the CP/M* and S-100 user's journal

$2.00
U.S.A.

MICROSYSTEMS™

SEP/OCT 1981

VOL.2/NO.5

THE LITTLE-ADA PROGRAMMING LANGUAGE

See Pages 14-25

Also in this Issue

and more

Complete Table of Contents on Page 3

MORROW DESIGNS

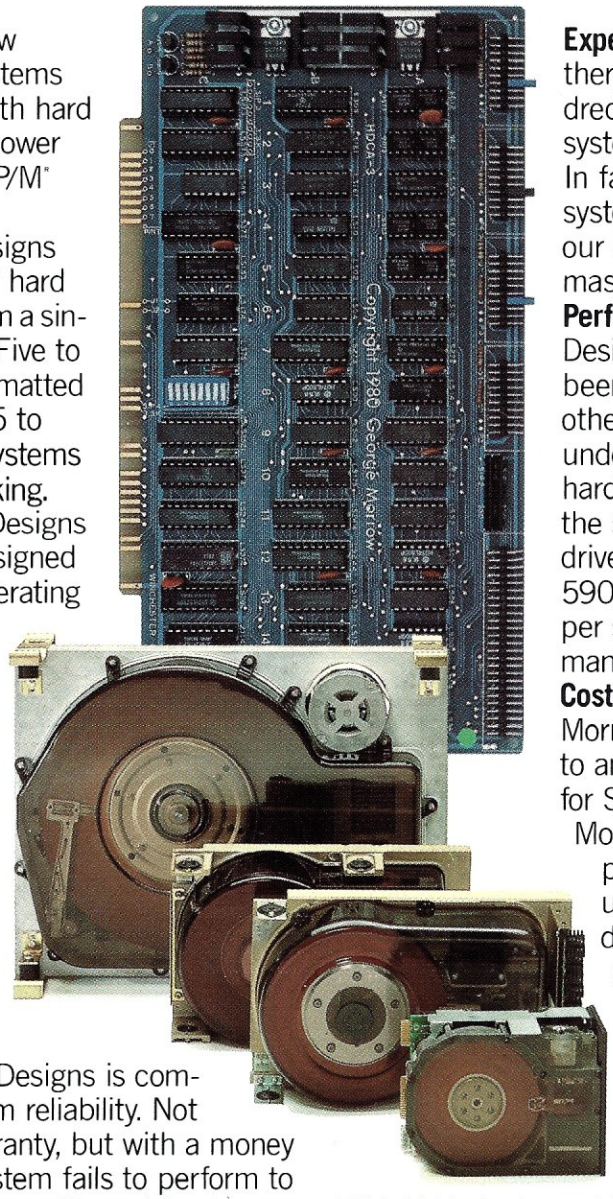
NOW, 26 Mbytes $4,495

Leading edge technology in hard disk systems.

Complete systems. Morrow Designs hard disk subsystems are delivered complete with hard disk, controller, cabinet, power supply, fan, cables and CP/M* 2.2 operating system.

Widest range. Morrow Designs offers the widest range of hard disk systems available from a single supplier. 5¼," 8," 14." Five to over 100 megabytes of formatted hard disk storage. $2,995 to $17,980. Cost effective systems that work. And keep working.

S-100 and more. Morrow Designs hard disk systems are designed for use with the CP/M operating system. Available software packages allow our systems to run on any IEEE696/S-100 Standard system with no hardware modification. Plus, Cromemco,*** North Star,** Vector Graphics, Godbout, Dynabyte, Exidy,**** IMSAI, Micromation, Processor Technology and California Computer Systems.



Reliable systems. Morrow Designs is committed to hard disk system reliability. Not simply with a 90-day warranty, but with a money back guarantee. If our system fails to perform to specification, send it back. We'll send back your money.

Experience. As of April, 1981, there were over fifteen hundred Morrow Designs hard disk systems successfully installed. In fact, over 200 independent systems integrators now use our hard disks to solve their mass storage problems.

Performance answers. Morrow Designs hard disk systems have been benchmarked against all other systems. None is faster under CP/M. Morrow Designs hard disks operate at 10 times the speed of a floppy disk drive. Transfer rates range from 590,000 bytes to 900,000 bytes per second. That kind of performance can become addictive.

Cost effective answers. Compare Morrow prices and performance to anything presently available for S-100 systems. You'll find Morrow's price/megabyte/performance ratio to be unmatched. Leadership in disk systems technology earned us leadership in price/performance. And that may have earned us a call from you. Circle the Reader Service Number for our full line data sheets. Can't wait? Call us at (415) 524-2101. And yes, OEM quantity prices are available. **LOOK TO MORROW FOR ANSWERS.**

*CP/M is a trademark of Digital Research.
**Northstar is a trademark of North Star Computers, Inc.
***Cromemco is a trademark of Cromemco, Inc.
****Exidy is a trademark of Exidy Corporation.

MORROW DESIGNS

5221 Central Avenue, Richmond, CA 94804
(415) 524-2101

MicroDaSys

68K MiniFrame Power

features:

- ***Virtual Memory Management***
- ***Motorola 68000 Processor***
- ***12 Mhz operation***
- ***32 bit computations***
- **4 Billion byte addressing**
- **LRU demand paging**
- **Peripheral processor**
- **UNIX multi-user operating system**

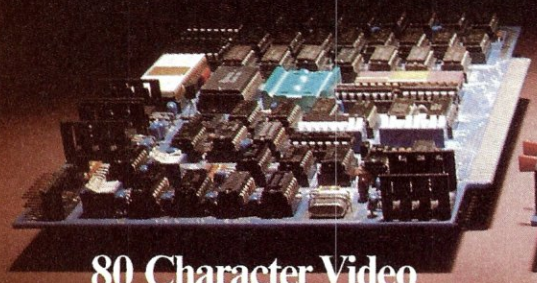
Only the world's finest minicomputer offers these features in systems available TODAY. Complete single-user systems start under $10,000. And a 6 user UNIX-based 16 MB virtual system with 512K Bytes RAM and 24M Bytes hard disc costs under $25,000! Want proof? Call or write now for brochures and information on our 68K Seminars. Or send $75 for the complete technical manual set.

MicroDaSys
2811 Wilshire Blvd., Santa Monica, CA 90403
Call (213) 829-6781

TWX: 910-321-2378

A free poster of this ad is available while supply lasts.

A Busload from SSM.

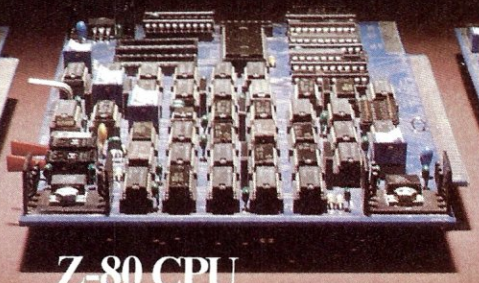


80 Character Video

With 80 characters per line our VB3 is the perfect video interface for word processing. It produces a standard 80x24 display of upper and lower case characters or as much as 80x50 for a full page of text. The matrix for graphic display goes up to 160x200. And with optional EPROM, as many as 256 user programmed characters or symbols can be produced.

VB3 is memory mapped for rapid screen updating. But it occupies memory only when activated. So one or more VB3s can be located at the same address with a full 65K of memory still available to the user.

It generates both U.S. and European TV rates and meets IEEE 696.1 standard. Other features include keyboard input, black on white or white on black, one level of grey, underline, strike thru, blinking char., blank-out char., and programmable cursor. Software includes a CP/M compatible driver and a powerful terminal simulator.



Z-80 CPU

We spent over a year designing the CB2 to assure that it is the most fully S-100 compatible Z-80 CPU on the market.

It operates at 2MHz or 4MHz by DIP switch selection and includes two sockets for 2716/2732 EPROMs or 2K RAMs. Memory sockets can be disabled. Separate run/stop and single step switches allow system evaluation without the benefit of a front panel.

CB2 also features an MWRITE signal, firmware vector jump, and an output port to control 8 extended address lines (allowing use of more than 65K of memory). Jumper options generate the new IEEE 696.1 signals.

8080 CPU

Our CB1A is identical to our popular CB1 with the exception that the on-board RAM has been increased from 256 bytes to a full 1K.

It also features an optional 2K of 2708 EPROMs, power-on/reset vector jump. MWRITE, parallel input port with status and DIP switch addressing.

Our S-100 line includes CPU, Video, I/O, 8 and 16 Bit RAM, EPROM, EPROM Programmer, Prototyping, Terminator, Extender, and Mother boards. Available assembled or as kits.

New SSM Products.

Please call for all the latest details on our newest products:

- **MB 64** 64Kb static RAM memory.
- **I05** two parallel, two serial input/output ports, with complete RS-232 protocol.
- **I08** multiple RS-232 serial I/Os.

SSM Microcomputer Products, Inc.
2190 Paragon Drive, San Jose, CA 95131, (408) 946-7400 Telex: 171171, TWX: 910-338-2077

the CP/M* and S-100 user's journal

MICROSYSTEMS™

Volume 2, Number 5 September/October 1981

Staff

MEMBER MITA

Editorial correspondence is welcomed and should be sent to: Sol Libes, c/o MICROSYSTEMS, Box 1192, Mountainside, NJ 07092. Phone: (201) 522-9347.

*CP/M is a registered trademark of Digital Research.

In This Issue

Departments

MICROSYSTEMS (ISSN #0199-7955) is published bi-monthly by Creative Computing, 39 East Hanover Ave., Morris Plains, NJ 07950. Second class postage paid at Morris Plains, NJ 07950 and additional mailing offices.

Subscriptions are $10 a year (6 issues USA). To subscribe, call our toll free number: (800) 631-8112, or (201) 540-0445 in New Jersey.

POSTMASTER: Send address changes to: MICROSYSTEMS, P.O. Box 789-M, Morristown, NJ 07960.

NEWS & VIEWS

68000 S-100 CPU Card Available

Empirical Research Group, Inc. appears to be the first supplier to start shipping S-100/696 CPU cards using the Motorola 68000 16-bit microprocessor. Complying fully with the proposed IEEE-696/S-100 standard, the card can directly address 16Mbytes of memory. An 8K x 16-bit ROM (containing a Forth development system) is included on the board. The user can select either a 4 or 8MHz clock (incidentally, Motorola has just announced a 10MHz version of the 68000). The board also contains a seven level interrupt controller and TMA (DMA) controller for multi-master operation. ERG also has software available to emulate a Z80 (e.g. can execute CP/M), disk drivers for several of the currently available S-100 disk controllers and a library of software tools and utilities. The CPU card (ERG68-696) is $1,995. ERG Inc., 28206 144th Ave. SE, Kent, WA 98031, (206)631-4855.

Incidentally, Godbout and Cromemco will also soon introduce 68000 CPU cards; see "Editor's Page" for details.

Oasis User Group Activities

The Oasis UG has just released its first volume of public domain software, and a second volume is in the works. Volume 1 contains an assortment of games, utilities and listings of software. Cost is $35. Membership in OUG is $35/yr. For information call Fred Bellomy (805)965-0265, or write OUG, Box 2400, Santa Barbara, CA 93210.

Phase One Systems, the Oasis supplier, is publishing a free newsletter. To get it, just write or call Chris Langewis, Phase One Systems, 7700 Edgewater Dr., Suite 830, Oakland, CA 94621, (415)562-8085.

Call For Papers On Packet Radio & Networking

The American Radio Relay League is sponsoring a conference on Amateur Radio Computer Networking, October 16th, at the National Bureau of Standards in Gaithersburg, MD. The purpose is to explore the possibilities of an integrated amateur computer network suing HF, VHF and satellite packet radio transmission. Papers are being sought on the following topics: network structure, protocols, message handling, equipment design and selection, software, integration with the National Traffic System, interconnection with Computerized Bulletin Board Systems, to name a few. The event will be hosted by Amateur Radio Research & Development Corp. (AMRAD) and Radio Amateur Satellite Corp (AMSAT) whose annual meeting will be held October 17th at the nearby Goddard Space Flight Center. For more information contact: Paul Rinaldo, W4RI, President, AMRAD, 1524 Springvale Ave., McLean, VA 22102.

Godbout Publishes Collection of User Manuals

Many manufacturers refuse to even supply schematic diagrams and service information to purchasers of their equipment. At the other extreme is Godbout Electronics, who have just published a volume including the complete user manuals from every product they have produced, through the end of 1980. It covers 29 products, including their current dual processor, CPU-Z, Spectrum and memory manager boards. The collection is nicely printed and contains a wealth of information. Price is $20; Godbout Electronics, Box 2355, Oakland Airport, CA 94614.

Three New Volumes Added To The CPMUG* Library

The CP/M Users Group (1651 Third Ave., New York, NY 100128) recently released three more volumes of public domain software. Volume 50 contains Pascal material and programs for printing via UNIX. Volume 51 contains the STAGE2 Macroprocessor, and Volume 52 contains COPYFAST 3.5 and BATCH/VARBATCH programs.

The releases are available on 8" IBM single density diskettes or on North Star diskettes readable by users of double density CP/M 1.4, double density 2.2, or quad capacity CP/M 2.2. For prices (orders must be prepaid) and further information, contact CPMUG at the above address. Contrary to reports previously published in *Microsystems*, CPMUG cannot be reached by phone. The phone number listed in past issues of this magazine actually belongs to *Lifelines* magazine. The staff of *Lifelines* is not equipped to handle CPMUG orders, being kept quite busy with the art of magazine publishing. So please use the mailing address for all inquiries.

*registered trademark.

SOL Software Available

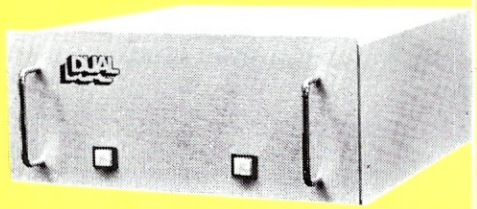
Proteus, the Processor Technology User Group, is distributing PTCo software for the SOL computer, in soure form, to Proteus members. For example, the ALS-8 or Extended Cassette Basic can be furnished on cassette for $65. They are also supplying CP/M public domain software on cassette, with a program to load it onto any CP/M disk. For more information write or call: Proteus, 1690 Woodside Rd., Suite 219-M, Redwood City, CA 94061, (415)368-2300.

POWER & RELIABILITY

68000 μP on the S-100 Bus?

YES, AVAILABLE NOW FROM DUAL SYSTEMS!

- ☐ **8 mHz 68000 microprocessor.**
- ☐ **16-megabyte direct addressing.**
- ☐ **32-bit internal arithmetic.**
- ☐ **Minicomputer type instructions** including MULTIPLY.
- ☐ **FULL IEEE-696 S-100 compliance.** Runs with all 4 mHz S-100 boards and automatically runs faster when accessing Dual Systems memory boards, for FULL SPEED OPERATION OF THE 68000.
- ☐ **Powerful vectored interrupts.** 7 Vectored interrupts, including NMI, as well as alternate mode having up to 256 interrupts.
- ☐ **On board monitor ROM** for immediate use.
- ☐ **Connector for future addition of memory management** unit for multi-user operating systems.
- ☐ **Built to the highest industrial standards** with 200 hour burn-in.



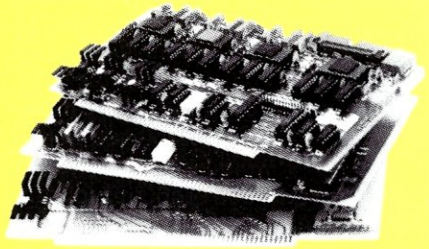
CPU/68000 CPU board..... $1195

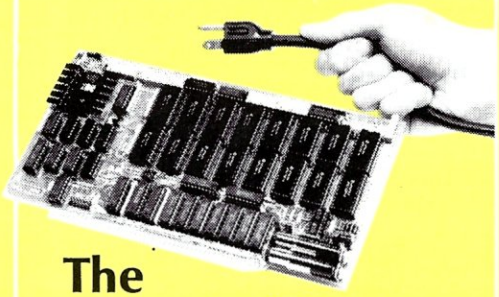
32K-byte 8/16-bit NONVOLATILE RAM board, for secure storage of programs you are developing. Allows FULL SPEED CPU operation. CMEM-32K, per 32K-bytes .. $895

32K-byte 8/16-bit EPROM board, EPROM-32K $395

Serial I/O board, SIO-2 ..... $285

All of the above with cabinet, power supply and backplane..... $3685





The Ultimate IEEE S-100 Memory Would...

- ☐ **BE NONVOLATILE** holding data for up to eight years with the power off.
- ☐ **RUN AT 6 MHZ** without wait states.
- ☐ **RUN IN 8 OR 16-BIT** systems with 8 or 16-bit wide data paths.
- ☐ **HAVE EXTENDED 24-BIT ADDRESSING** and bank select.
- ☐ **HAVE DYNAMICALLY MOVABLE WRITE PROTECT AREAS** to prevent accidental erasure of programs and critical data.
- ☐ **GENERATE POWER-FAIL** interrupts for orderly system shutdown & power failure recovery.

...Available Now from Dual Systems

The Dual Systems CMEM memory boards combine high-speed CMOS memories with new 5-10 year lithium batteries to give you the nonvolatility of an EPROM board while retaining the instant writability of a high-speed read/write RAM. These industrial grade boards are ruggedly built and are burned-in for 200 hours.

- ☐ CMEM-32K, 32K-bytes ... $895
- ☐ CMEM-16K, 16K-bytes.... $795
- ☐ CMEM-8K, 8K-bytes..... $695

Sales representatives in most metropolitan areas.

OEM and Dealer pricing is available.

Toughest Boards in Town... IEEE 696/S-100

SUPER RELIABLE NON-STOP CLOCK

Keeps time with power off. Our industrial clock utilizes a new lithium battery for 3-9 years use. Easiest clock to program you'll ever see. Runs in all S-100 systems.

- ☐ Year, date, hrs, mins, secs, msecs.
- ☐ Uses new LSI CMOS chip.
- ☐ Vectored interrupts.
- ☐ CLK-24 ..... $250

A/D CONVERTER

IEEE696/S-100 AIM-12 industrial standard module designed for industrial analog-to-digital use.

☐ Runs in all S-100 systems. ☐ 32-channel, 16-differential ☐ 12-bit resolution/accuracy. ☐ 25-microsecond conversions. ☐ Instrumentation amplifier. ☐ BASIC program provided. ☐ AIM-12, $695 or $785 w/1-1000 gain transducer amplifier.

D/A CONVERTER

AOM-12 IEEE696/S-100 industrial level digital-to-analog (D/A) converter.

☐ 12-bit ± 1/2 L.S.B. accuracy over full 0-70°C temperature range. ☐ Outputs 0-10, ± 5, or ± 10 volts. ☐ Short circuit protection, all outputs. ☐ Switch-programmable for multiple boards. ☐ AOM-12, $575

VIC 4-20

Standard output for industrial control 4-20 mA D/A converter. Used in conjunction with the D/A board.

VIC4-20, $445.

DUAL 77 Data Acquisition and Control System-

Built to industrial standards; designed for severe environments. BASIC language makes programming easy. Access to hundreds of sensors. Expandability to meet your increased needs. Nonvolatile memory. Power interruption recovery with automatic restart. DUAL 77 is economical; $5985 & up.

DUAL SYSTEMS CONTROL CORPORATION

DUAL

system reliability/system integrity

1825 Eastshore Highway • Berkeley • CA 94710 • (415) 549-3854 or (415) 549-3890 • TWX 910366 2035

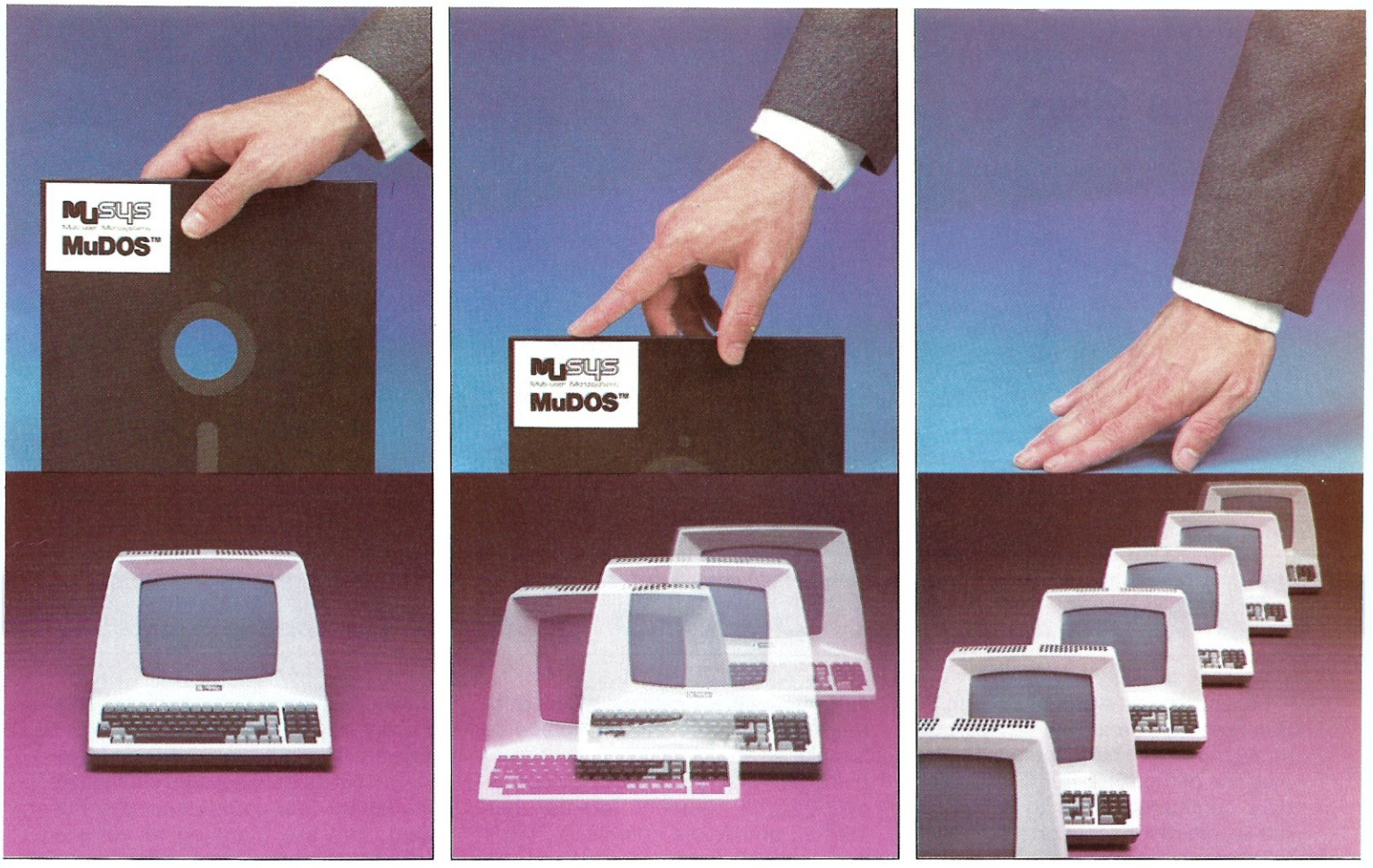
News & Views cont'd...

