HANDLING CP/M UTILITIES, BIOS & FILE OPERATIONS

See Pages 10-32

Also in this Issue

Renaming Files on North Star Disks by Mark M. Zeiger.................. Page 38
Running S.D. Systems' ExpandoRAM at 4 MHz by W. Howard Adams........ Page 44
Pascal Speed Comparisons by Fred Greeb................................ Page 48
Relocatable Code by Richard H. Mossip.................................. Page 54

and more

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

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

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

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

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

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

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

*CP/M is a trademark of Digital Research.
NEW! TPM* for TRS-80 Model II
NEW! System/6 Package

Computer Design Labs

Z80* Disk Software

S-100 MICROSYSTEMS

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

— Carl Galletti and Roger Amond, owners

Software with Manual/Manual Alone

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

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

BASIC I

A powerful and fast interactive BASIC interpreter with EDIT, RENUMBER, TRACE, PRINT USING, assembly language subroutine CALL, LOADDDO for "chaining", GOPY to move text, EXCHANGE, KILL, LINE INPUT, error intercept, sequential file handling in both ASCII and binary formats, and much, much more. It runs in a little over 12 K. Its excellent choice for game programs where the precision was limited to 7 digits in order to make it one of the fastest around. $49.95/$15.

BASIC II

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

BUSINESS BASIC

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

ZEDIT

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

ZTEL

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

TOP

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

MACRO I

A macro assembler which will generate relocatable or absolute code for the Z80 or Z80 using standard Intel mnemonics plus TDL/Z80 extensions. Functions include 14 conditionals, 16 listing controls, 54 pseudo-ops, 17 arithmetic/logical operations, local and global symbols, chaining files, and more. Requires TEL for the assembly of the assembler! This assembler is so powerful you'll think it is doing all the work for you. It actually makes assembly language programming much less of an effort and more creative. $79.95/$20.

MACRO II

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

LINKER

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

DEBUG I

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

DEBUG II

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

ZAPPLE

A Z80 executive and debug monitor. Capable of search, ASCII put and display, read and write to 1/0 ports, hex math, breakpoint, execute, move, fill, display, read and write in Intel or binary format, and more on disk. $34.95/$15.

APPLE

Bios version of Zapple. $34.95/$15.

NEW! TPM now available for TRS-80 Model III

A NEW Z80 disk operating system! This is not CP/M®! But it is so powerful! You can still run any program which runs with CP/M® but unlike CP/M® this operating system was written specifically for the Z80 and takes full advantage of all of its extra powerful instruction sets. In other words it's not heard of 8080 code! Available for TRS-80® (Model I or II), and Xitan DDDC, SD Sales "VERSA-FLOPPY", North Star (SD40), and Digital (Micro) Systems: $79.95/$25.

SYSTEM MONITOR BOARD FOR Z80 II

A complete BO board for S-100 systems. 2 serial ports, 2 parallel ports, 8080/2400 baud cassette tape interface, sockets for 2K RAM, 3-2708/2716 EPROM's or ROM, jump on reset circuitry. Base board $49.95/$20.

ROM FOR Z80 II

2K masked ROM of Zapple monitor includes source listing $34.95/$15.

PAYROLL (source code only)

The Osborne package. Requires C Basic 2.

5 disks $124.95 (manual not included)
8 disks $199.95 (manual not included)
Manual $20.00

ACCOUNTS PAYABLE/RECEIVABLE (source code only)

By Osborne. Requires C Basic 2.

5 disks $124.95 (manual not included)
8 disks $199.95 (manual not included)
Manual $20.00

GENERAL LEDGER (source code only)

By Osborne. Requires C Basic 2.

5 disks $99.95 (manual not included)
8 disks $99.95 (manual not included)
Manual $20.00

C BASIC 2

Required for Osborne software. $99.95/$20.

SYSTEM/6

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

Above purchased separately costs $339.75. Special introductory offer: Only $179.75 with coupon!

ORDERING INFORMATION

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

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

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

For information and tech queries call 609-599-2146

For phone orders only call toll free 1-800-327-9191 Ext. 676

(Except Florida)

OEBS

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

- Z80 is a trademark of Zilog
- TRS-80 is a trademark for Radio Shack
- TPM is a trademark of Computer Design Labs. It is not CP/M®
- XDF is a trademark of Digital Research
- Credit prices and specifications subject to change without notice.

DEALER INQUIRIES INVITED.
The Empire has expanded!

New Mainframe opens more areas for development

In one quantum leap Tarbell has expanded its popular Empire (the vertical disk subsystem) into a full line. This entire series now encompasses 5 variations. Each one contains different components so the S-100 system designer, hobbyist, or serious business user can arrive at the exact custom state he wants and needs.

The basic Empire still includes two Shugart or Siemens 8" disk drives; the compact cabinet with fan and power supply; a Tarbell floppy disk interface; CP/M®, Tarbell BASIC; the necessary cables, connectors and complete documentation. Naturally, it's fully assembled and Tarbell tested.

The new, top of the line Empire contains the basic model's components with the Tarbell design-approved Mainframe. Beside the 8-slot S-100 motherboard with an active terminated bus, there's a cardcage with card guides and a double-density interface.

You're the master of your Empire

You can call the shots in the Empire. Tarbell's made sure of that by offering them as complete subsystem packages . . . or, as separate units. For example, the mainframe may be ordered with 1, 2 or no drives. Whichever way you go, however, you always get the reliability of Tarbell tested components and leadership-engineering.

To get control of your own Empire, see your quality computer store for quick delivery. Or, contact us for dealer locations or further information.

CP/M is a trademark of Digital Research.

Tarbell Electronics
950 Dovlen Place - Suite B
Carson, California, 90746
(213) 538-4251 / 538-2254
**STAFF**

- **Sol Libes**  
  editor
- **Claudette Moore**  
  managing editor
- **Patti Balinski**  
  layout
- **Debbie Barbagallo**  
  typesetting
- **Jacob Epstein**  
  CP/M* editor
- **Jon Bondy**  
  Pascal editor
- **Don Libes**  
  assistant editor
- **David A. Gewirtz**  
  software editor
- **Suzanne Guppy**  
  circulation
- **Frances Miskovich**  

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

Commercial advertising is welcomed. Write to S-100 MICROSYSTEMS, 93 Washington St., Morris-town, NJ 07960, or phone Claudette Moore at 201-267-4558.

*TMK Digital Research*
The past year has seen very sizeable increases in the prices of software packages. The next few months should see sizeable increases in the prices for microcomputer software. This is all due to significant changes in the software marketplace.

I remember only five years ago buying my first copy of Basic for only $5. It was a mini-version of about 2K bytes of code on paper tape. It came in the mail in a standard number ten size envelope (I remember the postage was only 10 cents then) and included 5 photocopied pages of instructions. The author sold it directly, as a hobby. He had a good full time job and this was just a little project, on the side. The program provided only the very fundamental Basic functions. Storing a Basic program, by today's standards, was a real hassle and there were a few bugs in the program. But at $5, who could complain?

Today, things have changed radically. I have changed...from a pure hobbyist to an entrepreneur that relies on his system for income. My needs have changed. I need powerful software with a lot of bells and whistles. Further, I need software which will allow me to do my tasks quickly. And also, I need software that is bug free and does not consume my time with debugging.

Today I think little of spending $100 for a software package, and I have spent as much as $500 for a package. I think that I am typical of most microcomputer users. I am sure that most of the computer hobbyists of two to five years ago are the entrepreneurs of today. It seems like all of my old computer hobbyist buddies are making money with their systems, one way or another. Although we still play around with our systems just for the fun of it, most of our time is now spent on income producing projects.

The microcomputer market has thus been shifting from what was originally strictly a hobby to one that is now predominantly oriented to professional users. The likelihood is that the coming year will see much more of this, with business applications finally coming to dominate the microcomputer area.

The microcomputer software supplier area was started by part-timers working in their basements and attics...like my Basic supplier. There are still plenty of them around in fact, their number seems to be increasing. However, the software marketplace has matured greatly. The new breed of software supplier is catering to the professional and business users. Customers are not price sensitive. Rather, they are willing and able to pay high prices but are demanding value for their money. They want sophisticated software. They want a lot of support and handholding. They want software that is bug free and delivers on promises. This demand for higher quality software requires that the software supplier be a professionally run organization that is there to stay. This costs money and hence the increases in software prices reflect this.

An example of the changes in software prices is reflected by the price of the VisiCalc software package sold by Personal Software. This very popular software package was first introduced with a suggested selling price of $100. It was sold through dealers and hence the company may have netted only $60-70 on each sale. Probably, most of this amount was eaten up by production and marketing costs leaving little, if anything, to cover development and customer support costs. The result was that Personal Software recently increased the price to $200—a 100% increase—and the likelihood is that the price will be increased further. Most software suppliers, however, are attempting to hold prices on previously released packages and are raising prices on new releases.

There are already a few microcomputer software packages selling in the $1,000 area. These are the business-oriented packages. It is likely that by the end of next year typical software package costs will have shifted from the present $100-$500 to the $500-$1000 area, with few going to the $2000 area. After all, keep in mind that software packages for mini and large size computer systems sell in the over $10,000 area with several going for over $50,000.

One distressing fact is that although software suppliers have been raising software prices, the level of customer support and pre-testing of software has still not changed as dramatically.
At Intersystems, “dump” is an instruction. Not a way of life.

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

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

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

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

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

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

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

I'm glad that there is finally someone competent supporting the S-100 bus. The magazines have been doing at best a mediocre job and its been getting progressively worse. The S-100 bus certainly has had its problems and I'm well aware of them with my collection of over 150 S-100 boards. However, there's a lot to be said in favor of the S-100 bus and that just has not been done properly by the traditional magazines. The creativeness, innovativeness, and imagination of many of the S-100 designers has me in absolute awe. It really irks me when someone says an S-100 machine is no good because its a hobbyist machine and is therefore of inferior quality. True, there is a lot of slop in the S-100 area, but that's because there has been so much designed for it and because of the hesitancy of the traditional magazines to print legitimate criticisms of poorly designed and obsolete products. There is more depth in the S-100 area than anywhere else. Unfortunately, some of the really first rate quality products in the S-100 area (such as the Computer, Biotech products, ADS Noisemaker, Speechlab, Digital Graphic Systems CAT-100, Cambridge Development Labs graphic boards, Casheab music synthesizers, Marinchip 9900 MPU, Sublogic software, etc.) have not gotten the exposure and attention they deserve from the traditional magazines even though they've been around for several years. And as a result, many of these fine products are about to die.

One area of great hope and promise for the S-100 bus is the 16-bit microcomputer. Unfortunately, despite the IEEE S-100 standards, this could be a nightmare for the unwary consumer. For example, some 16-bit processors will work with only 16-bit memory and not the 8-bit memory that most of us S-100 users presently own. The new Ithaca InterSystems Z8000, the re-designed Alpha Micro 100T, and the Alpha-Micro 8086 are this way. Also it turns out that it is optional in the IEEE S-100 standards whether a manufacturer who designs IEEE S-100 memory boards wants to support a 16-bit data path for his memory. Most of the IEEE S-100 memory products on the market right now such as Godbout, Thilinker Toys, Measurement System & Control, etc. does not support true 16-bit data transfers. I should also add that though some of the new 16-bit products are electrically IEEE S-100 compatible, they may not be physically IEEE S-100 compatible. For example, the new Alpha-Micro 100T and Piceon's 32K 16-bit word memory cards are both 10 inches tall!! This means that you can't fit them into a normal IMSAI, ALTAIR, TEI, Vector Graphics, Godbout, or North Star chassis. Then there's Marinchip's 32K 16-bit word memory card which will only work with their 16-bit computer and will not work with any of the 8-bit computers. Also some of the memory-mapped devices such as the Star North miniflippy Interface, the North Star hardware multiply/divide board, and Thinkertoy's floppy disk probably won't work with all of the new 16-bit computers. Since many of these new 16-bit computers can directly address 128K bytes of memory or more, customers should begin to consider seriously forcing the manufacturers to provide memory with error-correcting code capability, as is done on mini-computers and traditional mainframes. And I haven't even mentioned anything in the area of software yet its pretty obvious that something needs to be done and said in the magazines to provide a smoother transition for S-100 consumers into the world of the powerful 16-bitters if the S-100 bus is to be successful in this area. I want it to be.

Kenneth Young
Los Angeles, CA

Dear Editor:

The following might be an item of interest to some of your readers: I have found a paperback book called "8080/8085 Software Design" to be an excellent publication. Perhaps it might even be called a reference book, as it contains a library of commonly used 8086 assembly language routines, which saves the reader considerable effort in not having to roll his own, and also, probably saves him from doing many of those in the worst possible manner.

Much of today's documentation, I find, is poorly written; and I think of it as not unlike asking a native for directions to get somewhere. Invariably the reply is something like, "Follow the road you're on all the way down. You'll see it. You can't possibly miss it." Having traveled up and down that same road each day, the native soon becomes oblivious to the fact that there may be three forks in the road along the way. To him, his way home, is the only 'straight road.' A newcomer usually can describe a route better than someone-presenting himself as an authority.

In any event, Dr. Christopher Titus seems to have an ability to point out the 'forks' in the 'road'. Mine was the Sams version consisting of two volumes at $10. each, although I believe that E&L instruments has since come out with the same text in one volume, also $10., which would seem to be a better buy.

Ernest J. Zitke
Brocton, Mass.

Dear Editor:

I am currently making a CPU selection for my S-100 system, and would like to advise you of some discrepancies I have found in the S-100 Processor Boards and Manufacturers article published on page 50 of the Jul-Aug issue of S-100 MICROSYSTEMS

1) I ordered a Z-80 processor board from Digital Research Computers in Texas. My check was returned with a note saying that they no longer offered the board that I had ordered. In fact, their literature accompanying my returned check makes no mention of any processor board now being offered.

2) The article mentions the 8085 and/or 8088 boards (Compupro) from Godbout Electronics, but fails to mention their enhanced Z-80A board. This was advertised on the back cover of the July/August issue of S-100 MICROSYSTEMS.

3) Page 202 of the July 1980 issue of KILOBAUD advertises a 68000 board for the S-100 bus with on board Z-80 emulation. It is to be marketed by Vandata, 17541 Stone Ave. N., Seattle WA 98133 PH (206)-542-8370. I have learned through a telephone conversation with an employee at Vandata that a prototype of this board is functioning and is being built by Imperial Research Group (IRG) in Kent WA. And that it may be available as early as October 1980.

Best of luck with S-100 Microsystems. I am finding it a useful and enjoyable tool while putting together my system. Perhaps when all its working I will be able to share an article, some facet of this system's operation or construction as it relates to the S-100 bus.

Tim Hamilton
Creswell, OR
Apple and the TRS 80 have been given. Mass production by a large well known company is not the only reason, as evidenced by the popularity of the Apple. Much of the innovative and useful software written for these products would be nothing without the graphics.

In my opinion, one of the main reasons for this success, a reason which should be considered carefully by the S-100 community, is our lack of a standard video graphics system. Hardware variety is both the joy and the curse of the S-100 world. According to my count, there are at least 21 different manufacturers of S-100 video boards alone, with some firms offering up to 3 different models of video boards. This does not count the wide variety of terminals. Some of the boards and terminals have no graphics capability except what can be managed with text characters. Others sport a wide variety of graphics formats, from character block graphics characters up to HiRes boards programmable down to the last pixel. This very variety virtually guarantees that writing software for S-100 graphics will rarely be very profitable.

Fortunately, due to bus, processor, exchange, and text screen standards, there is a very large amount of non-graphics oriented software around, much of it business and utility programs.

With a new generation of HiRes Color Video boards appearing, how can we encourage programmers to use more graphics in their programs, not just in games, but in business software also? I see three conceivable avenues: (1) unlikely; vote on a hardware standard video system which every one interested would buy (this would be unfair to many companies). (2) Get the companies to agree to at least one standard—each board would be switchable to run in this mode (this would require modifications to many existing boards and some could not be modified at all). (3) Cooperate in publishing complete tables of operating parameters for existing boards and systems, and prescriptions for translating formats where possible.

All of these ideas have serious drawbacks, but they are a start. I strongly urge you to make MICROSYSTEMS a coordinating medium for attacking this problem.

For example: can readers agree on specifications for a desirable video board like the following: 1) true memory mapped, switchable location or phantom: 32K at 4MHz, 2) 24 by 80 with optional 48-51 by 80 Black & white & Gray & Color Capability. 4) both character block and pixel mode graphics, ram chr. generator. 5) variety of graphics resolutions, but at least 512 by 512. 6) On-board keyboard port (parallel), 7) 256 character codes and inverse (each mode), 8) light pen hardware and software, 9) contiguous character blocks and decenders for text mode. 10) IEEE compatible, 11) software support especially utilities, CPM compatible. Some of the parameters which should be included in comparison tables should be: Company, Address, Tel. # Name of Board, Accessories A&T Cost.
No.16: "Take away!

Gnomes for Apple II

Apple II

All GSM programs are available
for CP/M, unless otherwise stated.

- COMPAQ 8080/280
- computer disk systems

CP/M FLOPPY DISK OPERATING SYSTEM - Digital Research's operating system configured for many popular microcomputers and disk systems.

SOFTWARE ASSOCIATES

- PHOENIX - Symbolic debugger included. Supports inter- process communication between CP/M and other systems. Available with source code.
- MACA-80 - Compiles enhanced Standard ALGOL-60 with linkable routines for use with CP/M.
- PASCAL/Z - Z80 native code PASCAL compiler. Produces ROM code that can run on any system. Available with source code.
- PASCAL/M - Compiles extended Standard ALGOL-60 with linkable routines for use with CP/M.
- BASIC UTILITY DISK - Consists of: (1) BASIC-16 compatible with CP/M, (2) TRS-80 and Z80 compatible ROM monitor, and (3) FORTRAN-80 compatible ROM monitor.
- IMPACT - Keyed Index Sequential Search, offers search and update program of indexed files. Can also be used as a substitute for COBOL-2. When used as a substitute, VSORT maximizes the use of buffer space by deferring process execution to disk for errors, reserving the imperfections for disk I/O. Requires 56K CP/M Spec. 1.080 CP/M. C025/65.
- STRINGS - Full disk BASIC features plus unique string handling routines. Supports full file transfer facilities between CP/M systems and includes a complete command language interpreter. Supports Unix-like commands. Provides full file transfer facilities between CP/M systems and includes a complete command language interpreter. Supports Unix-like commands. Provides full file transfer facilities between CP/M systems and includes a complete command language interpreter.
- COMPAQ 8080/280 - Computer disk systems

DIGITAL RESEARCH

- MP/M - Installed for single or multiple disk systems, supporting all major CP/M feature. Available with source code.
- MACA-80 - Compiles enhanced Standard ALGOL-60 with linkable routines for use with CP/M.
- BASIC-16 - Full disk BASIC features plus unique string handling routines. Supports full file transfer facilities between CP/M systems and includes a complete command language interpreter. Supports Unix-like commands. Provides full file transfer facilities between CP/M systems and includes a complete command language interpreter.
- IMPACT - Keyed Index Sequential Search, offers search and update program of indexed files. Can also be used as a substitute for COBOL-2. When used as a substitute, VSORT maximizes the use of buffer space by deferring process execution to disk for errors, reserving the imperfections for disk I/O. Requires 56K CP/M Spec. 1.080 CP/M. C025/65.
- STRINGS - Full disk BASIC features plus unique string handling routines. Supports full file transfer facilities between CP/M systems and includes a complete command language interpreter. Supports Unix-like commands. Provides full file transfer facilities between CP/M systems and includes a complete command language interpreter.
- COMPAQ 8080/280 - Computer disk systems

MICROSOFT

- BASIC-80 - Disk Extended BASIC. ANSI standard compatible with all major CP/M systems. Available with source code.
- Compiler-LANGUAGE PROCESSOR - Supports both 8080 and 8085 microprocessors. Available with source code.
- COBOL - Standard COBOL compiler fully validated by U.S. Navy, UNISYS, and other organizations. Supports level 2 including dynamic loading of COBOL modules. Supports full ISAM and 3800 file processing. Includes complete and untested CTRL screen formatting and supports standard terminal.
- KISS - Keyed Index Sequential Search. Offers search and update program of indexed files. Can also be used as a substitute for COBOL-2. When used as a substitute, VSORT maximizes the use of buffer space by deferring process execution to disk for errors, reserving the imperfections for disk I/O. Requires 56K CP/M Spec. 1.080 CP/M. C025/65.
- STRINGS - Full disk BASIC features plus unique string handling routines. Supports full file transfer facilities between CP/M systems and includes a complete command language interpreter. Supports Unix-like commands. Provides full file transfer facilities between CP/M systems and includes a complete command language interpreter.
- COMPAQ 8080/280 - Computer disk systems

Eidos Systems

- KISS - Keyed Index Sequential Search. Offers search and update program of indexed files. Can also be used as a substitute for COBOL-2. When used as a substitute, VSORT maximizes the use of buffer space by deferring process execution to disk for errors, reserving the imperfections for disk I/O. Requires 56K CP/M Spec. 1.080 CP/M. C025/65.
- STRINGS - Full disk BASIC features plus unique string handling routines. Supports full file transfer facilities between CP/M systems and includes a complete command language interpreter. Supports Unix-like commands. Provides full file transfer facilities between CP/M systems and includes a complete command language interpreter.
- COMPAQ 8080/280 - Computer disk systems

YXBSIC Interactive Process Control System

- CP/M - Character string handoff. Routines to find, fill, move, move, append, concatenate, and compare character strings. This package completely eliminates character string handoff associated with character string handoff in PASCAL and PASCAL/M.
- IMPACT - Keyed Index Sequential Search, offers search and update program of indexed files. Can also be used as a substitute for COBOL-2. When used as a substitute, VSORT maximizes the use of buffer space by deferring process execution to disk for errors, reserving the imperfections for disk I/O. Requires 56K CP/M Spec. 1.080 CP/M. C025/65.
WORD-STAR—Menu driven visual word processing system for use with standard terminals. Text formatting performed on screen. Facilities for simple keypunching, editing, viewing, and underlining. User can print one document with all the characteristics previously selected and underlined. Edits facilitate global search and replace. Word-to-word or block moves, etc. Requires CRT terminal with addressable cursor position.

WORD-STAR-MAIL-MERGE—As above, but with option for printing mailing of personalized documents. Character sets and fonts from DATASTAR or NAD.

WORD-STAR Customizes Notes—For sophisticated users who do not have one of the many standard terminal or printer configurations. Powerful editing and formatting features included. Requires WORD-STAR.

SUPER-TEXT—Text editor for intelligent word processing. Has support of CP/M's ED commands including simple editing, formatting, and backwards in file in video mode. Provides full screen editor for users with small addressable cursor terminal.

TEXTWRITER—TEXT file to format justify and paginate words and other documents. Special features include insertion of text during execution, viewing of text from other disk files or cassettes, permitting users to delete text from selected text on diskettes. Single disk driven system. Requires IBM or compatible.

**NEW LINES** for application Software

**PEACHTREE SOFTWARE**

General accounting software for small business. Many types of business may be used, with automatic posting to the general ledger. Must be used in conjunction with lifeboat accounting software. Requires CP/M.

**GRAHAM-DORIAN SOFTWARE SYSTEMS**

Dynamic systems accounting software written in CP/M and CP/M-80 and sold in source code. Each system is designed to be used with one stand-alone real-time or on-line system or integrated with the General Ledger for automatic posting to ledger accounts. Requires CP/M-80.

**CONDMENTS**


**FLIPPER SAVER**—Protection for center holes for S and 5 tracks. Single diskete. Kit contains center post, pressure plate and 7 mill mylar reinforcing rings for 25 diskettes.

$14.95

R. Sides only—$7.95

B. Sides only—$8.95.

**FASCAL USER MANUAL AND REPORT**—By the authors of LIFEBOAT. Used with the Lifeboat software. Recommended for use with FASCAL. Pasaki/MD and Pasaki/MT users $15.

**NEW NEWSLETTER FROM LIFEBOAT**

**LATEST VERSION NUMBER LIST**

**UPDATE** on CP/M lines Group

The Great ZOSO Speaks Out from Behind the Scenes

**NEW PRODUCTS**

**PRICES**


**STRUCTURED MICROPROCESSOR PROGRAMMING**—By the authors of SMAL/80. Covers structured programming on the 8080/8085 instruction set and the SMAL/80 language.

**ACCOUNTS PAYABLE & RECEIVABLE—CBASIC**—By Osborne/McGraw-Hill.

**GENERAL LEDGER—CBASIC**—By Osborne/McGraw-Hill.

**PAYROLL WITH COST ACCOUNTING—CBASIC**—By Osborne/McGraw-Hill.

**LIFEBASE DISK COPYING SERVICE**—Transfers data from one floppy or hard disk unit to another at a moderate cost.

**CP/M-MP/M are trademarks of Digital Real-Time Computing, Inc.**

**SOFTWARE SUPERMARKET**

The Software Supermarket is a trademark of Microsoft Associates.
Utilities and Bios—Part I

Introduction to CP/M—Part IV

by Jake Epstein

Standard mnemonics; 3. dump areas of memory showing both hexadecimal equivalents and ASCII equivalents of each location; 4. execute a program with breakpoints; 5. list and alter the contents of processor registers and flags; 6. perform hexadecimal addition and subtraction; 7. perform single or multiple step execution of programs showing machine register contents and flag status after each instruction; 8. load files from disk to any location in memory.

SUBMIT.COM - This program is used for batch processing. Using ED.COM, the user prepares a file that contains a list of CCP commands to be executed by the operating system. This file has the secondary name SUB. When SUBMIT is invoked, all the commands listed in the SUB file will be executed just as if they had been entered at the console. SUBMIT will only work when drive 0 (system drive) is logged in. In CP/M 1.4 and earlier versions, once the user leaves the CCP mode, as when executing a program, the SUBMIT function relinquishes control of the system until a warm boot (cntr-C) is executed by the user. In version 2.0 and later, commands that are interpreted by a program can be programmed using the XSUB.COM utility in co-ordination with SUBMIT. Thus, long and complex processes involving many different devices and programs can be executed with or without user interaction.

