showed up suddenly and gave Rick all the drawings and schematics and accepted our offer. His funding dwindled very quickly, even before he could have a working prototype made at a fab house. Ultimately, we'd like to have him come to work for us because he's an unusual talent. All of our programmers and engineers have been hand-picked for their special talents.

PR: Are you planning to do any custom chip designs for future Little Boards?

WD: I think we will have to. The price of custom chips has come down to where it is almost an affordable reality. It will be a necessity when we start work on a 32-bit Little Board. The overall architecture is so large that we will have to find some way of reducing the number of support components without crippling the power of the microprocessor. We're using custom IC's in our CMOS Little Board, but these were developed by Vadem, Inc. That's the firm that wrote the BIOS for the Morrow Pivot laptop computer that Zenith sold in huge quantities to the government.

William I. Dollar, Ampro President (left) with Don Costella (right) of Disk Plus, Inc., Ampro's midwest distributor at the Atlanta COMDEX Computer Fair.

PR: You've mentioned networking projects in some of our phone conversations. Are they still viable?

WD: More than ever. The big problem with networks is that most people don't know what they are and what they can do. There are more network interface devices on the market than you can count, and the prices asked for some of them are twice what we charge for a 16-bit microprocessor board. There's a local company that you know about that puts out a neat little product costing under a hundred dollars per station. They've revised the drivers to accomodate the interrupt structure of our BIOS. This will be our initial offering. But we're going to go beyond this concept by offering greater mobility, without a lot of wires running through an office. Ours, in fact, won't have any wires at all. It's a simple concept but requires some sophisticated technology that I can't mention until we've conferred with the FCC about their restrictions and requirements.

PR: How big would you like to see Ampro become?

WD: The obvious answer to that question is to say as big as we can. But I don't think that's quite true. Let's just say that I hope we don't become so big that we lose personal contact with our market and that Rick, Dave and I no longer get a kick out of having created something special out of one of our Little Boards. ■

NEW-DOS

Part 5: The CCP Commands Continued

By Thomas Hilton

The SAVE command is generally used only by system level programmers. This command is required for all modifications and installations of Hermit DOS. The function of the SAVE command is to create a document from the program currently residing in memory, which begins at the base of the TPA. The syntax of the SAVE command procedure is noted in the source code listing shown in LISTING ONE.

Our first concern is to extract the number of pages to be saved from the command line. The support routine NUMBER will parse this parameter, convert it to binary, and return it to us in the accumulator. As the number of pages it is possible to save from memory, in a 64K system, is an eight bit value, we clear the upper register of the HL pair to assure a valid number is represented by the 16 bit register set. Having loaded the registers we then save the page value upon the stack.

Next we test for the existence of a document by the name specified by the operator. If it does not already exist, or if the old document is to be overwritten we will return here. We then create a document index for the new document. Assuming all will be well we get the page count back, but if there was a problem in creating the document index, such as the index not having a vacant entry, we will jump out.

As with all sequential write operations we are required to zero out the current record counter. This assures that the FDOS segment will begin creating the document properly, at the beginning. Once we begin writing the document to disk the FDOS will advance the current record count as it allocates disk space. We are charged with setting the initial value only.

Recall that the FDOS has a sector mentality, and can only deal with data 128 bytes (one sector) at a time. A "page" of memory is 256 bytes, a sector is 128 bytes. We therefore convert the number of pages to be saved in the document by doubling the number of pages, which equates to the number of sectors to be saved. Having done that we save the working position in the command line and set the DE pair to the first byte of the TPA, or user program area, for use in defining the DMA buffer area for reading a sector of memory onto disk. With that done we drop into the procedure's main execution loop, shown in LISTING TWO.

The sector counting routine is a "count down" sequence. Though we placed the number of pages into the L register, and cleared the H register, this was before we doubled the count in converting the data to sectors. In order to determine if we have saved all of the sectors we are required to save we do a logical

LISTING ONE

```
;       PROCEDURE IPP.05
;       ----------------
;       Purpose: Save the system memory from the start of the TPA, for the
;                number of system pages specified, into a document specified
;                by the operator.
;       Verb Syntax:
;                save <pages in decimal>     <unambiguous document name>
;                save <pages in hexadecimal> <unambiguous document name>
;     . Examples: save 41 system.com
;                 save 41H system.com
;
SAVE:   CALL    NUMBER        ;get the number of pages from the command line
        LD      L,A           ;which is placed in the HL register pair
        LD      H,0           ;but only L is a valid value so clear H
        PUSH    HL            ;and save page count on the stack for later
        CALL    EXTEST        ;test for existence of document
        LD      C,22          ;if document does exist we come here and
        CALL    CODRET        ;open a document index entry
        POP     HL            ;and then get the pages to save back
        JR      Z,SAVE3       ;but jump out if we had a problem creating the
                              ;document index entry
        XOR     A             ;else we zero out the DCB record record count
        LD      (DCBCR),A     ;and prepare the new document's DCB
        ADD     HL,HL         ;then double the page value, which makes it the
                              ;number of sectors as there are 256 bytes per
                              ;page, or two sectors
        LD      (CMDPTR),DE   ;then update the command line pointer
        LD      DE,TPA        ;and point to the start of the TPA save area
                              ;and fall into the save loop
```

LISTING TWO

```
SAVE1:  LD      A,H             ;are we done with the sequence yet?
        OR      L               ;if HL is zeroes out we are
        JR      Z,SAVE2         ;so we jump out
        DEC     HL              ;else we decrement the sector count
        PUSH    HL              ;and save the new count on the stack
        LD      HL,128          ;then add 128 bytes, (a sector) to the pointer
        ADD     HL,DE           ;so we can transfer the next block
        PUSH    HL              ;and save that pointer value on the stack
        CALL    DMASET          ;and set dma address for write with the
                                ;address in the DE pair
        LD      DE,DCBDN        ;then load the DCB address for FDOS
        LD      C,21            ;load the FDOS function code
        CALL    FDOSB           ;and make the call saving BC
        POP     DE              ;then we get back the pointer to next sector
        POP     HL              ;and get the sectors to save count back
        JR      NZ,SAVE3        ;then if we had a write to disk error we would
        JR      SAVE1           ;jump out else loop till done
SAVE2:  LD      DE,DCBDN        ;when we get here we are done so we load the
        CALL    CLOSE           ;DCB and close the document
        INC     A               ;did we have an error in closing?
        JR      NZ,SAVE4        ;no, so jump over error routine
SAVE3:  CALL    PRNLE           ;else print the error message
SAVE4:  CALL    DFTDMA          ;reset the DMA buffer to PZBUFF
        JP      RSTCCP          ;and exit the procedure
```

"OR" of the upper and lower registers against each other. If the result is zero then we are finished. If we have a nonzero result then we have more to do.

With more to do we first decrement the sector count, save it, then add a sector worth of bytes to the memory pointer, advancing it upwards in memory 128 bytes, and save that value on the stack as well. We then use this new memory pointer to define the DMA buffer address for use by FDOS. With each pass the DMA buffer will be a sector higher in memory, though FDOS doesn't care, as long as it does have a buffer address to use. If we did not advance the DMA pointer we would read the same 128 memory space onto disk over and over until the sector count was exhausted. This is not a trivial, nor foolish concept. Several times a program has had a bug in it and the problem was discovered by examining the saved document.

With the DMA address defined we load the DE pair with the address of the user's DCB and make the write sequential FDOS service call.

Upon return from the write sector FDOS call we retrieve our pointers and counters and check to see if we have had any type of write error. If all has gone well we will jump back up for another sector. If we have had any type of error then we will jump out to the error handling routine.

When we come to SAVE2 we have exhausted the sector count. We then load the address of the user's DCB and call the FDOS "close" service. The "close service" updates the document index, does a little housekeeping, and writes the document index information to disk. If we have not had an error in trying to close the document then we will jump over the error routine and perform a standard procedure exit.

If there has been an error in creating a document index, writing a sector to disk, or closing the document we will enter our exiting code at SAVE3. From this point we will write an error message upon the terminal, and return. Upon return, or upon entry at SAVE4, indicating a completed procedure, we will redefine the DMA address to that of the page zero buffer, located at 0080H, and then return to the CCP command level entry vector to reset the file partition and active drive, as shown in LISTING THREE.

The CLOSE support routine does little more than load the FDOS service code for closing a document, and then jumping to a generic disk service vector for processing the service request, and returning a formatted status, or error code.

The support routine FILNUM actually belongs to the FILE command, but was included here due to its use of the number routine, and the use of the number routine's error codes. It parses the operator specified decimal number from the command line, and assures it is a valid partition number.

The number support routine begins by parsing a decimal number from the command line, which is placed in the user's DCB. We set an index pointer to point to the end of the possible number string. We then load the B register with the maximum number of characters a number might attain for conversion into binary format. The maximum number of characters is defined by the number of valid characters in a document description, or eleven characters.

Because the DCB is padded with spaces we may begin searching for the first decimal digit by scanning backwards, checking for any character that is not a space, continuing until a nonspace character is located.

When we get to NUM0 we have detected an nonspace character in the string buffer. Is it a token to signify a hexadecimal, (base 16), number? If it is then we will jump to the hexadecimal conversion routines, else we will attempt a decimal conversion.

We begin the decimal conversion by resetting the pointer to the start of the assumed numerical string, and preparing the BC register pair with the maximum number of characters which may be converted, and clearing the C register.

Next we get the first/next character and determine if we are finished with the conversion, which will be indicated by a space

LISTING THREE

```
CLOSE:  LD      C,10H
        JP      CODRET
;       Parse number from command line, and return to caller with value in A
;
FILNUM: CALL    NUMBER
        CP      MAXUSR+1
        JR      NC,NUMERR
        RET
NUMBER: CALL    PARSE           ;parse number and leave it in DCB
        LD      HL,DCBFN+10     ;point to end of string for conversion
        LD      B,11            ;the number string may not be longer than a
                                ;full document name, or eleven characters
;       Check to see if the number is in hexadecimal, base 16
;
NUMS:   LD      A,(HL)          ;get the number of characters from the end
                                ;of the string so we can check for suffix
        DEC     HL              ;back up a character
        CP      ' '             ;is it a space?
        JR      NZ,NUMS1        ;no, so check for suffix
        DJNZ    NUMS            ;and keep looping backwards till we have a
                                ;valid character or number
        JR      NUM0            ;if all else fails try to process it
NUMS1:  CP      'H'             ;is the character the hex suffix?
        JR      Z,HNUM0         ;yup so jump out
NUM0:   LD      HL,DCBFN        ;no, so process as a decimal and load the
                                ;address of first character in string
        LD      BC,1100H        ;C is the accumulated value, and B will hold
                                ;the character count. Now C=0, and B=11
NUM1:   LD      A,(HL)          ;get a character
        CP      ' '             ;is it a space?
        JR      Z,NUM2          ;we're done if so
```

LISTING FOUR

```
        INC     HL              ;else advance pointer to next character
        SUB     '0'             ;convert to binary (ascii 0-9 to binary)
        CP      10              ;is it greater than ten?
        JR      NC,NUMERR       ;it's an error if so
        LD      D,A             ;else put the digit in D
        LD      A,C             ;and make multiply the old value
        RLCA                    ;by ten
        RLCA
        RLCA
        ADD     A,C             ;and check for a range error
        JR      C,NUMERR        ;jumping out if so
        ADD     A,C             ;then check once more for a range error
        JR      C,NUMERR        ;jumping out if so
        ADD     A,D             ;else the new value equals the old value
                                ;multiplies by ten plus the digit
        JR      C,NUMERR        ;and exit if that caused a range error
        LD      C,A             ;else set the new value in C
        DJNZ    NUM1            ;and loop till done
NUM2:   LD      A,C             ;then put the value in the accumulator and
        RET                     ;return to caller
NUMERR: CALL    SYSPTC          ;here we write an error in number message
        DEFB    'Error in number, try again',0
        JP      RSTCCP
```

LISTING FIVE

```
HEXNUM: CALL    PARSE           ;we get here if the number is in hex so we
HNUM0:  LD      HL,DCBFN        ;parse the string into the DCB use HL as the
        LD      DE,0            ;pointer, DE holds the accumulated value
        LD      B,11            ;B holds the character count
HNUM1:  LD      A,(HL)          ;and we get the character
        CP      ' '             ;is it a space?
        JR      Z,HNUM3         ;jump out if so, we're done
        CP      'H'             ;else if it is the suffix we are done
        JR      Z,HNUM3         ;and jump out
        SUB     '0'             ;else we convert it to binary
        JR      C,NUMERR        ;and jump out on error
        CP      10              ;is the number in the range 0-9?
        JR      C,HNUM2
        SUB     7               ;is it in the range a-f?
        CP      10H             ;or how about a plain old error?
        JR      NC,NUMERR       ;go if so
HNUM2:  INC     HL              ;else bump the pointer
        LD      C,A             ;load the digit in C
        LD      A,D             ;get the accumulated value
        RLCA                    ;exchange the nybbles
        RLCA
        RLCA
        RLCA
        AND     0F0H            ;and mask out the low nybble
        LD      D,A             ;put it back in D
        LD      A,E             ;then switch the low order nybbles
        RLCA
        RLCA
        RLCA
        RLCA
        LD      E,A             ;and put the high nybble of new value in E
                                ;the low nybble in D
        AND     0FH             ;mask off the upper bits
        OR      D               ;then overlay the accumulator onto D
        LD      D,A             ;and put it back in D
        LD      A,E             ;then get R
        AND     0F0H            ;mask out lower bits
        OR      C               ;overlay C
        LD      E,A             ;put it back in E
        DJNZ    HNUM1           ;and repeat until done
HNUM3:  EX      DE,HL           ;then put the processed value in HL
        LD      A,L             ;and the low order byte in A and
        RET                     ;return to caller
```