Micropolis User Group

MUG, the Micropolis Users Group now in its second year of operation, provides a monthly newsletter, membership directory and software library ($15/disk if member does not submit program with disk). Membership is $18/year for North America, $25/year (U.S. dollars only) elsewhere. For more information, write or call Buzz Rudow, MUG, 604 Springwood Circle, Huntsville, AL 35803, (205)883-2621.

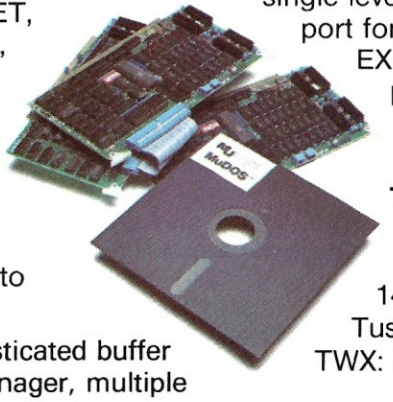
GSA Contract For S-100 Systems Awarded

IRISystems Corp, 10150 Sorrento Valley Rd., San Diego, CA 92121, reports that they have received the first government contract for S-100 based computer systems to be awarded by the General Services Administration (contract no. GS-00C-02635). This marks recognition, on the part of the Federal Government, of the stature of S-100 systems in the government's use of computer systems. ■



Introducing MuDOS.* The rest of the works for networks.

A CP/M compatible replacement for CP/NET****
MuDOS multiplies your micro capabilities with higher throughput, increased reliability and extra professional features for both single and multi-user environments. MuDOS works with any Z80-based micro, in place of CP/NET, MP/M**, or CP/M — and, of course, with MuSYS NET/80* and EXP/80* network slaves.



MuDOS works faster — MuDOS makes the most of the Z80's extra registers and instructions. Program loading is up to six times faster; file processing functions average three to five times faster than CP/M.

MuDOS works smarter — A sophisticated buffer manager, a totally re-entrant file manager, multiple print queuing, disk file support to 67MB, drive support to 2000MB and read-after-write verification of disk updates helps you make the most of your multi-user data and word processing programs. Modular design allows us to tailor the system to your configuration.

Build your network with MuSYS — MuDOS is ideal for use with our NET/80 board (64K RAM, single level interrupt, console port and parallel port for bus communication) and our EXP/80 expansion board (another serial port, Centronics port, priority interrupt control, real time clock, etc.) for S-100 based systems.

This is the year of the network — Make sure you have the works. Call or write MuSYS Corporation, 1451 E. Irvine Blvd., Suite 11, Tustin, CA 92680 (714)730-5692. TWX: 910-595-1967. CABLE: MUSYSTSTN.

MuSYS CORP.
Specialists in Multi-user Microsystems

*MuDOS, NET/80, and EXP/80 are trademarks of MuSYS Corp. **CP/M, MP/M and CP/NET are trademarks of Digital Research.

by Sol Libes

EDITOR'S PAGE

I spent the last week of June in the San Francisco area. The main purpose of my trip was to participate in the last meeting of the IEEE 696/S-100 Standard committee (prior to submitting the Standard for adoption—see article in this issue). At the same time I took the opportunity to visit with several S-100 manufacturers in the area: Cromemco, North Star, Morrow Designs and Godbout Electronics. The purpose of my visits was two-fold. First of all, I wanted to find out about upcoming products. Secondly, I have been attempting for some time to determine the size of the S-100 market and also the number of S-100 systems in operation.

I feel, after talking to several of the S-100 manufacturers, that I have a fairly reliable estimate of the S-100 marketplace. It is apparent that this year total gross S-100 sales will amount to at least $300 million and possibly as high as $400 million. This is greater than either the sales of the Apple or TRS-80, but not quite equal to the combined sales of both. The leaders in the S-100 marketplace are Cromemco ($50M), Vector Graphics ($30M) and North Star ($25M). The remaining business is shared by close to a hundred other manufacturers of which about a dozen will do over $10M, but less than $20M, with the rest doing under $10M.

It is much more difficult to determine how many systems have been built, since several S-100 manufacturers have gone out of business (and some kept no records), and several of the current manufacturers have kept poor records. Therefore, all I am able to do is make an educated guess. I estimate that there are now about 500,000 S-100 systems operating (greater than TRS-80 and Apple combined). More than half of them are running with CP/M. A very large number are being used in dedicated applications that are transparent to the user (e.g. inside a test set of controlling a machine).

Now let me fill you in on the doings of the four companies I visited.

Cromemco

Cromemco Inc. is presently the largest manufacturer of S-100 systems, world-wide. Sales for 1981 are expected to exceed $50M. They have a modern 200,000 sq. ft. facility in Mountain View, CA and employ over four hundred people. It is interesting to note that they expanded their plant this year by a factor of eight (it was 25,000 sq. ft. last year) in anticipation of a continued healthy growth rate. About 75% of their business is complete systems.

Harry Garland, the president of Cromemco, disclosed some of the new products we can expect to see in the coming months. In January they plan to announce a new 68000 16-bit CPU card and a 68000 version of their CROMIX operating system (UNIX-like). The S-100/696 CPU card will contain a Z80 as well as 68000, and the user can switch between the two under software control. Hence, the user can continue to run Z80 software as well as 68000 software. Cromemco expects to eventually make most of their current software available in 68000 code as well as Z80 code. They currently have fifty 68000 CPU cards running in the plant, and are planning to have a preproduction run of a hundred more cards before actual production starts in January. The pricing for the card and operating system has not yet been established. However, Garland indicated that it will be substantially more expensive than their Z80 CPU and OS. Cromemco will also introduce 16-bit wide memory with an extra parity-bit added to each byte and a parity-checker circuit on the card.

Coming in the late spring will be a networking system (for the "office of the future") which will interconnect "work stations" via a coaxial cable, operating in a manner similar to the Xerox Ethernet System. Its operation will be a little slower and less expensive. Naturally, Cromemco plans to introduce an integrated S-100 work station system which contains a 12-slot mainframe, display keyboard, dual mini-floppies and mini-winchester.

North Star

At North Star I met with Charles Grant, the president, co-owner and co-founder of the company. It may seem as if North Star Computers Inc. is no longer in business—discontinuing advertising in *Byte, Microsystems*, et al—but appearances are sometimes deceiving! North Star is actually going through a very aggressive expansion. Early this year they moved from

SOFTWARE
SOFTWARE
DIGITAL MARKETING™
DIGITAL MARKETING

September 1981

DIGITAL MARKETING • 2670 CHERRY LANE • WALNUT CREEK • CALIFORNIA 94596 • (415) 938-2880

MILESTONE™ - $295. Manual alone - $30.

"Critical path" network analysis program for scheduling manpower, dollars and time to maximize productivity.

An interactive project management program that runs under CP/M and can relate together different skills, hourly pay rates and projects to maximize efficiency. MILESTONE could be used to track paper flow, build a computer, check a salesman's performance, or build a bridge. MILESTONE can be used by executives, engineers, managers, and small businessmen to:

- Find the critical tasks which can't be delayed.
- Discover which tasks are not time critical.
- See how manpower and expenses vary versus time.
- Investigate tradeoffs between manpower, dollars and time.
- Give plans to others using a printed project schedule.
- Change details and immediately see the results on screen.

Requires 54K RAM and CP/M. Specify Z80, 8080 or CDOS. Also available for Apple Pascal, UCSD Pascal or CP/M-86 operating systems.

Formats: 8, NS, MP, CDOS, SB, TRS2, APPL

DATEBOOK II™ - $295. Manual alone $25.

- **NOW SCHEDULES APPOINTMENTS FOR UP TO 27 DIFFERENT PEOPLE**
- **IMPROVED FILE STRUCTURE ALLOWS APPOINTMENTS UP TO ONE YEAR**
- Replaces your office appointment calendar
- Searches for openings that fit time of day, day of week & day of year contraints
- Appointments made, modified or cancelled by a few key strokes
- Copies of day's appointments can be quickly printed

Requires 54K RAM and CP/M. Specify Z80, 8080 or CDOS. Also available for Apple Pascal, UCSD Pascal or CP/M-86 operating systems.

Formats: 8, NS, MP, CDOS, SB, APPL, TRS2

SUPERCALC™ - $295. Manual alone - $25.

Allows a non-programmer to manipulate business data in a variety of forecasting and accounting applications. Combines the interactive nature of an electronic spreadsheet with the power and convenience of a simple simulation language. Video display can be scrolled over entire worksheet using cursor controls. Symbolic vector references eliminate repetitive low-level data manipulation commands. Menu driven "Help" command may be invoked at any time.

Requires CP/M. Formats: 8, NS, MP, SB, APPL, TRS2

ACCESS/80™

Level I - Report Generator - $295.
Read ASCII files and create sorted reports with subtotaling capability. Provides multi-dimensional cross tabulation and computation. Includes operating system commands.

Level II - Output & Logic Processor - $495.
Everything in Level I plus, write out new files in any sorted order (including subtotaling). Load arrays from files. Performs binary search on sorted arrays in memory. Includes control language extensions for complex applications.

Level III - Information Management System - $795.
Available 1st/Qtr '82
Everything in Level I and II plus, create new files with data entry. Provides on-line updating and file indexing.

Requires CP/M & 54K RAM
Formats: 8, NS, MP, CDOS, SB, TRS2, APPL

PASCAL/M™ - $225. Manual alone - $20.

CP/M™ compatible language for 8080/Z80 CPUs, supports full Jensen & Wirth plus 45 extensions to Standard Pascal including Random access files, 40 segment procedures & 16 bit BCD real type. **NOW INCLUDES** symbolic debugger which features trapping on stores, examining and changing varibles and tracing of program execution.

Requires CP/M 2.2 & 56K RAM. Formats: 8, NS, APPL, TRS2

PASCAL/M for 8086/88 - $270.

Manual alone - $20.
All features of Pascal/M for the 8086 and 8088 processors running under the 8086/88 version of CP/M.

Requires CP/M-86™ & 128K RAM. Format: 8

TRANS 86™ - $125. Manual alone - $20.

8086/88 Translator for existing 8080/Z80 programs. The new source code can be easily edited and assembled using ACT II to produce hex code which can be executed by the 8086/88. Emphasizes the extensions and features available in the 8086/88.

Requires CP/M & 32K RAM. Formats: 8, NS, APPL

ACT I™ - $125. Manual alone - $15.

CP/M compatible macro assembler for Z80, 8080/85, 6502 & 6800.
One assembler that supports all major 8 bit micros. ACT features include full macro capabilities, comprehensive pseudo-ops, link-file structures, cross reference map, and algebraic expression processor. Requires 24K RAM & CP/M.

ACT II - $175. Manual alone - $20.

CP/M 2.x compatible cross assembler for 8086/88

ACT III - $125. Manual alone - $20.

CP/M 2.x compatible cross assembler for 6809.

ACT I and ACT II together - $225.

Formats: 8, NS, CDOS, MP/M, TRS2, APPL

SUPERDOS™ - $129.

Upgrade of CP/M 2.2 for Superbrain. Includes ADM/31, Hazeltine, or Superbrain Terminal emulation mode. Other new features include 132 character keyboard buffer, repeat on all keys, key click, user programmable numeric keypad, 30% disk read/write improvement, real time clock, baud rates to 19.2K on RS232 ports, printer handshake modes, 4 new utilities, and 4 fixes.

Requires Superbrain 3.0. Format: SB

MORE SOFTWARE

	PRODUCT/MANUAL ALONE
TEXTWRITER III	$125./$25.
PEARL LEVEL 1	$130./$35.
PEARL LEVEL 2	$350./$35.
PEARL LEVEL 3	$650./$50.
CBASIC2	$120./$25.
ULTRASORT-II	$175./$20.
FABS	$175./$20.
CBASIC/86	$325./$30.
SELECTOR/86	$650./$35.
SELECTOR IV	$550./$35.
GLECTOR for SELECTOR IV	$450./$35.
S-BASIC	$295./$35.
baZic	$150./$25.
dBASE II	$695./$50.
MAGIC MENU	$ 75./n/a
w/Spellguard option	$ 95./n/a
SPELL MENU	$ 95./n/a
MONEY MAESTRO	$200./n/a

DEALER INQUIRIES INVITED

PLAN80™ - $295. Manual alone - $30.

A financial modeling system that's easy to use yet powerful enough to replace most timesharing applications.

Lets you tackle any numeric problem that can be defined in a worksheet format. It performs complex calculations quickly and precisely and lets you examine "What if?" questions so you can evaluate more planning alternatives in greater detail.

With PLAN80 you get more than your calculated results . . . you know **how** you get them, because you define rows and columns with familiar names such as UNITS, PRICE and JANUARY and express calculations in terms such as SALES= UNITS*PRICE. It's easy to review your assumptions and methods with people who have never seen PLAN80.

At any point in the PLAN80 model you may display or print results on your screen, printer or disk, save all or part of the results for use by another model, or play "What if?" by inputting new values, recalculating and displaying or printing results.

Best of all, you can incorporate PLAN80 results into any report that requires a financial model — using your word processor — to create professional results.

Requires 56K RAM and CP/M. Also available for CP/M-86. Specify Z80, 8080 or CDOS. Formats: 8, CDOS, NS, MP, SB

SPELLGUARD™ - $295. Manual alone - $20.

20,000+ word dictionary containing commonly used words that find spelling & typographical errors in text files. Allows review of mis-matched words & speedy search routine. Proofs at 10,000 words/min.

Requires CP/M, 48K RAM & Magic Wand, WordStar™ or Spellbinder™

Formats: 8, NS, MP, SB, TRS2, CDOS

SPELLBINDER™ - $395. Manual alone - $50.

Full feature word processing system with Office Management capabilities. Its special features include ease-of-use by office personnel, flexible print formatting & output, and a powerful macro capability which allows features to be added for the unique requirements of each user. Mail list macro is included for mail merge with form letters.

Requires CP/M & 32K RAM. Formats: 8, NS, MP, CDOS, SB, APPL

MCALL™ - $85.

Communications program designed to drive an acoustic coupler. Features include:

- Time sharing Terminal emulation
- Disk file transfer between CP/M computer & Time Sharing Computer in either direction
- Disk file transfer between two CP/M computers with error detection and correction

AMCALL™ - $95.

Auto-answer, auto dial version of MCALL currently supports IDS 88-modem & Potomac Micro-Magic MM-103 boards.

Requires CP/M. Formats: 8, NS, MP, SB, CDOS

MICROSTAT™ VERS. 2.0 - $295. Manual alone - $25.

Now includes stepwise and more flexible file structure. Code size smaller (= more data area). Also a data management subsystem for editing, sorting, ranking, lagging, data file transfers PLUS eleven data transformations (e.g. linear, reciprocal, exponential, etc.)

- Frequency distributions, 8 probability distributions
- Multiple regression, correction analysis
- Time series
- Crosstabs/Chi-Square, non-parametrics
- Factorials, permutations, combinations
- Scatterplots
- ANOVA (one and two-way)

Requires 48K RAM, NorthStar Basic or CP/M & CBASIC2 or Microsoft Basic 80.

Formats: 8, NS, MP, SB, TRS2, CDOS, APPL

FORMAT CODES: 8 (8" single density IBM soft-sectored) NS (NorthStar DD) MP (Micropolis Mod II/Vector MZ) SB (Superbrain 3.0) CDOS (8" Cromemco CDOS) TRS2 (TRS-80 Mod II) APPL (Apple II)

Telex 17-1852 DIGMKTG WNCK DEALER INQUIRIES INVITED OUTSIDE CALIFORNIA CALL 501/442-0864 INSIDE CALIFORNIA CALL 415/938-2883

Berkeley, CA to a new 90,000 sq. ft. facility in San Leandro, CA. Further, they opened regional sales offices in Boston, MA and Columbus, OH. Employees now total close to two hundred, and they expect 1981 sales to reach $25M. Sales for 1980 were $12M. There are now over 22,000 North Star Horizon systems and over 12,000 North Star disk systems operating.

North Star has finally released their version II.0 of Pascal ($199, manual $40) which is greatly enhanced over version 1.5. They have also released an Application Development System package that will allow programmers to develop custom business software in the C language. Included are an editor, compiler, 8080/Z80 assembler and an extensive library of utilities and functions in source form (for programmers to include in programs) which can be included in applications at compile time. Included, too, is attendance at a seminar. The ADS package is $2000 plus a $1500/yr maintenance fee and $35 royalty charge for the OS.

New hardware from North Star includes a 5-1/4", 5MB hard disk system, an 820KB 5-1/4" floppy drive (96 TPI) and 1/4" tape cartridge (813.4MB) back-up system for hard disks.

It should be noted that North Star has made a big effort to improve their foreign sales organization and that foreign sales now account for 30% of their gross sales.

Morrow Designs

George Morrow is a unique guy. He is a former college math teacher who caught the micro bug and started designing "better designs" for other S-100 makers. Four years ago, he took the plunge and started his own business. Located in a 23,000 sq. ft. plant just north of Berkely, Morrow expects to do about $12M business this year. He currently employs a force of 35. By the end of this year Morrow will have shipped over 22,000 disk systems.

Morrow Designs is working hard on getting out initial shipments of its new multi-user system in which each user has his own Z80. Morrow also has developed, in house, a new Unix-like operating system and C language compiler. The operating system makes use of a specially designed memory management system circuit. Coming along toward the end of the year will be a new 5-1/4" winchester disk system.

Godbout Electronics

Bill Godbout refuses to talk about "gross sales" and "numbers of systems," but there is no doubt that Godbout Electronics would rank in the top-ten of S-100 suppliers. Although Godbout introduced a mainframe last year, most of their business is still selling S-100 boards. Kits are a small percentage of boards sold. Located in four old warehouse buildings at the Oakland Airport, the physical nature of the facilities hide the true size of Godbout's operation. Only the first runs and assembly of systems are handled here. All large production runs are subcontracted.

New products from Godbout will include three 16-bit CPU cards. There will be new 8086 and 68000 cards. There will also be a new 16-bit CPU card, to be introduced by year-end. Godbout is keeping tight wraps on this particular card (suffice to say I know of no other company currently planning a similar card). Godbout will also introduce a hard-disk controller and complete disk systems. ■

ANNOUNCING A REVOLUTION IN THE COST OF PROFESSIONAL SOFTWARE

VISACCOUNT is a fully integrated business and accounting system designed for use in small businesses. VISACCOUNT is extremely comprehensive and professional, yet it is very easy to use. The system is controlled from a series of interconnected menus permitting user-friendly operation. Everything you need to set-up and operate the system is provided with the VISACCOUNT package. Experts have estimated the development costs for a fully integrated software system ranges between $7,200 and $22,000.† When you buy software the developer has to recapture this expense. Computer Services Corporation of America is selling its software with a view that volume sales can almost negate this development cost.

OUR GUARANTEE — Buy both our software and that of our competitors (who will no doubt charge several times our price because they need to recapture their development cost). Compare the two systems and we know you'll return theirs (make sure they'll let you return their software). If you decide not to keep our system, then return it within 45 days for a full refund. Once you've used our system we're confident you'll be delighted.

†Microcomputers for Business, Applications, 1979

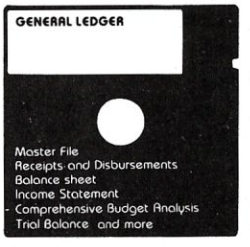
VISACCOUNT

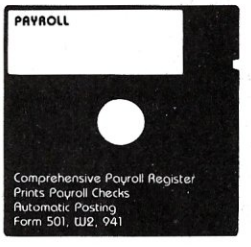
What You Receive

- Nine 5¼'' double density disks (or six 8'' single density disks)
- Easy-to-use operator's manual (over 200 pages)
- Self-study guide on bookkeeping and accounting (over 180 pages)
- Cassette based instruction program on set-up and operation

Available for Apple*, TRS-80, and most others

*The Apple version requires the Microsoft Z80 softcard.
CSCA has CBASIC2, CP/M and Microsfot Z80 softcard in stock.





EXTRA: MAILING LIST PROGRAM

Features

Menu Driven: The entire system runs from a single master menu which accesses numerous subsidiary menus, when needed, to perform the full spectrum of business and accounting functions.

Self-Documenting: All the information needed to use the sytem is provided in an easy to self-study format.

Requirements: 48K CBASIC2
2 DISK DRIVES CP/M

Send $159 for the VISACCOUNT system

COMPUTER SERVICES CORPORATION of AMERICA
332 East 30th Street New York, New York 10016
Order Toll Free 1-800-221-2486 ext. 1591
Technical Number 1-212-685-0090

Name _____

Address _____

City/State/Zip _____

☐ Master Charge ☐ Visa ☐ American Express

No. _____ Expires _____

Your System _____

Disk Size ☐ 5¼'' double density ☐ 8'' single density

LETTERS TO THE EDITOR

Dear Editor:

I am the owner of a SSM VB-3 video board (*Microsystems* Vol. 2/No. 2, pg. 26) and have had the same troubles mentioned in the article. I looked the board over very closely and decided that my problem would take more time to solve than I could spare. I returned it to SSM, they corrected a pair of shorted runs and returned the board, with a bill for $30.00. I was expecting a larger bill and was pleasantly surprised—until I plugged it in and got the waves mentioned in your review. The board would work on an extender so I assumed it didn't like my system. I proceeded to install the software anyway and had the board running, giving me 35 lines of 80 characters—such a great improvement over my 64 x 16 display that I decided to fix the waves.

The fix took about twenty minutes (aargh). The input capacitors on the five volt and twelve volt supply lines tie to this litle pad, and it doesn't go anywhere. Ground the pad to the IC above it and the board should give you a good display. If this does not do it, double the value of the capacitor on the five volt line.

SSG Stanley P. Miller
Fort Hood, TX

Dear Editor:

In the May/June 1981 issue is a letter from Ivan Berger of New York requesting CP/M BIOS for the Versafloppy. Despite the availability of the Versafloppy, getting a version of CP/M for it is difficult. Like the man in the letter, I attempted first to get help from S.D. Sales to no avail. For $25.00 I was able to purchase a ROM. Coupled with a diagnostic program furnished by them, this is useful for accessing tracks and sectors at random. I have yet to get CP/M to run with it. Communication with Lifeboat and other vendors also yielded no CP/M for the Versafloppy.

Finally, I did contact a small computer store in New York, now out of business, who would furnish this product. The last source for this item appears to be Ed Weixal, 505 West End Avenue, Apt. 7A, New York, New York 10024. His system requires Zapple, but otherwise is very fine. I was able to disassemble it and reassemble for CP/M 2.2.