MOVCPM.COM - System alteration for various sized memories is accomplished using this utility. MOVCPM will be discussed in depth under system alteration in a future article in this series.

PIP.COM - The peripheral Interchange Program is used to transfer information from one location to another. The locations involved can be disk files or I/O devices such as the console or the printer. PIP also contains several software switches that allow for verification or alteration of the data flow.

SYSGEN.COM - This program is used to read or write to the system tracks of a diskette. This allows the operator to alter components of the operating system. This utility will also be discussed in depth under system alteration in a future article.

In addition to the COM files listed above, Digital Research provides two ASCII files that are used in system alteration in a future article. These files, DISKDEF.Lib and DEBLOCK.ASM, will be discussed in

CP/M Utilities

Digital Research supplies several utilities with CP/M. These various programs are used for file management, text processing, program development, and information transfer. Rather than discuss each program in depth, I will merely list them and give a brief description of their function. Certain utilities will be covered in depth in following articles when system alteration and program development are discussed. If anyone is interested in submitting articles devoted exclusively to the description and use of specific utilities and/or other CP/M programs, feel free to contact S-100 Microsystems.

ED.COM - This is a line oriented text editor used for preparation of ASCII files. With CP/M, it is usually used in co-ordination with ASM.COM and SUBMIT.COM, but can be also used to prepare text for other programs such as BASIC or a TEXT PROCESSOR.

ASM.COM - This program is used to assemble Intel 8080 standard format ASCII files. Input files have the secondary name ASM while output files have secondary names of HEX and/or PRN. HEX files, which are in Intel hex format, contain a representation of the machine code of the assembled source program. PRN files are identical to the ASM source file but with machine code equivalents listed in hexadecimal code adjacent to each line of assembly code.

LOAD.COM - Hex format files are converted to executable binary files using this program. The output file produced is a COM file which can then be executed using the CCP LOAD and EXECUTE command.

STAT.COM - File characteristics such as length or write protection are checked with STAT. Logical devices can also be checked using this program. The STAT utility is much more comprehensive in CP/M 2.0 and later versions than in earlier versions.

DDT.COM - This program provides an extensive array of commands that are used to inspect and modify the contents of system memory. It can be used to create, analyze, and debug programs with the following functions: 1. alter memory using hexadecimal numbers or Intel standard assembler mnemonics; 2. disassemble machine code to Intel

Jake Epstein, Rainbow Associate, 25 Sagamore Cove, Branford, CT 06405
the next article under BIOS II. The term TRANSIENT COMMAND is often used for utilities such as PIP and STAT because they are often used in a fashion similar to CCP commands to monitor and alter system status. Rather than being built into the CCP, however, they are loaded and then executed in the TPA, Transient Program Area.

**BIOS - Part I**

As discussed in the first article of this series, BIOS (Basic Input/Output System) is the module of CP/M in which software interfaces to computer peripherals are located. The BIOS that is provided by Digital Research is for the Intel MDS INTELLEC computer system. If the user does not have this computer system, then the BIOS has to be modified for his/her hardware configuration. Once implemented however, software that is CP/M compatible may be executed without need for additional modification of the operating system. Manufacturers of disk controller boards for 8080/8085-based microcomputers supply BIOS's, but the user may still have to modify non-disk routines such as serial or parallel port drivers. The main focus of this section and BIOS - Part II will be on the structure and operation of BIOS with a few examples of modification. This section will be devoted to general organization of BIOS and non-disk I/O routines. Part II will deal with disk I/O routines with emphasis on version 2.0 enhancements. It is advisable that the reader obtain a listing of a working BIOS to use as a reference to this section. Issue number 2 of S-100 MICROSYSTEMS contains an excellent BIOS for version 1.4 written by Martin Nichols (see references).

**Structure**

BIOS is the last module in the CP/M memory map. All versions of CP/M have the following routines:

1: COLD BOOT This routine is used to initialize various areas of the operating system after the CP/M is loaded from the system diskette.

2: WARM BOOT Used after a system reset (cntr-C) to load in CCP and BDOS without the need to load and initialize the entire system.

3: CONSOLE STATUS Used to determine whether or not data is ready to be input from the console device.

4: CONSOLE INPUT Used to input data from the console.

5: CONSOLE OUTPUT Used to transmit data to the console device.

6: LIST OUTPUT Used to transmit data to the list (hard-copy) device.

7: PUNCH Used to transmit data to the paper tape punch.

8: READER Used to input data from the paper tape reader.

9: HOME Causes the logged in disk drive to seek track 00.

10: SELECT DISK Used to select the disk drive where disk I/O will take place. Used to log in a disk.

11: SET TRACK Initializes the disk controller to a specified track.

12: SET SECTOR Initializes the disk controller to a specified sector.

13: SET DMA ADDRESS Sets the beginning of a buffer area in system memory where data transfer will occur during disk I/O.

14: READ Data will be transferred from the disk to the DMA buffer as determined by the above routines. The amount of data is that which is contained in one physical unit on the disk - one sector of 128 bytes on a single density floppy diskette.

14: WRITE Same as read only data is transferred from the DMA buffer to the disk.

The following two routines have been added to CP/M 2.0 and later versions:

15: LIST STATUS Used to check status of the list device.

16: SECTOR Used to convert from a logical location TRANSLATE to a physical location of a sector. Used with translation tables found elsewhere in BIOS. See Article II in this series for a preliminary discussion of sector skew.

At the beginning of BIOS, a vector jump table is located to direct programs to the various routines. The various jumps have to be placed in the order given above, but the routines themselves may be anywhere. There are three different ways in which user programs may interface to I/O devices. 1. The actual software drivers may be part of the program. In this case the program is not useable in systems different from the development system without modification. 2. The program may use system calls to BOOS to accomplish I/O. In this case, BDOS and BIOS transfer data as determined by one of 36 numbers placed in general microprocessor register "C". 3. A final possibility is in direct calls to the appropriate routine in BIOS. This is useful when a desired I/O function is not implemented in BDOS. An example of this is console input. In CP/M 1.4 and earlier versions, any input via a console input system call will cause the data to be transmitted back to the screen. In order to eliminate this character echo, one merely needs to call the fourth jump vector, console input, of the BIOS jump table. One disadvantage to this procedure is that the calls from the user program must be done in a manner that is compatible with different sized CP/M systems. See
Intro to CP/M cont’d...

listing 1 for an example of how this can be accomplished. A final note; additional routines may be added to BIOS, but jump vectors must be placed at the end of the jump table so that its continuity is not upset.

Besides the above routines, various implementations may have areas for temporary storage or data buffering. In CP/M 2.0 and later versions, areas of BIOS are reserved for disk information. Each disk unit in the system has a disk parameter block and a sector translation table stored in BIOS. This gives the operating system the ability to handle different types of mass storage units. The disk parameter block contains information such as disk size, sector size, sectors per track, etc. The sector translation table is used to determine sector skew. These areas are used by BDOS to calculate physical sector locations from logical file information. Also, BIOS contains a buffer area for directory operations. These topics will be discussed at length in BIOS – PART II.

At this point I will go into more depth on the CP/M memory map. This information is important in understanding some of the parameters influencing the location and size of the BIOS. It will also serve to introduce material that will be explained in the article dealing with system alteration. The following table shows the size and relative locations in a minimal-size CP/M memory map.

<table>
<thead>
<tr>
<th>MODULE</th>
<th>SECTORS</th>
<th>STARTING ADDRESS</th>
<th>SIZE IN BYTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPA</td>
<td>100H</td>
<td>2900H</td>
<td>10240</td>
</tr>
<tr>
<td>CCP</td>
<td>3100H</td>
<td>3328</td>
<td></td>
</tr>
<tr>
<td>BDOS</td>
<td>3EOOH</td>
<td>512</td>
<td></td>
</tr>
<tr>
<td>BIOS</td>
<td>3E00H</td>
<td>4000H</td>
<td>16384</td>
</tr>
</tbody>
</table>

The minimal memory size configuration available in CP/M is 16K in 1.4 and earlier versions and 20K in CP/M 2.0 and later versions. For larger size-configurations, the CCP and BDOS have to be relocated using MOVCPM, while BIOS and the COLD START LOADER have to modified and reassembled to the new system memory size. Although the CCP, BDOS, and BIOS remain in the same relative locations, the TPA is either expanded or contracted to fill out added or deleted space in various sized systems. In the tables, the TPA begins at 100H and is either 2800H or 3300H long depending on version.

Adding 100H to either of these values gives the starting point of the CCP: 2900H in 1.4 and 3400H in 2.0. To calculate where the CCP would start in other sized systems, simply use the following formula:

\[
\text{CCP Start} (\text{CBASE}) = \text{TPA size} + (\text{BIAS} \times 400H)
\]

where: BIAS = Memory size - Minimal configuration (in K bytes)

100H = 1024 or 1K bytes

BIAS is actually the number of 1K segments that the memory is greater than the minimal configuration. The following equations should clarify this material.

For a 24K 1.4 System:

BIAS = 24K - 16K = 8K

CCP Start = 2900H + (8 * 400H) = 2900H + 3200H = 6100H

For a 62K 2.0 System:

BIAS = 62K - 20K = 42K

CCP Start = 3400H + (42 * 400H) = 3400H + 16800H = 20200H

To calculate the location of BIOS simply add the combined size of BDOS + CCP to the CCP starting location. The following tables give the CCP and BIOS starting points for various systems.

<table>
<thead>
<tr>
<th>MEMORY SIZE</th>
<th>CCP START</th>
<th>BIOS ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>16K</td>
<td>2900H</td>
<td>3EO0H</td>
</tr>
<tr>
<td>17K</td>
<td>3000H</td>
<td>4200H</td>
</tr>
<tr>
<td>18K</td>
<td>3100H</td>
<td>4600H</td>
</tr>
<tr>
<td>19K</td>
<td>3200H</td>
<td>5000H</td>
</tr>
<tr>
<td>20K</td>
<td>3300H</td>
<td>5400H</td>
</tr>
<tr>
<td>21K</td>
<td>3400H</td>
<td>5800H</td>
</tr>
<tr>
<td>22K</td>
<td>3500H</td>
<td>6200H</td>
</tr>
<tr>
<td>23K</td>
<td>3600H</td>
<td>6600H</td>
</tr>
<tr>
<td>24K</td>
<td>3700H</td>
<td>7000H</td>
</tr>
<tr>
<td>28K</td>
<td>3900H</td>
<td>7400H</td>
</tr>
<tr>
<td>32K</td>
<td>4000H</td>
<td>7800H</td>
</tr>
<tr>
<td>40K</td>
<td>4200H</td>
<td>8200H</td>
</tr>
<tr>
<td>48K</td>
<td>4400H</td>
<td>8600H</td>
</tr>
<tr>
<td>64K</td>
<td>4600H</td>
<td>9000H</td>
</tr>
</tbody>
</table>

A problem can arise when dealing with different implementations of BIOS. Notice that in a 1.4 system there are only 512 bytes of memory, xE00H-xFFFH, available for BIOS. Since most BIOS modules will be larger in size, especially when long drivers such as video routines are added, certain functions will have to be placed in ROM and then called from BIOS. Since the system diskette has a total of 9 sectors available -- the total number on tracks 0 and 1 minus the total needed by the COLD START Loader, CCP, and BDOS -- BIOS can be a total of 9 * 128 or 1152 bytes long. To build a working system, BIOS could start at location xA00H. To do this, the system would be built one K smaller than available memory. For example, for a system with 24K of available memory, a 23K CP/M system would be built using the MOVCPM utility, and BIOS and the COLD START LOADER would be reassembled for the proper CCP, BDOS, and BIOS starting points. To complete the modification, the...
number of sectors read by the COLD START LOADER and the WARM BOOT routine in BIOS would have to be adjusted. Often, distributors of CP/M systems may alter MOVCPM so that this 1K offset is handled automatically. Problems arise when a BIOS that has been built for the standard MOVCPM is added to one of these systems. The easiest solution to the problem is to simply build system a 1K smaller, and then proceed with the modification of CP/M in the normal manner.

In CP/M 2.0, the limitation is not in system memory size but in disk space. There are only 7 sectors (380H bytes) available on a single-density 8-inch disk for BIOS, and although this is enough storage for a minimal BIOS, any additions will quickly overflow this space. One way to get around the problem is to place all buffer and scratch RAM areas at the end of BIOS. Thus only permanent code need be loaded. If certain RAM locations need to be initialized, then routines will have to be placed in the COLD BOOT module of BIOS to accomplish this. This solution could also make the permanent code larger than 7 sectors. Another option is the placing of certain routines in ROM. The easiest and most common method would be to place the ROM in the last area of memory, usually F000H or F800H. Since most routines will need stack space and scratch areas, it is wise to have RAM located in this area also. Some of the newer ROM boards have space for RAM, but if this option isn’t available, then space will have to be reserved in an area outside the bounds of CP/M in system memory. A problem with this configuration is that any time system memory is added and/or CP/M is enlarged, the ROM may have to be erased and reburned if it is an EPROM; or a brand new ROM purchased if it is not. There are ways of computing these problems. The most common and easiest option isn’t available, then space will have to be reserved in an area outside the bounds of CP/M in system memory. A problem with this configuration is that any time system memory is added and/or CP/M is enlarged, the ROM may have to be erased and reburned if it is an EPROM; or a brand new ROM purchased if it is not. There are ways of computing these problems. The most common and easiest way is the one using a ROM/RAM board. Note: the Z80 and the new 16 bit microprocessors (Z8000, 8086, 68000) do provide relative and indexed addressing schemes that would help circumvent this problem.

Another solution is in configuring the system so that there are more than 2 system tracks. This can be done easily in CP/M 2.0 by using the DISKDEF.LIB file and the CP/M MACRO Assembler. I will discuss this in BIOS-PART II. A problem with this is that software supplied from other sources may not be compatible with the modified system. To solve this problem, having a logical device that can read standard diskettes could be implemented in BIOS. As an example, in a two-drive system, devices A: and B: could be interfaced to drives 0 and 1 with their configuration being the one most commonly used in the system. Logical device C:, on the other hand, could be set to access drive 1 with its configuration being different than A: and B:. Using PIP, software could then be transferred between A: and C:.

A final method would be to have a minimal BIOS stored on the system tracks with an expanded BIOS stored as a file. Upon COLD START, this file could be overlaid in memory either manually or automatically. CP/M 1.4 and 2.0 can be modified so that a file is loaded and executed upon COLD BOOT. I will discuss this option in the article on system calls.

BIOS - Modules

The following discussion describes the parameters of non-disk I/O modules in depth.

CONSOLE STATUS: Upon return from this routine, the value 00 in register 'A' indicates that no character has been input at the console device; 0FFH indicates a character has been input.

CONSOLE INPUT: Upon return, register 'A' will contain an input character. The most significant bit (parity bit) should be reset to zero. This is accomplished most easily with the operation ANI 07H. Routines may be added to convert characters to different values. For example: delete (7FH) backspace (08H).

CONSOLE OUTPUT: Upon entry, register 'C' contains the character to be printed. The 'A' register will be changed except where Z80 instructions are used. Routines may be placed to control output.

READER: Same parameters as CONSOLE INPUT. May be modified to function with other devices such as modems for telephone communication.

PUNCH: Same parameters as CONSOLE OUTPUT.

LIST: Same parameters as CONSOLE OUTPUT. Routines can be added to control printer paging to or route (SPOOL) data to a disk file.

LIST STATUS: Same as CONSOLE STATUS. Used with SPOOL or DESPOOL programs.

Programmed I/O

Martin Nichols' BIOS serves as an excellent example of how to implement the above routines. In all cases, there are four possible ways of programming these routines. The most common and easiest approach is polled I/O. In this method, I/O is accomplished only when the routines are called. In order for each routine to operate properly, status has to be checked for each device, with program loops being used to force the computer to wait until the device is ready for I/O. In other words, the routine polls the I/O device's status register to determine when it is ready to send or receive data. In most commercially prepared BIOS's, options that can be selected at assembly time are given for various boards. If a board is not covered, then then the user must program into the appropriate routine the location and the logic of the status register and the location of the DATA register. A BIOS routine that allows no I/O usually indicates an error in port location. A quick stream of characters with only one input or output usually implies improper status logic or bit location. The following routine shows how to program an I/O port:

```
CONM: IN 00 JICMPUT STATUS FROM PORT 0
ANI 02 FSCHEC BIT BY ANDING "A" WITH 0000 0001B
JZ CONMOT IIF "A" REGISTER WAS 0000 0000B THENLOOP
JF NOT ON ("J" MEANS JUMP IF ZERO)
MOV A+01 INPUT CHARACTER IN "A" TO PORT 1
OUT 01 WRITE CHARACTER IN "A" TO PORT 1
RET RETURN TO THE CALLING PROGRAM
```

In this routine, the logic is positive because the status bit must go from 0 to 1 to indicate that the device is ready. Negative logic is that case where the bit changes in a negative direction (1 to 0). In a negative logic system, the third instruction would be JNZ (JUMP if not zero).
Intro to CP/M cont’d...

A second method is memory mapped-I/O. This technique is almost identical to polled I/O except that memory access instructions are used instead of inputs and outputs. Although programming using this method can be quite efficient, a major disadvantage is that I/O ports use system memory space. An example of a memory mapped I/O scheme is the keyboard of the Radio Shack TRS-80 microcomputer.

Interrupts

Another method is interrupt-driven I/O. In this case, when a device is ready to perform I/O, it interrupts the computer from its current task so that it (the computer) can perform the operation. This eliminates the need for the computer to sit and wait while it checks status via a loop as in the above example. Also, interrupts can be used to control program execution. Suppose that you have written a program that outputs integers to the screen. The main problem with interrupts are that they are much harder to implement than polled I/O. This is true for both hardware and software. To understand what is involved, I will devote a bit of discussion to the hardware and software interfacing of the Intel 8080 microprocessor. I suggest consulting the Intel 8080 Users manual and the articles on the proposed IEEE S-100 standard that are listed under the reference section at the end of this article for diagrams, timing charts, and descriptions.

Before an interrupt can occur, the 8080 must be "software-initialized" (instruction is part of a program) using the ENABLE INTERRUPT (EI - FH) instruction. Upon initialization, a flag inside the 8080 is set to indicate that interrupts are enabled. To interrupt the computer, the I/O interface logic pulls pin 14 high on the 8080. This pin is connected to line 73 of the S-100 bus, but it is interfaced so that the I/O device has to pull it low to cause an interrupt. After this occurs, the computer will finish its current instruction and then service the interrupt. The interrupting hardware provides the next machine instruction instead of program memory as pointed to by the ‘PC’ (program counter) which normally is the case. This is accomplished via an 8080 status signal that indicates an interrupt read instead of a memory read. This signal is supplied by S-100 line 96, as opposed to line 47, which indicates a memory read.

Although any instruction may be supplied, the usual procedure is to use one of the 8 RST (restart) instructions. These instructions have two effects. First, they all cause the value of the ‘PC’ to be saved on the processor stack with the usual update of register ‘SP’ (the stack pointer), as is the case with a CALL instruction. Then, depending on the instruction, the program will jump to one of 8 locations: 0, 08H, 10H, 18H, 20H, 28H, 30H, 38H. With CP/M, three of these locations cannot be used. Location 00, which is restarted by instruction RST 00 - C7H, is reserved for the warm boot jump vector, as discussed in Article 3 of this series. Location 30H, RST 6 - F7H, is the first location of an 8-byte block reserved for future use by Digital Research. Finally, location 38H, RST 7 - FFH, is used by utility programs such as DDT to control and/or monitor program execution. By placing a RST 7 instruction in place of another instruction and saving the replaced instruction and its location, a program can be run using the computer itself to execute the program up to the instruction. At that time the computer breaks out of the program and control returns to the utility, which may or may not return the initially changed instructions to the program. In DDT, two RST 7 instructions may be placed in the program at one time and are called “breakpoints”.

Although the RST instruction may be supplied by the I/O port interface, a separate interface board will be used to supply the instruction. The I/O port will request an interrupt via one of 8 VECTOR INTERRUPT pins on the S-100 bus, pins 4-11. When activated, the vector interrupt interface will generate an interrupt through S-100 line 73 and then place the RST instruction on the data bus at the appropriate time. The CPU will indicate when it is ready for this transfer via line 96; this line is used in place of line 47, which indicates a memory read. Each vector interrupt pin corresponds to a specific RST instruction with pin 4 referring to RST 0, pin 5 - RST 1 and so on. The vector interrupt interface may also contain a priority encoder. This device allows the user to set up different priorities for vector interrupts. While an interrupt is being serviced, other requests may occur in systems with more than one interrupt-driven device. When the time comes, the device with the highest priority will be serviced next. In a system with more than one device, this facility is necessary to prevent timing errors. In my next article, I will include a diagram of a vector interrupt interface that I built to use interrupts with my serial I/O board.

Since the restart instruction saves the program counter, which is set to the next instruction of the interrupted program, a simple return at the end of the interrupt service routine will allow the program to resume where it left off. Of course, all CPU registers altered by the service routine have to be returned to their state prior to the interrupt. This is best done with PUSH and POP instructions. After the interrupt, the internal flag that enabled the CPU to accept an interrupt, as mentioned above is reset to zero. Thus, the interrupt service routine must reset the flag to 1 using the EI (enable interrupt) instruction. Where this instruction is placed is important. It if is placed at the beginning of the routine, then it is possible that nested interrupts could occur. In other words, an interrupt service routine could be interrupted, which also could be interrupted, and so on. If the EI instruction is placed at the very end of the routine, then a new interrupt will not be serviced until the prior routine is completed. Interrupts must be disabled during CPU controlled data transfers that must be continued to completion. In most CP/M systems, this is during the SECTOR READ or SECTOR WRITE routines when using polled
I/O boards such as the Tarbell single-density controller. If the disk controller has a data buffer (an area to hold data) then interrupts will not cause problems.

**Direct Memory Access**

The first three I/O methods are commonly termed programmed I/O techniques as distinguished from the fourth, DMA (Direct Memory Access). When S-100 line 74 is pulled low, 8080 pin 13 is pulled high. This causes the CPU to enter a HOLD state. During this time, the CPU relinquishes control of the computer address bus (group of lines used to transfer address information) and data bus. The I/O device then can directly transfer information to and from memory at much greater speed than is possible when the CPU is involved during standard programmed I/O. There are two general ways in which DMA can be implemented.

In the first, the calling routine loads certain registers of the device with data such as sector, track, or DMA address, etc., and then issues a data transfer command. In the second method, an area of memory is loaded with this information, and the DMA controller reads this area during a DMA sequence. I will give more information on this in BIOS-PART II. While in a HOLD state, the 8080 CPU will not honor interrupts; thus, interrupt-driven I/O routines will not conflict with DMA routines. An example of a non-disk device that may use DMA is a video board that uses system memory for its refresh memory as is the case with the Processor Technology VDM-1 video board.

**Sample Routines**

I will now give a general outline on how to implement interrupts in CP/M. The examples that I will use will be based on the MITS 2S10 board set up at port location 10H with interrupt vector 1 (RST 1 will be used). Port 10H is the status port and 11H is the DATA port. Listing 2 shows a simple output routine using interrupts, as opposed to polled I/O, as demonstrated in the above example. This could drive a printer or a terminal. The main difficulty with output versus input is that the I/O device interrupts the computer as soon as the current operation is completed, and thus could place the system in an infinite loop between the main program and the interrupt service routine. To avoid this problem, the I/O port's output interrupt is disengaged until data is ready to be printed. Since all disk I/O routines must have DI and EI commands, the DI command would not work to disable the interrupts. A way of getting around this would be to use an interrupt status byte in memory, which would be checked at the end of each disk I/O to determine whether interrupts should be enabled using the EI command.

In practice, this output routine is no more efficient than polled I/O. Studying the routine will show that there is a built-in wait period when the computer is ready to output information but the I/O device is not ready. Memory buffers can be written into the I/O routine to greatly improve its efficiency. Listing 3 shows an input routine with a buffer. In this example, the interrupts will be disabled once the buffer is full. Thus, any data input after this state occurs will be lost until the buffer is emptied. Although there are many ways to implement a buffer management scheme, I chose to demonstrate a circular buffer using three counters and two pointers. POINT1 is the location where each byte is stored in the buffer as it is input from the I/O device. POINT2 is the location where data is read from the buffer to the main program. POS1 and POS2 are used to keep track of each pointer's position in the buffer. When the counter reaches zero (the pointer is at the end of the buffer) the pointer is set to the beginning of the buffer, and the counter is loaded with the length of the buffer in bytes. Thus the pointers move through the buffer in an endless circle.

The counter labeled COUNT represents the current amount of data waiting to be read by the computer, and is the space in number of bytes between pointers 1 and 2. When data is input from the I/O device, COUNT is incremented, and when data is read by the main program, it is decremented. COUNT is also used by the INSTAT routine to indicate to a main program when data is ready to be read. Another scheme would be to use comparison routines to indicate buffer placement and status but, in experimentation, I found these to be much longer and less efficient in terms of processing time than the method used here. Finally, listing 3 contains additional routines and was taken directly from a working BIOS that I have been using.