LISTING SIX

```
;       See if the document name in the DCD exists, ask operator to delete
;it if it does exist, and abort procedure if operator responds to the
;negative
;
EXTEST: CALL    PARSE           ;parse the document name into the DCB
        JP      NZ,ERROR        ;wild cards are not permitted here
        CALL    SSELDSK         ;select a disk if needed
        CALL    SEARF           ;look for specified document
        LD      DE,DCBDN        ;use DE as a pointer
        RET     Z               ;and return to caller if doc. not found
        PUSH    DE              ;else save the pointer
```

Listing 6 continued

```
        CALL    SYSPTC          ;and ask operator wants to overwrite document
        DEFB    'Overwrite Existing Document?',0
        CALL    INPTRM          ;get operator's response
        POP     DE              ;get the pointer back
        CP      'Y'             ;did we get the go ahead?
        JP      NZ,RSTCCP       ;no, so abort procedure
        PUSH    DE              ;else save the pointer again
        CALL    DELETE          ;delete the existing document
        POP     DE              ;get the pointer back
        RET                     ;and return to caller
```

LISTING SEVEN

```
;       PROCEDURE IPP.06
;       ----------------
;       Purpose: Select a new file partition.
;       Verb Syntax:
;                 file <partition number>
;       Example:  file 12
;
FILE:   CALL    FILNUM          ;parse file number from command line
        LD      E,A             ;place file number in e
        CALL    SETFIL          ;set the new file number
RSTJMP: JP      RSTCCP          ;and exit procedure
;       PROCEDURE IPP.06
;       ----------------
;       Purpose: Rename a document
;       Verb Syntax:
;                 call <new document name> , <old document name>
;
RCALL:  CALL    EXTEST          ;test for existence
        LD      A,(TEMPDR)      ;get the temporary drive
        PUSH    AF              ;save it
CAL0:   LD      HL,DCBDN        ;move the new document name to a safe place
        LD      DE,DCBDM
        LD      BC,16           ;16 bytes
        LDIR
        CALL    ADVAN           ;advance the command pointer
        CP      ','             ;is the character the into character ','?
        JR      NZ,CAL3         ;no, so jump out
CAL1:   EX      DE,HL           ;swap the registers
        INC     HL              ;advance the pointer
        LD      (CMDPTR),HL     ;save pointer to old document name
        CALL    PARSE           ;parse out the document name
        JR      NZ,CAL3         ;no wild cards here!
        POP     AF              ;get old default drive
        LD      B,A             ;save it
        LD      HL,TEMPDR       ;compare it against current default drive
        LD      A,(HL)          ;match?
        OR      A
        JR      Z,CAL2          ;yes so jump
        CP      B               ;check for drive error
        LD      (HL),B
        JR      NZ,CAL3
CAL2:   LD      (HL),B
        XOR     A
        LD      (DCBDN),A       ;set default drive
        LD      DE,DCBDN        ;rename document
        LD      C,23            ;FDOS function code for rename
        CALL    CODRET
        RET     NZ              ;if all is well exit procedure
CAL3:   JP      ERRLOG          ;else write an error message
        DEFS    32              ;stack area
STACK   EQU     $               ;top of stack
        END
```

FOR TRS-80 MODELS 1, 3 & 4
IBM PC, XT, AND COMPAQ

THREE TOUGH QUESTIONS

WITH ONE EASY ANSWER:

1. **WHEN IS A COMPUTER LANGUAGE NOT A LANGUAGE?**
MMSFORTH includes DOS, Assembler and high level commands and extraordinary utilities, extends to become any other language (or application), is an interpreter and a compiler, and is remarkably fast and compact!

2. **WHICH SOFTWARE RUNS THE SAME DISKS IN IBM PC AND TRS-80 MODEL 4?**
MMSFORTH disks run on those and Compaq, and TRS-80 Model 3, and Tandy 1200, and TRS-80 Model 1, and AT&T 6300, etc., with your choice of formats up to 200K single-sided or 400K double-sided!

3. **WHO OFFERS SOURCE CODE WITH ITS LANGUAGE, UTILITIES, DATABASE, WORD PROCESSOR AND COMMUNICATIONS SOFTWARE?**
Nearly all MMSFORTH software includes source code.

MMSFORTH

All the software your computer may ever need.

The total software environment for IBM PC, TRS-80 Model 1, 3, 4 and close friends.

- Personal License (required):

MMSFORTH System Disk (IBM PC)	$249.95
MMSFORTH System Disk (TRS-80 1, 3 or 4)	129.95

- Personal License (optional modules):

FORTHCOM communications module	$ 39.95
UTILITIES	39.95
GAMES	39.95
EXPERT-2 expert system	69.95
DATAHANDLER	59.95
DATAHANDLER-PLUS (PC only, 128K req.)	99.95
FORTHWRITE word processor	175.00

- Corporate Site License Extensions . . . . **from $1,000**
- Some recommended Forth books:

UNDERSTANDING FORTH (overview)	$ 2.95
STARTING FORTH (programming)	18.95
THINKING FORTH (technique)	15.95
BEGINNING FORTH (re MMSFORTH)	16.95

Shipping/handling & tax extra. No returns on software.

Ask your dealer to show you the world of MMSFORTH, or request our free brochure.

MILLER MICROCOMPUTER SERVICES
61 Lake Shore Road, Natick, MA 01760
(617) 653-6136

character in the DCB buffer area. If we are finished we will jump out, else we will continue, in LISTING FOUR.

We next advance the counter and assure that the character represents a valid numerical character. If it is not a valid numerical character then we have an error condition and abort the conversion.

The conversion process is best understood by "paper computing." That is, perform the conversion process using paper and pencil, following each program step in sequence, and noting the condition, and contents, of the working registers. There are better decimal to binary algorithms, but few that are as economical as this one. When we have either detected a space character, or have exhausted the number of characters possible to convert, we will place the converted value, held in the C register during processing, into the accumulator, and return to caller.

If we have had an error in the conversion process, or the operator has input an illegal character, then we end up at NUMERR. Once here we print a generic numerical range error message and return to the CCP command level.

If we had detected a hexadecimal suffix, indicating that the number is a hexadecimal expression, then we will have been sent to HEXNUM, in LISTING FIVE. In a similar fashion to the decimal conversion routine we parse out the numerical expression, set the HL pair as an index, the maximum number of characters allowed in the expression string, and clear the DE register to hold the conversion working value(s).

Now we get the character, and check to see if we are at the end of the string, indicated by a DCB space pad character. If it is a space then we are done. If it isn't a space then perhaps it is our suffix character. If it isn't our suffix character, then is it a valid hexadecimal character? If not we have an error. If it isn't a decimal number is it in a valid range for a hex character? If not then we have an error, plain and simple. If all is well then we will begin the conversion sequence. Again, you are advised to "paper compute" the algorithm for understanding.

In LISTING SIX we test to see if the document exists, we use support routines from the other personality procedures. When installing your own procedures, and deleting others, be very sure that you do not remove a support routine used by others.

We begin the test by placing the document name into the DCB. If there is a wild card in the document name we will abort. Wild cards are not allowed in the document name for the procedures calling this routine. If all is well, thus far, we will select the user specified drive, if any, and use the FDOS search services to determine if the document already has an index entry. If no index entry for the specified document exists, then we may proceed, else we will ask the operator if the document is to be overwritten.

If the operator indicates a desire to preserve the existing document we will abort to the CCP command level. If not we will free the old document's allocations and return to caller.

The FILE command procedure sets the imaginary file partition area. There are 16 files under the basic version of Hermit DOS. Three of these files have special meaning.. File 0 is the system file, where command documents may be placed to get them out of the way, while still making them available to the system. File 15 is the archive file, where documents may be placed for safekeeping. File 1 is the default file assigned by the BIOS. Refer now to LISTING SEVEN.

The file command is very simplistic. We parse the file number from the command, using a routine we have discussed previously, which is returned to us in the accumulator. When we get the file number we place it into the E register and make an FDOS call to set the specified file number, and return to the command level return vector which will make the selection active.

The CALL procedure is very close to the CP/M REN function in structure. We begin by testing for the existence of the new document name in the current disk's index. We then get the temporary drive number and save it upon the stack.

The new document name is then moved into the DCB at DCB+16, where the FDOS will expect to find it, for the renaming process. In the above code the BC register pair is used as a counter for a Z80 specific block move instruction. Hermit DOS uses a large number of Z80 specific instructions, which is one reason why it will not run on older systems using the 8080 or 8085 microprocessor.

We then attempt to locate the second, or original, document name. We do this by looking for the delimitting comma which must separate the two document names. If it is not found then we will jump out.

We then swap pointer registers for use by other routines, update the command line pointer, save it, and parse the second document name, which will be placed into the DCB in proper position for the renaming process.

Next we check to assure that the default drive and the temporary drive are the same. We cannot request that the new document name be on a different drive than the old document. This is a simple renaming sequence, not a copy command.

If all is well with the drive selection then we will load the temporary drive variable with the default drive number, and set the DCB also for the currently active drive prior to making the renaming service call. Upon return, assuming we have had no indication of an error, we will return to the CCP command level by a return to the RSTCCP vector, whose address is on the stack.

If we have had an error then we will jump to a generic error handling vector, prior to returning to the CCP command level at the RSTCCP vector.

The following code defines the CCP stack area where our return addresses and temporary data has been stored. As might be noted, this is a very small stack, but our traffic is not that intense. As might also be noted, this also defines the last byte of the CCP, hence our discussion of the CCP has been completed.

From this point onward, the CCP is all yours, to do with as you feel best. Never be afraid to try something new. Just make sure that you make a working disk for your experiments, keeping an archive copy of your source code. Everyone makes mistakes, and back up copies can save one's temperament. ■

ZSIG

by Jay Sage, ZSIG Software Librarian

In my last column I promised that this time I would discuss some ideas for new Z-System programs and program enhancements. Before turning to that subject, however, I would like to address another issue.

ZCPR3 on a Floppy-Only System

Many people have said that ZCPR3 is a great operating system but that one must have a hard disk system to use it. I would like to put that myth to rest. ZCPR3 CAN BE USED AND USED VERY EFFECTIVELY ON A COMPUTER WITH FLOPPY DISK DRIVES ONLY. I have Z-System running on six 8-bit computers, and only one has a hard disk drive (that's the Z-Node machine, which I almost never get to use myself). Admittedly, ZCPR3 runs much better on a system with a hard disk drive and/or RAM drive, but everything runs much better on such a system. Z-System no more requires a hard disk than does standard CP/M.

Squeezing More Performance Out of a Floppy System

A full-up Z-System with shells, error handlers, a search path, and an extended command processor (ECP) can make for a lot of disk accesses, and this can slow operation down. However, there is a simple technique that can help a great deal and can even benefit a hard disk. To understand the technique, you need a little background on how the disk operating system (DOS — either BDOS, ZRDOS, or P2DOS) works and how data are stored on a diskette.

Let's take up the second matter first. Data on a diskette are stored in circular tracks numbered from the outside of the diskette. The first two tracks (sometimes one, sometimes three) are reserved for the operating system code and are called "the system tracks." When the computer is first turned on or when the reset button is pressed, the microprocessor starts running a program stored in a ROM (read-only memory chip). This program contains instructions that initiate a process whereby the entire operating system is read in from these system tracks. Additionally, with each warm boot, initiated by the user with a control-C command or automatically by some programs when they are done, the command processor code (and generally the DOS as well) are reloaded into memory from the system tracks.

With the next track inside the system tracks begins what is called the directory of the disk. Like the index of a book, the directory has entries for each file stored on the disk containing such information as the name of the file and where on the disk the data for that file are stored. The directory has a fixed size and can accommodate only a fixed number of entries (that is why if you have many small files, you can sometimes get a "disk full" message even when there appears to be plenty of room on the disk). The rest of the disk is devoted to the storage of the data for files.

Now let's look at what DOS does when it wants to read a file. First it has to find out where the file is stored, so it moves the drive heads to the directory. Since the directory, unlike the index of a book, is not sorted, DOS must start at the beginning and read through the directory until it finds the entry for the file it wants. Once it knows where to go on the diskette to find the data, it moves the heads to the proper track and reads the data. This head movement or seeking, as it is called, can take a long time, especially with older, slower floppies. Even a fast, modern drive that steps at a rate of six milliseconds per track will take nearly half a second to seek from the directory track to the innermost track on an 80-track drive. Clearly files located near the directory track will load much faster than those located near the inside of the diskette. And files with directory entries near the beginning of the directory will have a slight edge over those with their entries at the end.