It is strange that the Versafloppy has such poor software support. Perhaps others will use this information.

Robert C. Luckey, M.D.
Richland, WA

Dear Editor:

Your March/April issue was the best I've seen so far. Bill Machrone's ZTEL article was just great! I was very impressed with it.

However, I was not very impressed with Chris Terry's "CP/M Connection-Part 4" and with the suggestions made. After saying no one should use CBIOS drives, the author proceeds to spend half of a page explaining how to do so. Under CP/M 2.2 (and MP/M) you can do anything that you should be doing through BDOS calls. While these kluges into the CBIOS work fairly well under CP/M, they can cause magnificent system crashes under MP/M. Yet the programmers at MicroPro, Microsoft and numerous other places blithely pick up location 1 and so forth. One would get the impression that they've never upgraded from 1.4 CP/M. (MP/M? What's that?)

Secondly, to find the last byte of RAM from the BDOS pointer, you must subtract 7, not 1 as shown. (This goes for 1.4, too.)

It's really a shame when customers try to run a piece of professional software, and see the system go out to lunch instead. Let's avoid these pitfalls and do things the way Digital Research says to (in the manual—that dusty book with the blue and white cover).

Incidentally, double-byte results are returned in HL, not in BC; CP/M 2.2 requires a minimum 20K system; CP/M *always* uses a 128 byte buffer, regardless of density or disk type; and the print message (function 9) will print any characters at all, including those generated by cursor positioning macros.

How about let's make the new programs transportable; if we *strictly* adhere to points 1 through 4 in the first section of the article, programs we write will run under CP/M or MP/M with no conversion (unless, of course, you use the MP/M xdos functions). No applications program has any need to access the CBIOS, nor should any have ever been written that did.

Richard A. Rodman
McLean, VA

Dear Richard: Chris Terry does not actually say "no one should use CBIOS drivers" in this article. He points out the advantages and disadvantages of using CBIOS calls, carefully noting where problems may be created. —*Editor*

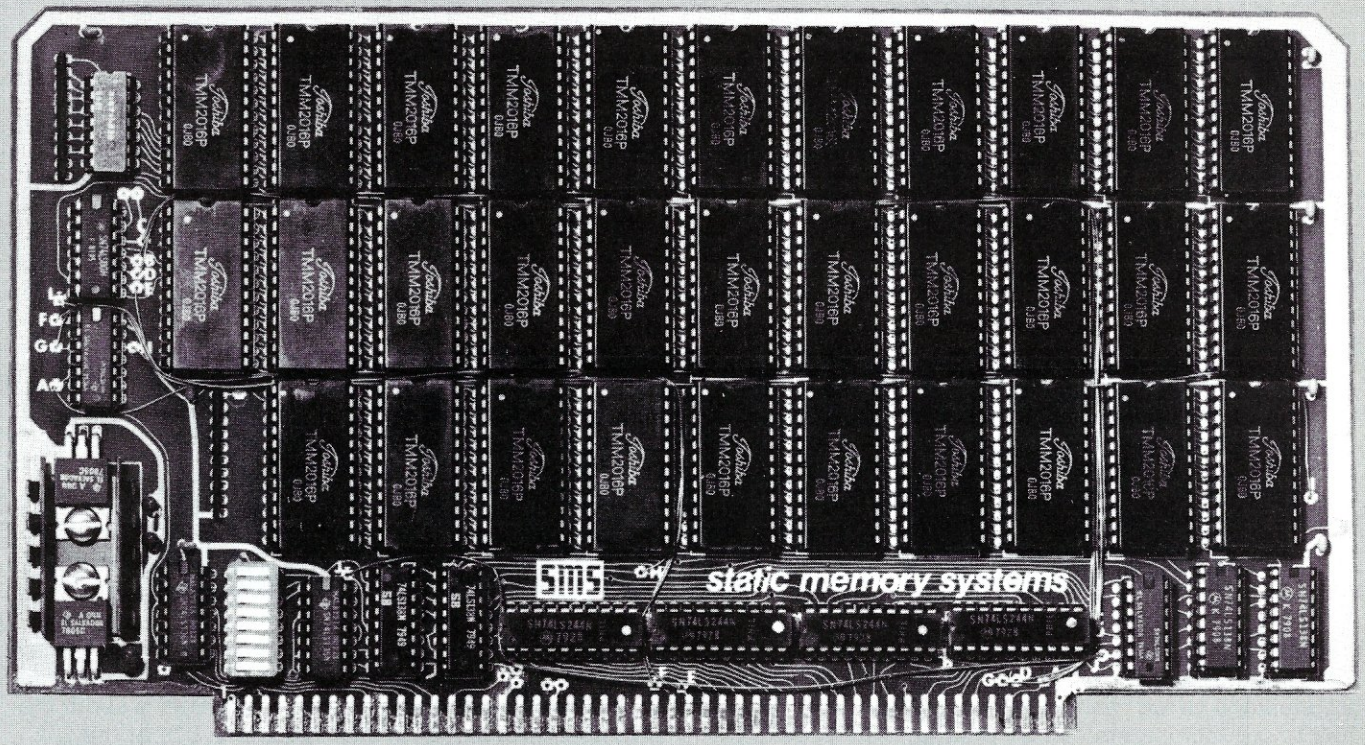
Response from Chris Terry

The truth of R.A. Rodman's statement that you "can do anything you should do" with BDOS calls depends on interpretation of "should." This is generally true for application programs. However, system utilities which want to access the directory (SAP, XDIR, etc.) or directly access and modify the disk (DU, RESOURCE, etc.) cannot use BDOS calls—the primitives are essential. Of course, you use them at your own risk!

Claude Kagan (creator of the SAM76 language) points out that for true transportability to Cromemco CDOS, as well as CP/M systems, the top of the TPA should be about 200 bytes below the address of locations 1-2.

Double byte results returned by CP/M come back with the high byte in B and the lower byte in A (see Interfacing Manual, page 20, CP/M Entry Point Summary). I wrote "in BC" by mistake. ■

THE LAST MEMORY



64K STATIC RAM/EPROM BOARD

At last a 64K *STATIC* memory board for S100 systems. But it's not just a 64K static RAM board, *EPROM's* can also be intermixed with RAM making it the only memory board needed for S100 systems. That's why we call it **THE LAST MEMORY.**

- *64K DENSITY*

THE LAST MEMORY uses the new 2016 byte-wide 16K static RAM to achieve a board density twice that possible with old 2114 static memories.

- *2716 EPROM COMPATIBLE*

A separate board is no longer required for EPROM's containing monitors, bootstrap loaders, etc. 2716 EPROM's can be inserted into the board without modification.

- *SIMPLE ADDRESS DECODING*

Where memory is required, just plug a RAM or EPROM in the corresponding socket. Empty memory sockets occupy no memory space, providing compatibility with memory mapped I/O devices.

- *EXTENDED ADDRESSING*

THE LAST MEMORY includes the IEEE S100 extended addresses. These are fully decoded allowing expansion to a full 16 megabyte system memory.

- *FAST*

The standard board allows 4 MHz operation.

- *LOW POWER*

Only one memory IC is ever active in byte-wide memory systems. The result is far less power consumption than older 16K static memory boards.

- *LOW COST*

Its best feature is the price:

	Kit	A&T
RAM-less Board	99.99	139.99
16K RAM	249.99	289.99
32K RAM	389.99	429.99
48K RAM	519.99	559.99
64K RAM	639.99	679.99

sms **static memory systems**

15 So. Van Buren Ave. Suite 209

Freeport, Illinois 61032 (815) 235-8713

VISA

Little Ada (Part I)

by Ralph E. Kenyon, Jr.

Notes on the implementation of a subset of the new DOD language on the PolyMorphic Systems System 8813 (an 8080A microcomputer).

Editor's Note

Due to the length and structure of this article, it will be presented in four parts. They are:
I. Background
II. Operation of Run-time Interpreter and compiler. Includes sample programs.
III. Run-time Interpreter—Source Code
IV. Compiler—Object Code.
Parts II, III and IV will be published in upcoming issues of *Microsystems*.

Part I. Backgound

As general purpose machines, computers can be programmed to do many different things. One rather unique task computers can be programmed to do is to 'simulate' another computer (even when the other computer does not exist). In fact, the design of some of the larger computers involved simulating the computers before they were built. On a much smaller scale, the design of a microprocessor chip instruction set is often simulated in order to test its design. This is done before a costly and irreversible investment is made in hardware.

Once someone invents an idea for one purpose, people have a way of using it for some other purpose. An example is the class of programs known as "emulators." Emulators are written to simulate one microprocessor on another. For example, an emulator for the 8080 has been written to run on the 6502. Programs written in the 8080 instruction set can be run using this emulator. In this instance, while the program was written for the 8080, and the hardware is the 6502, the program works just as if it were running on 8080 hardware.

In a similar manner, the design of the instruction set of the microprocessor being developed is used to write an emulator for the set. Since the machine being designed does not yet exist, the programs written with its instruction set are programs written for computers which do not exist. Machines can be designed and programmed without even being built. There is no rule that requires an instruction set eventually be made into a hardware device. Existing hardware machines can be used to allow writing programs in instruction sets for non-existant machines. These non-existant machines can be of special purpose or general purpose design.

My object here is to examine one of these non-existant machines which was designed to meet a specific need. I will also illustrate one emulator for it on the 8080A and briefly describe its application. The machine in question is called the L - Machine. Specific versions are referred to as the L/0, L/1, etc. versions. This article is about the structure, emulation on the 8080A and use of the L/1 - Machine.

The L - Machine was invented by Dr. Robert F. Mathis of Old Dominion University (ODU). Dr. Mathis is developing a compiler for the new Department of Defense (DOD) language Ada. The L - Machine was invented specifically for the purpose of implementing a subset of Preliminary Ada on the ODU DEC 10 computer. That subset of Ada is called "Little-Ada."

The L - Machine design is an attempt to produce a machine of very simple structure, but which is very powerful. It is a stack-oriented machine which executes all its instructions using a data stack. The data path is 16 bits wide with instructions of 8, 16, and 24 bits. There are only a very few primitive instructions which are shown in Figure 1.

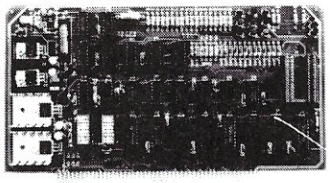
Figure 1

Instruction	Explanation
br addr	Branch or goto address specified
bnz addr	Branch conditional if top of stack is not zero
lic constant	Load an immediate constant into top of stack
opr subop	Operate on the stack using sub-operation
call level,addr	Call subroutine at level and address
lad level,addr	Load true stack location at level and address onto top of stack
nop	No operation

Ralph E. Kenyon, Jr., RFD Lower Prospect Hill, Chester, MA 01011.

PUT YOUR SYSTEM ON THE FOREFRONT OF TECHNOLOGY

PROM BLASTER

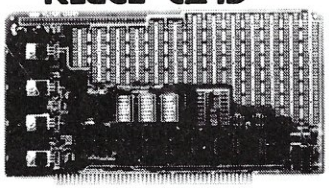


PROGRAMS MOST FAMILIES OF EPROMS!

- Accepts 1K/2K/4K or 8K EPROMS
- Extended Device Option
- Phantom Slave Option
- All Programming Characteristics Software Controlled
- Accepts Single or 3 Supply Parts
- Device Address Switch Selectable

Kit Price $199.95

KLUGE CARD

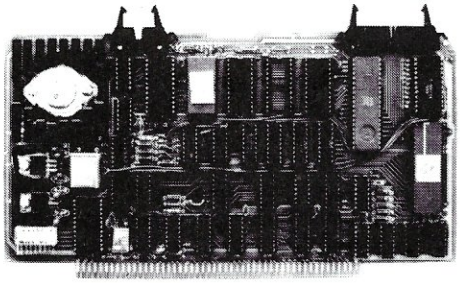


SIMPLIFY YOUR PROJECTS WITH A PROTOTYPE BREADBOARD WITH EXTRAS

- 4 On-Board Pwr. Supplies up to 3 of which can be +5, or ±5, ±12
- Switch Selectable Memory or Device Address
- On-Board Address/Device Decoding
- Bi-Directional Data Bus Buffering
- On-Board Wait States

Bare Board $39.95

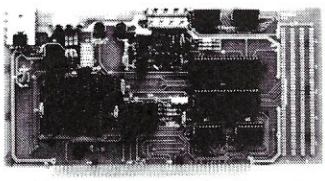
6809 SINGLE BOARD COMPUTER



- 2K ROM
- 4K/8K/16K RAM
- 20 Parallel I/O Lines
- RS 232 Interface
- Baud Rates from 110 to 9600
- All Memory/I.O. Relocatable on 4K Boundaries
- 8080 I/O Instructions Memory Mapped (Gives 256 I/O Ports)
- Complete Documentation
- ADSMON Monitor Includes User Callable Functions, Autopatch and more

Kit Price $299.95 (Includes Software)

NOISEMAKER



- Two On-Board Audio Amplifiers for Stereo Sound Effects
- Uses the GIAY 3-8910
- Six Tone Generators
- Two Envelope Generators
- Two Noise Sources
- Four 8-Bit I/O Ports
- Up to Two Wait States

ALL SOUND EFFECTS ARE SOFTWARE CONTROLLED FOR AN ENDLESS VARIETY!

KIT $84.95

MOTHER BOARDS

THREE S-100 MOTHER BOARDS AVAILABLE;

- 3 Slots for Stand-Alone Applications
- 9 Slots for Small Systems
- 18 Slots for Full Size Systems
- Active Termination (9 & 18 Slot Boards Only)
- Extensive Ground Shield
- Breadboard Area

(Write for complete information)

The products shown here are just an example of the quality products and service **Ackerman Digital Systems** offers. **ADS** is an engineering company dedicated to providing the micro computer industry with solid quality products to meet a variety of needs.

All **ADS** products adhere to the **I.E.E.E. 696/S-100 Bus** standard, a bus noted for its widespread acceptance and versatility.

For catalog containing complete hardware and software information, write Ackerman Digital Systems, Inc., 110 N. York Rd., Elmhurst, IL 60126. TEL 312-530-8992

ads

ACKERMAN DIGITAL SYSTEMS, INC.

Implementing The L - Machine

The L - Machine is a stack-oriented machine used in the implementation of higher level languages. A similar machine, the PL/O was discussed in Wirth 1976. A popularized and modified version of this general model is the familiar P-Code Interpreter presented in the September 1978 *Byte* by Kin-man Chuan and Herbert Yuen.

One of the goals set in developing the L - Machine approach is to support the development of a higher level language. This is done in such a manner that a hardware machine independent compiler can be realized. By designing an intermediate level machine to compile code for, (called a target machine), a compiler which is independent of hardware machines is possible. An interpreter or emulator for the L - Machine is written for each hardware machine. In this manner, the compiler which is independent of hardware machine. In this manner, the compiler is said to be transportable. Only a host-dependent interpreter is required to bring up the compiler on another system. That host interpreter is what this article is primarily concerned with. More will be said about the high level language later.

L/1 Machine Structure

As stated before, the L - Machine is a stack-oriented device with few primitive operations. The basic structure uses 16-bit words and 16-bit integers. Only one in/out port is available and the device does arithmetic and control operations. Table I has a summary of the L/1

Table I: *L/1 Instructions.*

```
br addr                 branch to address (14 bit max
                        addr)
bnz addr                branch if not zero (14 bit max
                        addr)
lic constant            load immediate constant (16
                        bit max)
lad level,address       Load address at level
call level,address      Call address at level
nop                     No operation

            (opr)operations with the stack:

halt    Halt processor
add     add top two elements of stack result in top
        of stack
sub     subtract top element from next result in top
        of stack
mul     multiply top two element of stack
div     divide top element into next element
mod     divide top element into next, save remainder
        on stack
neg     form negative of top element (two's complement
        form)
not     if top of stack is zero, replace with 1 (true)
        otherwise put a 0 on the stack (false)
sete    if top two elements are equal, replace with
        a true (1) otherwise put a false (0) on the
        stack
setlt   if the top element is less than the next,
        replace with a 1 (true), otherwise put a 0
        on the stack (false)
setgt   if the top element is greater than the next,
        replace with a 1 (true), otherwise put a 0
        on the stack (false)
swap    swap the top two element of the stack
ret     return (restore data put on stack by call to
        address)
rav     replace address on stack with the value from
        that addr
sto     store top of stack in address of next to top
        element
inc     increment the stack pointer by the number of
        elements in the top of the stack
inb     input a byte from the port into the bottom of
        the stack
outb    output a byte from the bottom of the stack to
        the port
```

mnemonics and explanations. The in/out limitation can be overcome by the interface with the host interpreter. The interpreter discussed here can do both input and output to either the console or files.

Host Processor Background

The host for this effort is PolyMorphic's System 8813. System 8813 is an 8080A microprocessor-based S-100 bus microcomputer. The system PROMS reside from 0000H TO 0BFF, and the interrupt feature is used. One unusual feature (at least for microcomputers) of the PolyMorphic computer is the use of overlays in the operating system. Large (usually 2K, but up to 3K) subprograms can be called by name into an overlay area (starting at 2000H). The L/1 interpreter uses one of the system overlays, namely Gfid.OV, which is used for looking up files in the directory.

An area of memory set aside for the system use starts at 0C00H. The system labels for this area (among others) are made available in a library file called SYSTEM.SY. The assembler has access to this library, allowing system labels to be referenced directly within user programs. System labels of particular relevance in the L/1 interpreter include:

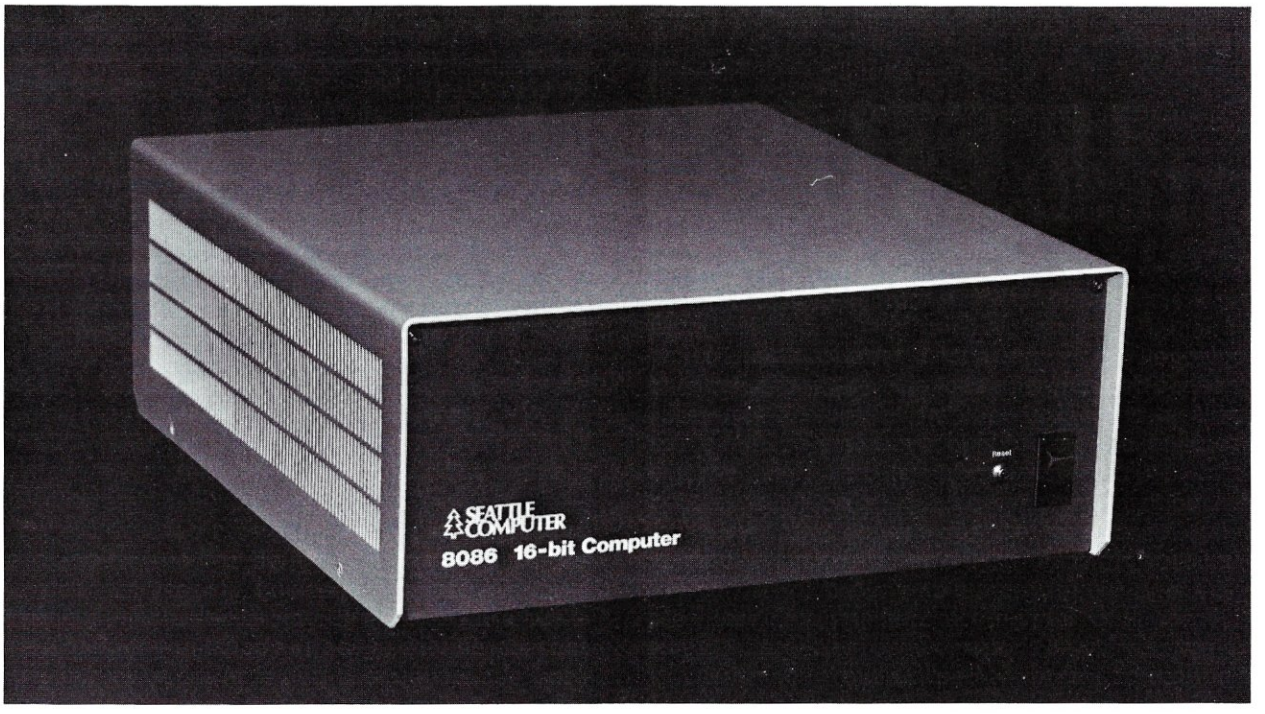
WHO	;Routine which, when called, inputs a character from the keyboard buffer (inputs to keyboard buffer are handled by interrupts).
WH1	;Routine which, when called, outputs a character to the screen.

Both WHO and WH1 are in system RAM and may be redirected by the system programmer.

Warm	;Routine which resets the stack pointer and warm starts the system.
Msg	;Routine which outputs text thru WH1 until a zero byte is encountered. The address of the start of the block of text is passed in register HL.
USER	;Address of start of memory available to user.
MEMTOP	;Storage for address of the highest usable memory byte.
Ret	;Address of a RET instruction in ROM.
Dio	;Routine which does Disk input and output.
Err	;Routine which handles errors passed in DE.
FILE	;Address of file descriptor buffer for use by Gfid.
Ovrto	;Routine for invoking an overlay (by name).
CMPTR	;Pointer to command buffer
Ioret	;Routine for returning from interupt level.
Gfid	;System overlay for looking up files.

Under the System 8813 operating system, an executable file is invoked simply by keying in the name of the file.

The system is here!



- 8 Mhz. 8086 CPU Set
- 128K Static RAM with 100 nsec. Chips
- Double-density Floppy Disk Controller
- 22-Slot Constant Voltage Powered Mainframe
- High-Performance 86-DOS Disk Operating System

$4,785.50 Fully Assembled, tested, ready to run. (Requires terminal and disk drives)

TO ORDER:
May be ordered through your local computer store or factory direct. Shipping paid by Seattle Computer on prepaid USA and Canadian orders. All boards fully assembled, tested, guaranteed one year. Mainframe guaranteed 90-days.

Software

We have the following Microsoft high-level languages running under 86-DOS.

• BASIC-86 Interpreter	$400
• BASIC-86 Compiler	$400
• Fortran-86	$600
• Pascal-86	$600
• Cobol-86	$900
• Macro-86 Assembler	$300

Check for new additions

SEATTLE COMPUTER

1114 Industry Dr. Seattle WA 98188

Information Hotline
206/575-1830

The operating system interprets the file extension to select among Basic, machine language or command file modes. By following the program name with another name on the same command line, the additional name is placed in the command file buffer and is available for the executing program. This option is implemented in the Little-Ada.L/1 interpreter. Before the interpreter loads the Little-Ada.L/1 object program, it asks if input is from a file. It also asks if output is to a file. Another version, which was implemented initially, but has temporarily been shelved, sets aside specific data flags at fixed relative locations in the object program for communication with the interpreter. If the input file options are selected, the interpreter will perform the necessary housekeeping to set up the input and output file handling routines. The coding is overhead in the interpreter rather than in the object programs.