**Character Traps**

A technique that proves quite useful is character-trapping. By using 8060 CPI (compare immediate) instructions in drivers, certain characters can initiate special routines that will implement a user-defined function. In listing 3, I have placed two character traps that have proven very useful extensions to normal CCP and/or BDOS operation. By pressing the ESC key (Escape - ASCII 1BH) on my terminal, the interrupt service routine branches to a routine that awaits a second character from the terminal. In this example I have implemented just two functions, but an unlimited number could be added (with the amount of memory space available for BIOS being the only restriction). The ESC-P sequence enables output to both the console device and the list device. This is similar to the cntr-P function of CP/M, but in this case printing can be enabled or disabled in all situations and is not disabled automatically after WARM boots (cntr-C). A variation on this would be to enable other printers that have I/O drivers in BIOS. Thus, high-speed dot matrix printers versus slower wordprocessing printers could be switched on or off quite easily during execution of programs such as Microsoft BASIC-80 that allow only one list device. The second function, ESC-C, is virtually the same as cntr-C except that a warm boot can be performed at any time unless BIOS has been altered or interrupts disabled. This function is especially useful when a program "hangs-up" or the user wishes to terminate program execution and cntr-C does not work. Since warm boots do not change the TPA (transient program area), ESC-C can be used instead of system reset, which does destroy...
the TPA. Thus, ESC-C can be performed, and the TPA can be saved on disk. Care should be used when implementing a function such as this during disk I/O to prevent data loss. Although the ESC-P function may be used in polled I/O routines, ESC-C will not work if I/O does not occur after a program bombs.

A final technique that can be implemented via character traps is character conversion. One interfacing project that my consulting associate and I had was to interface a word processing printer to S-100 hardware and CP/M software. This printer uses escape sequences similar to ones implemented above to initialize a wide variety of features such as underlining, reverse print, and boldface (double strike). In order to initialize these functions, I incorporated a trap in the output routine to convert a printing character such as "l" (ASCII 7DH) to an ESC (ASCII 1BH). In the following example, ESC-A causes the printer to boldface print and ESC-B causes it to print normally.

**DISPLAY AT TERMINAL:**
Now is the time for all...

**OUTPUT TO PRINTER**
Now is the (ESC A)time(ESC B) for all...

**PRINT-OUT**
Now is the time for all...

When writing routines such as these, care must be used in selecting trap characters. Since traps often filter characters out of the data input stream, trapped data must not be important to application programs. For example, ESC is used by the "MICROSOFT Basic-80" line editor.

A final example of character trapping is Listing 4. This partial listing shows how to implement back space character deletion in a 1.4 BIOS. This will emulate the backspace option of CP/M 2.0.

**The IOBYTE**

When implemented in the BIOS, the Intel standard I/O byte function can be used to convert logical devices to specific physical devices. As an example, the console device could be a CRT terminal, a video board with separate keyboard, a printing terminal with keyboard input, or an acoustic coupler that would allow a remote terminal to be the console device over the telephone lines. Although it is the responsibility of the BIOS to direct I/O to a physical device, the STAT utility or an application program can be used to modify device assignments if the BIOS is programmed to handle such functions. Location 3 of system memory is reserved for a software register labeled IOBYTE which indicates which physical device is to be assigned to a logical device. CP/M recognizes four logical devices:

1. CON: Console  The device used by the CCP
2. LST: List  The printer or hardcopy device
3. RDR: Reader  Paper tape reader
4. PUN: Punch  Paper tape punch

When BDOS directs I/O to one of these logical devices, it does so by calling a location in the BIOS vector jump table described above. The table below shows the link between logical and physical device drivers.

<table>
<thead>
<tr>
<th>LOGICAL DEVICE</th>
<th>VECTOR JUMP TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: CON:</td>
<td>3: CONSOLE STATUS</td>
</tr>
<tr>
<td>2: LST:</td>
<td>4: CONSOLE INPUT</td>
</tr>
<tr>
<td>3: RDR:</td>
<td>5: CONSOLE OUTPUT</td>
</tr>
<tr>
<td>4: PUN:</td>
<td>6: LIST OUTPUT</td>
</tr>
</tbody>
</table>

The IOBYTE is divided into four 2-bit segments. Each segment refers to one of the logical devices listed above. Information about physical assignments are placed in each segment in the form of binary numbers. Thus each segment can represent four different assignments. Below is a diagram of the IOBYTE showing the positional relationships of logical status information.

```
<table>
<thead>
<tr>
<th>bit 7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LST:</td>
<td>PUN:</td>
<td>RDR:</td>
<td>CON:</td>
</tr>
</tbody>
</table>
```

The following information is the Intel standard logical to physical conversion for the four numbers stored in each segment. The user may or may not follow this standard in BIOS, but the numbers are stored in this manner when using the STAT utility to modify the IOBYTE.

1. CON: 0. TTY
        1. CRT
        2. BAT - Batch mode
        3. UC1 - User-defined console

2. RDR: 0. TTY
        1. PTP - High-speed punch
        2. UP1 - User-defined punch
        3. UR1 - User-defined reader 1
        4. UR2 - User-defined reader 2

3. PUN: 0. TTY
        1. PTP - High-speed punch
        2. UP2 - User-defined punch
        3. UR2 - User-defined punch

4. LST: 0. TTY
        1. CRT
        2. LPT - Line printer
        3. UL1 - User-defined list

In the above list, CRT corresponds to Cathode Ray Terminal. TTY refers to ASR (automatic send/receive) printing terminals. These terminals generally have a built-in paper tape reader/punch and run at very slow speed (11 characters per second). Thus the paper tape device is referred to as a slow reader or slow punch. In contrast to the TTY punch, the fast reader or punch is a separate device that can run at relatively high speeds. For example, DIGITAL EQUIPMENT'S PC11 reader/punch will read at 300
cps and punch at 50 cps. Line printers are usually devices dedicated to hard copy (in contrast to TTYs) and have speeds that range from 30 cps to beyond 180 cps. I have used 180 cps as a general limit because true line printers which print a wholeline at a time (as opposed to character printers which do one character at a time) are really not practical for small systems. Although fast, line printers are quite large, use vast amounts of power, are expensive, and require a great deal of hardware and software interfacing.

Not all of these devices need be written into a BIOS. The following approach gives the most efficient use of space and processor time when writing IOBYTE-directed routines:

1: Write a set of drivers (input, output, input status) for each physical device to be accessed by the system.
2: Avoid including drivers that are not needed.
3: Write a linkage routine that reads and translates IOBYTE information for each position needed in the vector jump table.
4: If possible place these routines in ROM (Read Only Memory) with its own vector jump table to facilitate programming.

If this plan is followed, redundancy will be eliminated and system generation (as when updating or changing software) will be simplified. Hardware will tend to remain static, whereas software goes through a constant evolutionary process. Listing 5 gives an example of how to write a set of linkage routines for the console device. As a final note, reviewing the sections on STAT.COM and PIP.COM in the CP/M documentation will give further insight into the use of IOBYTE.

The next article in this series will deal with disk functions found in a standard BIOS. Material on CP/M file organization will be given as an extension of information introduced in Article II of this series.

References:

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

---PROGRAM BEGINS NEXT PAGE---
LISTING 1

THIS PROGRAM DEMONSTRATES WRITING CALLS TO BIOS I/O ROUTINES THAT WILL WORK IN ANY SIZED CP/M SYSTEM WITHOUT MODIFICATION.

THE TECHNIQUE USED IS TO COMPUTE THE LOCATION OF THE BIOS VECTOR JUMP TABLE USING THE WARM BOOT ADDRESS FOUND AT LOCATION 01, AND THEN COMPUTE THE ACTUAL LOCATION OF JUMP VECTOR USING OFFSETS.

THE FOLLOWING TABLE GIVES THE RELATIVE POSITIONS OF THE BIOS VECTOR JUMP TABLE BASED ON THE WARM BOOT ADDRESS

<table>
<thead>
<tr>
<th>Wboot</th>
<th>Cold</th>
<th>Const</th>
<th>Conin</th>
<th>Consol</th>
<th>Conout</th>
<th>List</th>
<th>Punch</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0003</td>
<td>0009</td>
<td>0000</td>
<td>0005</td>
<td>0006</td>
<td>0007</td>
<td>0008</td>
<td>0009</td>
</tr>
<tr>
<td>0001</td>
<td>0100</td>
<td>0101</td>
<td>0102</td>
<td>0103</td>
<td>0104</td>
<td>0105</td>
<td>0106</td>
<td>0107</td>
</tr>
</tbody>
</table>

WBOOT EQU 00 WARM BOOT
COLD EQU WARM-3 ICOLD BOOT ADDRESS
CONST EQU WARM+3 ICONSOLE STATUS
CONIN EQU CONIN+3 ICONSOLE INPUT
CONOUT EQU CONOUT+3 ICONSOLE OUTPUT
LIST EQU CONOUT+3 LIST
PUNCH EQU LIST+3 IPAPER TAPE PUNCH
READER EQU PUNCH+3 IPAPER TAPE READER
HOME EQU READER+3 HOME DISK
SELECT EQU HOME+3 ISSELECT DISK DRIVE
SETMK EQU SETMK+3 ISET MARK
SETSEC EQU SETSEC+3 ISET SECTOR
SETDMA EQU SETDMA+3 ISET DMA ADDRESS
READ EQU SETDMA+3 IREAD ON SECTOR
WRITE EQU READ+3 IWRITE ONE SECTOR
WBOOT EQU 01 LOCATION OF WARM BOOT JUMP
TPA EQU 100H LOCATION OF WARM BOOT JUMP

This program inputs a character from the keyboard and then sends it to the list device. No printout will appear on the CRT screen.

start:

routine to compute location of input vector jump and then branch to it

INCHR:

0100 2A0100 LHALD WBOOT LOAD THE H,L REGISTER WITH WARM BOOT ADDRESS IN BIOS VECTOR JUMP TABLE
0110 110C00 LXI D+LIST LOAD D,E WITH OFFSET
0120 19 EAD D Icompute location by adding D,E to H,L
0121 E9 FCHEL #branch

routine to compute location of list vector jump and then branch to it

PRNCHR:

011A 2A0100 LHALD WBOOT LOAD H,L WITH WARM BOOT ADDRESS
011D 110C00 LXI D+LIST LOAD D,E WITH OFFSET
0120 19 EAD D Icompute address
0121 E9 FCHEL #branch

This is a demonstration of a very simple interrupt driven output routine. The memory location labeled INSTAT is used indicate whether or not a character has been printed during an interrupt.

0031 = INTEN EQU 3H BYTE USED TO INITIALIZE INTERRUPTS ON I/O PORT FOR OUTPUT ONLY
0031 = INTDIS EQU 1H DISABLES INTERRUPTS ON PORT
0000 = CSTAT EQU 00 ISTATUS PORT
0001 = CDATA EQU 01 IPORT

CONOT:

0010 0A E7 ANA A SET FLAGS
0010 09 E9 OUTCHR SET CHARACTER
0010 0C E8 OUT CHAR
0010 0D E8 OUT CHAR
0010 0E E7 OUT CHAR
0010 0F E7 OUT CHAR

end

output interrupt service routine

int02h byte used to initialize interrupts on i/o port for output only

note: this routine would be entered via a jump from one of the b locations between 00 and 38h.
the location is determined by the data placed on the bus by the i/o port hardware during the interrupt.
INTSRV:

0114 F5 PUSH PSW #SAVE A AND STATUS FLAGS
0115 C5 PUSH B #SAVE B+C REGISTER PAIR
0116 3A2801 OUT CDATA #PRINT IT
0117 AF XRA A #SET A TO 0
0118 D301 OUT #PRINT STATUS
0119 AF XRA A #SET A TO 0
011C 322701 STA #GET DATA BACK
0120 F1 POP B #SET DATA BACK
0122 FB EI #ENABLE INTERRUPTS
0124 FI POP #GET DATA BACK
0125 FB EL #GET INITIALIZATION BYTE
0126 C9 RET #RETURN TO MAIN PROGRAM

0127 00 INSTAT1DB 00 #interrupt status
0128 00 DOUTCHR: DB #character storage

;***************
#Listing 4 *
***************

;This listing demonstrates interrupt driven input and
;output. These routines were extracted from a working
;BIOS and are only minimally commented. The input service
;routine allows the user to emulate CP/M CNTR-P option -
;printing on list device or program termination - CP/M
;CNTR-C option.

;The serial board used is the MITS 2SIO which uses the
;Motorola 6850 asynchronous communications interface.
;This chip is initialized by outputting OBIH for input and
;output interrupts. 031H for output alone, 91H for input
;alone, and 11H for no interrupts (POLLED I/O MODE).

EDU: BUFLEN 10H #Determines size of input buffer

;interrupt service routine
;checks CSTAT to find which port, IN or OUT
;interrupted. If both at same time will service
;IN first

SRVINT: PUSH PSW #Check console input status
ANl CSTAT #GET status
JNZ 01 #check for input ready
JNZ INTRSRV #if so go for input
JMP OINTRSRV #otherwise go for output

;check console input status

;CONST: LDA COUNT #Get number of characters waiting
;to be input

;CONST: MVI A+00 #SET flags

;read a character from console.

;CONIN:

;INTSRV: PUSH H #Save H
IN CDATA #Get data
ANl 7FH #Strip parity bit
CPI 1BH #ESC - escape
JZ AIN #Get pointer
LHLD POINT1 #Get character
MOV H,A #Save character
LDA COUNT #Get characters
INR A #Update count
STA COUNT
LDA POS1 #Get buffer position
DCR A #Decrement position
JZ INTRSRV2 #Get buffer position
INX H #Update pointer

INTSRV1STA POS1 #SET SHLD POINT1 #Save it
JMP INTRSRV1

INTSRV1: POP H
PUSH PSW
EI RET

INTSRV2: MVI A,BUFFLEN
LXI H,BUFFER
JMP CONIN1

INTERRUPT SERVICE ROUTINE

;INTSRV: PUSH H #Save H
IN CDATA #Get data
ANl 7FH #Strip parity bit
CPI 1BH #ESC - escape
JZ AIN #Get pointer
LHLD POINT1 #Get character
MOV H,A #Save character
LDA COUNT #Get characters
INR A #Update count
STA COUNT
LDA POS1 #Get buffer position
DCR A #Decrement position
JZ INTRSRV2 #Get buffer position
INX H #Update pointer

INTSRV1STA POS1 #SET SHLD POINT1 #Save it
JMP INTRSRV1

INTSRV2: MVI A,BUFFLEN
LXI H,BUFFER
JMP INTRSRV1
**Routine to Check for Special Control Characters**

**ATTN:** IN CSTAT
ANI 01
JZ ATTN
IN CDATA
ANI 7FH
CPI 'C' WARM BOOT
JZ ATTN
IN CDATA
ANI 7FH
CPI 'P' PRINT TOGGLE
JNZ

**JMP INTRSV3**

**LDA PRNECO**
STA PRNECO
JMP INTRSV3

---

*This listing shows an example of implementing character deletion using backspace for CP/M 1.4 and earlier systems. The memory location labeled DELSTAT+ DELETION STATUS+ is used by the output routine to determine when to erase a character from the console screen. This procedure should only be used with CRT devices as opposed to TTY's.*

**CONIN:**

; WRITE A CHARACTER TO THE CONSOLE DEVICE.
; THIS IS EXPERIMENTAL OUTPUT INTERRUPTED ROUTINE

**CONOT:**

LDA ISTAT
ANA A
JNZ CONOT
INR A
STA ISTAT
MOV A+C
STA OUTCHR
MVI A+091H
OUT CSTAT
RET

**INTRV:**

PUSH B
LDA OUTCHR
OUT CDATA
MOV C+A
XRA A
STA ISTAT
MVI A+091H
OUT CSTAT
LDA PRNECO IOET PRINTER STATUS
ANI 01
CNZ LIST IF PRINTER IS ON, PRINT CHARACTER
PDP B
PDP PSW
EI
RET

ISTAT: DB 00
OUTCHR: DB 00
PRNECO: DB 00 1=PRINTER ON, 0=OFF

COUNT DB 00
POIN1 DB BUFFER
POS1 DB BUFLEN
POIN2 DB BUFFER
POS2 DB BUFLEN
BUFFER DS BUFLEN

---

**0010 = CSTAT EDO 10H**
**0011 = CDATA EDO CSTAT+1**
**0100 = DB10 ORS 100H HIUSED FOR DEMONSTRATION PURPOSE**
**0102 E601 ANI 01 CHECK BIT 0**
**0104 CA0001 JZ 000000 READY**
**0107 DB11 IN CDATA IOET CHARACTER**
**0109 E67F ANI 07FH IOSET PARITY BIT TO 0**
**010B FE7F DPI 7FH IIS IT DELETE/RUBOUT**
**010C CO RNZ RETURN IF NOT SO**
**010E 3EFF MVI A+OFFH IOET DELETION IN PROGRESS BYTE**
**0110 323A01 STA DELSTAT ISAVE IT**
**0111 C9 RET**

**0114 DB10 IN CSTAT IOET PORT STATUS**
**0116 E602 ANI 02 IOSET FLAG**
**0118 CA1401 JZ CONOT IOET PORT READY**
**011A 3A3A01 LDA DELSTAT IOET DELETION STATUS**
**011C A7 ANA A IOSET FLAGS**
**011F C22601 JNZ CONOT1 JUMP IF NOT ZERO**
**0122 79 MOV A+C IOET CHARACTER**
**0123 D311 OUT CDATA IOPRINT IT**
**0125 C9 RET**

**0126 AF CONOT1 XRA A IOCLEAR A**
**0127 323A01 STA DELSTAT IOCLEAR DELETION STATUS**
**012A C208 MVI C+08 IOET BACKSPACE**
**012C CD1401 CALL CONOT IOPRINT IT**
**012F C2E0 MVI C+20H IOET SPACE - WILL CLEAR DELETED CHARACTER FROM SCREEN**
**0131 CD1401 CALL CONOT IOPRINT IT**
**0134 0E08 MVI C+08 IOET SPACE**
**0136 CD1401 CALL CONOT IOPRINT IT**
**0139 C9 RET**

**013A 00 DELSTAT DB 00 IOFF=DELETION+0=NO DELETION**
THIS SKELETON LISTING IS A DEMONSTRATION OF HOW TO PROGRAM A BIOS WITH IOBYTE IMPLEMENTED. ALTHOUGH THE ACTUAL DRIVERS ARE NOT LISTED, EACH ROUTINE IS SUPPLIED SO THAT THE READER CAN GET A FEEL FOR HOW THE SYSTEM WORKS. THE VARIOUS DEVICES IN A HYPOTHETICAL SYSTEM COULD BE:

- TTY - AN ASR TYPE TTY
- CRT - 9600 BAUD TERMINAL WITH DETACHABLE KEYBOARD
- BAT - A MODEM BOARD
- UCI - INPUT FROM CRT DETACHABLE KEYBOARD
- OUTPUT - VIDEO BOARD FOR HIGH SPEED WORDPROCESSING
- LPT - 180 CPS DOT MATRIX PRINTER
- ULI - 45 CPS DASISHEEL PRINTER FOR WORDPROCESSING

THE LISTING ALSO SHOW HOW TO COMPUTE THE STARTING POINT OF BIOS GIVEN VERSION AND MEMORY SIZE.

| 0003 | IOBYTE EQU 03 | IOBYTE IS LOCATED AT LOCATION 3 |
| 0014 | MSIZE EQU 20 | USER WOULD FILL THIS IN FOR SYSTEM |
| 1600 | SYSTEM EQU 800H+0000H | LENGTH OF 2.0 CCP AND BISO |
| 0006 | BIOS EQU MSIZE-20 | SET UP FOR CP/M 2.0 (SEE ARTICLE |
| 3400 | TPA EQU 3400H | TPA SIZE IN MINIMAL SYSTEM + 100H |

VECTOR JUMP TABLE FOR A 20K SYSTEM

| 4A00 C3124A | JMP BOOT | ICOLD BOOT |
| 4A04 C3134A | JMP WBOOT | IWARM BOOT |
| 4A06 C3144A | JMP CONIN | ICONSOLE INPUT |
| 4A09 C3294A | JMP CONOT | ICONSOLE OUT |
| 4A0C C33E4A | JMP LIST | ILIST DEVICE |

THE LISTING CONTINUES ON FROM HERE

ICOLD BOOT ROUTINE WOULD GO HERE IN ACTUAL LISTING.
RET USED SO THAT ASSEMBLER WILL WORK WITHOUT ERRORS.

4A12 C9 | BOOT: | RET | IFOR DEMO |

IWARM BOOT ROUTINE WOULD GO HERE IN ACTUAL LISTING

4A13 C9 | WBOOT: | RET | IFOR DEMO |

CONST IS USED TO DETERMINE IF A CHARACTER IS READY AT THE CONSOLE DEVICE, WHICH CONSOLE DEVICE THAT IS TESTED IS DETERMINED BY I/O BYTE.

CONST:

| 4A14 JA0300 | LDA IOBYTE | GET IOBYTE |
| 4A17 E003 | ANI 03 | CHECK CONSOLE SEGMENT |
| 4A19 CA684A | JZ TTYST | CHECK TTY INPUT STATUS |
| 4A1C FE01 | CPI 01 | SET FLAGS |
| 4A1E CA684A | JZ CRTST | CHECK CRT INPUT STATUS |
| 4A21 FE02 | CPI 02 | SET FLAGS |
| 4A23 CA6E4A | JZ BATST | CHECK BATCH INPUT STATUS |
| 4A26 CA384A | JMP CRTST | CHECK USER 1 INPUT STATUS |

ROUTINE TO DIRECT PROGRAM TO INPUT ROUTINE SPECIFIED BY IOBYTE

CONIN:

| 4A29 JA0300 | LDA IOBYTE | GET IOBYTE |
| 4A2C E003 | ANI 03 | CHECK CONSOLE SEGMENT |
| 4A2E CA684A | JZ TTYIN | INPUT FROM TTY |
| 4A31 FE01 | CPI 01 | SET FLAGS |
| 4A33 CA6C4A | JZ CRIT | INPUT FROM CRT |
| 4A36 FE02 | CPI 02 | SET FLAGS |
| 4A38 CA6F4A | JZ BATIT | INPUT FROM BATCH DEVICE |
| 4A3B CA6D4A | JMP CRIT | INPUT FOR USER 1 DEVICE |

ROUTINE TO DIRECT PROGRAM TO CONSOLE OUTPUT ROUTINE AS SPECIFIED BY IOBYTE.

CONOT:

| 4A3E JA0300 | LDA IOBYTE | GET IOBYTE |
| 4A41 E003 | ANI 03 | CHECK CONSOLE SEGMENT |
| 4A43 CA684A | JZ TTYOT | OUTPUT TO TTY |
| 4A46 FE01 | CPI 01 | SET FLAGS |
| 4A48 CA6D4A | JZ CRIT | OUTPUT TO CRT |
| 4A4B FE02 | CPI 02 | SET FLAGS |
| 4A4E CA704A | JZ BATOT | OUTPUT TO BATCH DEVICE |
| 4A50 C3714A | JMP CRIT | OUTPUT TO USER 1 DEVICE |

ROUTINE TO DIRECT PROGRAM TO LIST DRIVER AS SPECIFIED BY IOBYTE

LIST:

| 4A53 JA0300 | LDA IOBYTE | GET IOBYTE |
| 4A56 E000 | ANI OC0H | CHECK LIST SEGMENT |
| 4A58 CA684A | JZ TTYOT | OUTPUT TO TTY |
| 4A5B FE30 | CPI 04H | SET FLAGS |
| 4A5D CA6D4A | JZ CRIT | OUTPUT TO CRT |
| 4A60 FE80 | CPI 08H | SET FLAGS |
| 4A62 CA724A | JZ LPTOT | OUTPUT TO LINE PRINTER |
| 4A65 C3734A | JMP LOUT | OUTPUT TO USER LIST DEVICE |
# ACTUAL DEVICE DRIVERS WOULD BE PLACED HERE. FOR DEMONSTRATION PURPOSES, RET INSTRUCTIONS ARE USED.

**** TTY ****

; ROUTINE TO CHECK INPUT STATUS OF TTY
TTYST:
4A68 C9
RET ; FOR DEMO

; ROUTINE TO INPUT FROM TTY
TTYIN:
4A69 C9
RET ; FOR DEMO

; ROUTINE TO PRINT TO TTY
TTYOT:
4A6A C9
RET ; FOR DEMO

**** CRT ROUTINES ****

; ROUTINE TO CHECK CRT INPUT STATUS
CRTST:
4A6B C9
RET ; FOR DEMO

; ROUTINE TO INPUT FROM CRT
CRTIN:
4A6C C9
RET ; FOR DEMO

; ROUTINE TO PRINT TO CRT
CRTOT:
4A6D C9
RET ; FOR DEMO

##### BATCH ROUTINES #####

; ROUTINE TO CHECK MODEM INPUT STATUS
BATST:
4A6E C9
RET ; FOR DEMO

; ROUTINE TO INPUT FROM MODEM
BATIN:
4A6F C9
RET ; FOR DEMO

; ROUTINE TO PRINT TO MODEM
BATOT:
4A70 C9
RET ; FOR DEMO

##### USER CONSOLE 1 #####

; PRINTOUT TO VIDEO CARD
UC1OT:
4A71 C9
RET ; FOR DEMO

##### LIST DEVICES #####

; OUTPUT TO A DOT MATRIX PRINTER
LPTOT:
4A72 C9
RET ; FOR DEMO

; OUTPUT TO DAISY WHEEL PRINTER
UL1OT:
4A73 C9
RET ; FOR DEMO

4A74
END
ELECTRONIC CONTROL TECHNOLOGY's R²I/O is an S-100 Bus I/O Board with 3 Serial I/O Ports (UART's), 1 Parallel I/O Port, 4 Status Ports, 2K of ROM with Monitor Program and 2K of Static RAM. The R²I/O provides a convenient means of interfacing several I/O devices, such as CRT terminals, line printers, modems or other devices, to an S-100 Bus Microcomputer or dedicated controller. It also provides for convenient Microcomputer system control from a terminal keyboard with the 8080 Apple ROM monitor containing 26 Executive Commands and I/O routines. It can be used in dedicated control applications to produce a system with as few as two boards, since the R²I/O contains ROM, RAM and I/O.

The standard configuration has the Monitor ROM located at F000 Hex with the RAM at F800 Hex and the I/O occupies the first block of 8 ports. Jumper areas provide flexibility to change these locations, within reason, as well as allow the use of ROM's other than the 2708 (e.g. 2716 or similar 24 pin devices). Baud rates are individually selectable from 75 to 9600. Voltage levels of the Serial I/O Ports are RS-232.