I think you can now see what we should do to speed things up. But two further questions face us: 1) which are the files whose loading times are critical to the performance of a Z-System, and 2) how can one get the operating system to a) place directory entries for these files at the beginning of the directory and b) place the data for these files on tracks near the directory?

The second question is easy to answer. Whenever DOS writes a file out to disk, it uses the first free entry in the directory and places the data in the outermost available space on the disk. After a diskette has been used for a while and files have been written and erased many times, directory entries and file data can be rather disordered. However, if one starts with a freshly formatted disk (with only the operating sysgen'ed onto it), files copied to it will fill the directory and data spaces exactly in order.

Now comes the question as to which files should be written to the fresh disk first. The general answer is: the ones that are accessed most often. In a Z-System one must be careful when trying to answer this question because there are files that are accessed automatically by the system, and the user may not even be aware of them. Here are my suggestions for files to put on the list. The exact order in which they are copied to the disk is not that critical.

If you are like me and have been known to make typing mistakes when entering commands, then you will want the ZCPR3 error handler to be called in so that you can correct the mistake. Put your favorite error handler on the list (mine is VERROR). If you use an extended command processor (more on that subject in a future column), you should put it and its associated files on the list. I use ARUNZ, a program that enables one to store hundreds of alias scripts in a single, small text file, ALIAS.CMD. Whenever a command cannot be executed either as a built-in command (in the FCP, CPR, or RCP) or as a transient (COM file), then ARUNZ is invoked, It tries to find the given command name in ALIAS.CMD. Since these operations are performed often, I include ALIAS.CMD and

ARUNZ.COM on my list of files. Another popular extended command processor is LX (or LRUNZ), a program that extracts programs from a library (COMMAND.LBR) of COM files. If you use LX, then put LX.COM and COMMAND.LBR on your list.

The next category of programs to consider is shells and their associated files. I make frequent use of the history shell, HSH, with its command line history file HSH.VAR. It allows one to perform complete editing of the command line, just as if one were entering the line with WordStar. In addition, it lets one recall commands issued previously (even on a previous day!), edit them as desired, and run them again. What a pleasure to run a system like this! Another command-line shell, VCED (for video command line editor) performs similarly, and the choice between them is a matter of personal preference. Both of them can really slow down the operation of a floppy-based system if not placed on the diskette as described above. Other commonly used shells are VFILER (with the VFILER.CMD macro file) and MENU and VMENU (with their menu files MENU.MNU and MENU.VMN). If you use any of them, add them to the list.

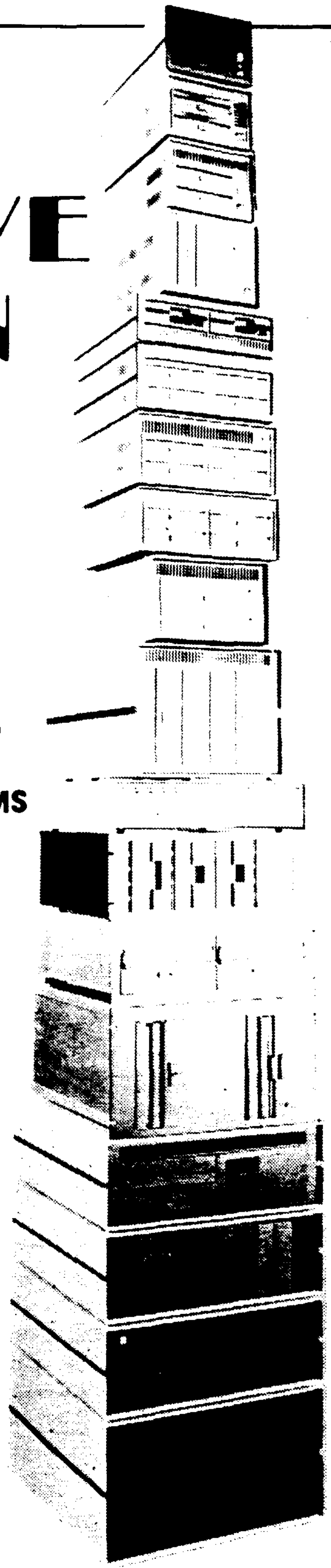
The last category of files I include on the list is files that I invoke manually very often and want to see the results of fast. This especially includes directory programs (SD and XD, for example). I also hate to wait for my text editor (PMATE), so it is on my list.

Let me finish by mentioning some files that I do not include on the list. Obviously I don't include files I rarely use, such as the system segments (SYS.RCP, SYS.ENV, etc.) that are used rarely except at coldboot time. I also do not include programs that take a fairly long time to run anyway or which are generally run in batch mode. I put assemblers and linkers in this category. With them, most of the time is spent actually computing or reading and writing the files on which they operate. Who cares if the assembler loads in 2 rather than 5 seconds if the assembly takes 20 seconds or more anyway. I would probably also leave off the list programs that I tend to spend a lot of time in once they are loaded. Turbo Pascal is an example. Since it has a built-in editor and keeps the source code and object code in memory simultaneously, one generally lives inside Turbo while developing a program. Batch programs, like SUB, SUBMIT, or ZEX, would not be on my list. Since they initiate a long, time-consuming sequence of events anyway, why be in such a hurry to get things started.

The mention of ZEX reminds me of one final hint on how to expedite the use of this speed-up technique. I make a batch file for ZEX (you could use SUBMIT just as well) with all the copy commands in the order in which I want the files to be on the new disk. The script for a MAKESYS.ZEX file to be run from the A drive to make a new system diskette on drive B might look something like

```
MCOPY B:=HSH.*
GO B:=ARUNZ.COM,ALIAS.CMD
GO B:=VERROR.COM
GO B:=VFILER.*
GO B:=SD.COM,XD.COM
GO B:=*.* N
GO B1:=1:*.* N
```

If you leave out a file and have to repeat the process, it is much easier to edit the batch file than to enter manually all the copy commands again, especially since you'll probably accidentally skip another file and have to repeat the procedure still again! The last two lines copy all the rest of the files in user areas 0 and



DRIVE INN

Enclosure & power supplies for
FLOPPY, WINCHESTER, TAPE DRIVES, SINGLE BOARD COMPUTERS & S-100 SYSTEMS
8 inch
5 inch
3 inch

CUSTOMIZING AVAILABLE

Call or write for free catalogs & application assistance

INTEGRAND
RESEARCH CORPORATION
8620 Roosevelt Ave. • Visalia, CA 93291
209/651-1203
TELEX 5106012830 (INTEGRAND UD)
EZLINK 62926572
We accept BankAmericard/Visa and MasterCharge

1; the 'no replace' option 'N' makes MCOPY skip over any files already on the destination directory. If you have files in other user areas, add lines for them, too.

New Program Suggestions

We now turn to the promised subject for this month's column — suggestions for new programs or program enhancements. I hope some readers will decide to take a shot at writing these programs and submitting them to ZSIG. If so, I recommend that you let me know in one of the ways (including writing to me) described in my column in the last issue of The Computer Journal. That way we won't have two versions of a program that have to be integrated. Those of you who perhaps do not have the assembly language programming knowledge can still contribute by sending in further suggestions and ideas. As I said in the last column, the real ingenuity in software is not so much in the actual coding as in the conceiving of new program ideas.

SETPATH

The path feature of ZCPR3, which allows a COM file to be loaded automatically from a drive and user area other than the one from which the command was issued, is probably the most used (and least noticed) feature. The utility used to support this capability, PATH.COM, operates in only two ways. When invoked with no command tail, it displays the search path currently in effect, listing each path element symbolically (possibly with '$' characters for current drive or user), in DU format, and in named directory format. When invoked with a command tail, the tail is interpreted as a new search path, replacing the old one.

CACHE22+CP/M 2.2 = CP/M Max!

CACHE22 is a front-end system program that buries all of CP/M 2.2 in banked memory. It helps 8080/Z80 computers to survive by providing up to 63.25K of TPA, plus the ability to speed disk operations, eliminate system tracks, and run Sidekick-style software without loss of transient program space. Complete source and installation manual, $50.00.

CP/M is a trademark of Digital Research Inc.
Sidekick is a trademark of Borland International

MIKEN OPTICAL COMPANY
53 Abbett Avenue, Morristown, NJ 07960
(201) 267-1210

I have often wanted to make simple changes to the path, such as adding a new element at the beginning or end of the path. Later I want to remove it. With the present PATH command one has to enter a complete new path specification each time. I would propose a new program, which might be called SETPATH, that would offer more flexible options. The syntax for SETPATH would be as follows:

```
SETPATH [/option element-list] [/option element-list] ...
```

The following option specifiers would be recognized:

C — clear existing path and append elements listed
A — append path element list to existing path
P — prefix path element list to beginning of existing path
D or R — delete (remove) listed elements from existing path

After the new path is built and loaded, the final path would be displayed. Option 'C' would be the default option if no option is given explicitly, and thus SETPATH would work the same as PATH when used the same way. Just to make sure that the syntax is clear, here is an example SETPATH command:

```
SETPATH /A NEW1 NEW2 /D OLD2 /P NEW3
```

If the old path had been "OLD1 OLD2", then the new path would be "NEW3 OLD1 NEW1 NEW2."

There are some technical details to consider. The new path should be built up in a temporary buffer and its length checked only at the end, since additions made before deletions might make the path temporarily longer than allowed. If the final path is too long, one should probably leave the path as it was, display an error message, and ring the console bell. One might also want to set the ZCPR3 program error flag (and clear it when the command succeeds). For security reasons, named path elements must be checked for passwords, and drive/user path elements must be checked against the maximum drive and user values specified in the environment. Naturally, like PATH, the entire utility should work only if the wheel byte is on.

To be complete and thorough, one might want to recognize a leading pair of slash characters not as an option but as a single slash in a path element name. Otherwise one would have no way to use a named directory whose name begins with a slash. The first version of SETPATH could omit this feature, since it is not critical. A command tail with '//' or '/' only should display a built-in help message.

There is an issue with the 'D' and 'R' options. In some cases an element may appear more than once in a path. My default path, for example, is 'SYS ASM MODEM SYS'. SYS appears both at the beginning, so that it is searched first, and at the end, so that programs that access the root element will also use SYS. My choice would be to make the 'D' and 'R' options delete only one occurrence of an element. Perhaps 'D' could delete starting at the beginning of the path and 'R' could remove starting from the end of the path. Usually, of course, the path element would appear only once, and both options would give the same result. I'm not completely sure what should be done when the element to be deleted is not found. The path should still be built, and an error message should probably be displayed, but the ZCPR3 program error flag should probably not be set.

SETPATH, as described here, would be fairly easy to write, since it could borrow most of its code from the present PATH command. An alternative enhanced PATH command, a video program which might be called VPATH, would display the path in a full-screen format and allow the user to edit the path, inserting and deleting elements using cursor controls and editing commands. My present feeling is that such a command would be overkill. SETPATH would be adequate and would have the advantage that, not requiring interactive input, it could be run from batch files and aliases.

SETNDR

The named directory facility in ZCPR3 can be very convenient. I find it handy to put all my assembly language tools in a directory called ASM and my modem related programs in a directory called MODEM. It is much easier to remember the names than the particular user numbers chosen. Some programs make automatic use of named directories. The HELP program, for example, looks in a directory named HELP for the file containing the help information. Echelon's disk cataloguing program DISCAT uses a directory named CAT to keep the catalogues and a directory BACKUP to determine the drive holding the diskette to be catalogued.

A program similar in concept to SETPATH, which might be called SETNDR, would be useful for making simple manual additions, deletions, or changes to the named directory register (NDR). For example, as mentioned above, DISCAT catalogues diskettes in the drive containing a directory named BACKUP. In order to make DISCAT work with a different drive, one would normally have to have or make another NDR file and load it with LDR. SETNDR would be much easier to use.

SETNDR could be a very simple, short program with the following syntax:

```
SETNDR DU: — remove name associated with area DU:
SETNDR DIR: — remove directory with name DIR
SETNDR DU:NAME — assign NAME to area DU:
SETNDR DIR:NAME — change directory DIR to NAME
```

A second name on the command line could be used to assign a password to the directory.

```
SETNDR DU:NAME PASS — define new name and password
SETNDR DIR:NAME PASS — assign new name and password
```

Before assigning a name to a drive/user area, the program should check to make sure that the name is not already assigned to a different DU. If it is, there are three reasonable possibilities.

1. The program could display an error message and refuse to make the change;
2. The program could delete the existing assignment, provided it is to a user-accessible area (i.e., not password protected or beyond the maximum drive and user values specified by the environment); or
3. The program could report the situation to the user and prompt as to whether the existing assignment should be deleted (again provided the assignment to be deleted is to a user-accessible area).

My preference is the second choice, but a message should be displayed reporting the assignment that has been deleted in case it was unintentional.

Like SETPATH, this program should be fairly easy to code. The parser built into ZCPR3 would do almost all of the work required to interpret the command line. The programmer would only have to detect the absence of any command tail or one with only '//'. In these cases a built-in help screen should be displayed. Most security issues would also be handled by the ZCPR3 command-line parser. However, the program code should check the wheel byte and prohibit changes if it is not set.