L/1 Code Structure

The L/1 code may be viewed as the preliminary design for the microcode of an actual hardware processor (perhaps like the famed "Pascal Engine"). In viewing the code this way, a software interpreter can be designed which performs the functions of the L - Machine by examining each byte to determine its instructional value. Table I provided a list of the L/1 mnemonics and explanations. Implementing these mnemonics by assignments of numerical values requires consideration of the manner in which these values will be "decoded." Here the leading two bits were chosen to indicate branch instructions, allowing the six remaining bits to be part of the branch address. By using a second byte, a maximum of 2↑14 or 16K, word directly addressable program space is achieved with only two bytes for branch instructions. The 40H bit indicates a conditional branch called bnz (branch not zero). Whenever the leading bit is not zero, the next two bits represent the opcode class with 00 for stack operations, 01 for immediate data operations, 10 for address operations and 11 for no operation. See Table II for a detailed breakdown.

The segmented addressing method uses a structure which can be described as having blocks of data linked by two pointers. Each block of data is marked by a call to a subprocedure. Variable depth calls can be made during the dynamic execution of a program. A so-called "dynamic link" is created with each new block of data to mark the previous data location. This permits recursive techniques in the higher languages being supported. At the same time a second pointer, the so-called "static link," points toward the base data as defined in the program structure. This segmented address method is described in detail in Wirth. Calls in the L/1 - Machine create these dynamic and static links. The level difference is passed in the call instruction in order to indicate where the base data is located.

Also, in order to refer to a data location at other than the current level, a load address instruction (lad) is included. The lad instruction passes the level difference between the current data location and the data area referenced.

Table II: *Binary Assignments For The L/1 Codes.*

```
BINARY 0 or 1
A - Address      0 - Opcode       C - Constant
L - Level        X - Don't care

Mnemonic                BINARY

Br A            00AA AAAA AAAA AAAA
Bnz A           01AA AAAA AAAA AAAA
opr 0           1000 0000
lic C           1010 CCCC
lic C           1011 0CCC CCCC CCCC
lic C           1011 1XXX CCCC CCCC CCCC CCCC
lad L,A         1100 LLLL AAAA AAAA AAAA AAAA
call L,A        1101 LLLL AAAA AAAA AAAA AAAA
nop             111X XXXX

Xxxx xxxx               0 = Branch
xXXx xxxx               Opcode Class
                        00 = Stack operations
                        01 = Immediate data
                        10 = Address operator
                        11 = No operation
(See also TABLE IV)
```

Joint Development of Interpreter And Test Programs

In order to complete the development of the interpreter, test programs had to be available. This effort was supported by the concurrent development of a Cross-Assembler for the L/1 mnemonics. The method is really quite simple when a macro assembler is available. Once the assignments are made for the hexadecimal values of the L/1 codes, macros can be written which convert the mnemonic to its associated hex value. Table III lists the appropriate hex values. A macro can be described as a subroutine which substitutes into a textual structure. When a macro is invoked, it passes parameters to be substituted into the text as given in the macro definition. In the PolyMorphic Systems Macro 88 Assembler, the location for substituting the parameter is specified by the $ sign. $1 means put parameter number 1 here. $L means put the label on the macro invocation here.

Table III: *Hex Assignments For The L/1 Codes.*

```
00 00     - 3F FF        br
40 00     - 7F FF        bnz
80        - 8F           opr

80 halt         88 sete         90 inb          98 --
81 add          89 setlt        91 outb         99 --
82 sub          8A setgt        92 --           9A --
83 mul          8B swap         93 --           9B --
84 div          8C ret          94 --           9C --
85 mod          8D rav          95 --           9D --
86 neg          8E sto          96 --           9E --
87 not          8F inc          97 --           9F --

A0        - AF           lic C           AC
B0 00     - B7 FF        lic C           BC CC
B8 00 00  - BF FF FF     lic C           B8 CC CC
C0 00 00  - CF FF FF     lad L,A         CL AA AA
D0 00 00  - DF FF FF     call L,A        DL AA AA
E0        - FF           nop
```

If the macro is defined as:

```
Test    MACRO
#L      DB '#1',0
        ENDM
```

then invoking:

```
First Test GREEN
```

will produce:

```
First   DB 'GREEN',0
```

"Magic Typewriter -- the most useful program I ever bought . . ."

MAGIC TYPEWRITER is the reason why I bought a computer.

When Pat Lajko installed my North Star, I told him what features I wanted to use it for. He couldn't find software that would suit my needs, so he had to write it himself. After two years of development, testing, and redevelopment, MAGIC TYPEWRITER is finally being brought to the market.

I wanted more than just another word processing program. I wanted to use my computer to catalog my files, to organize the notes for a trilogy, to keep track of phone numbers and addresses, to catalog my library of books and records, and a thousand other uses. And I wanted all of these functions to be easy to access and easy to use.

MAGIC TYPEWRITER provides all that for me -- and more. It is a system that does not limit my capabilities, but expands them because it provides word-processing and data-base management all in one program. And its command structure is powerful and easy to learn.

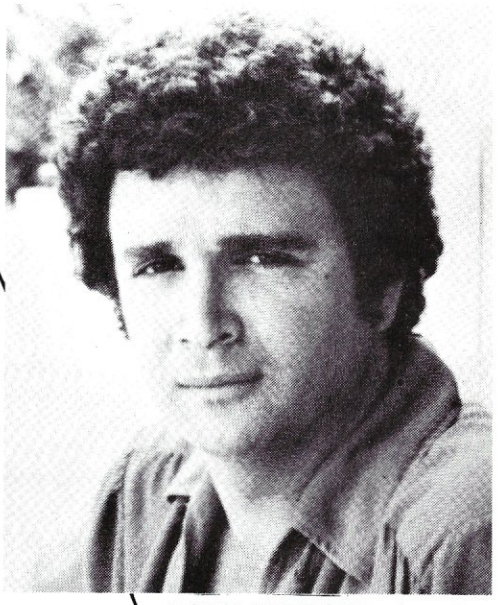
As a word processor, it is a pleasure to use because the system works with me instead of against. I particularly like its formatting functions. I can print out a whole novel with chapter and page numbers, page headings, full right and left justification and customized formatting where necessary. Because MAGIC TYPEWRITER allows for imbedded commands in the text, each file can contain its own instructions for output.

As a data-base, it is equally powerful. MAGIC TYPEWRITER can sort by record, by field, or by characters within a record; it can sort and reformat data on output so the same file can produce customized mailing labels or a sorted phone list. One of its nicest features is that it can retrieve data directly from a disk file in a matter of seconds without disturbing a file in memory. All of your data files are always accessible, even when you are working on something else.

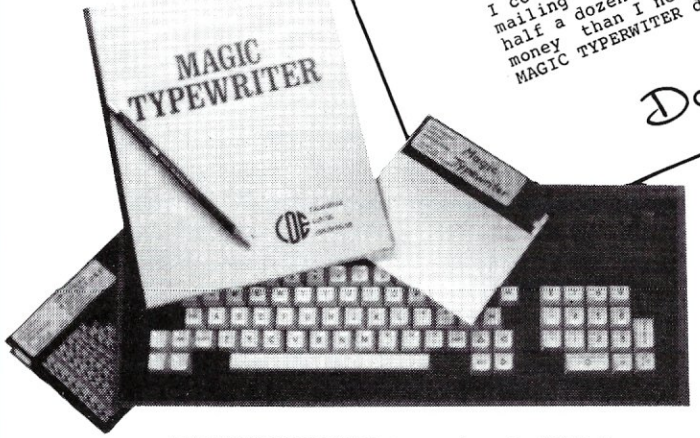
The only difficult part of the system is deciding which of its features I like best. MAGIC TYPEWRITER has too many virtues, but if I had to pick its strongest selling point, it is the fact that it is only $175. Some of the other popular word processing programs can cost up to twice as much. Some of the other popular data-base systems can also cost up to twice as much. MAGIC TYPEWRITER is for the home user who wants all the features of a powerful program without the powerful price. The amazing part of it is that MAGIC TYPEWRITER is extremely compact, requiring only 11K of program space!

I could have bought a lot of other software -- a fast-sort, a mailing list program, a text-formatter, a data-base manager, and half a dozen other special purpose programs -- and paid a lot more money than I needed to. Instead, I chose MAGIC TYPEWRITER -- and MAGIC TYPERWITER does it all!

David Gerrold



DAVID GERROLD is the author of "The Trouble With Tribbles," an episode of *Star Trek*. He has written almost a dozen novels, including *When Harlie Was One* and *The Man Who Folded Himself*. He has been nominated for the Hugo and Nebula Awards a total of seven times.



MAGIC TYPEWRITER is a trademark of C.D.E.

CALIFORNIA DIGITAL ENGINEERING
P.O. BOX 526 ★ HOLLYWOOD, CA 90028

EXPAND my system's capabilities with ***MAGIC TYPEWRITER.***

Please send _____ copies of MAGIC TYPEWRITER at $175 each. (Include $1.50 shipping. California orders include 6% tax.)

I have enclosed a payment of $_____

☐ Check payable to C.D.E.

☐ Visa ☐ MasterCard MC bank code_____

Exp. Date_____ Card No._____

Signature_____

NAME_____

ADDRESS_____

CITY_____

STATE_____ ZIP_____

PHONE (_____)_____ Ext._____

8" CP/M_____ 5" CP/M_____ Northstar DOS_____

Single Density _____ Double Density _____

Little-Ada cont'd...

Here is a complete set of macros which will do the job. (Of course, if your assembler doesn't distinguish between upper and lower case, you'll have to change some spellings to keep from conflicting with predefined opcodes.) These macros also show the hex codes for the L/1 Machine instructions. Although these macros appear in the 8080A context, they are generating the hexidecimal codes for L - Machine instructions. This is not the way L - Machine code is converted to 8080A instructions.

```
br      MACRO   ;parameter is address
#L      DB #1/256 AND 3FH,#1 MOD 256
        ;Strip off leading two bits
        ENDM

bnz     MACRO   ;parameter is address
#L      DB (#1 AND 3FH)+40H,#1 MOD 256
        ;Strip off leading two bits and
        ;set conditional bit
        ENDM

lic     MACRO   ;parameter is value of constant
        IF #1<10H       ;only need one byte
#L      DB (#1 AND 0FH) +0A0H

        ELSIF #1<400H   ;need two bytes
#L      DB ((#1/256) AND 7H) +0B0H, #1 MOD 256
        ELSE            ;need three bytes
#L      DB B8H, #1/256, #1 MOD 256
        ENDIF
        ENDM

opr     MACRO   ;parameter is value of
                ;operation
#L      DB (#1 AND 1FH) + 80H
        ENDM

lad     MACRO   ;First parameter is level
                ;Second parameter is address
#L      DB (#1 AND 0FH) + 0C0H, #2/256, #2 MOD 256
        ENDM

call    MACRO   ;First parameter is level
                ;Second parameter is address
#L      DB (#1 AND 0FH) + 0D0H, #2/256, #2 MOD 256
        ENDM

nop     MACRO   ;No parameters for nop
#L      DB 0E0H
        END M
```

It may be easier to write a separate macro for each of the operations with the stack. Example:

```
halt    MACRO
#L      DB 80H
        ENDM

add     MACRO
#L      DB 81H
        ENDM
```

and so on. Table VI contains a complete set of macros (without explanatory comments).

Once the L/1 macros are defined, programs in L/1 assembly language can be written. Assembling these programs produces a block of L/1 code for the L/1 interpreter to execute. For example, the following program can be used to test the L/1 interpreter for the lic, inb, br and outb instructions.

```
;       L/1 Test program

        ORG 0           ;Set program counter at 0
        IDNT 0,0        ;Load and start addresses
                        ;both zero for L/1 code

Start   lic 12          ;form feed
        outb            ;out port
        lic '?'         ;and a prompt
loop    outb            ;output to port
        inb             ;input from port
        br loop         ;and loop

        END
```

```
; This L/1 Test program expands to the following

        ORG 0           ;Set program counter at 0
        IDNT 0,0        ;Load and start addresses
                        ;both zero for L/1 code

Start   lic 12          ;form feed
        DB 0A0H+(12AND0FH)
0000 0C
        outb            ;out port
        DB 91H
0001 91
        lic '?'         ;and a prompt
        DB 0B0H+('?'/256 AND 7),'?' AND 0FFH
0002 0B3F
loop    outb            ;output to port
        DB 91H
0004 91
        inb             ;input from port
        DB 90H
0005 90
        br loop         ;and loop
        DB loop/256 AND 3FH,loop AND 256
0006 0004

        END
```

With the assurance that blocks of code can be obtained for the interpreter to execute, we can now proceed to the details of implementing the interpreter. Some of the most troublesome aspects of any program are what could be called housekeeping.

Host System Interface

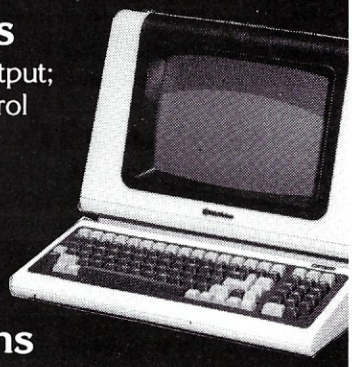
The PolyMorphic Systems System 8813 is an S-100 bus microcomputer. The System 8813 uses the Intel 8080A microprocesor CPU and three Shugart minifloppy (tm) SA-400 5 1/4" floppy disk drives. The PolyMorphic implementation uses the interrupt feature of the 8080A for various systems functions. Some of the interrupts are available for user level implementation. Unlike most microcomputer implementations, PolyMorphic Systems has provided a multi-level operating system with features similar to some mainframes and mini-computer systems. In particular, PolyMorphic Systems has implemented an overlay concept wherein parts of the operating system are "paged" in and out as required. A 2K area is used for the nine segments of the 18K operating system. The overlay area is also used to extend the size of the Basic interpreter by five segments and the text editor and the macro assembler by one segment each.

Single-step logic using a simulated front panel is implemented in the root ROMS. These consists of three 1K ROMS. PolyMorphic Systems has also incorporated many system programmer's utilities. One feature of the PolyMorphic Systems operating system allows for the recognition of file extensions. For example Exec "knows" that files ending in BS require loading and running BASIC.GO. I have modified Exec to recognize "L0" as requiring loading and running L0INTP.GO. The name of the file being evoked is communicated to the loaded program on the command buffer line. The location of the command pointer is stored in CMPTR. By adding this feature to the L0INTP.GO, and by the modification to Exec, L/1 programs can be invoked in two ways. The first is by keying in the program name only and the second is by keying in the interpreter name followed by the program name. Example: Prog.L0, or L0INTP.GO Prog.L0. Once the interpreter is running, it prompts for input source and output destination. In a purely interactive mode, input and output would be both via the consol. In a batch mode, input and output might both be via files.

Computer Systems for R & D

Data Acquisition & Control Systems

16 to 256 ch.; Programmable gain; Voltage or current output;
12, 14, or 16 bit; 30 to 125 KHz; Stepping motor control



8086 16 Bit Micro Systems

31 MByte Winchester Drives
256 KByte Memory Boards
8086 Software

Real Time Video Digitization Systems

Compatible with MP/M® and MP/M-86®
— for 8 bit systems one 256K board will support up to 7 users.

256 switch selectable I/O ports for memory bank addressing

Hardware and software bank select in 8K/32K segments

Parity detection with interrupt and LED indication

Industrial Quality

8/16 bit transfer for 8 & 16 bit systems

64K / 256K S-100 RAM

64K Boards each $595 4 for $2295

256K Boards each $2495 4 for $9495

Full DMA capability

24/20/16 bit address decoding

Meets IEEE 696 Specifications

Address selection on any 8K/32K boundary

Disable 8K/32K segments in hardware or software

Special circuitry allows error-free operation in even the noisiest S-100 systems

---

See our catalog for many other fine S-100 and Apple Boards including: — Analog to Digital Converters *(16 - 256 channels; 12, 14, or 16 bit accuracy; 30, 40, 100, or 125 KHz; programmable gain; timer/counters)* — Digital to Analog Converters *(12 bit accuracy, 3 microsecond conversion)* — 8086 CPU Board — I/O Boards — 16K Static RAM — Real-time Video Digitizer and Display.

®MP/M and MP/M-86 are registered trademarks of Digital Research, Inc.

TECMAR INC.

23600 Mercantile Rd. • Cleveland, OH 44122

TECMAR, INC.
(216) 464-7410

Little-Ada cont'd...

The first task of the interpreter is to identify and load the block of code which is named in the invocation. It uses system utilities to do this. What is important to the interpreter is that this block of code is loaded in a fixed location in memory. That location becomes the start of the relative program counter for the L/1 machine. Next, the interpreter sets up some initial stack data (the 0 level block mark) to simulate calling the main program. At this point the input and output selection choices are prompted and executed. All further actions will be as directed by the block of L/1 code.

In executing the L/1 code, the 8080A resources are allocated to perform certain functions of the L/1 machine. At the hardware level, the L/1 Machine consists of a stack pointer register, an instruction register, a 16-bit register for data transfer to and from the stack, a program counter, an 8-bit register and a base register.

Those L/1 functions are allocated as follows:

8080A	**L/1 Machine**
register BC	PC Program counter
register HL	SP Stack pointer
register DE	non-addressable 16-bit accumulator for use with stack operations
register A	non-addressable 8-bit bus data transfer register
register SP	not used directly
memory area	Base register

The L/1 machine implementation allocated the system memory and 8080A registers as follows: 8080A registers were used to simulate the L - Machine program counter and the L - Machine stack pointer.

The 8080A register BC serves as the L - machine program counter. The fetch routine points at the next L/1 instruction, returns the current L/1 instruction in the 8080A Accumulator, as well as saving a copy in the location "Inst" (target machine instruction register).

The 8080A register DE serves a general purpose 16-bit target machine accumulator (not explicitly addressable in the current version of the L - Machine). This register is used in the transfer of data to and from the stack in stack operations.

The 8080A register HL serves as the L/1 Machine stack pointer. In the current version, the L/1 Machine stack pointer is implemented in top down (8080) fashion and points at the next available location. The L/1 machine stack pointer is initially set at top of memory by the interpreter during the initialization routines. Implicit target machine "Push" and "Pop" operations are implemented which PUSH and POP the target machine accumulater (the 8080A register DE) to and from the top of the target machine stack. Interpretatin of each L/1 byte begins with a fetch cycle. The fetch code is repeated here to illustrate the method.

```
Fetch   LDAX B          ;Get current code byte
        INX B           ;and increment pointer
        STA Inst        ;save a copy of the
                        ;instruction
        ORA A           ;clear any leftover flags
        RET             ;return with byte in
                        ;accumulator also
```

The contents of BC must be preserved from L/1 instruction to L/1 instruction.

Two other routines not accessible to the L/1 machine, but which are necessary for stack operations, are "Push" and "Pop" operations from the top of the stack to the 16-bit accumulator (DE). The character "t" represents the relative stack position increasing by one for each 16-bit word pushed, whereas the actual location in memory decreases by two for each 16-bit word pushed.

```
Push    MOV M,E         ;DE to S(t)
        DCX H           ;t+1 to HL
        MOV M,D
        DCX H
        RET

Pop     INX H           ;S(t) to DE
        MOV D,M         ;t-1 to HL
        INX H
        MOV E,M
        RET
```

Once the interpreter "Fetches" the byte, it must interpret it to determine what action to take. It is easier to see the necessary relations for the interpreter to discover if the logic of the L/1 code assignments is laid out in a tree. Table IV lays out such a tree, assuming we start by testing the high order bit, bit 7.

Each of the L/1 code instructions has a subroutine in 8080A code which is eventually branched to during the interpretation of each byte of L/1 code. The complete source list of the implementation will be published in an upcoming issue of *Microsystems*.

Table IV: *Binary Relations Tree (For Interpreter)*

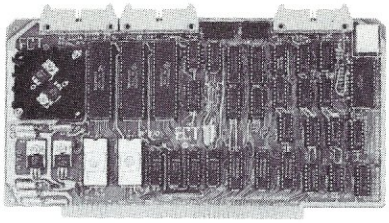
```
first       test bit 7

bit 7 = 0       branch or branch not zero
.    .          then test bit 6
.    .
.    bit 6 = 0  ..................... branch always
.    bit 6 = 1  ..................... branch not zero
.
bit 7 = 1       then test bit 6
     .
     bit 6 = 0       then test bit 5
     .    .
     .    bit 5 = 0  ................ opr with stack
     .    .          next 5 bits (0-4) for op-code
     .    .          (use computed jump table)
     .    .
     .    bit 5 = 1       lic group - test bit 4
     .         .
     .         bit 4 = 0  ............ lic 4 bits
     .         bit 4 = 1       then test bit 3
     .              .
     .              bit 3 = 0  ....... lic 11 bits
     .              bit 3 = 1  ....... lic 16 bits
     .
     bit 6 = 1    then test bit 5
          .
          bit 5 = 0       then test bit 4
          .    .
          .    bit 4 = 0  ............ lad
          .    bit 4 = 1  ............ call
          .
          bit 5 = 1  ................. nop
```

Use of L/1 Machine in Implementing Higher Languages

As mentioned earlier, the original development of the Little-Ada compiler was done on the ODU Dec 10 university computer. A Pascal version of the compiler was used in that effort. A parallel version of the Little-Ada compiler was written in Little-Ada—that program was compiled using the Pascal version of the compiler. The outputs of these compilers are both L/1 code and require an interpreter or emulator on the Dec 10.