8080 APPLE MONITOR COMMANDS

A - Assign I/O
B - Branch to user routine A-Z
C - Undefined
D - Display memory on console in Hex
E - End of file tag for Hex dumps
F - Fill memory with a constant
G - GOTO an address with breakpoints
H - Hex math sum & difference
I - User defined
J - Non-destructive memory test
K - User defined
L - Load a binary format file
M - Move memory block to another address
N - Nulls leader/trailer
O - User defined
P - Put ASCII into memory
Q - Query I/O ports: Qi (N)-read I/O; Go(N)-send I/O
R - Read a Hex file with checksum
S - Substitute/examine memory in Hex
T - Types the contents of memory in ASCII equivalent
U - Unload memory in Binary format
V - Verify memory block against another memory block
W - Write a checksummed Hex file
X - Examine/modify CPU registers
Y - 'Yes there' search for 'N' Bytes in memory
Z - Z END' address of last R/W memory location

Specializing in Quality Microcomputer Hardware
Building Blocks for Microcomputer Systems, Control and Test Equipment
Card Cages, Power Supplies, Mainframes, CPU's, Memory, I/O

ELECTRONIC CONTROL TECHNOLOGY
763 Ramsey Ave., Hillside, N.J. 07205
(201) 888-8080
CP/M is available for a variety of disk drives, controllers and methods, including single and double-density, hard and soft sectoring, 8-inch and 5-1/4-inch disks and Winchester hard disk drives, all of which vary considerably in their disk primitives. In this article, for the sake of simplicity, we consider only the standard distribution version of CP/M Version 1.4, issued on a single-density, soft-sectored, 8-inch disk.

DISK ORGANIZATION
Main Divisions of Disk Space

The standard soft-sectored, single-density, 8-inch disk is divided into 77 Tracks (numbered 0 through 76), and there are 26 Sectors (numbered 1 through 26) per track. This conforms to the IBM 3740 disk layout; such disks are called "IBM-compatible".

Each sector stores 128 data bytes; the two Cyclic Redundancy Check bytes and other overhead bytes which follow the data are not included in this count. Thus, the total storage space is 77*26*128 = 256,256 bytes. This, too, follows the IBM format, but again is a function of a BDOS table; it is perfectly possible to set the sector size to any multiple of 128 by changing the table entry, but files would not then be portable except to another system with the same blocking factor.

On every disk that runs under CP/M, the storage space is divided into three distinct areas:

- CP/M System Area
- File Directory Area
- File Storage Area

CP/M System Area. Tracks 0 and 1 are always reserved for the CP/M system, although the system need not be present on every disk. The coldstart loader is contained in Track 0 Sector 1; the CCP and BDOS occupy the rest of Track 0 as well as 17 sectors on Track 1; the remaining nine sectors (1152 bytes) of Track 1 are available for the CBIOS. The number of sectors actually used by the CBIOS depends on what drivers and features are included by the controller manufacturer.

File Directory Area. Sixteen sectors on Track 2 are always reserved for the file directory. Each directory entry is 32 bytes long; thus, there is room for (16*128)/32 = 64 entries in the standard system. Note, however, that sector allocation for the directory is controlled by a table in the BDOS; OEMs licensed by Digital Research Inc. to reconfigure the system can expand the number of directory entries to 255 by changing this table.

File Storage Area. The remaining ten sectors on Track 2 and all sectors on Tracks 3 through 76 are available for files.

Logical/Physical Sector Mapping

The standard logical record is one sector (128 bytes), and a file may occupy any number of sectors from zero up to the full capacity of the disk. Logically consecutive records are not physically contiguous on the disk. This is because the disk controller must process the CRC bytes after reading Logical Sector N, to verify that there were no read errors. Also, the BDOS has some housekeeping chores to perform. If Logical Record N+1 were in fact physically adjacent to Record N, it would probably pass under the read head before the chores were complete; the system would then have to wait until it came round on the next revolution of the disk, about 16 milliseconds later. This would make sequential reading unacceptably slow.
For this reason, logically consecutive records are mapped onto the disk with several physical sectors between each. The standard skew (sometimes called "interlace") for CP/M is six sectors, to be IBM-compatible, and is shown in Figure 1; this mapping is identical for all directory and file storage tracks. The translation from logical record numbers to physical sector numbers is performed by a lookup table that is usually in BDOS, though some versions put the table in the BIOS. When the table is in the BDOS, disk utilities that use the disk primitives directly must provide a separate translation table of their own. Thus, an application program using the disk primitives to gain access to logical Sectors 19, 20, and 21 of Track 3, would in fact access physical sectors 6, 12, and 18 on that Track. After completing the housekeeping for Sector 6, there is only a minimal wait before Sector 12 (the next in the logical sequence) arrives under the read head.

The routine that performs the logical/physical sector mapping is transparent to the user. In effect this routine says, "Whenever you give me a logical record number, I will convert it to a physical sector number. You don't need to know what that number is, but my mapping will give you quicker access to the data area."

**KEEPING TRACK OF DISK SPACE USAGE**

Unlike some other microcomputer operating systems, CP/M does not require that the size of a file be specified at the time of creation. Instead, space on the disk is allocated dynamically, as needed. Space that is released as the result of closing a file from which at least 1 K has been deleted, can immediately be re-used by another file. The tools that permit this dynamic space allocation are:

- The Allocation Bit Map
- The File Control Block (FCB)
- The Directory

**Allocation Bit Map**

For every drive configured in the system, the BDOS maintains a space allocation bit map consisting of 243 individual bits. This map is read into memory when the drive is logged in, is modified during Write operations, and is written back to the disk each time a file on that disk is closed. Typing Ctrl-C erases all bit maps from memory except those for Drive A and for the currently logged-in disk.

Each bit in the map represents a group (sometimes called a "cluster") of eight logically consecutive sectors on the disk. The bit positions and their associated groups are numbered 00 through 23 hex (see Figure 2). The first two bits are associated with the first sixteen logical sectors on Track 2. These two groups (00 and 01) contain the file directory, and bits 00 and 01 in the allocation map always contain 1's, even when no directory entries have yet been made. This ensures that the directory can never be overwritten by a file.
CP/M Connection cont’d...

This process has several important results:

- The minimum space that can be occupied by any file (even an empty file) is eight sectors (1K).
- Because BDOS always searches the allocation bit map from the beginning on a Write request and allocates the first free group it finds, logically, consecutive components of a file may be physically located anywhere on the disk and not necessarily in Track/Sector order.
- A Write request will never be denied until the disk is too full to hold the amount of data to be written. Of course, denial of a Write request is a fatal error unless the application program makes provision for mounting a fresh disk in such circumstances, but it very seldom occurs if reasonable care is taken. Use the STAT utility to check available space before undertaking any operation that creates backup or temporary files.
- Disk space is efficiently used. In other systems that require file size to be specified, overcaution can result in large amounts of unused space that is not available to other files. This can only be recovered by copying the data to a new file with the proper size specification. The same procedure has to be followed if it is desired to expand a file that has already used the space originally allocated to it.

File Control Block (FCB)

An FCB is a 33-byte block of read/write memory containing all the information needed by BDOS to find a file on the disk and to access any specified record. Whenever a new file is created, an FCB must be created for it. The area from 005C to 007C hex is the default FCB area used by the CCP; it may also be used by transient programs. If a transient program requires more than one file to be open at the same time, the program must create an FCB for each file that is to be accessed. These FCBS should be in the TPA.

**FCB Layout.** The layout of an FCB is shown in Figure 3. When a file is first created, the CCP or user program first clears all bytes of the FCB to zero, and then initializes the first thirteen bytes as follows.

**ET. Byte 0.** The CP/M Manual defines this as “Entry Type, not currently used but assumed zero.” While in the FCB area, this byte remains 0.

**FN. Bytes 1 thru 8.** The CCP or user program places the filename in this field, left-justified. If the name has fewer than eight characters, the remaining bytes are padded with ASCII blanks (20 hex).

**FT. Bytes 9 thru 11.** The CCP or user program places the 3-character file type in this field. Note that the period which separates filename and type in a command is only a delimiter and is not put in the FCB. If the file type has fewer than three characters, the remaining bytes of the FT field are padded with ASCII blanks. If the file is a temporary file, this field will contain "$\$$\$".

**EX. Byte 12.** This byte, initialized to zero, indicates the file extent number. As we shall see, an FCB describes a file segment up to 16K in size, i.e., 128 records (sectors). When 128 sectors have been written, bytes 0 thru 31 of the FCB are copied to the first free slot in the directory area, and the Extent number in the FCB is incremented. Thus, we shall find a separate directory entry, each containing a different extent number, for every 16K segment of a large file.

**Bytes 13 and 14 are not used and should always be zero.**

**RC. Byte 15.** This byte, initialized to 0, contains the current number of records in the Extent described by the FCB. As new records are written to the disk, this count is updated by BDOS. Transition of this count from 7F to 80 is the signal for BDOS to copy the FCB to the directory area of the disk and to create a new FCB with the EX and NR bytes updated.

**DM. Bytes 16 thru 31.** This is the Disk Map area, and is initialized to zeros. When a file is being built, the first Write request causes BDOS to insert the number of the group allocated into Byte 16. No further updating takes place in this field until all sectors of the group have been written. Then BDOS allocates another group and inserts its number into Byte 17, and so on, until all 16 groups (128 sectors) have been written.

**File Directory**

The BDOS maintains a directory for each disk. Upon booting CP/M, the contents of Groups 01 and 02

---

**IMPORTANT NOTE:** Because the FCB is not written to the disk directory area until either 128 records have been written or the file is closed, a system crash can cause the apparent loss of up to 128 records. The data is on the disk but is not recorded in the directory. In applications that entail much data entry, it is good practice to close the file frequently and re-open it; this can avoid painful reconstruction of the directory and the possible destruction of vital data as the result of overwriting from other files after a crash.

**NR. Byte 32.** This byte, initialized to zero, is updated by BDOS during sequential file operations, and shows the number of the next record to be read or written. For random access, the transient program must place the number of the record to be accessed in this byte before issuing the function call to BDOS. Note that this byte is not copied to the directory entry; it is meaningful only when the file has been opened.

**FCB Location.** Before we go on to discuss the directory, it is important to emphasize that there is no restriction on the location of an FCB. The CCP and DDT use the area from 005C to 007C hex; this is known as the default FCB area, and is usually given the symbolic label TFCB. When a large file has more than one 16K extent, sequential write operations build the data for each extent in the default FCB area. Sequential read operations cause each directory entry for the file to be fetched into TFCB, in turn. However, a user program may allocate enough memory to hold all the FCBS of a file simultaneously, passing the address of the appropriate FCB to BDOS as one argument of each access request. This will be explained in more detail when we discuss file access operations. Much time and head movement can be saved during random read operations if all of the FCBS for a file are available in RAM, so that they do not have to be fetched from the disk each time a new extent is accessed.

---

**File Directory**

The BDOS maintains a directory for each disk. Upon booting CP/M, the contents of Groups 01 and 02
the numbers of the 8-sector groups allocated to the
writing, closing the file causes BOOS to copy the FCB
file.
the file name and type. The extent number is in byte
nnnC, and the record count in nnnF. The second line
to the disk immediately.
causes the appropriate directory entry to be copied
of this layout may help to recover a file after a system
address digit always even) contains
occupies 32 bytes in two lines. The first line (3rd
address digit always odd) of the entry contains
(16 sectors) are read sequentially from the logged-in
disk, and the bit map for that drive is recalculated from
the DM bytes of each entry. A request to open a file
causes the appropriate directory entry to be copied
into the FCB area. If the file has been opened for
writing, closing the file causes BDOS to copy the FCB
back to the directory area and to rewrite all 16 sectors
to the disk immediately.

Directory Layout. A set of typical directory entries
is shown in Figure 4. They are for the same disk as the
Allocation Map in figure 3. Note that each entry
occupies 32 bytes in two lines. The first line (3rd
address digit always even) contains 00 in the first byte
to indicate that the entry is for a valid file), followed by
the file name and type. The extent number is in byte
nnnC, and the record count in nnnF. The second line
(3rd address digit always odd) of the entry contains
the numbers of the 8-sector groups allocated to the
file.

Restoring Erased or Crashed Files. An understanding
of this layout may help to recover a file after a system
crash or one that has been accidentally erased. The
CP/M ERAse command and the Basic KILL command
do not in any way modify the disk file. They merely
issue a Delete request to BDOS which places E5 in the
first byte of the directory entry to mark it as deleted.
BDOS also scans the group allocations in line 2 of the
entry and sets the corresponding bits in the Allocation
Map to 0, thereby freeing these groups for re-use. The
file data on the disk remains intact until a Write
request to BDOS finds one of these groups free and
writes the contents into the FCB. If it is desired to alphabetize the directory and to
purge entries for deleted files, the SAP (Sort and
Pack) utility by Bruce Ratoff (CPMUG Volume 19) can
be used. This utility reads the directory from the disk,
copying only the active entries into an empty 2K
buffer. It then sorts the selected entries into alpha-
numeric filename-type-extent order, fills the rest of
the buffer with E5, and then writes the sorted and
purged directory back to the disk. BE CAREFUL,
however. Early versions of SAP operate ONLY on
Drive A and are constructed for a particular system
size. And, to the best of my belief, existing versions of
SAP do not work on double-density systems. Be sure
to check the source code, and try it on a backup disk
first.

User Program File Access Procedures
When a file-related console command is given
(SAVE, ERA, REN, DIR, TYPE) with a drive, filename,
and file type, or when a CP/M utility (STAT, PIP, ASM,
DDT, etc.) is invoked with a filename as its argument,
the CCP or the utility perform all functions required
to access the named file, including the creation and
updating of the FCB and directory entry. We are here
concerned only with the procedures that must be
performed by user programs to create, modify, or read
data files. Some general principles are explained first;
these are followed by some concrete examples.

BDOS Function Calls
CP/M provides 27 different functions, all of which
are available to user programs. Functions 1 through
11 relate to peripheral I/O, and are discussed else-
where. Functions 12 through 27 are disk I/O functions.
Only those concerned with creating a new file, reading

Figure 3. Layout of File Control Block
CP/M Connection cont’d...

from or writing to an existing or newly created file, or deleting a file are discussed here.

A BDOS function call (that is, a request to BDOS to perform some function) always consists of three operations:

- Load Register C with the function number of the desired operation.
- Load Register Pair DE with the address of the FCB for the file to be accessed. For function 26, load the address of the buffer to be used for disk reads and writes.
- CALL BDOS (entry point is 0005H).

Some, though not all, functions return a result. Single-byte results are returned in the A register. Double-byte results are returned with the low byte in the A register and the high byte in the B register. It is the responsibility of the user program to interpret and use any results returned by BDOS. NOTE: BDOS uses all the registers. If any register values have to be preserved, save them before the BDOS function call and restore them when the function is complete.

Log-In Disk (Function 14)

If the file to be accessed is not on the same disk as the user program, the drive on which the file is (or will be) stored must be logged in. That is, its Allocation Map must be reconstructed in memory before any access can be attempted. Put function number 14 (0EH) in the C register, clear the D register, and load E with the drive number to be logged in. Then call BDOS. No results are returned.

If your program calls for a change of disk, it MUST call for a log-in; if it does not, BDOS will attempt to use the allocation bit map left over from the previous disk on that drive, and existing data on the new disk may be overwritten and permanently lost.

Create a New File

First, allocate space for an FCB. If no other file is open, the TFCB at 005CH may be used; if that is already in use, allocate FCB space (33 bytes) in the transient program area (TPA). Then move the filename (8 characters, left-justified, padded as necessary with ASCII blanks) and the file type (3 characters, left-justified, padded if necessary) into bytes 1 through 11 of the FCB (byte 0 must contain zero).

Load Register C with function number 22 (16H, Make File), load Register Pair DE with the address of the FCB, and call BDOS. BDOS returns the byte address of the directory entry allocated to the file (i.e., an address in the range 00 through 7FH that is relative to the start of the sector in which the entry will be stored on the disk). If the directory is already full, BDOS returns OFFH in the A register; the user program must check for this and take appropriate action if the directory is full. One possible course would be to print an error message instructing the operator to dismount the current disk and mount a blank formatted disk on the same drive. Upon receiving confirmation via the console that a new disk is mounted, start the operation over by logging in the disk (Function 14) and repeating the Make.

Opening an Existing File

If the file to be accessed already exists, it must be opened before reading or writing can take place. If the file is not on the same disk as the user program, it must be mounted and logged in as described above; if it is on the same disk, the log-in was done when the user program was called.

To open the file, do the following:

- Allocate space for the FCB.
- Move the filename and type into bytes 1 through 11 of the FCB as described for a Make, above.
- Load the C register with function number 15 (0FH, Open File). Load Register Pair DE with the with the address of the FCB.
- CALL BDOS (Entry point is 0005H).
- Clear register A and wait for the byte address of the directory entry if the file is successfully opened, or OFFH if the file cannot be found.

NOTE: Successful opening of a file says nothing about the mode in which it can be accessed. It merely indicates that the file exists and that its directory entry has been copied into the FCB area. The user program may either read from or write to the file, sequentially or randomly. If only Read operations are performed, the file need not be closed later (although it is good practice to do so). If any kind of Write operation is performed, the file MUST be closed later, in order to ensure that the added or modified space allocations are permanently recorded on the disk, both in the allocation bit map and in the directory.

Buffer Addressing

The starting address of the buffer from which data is to be written to disk, or into which data is to be read from disk, is called the DMA (Direct Memory Access) address. The minimum size of the buffer is 128 bytes, corresponding to one complete sector. Increases of buffer size must be in multiples of 128. The term “DMA” is not strictly accurate unless the controller contains DMA hardware that pre-empt the data bus and transfers a specified number of bytes (starting at the DMA address) at high speed, without intervention of the CPU. However, the term is convenient and has become standard in CP/M.

Unless otherwise specified by a user program, BDOS assumes that all data transfers will take place via the 128-byte buffer at locations 0080H through 00FFH. This is called the Default Buffer, and the standard name of its starting address is TBUFF. This buffer is also used by the CCP for string input from and output to the console.

A user program can change the disk buffer address with a Set DMA function call to BDOS. The procedure is:

- Load Register C with the function code (26=1AH=Set DMA).
THE NEXT GENERATION OF MICROCOMPUTERS IS HERE AT QUASAR DATA PRODUCTS

16BIT POWER
Z-8000
AND STILL RUN YOUR 8 BIT SOFTWARE

Z-8000 SERIES 16 BIT CPU S-100 BOARD — CAN BE PLUGGED INTO YOUR EXISTING SYSTEM

- FULLY S-100 IEEE COMPATIBLE
- SUPPORTS EXISTING 8 BIT MEMORY AND 8 BIT PERIPHERAL BOARDS
- CAPABLE OF READING AND/OR WRITING 8 BIT, 16 BIT OR MIXED 8 BIT AND 16 BIT MEMORIES AUTOMATICALLY
- 8 BIT AND/OR 16 BIT PERIPHERAL MODULES CAN SIMULTANEOUSLY CO-EXIST IN THE SAME BUS WITHOUT ANY MODIFICATIONS
- CAPABLE OF OPERATING AS A SLAVE PROCESSOR TO ENABLE YOUR EXISTING CPU TO CONTROL THE Z-8000
- SUPPORTS ON BOARD HARDWARE SINGLE STEPPING
- SUPPORTS EITHER SEGMENTED CPU OR NON-SEGMENTED CPU
- POWER ON AND RESET JUMP DIP SWITCH SELECTABLE
- JUMPER SELECTABLE 2 OR 4 MHZ. OPERATION
- DIP SWITCH SELECTABLE NUMBER AND TYPE OF WAIT STATES
- SOFTWARE
- Z-80 EMULATOR ENABLES YOU TO EXECUTE YOUR EXISTING 8 BIT SOFTWARE WITHOUT ANY MODIFICATIONS AND ALLOWS YOU TO RUN CP/M IMMEDIATELY
- EXTENDED MONITOR, DEBUGGER, DISASSEMBLER

INDUSTRIAL QUALITY
Z-80

- REAL TIME CLOCK
- RESET-JUMP CIRCUIT ALLOWS CPU TO JUMP TO MONITOR SOFTWARE ON RESET
- CAPABLE OF PROGRAMMING AND VERIFYING INTEL 2716 ON BOARD WITH EXTERNAL POWER SUPPLY
- UTILIZES Vectored Interrupts of Z-80 CPU
- MONITOR PROVIDED

INDUSTRIAL QUALITY
SYSTEMS

Z-80 SERIES 8 BIT CPU S-100 BOARD
- 4 MHZ, Z-80 MICROPROCESSOR
- FLOPPY DISK CONTROLLER ALLOWING SINGLE- AND DOUBLE-DENSITY USING EITHER 8" or 5½" DISK DRIVES
- ROOM FOR THREE 2716 EPROMs or THREE 2316 ROMs
- TWO RS-232C SERIAL PORTS
- TWO PARALLEL PORTS
- HARD DISK CONTROL VIA THE PARALLEL PORTS

QDP-8100 WITH 2 MEGABYTES STORAGE STANDARD (OPTIONAL 4 MEGABYTES)
- Z-8000 SERIES 16 BIT CPU S-100 BOARD - SEE ABOVE
- SOFTWARE (PROVIDED WITH SYSTEM)
- CP/M 2.2 OPERATING SYSTEM
- BASIC
- Z80/8080 EMULATOR
- MONITOR, DEBUGGER, DISASSEMBLER
- SOFTWARE OPTIONS: PASCAL
- UNIX OPERATING SYSTEM

QDP-100 WITH 2 MEGABYTES STORAGE STANDARD (OPTIONAL 4 MEGABYTES)
- Z-80 SERIES 8 BIT CPU S-100 BOARD - SEE ABOVE
- SOFTWARE (PROVIDED WITH SYSTEM)
- CP/M 2.2 OPERATING SYSTEM
- BASIC
- ACCOUNTS RECEIVABLE, GENERAL LEDGER, ACCOUNTS PAYABLE, PAYROLL WITH COST ACCOUNTING
- OPTIONAL SOFTWARE: FORTRAN, PASCAL, COBOL, C

TELETek DBL. DENSITY, DBL. Sided
Disk Controller Board

- MFE Double Sided - Double Density 8" Floppy Disk Drives. (the best) .......................... '650**
- Using the Teletek Controller under CP/M, THIS DRIVE WILL GIVE YOU ALMOST ONE MEGABYTE PER DISK DRIVE.
- Power supply for above .................................. '110**

PAPER TIGER
Includes Graphics .................................. '949**
Cable for TRS-80 ................................ '39**
Call for Apple

MFE 64K Dynamic RAM
- 16K - '250**
- 32K - '350**
- 48K - '450**
- 64K - '549**

QUASAR FLOPPY SYSTEM
- Two MFE DBL sided drives
- Cable
- Case & Power Supply
- Wood cabinet .................................... '1895**

QUASAR 2 MEG FLOPPY
- 2 MFE double sided drives
- Teletek disk controller board
- Power supply & cable
- Wood cabinet
- CP/M version 2.2 & bios
- Assembled & tested .......................... '2295**

Dealer Inquiries Invited. Hours: 9-5:30 M-F
Specifications Subject To Change

Checks, money orders accepted
Add $2.50 freight charges on orders under 10 lbs. Over 10 lbs. F.O.B. Cleveland

S-100 MICROSYSTEMS
25151 Mitchell Dr., No.Olmsted, Ohio 44070 (216)779-9387
Reading and Writing

After a file has been opened, a separate Read or Write request must be issued for each and every sector transferred. Unless otherwise specified by the user program, BDOS assumes that all transfers will take place via the Default Buffer starting at TBUFF (0080H). The user program is responsible for emptying the buffer after each Read (or processing the data while it is still in the buffer), and for filling the buffer before each Write. The request procedure is:

- Load the DE Register pair with the starting address of a user buffer.
- CALL BDOS (Entry point at 0005H).

BDOS does not return anything.

Random Reading

Once a file has been opened, random access to any record is possible by placing the desired record number in byte 31 of the FCB before issuing a Read or Write request (remember that FCB bytes are numbered from 0). Some computation is required here, first to derive the logical record number from the block number (if blocks are larger than 128 bytes), and then to find what extent this record is in.

Suppose that our logical records are 256 bytes (2 sectors) long, and we wish to access record 134. The data we want is in the two sectors starting at 134*2=268 in the sector sequence. However, since a file extent can hold only 128 sectors, sector 268 must be in the third extent. Extent numbers start at 0, so this will be numbered 02. The number of sector 268 relative to the start of Extent 02 is found by taking the remainder of 268 modulo 128; that is, 268%2=2, and the remainder is 268-(128*2)=12 decimal. Since Next Record counts in the FCB also start at 0, the required sector number is 11 (0BH).

Thus, before issuing the access request we must first fetch the directory entry for Extent 02 into the FCB area, and place 0BH in the NR field (Byte 31). Now we issue a Set DMA request pointing to the start of a 256-byte buffer, followed by a Read request for the first half of our record. To obtain the second half, we must add 128 to the DMA address and issue a new Set DMA request. We do not need to change the NR field of the FCB because this was incremented automatically by BDOS after the first read, so we finish the operation merely by issuing the second Read request.

Random Writing

Some care must be taken when writing randomly. If we wish to write record 129, for example, we must first have created space for records 1 through 128. We can write 128 records containing nulls, and then add record 129 to the end of these; however, this may be wasteful of disk space, and we could run into trouble if we attempt to write record 2001 (or some high number). Most data management systems use a special CREATE program to create a file of finite size, and then an UPDATE program that enters data into this file in a manner that makes efficient use of the space. There have been a number of articles during the last year on hashing techniques, tree techniques, and indexed sequential access methods; consult these for further details, which are outside the scope of this article.