If the ZCPR3 parser is used, some possibly erroneous input would pass undetected. For example, the command lines

```
SETNDR DU:NEWNAME DIR:
SETNDR DU:NEWNAME U:
```

would ignore the password field (since the parser would see no file name). Also, a command like

```
SETNDR NEWNAME
```

will assign the new name to the currently logged-in area. The latter is probably acceptable (perhaps desirable). In any case, if one wants to control these cases, the code would have to double check the actual command line tail and not rely only on the default file control blocks.

There is already a utility called LDSK (load disk) for automatically setting up named directories after one switches diskettes in a floppy drive. If a user area on the diskette contains a file whose name begins with a hyphen, then LDSK assigns that name (not including the hyphen) to that area. LDSK might have some code one could borrow for SETNDR.

• Z Best Sellers •

Z-COM (7 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 70+ utility programs and **ZCPR3: The Manual**.

Z-Tools (4 disks) $150.00
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!

TERM III (6 disks) $99.00*
TERM III has been called "a cosmos". Why? Because it's 29 programs, featuring T3MASTER and T3SERVER, are galactic in scope. KERMIT, XMODEM, 1k XMODEM, and batch YMODEM protocols supported.

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.

*ZCPR3 required.

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)

Enhanced IF.COM

Conditional command processing, though rarely invoked directly in user entered commands, gives ZCPR3 alias and batch scripts tremendous power and flexibility. Scripts can test such conditions as:

- whether a previously run program has set or cleared the program error flag;
- whether a parameter value has been specified or omitted in an alias invocation;
- whether a specified file exists;
- whether the file not only exists but has non-zero length;
- whether one of the ten user registers contains a specified value;
- whether the wheel byte is set or clear; or
- whether or not a terminal definition (TCAP) is loaded.

Some of the conditional testing is performed with the ZCPR3 resident FCP (flow command package). Optionally, the FCP can pass on conditional testing to the transient program IF.COM. The user can also force invocation of the transient IF processor by including a directory specification with the command, such as "A0:IF..." or even just ":IF...". Since IF.COM does not have to be resident and permanently reduce the memory available for program operation, it can be a bigger and more capable program. I will mention here some enhancements that would be useful in IF.COM.

Current IF processing allows one to determine the presence or absence of a parameter with the NULL option. Howard Goldstein has requested an ambiguity option:

```
IF AMBIG AFN
```

This would allow an alias script to determine if a parameter passed to it was ambiguous, so that an error message could be echoed instead of passing an ambiguous file specification to a program that requires an unambiguous file name.

The current IF processing allows one to determine the existence and non-zero size of files with the EXIST and EMPTY options. It would be useful to be able to test the attributes of files, as in

```
IF ARCHIVE AFN — do files have archive bit set
IF RO AFN — are files read-only
IF RW AFN — are files read-write
IF WP AFN — are file wheel protected
IF SYS AFN — are files of system (SYS) type
IF DIR AFN — are files of directory (DIR) type
```

The current version of IF.COM allows a test of the form

```
IF AFN1=AFN2
```

which compares two possibly ambiguous file specifications. It can be used fairly generally to compare command line tokens, as in the following VFILER.CMD macro script:

```
IF %PT=LBR;LDIR %$;ELSE;ECHO NOT LBR FILE;FI
```

VFILER replaces the parameter %PT with the type of the file currently pointed to. If the pointed-to file is a library, its directory will be displayed; otherwise an error message will be echoed.

There have been a number of times when I have wanted to test inequalities. ARUNZ alias scripts can read values from the user registers and from memory locations. Sometimes one would like to test for values that are less than or greater than some other value. Thus it would be handy to have tests other than just equality. Equality is the easiest to code because the ZCPR3 command line parser already handles the equal sign (since it is used in commands such as REN and CP). For the extended comparisons, I would propose a syntax of the form

```
IF TOKEN1 TOKEN2 RELATION
```

The first two tokens (words) on the command line will be parsed by the ZCPR3 command processor and placed into the two default file control blocks at 5Ch and 6Ch. The IF.COM code would have to scan the command tail saved starting at memory location 80h to see if there is a third token. The following tokens could be recognized:

```
EQ or = — token1 same as token2
NE or < > — token1 not same as token2
LT or < — token1 less than token2
LE or < = — token1 less than or same as token2
GT or > — token1 greater than token2
GE or > = — token1 greater than or same as token2
```

Any wildcard characters in either token would be taken as equal to any corresponding character in the other token. Thus "ABC*" would be equal to "?BC", since the "A" in the first position matches the "?" in the second token and the "*" in the first token matches all the blank spaces in the last five character positions in the second token.

I think these ideas should be enough to keep you busy for a while! I have several possible subjects for my next column but will await your response before deciding what to cover. Please send in your suggestions. ■

Further details on program submission are on page 43 in issue #25 of TCJ. Jay can be contacted by modem on his Newton Centre Z-node #3 at 312-649-1730, or by mail at 1435 Centre St., Newton Centre, MA 02159.

Affordable C Compilers

by Don Howes

One of the major problems with learning the C language is the price of most of the available C compilers. Prices for a full featured compiler can be up to 500 dollars (or more), so what can a person who wishes to explore the language without making a major software investment do?

One thing they can do is take a look at one of the three compilers I will be reviewing here. These compilers are; Eco-C88 by Ecosoft, Datalight C by Datalight, and Mix C by Mix Software (see Table 1 for pricing and vendor information). The nice thing about all of these compilers is that they are priced under one hundred dollars each (that's right!). In fact, you can buy all three of them for $158.90, but that may be carrying things a little far.

Table 1: Vendor Information.

Datalight
11557 Eighth Avenue, NE
Seattle, WA 98125
(206) 367-1803
Datalight C $69.00
Datalight C Developers Kit $99.00

Ecosoft Inc.
6413 North College Avenue
Indianapolis, IN 46220
(317) 255-6476
Eco-C88 $59.95

Mix Software, Inc.
2116 East Arapaho, Suite 363
Richardson, TX 75081
(214) 783-6001
Mix C $39.95

The Benchmark Suite

Rather than focus on one or two benchmarks of arguable validity, I have used a total of ten small programs each designed to examine one restricted aspect of a compiler's performance. Also, knowing that it would be impossible to ignore, I ran a vanilla version of the venerable Sieve of Eratosthenes (no register variables or pointers). All the benchmarks were executed on an IBM compatible computer running at a clock speed of 4.77-MHz, without an 8087. All benchmarks were loaded and executed from a hard disk, except for cpychar, which was run independently on a floppy disk. Times reported for the benchmarks were obtained using a digital stopwatch, with most benchmarks being timed five times and the reported time found by averaging the total time. The exceptions to the five trials are float and double, since the run times for these benchmarks were very long. These two benchmarks were timed three times each. Although the stopwatch tracked to 1/100th of a second, this is well beyond my reaction time, so I will be reporting the times to the nearest 1/10th of a second. This should be accurate to a tenth of a second either way. Also, I consciously made the decision to compile the benchmarks to what may be termed "the lowest common denominator." Although both Datalight and Mix offer methods to optimize the code to some degree, this wasn't done for these benchmark timings. Be warned therefore, that the reported times should be viewed as worst case scenarios for those two compilers and some tweaking with the compiler switches or additional programing may increase the execution speed.

After saying that, let's take a short look at the benchmark suite. The benchmarks can be broken into two basic types, those which test the code size generated by a compiler and those which exercise various aspects of the compiler and the run-time library.

Code size benchmarks are represented by min, minputs and minprt. min compiled a minimum size C program to measure the amount of compiler generated startup code linked into every program. minputs looks at additional code added by a call to the run-time library, while minprt provides a measure of the amount of code necessary to use the printf() formatted print function. None of these three benchmarks were executed.

Of the executable benchmarks, the simplest was loop. This program consisted of two empty, nested for loops, the outer loop executing twenty times and the inner loop two thousand times. This benchmark served two purposes. First, it was a measure of the capability of the compiler to implement loop constructs and, second, it was the same loop structure used to control the integer, float and double benchmarks. The times for loop can be subtracted from those benchmark times if you wish (this wasn't done with the times as reported here).

The benchmarks for integer, float and double are considered together, since they vary only in the declaration of variables. Each of these benchmarks used the double loop structure mentioned above, with each program performing a total of eighty thousand additions and forty thousand divisions. The times for these benchmarks measure how each compiler handled the different variable types in simple arithmetic.

The trig benchmark is designed to provide a measure of the compilers ability to work with trancendental functions. In this benchmark, a total of 720 function calls were made, evenly divided between sine, cosine and tangent calls. The value passed to the function was given in radians.

The screen benchmark examined the compilers ability to write to 'stdout' using putchar(). A total of 1600 characters were written to the console by this benchmark. Each run of the benchmark started from a blank screen, so that scrolling did not have to be taken into account.

The cpychar benchmark performed an unbuffered character oriented file transfer using a 1058 byte C source code file as its test file. This benchmark was run on a freshly formatted floppy disk, so that sector allocation in the file copy could be controlled. Using an empty disk allowed the file to be copied in contiguous sectors and provided the minimum copy time.

The sieve benchmark finds prime numbers using the Sieve of Eratosthenes method. The benchmark did ten iterations of the sieve.

Benchmark Results

Results of the code size benchmarks are given in Table 2 and the execution benchmarks in Table 3. Only two of the benchmarks could not be compiled. The Datalight compiler could not compile the float benchmark, generating a compiler error message of "DLC2 bug: TH2 in main." The Mix compiler was not able to compile the trig benchmark, since the compiler does not have a tangent function (strange, since sine and cosine functions are present!). Also, code sizes for the Mix compiler are reported with and without linking in the runtime package. If the runtime support is linked in, the program can run as a standalone program. If not, the runtime overlay package must be present for the program to operate.

Editor's Note: I called Roy Sherrill at Datalight, and he advised that the problem was caused where float.c *increments a float. Roy said that this will be fixed in the next revision.*

Both Ecosoft and Datalight have reasonably small amounts of startup code linked in the min benchmark (see Table 2). The thing to note here is the large increase in minimum code size when a single call to printf() is made, due to the necessity of the compiler linking in code to handle both integer and floating point variables. I find it hard to compare the Mix compiler in these code size tests, but the code bloat that occurs when the run time package is linked in is very evident.

Table 2. Code Size Benchmarks.

	Ecosoft	Datalight	Mix
Min	1536	2674	2927/21714
Minputs	2710	4098	3074/21861
Minprt	8820	9090	4540/23327

Table 3. Execution Benchmarks
(time in seconds or minutes:seconds)

	Ecosoft	Datalight	Mix
Loop	0.9	0.8	12.6
Integer	3.1	3.1	26.4
Float	5:58.7	----	5:35.2
Double	5:21.0	3:25.9	5:07.2
Trig	29.7	26.2	----
Sieve	10.8	10.2	3:01.0
Screen	3.6	3.8	4.8
Cpychar	8.7	8.2	9.3

In the timing benchmarks two things stand out. First, the very good performance of the Datalight compiler in the double benchmark, taking approximately 2/3 the time of Ecosoft and Mix, as well as its smaller, but still significant, edge in the trig benchmark. The second thing is the woeful times returned by the Mix compiler for the loop, integer and sieve benchmarks. I find it hard to believe that any compiler generates such bad code for what are the most common of operations and calculations (loops and integer arithmetic). I was very surprised, therefore, by the good showing of this compiler in the float and double benchmarks. Apparently, the slow integer arithmetic times are due to bad compiler design.

As a note to those who may be wondering, the float benchmark will always time out slightly longer than the double, since a variable declared as a float is promoted to a double before being

Benchmark Listings

```
/*min.c*/
main()
{
}

/*minputs.c*/
main()
{
    puts("This is a string.");
}

/*minprt.c*/
main()
{
    printf("This is a string.");
{

\*loop.c*\
main()
{
    int i,j;
    puts("start");
    for (i=0;i<20;++i)
    {
        for(j=0;j<2000;++j)
            ;
    }
    puts("stop");
}

/*integer.c*/
main()
{
    int i,j,k,l;
    j=1;
    k=2;
    puts("start");
    for (l=0;l<20;++l)
    {
        for(i=0;i<2000;++i)
        {
            k=(j+1)/j+i;
        }
    }
    puts("stop");
}

/*float.c*/
main()
{
    float i,j,k,l;
    j=1.0;
    k=2.0;
    puts("start");
    for (l=0;l<20;l=l+1)
    {
        for(i=0;i<2000;++i)
        {
            k=(j+1)/j+i;
        }
    }
    puts("stop");
}

/*double.c*/
main()
{
    double i,j,k,l;
    j=1.0;
    k=2.0;
    puts("start");
    for (l=0;l<20;++l)
    {
        for(i=0;i<2000;++i)
        {
            k=(j+1)/j+i;
        }
    }
    puts("stop");
}

/*trig.c*/
main()
{
    int x;
    double i, j, k, l;
    double sin(), cos(), tan();
    puts("start");
    for (x=0;x<20;++x)
    {
            i = sin(0.392699);
            j = sin(0.785398);
            k = sin(1.178097);
            l = sin(1.963495);
            i = sin(2.356194);
```

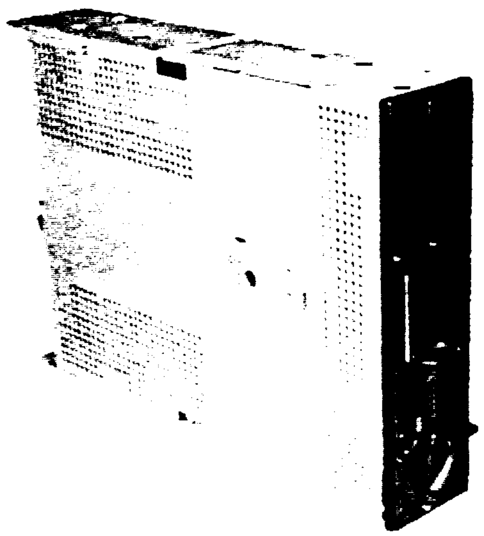
used in an arithmetic expression. This promotion, and subsequent demotion when being stored, adds extra overhead to the program and causes it to run slightly slower. It's for this reason that I'm not very upset about the Datalight compiler not being able to compile the float benchmark. In situations where you wish to do a lot of floating point operations (graphics, for instance) it is better to declare the relevant variables as double. This, of course, displays my own bias for sacrificing storage space in favour of execution time.