CUSTOM PRODUCTS

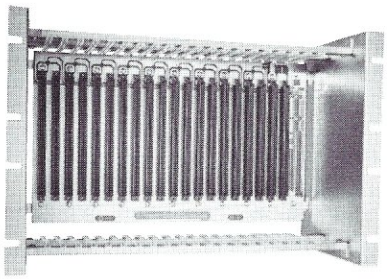
DESIGN • LAYOUT MANUFACTURING



R ²I/O

ROM/RAM & I/O

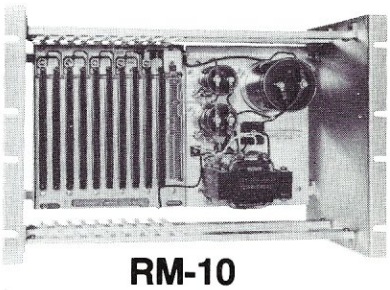
S-100 PRODUCTS



ECT-100-F

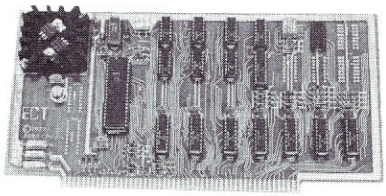
RACKMOUNT CARD CAGES

ECT™



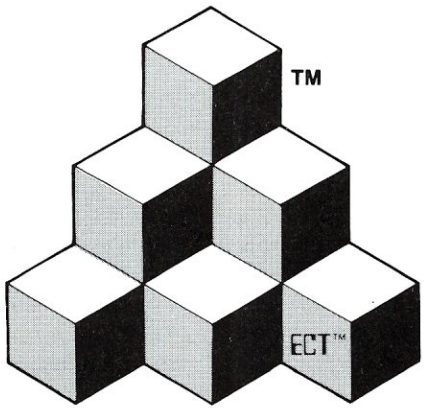
RM-10

CARD CAGE & POWER SUPPLY

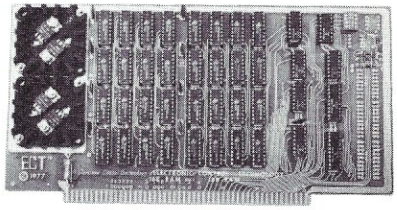


8080 CPU

CENTRAL PROCESSING UNITS

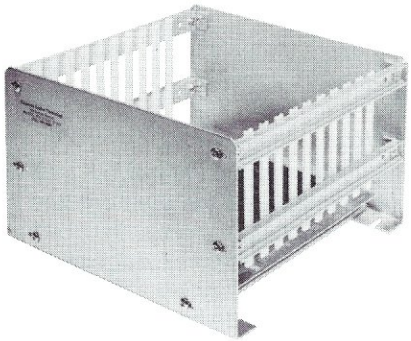


™



16K RAM

FULLY STATIC MEMORY

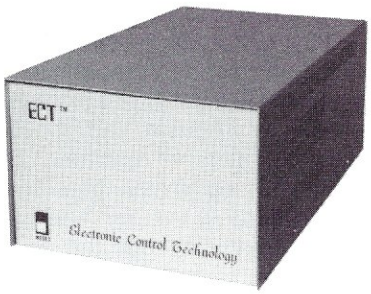


CCMB-10-F MIN

6,10 OR 20 SLOT CARD CAGES

BUILDING BLOCKS FOR MICROCOMPUTER SYSTEMS, DEDICATED CONTROLLERS AND TEST EQUIPMENT

CARD CAGES, POWER SUPPLIES MAINFRAMES, CPU'S, MEMORY I/O, OEM VARIATIONS



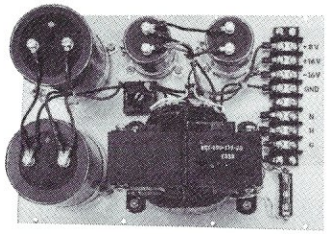
TT-10

TABLE TOP MAINFRAMES

ELECTRONIC CONTROL TECHNOLOGY, INC.

763 Ramsey Ave., Hillside, NJ 07205 (201) 686-8080

SPECIALIZING IN QUALITY MICRO COMPUTER HARDWARE



PS-30 A

POWER SUPPLIES

MULTIBUS® PRODUCTS

MULTIBUS IS A TRADEMARK OF INTEL CORP.

Concurrently, a Pascal interpreter for L/1 code was written on the Dec 10 to execute the compiled L/1 code. The L/1 code can be interpreted by another L/1 Machine emulator, hence the L/1 programs are transportable. Subsequently, the compiled Little-Ada version was used to re-compile itself (with the Pascal interpreter). The L/1 code was produced in an ASCII text byte format and transferred from the Dec 10 to the PolyMorphic 8813 using a standard 300 baud modem. Once the ASCII text L/1 code (the compiler) was converted back to a binary byte format, it was executed on the Poly using this interpreter. It was well able to compile Little-Ada programs to L/1 code. And, when these compiled programs were run on the Poly interpreter, they worked!

Getting the L/1 code ASCII byte version of the compiler correctly transferred required dealing with data transmission noise. Occasionally a character would be changed or dropped in the transmission. Data transmission errors had to be taken into account to insure a complete and correct copy of the compiler was transferred. Redundancy was used to detect and correct errors. A second copy of the compiler was transferred in a separate transmission. Using a file compare program, the differences were examined to find and correct errors. The ASCII text byte format greatly enchanced detection and correction of these errors.

In a nutshell, a block of L/1 Code was transferred from the Dec 10 to my PolyMorphic 8813 and interpreted by my native interpreter. The resulting combination was able to process text files of Little-Ada into code as well as listings.

Ada Background

Since 1975 the High Order Language Working Group (HOLWG) has been preparing and revising standard specifications for a high order language to be used in DOD embedded systems. In 1979, after several stages of revision and competitive procurement, a preliminary reference manual was released. The language was named after Ada Agusta, Lady Lovelace, the daughter of Lord Byron, but more relevantly, Babbage's programmer. She may be said to have been the first programmer.

Already, a text for Ada has been published (Wegner 1980), even though the language was still 'preliminary' and there are not yet any compilers! It was only in September of 1980 that the Defense Advanced Research Projects Agency hosted a formal debut of Ada. There are several compilers under development, both large and small. The compiler being developed at ODU could be called one of the "small" ones. The method of implementation is of particulr relevance to this article.

As discussed earlier, this L - Machine is used in the implementation of Little-Ada, which is a limited subset of the Ada language. In implementing a recursive descent compiler for this language, certain functions are performed by the compiler during compile time only, and are not in evidence at run time. A symbol table is constructed by the compiler at compile time and the data in this table is used to insure that the block structure is not violated. The symbol table also keeps track of the level of the variables used, as well as their addresses.

Compilers can be described in terms of three functions performed. Each compiler must check the input text (program) to ensure that the statements are syntactically correct. A second function of the compiler is to keep track of the symbols. These symbols may be variables, other data structure names, program block names such as procedure calls or functions (two types of subprograms), or labels for the program locations. The third function of a compiler is to generate code, code for the target machine (in this case, the L/1 Machine).

All variables and other objects must be declared before use. The symbol table keeps track of this process. At each declaration, the name is added to the table. In subsequent references, the compiler looks up the name in the symbol table and picks out required data. Many parts of the error checking process are part of this symbol table process during compile time. No object code is needed for these checks because the logic has been checked during compile time. The level of the program and the names of the object are present in the symbol table at compile time, but not at run time.

The L - Machine is a stack processor of the last in, first out kind. All operations are done on the top of the stack. Anyone who has used a Reverse Polish Notation (RPN) or "post-fix" notation calculator should be familiar with the method for using the stack. Fortunately, there is a direct relation between algebraic or "in-fix" notation and post-fix notation. The arguments, or values, are in the same order, but the location of the operations is changed. Example: Add A to B; (which just happens to be in the operation Argument-Argument or pre-fix notation format).

IN-FIX NOTATION	POST FIX NOTATION
A + B	AB+

With stack operations, the post-fix notation corresponds directly with the stack.

A B ADD works out as:
Push A onto the stack.
Push B onto the stack.
Add the top two elements.

Suppose we illustrate the assignment statement: A :=B + C; The parser would recognize A as the element and expect an assignment statement, so it looks for ":=". Having found ":=", the parser then looks for arguments. Finding "A", the interpreter would push the value of A onto the stack. Next the parser would be programmed to expect an operation and would find +. Having recognized "+", the parser saves a code for the operation found, and then will again look for a variable onto the stack (causing the value of "B" to be pushed onto the stack). When this subprocedure returns, the interpreter now performs the saved operation, "+".

To describe this process more abstractly, we can say that an operation and its arguments were recognized in the in-fix notation format or sequence. While an interpreter may do these operations directly as they are identified, a compiler must do the operations indirectly by generating code for later execution. The compiler does work similarly and is illustrated later in this article. A generalized scheme is used by which two values connected by an operation are recognized. The portion of the routine which recognizes a value allows it to be of the form sub-value, sub-operation, sub-value; this allows using the recognition routine recursively. This method is called "recursive

descent." Essentially, the recursive descent method allows the conversion from in-fix to pre-fix notation with relative simplicity. We expect a value, operation, and another value, so we write a procedure to push the first value onto the stack, recognize the operation and save it; recognize and push a second value onto the stack; and finally, perform that operation we recognized and saved, leaving the result on the stack. By allowing the value to be the result of other such processes, we can have recursive techniques to arbitrary depths. Example:

```
[(value,operation,value),operation,value]
```

The basic procedure is only slightly complicated by the compiler's requirement to generate code rather than perform the operation. A more detailed discussion of the process can be found in Wirth.

To illustrate the code required for the compiler output, we use a set of examples beginning with the simple assignment statement LENGTH := 10; and continuing to more structured examples. Here, we must deal with the location where LENGTH is stored, rather than its actual value. We must set the ADDRESS of LENGTH from the symbol table and put this on the stack. The code is "lad L,Addr" where Addr is the address of LENGTH from the symbol table. The "level,"L, is the difference in level from the current level in the program being compiled and the level where LENGTH was declared. This procedure is accomplished by two means. First, when a symbol is declared, its absolute level is put into the symbol table. Then, when we want to refer to it, we subtract that level from the current absolute level. The absolute level is computed by going up one for each call and down for each ret. To get L, we simply subtract Level(LENGTH) from Level(Current) or L := Level(Current)-Level(LENGTH). Since 10 is immediate data, we can put its value on the stack directly. All that remains is to stow the value 10 in the address of LENGTH, which the "sto' instruction does.

In parsing the input text, the compiler would have already found LENGTH declared and given it an address and a level. The kind of L/1 code generated for this execution is:

```
lad Level(Current)-Level(LENGTH),Addr(LENGTH)
lic 10
sto
```

As the next example of this procedure, consider the assignment statement: LENGTH := WIDTH; Here we must get the value of WIDTH from its address before we can stow it in LENGTH. That requires loading the address of WIDTH onto the stack and then getting its value. The rav instruction does this. The "lic 10" instruction must be replaced by the lad and rav combination. The kind of L/1 code generated for this is:

```
lad Level(current)-Level(LENGTH),Addr(LENGTH)
lad Level(current)-Level(WIDTH),Addr(WIDTH)
rav
sto
```

Now, to graduate this example one more step, consider the assignment statement SIZE := WIDTH + LENGTH;

```
lad Level(current)-Level(SIZE),Addr(SIZE)

lad Level(current)-Level(WIDTH),Addr(WIDTH)

        rav     ;Puts value of WIDTH onto stack
        lad Level(current)-Level(LENGTH),Addr(LENGTH)
        rav     ;Puts value of LENGTH onto stack
        add     ;Adds top two (LENGTH and WIDTH)
        sto     ;Stores sum in Addr of SIZE
```

As two final examples, try the following:

```
AREA := BASE * HEIGHT / 2; -- Area of triangle
        lad Level(current)-Level(AREA),Addr(AREA)
        lad Level(current)-Level(BASE),Addr(BASE)
        rav
        lad Level(current)-Level(HEIGHT),Addr(HEIGHT)
        rav
        mul
        lic 2
        div
        sto
S := S0 +V0*T + A*T*T/2; -- Formula for distance
        lad Level(Current)-Level(S),Addr(S)
        lad Level(Current)-Level(S0),Addr(S0)
        rav
        lad Level(Current)-Level(V0),Addr(V0)
        rav
        lad Level(Current)-Level(T),Addr(T)
        rav
        mul
        lad Level(Current)-Level(A),Addr(A)
        rav
        lad Level(Current)-Level(T),Addr(T)
        rav
        mul
        lad Level(Current)-Level(T),Addr(T)
        rav
        mul
        lic
        2
        div
        add
        add
        sto
```

The compiler is normally not written to recognize these examples directly, but to recognize the general structure (value 1, operation, value 2).

The source list of the compiler has not been made available, but the L/1 machine structure is fully examined here. Before proceeding with the details of the relation between the L/1 machine and Little-Ada, a brief decription of Little-Ada is appropriate. Syntax diagrams are provided for the structure of the L/1 implementation of Little-Ada. ■

Software Product Review

OS-1 — A Diamond In The Rough

by David Fiedler

OS-1 is a disk operating system designed to run on minimally-configured Z-80 systems. To the user, it appears very similar to the popular UNIX system, incorporating many of its features. Since OS-1 isn't a multi-user or multi-tasking system, it does not need special hardware for bank-switching as do Cromix and uNIX (which are other Z-80 based UNIX look-alikes). A further attraction is the inclusion of a CP/M adapter, so that CP/M programs can be run under OS-1 control. OS-1 costs $249 (including one year's "software support") and is available from Software Labs, 735 Loma Verde, Palo Alto, CA 94303.

Bringing Up The System

If you have a CP/M 1.4 system, or can emulate one (apparently only the BIOS routines are used, and they expect strict conformance to standard single-density), bringing up OS-1 is simplicity itself. Just run the OS32 or OS48 program under CP/M (depending on your available memory), insert the eight inch OS-1 disk, and press any key. The OS-1 system doesn't care if you are running exactly a 32K or 48K CP/M system, as long as you have enough memory; however, any memory over 48K will not be used. No memory map is provided with OS-1, so it is not clear where everything goes. It just runs merrily along.

Three disks are included, one of which is in standard CP/M format, and contains the special OS boot programs mentioned above, the source code for certain parts of the OS-1 system, some library files and a loader. The other two disks contain a runnable OS-1 system and many utilities.

David Fiedler, Box 33, East Hanover, NJ 07936.

Understanding The System

Since OS-1 was modeled after UNIX, it comes as no surprise that the documentation also follows the UNIX format. The OS-1 User's Guide is 3/4 inch thick (the pages are not numbered in the normal fashion, again UNIX style) and describes every command, system call, and system concept, along with any particular files involved. It is an extremely impressive piece of work, and is less ambiguous than the corresponding UNIX manual in several places, especially in terms of the examples cited. Certainly it ranks as one of the most complete and well-organized software documents in the microcomputer industry. However, I do have some complaints. In consciously sticking to the UNIX guidelines, the authors sometimes get too attached to the UNIX terminology (e.g., talking about "dump tapes"), are unnecessarily terse, and can fall into the trap of giving overly general examples. Also, like UNIX, pure luck is necessary to find out certain minor, but important details—like at what address to assemble your files. (The answer is found in the section about the loader.)

An "Introduction to OS-1" was enclosed along with the OS-1 User's Guide. Although only 62 pages long, this manual proved an excellent beginning towards learning the essential concepts of OS-1. In fact, it is necessary to read this and the expository material in the User's Guide before you can make intelligent use of OS-1. This is difficult to do, as it is tempting to play around with the system before exercising your brain. I managed to survive this crisis, but only because I'm familiar with UNIX. I feel a very simple hand-holding guide is necessary for any system this powerful—something that would literally show you how to log in, tell you what to type in order to get certain results, and explain why you got the results. For

Go FORTH AND Conquer

ENHANCED

FIG* FORTH

for **CP/M**† with

FLOATING POINT

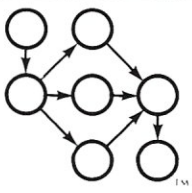
VOCABULARY OPTION

- **FORTH** by Timin Engineering Release 2 with manual (includes full 8080/Z-80 assembler, fast disk I/O [.17 sec. per 1K block on 8'' disk], editor with 20 commands) $95
- **FORTH** user manual & tutorial (price credits towards disk purchase) $20
- **FORTH** by Timin Engineering Release 3 (includes CP/M utility and visual editor plus other enhancements) $235
- **FORTH** — CP/M file conversion utility (Supplied in FORTH source code) $50
- Romable **FORTH** by Timin Engineering (similar to Release 3) $350

Any of the above with MPM interface $40 extra
Any of the above with **Floating Point** vocabulary $100 extra

- Visual (screen) editor with cursor control (Supplied in FORTH source code) $75
- Disk of useful **FORTH** screens (Development tools, programming examples, etc...) $75
- **FORTH** Information booklet $2 cash or stamps
- Custom FORTH programming available

TECHNICAL HOTLINE, answered only by software technician (714) 455-9041



TIMIN ENGINEERING COMPANY
9575 GENESEE AVENUE • DEPT. E-2 •
SAN DIEGO • CALIFORNIA 92121 •

TELEPHONE (714) 455-9008

VISA

master charge THE INTERBANK CARD

Software price is for single licensee and includes User Manual and shipping except C.O.D. Distribution is on 8" single density disk. Other disk formats, add $15. Items shipped within 48 hours for C.O.D., credit cards, certified check, or money order. California residents please add 6% tax.

*FORTH Interest Group †® Digital Research Corp.

many people, it is not enough to describe what the commands do; it helps to be shown.

When OS-1 comes up, you are the "super-user" in the WIZARDS group (no, there is no ADVENTURE game on OS-1—yet). The super-user has all sorts of special access rights and powers, and can get into big trouble very easily. In particular, it is the super-user who has the capability to authorize new users, delete and edit special system files, and bypass virtually all file permissions. However, this power is very easily abused by a new user. My suggestion to Software Labs regarding this is to change things so that the beginner logs in as a normal user. You can still become the super-user at any time.

Backing Up The System

The disks I received were quite scratched on the recording surface, although the envelopes and outer jackets were in perfect condition. I suspect Software Labs is using some very old disk drives. I was able to read all three disks, although one was definitely close to being unsalvageable. I have found that banging the side of a disk drive while it is having trouble with particular disk will enable it to recover—sometimes.

Due to some peculiarities in my system which prevent me from using a "normal" single-density CP/M disk as a system disk, I was originally forced to pretend, as far as OS-1 was concerned, that I only had one disk drive. This also meant that I was unable to back up the distribution diskettes, since the two OS-1 disks are not in normal CP/M format. A sheet provided with the system advises the user to use a "raw disk utility" rather than CP/M's PIP to perform the backup, but none was provided, and I didn't have one that would let me back up on one drive. When I got the system up, I tried to use the OS-1 backup program. As a result, the system eventually crashed, leaving me with no working disks.

If you have a CP/M 1.4 system, or can emulate one, bringing up OS-1 is simplicity itself.

At this point, I was forced to call Software Labs. Without revealing my secret identity as a klutzy software reviewer for *Microsystems*, I explained what happened. The consensus was that I has run across a known (known to them, anyway) bug in the cache routine that controls the updating of the disk. This made some sense in light of the particular way that the system crashed, and they agreed to fix up my disks and make backups for me in case I had more trouble. They suggested that I send the disks back, and they would return them as soon as possible, UPS Blue Label, C.O.D. To prevent even further delay, I sent $6 to prepay the postage. Then I waited for the swift return of the disks.

And I Waited...

Five weeks later, I received the disks, the backups they made, and a letter which implied that it had not been the cache routine bug that had laid my disks low. They did not mention whether or not the bug had been fixed. Unfortunately, (this is not completely clear in the documentation) the backup program cannot be used to back up only one disk drive, even though the program prompts lead you to believe this can be done. Therefore, it isn't possible to back up OS-1 on a single-drive system with the programs provided. It was disheartening, also, to find out it had taken five weeks to just copy a few disks.

In fact, it is not possible to run OS-1 on an unmodified CP/M 2.0 or later, which includes MP/M also. The reason lies in the sector blocking/deblocking algorithms present in these new versions, which permit certain disk write operations to take place without actually writing to the disk. What must be done involves patching your BIOS to fool it into thinking that all disks writes are TYPE 1 (write to directory sector—see page 34 of your "CP/M 2.0 Alteration Guide"). If this means little or nothing to you, there will be much grief before you ever get OS-1 running! After discovering this remedy, however, I was able to run OS-1 with both disks, and without crashes.

Running The System

OS-1 generally runs as advertised. After using CP/M on your machine, it is nice to see a system that "learns" what you are doing, buffering your most recently used data in RAM to avoid extra disk accesses. It is possible to run several commands sequentially in this way, and (of course) the response time improves greatly when the system knows about the buffering. Some of the programs seem to take longer to run than their size would indicate; this is probably because they have to traverse OS-1's tree-like file structure. It is hard to reconcile certain utilities' size with their functions—for instance, PIP is only 3.5K, while STTY (which simply sets up certain modes for the console terminal) takes up 13K. I suspect that some of the larger programs were written in C or Fortran, rather than assembler.

The system indeed looks like UNIX. The chief differences are in some of the special characters used:

	UNIX	OS-1
Prompt	$	-=>
Directory	/	:
Argument or Switch	-	/

So a typical command line, asking for a "long" or "detailed" directory listing would look like this under UNIX:

```
$ ls -l /usr/src/cmd
```

while the equivalent command for OS-1 would be:

```
-=> list :usr:src:cmd: /d
```

Although Software Labs feels it adds identity to OS-1 by using this syntax, I personally think it could avoid confusion if they went along with the UNIX conventions, perhaps by making the characters in question could be made user-modifiable. In any case, redirection of input and output to any device or file is supported, as is pipelining between programs (done by using temporary files). These features, along with the hierarchical file structure, are among the most useful and desirable in an operating system.

There are 68 commands associated with OS-1, comprised of 28 intrinsic commands built into the system (as opposed to five in CP/M 1.4), and 56 executable program files. Several reasons exist for this disparity:

1. Certain programs (the commands line interpreter, or "shell"; the debugger; and the CP/M adapter) exist as three separate files, and the appropriate one is executed depending on available memory. This decision, however, must be made by "linking" the name of the file you wish to execute to the correct size program for your system. As such, this is not even mentioned in the manual, and can leave you with the impression that you have much less memory available than is actually the case—especially with the CP/M adapter. It would be better if these files were relocatable, so you would only need one. Also, "login" appears both as an executable file and as an intrinsic command, implying that the file portion is needed to execute.

2. Some programs listed in the manual are not included on the disks (i.e., find and lead (a UNIX-type editor)). These programs I would classify as "missing."

3. Some programs included on the disks are not listed in the manual. Aget, aput, bget and bput seem to be command files (similar to .SUB files on CP/M) for transferring data between CP/M and OS-1. If this is truly their function, a little explanation would be helpful. It is not clear whether they were ever meant to be included, as they reference directories that do not exist on the distribution disks. Also, a bit of imagination is necessary to discover that the documentation for "sdcheck" (small directory checker?) is included with "dcheck."

Getting Annoyed With The System

The trouble with the large number of utilities is that it's difficult to exercise them properly. Both disks are rather full, and you quickly run out of i-nodes (somewhat akin to directory space in CP/M) when attempting to add users, test files, or the like. While the "init" program is all that you need to create a new usable disk for OS-1, I couldn't discover how to make a new system disk that would boot. I was unable, therefore, to put together a set of disks that had enough room, and utilities, to do what might be considered "useful work." This was made more difficult by the condition of the utilities themselves, as well as the warning Software Labs includes, "files currently included will not be compatible with later equivalent editions." In fact, the "loader will not load files produced by itself." At least they warn you!

In the process of looking through the various files, printing out listings, logging on, and being super-user, I noticed funny things happening. An entry for a new user would be created; it could not be deleted. The rmuser (remove user) program for doing this would not work properly, so I finally gave up and edited the user file directly. The system, however, prevented me from writing on this file, even though I was the super-user and supposedly could bypass all file permissions. So I tried to test the disk—perhaps there was a bad sector. Running the "badblk" program, with the /1 switch set as shown in the manual, would list out the bad areas of the disk. Instead I was told I had a bad switch. This same sort of problem cropped up in various other utilities, and I was continually frustrated when things didn't work as they were shown in the manual. Sometimes the switches in a given program would respond to upper case, even when lower case was shown in the manual.