Closing a File

It is not necessary to close a file if ONLY read operations were performed on it. This is because reading alone does not change either the Allocation Map or the directory entries for the file. Closure is highly desirable, however, to maintain upward compatibility of the user program with revisions later than 1.4 of CP/M. In a multi-user system, for example, the file would have to be closed before any other user program could access it.
Further, if any type of writing was done, the space allocations for the file were probably changed, and must be written back to the disk to ensure integrity of the data. The Close function copies the current FCB to the matching directory entry, if one exists, or to the first free slot in the directory area if the current FCB describes a new Extent. Then the allocation bit map is written back to the disk area on which it resides, and the entire updated directory is written out to the first sixteen sectors of Track 2.

To close a file, do the following:

- Load register C with the function code (16=10H=Close).
- Load Register Pair DE with the address of the FCB for the file.
- CALL BDOS (entry point is at 005H).
End-of-File Detection

Files of type .COM and .HEX do not contain any built-in end-of-file (EOF) marker. If BOOS starts to process a Read request and finds that the count in the NR field of the FCB is greater than the count in the RC field, the request is aborted and a completion code of 1 (read past end of file) is returned. A transient program may use this indication to break out of a data transfer loop; more usually, such a loop is initialized to read only the number of sectors specified by the RC field.

ASCII files of types ASM, TXT, DOC, etc. can also use the above methods. However, the transient programs that process such files (assemblers, editor, text formatters, etc.) expect to find at least one ctrl-Z (1AH) code after the CRLF of the last record in the file. Some programs fill all unused space in the last sector with this code. The EOF marker is not recognized by BOOS, but it acts as a signal to the transient not to read any further sectors, and to ignore the first EOF marker and all subsequent bytes in the buffer.

When random file access is in progress, the user program should always check the completion code returned by each Read or Write request, because BOOS distinguishes between an attempt to read a sector beyond the true end of file (code 1), and an attempt to read a sector which is within the file area but has no data in it (code 2). The latter condition could occur while building a tree, or while using hashed key techniques.

Deleting or Renaming a File

A user program can delete or rename a file by function calls to BOOS. The delete procedure is:

- Place the name and type of the file to be deleted in bytes 1 through 11 of an FCB (bytes are numbered from 0).
- Load Register C with the function code (19=13H=Delete).
- Load Register Pair DE with the address of the FCB.
- CALL BOOS.

No information is returned by BOOS after a Delete request.

The rename procedure is:

- Place the old name and type of the file to be renamed in bytes 1 through 11 of the FCB.
- Place the new name and file type in bytes 16 through 26 of the FCB.
- Load Register C with the function code (23=17H=Rename).
- Load Register Pair DE with the address of the FCB.
- CALL BOOS.

If BDOS finds a directory entry matching the filename and type in FCB bytes 1-11, it changes these to the filename and type specified in FCB bytes 16-26 and returns the byte address (within the sector) of the changed entry, in Register A. If no matching entry is found, BDOS returns 255 (OFFH) in Register A. The user program should check the completion code and take appropriate action if the renaming was not successful. After a successful renaming, a Close request must be issued for the file under the new name, otherwise the modified directory will not be written to the disk.

In the next issue of S-100 MICROSYSTEMS we will examine: "Implementing the IOBYTE Function".

S-100 MICROSYSTEMS' Bugs

The list of processor boards and manufacturers in the July/August issue was incomplete in that all three processor boards that IRISystems uses were omitted. These are:

- a. 8085 - TEI
  5075 South Loop East
  Houston TX 77033

- b. 8080 - Industrial Micro Systems
  628 North Eckhoff Street
  Orange CA 92668

- c. Z80 - International Product Development Inc.
  1708 Stierlin Road
  Mountain View, CA 94043

Also, Digicomp Research's Pascal 100 contains both the Western Digital Microengine and a Z-80 processor. Two portions of Chris Terry's CP/M Connection were transposed in the July/August issue. Corrections can be found on the first page of this month's installment "Part II-File Operations."
Hard disk and hardtape™ control

Up to 2400 Megabytes of hard disk control for the S-100 bus.

Konan's SMC-100 interfaces S-100 bus microcomputers with all hard disk drives having the Industry Standard SMD Interface. It is available with software drivers for most popular operating systems. Each SMC-100 controls up to 4 drives ranging from 8 to 600 megabytes per drive, including most "Winchester" drives -- such as Kennedy, Control Data, Fujitsu, Calcomp, Microdata, Memorex, Ampex, and others.

SMC-100 is a sophisticated, reliable system for transferring data at fast 6 to 10 megahertz rates with onboard sector buffering, sector interleaving, and DMA.

SMC-100's low cost-per-megabyte advanced technology keeps your micro computer system micro-priced. Excellent quantity discounts are available.

Konan's HARDTAPE™ subsystem...very low cost tape and/or hard disk Winchester backup and more.

Konan's new DAT-100 Single Board Controller interfaces with a 17½ megabyte (unformatted) cartridge tape drive as well as the Marksman Winchester disk drive by Century Data.

The DAT-100 "hardtape" system is the only logical way to provide backup for "Winchester" type hard disk systems. (Yields complete hard disk backup with data verification in 20-25 minutes.)

Konan's HARDTAPE™ subsystem is available off the shelf as a complete tape and disk mass storage system or an inexpensive tape and/or disk subsystem.

Konan controllers and subsystems support most popular software packages including FAMOS™, CP/M® version 2.X, and MP/M.

Konan, first (and still the leader) in high-reliability tape and disk mass storage devices, offers OEM's, dealers and other users continuing diagnostic support and strong warranties. Usual delivery is off the shelf to 30 days with complete subsystems on hand for immediate delivery.

Call Konan's TOLL FREE ORDER LINE today:
800·528·4563

Or write to Bob L. Gramley
Konan Corporation, 1448 N. 27th Avenue
Phoenix, AZ 85009. TWX/TELEX 9109511552

CP/M® is a registered trade name of Digital Research.
FAMOS™ is a trade name of MVT Micro Computer Systems.
HARDTAPE™ is a trade name of Konan Corporation.
In the last issue I described a program for a North Star disk system which implements input/output redirection, a printer SPOOL and a command stack. In order to redirect I/O to and from the disk as well as drive a printer with the contents of a disk file, this program must necessarily access disk files. I consider the part of the program which does North Star disk I/O to be a skeleton operating system. By my definition, an operating system manages the physical machine resources to provide an environment to execute programs. One of the significant features of an operating system is the management of all input/output. The simpler an operating system makes I/O appear to the program environment, the easier it is to develop programs. One aspect of operating systems which distinguishes them is the specification of the program interface for input/output operations.

In the case of the North Star disk system, the I/O interface available for software development is extremely low level. The North Star DOS provides character I/O for terminals and other character oriented devices such as paper tape devices and line printers. No character I/O is available for disk files. The only access North Star DOS provides for disk files can be best described as the unbuffered do-it-yourself variety. North Star Basic does provide character by character as well as record access to disk files, but only for North Star Basic programs. For disk access, the North Star DOS provides a method for naming files and a method for accessing arbitrary blocks of data on the disk. The North Star DOS does not provide any interface which will associate a file name with a particular area of the disk (open a file), read/write a byte or record and close a file. The System Software Manual hints at how this can be done but no software is provided.

In order to implement input/output redirection and a printer spool, I needed to provide a means to buffer disk accesses. To understand this, you need to consider the type of input/output which the hardware controlling a disk device can accept. The most versatile type of disk controller will make the disk device appear as an extension of the computer's main memory. Hence, a limited amount of main memory can be made to appear to be much larger to an application program because the disk controller (and other hardware) is constantly swapping the machine's main memory with the memory available on the disk device. A simpler disk controller, such as the North Star, will only sort out the timing considerations necessary to locate a particular block of data (track and sector) on the rotating disk. Once the particular spot on the disk is found, all the information must be read or written as one block. The North Star DOS entry DCOM will find the requested spot on the disk and then read or write one or more consecutive blocks. The North Star disk controller is set up to recognize 10 such blocks on each track of a disk. The North Star DOS is set up to recognize 35 such tracks for a total of 350 blocks. In order to access one byte in a selected block, additional software is needed.

The process of accessing a single byte or record of data on a disk device requires a whole block of data to be buffered in the computer's main memory. In addition, the application program is greatly simplified if it can refer to a logical block in a named file and let the operating system sort out the actual physical block number, disk track and sector (another name for block) number. The second part of the spooling program implements a version of buffered disk I/O. I had never tried to implement this type of program before, and this experience proved to be very educational. I can now better appreciate the complexity of an operating system such as CP/M.

The CP/M operating system uses a 33-byte string of data to hold relevant information about an open disk file. This 33-byte string of data is called the File Control Block (FCB). The FCB is supplied by the application program and CP/M uses this memory as a scratch pad while it performs disk access. Most operating systems require some type of scratch pad when disk access is performed. I can remember doing battle with IBM's Job Control Language and specifying all that DCB (Data Control Block) information and never really understanding why it was needed. I kept thinking the computer should figure it all out. The only operating system I have used which does not require an area of the application program's memory for a disk I/O scratch pad is UNIX. The UNIX system is completely unique and I hope an affordable operating system of this type will be available someday for an S-100 type.
The operating system I am about to describe uses a 19-byte scratch pad. The layout is as follows:

<table>
<thead>
<tr>
<th>ITEM</th>
<th>LENGTH</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>XXFILE</td>
<td>12</td>
<td>Contains the name of the disk file</td>
</tr>
<tr>
<td>XXUNIT</td>
<td>1</td>
<td>Contains the disk drive number</td>
</tr>
<tr>
<td>XXPTR</td>
<td>1</td>
<td>Pointer to RAM buffer (low byte)</td>
</tr>
<tr>
<td>XXBUF</td>
<td>1</td>
<td>Pointer to RAM buffer (high byte)</td>
</tr>
<tr>
<td>XXADR</td>
<td>2</td>
<td>Disk address for next read/write</td>
</tr>
<tr>
<td>XXLEN</td>
<td>2</td>
<td>Disk file length</td>
</tr>
</tbody>
</table>

The longest filename allowed by North Star DOS is eight characters. I chose to include the drive specification in XXFILE, so this adds two more bytes (a comma and drive number). The filename is terminated with a CR (ASCII 13) and RUBOUT (I use FF hex) so a total of 12 bytes is required. When the disk drive number is stripped from the filename, it is stored in XXUNIT. When XXUNIT is zero, the file is considered closed. The pointer to the buffer used to temporarily hold the block of data is in the next two bytes. The data blocks on North Star disks are 256 bytes in single density. This is a very convenient number since exactly one byte is needed to indicate the next position in the buffer to be read/written. This convenience disappears with double density (which is why the program won't work with double density). To compensate I used this trick in several places: when the low byte overflows—it's time to get/put another block. Now that I reflect on this decision, using tricks such as this (sometimes called magic numbers) makes a program depend too much on the particular hardware that it runs on. Finally, the scratch pad contains two numbers relevant to the particular disk file. The first contains the address of the next physical block which will be read/written when the buffer overflows. The second number is initially the total length (in blocks) of the disk file. The disk address can range up to 349 and the length can range from 0 to 345 (the first 4 blocks on every North Star disk contain the directory), so two bytes are needed for each number.

With these preliminaries out of the way, we can discuss how the program works. The character-by-character write routine starts at location WRDISK. Throughout the program, access to the information in the scratch pad is made through addresses contained in the 8080 machine registers. This allows the scratchpad and buffer to be located (and relocated) anywhere in memory. Each routine starts with comments indicating the information expected in the registers. Error conditions are indicated using the 8080 flag register. Usually the carry bit set indicates an error and the state of the zero flag and minus/plus flag will indicate the type of error. The WRDISK simply saves the byte to be written (A-register) in the buffer and increments the pointer (low byte of the buffer address). If the buffer is not full, the routine returns. If the buffer is full, the routine sets up the registers for the DODSK routine which writes the buffer to disk. Any error in DODSK will be passed back to the calling program by WRDISK.

The RDDISK is very similar to WRDISK and will return the next character from the buffer in the A-register. If all the characters in the buffer have been read, the registers are set up and DODSK is called to fill the buffer. The DODSK routine does the unbuffered disk I/O. This routine uses the DCOM entry in the North Star DOS after the information contained in the scratch pad is updated and all the 8080 registers are set up. The DODSK routine also checks for a length error (trying to read/write more blocks than the file contains). The file length byte is decremented for each block read/written. When this byte in the scratch pad is zero, a length error is indicated. If the North Star DOS routine DCOM indicates an error, DODSK interprets the error to be that the disk is write-protected. This is a good guess if you are sure the information in the 8080 registers upon entry to DCOM is correct. Unfortunately, the System Software Manual does not tell you this.

The SEARCH and CREATE routines are the ones necessary to open a disk file. I don't have an open routine in the operating system. The disk file open is done in the spool driver routines discussed in the last issue. I developed an open routine subsequent to writing this program and have used it for other work. To open a file, the following steps are needed: 1) set up a 19-byte scratch pad and insert the filename, 2) call SEARCH to see if the file exists on the selected disk device (extracted from filename), and 3) call CREATE if the file does not exist. If SEARCH finds the file, the starting disk address and total file length are loaded into the scratch pad. If SEARCH is unsuccessful, the next available address on the disk is in the DE-register pair. This is where CREATE expects to find it so SEARCH must be called before CREATE. Also the CREATE routine requires the desired file type to be in the A-register and the desired file length (in blocks) to be in the BC-register pair. If CREATE is called with a requested length of zero (BC = 0) then all available space on the disk will be used. When this file is closed, the length will be adjusted to the number of blocks actually used. This is the closest I could come to dynamic files in a North Star system. The CREATE routine will determine if the disk is too full to contain the requested file, if the file name is bad and if the disk is write-protected.

The last routine in the operating system is CLSIT which closes a disk file. This routine will put an end-of-file byte (ASCII 1, SOH) in the buffer and write the final buffer to disk. The length of the file contained in the disk directory will be adjusted if the dynamic file sizing was requested. The XXUNIT byte in the scratch pad is zeroed to indicate that the file is closed. If the file was open for reading only, the CLSIT routine should not be used since this file can be closed merely by zeroing the XXUNIT byte. The CLSIT routine will signal an error if the disk in the drive has been disturbed since opening the file.

The remainder of the code in the program is service routines for the operating system and disk spool parts of the program. The largest block of code handles errors when the operating system detects disk problems. Smart error routines are difficult to write. These error routines attempt to give a good message and return after cleaning things up as much as possible. The program ends with a data section which allocates space for program variable storage, the three 19-byte scratch pads for the write, read and output.
MICROSOFT SIGNS UNIX AGREEMENT

Microsoft, Bellevue Wash., has signed an agreement with Western Electric for the rights to develop and market a version of UNIX, developed by Bell Labs. Their version will be specifically for 16-bit microprocessors, such as the 8086, Z8000 and 68000 and will be called "XENIX". UNIX is probably "the" most popular minicomputer time-sharing operating system in current use. It is very popular in the educational community, most probably because we sold it to these institutions for a very low fee. However, it has been gaining in popularity in the business world as well.

The charge for the software package will be based on volume and number of users. The initial fee for a four user system will range from $500 to $300,000. Release of the package is expected in early 1981.

MICROSOFT RELEASES BASIC FOR Z8000

Microsoft again strikes a first with its implementation of Basic for Z8000 based systems. This is the first worthwhile software package to be released for the Basic for 8-bit machines and will execute about five times faster. Microsoft's marketing emphasis for the Z8000 Basic will be primarily to OEMs.

RUMORS

Digital Equipment Corp. (DEC) will soon release a 16-bit microprocessor chip that will be compatible with 8080/820/8600 support ICs. It will have the power of a PDP-11/23. At least one outfit is presently investigating an S-100 implementation with plans of running the popular RT-11 time-sharing operating system.

CPM USER GROUP LIBRARY

Now on North Stardisk

Finally, the largest and best microcomputer software library has been made available on North Star Disk. The CP/M Users Group, a non-profit operation, is making available all 42 volumes of the CP/M Software Library. Until now, it was only available on 8" disks. These disks may be copied without restriction. The usual practice is for a group or club to bus a set of disks, then each member makes a copy from this master set.

The disks are available in either double density Version 1.4 or 2.2 and quad density Version 2.2 formats. They are not available in single density formats. A catalog of the volumes is available for $6 ($11 overseas). The disks are either $8 or $12 per volume depending on whether they require one or two disks per volume (overseas $12 or $16). For information and ordering contact: CP/M Users Group, 1651 Third Ave., New York, NY 10028.

SYMBOL TABLE

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BADCL 1F80</td>
<td>BADESC 1C18</td>
<td>BADMK 1BCC</td>
</tr>
<tr>
<td>BADPA 1B65</td>
<td>BADTY 1B4E</td>
<td></td>
</tr>
<tr>
<td>BELL 0007</td>
<td>BLANK 1C110</td>
<td>BS 0005</td>
</tr>
<tr>
<td>BCKYB 17A9</td>
<td>CKMOT 1B22</td>
<td>CLOSE 1BDC</td>
</tr>
<tr>
<td>CLSPO 1B52</td>
<td>COUT 2000</td>
<td>CR 000D</td>
</tr>
<tr>
<td>CTRLB 1F03</td>
<td>CTRLN 0013</td>
<td>Curr 1906</td>
</tr>
<tr>
<td>DRLRO 1A0D</td>
<td>DISPL 194F</td>
<td>DLOOK 201C</td>
</tr>
<tr>
<td>DORPT 170C</td>
<td>Dosen 2028</td>
<td>Doser 202C</td>
</tr>
<tr>
<td>DRGBL 172F</td>
<td>DISPL 1A77</td>
<td>DISPA 1789</td>
</tr>
<tr>
<td>DSPE 194D</td>
<td>SWRT 201F</td>
<td>ESC 001B</td>
</tr>
<tr>
<td>FILEM 1809</td>
<td>FIN 1659</td>
<td>FLERR 184A</td>
</tr>
<tr>
<td>GETCP 17F5</td>
<td>GETIT 1995</td>
<td>GETSP 1786</td>
</tr>
<tr>
<td>GOSTM 172B</td>
<td>IDDEV 1C55</td>
<td>IPOPA 1C54</td>
</tr>
<tr>
<td>KEBVD 168C</td>
<td>KILL 175C</td>
<td>KILLS 1780</td>
</tr>
<tr>
<td>LNERR 1B3D</td>
<td>LOAD 174A</td>
<td>LOADH 1920</td>
</tr>
<tr>
<td>MO 0010</td>
<td>MOVSL 1987</td>
<td>MOVVL 1A2C</td>
</tr>
<tr>
<td>NEW 1913</td>
<td>NF 1B07</td>
<td>NSERR 1851</td>
</tr>
<tr>
<td>OKI 17E4</td>
<td>OK2 17F3</td>
<td>OKLEN 19C3</td>
</tr>
<tr>
<td>PMSC 1C28</td>
<td>PRACT 1B67</td>
<td>PREX 1958</td>
</tr>
<tr>
<td>PRNAM 1852</td>
<td>PRSTC 195B</td>
<td>PRSTT 1787</td>
</tr>
<tr>
<td>RDERR 1B43</td>
<td>RDPFL 1C5F</td>
<td>RDNAM 195C</td>
</tr>
<tr>
<td>RDUNI 1C7B</td>
<td>READA 1B60</td>
<td>READN 19DC</td>
</tr>
<tr>
<td>RUBOU 1007</td>
<td>SANGER 1C56</td>
<td>SAVPT 173F</td>
</tr>
<tr>
<td>SCPE 1B2C</td>
<td>SKTP 190B</td>
<td>SPLFU 1D00</td>
</tr>
<tr>
<td>SPFL 1C5C</td>
<td>SPLTR 1B60</td>
<td>SPLSR 1844</td>
</tr>
<tr>
<td>STATP 1603</td>
<td>STOP 1C3C</td>
<td>STOPK 1C39</td>
</tr>
<tr>
<td>TOLCM 1BFE</td>
<td>TYEPP 1987</td>
<td>UPDAT 1926</td>
</tr>
<tr>
<td>USRCC 1652</td>
<td>WRDR 15B</td>
<td>WRDS 195D</td>
</tr>
<tr>
<td>WRITE 1880</td>
<td>WRMAR 1874</td>
<td></td>
</tr>
</tbody>
</table>

North Star cont'd...

spool files, the command stack space and finally the 8080 machine stack.

As I pointed out at the start of this article, this program was my first attempt to handle disk input/output. As I reflect on this program I probably would have done many things differently. Nevertheless, I hope you can use this as an example of some of the considerations required when you do disk I/O with the North Star DOS. Maybe this exercise will convince you to buy CP/M for your North Star system. This program will only give you I/O redirection, etc. with North Star DOS. If you want these features and don't want to type this program in, I can supply you a copy on disk for $15. I will include the source code and the assembled version shown here.

36
The North Star Horizon computer can be found everywhere computers are used: business, engineering, home — even the classroom. Low cost, performance, reliability and software availability are the obvious reasons for Horizon's popularity. But, when a college bookstore orders our BASIC manuals, we know we have done the job from A to Z.

Don't take our word for it. Read what these instructors have to say about the North Star Horizon:

"We bought a Horizon not only for its reliability record, but also because the North Star diskette format is the industry standard for software exchange. The Horizon is the first computer we have bought that came on-line as soon as we plugged it in, and it has been running ever since!"
— Melvin Davidson, Western Washington University, Bellingham, Washington

"After I gave a ½ hour demonstration of the Horizon to our students, the sign-ups for next term's class in BASIC jumped from 18 to 72."
— Harold Nay, Pleasant Hill HS, Pleasant Hill, California

"With our Horizon we brought 130 kids from knowing nothing about computers to the point of writing their own Pascal programs. I also use it to keep track of over 900 student files, including a weekly updated report card and attendance figures."
— Armando Picciotto, Kennedy HS, Richmond, California

"The Horizon is the best computer I could find for my class. It has an almost unlimited amount of software to choose from. And the dual diskette drives mean that we don't have to waste valuable classroom time loading programs, as with computers using cassette drives."
— Gary Montante, Ygnacio Valley HS, Walnut Creek, Calif.

See the Horizon at your local North Star dealer.
A Program for Renaming Files on North Star Disks

by Mark M. Zeiger

Presently, to change the name of a file on a North Star formatted disk using the standard DOS commands, these instructions are needed:

1. CReate (new file name) (length)
2. TYpe (new file name) (type) (address)
3. CF (old file) (new file)
4. DElete (old file)

and probably

5. COmpact (if you have release 4 or 5 DOS, you will have to load COMPACT as a separate program).

The program described in this article will rename a file without changing anything on the disk except the directory entry for the file being renamed. Directions are given at the beginning of the program and the user is prompted throughout the program's run. Anytime the program is waiting for a filename, typing a Control-D will list the directory (and restart the program), a Control-X will restart the program, and a Control-C will take the user back to North Star DOS.

The program first asks for the drive number. The user has the option of typing in the drive number (the 'return' key is not needed) or typing Control-C to return to DOS. If a drive is selected, the program to be renamed is then entered followed by a return. If the name is over eight characters, an error occurs and the prompt for the file to be renamed is given again. The program then asks for the new name of the file, again accepting a name of at least one but not more than eight characters. “Delete” (7F hex) or “Rubout” (5FH) may be used to erase a character when inputting names. The program will automatically convert lowercase characters to uppercase. There is a built-in safety check that will not allow entry of just a “Return” when asked for the new name since to do so would be tantamount to erasing the file from the directory (as I unfortunately discovered). After the new name is entered, the file will be updated and the program will restart itself.

If the new name already exists on the disk, the user will be informed of the fact and be asked to pick another name. If the file being renamed is not on the disk, an appropriate message will be given, the directory will automatically be listed, and the user will have to type in both a correct old name and a new name. If there is any doubt as to a name, type Control-D to list the directory.

The RENAME program uses the machine language routines DLOOK, DWRIT, and DLIST as described in the DOS section of the North Star manual. The source program is annotated for clarity (hopefully). Since this is one of my earlier efforts at machine language programming, the code is not as structured as it could be, but since it works perfectly I am not going to rewrite it.

The program flow is as follows:

Print sign-on message
↓
Accept old name (uses INAME)
↓
Accept new name (uses INAME)
↓
Check to see if new name is already in directory
↓
Find old file
↓
Change old file name to new file name in RAM.
Pad name with blanks until eight characters.
↓
Write updated directory to disk.

The major subroutine is INAME which calls input, places the name in the correct buffers (old or new), performs checks on the names, allows deletion of characters, and accepts Control D, X, or C.

Mark M. Zeiger, 198-01B 67th Ave., Flushing, NY 11365

38
You will have to change two lines in the source code to adapt the program to your system. Line 0100 should signify your console backspace character and 0110 should contain the number of drives on your system. If you have a non-standard DOS, you will have to change the equates at the end of the program. If your system has only one drive, you may delete the section which asks for the drive number. Delete lines 150 to 290, change the label in line 300 to “START”, and change line 340 to “JC START”.

The routines in North Star DOS make this program very easy to implement. The simplest routine to use is the LIST routine that lists the disk directory. This routine is called with the drive number in the A-register and the output device number in the L-register (the latter is not used in RENAME). The DOS routines which really do the work of this program are DLOOK and DWRIT. To use DLOOK the Accumulator must again point to the RAM address containing the first letter of the file name being sought. The name in RAM should contain the drive number and the HL registers should contain the drive number in the A-register (although only names, types, and GO addresses may be changed without destroying the integrity of the directory). The DOS command DWRIT may then be used to write the changed directory in RAM back onto the disk. No disk activity should take place between using DLOOK and DWRIT and no parameters are needed when using DWRIT.

RENAME uses the above routines in the following manner. Lines 320, 330, 370, and 380 load the old and new file name into buffers called OBUF and NBUF respectively. Lines 430 through 490 then use DLOOK to make sure the new name is not already being used. In this case the program will continue if DLOOK returns with the carry flag set. Lines 530 to 560 use DLOOK to determine if the old name is on the disk. Hopefully the routine will return with the carry flag not set. If the old name is on the disk, then the HL register pair must be decreased by eight (lines 690 and 700) since the register pair is pointing to the end of the name. Lines 740 through 920 transfer the name in NBUF to the DOS directory in RAM and lines 960 and 970 use DWRIT to put the altered directory back on the disk.