Datalight C

This may be the best C compiler buy present today. Two versions of the compiler are available, a $69.00 version containing code which supports the small memory model and the $99.00 Developers Kit, which has three additional memory models. Other than the additional memory models, both versions of the compiler are the same. I personally would recommend spending the additional thirty dollars for the Developers Kit. You would have a compiler capable of doing virtually any job, for a fraction of the price of any other multiple memory model compiler.

Documentation comes packaged in a 3-ring binder, in a 5½ × 8½ page size format. The documentation totals 216 pages, complete with a table of contents and a good index. There is no language reference bundled with the documentation, so have other reference materials on hand for learning the language. In general, I found the layout of the documentation to be good and easy to read. Library functions are listed one to a page in alphabetical order, so things will be easy to find. However, there are no code fragments given with each function demonstrating how the function should be used, nor are there any programming examples in the documentation.

Loading the compiler onto my hard disk was relatively easy. Two batch files are provided, one for setting up the compiler to run from floppy disks and the other to run from a hard drive. I didn't test out the floppy installation procedures, but I did find two problems in the installation on the hard drive. First, the batch file did not properly specify the "set" variables in its creation of an AUTOEXEC.BAT file. This caused the AUTOEXEC file to fail. Second, the DLC.COM file was not loaded to the hard disk. Since this is Datalight's version of the CC driver which oversees the compile and link process, this was a significant (but easily correctable) oversight.



SUPER DRIVES!!

8″ SHUGART ½ Height DSDD

MODEL 860-1 MINT CONDITION WITH 1 FULL YEAR WARRANTY. THESE DRIVES DO NOT REQUIRE AC, TRACK TO TRACK ACCESS TIME 3MS, CAPACITY IS 1.6 MB.

CEC **$279.00**
P.O. Box 1965
Burlingame, CA 94010
(800) 228-3411

Once everything was loaded, I found the compile and link process using this compiler to be very straight forward. Datalight C is composed of two modules, with each module probably making more than one pass through the source code (the total number of passes isn't documented). One touch that I liked is that the compile time for each module is reported when the module is exited. Command line switches on DLC allow for the creation of either EXE or COM files, as well as files using integer only code or code which will support both integer and floating point arithmetic. The ability to specify integer only support will create a significant decrease in the size of a compiled module. In addition, the compiler supports the creation of ROM-able code through the use of a DLC command line switch, as well as providing support for third party debuggers with the ability to specify the output of line numbers to object files created by the compiler.

Provided with the compiler as separate programs are versions of MAKE and TOUCH which allow for the automatic compiling and linking of large, complex programs when only a few of the programs modules have been modified. Datalight also provides the assembler source code for the compiler startup code, which you can modify if needed to suit your environment.

Compiler features include full K&R compatibility as well as extensions which match some of the likely changes to come out of the ANSI X3J11 C Standards Committee. These are; enum and void data types, structure and union assignment and, structure passing to, and return from, function calls.

In all, I found this to be a compiler with almost no flaws. It produces fast code and compiles quickly. If the documentation would include examples of function use along with each function description, and the company fixes the floating point compiler bug, that would take care of just about all my wishes. This is an excellent buy.

Ecosoft Eco-C88

Documentation for the Ecosoft compiler comes bound in a 5½ × 8½ paperback book. The manual totals 172 pages, including documentation for the CED text editor (which Ecosoft is currently including in the price of the compiler). The manual has a table of contents and an index. I found the documentation somewhat hard to work with, due to the small size of the manual. It would be more functional for Ecosoft to package the documentation in a 3-ring binder.

There is no language reference included in the documentation, although there is a useful chapter on common C programming errors which should help the novice C programmer avoid some of the pitfalls which tripped up the rest of us. Ecosoft recommends the "C Programming Guide" by Jack Purdum as a reference. Error messages returned in the CED editor are keyed to this text by page number. Library functions are presented in alphabetical order, with multiple functions being present on a single page. The chapter containing the function descriptions starts with a listing of all the library functions grouped by functional category, which will assist a new user in locating a particular type of function. No code fragments demonstrating the use of each function are given along with the function description, although there is a chapter giving some programming examples (mainly to do with compiler specific functions).

The compiler and its associated files can be loaded to run either from a hard disk or from floppy disks, using one of two batch files for setup. I encountered no problems with installing the software on my hard drive and found the installation instructions to be clear and complete.

```
            j = sin(2.748893);
            k = sin(3.534292);
            l = sin(3.926991);
            i = sin(4.319690);
            j = sin(5.105088);
            k = sin(5.497787);
            l = sin(5.890486);
            i = cos(0.392699);
            j = cos(0.785398);
            k = cos(1.178097);
            l = cos(1.963495);
            i = cos(2.356194);
            j = cos(2.748893);
            k = cos(3.534292);
            l = cos(3.926991);
            i = cos(4.319690);
            j = cos(5.105088);
            k = cos(5.497787);
            l = cos(5.890486);
            i = tan(0.392699);
            j = tan(0.785398);
            k = tan(1.178097);
            l = tan(1.963495);
            i = tan(2.356194);
            j = tan(2.748893);
            k = tan(3.534292);
            l = tan(3.926991);
            i = tan(4.319690);
            j = tan(5.105088);
            k = tan(5.497787);
            l = tan(5.890486);
    }
    puts("stop");
}

/*sieve.c*/
#include "stdio.h"
#define true 1
#define false 0
#define size 8190
#define sizepl 8191
char flags[sizepl];
main()
{
    int i,prime,k,count,iter;
    printf("10 iterations\n");
    for (iter = 1;iter <= 10;iter++)
    {
        count = 0;
        for (i=0;i<=size;i++)
            flags[i] = true;
        for (i=0;i<=size;i++)
        {
            if (flags[i])
            {
                prime = i+i+3;
                k = i+prime;
                while(k <= size)
                {
```

The Ecosoft compiler is a four pass compiler (preprocessor / compiler, optimizer, code generator and assembler) which supports the small memory model only. Command line switches for the CC driver allow access to a version of make (which Ecosoft terms "mini-make") and to a "lint"-like compiler utility which will check for semantic errors in the source code. Use of this option allows for the development of code which can be easily transported to other systems.

As I mentioned above, Ecosoft is currently shipping their compiler with the CED text editor (normally available as an additional item). This editor is designed to work with the compiler to form a complete development environment, and I found it to work very smoothly. The editor is modelled closely after the Turbo Pascal editor and allows the user to write code and perform the compilation/debug process all from within the editor. Using the "-ed" switch when compiling caused the compiler to create an error file which is used by the editor. If errors are found during compilation the editor returns to its entry screen, with the cursor on the first line containing an error (there is an associated error message). All errors are flagged and you can move through the file fixing things as you go. I found this to be a very good development environment which significantly eases the pain normally associated with debugging C code.

Compiler features include K&R compatibility (with the exception of bit fields) plus X3J11 extensions. These include; structure passing and return from function calls, structure assignment, enum and void data types and prototyping. In addition, the source code for the CC driver is included, so you can modify the driver to suit your development environment.

In all I like this compiler and have spent quite a bit of time developing code on it. Together with its editor, it makes a development environment which is hard to beat. If Ecosoft supported additional memory models for the development of larger systems there would be nothing more you would need.

Mix C

Unfortunately, Mix C can best be described as documentation in search of a compiler. The documentation shipped with the compiler is impressive. It comes bound as an 8½ × 11 paperback volume which I found a little large to work with around my computer, but is big enough that it will stay open fairly easily. All together, the documentation totals 433 pages, divided into five sections (Getting Started, Tutorial, Reference Manual, Functions and Tools). There is no overall table of contents or index, but each section has its own detailed table of contents and index associated with it. This may make it a little hard to find a specific topic. The Tutorial and Reference Manual sections combine to form a comprehensive language reference, with numerous programming examples. The Functions section has the library functions divided between five different chapters and the functions are listed in alphabetical order within each chapter. I found that this made for a lot of flipping of pages, trying to find the right chapter for a function description. There may be multiple function descriptions present on a single page, but each description is accompanied by a code fragment demonstrating the use of the function. All in all, the documentation itself is almost worth the price of the compiler.

I believe that the compiler was first written for CP/M, and still shows its origins. Unlike the other two compilers examined here, Mix C does not have any batch files (or indeed any real instructions) for loading the compiler and its associated files onto either a floppy system or a hard disk. In addition, the compiler expects files to be present either on the default directory of the

DISK DRIVE SERVICE

5¼"	$35
8"	$45
SERVICE SPECIALS	
Apple II Drives	$30
Shugart SA 400/400L	$25
Shugart SA 800/801	$25
Shugart SA 850/851	$35
DRIVES FOR SALE	
Shugart SA 800-2 (wide frame)	$59
Shugart SA 850 (wide frame)	$99
MPI 52S 5¼" DS/DD full ht.	$55
Tandon 100-2 DS/DD full ht.	$70
Tandon 100-1 SS/DD full ht. (new)	$60
Apple II Drives	$85
Genuine "IBM" (PC) floppy contr.	$60

60 day warranty on all drives and service. Turnaround time usually 24-48 hours. Trade-in available for drives too costly to repair. Prices do not include parts or shipping. If parts are more than $20 we get permission before repairing. Units returned UPS COD unless otherwise requested. All drives for sale are reconditioned unless otherwise noted and documentation is included.

LDL ELECTRONICS

13392 158 St. N., Jupiter, FL 33478 (305) 747-7384

Surplus Parts Resource

Here's a catalog any serious computer tinkerer needs. It's a treasure-trove of stepper motors, gear motors, bearings, gears, power supplies, lab items, parts and pieces of mechanical and electrical assemblies, science doo-dads, goofy things, plus project boxes, lamps, lights, switches, computer furniture, and stuff you might have never realized you needed.

All at deep discounts cause they are surplus!

Published every couple of months, and consecutive issues are completely different. Send $1.00 for next three issues.

JERRYCO, INC. 601 Linden Place, Evanston, Illinois 60202 hard drive or on drive A. If you wish to change this, you must patch the compiler using DEBUG (the necessary patch procedure is given in a read.me file). I find this to be a rather archaic and unnecessary procedure. The compiler is not well suited for the DOS environment and apparently does not make use of information kept in the environment table.

Once everything is loaded the compilation process is fairly straight forward. However, the compiler produces only COM files, which I take to mean that only the 8080 memory model is supported (code and data must fit in 64K). Since this isn't documented, I may be wrong. One thing I found is that the CC driver does not have any command line switches. Instead, compiler options are passed to the compiler as source code comments using the special form "/*$compiler_option*/". This is a format similar to that used by Modula-2, but is very non-standard for C compilers. As well, Mix C does not use the Microsoft linker, but rather, a proprietary linker of their own. Since the format of this linker is different from Microsoft's, compiler output is also different and object files created for use with the Microsoft linker (by other compilers or assemblers) cannot be linked using the Mix linker. I found the linker to be rather awkward to use, since the default mode of the linker is to not link in the run time overlay package. Since I generally wish to be able to create stand alone programs, this meant that for each program I wished to link, I had to go through a menu version of the linker answering questions (no command line switches). This got tiring very quickly.

Included with the compiler are two separate programs designed to perform optimization of the compiler output before it is passed to the linker. The first of these is SHRINK which optimizes for space and the second is SPEEDUP which optimizes for speed.