Certainly the many prompts are helpful in telling you where you are going wrong (with the possible exception of "That command did nothing"). But the problems faced in trying to work with OS-1 have, for me, outweighed the nice features. The system seems to be carefully thought out, yet the implementation is lacking.

In a phone conversation with John White, president of Software Labs, I was asked to note in the review that OS-1 was purchased from the original owners (a company called Electrolabs, now out of business) and is currently being supported completely by Software Labs. I was told that Software Labs was writing their own C compiler, that OS-2 (multi-tasking) was coming along, and that they were working on OS-3 for 16-bit machines. Also, the new update for OS-1 was on schedule, and in fact he had the first draft on his desk.

Over three months later, I have not heard anything about the promised update, the compiler, or even the cache bug fix. I believe the people at Software Labs have the capacity to develop fine products, but it sounds to me as if they are moving ahead before they finish their past commitments. OS-1 has not been finished yet, and I can't recommend it until it has been. ■

Trademarks: OS-1 is a trademark of Software Labs. UNIX is a trademark of Bell Laboratories. Cromix is a trademark of Cromemco. uNix is a trademark of Morrow Designs.

The BDS C Compiler

by David Fiedler

In the past few years, there has been a great deal of interest in the C programming language. Proponents claim C is everything Pascal should have been, critics sneer that its syntax is too cryptic (and sometimes call it "C" with quotation marks, as if it weren't quite legitimate), and software houses write compilers for it.

C will always be associated with the UNIX operating system, because one of UNIX's special features is that it is easily maintainable. It is easily maintainable because it is written in C. A slightly less well known fact is that C was developed to be a suitable language in which to write UNIX. C is, therefore, a high-level language capable of the efficiency necessary for writing serious system program. It does not have either the strong type-checking or the addressing constraints of Pascal. Programs written in C are highly portable between machines having C compilers. All this makes C just the kind of language useful to serious microcomputer users—that's how C ended up running under CP/M.

There are three C compilers that have been generally available to the CP/M user:

Small-C	$ 15 by Ron Cain
BDS C	$150 by Leor Zolman, BD Software
Whitesmiths C	$600 by Whitesmiths, Ltd.

These compilers represent a wide range of price and capability. This software review deals mainly with the "mid-priced" model, the BD Software C compiler (BDS C), comparing it to the other two when appropriate.

In almost every field of endeavor where a purchase is involved, the question, "Which should I get?" comes up. The answer, as usual, is "What do you want to do with it?" If you just like to collect compilers, or enjoy fooling with them, then Small-C is a bargain. The Small-C compiler is completely written in Small-C, and the source code for both the compiler and the run-time library is included for $15. If you absolutely must have a compiler with the full language capabilities of Version 7 UNIX, then only the Whitesmiths compiler will satisfy you.

However, if you are looking for an affordable compiler that is easy to use, produces respectable code, is usable for systems programming, and is supported by an active users' group, you should take a serious look at BDS C.

The "full language" clause above deserves some explanation. Unlike other popular languages such as Basic, there is a published standard for C (Appendix A of *The C Programming Language*, by Kernighan and Ritchie, Prentice-Hall, 1978). So it is easy to determine the extent to which a given C compiler supports the language. But, similar to Pascal, the language definition does not mention I/O facilities (though a standard I/O library is defined by Dennis Ritchie in the UNIX documentation). Since most C programs need to do some kind of I/O, portability suffers if the supplied I/O library does not follow the standard.

If you absolutely must have a compiler with the full language capabilities of Version 7 UNIX, then only the Whitesmiths compiler will satisfy you.

What does all this mean? Let's take the archetypal C program, the first one mentioned in the book:

```
main()
{
    printf("Hello, world\n");
}
```

Of the three compilers studied, this program will only compile "as is" on BDS C, due to differences in the supplied I/O libraries. Similarly, differences in names and calling sequences of other library functions prevent typical (i.e. copied out of Kernighan and Ritchie) C programs from being compiled on Whitesmiths or Small-C without a certain amount of editing and rewriting. While BDS C is not totally free from these restrictions, it is more amenable to running programs straight from the book.

David Fielder, InfoPro Systems, P.O. Box 33, East Hanover, NJ 07936.

This is extremely important because it is impossible to learn C without writing and running programs in it, and it helps to start with the examples in the book.

Documentation

The Kernighan and Ritchie book is best for introduction and reference, but still does not serve all tutorial needs. So the compiler manual will become your constant companion.

BDS C comes with a 72 page user's manual written in a clear, personal style. There is no index or table of contents, but the manual is separated into logical sections. What the manual lacks in formality, it makes up for in hints, short tutorials, program examples, and pithy comments on software in general. Leor went to the trouble of getting the entire manual typeset in genuine Bell Labs style, which helps readability greatly. You are expected to be familiar with CP/M, C, and things like hex and bytes—but this knowledge is not necessary to use the compiler.

Small-C comes with an eleven-page user's guide, which is more than adequate. It clearly explains how the function calls work, what the compiler's limitations are, and how to work the compiler itself. It's all you really need to sit down and get started—one part walks you through a compile-and-load session. No index here either, it's really not necessary.

Whitesmiths C comes with two 1/4 inch thick reference manuals. It's hard to describe the style of writing; the word "terse" does not quite do it justice. The authors are fond of using their own definitions to help explain how the functions are supposed to work, a habit which can confuse the reader.

The explanations of almost every subject are included, but they are often hard to find, extremely cryptic once found, and sometimes inconsistent. An index is absolutely necessary, but only a totally inadequate (I feel page numbers would help greatly) table of contents is included. These manuals are completely lacking in readability, and detract from the product as a whole.

General Comments on BDS C

One reason for BDS C's speed of compilation is that, unlike the other C compilers, it does not produce an assembly code directly. This could be a factor if you must trace through program execution; you won't have a listing to guide you. Generally, though, the code generated is straightforward, and can be followed with DDT. If you have Digital Research's SID debugger, you can use a loader option under BDS C to write out a symbol table to disk. This aids debugging. The loader also will allow you to create ROMable code, and overlay segments. This means that you can write programs bigger than available memory, and bring extra segments in when needed.

Most of the limitations of BDS C are due to its restricted subset of the language, and to the way CP/M operates.

Functions are stored in a relocatable format, and are kept together in larger "library" files. An interactive librarian program is included, which lets you create new function libraries and change old ones. When I found a bug in one of the library routines, this let me simply replace the old one with the new improved version. There is a section of the user's manual about the format of these relocatable files, for those who wish to write their own in assembly language, and a macro library to help even further. Of course, any C function can become part of the library by just compiling it, and using the librarian.

The compiler does some optimization (for speed or size—your choice). Also, if you are using a Z80 processor, this fact will be automatically sensed, and block move instructions will be used when appropriate. I have found that BDS C generally produces the fastest code of the three compilers.

If you write programs for sale using their compiler, Whitesmiths will expect you to pay them royalties because you will be distributing parts of their run-time and function libraries along with their code. Leor Zolman has specifically declared that none of this sort of thing is necessary with the BDS C compiler. And the entire Small-C compiler is in the public domain.

Nobody Is Perfect

Most of the limitations of DBS C are due to its restricted subset of the language, and to the way CP/M operates.

BDS C does not support float, long or double types; static or register storage classes; initializers; blocks, or parameterized defines. Externals are handled like COMMON in FORTRAN; there are external variables, and no external keyword. There are other differences covered in the manual, but these are the major ones.

What About Tiny-C?

Conspicuously absent from this issue is a review of the Tiny-C Two compiler from Tiny-C. The company had furnished us with a review copy of their package and we arranged for the review with an individual who very regretfully failed in his committment. We then asked Tiny-C to supply a second copy for another reviewer who still has not completed his review. We're hopeful that we will be able to furnish you with this review in the near future. The Tiny-C people certainly have been cooperative in supporting our reviewing efforts, and we feel most guilty since they were the first to supply a review copy of a C compiler, and will be the last to be reviewed.

I would therefore like to point out that Tiny-C Two does support the redirection features of Unix C which are missing from the Small-C and BDS-C compilers. Further, it supports 32-bit integers (compared to only seven in BDS-C) and hence is viable for scientific and business applications. Also, the full source code is included with the $250 price (manual alone is $50) and a 20% reduction is given to owners of the older Tiny-C One interpreter. *—Editor*

NEW LOWER 8088 PRICES

LDP1/1

The LDP1/2 utilizes the advanced 8088 processor to provide up to 8 times the throughput of a 4 MHz Z80A processor. The powerful instruction set of the 8088 is ideally suited to higher level languages such as PASCAL and PL/1. The 10 slot motherboard leaves 7 slots for USER expansion. With the option of a 10 MByte Winchester and MP/M-86, the LDP mainframe becomes a powerful multiuser system with the capability of handling 8 users without the degradation in performance experienced with Z80 CP/M systems. The performance of the LDP1 and LDP2 has never before been available for such an affordable price.

FEATURES:
— LDP88, 8088 CPU board
— LDP72, advanced floppy disk controller
— LDP64K dynamic RAM
— 1 serial RS232 port
— 10 slot motherboard
— 1 8" Shugart 801R drive (LPD1), 2 Shugart 801R's (LDP2)
— 4K EPROM socket for user population

OPTIONS:
— HAZITALL
— 8" 10 MByte Winchester (replaces 1 Shugart 801 in LDP2, available Sept. 81)
— MP/M-86 multiuser system
— Woodgrained 7 slot chasis

PRICES	ASSEMBLED & TESTED
LDP88 CPU	$ 349.95
LDP72 FDC	274.95
LDP64K RAM	695.00
LDP128K RAM	1295.00
LDP256K RAM	2095.00
HAZITAL	325.00
LDP1	3295.00
86-DOS	195.00
CP/M-86	250.00
Microsoft BASIC 86	350.00

Call for LDP1 option prices and board kit prices.

CP/M-86 and MP/M-86 are trademarks of Digital Research
86-DOS is a trademark of Seattle Computer Products

LDP88 8088 CPU BOARD

• 8088 CPU 5 MHz operation upgradeable to 8 MHz • 9 vectored interrupts • Fully complies with IEEE 696 electrical and timing specs • RS232 serial port with modem controls • 1K bytes of static RAM • 2 EPROM sockets (2716 or 2732) • 8087 upgrade kit • 8 bit bus eases interface to other S100 bus boards • 1MByte address space • 65K I/O ports

64/256K MEMORY

• 8 or 16 bit operation • Meets all IEEE 696 specs • Access time 350 ns from PSYNC low • Intel 8203 dynamic RAM controller • 24 or 16 bit address decoding • No wait states with 5 MHz 8088 or 8086 • Parity with Error interrupt generation • No DMA RESTRICTIONS • 64K board is upgradeable to 246Kbyte board

HAZITALL

• 2 Serial RS232 ports • 2 parallel ports with handshake control • Math processor support (8231/9511 or 8232/9512) • WINCHESTER DISK support • Real time programmable interrupt • Clock/calendar with battery back up • Synchronous data communication supported

LDP72 FLOPPY DISK CONTROLLER

• IBM compatible single and double density format • Single or double sided drives • Programmable data record length (128 to 8192 bytes/sector) • Multi sector and multi track transfer capability • Parallel seeks on up to 4 drives • On board digital data separator • Software selectable single or double density operating • Separate connectors for 5 1/4" and 8" drives • Software selection of standard or minidrives allowing mixing of both drives on a single controller

Lomas Data Products

11 Cross Street
Westborough, MA 01581
Telephone (617) 366-4335

How important are these missing features? It depends on the type of the work you plan to do. While the lack of floating point might lead you to assume that you couldn't write *any* business or scientific programs, this is not quite the case. A floating point package is included with BDS C, which lets you work to approximately seven digits of precision. It does work and is not terribly hard to use, but it is rather slow. You wouldn't want BDS C if you intend to write an Accounts Receivable system.

Initializers are a convenience only; several functions included let you initialize variables in the code. External storage can often be used instead of static, and there is now a "long" package available from the BDS C User's Group. Generally, long variables and static storage are the only things you will really ever miss for system-type programming.

You might still be wondering if there would be a limit on your creativity. Currently, you can obtain the following types of programs, all written in BDS C:

Adventure
6800 and 1802 Cross-Assemblers
A text processor
A file comparison utility
Extended Directory Listing
The Game of Life
A disk "zapping" program
A full-screen text editor
An intelligent file transfer and communications package

One of the attractions of Whitesmiths' product is that redirection to/from files and devices is supported as on UNIX.

With the exception of the last two products mentioned (which are commercially sold under the names MINCE and AMCALL), these useful programs, and others like them, are available free from the BDS C Users' Group. And apart from the editor and the Life game, none of the programs appear to run slower than an equivalent assembly language version might. The point is that BDS C provides a viable subset of the C language.

While not a limitation of BDS C, it should be noted that CP/M is not exactly like UNIX. On UNIX, you can redirect input so that a program which expected input from the console could get that input from a file instead, without changing the program. Similar redirection is possible for output, in fact, any device could be treated like a file for this purpose. CP/M, of course, doesn't have these features. For this reason, BDS C programs which call the **getchar** function (used frequently on UNIX for reading files as above) will get their input from the console only. File I/O must be done deliberately. Small-C works in a similar manner.

One of the attractions of Whitesmiths product is that redirection to/from files and devices is supported as on UNIX. This is done in their run-time I/O library, and is one reason why programs written in Whitesmiths C tend to be somewhat large. The ideal situation is to have this facility in the operating system.

Do You Want to Own It, or Just Use It?

In my opinion, the usefulness of a piece of software is in part measured by its speed and size. If you intend to use a program often—an editor or compiler, for example—it will become a burden if the time needed to use it is too great. I compiled the simple C program listed above with all three compilers (and on UNIX just for comparison), making changes as necessary. The results are listed in Table 1.

Table 1.

COMPILER	mins:secs to compile, link, and load	actual code generated (bytes)	executable program size	time to execute
BDS C	0:25.1	34	2304	2.8
Small-C	1:42.8	44	3200	4.2
Whitesmiths	4:04.1	28	15232	5.1
UNIX (20 users)	1:02	20	4184	1.0
UNIX (optimized)	1:25	20	2468	1.0
UNIX (single user)	0:17	20	4184	1.0

The intent of testing such a simple program was to get the minimum times and sizes (due to the size of the run-time package) possible for each compiler. Be assured that compile times can increase considerably for even slightly larger programs. In the case of BDS C, the entire compile and link process was faster than the assemble and load alone for Small-C (done by the standard CP/M utilities). This means that BDS C can give you executable code more quickly than if you wrote the equivalent program in assembler. And it is usually much faster to write in C than in assembler.

While repeated compilations are no substitute for careful program design, it is maddening to wait many long minutes, only to find that you made a slight error. Then you have to edit and recompile. I first began programming in C with the Whitesmiths compiler and much slower disks. The frustration level was so high that I gave up C altogether for several months. Using this compiler also necessitated doing all sorts of hardware modifications to my system, so I could make ROMs disappear and have enough memory to run it. Even now, I am looking for a way to make my Zapple monitor respond to the PHANTOM bus signal, so I can squeeze out another 2K of RAM.

When I got Small-C, I was favorably impressed by its speed and error reporting, but the severely limited subset of the language and I/O facilities frustrated me again. The simplest programs available had to be completely reworked to fit the Small-C format, and I found myself spending more time working around the compiler than using it.

BDS C changed the whole picture. I was able to write programs the way I wanted, getting fast results. When I made a mistake, the compiler was able to show me how

and where, in unambiguous language. Within days, I felt comfortable enough with the documentation provided to contact Leor Zolman telling him I thought I had found an obscure bug in a library routine. By the time he had called me back with the fix, I was able to fix it myself, and install the new version in the function library. This was only a matter of an hour or so. The importance of all this? I could do these things because the entire function library is included on disk in C source code form. Not only that, but the run-time library is also included (written in 8080 assembly language). This is invaluable material for learning more about how the functions work, as well as having a lot of good C code to study.

Bugs? After looking through issues of *Lifelines*, which lists bug reports and new versions of both BDS C and Whitesmiths, I found that Whitesmiths tends to release versions at longer intervals. Both compilers had serious bugs reported (crashes, incorrect code, etc.). This means that you would have longer to wait for a fix if you own Whitesmiths.

Now, what about updates? Since you get the source for much of what is important in the BDS package, it is possible, in many cases, to get bug fixes in source form as I did. And disk updates for the whole package are available for $8 from the BDS C Users' Group.

In the case of Whitesmiths, things are a bit more complicated. I got an early version of their compiler (1.1), and received one "free" update, for which I had to pay a $30 "media charge." Now, as the proud owner of Version 1.2, I find in *Lifelines* that this version has such a multitude of serious bugs that immediate upgrade to 2.0 is recommended. The only problem is that an update from Whitesmiths would cost me a $200 fee (plus another $30 media charge).

Final Words

A good analogy for these compilers might be to compare them to items with similar price relationships:

For $15 or so, you can buy a cheap camera, and take properly exposed, slightly fuzzy pictures at an average distance. You don't have to be very committed to photography to be able to use or afford one of these.

For maybe $150, you can find a high-quality 35mm SLR camera, which can take professional quality pictures. It will be more versatile, and will give you a great deal of enjoyment. While you don't have to use it to potential, it is nice to know that the user will almost always be the limiting factor in its performance.

Finally, you can spend $600 and get one of the most well-known and respected models, the kind all the "pros" use. You can impress your friends with how much it cost, and you might just take some good pictures with it. Try to take good care of it, for if something goes wrong, it will probably cost more to fix than the purchase price of the $150 camera. It might be a bit harder to use, and have its own peculiar problems, but that can be expected—it's special, remember?

Unless you must have floating point and statics (and have a very fast disk system with lots of RAM), I suggest you buy BDS C if you want to use a C compiler on CP/M. ■

Notes

1. All CP/M times were obtained on a 59K 3 MHz Z-80 based system running double-sided, double-density 8" floppies.
2. Times for BDS C were done on a single-density disk, and so are probably 20% higher than they would be otherwise.
3. Whitesmiths compile alone (all three passes) took about 1:30. The rest was spent in loading (this loader is notoriously slow). Note that large size of run-time package reflects additional UNIX-like capabilities.
4. Sizes of executable code (.COM files) were calculated by the number of records, so each could be as much as 127 bytes too high.
5. The string being printed takes up 14 bytes itself, which is included in the totals for actual code generated.
6. UNIX times were obtained on a PDP-11/70 running UNIX Version 7, with two 176 MB hard disks. The ideal home system.
7. PDP-11 code was used for code sizes on UNIX. Notice also the difference in compile times between average (20 users) and no (1 user) system loading.

UNIX is a trademark of Bell Laboratories.
PDP-11 is a trademark of Digital Equipment Corp.

Where To Find Them

BDS C Users' Group
409 E. Kansas
Yates Center, KS 66783

Membership is $10/year domestic, $20/year foreign. This brings you regular newsletters. You don't have to join to buy disks, at $8 apiece for 5 1/4" or 8" size. In the future this group hopes to support other C compilers but currently they only support BDS C.

Whitesmiths, Ltd.
P.O. Box 1132
Ansonia Station
New York, NY 10023
(212) 799-1200

The C compiler requires 60K CP/M at a minimum. Version 1.2 was reviewed.

(source for BDS C)
Lifeboat Associates
1651 Third Avenue
New York, NY 10028
(212) 860-0300

Version 1.43 of BDS C was used for this review. System size: 32K CP/M minimum, 48K recommended.

Lifeboat distributes BDS C and Whitesmiths. The update fees may be a bit better than directly from Whitesmiths, and they sell software in almost any format you can name.

The Code Works
P.O. Box 550
Goleta, CA 93017

Version N was used. No recommended system sizes are given, but the compiler itself takes 22K, and it was developed on a 40K system with a single mini-floppy.

Desk Main/Frame

Desk Main/Frame

LOW COST & ATTRACTIVE STYLING

- MAIN/FRAME INTEGRATED INTO FURNITURE QUALITY DESK
- ELECTRONICS PACKAGE SLIDE MOUNTED FOR EASY ACCESS
- SUPPORTS TWO 8" FLOPPY DRIVES FROM SEVERAL MANUFACTURERS (DRIVES NOT INCLUDED)
- 10 SLOT MOTHERBOARD INCLUDES CONNECTORS
- POWER SUPPLY FOR DRIVES AND CARDS
- DESK AND MAIN/FRAME AVAILABLE SEPARATELY
- MATCHING PRINTER DESK AVAILABLE

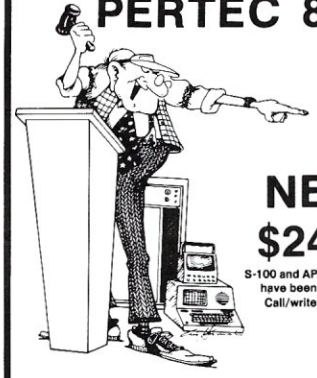
WRITE OR CALL FOR OUR BROCHURE WHICH INCLUDES OUR APPLICATION NOTE: 'BUILDING CHEAP COMPUTERS'

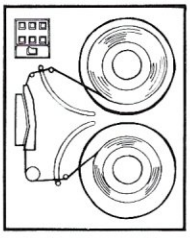
INTEGRAND

8474 Ave. 296 • Visalia, CA 93277 • (209) 733-9288
We accept BankAmericard/Visa and MasterCharge

ALMOST AN AUCTION

9-TRACK TAPE DRIVES
800 BPI, 45 IPS
PERTEC 8840A-9-45





NEW $2400

S-100 and APPLE interfaces have been developed. Call/write for details.

- BARGAIN PRICED MINI COMPUTER UPGRADE
- READ THE NEW 9 DIGIT ZIP CODE TAPES AVAILABLE FROM USPS

A large OEM's large overstock makes these industry standard drives available at a fraction of their current list price. Full size drives handle up to 10.5 inch reels of standard 1/2 inch mag tape. 19 inch rack mount or use right out of the box on steel shipping frame.

SPECIFICATION SUMMARY: 9-Track, 800 BPI, dual head (read-after-write), 45 ips read/write, 200 ips rewind, BOT/EOT sensing, 110 VAC/60Hz, recent manufacture, all i/o signals TTL/DTL compatible, tension arm tape buffering, full control panel, compatible with many other drives including lower priced Pertec 6840s.

PRICING: Current OEM list price is $4800+ quantity 1, $3400+ quantity 100. Cash price, new in original box with full manual, guaranteed, is $2400 FOB Morris Plains. Large quantity in stock in our warehouse, we are not brokers.