Note that the hex code listing stops at 2E60. The code from this point on are the ASCII messages. The assembler prints these on multiple lines making the source code difficult to read. Therefore, only the source code was printed out from this point on.
Renaming Files cont'd...

2050  ; FIND OLD NAME IN DIRECTORY
2050  
2050  ; PUT DISK UNIT NUMBER IN ACCUMULATOR
2050  
2050  ; IL MUST POINT TO FILE NAME IN RAM
2050  
2050  ; CARRY IS SET IF FILE IS NOT FOUND
2050  
2050  ; MESSAGE THAT FILE WAS NOT FOUND
2050  
2050  ; AUTOMATICALLY LIST DIRECTORY IF NAME NOT FOUND.
2050  
2050  ; DELETE NEXT TWO INSTRUCTIONS IF FEATURE NOT DESIRED.
2050  
2050  ; LAST CHARACTER?
2050  
2050  ; DECREASE HL BY 8 SO THAT IT POINTS TO BEGINNING OF
2050  
2050  ; FILE ENTRY IN DOS RAM
2050  
2050  ; FOUND LXI D,-8
2050  
2050  ; MOVE NEW FILE NAME FROM NBUF TO DOS RAM ADDRESSED BY HL-REG
2050  
2050  ; MOVE TO BEGINNING OF NEW NAME BUFFER
2050  
2050  ; MOVE NEW NAME CHARACTER TO ACC
2050  
2050  ; MOVE NEW NAME TO DOS RAM
2050  
2050  ; LAST CHARACTER?
2050  
2050  ; FILE ENTRY
2050  
2050  ; NAME HAS AT LEAST ONE CHARACTER
2050  
2050  ; PUTS OLD AND NEW FILE NAMES INTO RESPECTIVE BUFFERS.
2050  
2050  ; PERFORMS VARIOUS TESTS TO MAKE SURE THAT:
2050  
2050  ; NAME IS NOT LONGER THAN EIGHT CHARACTERS
2050  
2050  ; NAME HAS AT LEAST ONE CHARACTER
2050  
2050  ; COUNTS OUT 8 CHARACTERS AND CR FOR NAMES
2050  
2050  ; CTRL-C RETURNS CONTROL TO N.S. DOS
2050  
2050  ; CTRL-X RESTARTS PROGRAM
2050  
2050  ; CTRL-D LISTS DIRECTORY
2050  
2050  ; IS CHARACTER A CARRIAGE RETURN?
2050  
2050  ; SKIP NEXT TWO INSTRUCTIONS IF IT IS.
2050  
2050  ; IS CHARACTER A SPACE OR LESS?
2050  
2050  ; SKIP NEXT INSTRUCTION IF IT IS.
Renaming Files cont’d...

2D02 EC 0F
2D04 FE 5F
2D0C CA 05 2E
2D0F 3E 0A
2DE1 B9
2DE2 CA AF 2D
2DE5 7A
2DE6
2DE7 77
2DE7 L7
2DE8 CD 19 2E
2DEB 0D
2DEC CA FD 2D
2DF0 FE 0D
2DF2 C2 B1 2D
2DF3 31 A2 2E
2DF8 CD A3 2D
2DFB 87
2DFC C9
2DFD
2DFD 21 8F 2E
2E00 CD A3 2D
2E03 37
2E06 C9
2E05
2E05 0C
2E06 3E 0B
2E08 B0
2E09 CA 15 2E
2E0C 2B
2E0D 06 06
2E0F CD 19 2E
2E12 C3 01 2D
2E15 0D
2E16 C3 B1 2D
2E19
2E19 F5
2E1A CD 0D 2D
2E1D F1
2E1E C9
2E1F
2E1F 33
2E20 33
2E21 FE 03
2E22 CA 2E 2E
2E26 FE 04
2E28 CA 37 2E
2E2B C5 06 2D
2E2C 23 A2 2E
2E31 CD A3 2D
2E34 C3 28 2D
2E37 21 A2 2E
2E3A CD A3 2D
2E40 CD 25 2D
2E43 C5 06 2D
2E46
2E46 20
2E47
2E47 33
2E50 77
2E50 L7
2E52 CD 19 2E
2E55
2E55 0C
2E56 3E 0B
2E58 B0
2E59 CA 15 2E
2E5C 2B
2E5D 06 06
2E5F CD 19 2E
2E62 C3 01 2D
2E65 CD A3 2D
2E68 C3 B1 2D
2E6B
2E6B F5
2E6C CD 0D 2D
2E6F F1
2E70 37
2E70 C9
2E73 33
2E74 33
2E75 FE 03
2E76 CA 2E 2E
2E7A FE 04
2E7C CA 37 2E
2E7F C5 06 2D
2E80 23 A2 2E
2E85 CD A3 2D
2E88 C3 28 2D
2E8B 21 A2 2E
2E8E CD A3 2D
2E94 CD 25 2D
2E97 C5 06 2D
2E9A
2E9A 20
2E9B
2E9B 33
2E9C
2E9C 33
2E9D FE 03
2E9E CA 2E 2E
2EA2 FE 04
2EA4 CA 37 2E
2EAB C5 06 2D
2EAD 23 A2 2E
2EAF CD A3 2D
2EB2 C3 28 2D
2EB5 21 A2 2E
2EB8 CD A3 2D
2EBD CD 25 2D
2EBE C5 06 2D
2EE0
2EE0 20
2EE1
2EE1 33
2EE2
2EE2 33
2EE3 FE 03
2EE4 CA 2E 2E
2EE8 FE 04
2EED CA 37 2E
2EF0 C5 06 2D
2EF2 23 A2 2E
2EF7 CD A3 2D
2EFA C3 28 2D
2EFB 21 A2 2E
2EFC CD A3 2D
2EFF CD 25 2D
2F00 C5 06 2D
2F03
2F03 20
2F04
2F04 33
2F05
2F05 33
2F06 FE 03
2F07 CA 2E 2E
2F0B FE 04
2F0C CA 37 2E
2F10 C5 06 2D
2F12 23 A2 2E
2F17 CD A3 2D
2F1A C3 28 2D
2F1B 21 A2 2E
2F1C CD A3 2D
2F1D CD 25 2D
2F1E C5 06 2D
2F20
2F20 20
2F21
2F21 33
2F22
2F22 33
2F23 FE 03
2F24 CA 2E 2E
2F28 FE 04
2F2C CA 37 2E
2F30 C5 06 2D
2F32 23 A2 2E
2F37 CD A3 2D
2F3A CD 25 2D
2F3B C5 06 2D
2F3C
2F3C 20
2F3D
2F3D 33
2F3E
2F3E 33
2F40 FE 03
2F41 CA 2E 2E
2F45 FE 04
2F49 CA 37 2E
2F4B C5 06 2D
2F4D 23 A2 2E
2F52 CD A3 2D
2F55 CD 25 2D
2F56 C5 06 2D...
Renaming Files cont'd...

2270  DW  0DDAH
2280  ASC  'CONTROL-C EXITS TO DOS  CONTROL-X RESTARTS PROGRAM'
2290  DW  0DDAH
2300  ASC  'CONTROL-D LISTS DIRECTORY'
2310  DW  0DDAH
2320  ASC  'RUBOUT OR DELETE WILL DELETE THE LAST CHARACTER TYPED'
2330  DW  0DDAH
2340  DW  0DDAH
2350  DB  0
2360  MESG0  DW  0DDAH
2370  ASC  'ENTER DRIVE NUMBER (CTRL-C TO RETURN TO DOS): '
2380  DB  0
2390  COUT  EQU  200DH  ;NORTH STAR COUT JMP
2400  INPUT  EQU  2010H  ;NORTH STAR CIN JMP
2410  DLOOK  EQU  201CH
2420  DPRINT  EQU  201FH
2430  DLST  EQU  2025H
2440  DOS  EQU  2028H
2450  RUBOT  EQU  5FH
2460  STOP  EOH

SYMBOL TABLE
BEGIN 2D00  BS  0008  COUT  200D  CR LF  2EA2  DELET  2E05
DIR  2E37  DLST  2025  DLOOK  201C  DOS  202B  DRIVE  2E58
DRNUM  0002  DWRITE  201F  ERRMSG  2E8F  ERROR  20FD  FOUND  2077
GO  2E2E  ITNAME  20AF  INPUT  2010  LEAVE  2E1F  LOKUP  205C
HEC  2A85  MESS0  2FC2  MESS1  2E59  MESS2  2E76  MESS3  2E4A
MESG  2EB9  NBUF  2E4F  NOTEL  205F  NOERROR  2077
OBUF  2E46  OLD  202D  OUTF  2E19  RUBOUT  005F  SKIPI  20CD
SK1P2  20D4  SKIPS  20E6  SPACE  2D90  START  2006  STOP  2FF3
TIME  2E6A  TRANS  2D86

CAT-100 FULL COLOR GRAPHICS
The original 256-color imaging system with high resolution video FRAME GRABBER for the S-100 bus.
Capture and digitize a video frame in 1/60 of a second. Select the best resolution for your application, from 256 to 1280 pixels per TV line. Display your digitized or computer processed image with 256 gray levels or 256 colors on standard B&W, NTSC or RGB color TV monitors.

Features:
- Highest possible quality 480x512x256 digital video image presently available on the market
- Input capability from TV camera or other sources
- Variety of synchronization choices
- 2 selectable video A/D conversion circuits
- Choice of 1, 2, 4, 8, 16 or 32 bits per pixel
- 32K-byte image memory on the basic system
- 32, 64, 128 & 256K byte system capacity
- Lightpen input
- Photographic trigger control input
- Software selectable system parameters
- Interfaces for TRS-80 and other processors
- Comprehensive line of accessories, monitors and support software

SEND FOR FREE CATALOG
DIGITAL GRAPHIC SYSTEMS
441 California Ave., Palo Alto, CA 94306  415/494-6088

Software Consultants
Harken!

Do you do software customizing? Do you do software package installation? Do you provide turnkey systems? Do you run a programming shop? Full-time? Part-time?
Starting with a future issue MICROSYSTEMS will publish a directory of "software shops." Places where computer customers can go for software package customizing. Most purchases of computer software packages find that the package does not exactly meet their needs—customizing is required—sometimes this is minor and sometimes it is major. Our directory will provide low cost advertising for "software shops." If you wish to be listed write to MICROSYSTEMS, Box 1192, Mountainside, NJ 07902 for information.
An Introduction to Microcomputers, Vol 1 - Basic Concepts
Adam Osborne. Thoroughly explains hardware and programming concepts common to all microprocessors: memory organization, instruction execution, interrupts, I/O, instruction sets and assembly programming. One of the best selling computer texts worldwide. 350 pp. $9.90 [9K].

Vol 2 - Some Real Microprocessors.
Vol 3 - Real Support Devices
Adam Osborne. These volumes complement Volume 1. Vol. 2 discusses the operation of each of the following MPlUs in detail: 80, SC/MP, 8080A, 6800, 68000, PPS-II, 2560, COS-MAC, 9002, 6100 and seven others. Also information on selecting a micro. Vol. 3 discusses various support and I/O chips. 895 pp. Vol. 2 [9L] $25.00, Vol. 3 [10Q] $20.00

Small Computer Systems Handbook
Sol Libes. The emphasis throughout this primer is on the important practical knowledge that the home computer user should have to be able to intelligently purchase, assemble, and interconnect components, and to program the microcomputer. Only a minimal knowledge of electronics is required to use this book. 196 pp. $8.45 [11D].

Accounts Payable and Accounts Receivable
Poole & Borchers. Includes program listings with remarks, descriptions, discussion of the principles behind each program, file layouts, and complete step-by-step instructions. Covers accounts payable and receivable in regard to invoice aging, general ledger, progress billing, partial invoice payments, and more. 375 pp. $20.00 [10V].

Microprocessors: From Chips to Systems
Rodnay Zaks. A complete and detailed introduction to microprocessors and microcomputer systems. Some of the topics presented are: a comparative evaluation of all major microprocessors, a journey inside a microprocessor chip, how to assemble a system, applications, interfacing (including the S-100 bus) and programming and system development. 416 pp. $9.95 [10S].

Programming in PASCAL
Peter Grogono. This book is an excellent introduction to one of the fastest growing programming languages today. The text is arranged as a tutorial containing both examples and exercises to increase reader proficiency in PASCAL. Contains sections on procedures, files, and dynamic data structures such as trees and linked lists. 359 pp. $11.50 [10A].

A Simplified Guide to Fortran Programming
Daniel McCracken. A thorough first text in Fortran. Covers all basic statements and quickly gets into case studies ranging from simple (printing columns) to challenging (craps game simulation). 278 pp. $13.50 [7F].

Some Common BASIC Programs
Poole & Borchers. This book combines a diversity of practical algorithms in one book: matrix multiplication, regression analysis, principal on a loan, integration by Simpson’s rule roots of equations, chi-square test, and many more. All the programs are written in a restricted BASIC suitable for most microcomputer BASIC packages and have been tested and debugged by the authors. $12.50 [7M].

The Best of BYTE
This blockbuster of a book contains the majority of material from the first 12 issues of BYTE magazine. The 146 pages devoted to hardware are crammed full of how-to articles on everything from TV displays to joysticks to cassette interfaces and computer kits. But hardware without software might as well be a boat anchor, so there are 125 pages of software and applications ranging from on-line debuggers to games to a complete small business accounting system. A section on theory examines the how and why behind the circuits and programs, and “opinion” looks at where this explosive new hobby is heading. $11.95. 386 pp. softbound from Creative Computing Press. [6F].

The Art of Computer Programming
Donald Knuth. The purpose of this series is to provide a unified, readable, and theoretically sound summary of the present knowledge concerning computer programming techniques, along with their historical development. For the sake of clarity, many carefully checked computer procedures are expressed both in formal and informal language. A classic series. Vol. 1: Fundamental Algorithms, 634 pp. $23.50 [7R]. Vol. 2: Seminumerical Algorithms, 624 pp. $23.50 [7S]. Vol. 3: Sorting and Searching, 722 pp. $23.50 [7T].

BASIC With Business Applications
Richard W. Lott. This book focuses on the BASIC language and its application to specific business problems. Part One introduces the BASIC language and the concept of logical flowcharting. Part Two presents problems and possible solutions. Topics include: interest rate calculations, break-even analysis, loan rates, and depreciation. This book is a great aid to the beginner wanting to learn BASIC without having a technical or scientific background. 284 pp. $11.95 [10Z].

To order: send check or credit card number and expiration date (Visa, Master Card or American Express) plus $2.00 per order for shipping and handling to: S-100 MICROSYSTEMS, P.O. Box 789-M, Morristown, NJ 07960.
The S.D. Systems' ExpandoRAM (the original 2 MHz version) is a quality product; the design exemplifies that dynamic RAM can be of real benefit to a computer system. This design draws less current than any conventional static RAM design. The ExpandoRAM is 64K dense, eliminating the redundant, power-hungry TTL buffering and decoding circuitry necessary in multi-board memory systems. The board runs at 2 MHz with no wait states. It is clear that a great deal of forethought was put into the design since 8080, 8085 and Z80 processor timing differences have all been accounted for. S.D. Systems did a fine job in producing a super reliable 2 MHz card!

However, S.D. Systems could have gone further; the design has the potential to run at 4 MHz, with an added wait state. The purpose of this article is to save replacement costs of new memory by allowing 4 MHz operation through a simple modification.

The first and obvious requirement is to add a wait state to memory references. Refer to Figure 1 for a suggested implementation of a single wait state circuit. With very little time and effort it can be wire-wrapped in a kluge area, if a wait state circuit does not already exist in the system.

The second requirement is that the system's Z80 processor card must generate RFSH (refresh, active low) onto bus pin 66 which is used by the RAM board for refresh. Assure that on the ExpandoRAM card, E11 to E12 is jumpered and E10 to E12 is open. The ExpandoRAM's refreshing is now under the complete control of the Z80's timing.

Before presenting the last requirement, some discussion of the board's only shortcoming is in order so that the modification will be better understood. Refer to the timing diagram in Figure 2 of this article, and to the schematic in the ExpandoRAM manual.

The problem is the generation of memory REFRESH CYCLES which last much too long and run into the next M-cycle timing, completely obliterating it! As a fix was being sought, it was interesting to note how close to 4 MHz the board can work. While running a memory test there really were very few errors, and the addresses at which errors did occur were randomly located. This is a classic symptom of refresh interference!

The timing diagram of Figure 2 has one wait state in it, and is of an M1 cycle. Its inherent transparent REFRESH CYCLE occurs during the T4 state. Notice that the M1 and RFSH bus signals are for all intents and purposes the same signal, only the definitions of their active states are complemented. On the RAM board schematic observe that the RFSH signal eventually triggers one-shot U5-10, and a REFRESH CYCLE begins. U5-5 goes high, and U5-12 goes low throughout the REFRESH CYCLE, resulting in four occurrences necessary to successfully refresh the RAMs.

First, U5-12 going low blocks the TIME-OF-CAS signal at U13-1 from being seen at U13-3 during a refresh. This permits the so-called RAS only REFRESH CYCLE, which is desirable.

Secondly, the active low at U5-12 also causes U8-8, 6, 11 and 3 to all go high enabling RAS drivers outputs at U3-8, 6, 11 and 3 to go low at TIME-OF-RAS. This produces the refresh RAS at all banks simultaneously, giving the RAM array its breath of fresh air.

Thirdly, U5-5 enables the refresh addresses of U11 and U15 to the RAM array by raising Select (pin 1) of the U21 and U16 multiplexers high.

Finally, after a time delay through R1 and C6 at U9-1, and onto U10-5, the active low leading edge at U10-6, defined as MEMORY CYCLE, triggers U5-1. Firing this one shot produces the TIME-OF-RAS signal, and a REFRESH CYCLE starts.

The timing diagram clearly differentiates between the M1 opcode fetch and the refresh portion. Note how long the REFRESH CYCLE waveform at U5-5 lasts past...
Tecmar's new A/D and Timer Board is designed to meet sophisticated data acquisition needs. The board can accommodate various A/D modules providing options such as 12, 14, 16 bit accuracy; 100 MHz throughput; variable ranges and gains. It contains a powerful timer circuit (AMD 9513) which can start A/D conversion and can also be used independently for time of day, event counting, frequency shift keying and many other applications.

TM-AD200 FEATURES

- Complies with IEEE S-100 specifications
- Transfers data in 8 or 16 bit words
- 30 KHz throughput standard
- 12 bit accuracy standard
- Jumper-selectable for 16 single-ended or 8 true differential channels
- External trigger of A/D
- Provision for synchronizing A/Ds
- Data overrun detection
- Data is latched providing pipelining for higher throughput
- Input ranges: ±10V, ±5V, 0 to +10V, 0 to +5V
- Output formats: Two’s complement, binary, offset binary
- Auto channel incrementing

- I/O or memory mapped
- Utilizes vectored interrupt or status test of A/D
- Provision for expansion to 256 channels

TIMER FEATURES

- 5 independent 16 bit counters (cascadable)
- 15 lines available for external use
- Time of day
- Event counter
- Alarm comparators on 2 counters
- One shot or continuous frequency outputs
- Complex duty cycle and frequency shift keying outputs
- Programmable gating and count source selection
- Utilizes vectored interrupt

TM-AD200 OPTIONS

- Programmable gain up to 500
- 14 bit accuracy
- 16 bit accuracy
- Screw terminal and signal conditioning panel with optional thermocouple cold junction compensation

- 100 KHz throughput with 12 bit accuracy
- Low level, wide range (10mV to 10V FSR) permitting low level sensors such as thermocouples, pressure sensors and strain gauges to be directly connected to the module input

TECMAR, INC.
(216) 382-7599
23414 Greenlawn • Cleveland, OH 44122

S-100 A/D & TIMER

If your data acquisition needs are simple, the original Tecmar S-100 A/D Board will meet your needs.

TM-AD100 FEATURES

- Complies with IEEE S-100 specifications
- 16 single-ended or 8 true differential channels
- 12 bit accuracy
- 25 KHz throughput
- I/O or memory mapped
- Input ranges: ±10V, ±5V, 0 to +10V, 0 to +5V
- Minimal software required.

S-100 BOARDS

<table>
<thead>
<tr>
<th>Component</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>8086 CPU</td>
<td>$450</td>
</tr>
<tr>
<td>W/ vectored interrupts</td>
<td>$395</td>
</tr>
<tr>
<td>RAM 8Kx16/16Kx8</td>
<td>$495</td>
</tr>
<tr>
<td>8086 PROM/M/O</td>
<td>$495</td>
</tr>
<tr>
<td>Serial &amp; Parallel I/O</td>
<td>$350</td>
</tr>
<tr>
<td>Parallel I/O</td>
<td>$350</td>
</tr>
<tr>
<td>&amp; Timer</td>
<td></td>
</tr>
</tbody>
</table>

TR-501 PET2 KIM2 (available soon)

A/D

- 12 Bit
- High Speed
- 8 Ch. Differential
- 16 Ch Single-ended
- Each A/D Module $495

D/A

- 12 Bit High Speed
- 4 Channel

Each D/A Module $395

TRS-80 or PET expansion board, power supply, and enclosure $200.
Kim expansion board and power supply $150.

For digital to analog conversion, Tecmar's D/A Board provides four independent 12 bit high speed D/A channels.

TM-DA100 FEATURES

- Complies with IEEE S-100 specifications
- 4 independent digital to analog converters
- 12 bit accuracy
- 3 µsec settling time
- I/O or memory mapped
- Output ranges: ±2.5V, ±5V, ±10V, 0 to +5V, 0 to +10V

S-100 Real Time Video Digitizer

- Digitizes and Displays in 1/60 sec, flicker-free
- 16 Gray Levels
- Switch Selectable to display Black and White Graphics (8 pixels/byte)
- Maximum Resolution: 512 pixels/line x 240 lines
- Minimal software requirements $850

Data Acquisition Systems and Video Microcomputer Systems Available
Waveforms Below Relate Timing Before Modification:

Refesh Cycle, U5-5

Time-of-RAS, U3-2

Time-of-CAS, U13-1

Memory Cycle, U18-6

Waveforms Below Relate Timing Changes After Modification:

Refresh Cycle, U5-5

Memory Cycle, U18-6

Time-of-RAS, U3-2

NOTE: 1) Times are in nanoseconds
2) Times relative to rising edge of M1.
3) Dotted lines represent 'ghosting', which are expected.
4) A wait state is pulled in this timing diagram.

Figure 2
Timing Diagram—SDS ExpandoRAM (Driven by 4 MHz Intersystems Z-80 CPU Processor Board)

Figure 3
Modification Schematic—SDS ExpandoRAM

S-100 MICROSYSTEMS
modification clears the one shot efficiently and asynchronously.

The 2 MHz ExpandoRAM should now function at 4 MHz with its required wait state without sacrificing system throughput too drastically. (The particular board used to find this fix is populated with 250 nanosecond RAMs, and the only wait state needed is during the shorter accessing of the M1 cycle. So this system is hardly being held back at all!)

So go ahead and spend that memory replacement money on the new peripheral you’ve wanted for so long. There’s still plenty of life in the 2 MHz ExpandoRAM with this new 4 MHz system!

WRITE FOR S-100 MICROSYSTEMS

S-100 MICROSYSTEMS is always looking for quality articles on software, hardware and product reviews related to S-100 systems, CP/M, Pascal, etc. To receive an author’s guide, send a stamped self-addressed envelope to: S-100 MICROSYSTEMS, Box 789-M, Morristown, NJ 07960. Payment for published articles is based on the printed page length.

CATCH THE S-100 INC. BUS!

Integral Data—"Paper Tiger"
440-G U/L Case w/Graphics
S.D. Systems PROM 100 Programmer Kit
Godbout 3P+S Interfacer II "Unkit"
North Star Z-80A 4MHz CPU A&T
Morrow Design Discus II D Dual Density 8" Disk System w/CP/M°

*Included free with every S.D. Systems Board is an additional $25.00 manufacturer’s rebate coupon.
°CP/M is a trademark of Digital Research
Subject to Available Quantities • Prices quoted include cash discounts. Shipping & Insurance extra.
We carry all major lines such as S.D. Systems, Cremenco, Ithaca Intersystems, North Star, Sanyo, ECT, TEL, Godbout, Thinker Toys, Hazeltine, IMC. For a special cash price, telephone us.

Bus ......... S-100, Inc.
Address ....... 7 White Place
Clark, N.J. 07066
Interface ....... 201-382-1318

S-100 MICROSYSTEMS
How fast is Pascal? Current advertisements for UCSD Pascal state “A UCSD Pascal program executes up to ten times faster than a comparable BASIC program”. Since I recently acquired UCSD Pascal this statement led me to conduct a speed comparison between Pascal and the other programming languages available to me.

The four languages used for the comparison were Microsoft Extended Basic, version 4.51; CBasic-2, compiler version 2.01, interpreter version 2.03; Microsoft Fortran, version 3.2; and UCSD Pascal, version 11.0. With the exception of Fortran, these languages are all interpreters, operating on an intermediate code which is either generated directly as the program is entered (for Microsoft Basic), or generated by a separate compilation process. Fortran operates as a true compiler generating native machine code.

Four programs were used for the test. These programs are presented in Table 1. The first program is a simple do-nothing loop; the second includes addition, multiplication, and division within the loop; and the third adds a subroutine call within the loop. The fourth program adds the process of raising a number to a power. Since my UCSD Pascal manual states that there is no explicit provision for this process, it is handled as a programmed function for all four languages.

The results of the test are shown in Table 2. All programs were run on my Altair 8800 system, which uses an 8080 microprocessor with a 2 Mhz clock. The execution times were measured between the printout of the “start” and “done” messages within the programs, using the seconds display of a digital watch. Each program was run three times, with the average time presented in the table, to remove potential timing errors.