CTM

"...received my moneys worth with just one issue..."
—J. Trenbick

"...always stop to read CTM, even though most other magazines I receive (and write for) only get cursory examination..."
—Fred Blechman, K6UGT

U.S.A.	$15.00 for 1 year
Mexico, Canada	$25.00
Foreign	$35.00(land) - $55.00(air)
(U.S. funds only)	
Permanent (U.S. Subscription)	$100.00
Sample Copy	$3.50

CHET LAMBERT, W4WDR
1704 Sam Drive • Birmingham, AL 35235
(205) 854-0271

Benchmark Listings Continued

```
                        flags[k] = false;
                        k += prime;
                    }
                    count = count + 1;
                }
            }
        }
        printf("\n%d primes",count);
}

/*screen.c*/
#include  "stdio.h"
main()
{
        int i;
        puts("start");
        for (i=0;i<1600;++i)
            putchar('.');
        puts("stop");
}

/*cpychar.c*/
#include    "stdio.h"
main(argc,argv)
int argc;
char *argv[];
{
        int c;
        FILE *fp1, *fp2, *fopen();
        puts("start");
        fp1 = fopen(argv[1],"r");
        fp2 = fopen(argv[2],"w");
        while ((c = getc(fp1)) != EOF)
            putc(c,fp2);
        fclose(fp1);
        fclose(fp2);
        puts("stop");
}
```

The compiler is K&R compatible, including bit fields. The only X3J11 extension included in the compiler is structure assignment.

If this compiler lived up to its documentation it would be a good buy, unfortunately, it does not. The compiler still appears to be in transition from CP/M to MSDOS and does not function well in the DOS environment. Until the compiler is significantly upgraded, I can't recommend it.

So Now What?

Here is where I go out on a limb and give some recommendations about what I think would make a good investment of your money. Of the three compilers I've looked at, either Ecosoft or Datalight (especially the Developers Kit) would be excellent buys. If you are looking for a good, streamlined development environment and don't need additional memory models, then the Ecosoft compiler would be the one to buy. If you want a compiler which is clean, with good documentation and is undoubtedly the cheapest multiple memory model compiler around, then the Datalight C Developers Kit is for you. ■

Concurrent Multitasking

A Review of DoubleDOS

by Jerry Houston

Once in a while a product comes along that is so good, something inside me fairly screams to tell everyone about it. Most recently I've felt that way after using a utility program called DoubleDOS from a company called SoftLogic Solutions. This product offers true concurrent multitasking at a very affordable price. It's well-thought-out, convenient to use, and provides some additional features that are, themselves, worth the price of admission.

Everyone has different reasons for needing multitasking, but I'm convinced that everyone has reasons. In my case, the need was clear from the moment I set up a bulletin board system for use by my students. Called COLLEGE CORNER, it provides a very useful E-mail system in addition to making public domain software conveniently available to the one class of individual who's almost always too broke to buy the comercial stuff — college students.

There were a lot of hours when I wasn't using the computer, and I reasoned that someone else could be benefitting from it during that time. After all, I never shut if off, so why not put it to good use? It wasn't long before I found myself needing the computer for a few minutes to enter grades into a spreadsheet, but I couldn't because someone else was in the middle of downloading the public domain equivalent of the Encyclopaedia Britanica. On the other hand, there were times when I was using my word processor to prepare a test or a handout, and I'd feel really guilty when I'd notice the CD light on my modem turn on — someone wasn't able to log on because I was hogging the system. Clearly something needed to be done. I needed a way to provide concurrent multitasking, and it had to be a way I could afford.

There are a number of public-domain approaches to multitasking, and I tried two of them (PS and DOSAMATC, both available from the BBS here at TCJ). Though they are both very useful programs, and we certainly can't argue about the price, they don't provide the answer. Both programs provided multitasking, that is, partitioning of the computer's workspace so that more than one program, or task, could be resident at a time, but they didn't provide concurrency, allowing more than one program to appear to be running at the same time.

Both are handy, and I recommend them highly to anyone who doesn't actually need concurrency. The idea is that you can switch back and forth quickly from one program to the other, without having to reload either of them.

An obvious application is when using a word processor such as Jim Button's excellent PC-WRITE, which includes one program called ED.EXE for editing text and another one called PR.EXE for printing files. Rather than changing back and forth between them (especially in these days of slow floppy drives and cheap memory), why not keep both in memory and ZAP back and forth between them by pressing a simple key combination? Either DOSAMATC or PS works fine for that.

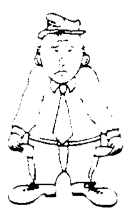
Another good use is to run a term program like ProComm in one partition, and your word processor in another. While trying unsuccessfully to log onto a BBS, you can ZAP over to the word processor and record a few precious thoughts, then while waiting for the next inspiration to strike (or while running to the kitchen for a snack) you can ZAP back to the term program and let it continue dialing. Notice that when you leave the term program, it will not keep trying your number, but that it will pick up right where it left off when you return. That's multitasking without concurrency.

Of course, my needs here were different, and so might be yours. I need to have COLLEGE CORNER available whenever a student calls, and still be able to use the word processor or spreadsheet when I have some work to do.

I mentioned that I tried PS (Program Shift) and DOSAMATC without luck, and I tried TopView. I disliked the formality of TopView — all the set-up that was required and the inflexibility. You can't just use TopView to run whatever program you need at the time, each of them has to be individually installed first, with special configuration files created for it. So much for those 250 useful little utilities on the hard disk, and if you want to do something with DOS, it had better be one of the services provided through TopView — not all of them are. All in all, TopView might be useful for someone with a few specific programs they need to run concurrently at times, but NOT for unrestricted use of the computer up to the limits of memory.

In desperation, willing to put up with whatever was necessary for the sake of concurrency, I went through the tedious process of installing a few programs to run under TopView, and got to the point that I could share the computer between RBBS-PC and PC-Write — not exactly my idea of heaven, yet I could keep the RBBS going during those hours when I was writing. All seemed functional, if not ideal, until version 14.1B of RBBS arrived, and later version 14.1C, all set up to use a fantastic new sliding-window version of the Kermit file transfer protocol. Trouble was, using Kermit (which I very much wanted to do) required another 128K of memory IN ADDITION to the 192 used by RB-BS-PC. I soon found out that TopView didn't leave me enough of my 640K to run the RBBS system and any other significant programs. I don't know exactly how much overhead it requires — frankly I no longer care enough to check — but I can assure you it's significant.

I use a monochrome adapter and a high-resolution amber monitor, because I spend literally hours at a time writing articles for computer magazines. I don't generally do anything with my computer that requires pixel graphics, so well-formed easy-on-the-eyes monochrome letters are best for me. I bought Microsoft Windows, thinking that nearly anything would be an improvement over TopView, only to find out that I can't use it with my monochrome adapter, so that package remains unopened.



POOR MAN'S NETWORK

Now you can implement networking on your own CP/M computers and share resources whenever you want. Each user can access files and printers on the other computer directly, without expensive hardware or switches, and without a communications program. Share floppies. RAM-disk, hard disks, and printers between two users. Works with most standard CP/M programs, like Wordstar, PIP, dBase II, etc.

Poor Man's Network works best with bidirectional parallel ports, or standard serial ports at 9600 baud or greater. Present version supports two computers only, and requires CP/M 2.2 on each computer. Uses only 6K of memory. Does not require a hard disk; runs on 8080, 8085, Z-80, HD-64180.

Poor Man's Network comes ready to run on BigBoard I and II, Xerox 820, NorthStar Horizon, NorthStar Advantage, Televideo TPC I, Kaypro (not 2000), Apple II (Super Serial required). S-100 with Compupro Interfacer 4. Can be installed on other computers by altering one of the assembler overlay files provided. Each disk contains drivers for all the specified computers.

Best of all is the price: only **$69!**

Specify disk format: 8" SSSD, NorthStar, Kaypro, or Apple CP/M. Phone orders accepted; sorry, no CODs or Purchase Orders. Payment may be by certified cheque, money order, Visa, or Mastercard. Personal cheques from outside Canada require up to 5 weeks to clear. Canadian orders please pay in Canadian dollars; others in US dollars. Price includes diskette, manual and postage. Ontario residents please add provincial sales tax.

Note: If you use another operating system, such as CP/M 3 or PC-DOS, send us your name and address, but no money, and we will notify you when it is available for your system.

ANDERSON TECHNO-PRODUCTS INC.
947 Richmond Road, Dept T
Ottawa, Ontario K2B 6R1, Canada

Telephone **613-722-0690** for more information or to order.

CP/M is a trademark of Digital Research. Z-80 is a trademark of Zilog. Wordstar is a trademark of MicroPro. dBase II is a trademark of Ashton-Tate. Poor Mans Network is a trademark of Doug Anderson Techno-Products Inc.

I knew of a couple other SYSOPs who have used DoubleDOS, so I ran out and actually BOUGHT a copy of it yesterday ($32.95 at my local software store, and only $49.95 at full retail). I wouldn't want anyone to think this review is so favorable because those folks gave me a copy. It's been the answer to my prayers, and COLLEGE CORNER no longer needs to come off the air when I have to use my computer.

DoubleDOS simply splits the computer's memory into two partitions, and then lets you do ANYTHING YOU WANT TO with those partitions! No special restrictions, no special configuration or PIF files to create for each program that you intend to run under it.

Now, I have some programs that do things with DOS and the BIOS that would probably be illegal in Mississippi. I know that sooner or later I'll try to run something that DoubleDOS objects to, but so far I haven't found it. The program comes with a large number of patches that are easy to apply to programs that are "badly behaved", that is, that write directly to the screen refresh buffer rather than using DOS services for display. It comes with special print drivers for LOTUS, including one that allows a single copy of 1-2-3 to be used with color AND monochrome adapters, and an improved version of the ANSI.SYS file that can simply be substituted for it in a CONFIG.SYS file. None of the patches affects the way these programs work without DoubleDOS, so they can be installed without worry.

Though it can vary somewhat depending on your installaton (how many display adapters you use of what kind), my implementation of DoubleDOS seems to require about 20K of overhead, a 20K I'm glad to devote for the purpose. The program works by partitioning DOS into two areas to load an "upper program" and a 'lower program". I have it set aside 320K for RBBS-PC and Kermit in the upper area, and I am left with 210K in the bottom area, to do with as I want. So far, as I said, I haven't been able to find anything I can't do with it. Not only can I run RBBS-PC and still use my word processor, I can do so and edit a file of almost 64K in memory, AND be printing from the optional 16K buffer at the same time!

That's right, the optional 16K buffer. DoubleDOS lets you set up an optional printer buffer using anywhere from 1K to 64K of your RAM for the purpose. One of my printers has only an 80-character buffer, the other a 2K buffer, and I've wanted a big printer buffer for a long time but never badly enough to justify the considerable cost. Using DoubleDOS, I set aside 16K that I'm willing to spare for the purpose. Printing to that memory is done at about 5K per second, and when the last 16K has been sent to the printer, the computer is free again for the next job.

Now, I know someone is thinking "doesn't he know about PRINT.COM from DOS", but believe me, it's not the same. DoubleDOS offers a choice of four different methods of implementing the printer driver, using interrupts, the clock, direct printing (no buffer), or the ROM printer interface (also no buffer). Depending on how your particular computer is set up, you may not be able to use the interrupt method (the best), but the clock method is nearly as good, and it works on my not-100%-compatible clone just fine. If no buffering is needed, the direct method can be used, and if all else fails, the ROM interface will almost certainly work with any computer, though with a speed penalty.

Actually, PRINT.COM is vastly improved by DoubleDOS, and in fact two separate copies of it can be used, one in the upper memory partition and one in the lower. That means that four programs can actually be active at one time, two applications and two copies of PRINT.COM. Now listen closely, 'cause you're not gonna believe this — you can not only RUN four programs that print at one time, but they could conceivably operate FOUR DIFFERENT PRINTERS at once!

DoubleDOS makes it easy to take full advantage of whatever resources your computer has available. Besides being able to operate multiple printer ports, a computer running DoubleDOS is able to operate two monitors at once, according to SoftLogic, one displaying the output from each program. They have finally provided the reason I need to go ahead and get a color graphics adapter and another monitor. I expect it to be a real thrill to see two programs producing screen output simultaneously, and to be able to ZAP the keyboard back and forth between them by pressing a hot key!

To sum things up, DoubleDOS has made concurrent multitasking not only possible, they've made it affordable and simple. It's easy to install, easy to modify with its own DDCONFIG.SYS file, and easy to use. Pressing ALT-DEL at any time produces a full status page that identifies what programs are running in the two partitions and provides control over them. Pressing ALT-ESC changes easily from one partition to the other. Rather than requiring special configurations for each program to be used, DoubleDOS lets you run whatever you want to, from an ordinary looking DOS prompt in each partition. (There is an additional message above the DOS prompt that shows which partition you're in, and how much memory is available there.)

The DoubleDOS manual is professionally written and understandable. It is organized, well indexed, and offers clear examples whenever they might help. SoftLogic even provides an after-hours BBS support system in addition to their regular technical support (numbers below).

The config file (DDCONFIG.SYS) that DoubleDOS looks for each time it is executed already contains most of the available alternatives, arranged so that each can be rejected by preceeding it with an asterisk, or selected by removing that asterisk. The same file can contain the equivalent of AUTOEXEC.BAT statements for both of the memory partitions, so that each of them can be individually customized.