ELECTROVALUE INDUSTRIAL INC.
P.O. BOX 157 S
MORRIS PLAINS, NJ 07950
Formerly Electravalue Industrial

Phone reservations and questions are welcome.
201/267-1117

THINKING ABOUT WORD PROCESSING???

Alpha Data Services offers the small businessman and serious computer enthusiast a choice of the finest S-100 equipment and low cost Daisy Wheel Terminals.

We Carry a Complete Selection of Equipment By:

California Computer Systems

EXAMPLES:		LIST:	ALPHA:
2422A	FLOPPY DISK CONTROLLER WITH CP/M 2.2*	425	329
2810A	4 MHZ Z-80 CPU DMA/SERIAL/FR. PANEL	310	249
2065C	64K DYNAMIC MEMORY 200 NS	720	499
2116B	16K STATIC MEMORY 300 NS	389	269
2032B	32K STATIC MEMORY 300 NS	755	579
2200A	12 SLOT MAINFRAME 20 AMP POWER SUPPLY	435	379

COMPARE OUR LOW, LOW PRICES BEFORE PURCHASING!

DIABLO 1620

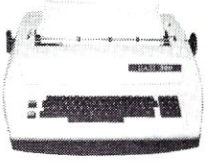
- Hytype-II Mechanism
- Mint Condition Refurbished and Tested
- 110, 150, 300, 1200 Baud
- Menu Installation Under Word-Star**

$1795.00

DTC-300S

- Reliable Hytype-I Mechanism
- Completely Refurbished and Tested
- 110, 150 or 300 Baud

$1249.00



RIBBON SPECIAL!!

- Hytype I & II Cloth and Flim Ribbons . . . black and colors **$17.50**/box of 6
- Diablo Matrix Cloth Ribbons **$25.00**/box of 6

All our terminals come with RS-232 Cable, forms tractors, print wheel, ribbon, and typewriter height roll-around stand. They are warranteed for 30 days. Crating is FREE!!

*CPM is a trademark of Digital Research
**Word-Star is a trademark of Micropro Int.

ALPHA DATA SERVICES

810 Daleview Place
Greensboro, North Carolina 27406
(919) 373-1726

North Star Computers' New 5.2 Software

by Steven Leibson

The fine folks at North Star Computers (1440 Fourth Street, Berkeley, CA 94710) have done it again. Release 5.2 of North Star DOS and Basic has several new features that make the software much more powerful. The new release will run on any North Star Horizon or other computer system equipped with a North Star MDS disk subsystem. Versions are available for both single and double/quad density systems. I purchased my copy through Computerland of Denver for $25. I think it's a steal.

Relocation

First and foremost of the new features is the MOVER program, written in North Star Basic. This program will relocate the DOS, Basic and the utility programs of Release 5.2 to *any* start address. The big advantage for North Star users is getting an extra 8K of memory space for programs.

Since its introduction, users have complained about North Star's DOS starting at address 2000 (hex). The lower 8K of memory had to go unused. Since the disk controller was located at E800 (hex) there was only about 28K of memory available for Basic programs. North Star did offer relocated operating systems on special order.

Release 5.2 allows you to relocate any of the utility programs in the standard North Star software system to any location. By relocating DOS to address 100 (hex) and Basic to E00 (hex), you can get an extra 8K bytes of memory. If you have CP/M on your North Star, you have memory down there anyway. Now you can take advantage of it for both operating systems.

Unfortunately, the single-density version of Release 5.2 cannot be relocated, as shipped from North Star. This is because there are absolute-address references in the single-density boot ROM that require the DOS to start at location 2000 Hex. John Dvorak (704 Solano Ave., Albany, CA 94706) is marketing a utility program that does allow relocation of single-density 5.2. This is accomplished by including a relocatable version of the boot ROM in the disk file with the DOS.

Steve Leibson, 4040 Greenbriar Blvd., Boulder, CO 80303.

System Software Errata

Before we look at the new features of Release 5.2 in detail, there are a few errors in the North Star System Software Manual to correct. I'll cover them so you won't have to buy a copy of the new DOS just to get the corrections.

1. On page A-2 of the DOS section, file directory entry bytes 10-11 are described as, "number of blocks in file," but should be described as "number of sectors in file." For single-density systems, sectors and blocks are equivalent. Double and quad-density systems pack two blocks per sector.

2. On page G-2 of the DOS section, one of the comments before DCOM reads "ACC=NUMBER OF BLOCKS." It should read "ACC=NUMBER OF SECTORS."

3. After a call to DCOM, the stack pointer will be left unchanged whether DCOM exited via the return address on the stack, or via the HDERR vector. This should be noted on page G-2 of the DOS section.

4. When performing DOS personalization, it is important to note that DOS 5.2DQ loads in two parts:

    Sectors 4-8 load into locations 100-AFF hex.
    Sectors 8-9 load into locations A00-DFF hex.

Usually this means that when loading an image of DOS into memory to modify using the LF command, the data in sectors 8 and 9 of the DOS file will be loaded 100 hex bytes lower than their operating address would indicate. This should be noted on page F-2 of the DOS section.

New DOS Features

The first new feature in the DOS is the acceptance of

Z-80 and 8086 FORTH

Z-80 FORTH — a complete program development system. Uses standard CP/M® compatible random access disk files for screen storage. Package includes: interpreter/compiler with virtual memory management, line editor, screen editor, Z-80 Assembler, decompiler, utilities, demonstration programs, and 80 page user manual. System requirements: Z-80 microcomputer, 48 kbytes RAM, CP/M 2.2 or MP/M 1.1 ..... **$50.00**
With software floating point arithmetic **$150.00**
With AMD 9511 support routines **$150.00**

Z-80 FORTH WITH NAUTILUS SYSTEMS CROSS-COMPILER. Extend/modify the FORTH runtime system, recompile on a host computer for a different target computer, generate headerless code, generate ROMable code with initialized variables. Supports forward referencing to any word or label. Produces load map and list of unresolved symbols. 107 page manual. System requirements as for Z-80 FORTH above ... **$200.00**

8086 FORTH with line editor, screen editor, assembler, and utilities. Uses standard CP/M compatible random access files for screen storage. Requires 8086 or 8088 microcomputer, 64 kbytes RAM, and CP/M-86 operating system..... **$100.00**
With AMD 9511 support routines **$200.00**

MACHINE TEST PROGRAM PACKAGE FOR Z-80 systems. Includes memory, floppy disk, printer, and terminal tests with all source code. Requires CP/M 2.2 ..... **$50.00**

All software distributed on eight-inch soft sectored single density diskettes. Prices include shipping by first class or UPS within USA or Canada. COD charges extra. Purchase orders accepted at our discretion. (CP/M and MP/M are registered trademarks of Digital Research, Inc.)

Laboratory Microsystems
4147 Beethoven Street
Los Angeles. CA 90066
(213) 390-9292

lower-case commands. I don't find this to be a major feature because all my file names are in upper-case letters. Unlike CP/M, the case of the file names matters to Release 5.2 DOS, so I still keep the alpha-lock key down on my keyboard and continue to ignore lower case for commands.

Other new features in the DOS are definitely designed with third party assembly-level software in mind. There are two new entries in the I/O jump table to check for device status. One is for input devices, and the other is for output devices. The routines associated with these locations accept the device number in the accumulator (0 through 7) and return the Z flag true if the specified device is ready. Ready for an input device means that a byte is available from the device. Ready for an output device means that another byte may be output.

The two status calls in the DOS mean that programs need no longer know how the I/O of any given system works. To find out if the device is ready, one standard call is all that is required. The same has always been true for input and output in North Star software. With the addition of status routines, the I/O can be truly independent.

A big change in the DOS has been made to the disk routines. North Star users have previously not been able to use interrupts and the disk system at the same time. The North Star disk routines are time critical and cannot be interrupted. Now interrupts are unconditionally disabled during disk transfers and conditionally enabled when disk access is complete.

A routine called OFTEN has always been called while disk transfers are taking place. By combining this routine with interrupts during the times the disks aren't being used, keyboards and other input devices may be interrupt driven.

A new routine in the DOS is used with the addition of interrupts. North Star's RAM boards have the ability to interrupt if a parity error is detected. Since the software previous to Release 5.2 did not allow interrupts, this feature was not used. Instead, an LED inside the computer lit up if there was a parity error, a feature useless for those of us who keep the cover on the computer most of the time.

Interrupts may now be used to monitor parity errors. The new parity routine will send the message "RAM PARITY ERROR" to the console (device 0) if an error is detected. If you wish to use interrupts for other purposes, then the DOS routine written by North Star will have to be modified.

DLOOK is a routine in the DOS that allows you to find the location of a file. Correct file name syntax had to be used or the routine wouldn't work. DLOOK has had error checking for file name syntax added. If the syntax is wrong, a zero is returned in the accumulator. Otherwise, the drive number (1 through 4) is returned, indicating which disk has the requested file.

The disk access routines for double and quad-density machines have been rewritten. Disk access is now noticeably faster. North Star suggested that this could be seen during copy operations. I found this to be true.

Finally, a byte has been added to point to the location of the disk controller. Just as with the DOS, North Star has provided non-standard start addresses for the disk controller ROM. One way of rebooting a North Star is to jump to the first address in this ROM. A pointer allows a standard location to be accessed to find out where the reboot location is. Unfortunately, since the DOS need no longer start in a standard location, this byte may be hard to find.

New Utility Programs

One problem a user faces is where to start when the system won't wake up. Is the problem in the hardware, the new I/O routines just added, or in the software from the manufacturer? North Star has added a program to their Release 5.2 disk that runs a checksum on the entire diskette. This allows a dealer, who presumably has a functioning system, to check out the diskette and verify its integrity. This assumes you have such a local dealer.

Another problem in maintaining a system is verifying the RAM. North Star has provided two copies of a RAM test program written especially to test North Star RAM boards. The difference in the programs is where they are located in memory. The test is a six-phase checkout of RAM operation. The display shows individual chips, so it is easy to locate the troublemaker.

The test supplied by North Star requires the console to be attached to the Horizon serial console port. Randy Reitz suggested searching the RAM test program for all outputs to the serial port. This is done with the monitor commands:

```
> LF RAMTEST 5000
> SM 5000,400 D3 2
```

These commands will identify the three locations that output to the console. If you use a different I/O port for the console, the only bytes that will need to be changed are the 2's that specify the serial port. My memory-mapped video display requires more than a simple OUT command, so I have not yet decided how to get the RAMTEST routines working on my system.

New Features In Basic

Several new features have been added to North Star Basic, the first of which I remember asking North Star for myself. The output routines in Basic have always been too smart. Each of the output devices has a line length. If enough characters are output to a device to exceed the line length defined for that device without sending it a carriage return and line feed, good old Basic sent those characters out for you. There was no way to disable this feature.

The reason I wanted to disable automatic transmission of carriage return and line feed was because I was building an editor in Basic. I found that if you made enough mistakes and used the backspace enough times, the cursor would automatically jump to the next line. It took a long time to figure out that the line length was being exceeded.

My solution for this problem was to periodically go into the print head table where Basic keeps the current character count for each output device, and to zero out the count using the FILL statement. That stopped the automatic character generation, but placed obscure lines of code in my program.

How Much is Your Sanity Worth?

	List Price	Our Price
WordStar Version 3	495	425
SpellStar	250	225
MailMerge Version 3 (requires WordStar Version 3)	125	110
All Three Above (requires WordStar Version 3)	870	725
Condor DBMS Level II Relational Data Base	995	850
C. Itoh Starwriter I 25 cps Daisywheel Printer	1895	1650
C. Itoh Starwriter II 45 cps Daisywheel Printer	2495	2250
Televideo 950-C Terminal With Detachable Keyboard	1195	1050

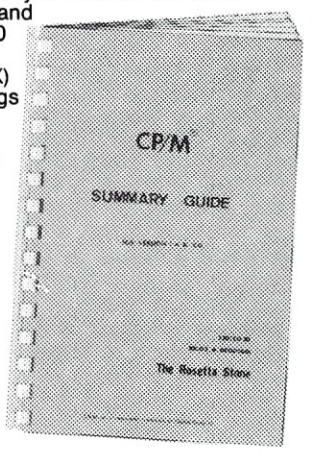
All orders should be prepaid
Add 5% shipping charges for hardware
NJ residents add 5% sales tax

Information Technologies

38-67 Taylor Road
Fair Lawn, NJ 07410
201-796-3140

CP/M SUMMARY GUIDE

Tired of fanning through your CP/M manuals or writing notes that remind you of the commands, functions and error codes? Well it's about time you ordered our CP/M Summary Guide! Spiral bound and handy to hold, our guide is a 60 page booklet summarizing the features of CP/M (Ver. 1.4 & 2.X) and 2 totally alphabetical listings of the commands, functions, statements and error codes of MICROSOFT BASIC-80 Ver. 5.0 and CBASIC™ -2. Areas summarized are in table form and include all direct and transient commands plus MAC™, DESPOOL™ and TEX™. Our booklet is a much needed supplement to any of the literature currently available on CP/M and has been recommended by Digital Research.



P.S. Over 4000 users can't be wrong!

Ask your local computer store for our guide or send $6.95 plus $1.00 (postage and handling) to:

THE ROSETTA STONE, P.O. BOX 35, GLASTONBURY, CT 06025 (203/633-8490)

Name _____

Street _____

City_____ **State**_____ **Zip**_____

CP/M™, DESPOOL™, MAC™ are registered trademarks of Digital Research.
CBASIC™ is a registered trademark of Compiler Systems.

Bring the flavor of Unix To your Z80-based CP/M system with Unica

"Unicum: a thing unique in its kind, especially an example of writing. Unica: the plural of unicum."

The Unica: a unique collection of programs supporting many features of the Unix operating system never before available under CP/M. The Unica are more than software tools; they are finely crafted instruments of surgical quality. Some of the Unica are:

bc - binary file compare
cat - catenate files
cp - copy one or more files
dm - disk map and statistics
hc - horizontal file catenation
ln - create file links (aliases)
ls - directory lister
mv - move (rename) files, even across users
rm - remove files
sc - source file compare, with resynchronization
srt - in-memory file sorter
sr - search multiple files for a pattern
sp - spelling error detector, with 20,000 word dictionary

Each Unicum understands several flags ("options" or"switches") which control program alternatives. No special"shell" is needed; Unica commands are typed to the standard CP/M command interpreter. The Unica package supports several Unix-like facilities, like filename user numbers:

```
sc data.bas;2 data.bas;3
```

(compares files belonging to user 2 and user 3);
Wildcard patterns:

```
rm *tmp* -v
```

(types each filename containing the letters TMP and asks whether to delete the file);
I/O redirection:

```
ls -a ►list
```

(writes a directory listing of all files to file"list");
Pipes:

```
cat chap* ! sp ! srt ►lst:
```

(concatenates each file whose name starts with"chap", makes a list of mispelled words, sorts the list, and prints it on the listing device).

The Unica are written in XM-80, a low level language which combines rigorously checked procedure definition and invocation with the versatility of Z80 assembly language. XM-80 includes a language translator which turns XM-80 programs into source code for MACRO-80, the industry standard assembler from Microsoft. It also includes a MACRO-80 object library with over forty "software components", subroutine packages which are called to perform services such as piping, wildcard matching, output formatting, and device-independent I/O with buffers of any size from 1 to 64k bytes.

The source code for each Unicum main program (but not for the software component library) is provided. With the Unica and XM-80, you can customize each utility to your installation, and write your own applications quickly and efficiently. Programs which you write using XM-80 components are not subject to any licensing fee.

Extensive documentation includes tutorials, reference manuals, individual spec sheets for each component, and thorough descriptions of each Unicum.

Update policy: each Unica owner is informed when new Unica or components become available. At any time, and as often as you like, you can return the distribution disk with a $10 handling fee and get the current versions of the Unica and XM-80, with documentation for all new or changed software.

The Unica and XM-80 (which requires MACRO-80) are priced at $195, or $25 for the documentation. The Unica alone are supplied as *.COM executable files and are priced at $95 for the set, or $15 for the documentation. Software is distributed on 8" floppy disks for Z80 CP/M version 2 systems.

Knowlogy

"Shaping Knowledge for Evolving Worlds"
P.O. Box 283
Wilsonville, Oregon 97070

Visa/Mastercard customers call (503) 635-5701 after hours for next day shipment.

CP/M is a trademark of Digital Research; Unica is a trademark of Knowlogy; Unix is a trademark of Bell Telephone Labs; XM-80 is a trademark of Scientific Enterprises; Z80 is a trademark of Zilog Inc.

The upgraded LINE statement in Basic takes an optional third argument. If zero, the automatic feature is suppressed. Otherwise carriage return and line feed are generated as before

There are two new functions and a modification to the WRITE statement for dealing with disk files. FILESIZE(N) returns the size of the file specified by N in blocks. FILEPTR(N) will tell you where the file pointer is in file N. Many disk oriented programs in Basic had to keep track of this, so FILEPTR is a welcome addition. WRITE can now be used to directly set the file pointer using the syntax:

WRITE #N %P, NOENDMARK

This syntax also allows resetting of the file pointer to the beginning of the file without closing and reopening it.

DOS Bugs Fixed

All software has bugs, and North Star's is no exception. Release 5.2 documentation has a list of bugs that have been eradicated. The CR (create file) DOS command now fails if an improperly formed file name is given. This is undoubtedly tied to the error checking added to DLOOK mentioned above.

The IN (initialize) command can now use the read-after-write option. Use of read-after-write during media initialization will serve as a media verification in addition to the initialization. It's always nice to know that the diskette you are about to commit your software to has just passed some kind of test.

"Hard disk error messages are now routed to device 0." The documentation tells you this, but I don't know where they went before. Error messages have always appeared on console.

Single-density DOS has been corrected to accept carriage returns from device 0 when paging causes a stop during a directory listing. After pressing return in response to the message "PRESS RETURN TO CONTINUE," a carriage return and line feed are output to clear the line. Auto-start can now be selected in single-density without using a "fresh" copy of the DOS. If auto-start is selected, the initial message sign-on message is not displayed. That's good news for writers on turnkey software. They will now be in full control of the display screen.

A big bug I could never get verification on has been fixed in double/quad-density DOS. It will now boot correctly, regardless of what is in memory prior to loading the DOS. Sometimes after using CP/M or other programs, I couldn't get North Star DOS to load until I turned my system off and then on again. I believe that I was experiencing this bug. So far, I haven't seen it using Release 5.2. (According to Randy Reitz, Release 5.1 may be fixed by changing location 203B hex to 00 and saving the patched DOS.)

Some bugs in the double/quad-density directory listing have also been fixed. A hard disk error will no longer cause the directory listing to hang-up forever. The directory listing will no longer destroy software that has overlaid the high part of DOS starting at location 2A00 (hex) if the DOS starts at 2000 (hex). OFTEN will now be called even for drives with fast stepping capability.

Utility Bugs Fixed

The CO (copy disk) utility has also been upgraded. CO will not proceed if a disk has overlapping files. It will no longer offer to convert a double-density-only diskette to double density, and it will now handle all density mismatch errors properly.

CF, the copy file utility, can now handle improper file names like the CR command. It can also handle density mismatches like the CO utility. Small files can now be copied into large files.

Basic Bugs Fixed

Basic has also had bugs fixed. CREATE had an obscure bug that affected other statements. This has been corrected. (I never got involved enough to know what the bug was.) Numeric overflows will be handled correctly when dividing large numbers by very small numbers. (I never saw that one either.) The user-defined functions (UDF's) precluded the use of ELSE when they appeared in a THEN clause. ELSE is now permitted after a UDF. DEF statements may now have other statements following on the same line.

Getting 5.2 Running

I had very little trouble getting Release 5.2 up and running once I decided to do it right. All of my I/O routines had to be rewritten since the DOS now starts at location 100 (hex). This forced me to finally put the routines in an assembler source file so that they would be easy to modify in the future. I also had to add the new status routines for my devices.

That won't be necessary if you have a Horizon with I/O as North Star designed it. If your console uses the first serial port and you only use the Horizon's serial and parallel ports for I/O, you can use the DOS as supplied. Those of you who have non-standard Horizon I/O or MDS subsystems in "foreign" computers will have to write new I/O drivers as I did. If you've done this before, you should have no problem.

One other problem I had was attributable purely to "pilot error." The double/quad-density DOS is supplied in double density configuration. I left it that way and got a lot of file errors on old files before I realized that I needed to change the CONFG byte to let the software know I had a quad-density system.

After the problems listed above were solved, I was able to run old Basic programs with the new system, without fail. The newly relocated DOS conveniently starts at the beginning of CP/M's TPA (transient program area).

Recommendation

The status of North Star DOS is uncertain now that North Star is also offering CP/M for their systems. I still do not favor CP/M for my program writing because I'm not able to use North Star Basic under CP/M.

If you are still using North Star DOS, get Release 5.2. It's faster, has fewer bugs (especially that boot problem), and more features. The cost is low compared to what other operating systems are going for, and you get a language (Basic). If you have a North Star system, Release 5.2 will give you more flexibility and ease of use. ■

The $150 48K Memory Board

by Richard A. Rodman

A lot of people ask the question, "How can the S-100 bus be better, when it costs so much less to expand the memory of (choose one) TRS-80, Apple, Sorcerer, Wankel Rotary Engine?" For those of us who are dyed-in-the-wool Altair/Imsai/Ithaca Audio fanatics, such queries are usually answered by muttering the sacred incantation "zee-ay-tee-sepiem-floppydisk" or appealing to Ecstatic Ram.

Well, now you can share their anxieties about alpha radiation induced soft errors, long wait states and DMA. But if you don't use long wait states (e.g. Tarbell disk) or DMA (Dazzler et al), you too can use 4116's and get 48K of RAM at bargain prices. I must note here that this circuit requires the ubiquitous Z-80 processor, as it depends on its usually-ignored Dynamic Memory Refresh.

The circuit herewith described makes use of only common TTL circuits. Instead of exotic chips like the 8202 or 3242, I use garden-variety logic like 74157's and 7442 and 7400. The 4116's may be obtained from surplus mail-order shops, or you can buy expansion kits put together by a myriad suppliers for the TRS-80, Apple, or Sorcerer.

I wire-wrapped mine on a California Computer Systems prototyping board, using heavy-guage wire for power buses. All other connections were made with Slit'N'Wrap (Vector, Inc.).

Layout of the board is shown in Figure 1. First, insert pins at power supply points marked with heavy dots, install and test regulators. Then, run #22 or larger bare wire to create buses for +5, +12 and -5 volt power supplies. Then install all sockets. I suggest wiring power supply pins first. Insert pins at appropriate S-100 bus pins, which are labeled on the CCS board.

Richard A. Rodman, 1944 Kennedy Dr. #201, McLean, VA 22102.

Now you are ready to wire the signal pins. I used a daisy chain, going up a column, making a big loose loop back to the bottom of the column, going to the next column etc. until I finished with the signal. I then cut off the big long loops. *Never* run signals across the chips or between pins! If you do, you are asking for shorts—maybe not now, but surely within the next few years. Just try tracking them down in *that* maze of wires! (Got a weekend to spare?)