I am not going to attempt to interpret the results of this test, but the following observations do seem appropriate. First, I have only had access to Pascal for a few weeks and am not yet very familiar with its operation. Therefore, I chose simple programs (and program structures) which could easily be programmed similarly in the four languages. Second, these tests primarily compare the floating point arithmetic capabilities of the four languages, rather than any other of the capabilities or features they may possess. Since this is a common feature of many of the applications I am involved in, it is an area of primary interest to me. The poor showing of CBasic in this application can be partially attributed to the fact that it carries 14 digit precision for all real variable operations. To achieve this same precision with the other languages would require double precision variables, which would undoubtedly slow them down.

A more thorough timing comparison of Pascal with other languages would be of interest to me. However, this project will have to wait either until I become more familiar with Pascal (and have time to do the comparisons), or until someone else undertakes such a project.
Program A

100 DEFINT K
200 PRINT "START"
300 FOR K = 1 TO 10000
500 A = X / 7.5 + X * 6.2
600 NEXT
700 PRINT "DONE"
800 STOP
900 END

Program B

100 DEFINT X
200 X = 33.24
300 PRINT "START"
400 FOR K = 1 TO 1000
500 A = X / 7.5 + X * 6.2
600 NEXT
700 PRINT "DONE"
800 STOP
900 END

Program C

100 DEFINT X
200 X = 33.24
300 PRINT "START"
400 FOR K = 1 TO 1000
500 A = X / 7.5 + X * 6.2
600 NEXT
700 PRINT "DONE"
800 STOP
900 END

Program D

100 DEF FN(E, X) = EXP(X * LOG(E))
200 X = 1
300 Y = 1.57
400 PRINT "START"
500 FOR K = 1 TO 1000
600 Z = FN(E, X, Y)
700 X = X + 1.0
800 NEXT
900 PRINT "DONE"
1000 STOP
1200 END

Table 1. Programs Used for Comparison
The ADS Noisemaker Board

by Steve Levine

PART-1 Building and testing this dual programmable sound generator kit.

The Ackerman Digital Systems NOISMAKER is an S-100 standard board incorporating 2 General Instruments AY3-8910 integrated circuits. The 8910 is a large scale integrated circuit which can produce a wide variety of complex sounds under software control. Once the internal registers have been programmed, sound will continue to be produced, allowing the processor's time to be used for other tasks.

KIT DESCRIPTION

Construction of this kit was quite simple. It should pose no problem for the novice or experienced hobbyist. Adequate documentation was included with the board consisting of a brief overview of the 8910 chip, clear assembly instructions and a good parts list. Using only the schematic diagram and parts list supplied, I had it going in a little over an hour. The circuit board is good G-10 solder-masked plated through construction.

Included on the board is a 1 watt audio amp, which can be connected directly to a speaker for output. The edge fingers for both the bus and interface connectors are gold plated. A very nice feature of the board is the plated through 'kluge' area provided on the right hand side. I found this particularly handy for adding a second audio amplifier chip or anything my imagination could muster.

Provisions are made for the 2 Mhz system clock to be used as the time base for the synthesizer. There is also a custom clock prototype area with enough room for the 3.57945 mhz crystal clock circuit that G.I. calls for in the data sheet. ( This frequency facilitates using an equally tempered scale [ETS] set of numbers for the tone generators. ) This area is supposed to be designated for adding a wait state generator, but will serve in the above capacity just fine.

Note: The manufacturers of the board advise using wait states with a 4 Mhz system ( this is an inherent limitation of the AY3-8910 IC).

Another feature of the PSG is the on board I/O ports. ADS brings three of the four 8 bit ports out to the edge connector on top of the card. The fourth one is routed to the kluge area. In computer music experimentation you will undoubtedly find a use for these lines, say for scanning a keyboard to facilitate an input device.

OPERATION

The board is addressed in the I/O map of a standard S-100 system. It occupies 4 concurrent I/O addresses. Using the dip switch -- select the starting block of four I/O addresses you want the board to appear in. I chose 0 Hex for simplicity.

<table>
<thead>
<tr>
<th>Address bit</th>
<th>0</th>
<th>1</th>
<th>Looking at:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALADRA</td>
<td>0</td>
<td>0</td>
<td>Latch address for psg 'a'</td>
</tr>
<tr>
<td>ADATA</td>
<td>0</td>
<td>1</td>
<td>Register data for 'a'</td>
</tr>
<tr>
<td>BLADRA</td>
<td>1</td>
<td>0</td>
<td>Latch address for 'b'</td>
</tr>
<tr>
<td>BDATA</td>
<td>1</td>
<td>1</td>
<td>Register data for 'b'</td>
</tr>
</tbody>
</table>

By writing the desired register number to the ALADRA or BLADRA I/O port on the noismaker board, and then writing the value to the ADATA OR BDATA I/O port, simple tests can be performed on all the different registers.

The ADS NOISEMAKER takes very good advantage of the AY3-8910 for the S-100 bus. With an assembly language driver and a good high level language such as PASCAL or FORTH, one can easily implement a tiny music composition and performance system. The 8910 is not a complicated device to use, but a good understanding of the PROGRAMMABLE SOUND GENERATOR will of course be necessary. Refer to architecture chart and diagram for the following discussion:

The AY3-8910 Programmable Sound Generator is a 3 channel digitally programmable synthesizer in one IC. There are 16 internal registers which are individually addressed , into which tone, envelope and control information are loaded prior to producing a sound. Included on chip
are register decoding and data bus buffering circuitry. Upon a RESET (negative) all registers are reset to zero.

Each chip contains the following:

* 1 I/O enable, mixer control register
* 3 - 12 bit tone generators
* 3 - 4 bit amplitude control registers
* 1 - 5 bit noise generator
* 1 - 16 bit envelope generator
* 1 - 4 bit envelope control register
* 2 - 8 bit parallel I/O ports

Individual registers can be set to specific values for producing sound. Refer to architecture chart. There are 3 individual 12-bit tone generators. When a 12-bit number is loaded into a set of tone period registers, the value is then latched. The clock input is divided by 16 and again divided down by a decimal value equal to the tone period registers. When the counter reaches zero, the register containing the original number is reloaded into the counter and the process is repeated. The result is a continuous square wave tone, whose period is a function given by the following equation:

\[
\text{TONE FREQUENCY } f_t = \left( \frac{\text{clock}}{16 \times \text{TP}} \right)
\]

where:
- \( \text{clock} \) = frequency of clock to AY3-8910
- \( \text{TP} \) = 256 * coarse tone + fine tone
- coarse tone = Coarse tone register value
- fine tone = Fine tone register value

(Numbers are assumed decimal)

Register 7 is used to control the selection (mixer) of either tone or noise or both, for a particular channel. Registers 8, 9 and 10 are amplitude control registers. By setting the mode bit low, the processor can directly control amplitude of the 8910 have greatly simplified this with the envelope generator. Referring to the envelope shape diagram, HOLD, ALTERNATE, ATTACK, and CONTINUOUS bits control envelope generation in conjunction with the envelope period registers. This particular function could be discussed in endless detail, but the manufacturers of the 8910 have greatly simplified this with the envelope shape diagram. Experimentation with this module is probably the best way to fully understand its capabilities.

In a future article, I will discuss software and hardware for a complete music system, which is written in FORTH. A musical keyboard with polyphonic and velocity sensitive capabilities will be used as the musical input device.

I consider the ADS NOISEMAKER an ideal and inexpensive way for an S-100 system owner to experiment with computer music.

I welcome correspondence related to this article and computer music generally.

Using the above formulas, and a 2 Mhz clock, we can obtain a usable range of tone frequencies from 30.5 hz to 125 KHz. With the noise generator period value we obtain a noise range of 4 KHz to 125 KHz. The envelope generator has more sophisticated control features of the PSG. Envelope period can be calculated to 16 bits resolution. A usable range for the envelope period would be 0.12 hz to 7812.5 hz. This period will control the frequency of the envelope generator. Envelope shape/cycle (4 bits) provides a function control for the envelope generator. Referring to the envelope shape diagram, HELD, ALTERNATE, ATTACK, and CONTINUOUS bits control envelope generation in conjunction with the envelope period registers. This particular function could be discussed in endless detail, but the manufacturers of the 8910 have greatly simplified this with the envelope shape diagram. Experimentation with this module is probably the best way to fully understand its capabilities.

In a future article, I will discuss software and hardware for a complete music system, which is written in FORTH. A musical keyboard with polyphonic and velocity sensitive capabilities will be used as the musical input device.

I consider the ADS NOISEMAKER an ideal and inexpensive way for an S-100 system owner to experiment with computer music.

I welcome correspondence related to this article and computer music generally.
Evaluating the New 8088 Microprocessor

This article compares the new Intel 8088 8-bit microprocessor to the Motorola 6809 and several other processors.

This report was furnished by a source within Intel who requested that his name not be disclosed. He did however give permission to reprint the report. Some editing was done on the report to improve readability.

8088 ASSEMBLY LANGUAGE BENCHMARKS

Following is a summary of 8088 implementations of the four benchmarks analyzed in the article entitled 'Assembly Language Benchmarks' written by A. Flippin, which appeared in the March 1980 issue of KILOBAUD magazine. The reader should refer to the actual article for a detailed explanation of each benchmark and the criteria used for evaluation.

The 8088 is compared against the following machines (listed in order of overall performance ranking):

- 370-145
- LSI-11
- 9900
- Z80
- 6502
- 6800
- 8080
- 6100
- 1802
- SC/MP

The new ranking now is:

- 8088
- 370-145
- LSI-11
- 9900
- Z80
- 6502
- 6800
- 8080
- 6100
- 1802
- SC/MP

All timings are calculated for the standard speed (5mhz) 8088. The 8mhz version, expected about the middle of '81, will provide even better performance!

Intel claims therefore that "The 8088 is clearly the highest performance 8 bit microprocessor in the world!"

---

Figure 1

Averages by Benchmark

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Table Lookup</th>
<th>Block Move</th>
<th>Jump Table</th>
<th>Multiply</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of instructions</td>
<td>rank - #1 (tie 370, LSI-11, 9900)</td>
<td>rank - #2</td>
<td>rank - #1 (tie 370, LSI-11, 9900)</td>
<td>rank - #2 (tie 370, LSI-11, 9900)</td>
</tr>
<tr>
<td>Memory utilized (bytes)</td>
<td>rank - #2</td>
<td>rank - #2 (tie 8080)</td>
<td>rank - #1 (tie LSI-11)</td>
<td>rank - #1 (tie LSI-11)</td>
</tr>
<tr>
<td>Execution speed (μS)</td>
<td>rank - #1</td>
<td>rank - #2</td>
<td>rank - #2</td>
<td>rank - #2 (tie 370, LSI-11, 9900)</td>
</tr>
</tbody>
</table>

Figure 2

Averages by Category

<table>
<thead>
<tr>
<th>Execution Time</th>
<th>Ease of Programming</th>
<th>Memory Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>rank - #1 .47</td>
<td>rank - #1 .490</td>
<td>rank - #1 .64</td>
</tr>
<tr>
<td>Overall</td>
<td>rank - #1 .53</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3

8088 Versus 6809

In an effort to determine which microprocessor actually delivers the most performance, the Intel 8088 was compared to the Motorola MC6809. Nine programs

S-100 MICROSYSTEMS
were written, and the two processors were compared on the basis of execution speed, ease of programming (number of lines of code), and memory efficiency. The results in the table below show that the 8088 significantly outperformed the MC6809 in all three categories.

<table>
<thead>
<tr>
<th>PROGRAM</th>
<th>SPEED</th>
<th>LINES</th>
<th>BYTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Graphics</td>
<td>23.44</td>
<td>5.80</td>
<td>4.50</td>
</tr>
<tr>
<td>16-Bit Multiply</td>
<td>2.01</td>
<td>7.00</td>
<td>4.00</td>
</tr>
<tr>
<td>Vector Add</td>
<td>1.10</td>
<td>1.00</td>
<td>1.17</td>
</tr>
<tr>
<td>Reentrant Call</td>
<td>.87</td>
<td>.88</td>
<td>1.02</td>
</tr>
<tr>
<td>Computed Jump</td>
<td>.64</td>
<td>1.17</td>
<td>.93</td>
</tr>
<tr>
<td>Block Move</td>
<td>2.06</td>
<td>2.00</td>
<td>1.73</td>
</tr>
<tr>
<td>Block Translate</td>
<td>1.78</td>
<td>1.30</td>
<td>1.54</td>
</tr>
<tr>
<td>Character Search</td>
<td>2.12</td>
<td>1.13</td>
<td>1.06</td>
</tr>
<tr>
<td>Word Shift</td>
<td>3.09</td>
<td>2.25</td>
<td>1.80</td>
</tr>
<tr>
<td>Normalized Average</td>
<td>4.12</td>
<td>2.50</td>
<td>1.97</td>
</tr>
<tr>
<td>Adjusted Average</td>
<td>1.86</td>
<td>2.09</td>
<td>1.76</td>
</tr>
</tbody>
</table>

The normalized averages show that the 8088 was more than four times faster than the MC6809, required 2.5 times less lines of code, and was nearly 2 times as efficient in memory usage. The normalized averages were greatly influenced by the Computer Graphics benchmark which requires many 16-bit multiplies and divides. For this reason an adjusted average was computed ignoring the highest and lowest value in each category.

A brief description of the benchmark programs follows:

**COMPUTER GRAPHICS**...Scales the X and Y values of an array for a graphics display.

**16-BIT MULTIPLY**......Multiplies two unsigned 16-bit numbers to produce a 32-bit product.

**VECTOR ADD**...........Adds the elements of two 20 element vectors and stores the resultant vector in memory.

**REENTRANT CALL**......Calls a subroutine, saves all registers, adds parameters passed to the subroutine and returns after restoring all registers.

**COMPUTED JUMP**.......Uses a control byte and a vector table to compute a jump address.

**BLOCK MOVE**...........Moves a block of data in memory to another location in memory.

**BLOCK TRANSLATE**.....Translates a block of EBCDIC characters to ASCII.

**CHARACTER SEARCH**....Searches a block of memory for a character.

**WORD SHIFT**...........Shifts a 16-bit word 5 places to the right.

These benchmark programs demonstrate the ability of the processors to perform many of the tasks typically required of a high performance microprocessor. The 8088 performs so well here due to its powerful instruction set which includes 8- and 16-bit signed and unsigned multiplies and divides, and string processing instructions with optional repeat prefixes. The register set of the 8088 is also extremely versatile with four 16-bit accumulators which can be used as eight 8-bit registers, four pointer registers, and four segment registers. The MC6809 has only an unsigned 8-bit multiply and no divide. It has a 16-bit register which can be used as two 8-bit accumulators or one 16-bit accumulator. The MC6809 also has four pointer registers.

The Z80 was also looked at, and was out performed by more than three to one by the 8088 in the adjusted average for execution speed.

These results clearly demonstrate that the Intel 8088 outperforms the Motorola MC6809 in the categories of speed, ease of programming, and memory efficiency. The 8088 also provides a clear upgrade path to 16-bits with complete software compatibility with Intel's 16-bit 8086.

---

**At last—NETWORKING!**

The CDL Network Operating System in your computers will give you:

**SAVINGS** One disk and printer can be shared among any number of satellite computers. Hard and floppy disk versions are available.

**POWER** All satellites have a full CPM environment. You can access the largest collection of software available for microcomputers.

**FLEXIBILITY** The network software can be adapted to your input/output hardware. It even adapts automatically to available RAM.

The CDL Networking Operating System is a quality software product with excellent documentation backed by extensive testing.

Write for information.

CAMBRIDGE DEVELOPMENT LAB.
36 Pleasant St., Watertown, MA 02172

---

**Attention S-100 Board Suppliers**

We are currently attempting to compile a listing of all S-100 board and mainframe suppliers and their products. If you are a manufacturer of S-100 products please send us a complete set of specification sheets on your products. We hope to publish this listing in the NOV/DEC issue of S-100 MICROSYSTEMS.
Relocatable Code

by Richard H. Mossip

WHAT IS IT?... Relocatable code is a program which can be run correctly when located at any memory address. In some computers it can be written directly, as all memory accesses are processed through an indexing scheme where a pre-defined constant is added to each memory address to obtain the address used in real memory. This is a hardware approach used on many large machines where the actual address of any program will be different each time it is run, depending on what is in the machine with it in a multi-programming setup. All addresses in a program are logical, and are mapped by hardware and the operating system into physical addresses used at the time. It is much more difficult in micros, particularly the 8080 & 8085, as they lack an indexed addressing mode (which the Z-80 has but is rarely used).

HOW DO YOU DO IT?... In the 8080, and other simple processors which have only direct addressing (at a specified 16 bit address) or register indirect addressing (at the content of a specified register-e.g. LDAX B), when a program requires to run at a different location all 2-byte addresses located within the code module (particular program segment to be relocated) have to be changed to reflect the new location, so that all jump and call statements have the correct destination, and all addresses for data are correct. There are two approaches generally used for this. The first is usually used for "Page Boundary relocation", where a program can be relocated in 256 byte increments, and only the upper byte of addresses is changed. Conceptually the same approach could be used for full relocation, but this is rarely done as it is more complex. This can be performed with a standard assembler by making two assemblies; the first ORGed at 0000 and the second at 100 Hex. These two are then compared, and a list is made of all locations where a difference exists. These are the locations which have to be patched with the correct page location where the program will reside. The page boundary at which the program starts is added to the contents of each location in the list, and the program will now run without any special code.

The second method uses a special assembler called a relocating assembler. Examples of this are the TDL relocating Macro assembler, and the Microsoft ASM80. In these the object code file produced is different from that produced by a conventional assembler in that all addresses generated by a label within the program are flagged for relocation, while all addresses given as constants are assumed to be fixed. The program is loaded into working memory by a special "linking loader" which adds the appropriate constant to the flagged addresses (which reference the beginning of the program as their base) and produces properly running code which can start at any address in memory.

WHY BOTHER?... Two reasons come to mind which make this useful in a microprocessor situation. The first occurs in CP/M, where the operating system resides at the top of memory in use, and is different for different system sizes. System sizes usually only differ in 1K or more increments, so page boundary relocation is adequate, and Digital Research only wishes to distribute object code, thus avoiding the simple approach of re-assembling with a new origin. There is a program called MOVCPM distributed with CP/M which contains a table of all locations requiring change, and accepts console input of the new system size. This builds in memory a copy of the code with the new addresses. It is then written out to disk as the new system, after having the user-provided parts of the program suitably re-assembled and grafted onto it with the debugger. This is a process which requires a certain amount of operator attention and knowledge, but works well in the application where it seldom needs to be done.

Where a program must be run in minimum memory, as in special purpose industrial applications, it is very convenient to write the program in modules which can be tested and debugged separately. Once a module is debugged, one does not want to compile or assemble it again, to avoid introducing errors in patching several short segments together to make a long program. Relocatable code will do this for you. It also enables you to write code in Basic or Fortran, debug it and then re-code it a module at a time in assembly.
language, testing as you go—just linking them together as they are written. If the length of one module is changed, the required changes to the others are made entirely automatically at load time. This is the approach used by the Microsoft ASM80, which produces code directly compatible for linking with Basic Fortran or Cobol programs. The format is totally different from that used by TDL (a modification of the Intel checksum HEX loader) as the code is only stored on disk as a bitstream (not located on BYTE boundaries) to conserve disk space. This is at least a 2:1 compression, and probably better. Details are given in the manual (barely) and are of little interest to others.

GILDING THE LILY... This was clearly too simple, so the designers rushed out to make it more powerful (e.g. complicated). The concepts of INTERNAL, EXTERNAL, and ENTRY labels are needed, if programs are to be linked together. An address flagged to the assembler as EXTERNAL means that it is defined in another module (and must NOT be redefined in this one). An INTERNAL address label is flagged as available for access by other external programs. An ENTRY label is an internal label at which the program can be entered, rather than referenced. This is used when modules are placed in a library, and is a name for which the library can be searched in the TDL system. All other labels are for internal consumption only and can be duplicated in other program segments without fear of confusion.

More details will be given in a later article as I become more familiar with using the system.

---

S-100 4 Channel Serial Interface

Economical interface flexibility for the advanced amateur or small business computerist

- Industry standard LSI UARTS
- 8 reversible status and data ports
- Optically isolated current loop operation
- Independent channel operation - RS-232, 20mA or 60mA current loop
- On-board crystal time base
- One year defect and workmanship warranty

$249.95 Plus $1.50 postage and handling in the U.S.

---


Power Line Spikes, Surges & Hash could be the culprit! Floppies, printers, memory & processor often interact! Our unique ISOLATORS eliminate equipment interaction AND curb damaging Power Line Spikes, Surges and Hash.

ALL ISOLATORS: 125 VAC, Standard 3-prong plug
- 1875 W Max Load - 1 KW/Socket or socket bank
- Balanced Pi Filtered sockets or socket banks
- Spike/Surge Suppression - 1000 Amps, 8/20 usec
(SUPER ISOLATORS offer expanded filtering and Spike/Surge Suppression capabilities)

ISO-1
ISO-2
ISO-3 -3 individually filtered sockets $56.95
ISO-4 -6 individually filtered sockets $96.95
ISO-2 -2 filtered banks; 6 sockets $56.95
ISO-5 -3 filtered banks; 9 sockets $79.95

ISO-6
ISO-7

SWITCHABLE ISOLATORS - ALL ISOLATOR advantages combined with the versatility, convenience and utility of individually switched sockets. Each switch has associated pilot lite.

ISO-6 -3 switched, filtered sockets $128.95
ISO-8 -5 switched, filtered sockets $161.95

SUPER ISOLATORS - Cure for severe interference problems. Useful for industrial applications and heavy duty controlled equipment or peripherals.

ISO-3 -3 super filtered sockets $85.95
ISO-7 -5 super-filtered sockets $139.95

CIRCUIT BREAKER any model (add-CB) ADD 7.00
CKT BKR/SWI TCH/PILOT any model (CBS) ADD 14.00

PHONE ORDERS 1-617-655-1532

Electronic Specialists, Inc.
171 South Main Street, Natick, Mass. 01760

DEPT 596
NEW PRODUCTS

EXPANDORAM MOD FOR 4MHz OPERATION

J. E. S. Graphics, Box 2752, Tulsa OK, 74101 (tel: 918-742-7104) is providing a modification kit (4 parts) which when installed in an ExpandoRam Memory Board enables it to operate at 4MHz, with no wait states. Cost is $10.00.

DESKTOP PASCAL GRAPHICS COMPUTER SYSTEM

The demand for lower-cost graphics capabilities has resulted in a new product from Integrated Research and Information Systems (IRISystems) Corporation, the ENSEMBLE 120GX. The new product is a self-contained computer system, packaged in a configuration that might easily be mistaken for a terminal. Standard versions are based upon the Western Digital Pascal MicroEngine, high resolution graphics, a 15" monitor, 12-slot S-100 standard motherboard, detachable keyboard, double density, double-sided dual floppy disk subsystem, 280 alternate on-board MPU, memory parity, 128KB dynamic RAM, UCSD Pascal, CP/M, constant voltage transformer, and printer port for graphics hard copy output. The ENSEMBLE 120GX can present a 768 X 480 pixel format in 8 X 16 pixel characters; some of these can be user-defined. Single quantity price is $9796.

The ENSEMBLE 120GX in its standard configuration runs under UCSD Pascal and CP/M. Alternate configurations run only under CP/M, do not use the MicroEngine, but offer a Pascal compiler which generates floating point instructions for the AMD9511 chip. A low-cost version of the ENSEMBLE 120GX uses the 280 MPU, a 9" black and white monitor, and one 5.25" drives. It costs $3464 in single quantity.


S-100 8088 CPU BOARD ANNOUNCED

ACOM Electronics has released its model P188 8088-based S-100 CPU board. The P-188 will run in either of three modes: 1) As a stand-alone processor on S-100 bus, 2) As a slave processor with other processor cards, and 3) with one or more additional processors in a shared processor environment. The P188 emulates all necessary S100 Bus signals. Numerous jumpers allow configuring the card to run different operating modes, as well as static and dynamic memory.

The 8088 processor is an 8086 with an 8 bit data bus. It has 16 bit internal architecture, addresses 1 million bytes of memory, and 8-bit/16-bit signed and unsigned arithmetic in binary or decimal, including multiply and divide. The introductory price is $385. Tested & Assembled. For more information contact: Sam R. Hawes, jr, ACOM Electronics, 4151 Middlefield Rd, Palo Alto, CA 94303; Tel: (415) 494-7499.

8080/8085 PLASTIC ACTION SHEET

Micro Chart is a multiple color plastic sheet packed with instant access technical information. #100A covers the 8080 / 8085 microprocessors and is the first in a series. Made from credit card type plastic, it measures 8½" by 11" and is puched for optional notebook use. Primarily software oriented, it also includes critical hardware data. It is organized to save time for professionals, hobbyist, and students.

Micro Chart is available at many dealers for only $2.95. Dealer inquiries invited. For mail orders to Micro Logic add $1.00 for P&H. Minimum order to Micro Logic is 5 days ARO. Micro Logic Corp. Dept. CSL POB 174, Hackensack, NJ 07602; (201) 342-6518.

VERSATILE CPU AND I/O CARD WITH DISK CONTROLLER

The CP/10-1, from Arkon Electronics, provides a single S-100 card, all of the CPU and I/O facilities required to construct a disk-based microcomputer system. This 8080A based card features fully vectored interrupts, 5 programmable interval timers, 24 parallel I/O lines, RS-232 serial terminal port at 110 baud to 76 kilobaud, and an RS-232 pseudo-synchronous port suitable for printer interface. On-board EPROM (2708 or 2716), with power on vectoring, allows user memory to reside from 0000H to DFFFH. Disk I/O supporting IBM 3740 soft sectored format is provided for up to four 8" or 5¼" drives and auxiliary software driven cassette. All resident I/O devices may be accessed either in I/O space or as memory locations allowing for optimal program I/O access. A CP/M bootstrap and BIOS EPROM (complett with source listing) is included with the standard system and custom configurations are available on special order. Despite its high functional density the CP/10-1 typically requires only 8V at .8A, +15V at .15A and -15V at .05A.