I haven't had time to try DoubleDOS with every program on my hard disk, but so far there have been no disappointments at all. I regularly use the CED program (public domain command line editor) with a large file of aliases, so I can enter something like

```
tcj
```

and end up in the directory with my articles for The Computer Journal, rather than having to type a long path name like

```
cd \ wordproc \ articles \ compjour
```

and so far DoubleDOS hasn't had any problems with CED. I use QuickKey, a little PD program to speed up keyboard repeats, and PCWNDW22, a PD version of SideKick, complete with alarm clock, notepad, and so on. SoftLogic recommends installing DoubleDOS before any of the resident programs. That's kinda refreshing, compared to all the other programs that each insist on being the LAST one loaded!

You don't have to be the SysOp of a bulletin board system to appreciate multitasking. If you've ever twiddled your thumbs waiting for the computer to finish downloading a large file, or if you've ever sat around waiting for your printer to finish, you'll appreciate concurrent multitasking. If you develop programs, you can compile and link in the background while editing the next source module.

If you've ever experimented with real-world computing you will appreciate how easy it is to provide computer control of your house in the background, while keeping the computer available for other needed tasks. In a commercial or office building, one of the computers can monitor a security system in the background, while an operator also uses it to post receivables and print invoices. If you have two parallel printers, you can buy a copy of DoubleDOS AND another parallel port for less than the cost of a printer "A-B" switch.

Fact is, our CPUs are fast enough these days, and we have enough cheap memory available, that it just doesn't make sense for a computer to be limited arbitrarily to one task at a time. Not with DoubleDOS so easy to use, and available at a price anyone can afford.

The DoubleDOS multitasking, concurrent DOS utility is produced by Softlogic Solutions, 530 Chestnut St., Manchester, NH 03101. Tech support is available by calling (603) 644-5555 (9 a.m. to 5 p.m. Eastern time) or on their BBS at (603) 644-5556 (5 p.m. to 8 a.m. Eastern time). The published list price is $49.95, but it is discounted by many sources. ■

The public domain files PS and DOSAMATIC mentioned in this article can be downloaded from the 16-BIT file section of the TCJ bulletin board (406) 752-1038, and are available on MS DOS 5.25" with other public domain utilities for $10 postpaid in the U.S.

News

AMPRO Distributor

On October 1, 1986, BEAR Electronics became an authorized dealer for AMPRO Computers. They now stock all AMPRO Little Boards and other products for immediate shipment to customers.

BEAR Electronics, located in Moraga, California, 5 miles east of Oakland, was founded by Bob Christensen, Ed Wrinkle, and Rich Tumin as a distributorship for electronic products which they had previously been marketing individually as manufacturers reps. Both Bob and Ed have represented AMPRO successfully for more than a year in Northern California and Nevada. The staff at BEAR all have extensive experience in the electronic and computer fields.

To celebrate their new dealership, Bear is selling all AMPRO products at 10% off thru December 31, 1986. To purchase AMPRO products, or for more information, call BEAR Electronics at (415) 376-0125.

Multi-C Function Library

Cytek has announced its Multi-C multitasking library for C programmers. Multi-C is an easy to use library of functions which, when linked with the user's code, provides a complete multi-tasking environment within the program. Designated functions become tasks which can schedule other tasks, communicate with each other using queues and flags, and utilize almost all standard C functions. Interrupt handlers are supported.

Cytek originally developed Multi-C to meet their customers' needs for multitasking within their products. By incorporating Multi-C into the program design they are able to implement complex communications functions or data logging, for example, without compromising the user interface or using difficult to maintain polling techniques.

Multi-C is available immediately for Lattice, Microsoft, and Computer Innovations 8086 C compilers as well as Paragon's Z-80 C Cross Compiler for the PCDOS/MSDOS systems at $149. Versions for other compilers as well as "Accessory Kits" for Communications and Windowing facilities will be announced shortly.

Contact Alan Finger at Cytek, 805 Turnpike Street, Unit 202, North Andover, MA 01845 phone (617) 687-8086 to order, or for more details.

68000 Operating System

Hawthorne Technology is supplying their K-OS ONE operating system for the 68000 to establish a standardized operating system that will allow an affordable software base to be developed for it. Several hardware products are on the market using the 68000, but the operating systems that are available are either very expensive for the average user, very awkward with their limitations, or both.

The operating system is designed to be

(Continued on page 55)

Back Issues Available:

TCJ ORDER FORM

Subscription:	U.S.	Can & Mex	Surface Foreign	Total
6 issues per year ☐New ☐Renewal	1 yr. $14.00 2 yr. $24.00	$22.00	$24.00	
Back Issues #'s 1 thru 21	$3.25 ea.	$3.25 ea.	$4.75 ea.	
#'s 22 thru 25	$3.75 ea.	$3.75 ea.	$5.25 ea.	
#'s				
User Disk Description: Size: Format: System:	$10	$10	$10	
Total Enclosed				

Check or Money Order on U.S. Bank in U.S. Funds.
Make payable to THE COMPUTER JOURNAL.

☐ Check enclosed ☐ VISA ☐ MasterCard Card # ____________________

Expiration date ________ Signature ____________________

Name ____________________

Address ____________________

City ____________________ State ________ ZIP ________

The Computer Journal

190 Sullivan Crossroad, Columbia Falls, MT 59912 Phone (406) 257-9119

*** **Orders can also be entered by modem on the TCJ BBS (406) 752-1038 by leaving a private message for SYSOP with the required information.**

Books of Interest

Turbo Pascal A Problem-Solving Approach — by Elliot B. Koffman
Published by Addison-Wesley Publishing Company, Inc., 1986, 532 pages plus Appendixes, 7 × 9".

An excellent college text for those new to Pascal and Turbo Pascal. The author proceeds through the beginning material to more complex techniques with many programming examples and exercises at the end of each chapter to allow readers to test their learning. Especially helpful are the case studies and the discussion of common programming errors in every chapter. The contents of the book are as follows:

Problem Solving and Programming; Processing a High-level Language Program; Formatting Program Output; Introduction to Data Types; Representing and Refining Algorithms; Using Procedures for Subproblems; Decision Steps in Algorithms; Tracing a Program or Algorithm; Problem Solving Strategies; Repetition in Programs—Counting Loops; Generalizing a Solution; Repeating a Program Body; Debugging and Testing Programs; Syntax Diagrams; The if Statement Revisited; The while Statement; Procedure Parameters; Adding Data Flow Information to Structure Charts; Nested Procedures and Scope of Identifiers; Debugging a Program System; Constant Declarations; Numeric Data Types—REAL and INTEGER; Functions in Arithmetic Expressions; BOOLEAN Variables, Expressions, and Operators; Character Variables and Functions; Introduction to Programmer-defined Data Types; The case Statement; Set Values in Decisions; The General for Statement; The repeat Statement; Nested Loops; User-defined Functions; Declaring and Referencing Arrays; Arrays with Integer Subscripts; Manipulating Entire Arrays; Reading Part of an Array; Strings and Arrays of Characters; General Arrays; Arrays of Arrays; String Manipulation; Declaring a Record; Manipulating Individual Fields of a Record; Manipulating an Entire Record; Arrays of Records; Searching an Array; Sorting an Array; General Data Structures; Variant Records; Set Data Type and Set Operators; TEXT Files; User-defined File Types; Random Access Files; Data Base Inquiry; The Nature of Recursion; Recursive Procedures; Recursive Functions; Binary Search of an Array; Searching by Hashing; Additional Sorting Algorithms; The Quicksort Algorithm; The NEW Statement and Pointer Variables; Understanding Dynamic Allocation; Linked Lists; Manipulating Linked Lists Using Pointer Variables; Maintaining a Linked List; Stacks and Queues; Multiple-linked Lists and Trees; Maintaining a Binary Search Tree.

Advanced Turbo Pascal Programming and Techniques — by Herbert Schildt
Published by Osborne McGraw-Hill, 1986, 269 pages, 7 × 9"

A well written book which gives the reader a good look at intermediate level topics and provides insight into many common programming problems. A thorough discussion of interfacing to assembly language routines and the operating system includes a comprehensive list of the PC-DOS System Routines accessed through interrupts using the MsDos procedure. Many programming examples are included in the book and the programs are also available on disk. The contents are as follows:

Pascal as a Structured Language, Turbo Pascal Enhancements; Built-in Procedures; Differences Between Turbo Pascal and Standard Pascal; Classes of Sorting Algorithms; Judging Sorting Algorithms; Sorting by Selection; Insertion Sort; Improved Sorts; Shell Sort; Quick Sort; Sorting Strings; Sorting Records; Sorting Disk Files; Search Methods; Queues; The Circular Queue; Stacks; Linked Lists; Singly Linked Lists; Doubly Linked Lists; A Mailing List That Uses a Doubly Linked List; Binary Trees; New; Dispose; Mark and Release; Sparse-Array Processing; The Linked-List Approach to Sparse Arrays; The Binary Tree Approach to Sparse Arrays; The Pointer-Array Approach to Sparse Arrays; Reusable Buffers; The Unknown Memory Dilemma; Fragmentation; Dynamic Allocation and Artificial Intelligence; Assembly Language Interfacing; Internal Data Formats and Calling Conventions of Turbo Pascal; Creating an External Assembly Code Routine; In-Line Assembly Code; When to Code in Assembler; Operating-System Interfacing; BIOS and DOS Functions; Using the Scan Codes From the PC Keyboard; Final Thoughts on Operating-System Interfacing; Samples, Populations, Distributions, and Variables, Basic Statistics; Simple Plotting on the Screen; Projections and the Regression Equation; Making a Complete Statistics Program; Types of Ciphers; Substitution Ciphers; Transposition Ciphers; Bit-Manipulation Ciphers; Data Compression; Code Breaking; Random Number Generators; Determining the Quality of a Generator; Using Multiple Generators; Simulations; Simulating a Check-out Line; Random-Walk Portfolio Management; Dissecting an Expression; Expression Parsing; A Simple Expression Parser; Adding Variables to the Parser; Syntax-Checking in a Recursive Descent Parser; Converting C to Turbo Pascal; A Comparison of C and Turbo Pascal; Converting BASIC to Turbo Pascal; Creating Turbo Pascal Subprograms From BASIC Programs; Getting Rid of Global Variables; Avoiding Code Duplication; Use of Procedures and Functions; The case Statement Versus if/then/else Ladders; Porting Programs; Using const; Operating System Dependencies; Turbo Pascal's Extensions; Debugging; Pointer Problems; Redefining Built-in Procedures and Functions; Unexpected Syntax Errors; if/then/if/else Errors; Forgetting the var Parameter in Procedures and Functions.

Editor

Continued from page 2

and interfacing to bus systems. There'll also be information on implementing dedicated controllers using single board computers, and this will entail both software and hardware techniques.

The emphasis is on solutions to problems, and in order to serve you we need to know the problems you face so that we can concentrate on the solutions you need. As Dave Thompson stated in the last issue of Micro Cornucopia, harass an editor. We receive very little feedback from our readers, and your input is vital.

Market Developments

The latest financial reports show that IBM's market share is declining (both in micros and mainframes), and that they are losing control of the PC market because of the success of the clones. IBM usually replaces each product after two to three years so that people have to upgrade because the old products are no longer supported, but there are now so many clones in use that third party support and continued life is assured regardless of what IBM does. There will be tremendous advances in computers, but there'll alway be some Apple II, CP/M, and IBM-PC systems in continued use.

Many of the changes will be as a result of the advances in chip technology which will provide the performace improvements that will force us to entirely change our concepts of computer use and programming. As an indication of what is coming, Faraday has announced a two chip set which provides all the functions of an IBM-PC, and they say that they will be be able to provide everthing on one chip! Putting the entire CPU on a chip was the milestone which established the micro revolution — it's difficult to envision what will happen when we have the entire computer on a chip.

Current useage is moving from floppy disks to hard drives, and it will soon move to very large RAMs for storage. There are already systems with 4 to 8 Mega bytes of available RAM, and RAM will soon replace the hard disks leaving high capacity removable media drives which will be used for backup. Maxell (a subsidiary of Hitachi) has announced a 5¼ inch high density cartridge based media with 100 Mega byte capacity. It uses perpendicular recording techniques along with 10,000 tracks per inch to accomplish this. Perpendicular recording has been discussed for some time, but

BD Software, Inc., maker of the original CP/M-80 C Language Development System, knows

Time is precious

So the compilation, linkage and execution speeds of BDS C are the fastest available, even (especially!) on floppy-based systems. Just ask any user! With 15,000 + packages sold since 1979, there are *lots* of users . . .

New! Ed Ream's RED text editor has been integrated into the package, making BDS C a truly complete, self-contained C development system.

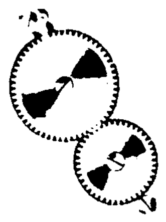
Powerful original features: CDB symbolic source-level debugger, fully customizable library and run-time package (for convenient ROM-ing of code), XMODEM-compatible telecommunications package, and other sample applications.

National C User's Group provides direct access to the wealth of public-domain software written in BDS C, including text editors and formatters, BBS's, assemblers, C compliers, games and much more.

Complete package price: $150.
All soft-sectored disk formats, plus Apple CP/M, available off-the-shelf. Shipping: free, by UPS, within USA for *prepaid* orders. Canada: $5. Other: $25. VISA, MC, COD, rush orders accepted.