Priced at $499.00, the CP/10-1 is available from: ARKON ELECTRONIC 409 Queen Street West Toronto, Ontario.
MULTI-BANK MEMORY CARD

The Digiac MAPS-1001 memory card is compliant with the IEEE-100 bus standards for extended memory bank addressing. It has the following features:

- Two independent memory segments. Typical configuration—2 segment by 32K bytes per segment, each originated from OH to 7FFFH.
- Single memory segment 16K to 64K selectable.
- 375ns memory access and 500ns memory cycle time.
- On-board memory bank address decoder-Dip switch selectable.
- Utilizes 4116 memory technology.
- Power dissipation less than 6 watts.

For more information contact: Digiac Corp., 175 Engineers Road, Smithtown N.Y., 11787; tel: (516) 273-8600.

4-CHANNEL SERIAL I/O FOR S-100 SYSTEMS

The Digiac MAPS-1000 MP/M® Universal support Module is an S-100 card designed to meet the demands required by Digital Research's MP/M multi-tasking operating system. All input/output, interrupt generation for task switching, and disk bootstrapping are resident functions on the MAPS-1000.

The MAPS-1000 is totally compatible with the IEEE S-100 bus standards. It provides the business professional or systems designer with a simple and effective way to rapidly install and exploit the total benefits of MP/M.

The MAPS-1000 has the following:

- Four (4) independent EIA RS-232C serial data channels w/jumper selectable data bits, start/stop bits, parity select, and baud rate up to 19.2Kbps.
- Two (2) TTL parallel input/output channels, each with strobe-acknowledge capability, and latched or transparent output.
- On-board phantom bootstrap prom memory w/jumper selectable enable/disable.
- High accuracy crystal controlled interrupt generation capability. Required by MP/M for task switching and time of day/day of week calculations. Utilizes a proprietary interrupt count correcting circuit to insure accurate time calculation for scheduler and TOD functions. Interrupt rate selectable for either 30 or 60 Hz.
- Software selectable restart vector mode-supports all 8080 and Z80 interrupt operations.

Complementing the MAPS-1000 are Digiac developed MP/M xios packages that support several single and double density S100 disk controllers. The Digiac developed xios packages have been preconfigured to operate exclusively with the MAPS-1000. For more information contact: Digiac Corp., 175 Engineers Road, Smithtown N.Y., 11787; tel: (516) 273-8600.

THE CHANNEL PARALLEL/SERIAL I/O CARD AVAILABLE

MicroDaSys has introduced the 4P4S combining four parallel bi-directional data ports (32 I/O bits) with full handshaking and interrupt control (another 8 I/O bits) for S-100 systems. In addition there are four serial RS-232 input and output ports. These serial ports also operate under full handshaking and interrupt control. One portion of the board is pre-drilled and plated through for use as a proto-typing area for custom applications.

Price of the 4P4S is $199 in kit form and $299 completely assembled and tested and is available from MicroDaSys, P.O. Box 36215, Los Angeles, CA 90036, phone (213) 731-0876.

S-100 MICROSYSTEMS
SOFTWARE DIRECTORY

Program Name: TED
Hardware System: 24K or larger Z80 CP/M system
Minimum Memory Size: 20K minimum, 24K recommended
Language: Z80 assembly
Description: TED is an advanced text editor which implements an enhanced subset of DEC TECO commands providing the following capabilities for editing of ASCII text.
*36 command/text buffers
*32 entry push down stack
*sophisticated macro command capability
*conditional and iterative command execution
*conditional and absolute branching
*multiple open files
Release: Available now
Price: $90.00
Included with price: 8" CPM compatible disk with object file, TED. COM and comprehensive manual (manual $20 if purchased separate)
Where to purchase it:
Small System Design
P.O. Box 4546
Manchester, New Hampshire 03108

Program Name: MULTI-USER CP/M
Hardware System: CP/M with 8" floppy disk controller & two 8" hard sectored disk drives.
Minimum Memory Size: 32K RAM
Language: Altair Disk Basic (Rev. 3.4, 4.0, 4.1, & 5.0)
Description: The Speedy Disk Copy routine will copy all or any part of an Altair Disk Basic or DOS formatted diskette in less than 100 seconds. By comparison, the Altair supplied PIP utility requires about 40 minutes. The program is self-prompting, performs both read and write validation checks with a listing of the location and quantity of errors upon completion. Recorded twice in Altair Disk Basic ASCII format on 8" hard sectored diskette.
Release: Currently available
Price: $19.50 Postpaid.
Author: Joe Konrad
Where to purchase it:
Jack Compute
33 Plant Street
New London, CT 06320

Program Name: Multi-User CP/M
Hardware System: CP/M with 8" floppy disk controller & two 8" hard sectored disk drives.
Minimum Memory Size: 32K RAM
Language: Altair Disk Basic (Rev. 3.4, 4.0, 4.1, & 5.0)
Description: The Speedy Disk Copy routine will copy all or any part of an Altair Disk Basic or DOS formatted diskette in less than 100 seconds. By comparison, the Altair supplied PIP utility requires about 40 minutes. The program is self-prompting, performs both read and write validation checks with a listing of the location and quantity of errors upon completion. Recorded twice in Altair Disk Basic ASCII format on 8" hard sectored diskette.
Release: Currently available
Price: $19.50 Postpaid.
Author: Joe Konrad
Where to purchase it:
Jack Compute
33 Plant Street
New London, CT 06320

Program Name: Small-C Compiler
Hardware System: 8000/260
Minimum Memory Size: 48K
Language: C
Description: Small-C is a version of the popular high level language C adapted to the CP/M operating system. The compiler (written in C) produces assembly language for ASM or MAG as its output. The compiler supports a subset of C and also allows assembly language to be included within the C source code with its "#asm...#endasm" feature.
Release: Available now
Price: $15 plus shipping
Included with price: Manual, 8" single density CP/M floppy with executable Small-C, full source code for compiler, the runtime library, and a demonstration program in.
Author: Ron Cain, adapted for CP/M by The Code Works
Where to purchase it:
The Code Works
Box 550
Goleta, CA 93017

Program Name: Small-C Compiler
Hardware System: 8000/260
Minimum Memory Size: 48K
Language: C
Description: Small-C is a version of the popular high level language C adapted to the CP/M operating system. The compiler (written in C) produces assembly language for ASM or MAG as its output. The compiler supports a subset of C and also allows assembly language to be included within the C source code with its "#asm...#endasm" feature.
Release: Available now
Price: $15 plus shipping
Included with price: Manual, 8" single density CP/M floppy with executable Small-C, full source code for compiler, the runtime library, and a demonstration program in.
Author: Ron Cain, adapted for CP/M by The Code Works
Where to purchase it:
The Code Works
Box 550
Goleta, CA 93017

Program Name: Small-C Compiler
Hardware System: 8000/260
Minimum Memory Size: 48K
Language: C
Description: Small-C is a version of the popular high level language C adapted to the CP/M operating system. The compiler (written in C) produces assembly language for ASM or MAG as its output. The compiler supports a subset of C and also allows assembly language to be included within the C source code with its "#asm...#endasm" feature.
Release: Available now
Price: $15 plus shipping
Included with price: Manual, 8" single density CP/M floppy with executable Small-C, full source code for compiler, the runtime library, and a demonstration program in.
Author: Ron Cain, adapted for CP/M by The Code Works
Where to purchase it:
The Code Works
Box 550
Goleta, CA 93017

Program Name: Small-C Compiler
Hardware System: 8000/260
Minimum Memory Size: 48K
Language: C
Description: Small-C is a version of the popular high level language C adapted to the CP/M operating system. The compiler (written in C) produces assembly language for ASM or MAG as its output. The compiler supports a subset of C and also allows assembly language to be included within the C source code with its "#asm...#endasm" feature.
Release: Available now
Price: $15 plus shipping
Included with price: Manual, 8" single density CP/M floppy with executable Small-C, full source code for compiler, the runtime library, and a demonstration program in.
Author: Ron Cain, adapted for CP/M by The Code Works
Where to purchase it:
The Code Works
Box 550
Goleta, CA 93017

Program Name: Small-C Compiler
Hardware System: 8000/260
Minimum Memory Size: 48K
Language: C
Description: Small-C is a version of the popular high level language C adapted to the CP/M operating system. The compiler (written in C) produces assembly language for ASM or MAG as its output. The compiler supports a subset of C and also allows assembly language to be included within the C source code with its "#asm...#endasm" feature.
Release: Available now
Price: $15 plus shipping
Included with price: Manual, 8" single density CP/M floppy with executable Small-C, full source code for compiler, the runtime library, and a demonstration program in.
Author: Ron Cain, adapted for CP/M by The Code Works
Where to purchase it:
The Code Works
Box 550
Goleta, CA 93017

Program Name: Small-C Compiler
Hardware System: 8000/260
Minimum Memory Size: 48K
Language: C
Description: Small-C is a version of the popular high level language C adapted to the CP/M operating system. The compiler (written in C) produces assembly language for ASM or MAG as its output. The compiler supports a subset of C and also allows assembly language to be included within the C source code with its "#asm...#endasm" feature.
Release: Available now
Price: $15 plus shipping
Included with price: Manual, 8" single density CP/M floppy with executable Small-C, full source code for compiler, the runtime library, and a demonstration program in.
Author: Ron Cain, adapted for CP/M by The Code Works
Where to purchase it:
The Code Works
Box 550
Goleta, CA 93017

Program Name: Small-C Compiler
Hardware System: 8000/260
Minimum Memory Size: 48K
Language: C
Description: Small-C is a version of the popular high level language C adapted to the CP/M operating system. The compiler (written in C) produces assembly language for ASM or MAG as its output. The compiler supports a subset of C and also allows assembly language to be included within the C source code with its "#asm...#endasm" feature.
Release: Available now
Price: $15 plus shipping
Included with price: Manual, 8" single density CP/M floppy with executable Small-C, full source code for compiler, the runtime library, and a demonstration program in.
Author: Ron Cain, adapted for CP/M by The Code Works
Where to purchase it:
The Code Works
Box 550
Goleta, CA 93017

Program Name: Small-C Compiler
Hardware System: 8000/260
Minimum Memory Size: 48K
Language: C
Description: Small-C is a version of the popular high level language C adapted to the CP/M operating system. The compiler (written in C) produces assembly language for ASM or MAG as its output. The compiler supports a subset of C and also allows assembly language to be included within the C source code with its "#asm...#endasm" feature.
Release: Available now
Price: $15 plus shipping
Included with price: Manual, 8" single density CP/M floppy with executable Small-C, full source code for compiler, the runtime library, and a demonstration program in.
Author: Ron Cain, adapted for CP/M by The Code Works
Where to purchase it:
The Code Works
Box 550
Goleta, CA 93017
At last there is a magazine written exclusively for S-100 system users. No other publication is devoted to supporting S-100 system users. No longer will you have to hunt through other magazines for an occasional S-100, CP/M* or PASCAL article. Now find it all in one publication. Find it in S-100 MICROSYSTEMS.

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

SOFTWARE

CP/M*
Assembler
BASIC
PASCAL
applications
and lots more

SYSTEMS

Cromemco
Inter systems
North Star
IMSAl
SOL
and lots more

HARDWARE

8 bit & 16 bit CPUs
interfacing
hardware mods
bulletin board systems
multiprocessors
and lots more

*TMK
Digital
Research

Edited by Sol Libes
Published every other month

USA Canada, Mexico Other Foreign (Air)

$23.00  $32.00  $65.00
$17.00  $23.00  $45.00
$9.50   $12.50  $23.50

New Renewal
Payment Enclosed
Visa
MasterCard
American Express
Signature
Card No.
Expiration date

Please bill me ($1.00 billing fee will be added) foreign orders must be prepaid)

S-100 MICROSYSTEMS
P.O. Box 789-M, Morristown, NJ 07960

ORDER FORM

Name ____________________________________________
Address __________________________________________
City ___________________ State ______________ Zip _______

BACK ISSUES

1-1 Jan/Feb 1980 $5.00
1-2 Mar/Apr 1980 $2.50
1-3 May/June 1980 $2.50
1-4 Jul/Aug 1980 $2.50
1-5 Sep/Oct 1980 $2.50
1-6 Nov/Dec 1980 $2.50

Postpaid in USA; add $1.00 per issue foreign postage. Subscriptions start the month following receipt of order. Subscriptions cannot start with earlier issues.
**Program Name:** Master Ledger  
**Hardware System:** CP/M, 48K, 2-8" Drives  
**Language:** CBASIC-2  
**Description:** Master ledger analyzes your business. It includes 12 month's budgets plus 12 month account history, making possible for any tape of financial comparisons will show management if they are meeting their financial goals. Quarterly and year-to-date are also available for any period. Ten different journals, general ledger, trial balance, and budgets are some of the other major reports. The input routines were designed for the operator, easy, fast and verifies all input. Special features include a forced audit trail and a forced balancing system.  
**Release:** Available now  
**Price:** $300.00  
**Included with price:** User documentation, 31 computer programs, warranty.  
**Author:** Keystone Systems, Inc.  
**Where to purchase it:**  
- Keystone Systems, Inc.  
- P.O. Box 767  
- Spokane, WA 99210.

---

**Program Name:** MCALL (Micro proto'Cs.l.L' & communications program)  
**Hardware System:** 8080 or Z80 under CP/M serial port connected to an acoustic coupler, video display terminal.  
**Minimum Memory Size:** 16K  
**Description:** MCALL provides the following major functions:  
1. Time Sharing Terminal emulation.  
2. Disk file transfer between PC (Personal Computer) and TSC (Time Sharing Computer) in either direction.  
3. Disk file transfer between two PC's with error detection and retransmission.  
   
To perform function 3, no coordination between operators is required. The file to be transferred is specified by the transmitting operator; then the transmit command is issued (ESC T). The specified file is automatically opened at the TX end and created at the RX end. Subsequently, each file is closed and a message notifies each operator that the transfer was a success (or not) and displays the total retransmission count. The INFOWORLD Software Report Card for MCALL rated: "Ease of Use" and "Support" as excellent; and "Functionality", "Documentation", and "Error Handling" as good.  
**Release:** Currently available  
**Price:** $50.00  
**Included with price:** Operating Instructions Manual (20 pgs) and 8" single density disk with 84K source file and 8K com file.  
**Author:** Tim Pugh  
**Where to purchase it:**  
- Micro-Call Services  
- 9655-M Homestead Ct.  
- Laurel, MD 20810.

---

**Get 8080/Z80 Source Code on your CP/M compatible disk**  
**DDB** — Directory Data Base Program reads disk directories and builds a data base file for inquiries, find files fast, catalog your library — $30/25  
**COMM** — Menu driven intercomputer communications package for timesharing and other CP/M systems via serial port w/ w/o modem. Modes are: terminal (+ session log), file-to-file (w/ wo protocol + CRC 16), local (dir, ren, era, log, display), FDX/HDX, disk. — $150/75  
**CDIR** — Comprehensive directory utility: Alphabetical list of file extents & allocated blocks: Checks dup allocation for file integrity — $40/20  
**DXAM** — Disk exam/update utility: For memory map video: Any drive, track, sector. Display or update data in ASCII, EBIDIC, HEX or user decode option — $40/20  
**DASM** — Self relocating 8080 dis-assembler: Outputs to CRT, printer & disk as .ASM — $40/20  
**DGEN** — Generator for IMSAI VIO: Runs on VIO: Inputs 3K disk file & edits characters in a 7 x 9 block character simulation — $40/20  
**PRED** — Pre-edit program will update version number maintained in program file, then locate and load CP/M editor or (frame ed. com) and execute — $40/20  
**VIDE** — Memory mapped video drivers: 8080: Any size char/line configuration — $40/20  
**VDR** — Vector draw & plot for memory map video using 2 x 3 block graphic char array, pixel grid: Call from MBASIC or DDB: Test program included (In BASIC) — $30.00  
**SYST** — 16K ROM/"BASIC"/Assembler/Monitor/Debug/Terminal cartridge operating system supplied on cassette, disk or 2708 EPROM (add $150 for firmware) — $90.00  
**GAMES** — Incredible graphics: Space Invaders: Target: StarTrek & more — $150/40  

**Supplied on your disk/or ours (add $15) / or listings-free brochure**  
**Price codes**  
(C) = source/com: (V) = CP/M I/O: (+) = CP/M keybd + Mem Map Video  
**HAWKEYE GRAFIX, 23914 Mobile, Canoga Park, CA 91307  
(213) 348-7990  
S-100 MICROSYSTEMS**
25% Down - Balance C.O.D. - Prepaid orders save freight charges. Minnesota residents must add 4% sales tax or provide evidence of tax number for items to be resold. Twin city - metropolitan prices are slightly higher.

**CRT TERMINALS:**

<table>
<thead>
<tr>
<th>CRT TERMINAL</th>
<th>RETAIL</th>
<th>BIS PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>All ADDS Regent Terminals in stock (20, 25, &amp; 40)</td>
<td>995.00</td>
<td>850.00</td>
</tr>
<tr>
<td>Microterm 2A</td>
<td>945.00</td>
<td>795.00</td>
</tr>
<tr>
<td>Microterm 5A</td>
<td>895.00</td>
<td>750.00</td>
</tr>
<tr>
<td>Microterm 3A</td>
<td>895.00</td>
<td>750.00</td>
</tr>
<tr>
<td>Microterm 100 (DEC VT100 emulator)</td>
<td>1895.00</td>
<td>1695.00</td>
</tr>
<tr>
<td>TVI 912B &amp; 912C</td>
<td>to 995.00</td>
<td>850.00</td>
</tr>
<tr>
<td>TVI 920B &amp; 920C</td>
<td>to 995.00</td>
<td>850.00</td>
</tr>
</tbody>
</table>

**MATRIX PRINTERS (RO):**

<table>
<thead>
<tr>
<th>Printer</th>
<th>RETAIL</th>
<th>BIS PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anadex 9500 printer</td>
<td>1650.00</td>
<td>1475.00</td>
</tr>
<tr>
<td>Centronics 777TF</td>
<td>1350.00</td>
<td>1095.00</td>
</tr>
<tr>
<td>Centronics 737 (add $95 for serial)</td>
<td>995.00</td>
<td>850.00</td>
</tr>
</tbody>
</table>

**DAISY WHEEL CHARACTER PRINTERS (RO):**

<table>
<thead>
<tr>
<th>Printer</th>
<th>RETAIL</th>
<th>BIS PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BudgetWriter III (or Qume 3/55) w/S100 interface card</td>
<td>3050.00</td>
<td>2495.00</td>
</tr>
<tr>
<td>Qume Sprint 5/55</td>
<td>3195.00</td>
<td>2895.00</td>
</tr>
<tr>
<td>Plessey CF4282 (by Data Products)</td>
<td>3505.00</td>
<td>2995.00</td>
</tr>
</tbody>
</table>

**MASS STORAGE SUBSYSTEMS:**

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>RETAIL</th>
<th>BIS PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector 64K RAM Card</td>
<td>495.00</td>
<td>445.00</td>
</tr>
<tr>
<td>Vector Flashwriter II</td>
<td>80 x 24 Video</td>
<td>449.00</td>
</tr>
<tr>
<td>Vector Video Digitizer Card</td>
<td>201.00</td>
<td>175.00</td>
</tr>
<tr>
<td>BudgetStor III-dual 8&quot; drives (1.2MB)</td>
<td>1495.00</td>
<td>1275.00</td>
</tr>
<tr>
<td>BudgetStor III-dual 8&quot; Qume drives (2MB)</td>
<td>1895.00</td>
<td>1625.00</td>
</tr>
<tr>
<td>OPUS SD/SS Mini-diskettes 10 in plastic box - specify 0 or 32 sectors &amp; system SPECIAL 29.50/10 265.00/100 in bulk</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPUS SD/SS 8&quot; Diskettes 10 in plastic box - specify 0 or 32 sectors &amp; system SPECIAL 29.50/10 265.00/100 in bulk</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SOFTWARE:**

<table>
<thead>
<tr>
<th>Software</th>
<th>RETAIL</th>
<th>BIS PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intersystems WW1 Prototyping card</td>
<td>NET</td>
<td>35.00</td>
</tr>
<tr>
<td>Konan SMC-100 Hard Disk Controller</td>
<td>2195.00</td>
<td>1795.00</td>
</tr>
<tr>
<td>NEW! Vector Graphics “ZBP” SBC card</td>
<td>395.00</td>
<td>325.00</td>
</tr>
<tr>
<td>Vector Z800A CPU card</td>
<td>247.00</td>
<td>195.00</td>
</tr>
<tr>
<td>Vector Bitstreamer II card</td>
<td>270.00</td>
<td>225.00</td>
</tr>
<tr>
<td>Vector Motherboard, 16 slot - terminated</td>
<td>291.00</td>
<td>235.00</td>
</tr>
<tr>
<td>Vector Analog Interface Card</td>
<td>132.00</td>
<td>110.00</td>
</tr>
<tr>
<td>Vector Precision Analog Interface Card</td>
<td>449.00</td>
<td>350.00</td>
</tr>
<tr>
<td>Vector High Res. Graphics Card</td>
<td>270.00</td>
<td>225.00</td>
</tr>
<tr>
<td>Vector Video Digitizer Card</td>
<td>201.00</td>
<td>175.00</td>
</tr>
<tr>
<td>Vector Flashwriter II 80 x 24 Video</td>
<td>368.00</td>
<td>295.00</td>
</tr>
<tr>
<td>Vector PROM/RAM III Card</td>
<td>247.00</td>
<td>195.00</td>
</tr>
<tr>
<td>Vector 8K Static RAM Card</td>
<td>282.00</td>
<td>230.00</td>
</tr>
<tr>
<td>Vector 64K dyn. RAM Card</td>
<td>975.00</td>
<td>795.00</td>
</tr>
<tr>
<td>Micropolis Controller Card</td>
<td>518.00</td>
<td>425.00</td>
</tr>
</tbody>
</table>

**CF&A DATA DESKS:**

<table>
<thead>
<tr>
<th>Desk</th>
<th>RETAIL</th>
<th>BIS PRICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anderson-Jacobson A242A (remanufactured)</td>
<td>185.00</td>
<td>150.00</td>
</tr>
<tr>
<td>Novation CAT ans/long, modem</td>
<td>185.95</td>
<td>150.00</td>
</tr>
<tr>
<td>Qume Single sheet paper feeder</td>
<td>1390.00</td>
<td>1195.00</td>
</tr>
</tbody>
</table>

**OPUS S/D/S Mini-diskettes 10 in plastic box — specify 0, 10, or 16 sectors & system SPECIAL 27.50/10 250.00/100 in bulk**

**OPUS S/D/S 8" Diskettes 10 in plastic box — specify 0 or 32 sectors & system SPECIAL 29.50/10 265.00/100 in bulk**

**OPUS RECORDING MEDIA ARE A PREMIUM GRADE FULLY GUARANTEED PRODUCT OFFERING SUPERIOR VALUE... DOUBLE DENSITY OR SPECIAL FORMATS AVAILABLE AT SLIGHTLY HIGHER PRICES**

**DEVELOPMENT SYSTEMS & APPLICATIONS PROGRAMS**

We have a wide variety of development and applications packages from several of the major software houses (RSI “Peachtree,” IMS, Graham- Dorian, and others), plus our own line of proprietary software packages. All attractively priced.
Throughput is the Only True Measure of Computer Performance.

Want a 300% improvement in throughput compared to 2 MHz systems? IEEE-compatible CompuPro boards are designed from the ground up to operate at 6 MHz and beyond, dramatically increasing computing power and performance. Don’t settle for less... select high speed, high reliability S-100 products from CompuPro.

**HIGH SPEED S-100 MEMORY and MOTHERBOARDS**

**RAM XX** (with bank select AND extended addressing) is the perfect match for either CPU board — thanks to fully static operation, extremely low power consumption, and complete IEEE spec compatibility. All unkit and assembled memories work up to 5 MHz, while Certified System Component boards run up to 8 MHz and are guaranteed to work with 6 MHz Z80s. All CompuPro motherboards work up to 10 MHz.

<table>
<thead>
<tr>
<th>Type</th>
<th>unkit</th>
<th>assm</th>
<th>CSC</th>
</tr>
</thead>
<tbody>
<tr>
<td>16K RAM XX-16</td>
<td>$349</td>
<td>$419</td>
<td>$519</td>
</tr>
<tr>
<td>24K RAM XX-24</td>
<td>$479</td>
<td>$539</td>
<td>$649</td>
</tr>
<tr>
<td>32K RAM XX-32</td>
<td>$649</td>
<td>$729</td>
<td>$849</td>
</tr>
<tr>
<td>20 slot motherboard with edge connectors</td>
<td>$174</td>
<td>$214</td>
<td>n/a</td>
</tr>
<tr>
<td>12 slot motherboard with edge connectors</td>
<td>$129</td>
<td>$169</td>
<td>n/a</td>
</tr>
<tr>
<td>6 slot motherboard with edge connectors</td>
<td>$89</td>
<td>$129</td>
<td>n/a</td>
</tr>
</tbody>
</table>

**SEE COMPUPRO PRODUCTS IN PERSON AT COMPUTER STORES WORLDWIDE, OR WRITE US DIRECT IF THERE’S NO STORE IN YOUR AREA.**

**TERMS:** Cal res add tax. Allow 5% for shipping, excess refunded. VISA®/Mastercard® orders call (415) 562-0636, 24 hours. Please include street address for UPS delivery. Sale prices good through cover month of magazine, other prices subject to change without notice.

CompuPro from GODBOUT ELECTRONICS

Bldg. 725, Oakland Airport, CA 94614