BDSoftware, Inc.

BD Software, Inc.
P O Box 2368
Cambridge MA 02238
617 • 576 • 3828



development has been held back by the lack of a suitable recording head. Now, Maxell has developed a perpendicular recording head using an amorphous metal-ferrite magnetic conduit to convey vertical recording signals to a drive integrator s read-write electronics. A drive would have to include a lot of track-servo control electronics to achive 10,000 tracks per inch, but even existing stepper servo technology would present significant data density and data transfer rate improvements. With the new systems you'll operate entirely from RAM, using the magnetic media only for uploading new programs and downloading data for backup.

New Subscription Rates

We've resisted any rate increases as long as possible, but steadily increasing production and fulfillment costs have forced us to revise our rates. Effective January 1, 1987, our rates will be one year $16 or two years $28 in the U.S. You have the opportunity to extend your subscription before January 1, at the old rates.

Computer Club Activity

The current issue of 'Push & Pop' (The Sacramento' Microcomputer User's Group newsletter) reports on the declining participation from the members. Is the computer club era over? Perhaps now that it is so easy to buy a working computer with the needed software the need for clubs has diminished. What's happening with the club scene in your area? Are people just interested in learning to boot their system and call up the files? Your thoughts and comments will be appreciated.

Computer Mania

I (sometimes) envy those people with one 'computer-in-a-box' systems. Their life must be so simple and easy.

I don't know what your area looks like, but I'm sitting here with four different computers, two different terminals, various drives, and one printer. There are also other various systems and peripherals scattered around the office, and I'm always juggling cables, disk formats, and terminal configurations, when switching from system to system. My problems will get worse instead of better because I want to keep on trying new systems in order to learn about them, but I sure wish that there was more standardization as far as terminal interfaces, disk formats, and cable connectors. It's gotten so bad that I have three installed versions of WordStar on my 8 inch CP/M drive so that I can work with whichever terminal is available. That's ridiculous! It's not doing any fancy graphics and the terminal manufacturers should have been able to agree on a core set of general purpose cursor commands for standard text operations — they could have added their fancy special purpose commands on top of these, but the standard commands should work on any terminal. Lacking that, they should have provided a means for the software to query the terminal and download the commands for whichever terminal I happen to be using at that time. They've designed with the idea that everything is locked together in one system and stays that way for ever, but that's not the way I operate.

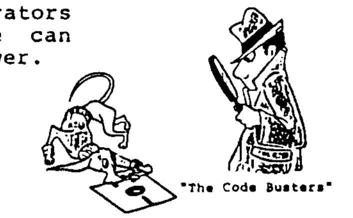
Yesterday the Morrow S-100 system which runs the photo typesetter went up in smoke, the power supply is dead and I'm not sure if it took out any of the boards. I grabbed a single board computer and lashed it up so that I could continue setting type because we are already behind schedule. There is no real problem moving the software over because everything on this system is either my own programs, software with realistic agreements, or public domain software — that's one of the real pleasures about avoiding commercial software with restrictive licenses. I'll have to revise some of the communications procedures to talk to different I/O chips and ports, which means that I'll also have multiple copies of these programs. I'll have to figure out how to have the software determine which system it is running on and patch itself to suit. The easiest way would be to change all the BIOSs to include an identifier (something like ZCPR3's environmental descriptor), but I'd rather not mess with the BIOS. Any one have any tips on accomplishing this in CP/M? Maybe the best thing would be to take the time to get ZCPR3 on all the systems and then just use the environmental descriptor.

I really like the power and flexibility of having the source code for programs so that I can fine tune them. I'll continue to buy programming tools like compilers, libraries, etc., but I'm trying to avoid using anything without the source code for my working programs. I hate to think about giving up WordStar because I really like it, but I'm looking at several public domain programs which will allow me to patch in things like the terminal identifier subroutine (if I can figure out how to do it). ■

Ever Wondered What Makes *TURBOPASCAL* Tick?

Source Code Generators by C. C. Software can give you the answer.

"The darndest thing I ever did see..." "... if you're at all interested in what's going on in your system, it's worth it." Jerry Pournelle, BYTE, Sept '83



The SCG-TP program produces <u>fully</u> commented and labeled source code for your TURBO-Pascal system. To modify, just edit and assemble. Version 3.00A (Z80) is $45. SCG's available for CP/M 2.2 ($45) and CP/M+ ($75). Please include $1.50 postage (in Calif add 6.5%).

C. C. Software, 1907 Alvarado Ave. Dept M
Walnut Creek, CA 94596 (415)939-8153

CP/M is a registered trademark of Digital Research, Inc.
TURBO Pascal is a trademark of Borland International

Computer Corner

(Continued from page 56)

(818) 889-1646) brought several of their add-ons, a 32032 and a 68020 both with up to 4 megabytes of memory. These units are described as small mini-computers and will match the speeds of VAX system. What must be kept in mind is these units use a XT clone for their system base. The clone base then allows users to use the 68K software and standard PCDOS software without changing machines. You get functionality, speed, and standardization all at reasonable prices.

A last comment on the SOG was a general consensus about computing. It was felt that all the new devices were for the most part equally good. When conducting speed test one type might be faster than another, but the next test might reverse these finding. In short they are all good devices which can supply the average person with more computing power than they might ever need. That last statement was also a discussion area in itself as several vendors pointed out how they thought 2 megabytes of memory was more than enough, now 4 is what most want, and they say several user are asking for 8 megabytes. It seems no matter what you feel is enough there is always someone who needs more.

With the SOG laid aside, a problem I have been seeing is with hard disks. Two problems seem consistent, disk failures and no backups. The first problem is the heat and wear that these disks are getting. When using floppies the motors and heads are usually unloaded when not being used. A hard disk on the other hand is running and loaded at all times. These units are tightly sealed to keep out dust that would shorten their life but still bearings will only last so long. Because of this it is most important to buy good hard disks, but also to practice backing up your data regularly.

It seems lately that more systems are getting into the hands of beginners who have little knowledge of losing data. Most manuals and sales people spend far too little time explaining the importance of backing data up. Usually losing data once cures the problem. Extended life might be possible if larger ram-disk systems were used and the drives turned off. This would require buffering a meg or two of data before accessing the drive. As most users know it can take as much as one minute for the hard disk to reach speed, so considerable buffering would be needed to make it work. My reasoning is that most users turn the machine on at 8 a.m. and then off at 5 p.m., while only really using the unit for several hours during the day.

I plan on using floppies and ram drives. My ram drives are faster than hard disks, and the floppy drives last forever with few problems.

As for my problems they are only starting, I am going back to college. I have decided to pursue my interests in using computers for education by starting a master's program in that area. It should only stop my major articles, not this column as I will still be working 8 hours a day on electronic systems. In the past I have written many articles and have always tried to cover the topic for novice users to the pros. Awhile ago the Journal recieved a letter indicating that the reader had missed some points in one of my articles. Should that happen again, please drop us a card letting us know what you didn't understand. Someone will then write an article covering that subject fully, so that you can get the most out of this magazine. With my busy schedule it will not always be possible for me to be sure I have not missed some point, but bring it to my attention and I will be glad to spend the time to cover it. ■

News

(Continued from page 49)

easy to implement, easy to use, and low priced. It incorporates all of the features you would expect from an operating system like CP/M or MS-DOS, bringing these features to the 68000. A simple design was used to allow implementation of the operating system on most 68000 hardware. Having the source code makes it possible for you to modify and maintain the operating system. You can read and write ASCII files on MS-DOS format disks, and the number of devices or open files is not restricted.

K-OS ONE is availble for $50 from Hawthorne Technology, 8836 S.E. Stark, Portland, OR 97216 phone (503) 254-2005, and includes the source code for the operating system and command processor, the HTPL Compiler, a 68000 Assembler, a line editor, and a manual. The operating system and command processor are written in Hawthorne Technology Programming Language (HTPL) which is a high level language hybrid. It has a structure like Modula-2 and RPN Expressions like Forth.

Advertiser's Index

AMPRO Computers	5, 14
Anderson Techno Prod	48
BD Software	53
Bear Electronics	18
Bersearch	11
BV Engineering	21
C.C. Software	54
CEC	43
Computer Journal	50, 51
Computer Trader	46
Echelon, Inc	19, 37
Hawthorne Technology	2
Integrand	35
Jerryco	45
Kenmore Computer Tech	18
LDL Electronics	45
Miken Optical	36
Miller Microcomputer Services	32
Rockland Publishing	22

THE COMPUTER CORNER

A Column by Bill Kibler

Computer mainia has not stopped, especially at meetings of computer hackers. Those who haven't been to a SOG meeting are probably wondering what I am talking about. Cryptic it may seem, but crazy it was. That is the SOG V that I went to last month. Held by the Micro Cornicopia at Bend Oregon, this event is getting better each year.

Last year when we were there, it was the last few days of a two week vacation. This year I took a special trip by myself, and had more fun. The seminars were still too short but then how do you stuff fifty hours of talk into twenty-five. Dave Thompson who is the editor of MicroC started things off with a talk about the magazine and was followed by technical sessions covering all aspects of computing.

For me there were a few good things which I will cover, the other stuff you will have to get from MicroC magazine or plan on going next year. One important thing for me was meeting Art our editor here at Computer Journal. Art decided to take the SOG in and set up a table for pushing the magazine. I think he got some good response but he also got to meet several of his writers first hand. The socializing with Art and others was very beneficial for me. I had been losing drive about computers but the SOG gave it back to me.

One of the items that got me going was an 68K operating system. Several of us were in need of a cheap way to bring up a 68K system. Readers of this column know that I am trying to work on a 68K Forth operating system to be ROM based. CPM68K is available, although I don't think DRI has done anything with it lately. The cost for DRI CPM68K is over $300 which is much too expensive for most of us who want to just play around. Hawthorne Technology (8836 SE Stark, Portland, OR 97216, (503)254-2005) has a single board 68K multi user system they are OEMing for which they also wrote their own operating system. It is written in a high level language they developed, based on Forth and Modula. The language and operating system have some merits, but what is important is they decided to supply a simpler version of it for single users. I have received a document on the language and operating system already. There was a special demonstration meeting at the end of August, but I was unable to attend.

This operating system will sell for about $50 and should help establish low-cost 68K systems. I live in Sacramento California, and a fellow from our area who was at SOG will be producing a single board experimenters 68K system, including the operating system for somewhere around $500. These two items should give people like myself who want to play with 68K's a cheap way out. These items will also have open-design which will allow users to customize or change it to suit their needs. There are several 68K systems around now, but few if any will allow you to change anything on them. As an experimenter/hacker, not having access to the insides of a system makes me uninterested in todays current batch of 68K units.

Some time ago I talked about using a modified BIOS that would allow a second system to take control of the first unit and transfer files as if they were your own drives. The idea was to avoid having to use modem 7 where some one must enter the files to be transfered on each terminal. Using BYE has problems as you must use XMODEM to actually transfer files. What I want to do is to use all my normal programs with the other systems disks or files and have them appear to my system as if they were two extra drives.

I bought a program at the SOG that pretty much matched or exceeded the program I have been looking for. Called POOR MAN'S NETWORK, (by ANDERSON TECHNO-PRODUCTS 947 Richmond Road, Ottawa, Ontario, Canada K2B 6R1, (613)722-0690) this software package for $65 is pretty nice. It doesn't work the same as if I had designed it, but then nothing ever does. What the manual says it will do, it does. My main dislike is the absence of a seperate module which you could load or better yet ROM to make a defined slave unit. As the package comes, you SIGNON a terminal setting drive and printer options. The other unit also assigns drives and printers, and after that your drives can be used by the other machine without you being aware of it. You can leave messages and write to each others drive but you can not change the disks in the drives. This non-changing disks I don't like but then it only takes a few seconds to SIGNOFF and then re-SIGNON with new disks. I have installed it on several systems and the patching for different systems is very easy. They do provide several already asembled overlays for Kaypros, Bigboards, Apples, and a couple other units. I will be sending him my overlays soon, so there should be quite a few systems it will work on. It works only on CPM 2.2 systems and is 6K long, hiding at the TOP of your TPA. This 6k file size is bad but then you most likely will not use it all the time. My other complaint is the lack of a CP/M 68k version so we can use it with the development systems. I guess I am still going to have to get my Forth system going.

The recent issue of Forth Dimensions, the magazine of the Forth Interest Group, has an article reviewing Xmodem protocol and how to do it in Forth. This with some other recent Forth articles, may make it possible to do just such a program in Forth. A network system that is written in a portable language, which can be moved between different processors would sure help me get my different systems communicating with each other.

A considerable amount of time both in official sessions and group get-togethers, was spent on discussing new devices and systems. The effects of the clones was always a good topic. I also found lots of other people that agreed that IBM made considerable mistakes with the PC (both in design and marketing). It has become clear to almost all users that having technically better hardware is no longer important. A users software base has become the major buying point. To combat this need for a fixed design and yet still have better systems, several hardware add-on boards are now available to make clones small MINIs. Definicon Systems Inc. (31324 Via Colinas, Suite 108, Westlake Village, CA 91362 phone

(Continued on page 55)