I don't waste time making up wiring lists or charts, so naturally I don't think you should either. Wire directly from the schematic in Figure 2, checking off wires as you go.

So how does it work? Just fine. The MRQ signal, which is an inverted replica of MEMRQ* from the Z-80 chip, generates RAS* directly. Most CPU boards put this out in one form or another. The pin given is that used by the Ithaca Audio CPU.

This signal is also clocked into flip-flop IC7 so that one-half of a clock cycle later, the multiplexers are switched to the column address, and one-half clock cycle later, CAS* is generated on the selected bank of chips. Note that RFSH*, if true, prevents the generation of CAS*. The Z-80 chip only allows one clock cycle for memory refresh, so that the refresh cycle ends when MRQ goes away. That's about all there is to it. I've had no problems at all with this board.

An interesting note on the alpha radiation problem: It seems to be generated by the ceramic chips with metal lids. My chips are ceramic, but have no metal lids. Other all-plastic chips are also available. Even in the metal-cap chips, the error rate is negligible for most purposes. But there's plenty of board space left in case you want to put on a parity generator-checker, extra chip per row, etc., or if you want to add bank select.

So there you have it—48K for less than the cost of a 16K static board. Now you, too, can be Ex-Static! ■

Figure 1. *48K Dynamic RAM Chip Layout*

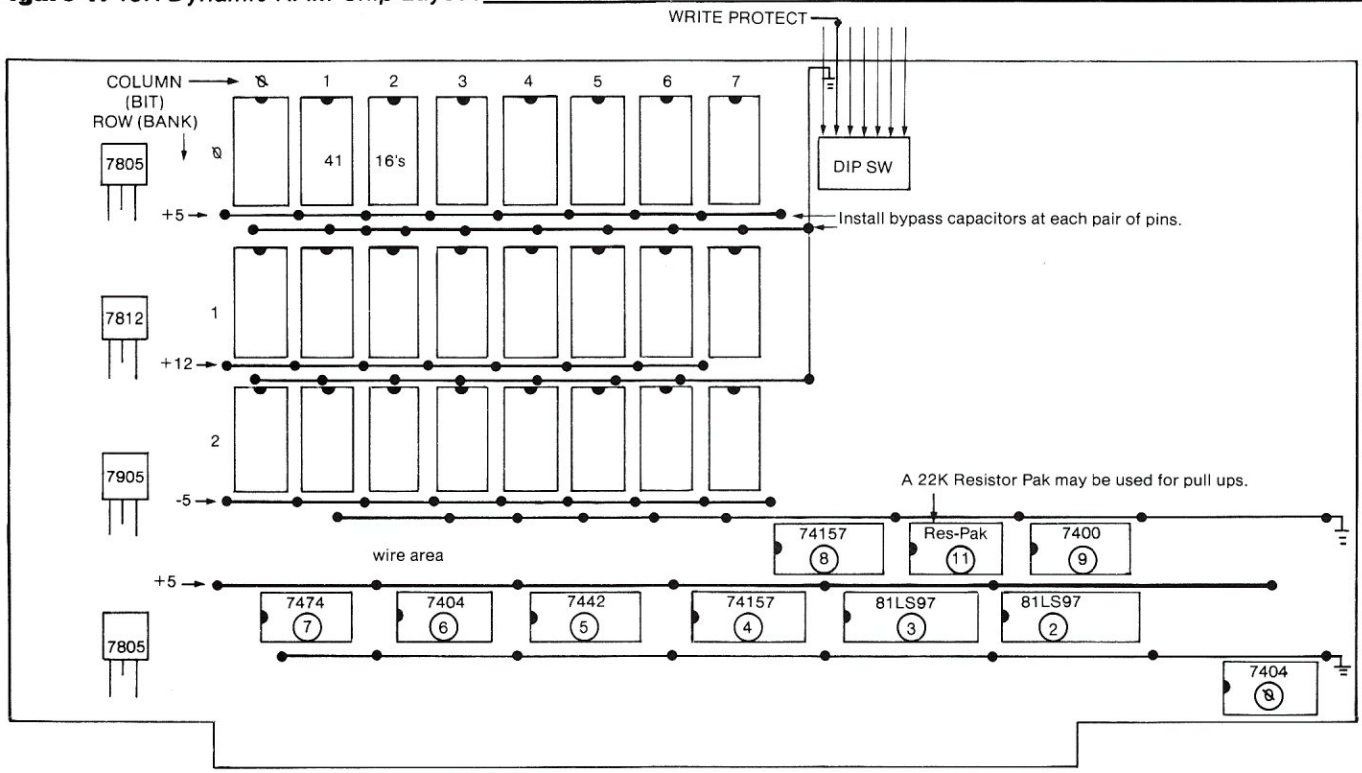
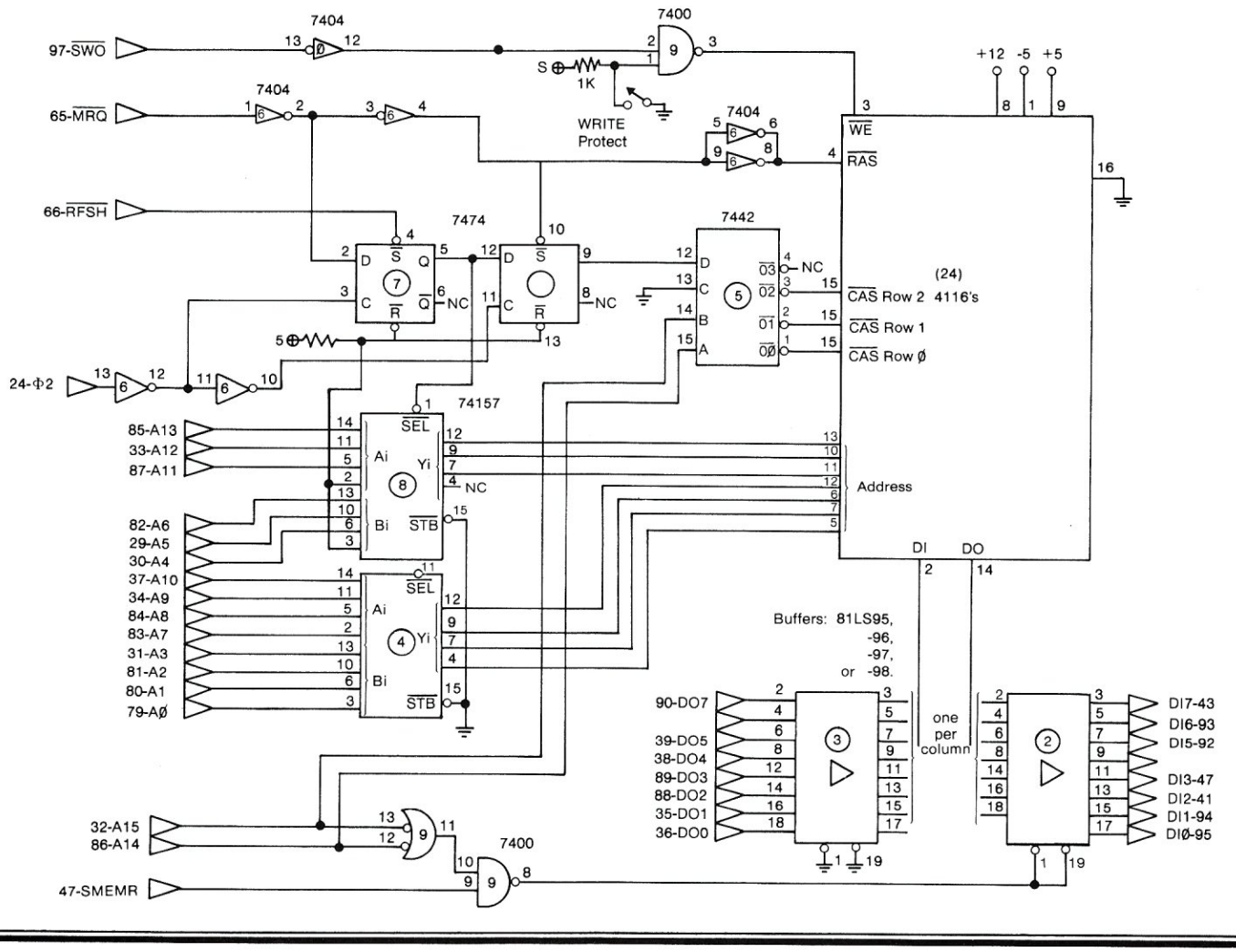


Figure 2.



The IEEE S-100/696 Standard— A Progress Report

by Sol Libes

The IEEE 696 (S-100) Committee met on June 30th to resolve the last objections, requests for changes, etc., prior to submitting the proposed standard to the IEEE Computer Standards Committee this month. Twenty members attended the meeting which began at 10AM and ended after 11PM. This meeting cleared the way for preparations of the final draft of the proposed standard.

The changes that were made to the standard as published in the January 1980 issue of *Microsystems* and the July 1979 issue of *Computer* magazines correct errors in the standard and revise the sections relating to multi-master operation, extended addressing and operation of the PHANTOM line. Further, the committee added an entirely new section to the standard (Section 5) titled "levels of compliance." Since some manufacturers have already begun to use these terms in advertising their products, we are publishing that entire section here. If every thing goes according to schedule, we will print the entire revised standard (as submitted for adoption) in the next issue of *Microsystems*.

5) Levels Of Compliance

This section presents the concept and notation of levels of compliance with the 696 Bus Standard as follows:

(1) Variable Elements of capability composing the essence of 696 Bus standard compliance.

(2) General discussion of compliance relationship for masters and slaves.

(3) Notation for describing level of compliance with the 696 Bus Standard.

The notion of levels of compliance is introduced to facilitate the use of 696 Bus products of varying capability manufactured by diverse vendors. It bounds the variability allowed within the 696 Bus specification and provides a succinct and convenient notation for these variables.

5.1 *Variable Elements of Capability*

The 696 Bus is versatile, allowing systems to be constructed with boards of varying capability. The 696 Bus allows for variations in data path width, memory address patch width, I/O address path width, and interrupt attributes.

5.1.1 *Data Path*

The 696 Bus allows for both 8- and 16-bit data path products, thus allowing the 8- and 16-bit products to work together.

5.1.2 *Memory Address Path*

The 696 Bus Standard designates a 16- or a 24-bit address path.

5.1.3 *I/O Address Path*

The 696 Bus allows for both 8- and 16-bit I/O address paths. The 16-bit path products must also be configurable to act as 8-bit path products.

5.1.4 *Interrupt Attributes*

The 696 Bus allows for considerable variety in interrupt attributes. All permanent masters must respond to the INT* line. Temporary masters must respond to the INT* line, or else not hold the bus indefinitely. A product may also support vectored interrupts and/or have the ability to generate INTA Bus cycles.

A master may support any of the above interrupt handling methods. It is necessary to configure the system such that the sources of the interrupt request correspond to the interrupt handling of the masters.

5.2 *Masters and Slaves*

When contructing 696 Bus systems, it is not necessary that all modules have identical capabilities. One may, for instance, have a master with an 8/16-bit data path and a slave with an 8-bit data path. The system is completely functional, though the application must restrict itself to 8-bit accesses to that slave.

The key concept when constructing a 696 Bus system is that of required capability versus supplied capability. Each product will provide some set of capability. A transaction between two such products will be restricted to that capability which is the intersection of the sets of capability of the two products. In some cases, the intersection may be null, implying fundamental incompatibility. It is the responsibility of the system designer to assure the viability of this intersection.

5.3 *Compliance Level Notation*

A notation is introduced which allows a vendor to succinctly and accurately specify a product's level of compliance with the 696 Bus standard. For boards which may act as either masters or slaves, the compliance levels must be specified for both cases. Increasing levels of compliance subsume lesser levels for data path width, memory address path width and I/O address path width.

The default compliance levels are shown in parentheses and may be omitted from the compliance level specification (5.4).

5.3.1 *Data Path*

(D8) Represents an 8-bit data path
D8/D16 Represents an 8- or 16-bit data path
D16 Represents a 16-bit data path

5.3.2 *Byte Significance for 16-Bit Transfers*

(LO) Represents least significant byte at odd address
LE Represents least significant byte at even address

5.3.3 *Memory Address Path*

(M16) Represents a 16-bit memory address path
M24 Represents a 24-bit memory address path

5.3.4 *I/O Address Path*

(I8) Represents an 8-bit I/O address path
I16 Represents an 8- or 16-bit I/O address path

5.3.5 *Interrupt Attributes*

5.3.5.1 *Slave Interrupt Attributes*

(NI) Represents no support for interrupts
VI Represents support for vectored interrupts
AI Represents support for interrupt acknowledge cycles

5.3.5.2 *Master Interrupt Attributes*

(NI) Represents no support for interrupts (default for temporary masters)
(II) Represents support for INT* (default for permanent masters)
VI Represents support for vectored interrupts
AI Represents support for interrupt acknowledge cycles

5.3.6 *Bus Arbitration Capability of Masters*

(PM) Represents Permanent Master Only. No Bus Arbitration Logic
TM Represents Temporary Master Only. Has Bus Arbitration Logic and Bus Transfer Logic
P/TM Represents a device that can function as the permanent or as a temporary master. Has Bus Arbitration Logic and Bus Transfer Logic.

5.3.7 *Bus Clock Frequency*

Fx Where x is the maximum allowable frequency for 0 is megahertz.
(F2) The default is 2MHz.

5.3.8 *Access Time for Slaves and Masters*

Tx Where x is the worst case access time, not including any wait states, in nanoseconds, (t ACC in Table 4). There is no default for access time, it must be specified for all products.

5.3.9 *Wait States for Slaves*

(W0) Represents no ability to generate wait states.
WX Represents the ability to generate X wait states. X may also be

5.3.10 *Board Height*

(SH) Represents standard height boards (5 in).
HH Represents high height board (10 in, 10.125 in max).

5.3.11 *An Example*

A combination I/O and memory slave board which supports an 8/16 data path, a 24 bit memory address, 8 or 16 bit I/O address, is standard height, and has a t ACC of 150 ns would be specified as follows:

696 Bus Compliance: Slave D8/16 M24 I16 T150

5.4 *Compliance Labeling Specifications*

Compliance labeling specifications of a board shall be clearly marked on the board, the board's documentation and all advertisements for the board. ■

NorthStar Topics

DOS/BIOS Directory And File Conversion In North Star UCSD Pascal

by Chris Young

The first part of this article describes the procedure for creating a UCSD disk volume directory from a North Star DOS directory under North Star's implementation of UCSD Pascal. Another program writes a North Star DOS directory on a UCSD Pascal disk. The third program produces a directory listing of a DOS disk from Pascal.
Part II, which will be presented in the November/December issue, describes the procedure for converting North Star data and text files into UCSD data and text under North Star's implementation of UCSD Pascal.

The UCSD Pascal programming system is famous because it has been implemented on a wide variety of computers. One of those implementations is for the popular North Star Horizon computer and its MDS-A mini-disk system for S-100 computers. Programs in North Star Basic, although not compatible with most Basics, have become widely available through articles published in major computing magazines. With the introduction of UCSD Pascal for North Star, users of North Star products have an even more powerful program development system. Because of the extreme portability of UCSD Pascal, even more software will be available. North Star users now are also relieved of the task of converting software from one language to another as required for programs written in a Basic which is not compatible with North Star Basic.

There is one conversion problem still remaining for users of North Star Pascal. How does one access data used with their old North Star operating system? Such data might consist of data bases, numerical tables, text files, or "Tiny Pascal" programs which the user wishes to run in UCSD Pascal.

Because the UCSD Pascal is designed to be portable, it uses standardized file and disk directory formats. These formats, as one might expect, are not directly compatible with North Star DOS (Disk Operating System). An entire new set of disk software called BIOS (Basic I/O System) is provided by North Star. The first part of this article deals with a procedure to convert directories from DOS format to UCSD Pascal's BIOS format and back again automatically. Three Pascal programs to aid in conversion are discussed. DOSTOBIOS converts a North Star Basic/DOS format disk into a Pascal/BIOS readable disk. BIOSTODOS reads a BIOS directory and creates a North Star DOS directory on the disk. DOSCAT gives a catalog listing of a DOS directory from Pascal. The second part of this article, which will be published in the next issue of *Microsystems*, deals with the conversion of the files' contents once the user has gained access to them through the converted directory.

These methods work for North Star UCSD Pascal I.5, Version 1, DQ-Release 2 and 3. The procedure assumes you are using Release 5.1 DQ DOS and one or two double density drives. The user must adapt this method to the system he has available. This will eliminate the need for such cumbersome phrases such as "except in single density" or "track 0 through 69 in double sided" in this discussion. Some of the procedures described will not work with earlier versions or other configurations of this system. I did not have access to other configurations and cannot speak with any authority about them.

Part I: Directory Conversion

Disk Sector Allocations

North Star software and data reside on tracks 0 through 34 of a ten sector-per-track mini-disk. Each double density sector contains 512 bytes of information. North Star allocates sectors 0 through 3 for the DOS directory (see Table 1). If the disk is to be used as a bootstrap disk, sectors 4 through 9 must contain a boot program. Otherwise this area is used for data. Pascal usually does not access this area because UCSD Pascal has its own plans for the first ten sectors.

In order to allow ease of customizing I/O routines, North Star has decreed that track 0 (sectors 0-9) be reserved for their normal DOS purposes. BIOS is tricked into thinking that physical tracks 1 through 34 are really tracks 0 through 33. Special "undocumented" procedures are required to make physical track 0 accessible to BIOS. DOS always has access to the entire disk. Note that BIOS does not maintain the North Star directory nor does DOS and its associated utilities maintain BIOS directories.

BIOS uses logical track 0 (physical track 1) for its directories and other purposes. To allow BIOS to access the DOS directory, we must move the data on tracks 0

Chris Young, 3119 Cossell Dr., Indianapolis, IN 46224.

through 32 down to tracks 2 through 34 of the same or another disk. Why not 1 through 34? Because physical track 1, although accessible to BIOS, must be kept "clean" so that the newly created BIOS directories do not destroy any DOS data. This means that any valuable data on tracks 33 and 34 must be moved to other disks before the conversion can continue.

Table 1. *Disk Sector Allocations.*

		DOS/BASIC		BIOS/PASCAL	
Physical Trk.	Sec.	Logical Blk.	Use	Logical Blk.	Use
0	0	0,1			
	1	2,3	[Directory]	[N O T]	
	2	4,5	[ ]	[ ]	
	3	6,7		[N O R M A L L Y]	
	4	8,9		[ ]	
	5	10,11	[Boot]	[A C C E S S I B L E]	
	6	12,13	[program]	[ ]	
	7	14,15	[or Data]	[B Y P A S C A L]	
	8	16,17	[areas]	[ ]	
	9	18,19			
1	10	20,21		0	[N O T]
	11	22,23	[Data]	1	[U S E D]
	12	24,25	[areas]	2	
	13	26,27	[ ]	3	[Primary]
	14	28,29	[ ]	4	[directory]
	15	30,31	[ ]	5	
	16	32,33	[ ]	6	
	17	34,35	[ ]	7	[Back-up]
	18	36,37	[ ]	8	[directory]
	19	38,39	[ ]	9	
2	20	40,41	[ ]	10	
	21	42,43	[ ]	11	[New DOS dir-]
	22	44,45	[ ]	12	[ectory loc.]
	23	46,47	[ ]	13	
	24	48,49	[ ]	14	
	25	50,51	[ ]	15	[New DOS data]
	26	52,53	[ ]	16	[areas]
	27	54,55	[ ]	17	[ ]
	:	:	:	:	:
	:	:	:	:	:
32	328	656,657	[ ]	318	[ ]
	329	658,659		319	[ ]
33	330	660,661		320	[ ]
	331	662,663	[Not moved]	321	[ ]
	:	:	[by DOSTO-]	:	[:]
	:	:	[BIOS. Must]	:	[:]
34	348	696,697	[be empty.]	338	[ ]
	349	698,699		339	

Using The DOSTOBIOS Program

Listing 1 is a Pascal program called DOSTOBIOS that is used to create a Pascal/BIOS directory on a Basic/DOS disk. Before processing, some re-arranging of files should be done. All files in tracks 33 and 34 should be moved to other disks. Note also that BIOS only allows 77 files on a disk volume. If more exist, they must be copied to a separate disk, and deleted before conversion. The disk to be converted (which we will call the "SOURCE" disk) should be COmpacted using the DOS CO utility. Note that CO requires that no files overlap. This is also a requirement of the Pascal conversion process. Another disk called "DEST" should be initialized using the DOS "IN" command. DEST may be the same disk as SOURCE. If so, do not use the "IN" command. It is not advised to have SOURCE and DEST the same. If something goes wrong, you may ruin the DOS directory and leave the disk in an indeterminant state.

To copy the data from the SOURCE (i.e. DOS) disk to DEST (i.e. BIOS) disk, eX(ecute the program DOTOBIOS. Enter the unit numbers of SOURCE and DEST. The data is moved, the DOS directory is sorted, and the new BIOS directory is created. The sorted DOS directory is rewritten with corrections made for the new data locations.

At this point, the files may be read by Pascal programs as files which have a type of "Datafile." They are most easily accessed using the BLOCKREAD Pascal intrinsic with the files RESET as untyped files. The various F(iler functions such as T(ransfer, R(emove, K(runch, and C(hange may be used on the files, and other files may now be safely written on the disk. The problem of converting data within the files is highly application dependent. In Part II of this article we will discuss some Pascal PROCEDUREs and FUNCTIONs to ease the actual data conversions. We will discuss some of the internal workings of DOSTOBIOS.

Accessing Inaccessible Blocks

The key to accessing track zero from BIOS is an "undocumented" feature of the UCSD Pascal intrinsics UNITREAD and UNITWRITE. By using special parameters to these routines, the one track offset (ten blocks) which maps logical tracks into physical tracks is eliminated. Normally, a call to UNITREAD would look like this:

```
UNITREAD(UNITNUM,BUFFER,NUMBYTES,BLOCKNUM,TRANSCODE);
```

UNITNUM is the integer unit number to be read. BUFFER is a packed array of char. NUMBYTES is the integer number of bytes to be transferred. BLOCKNUM is the integer number of the block to be read. TRANSCODE is an integer transfer code which usually selects synchronous or asynchronous transfer. The special parameters look like this:

```
UNITREAD(UNITNUM,BUFFER,0,BLOCKNUM,2);
```

By setting NUMBYTES to zero and TRANSCODE to 2, the procedure will transfer 512 bytes from the physical block pointed to by the value in BLOCKNUM. According to a staff technical consultant at North Star, this works for Pascal-DQ Release 2 and 3. Tests of the DOSTOBIOS programs have verified that this works for the DQ version. This does not work for Pascal-S (single density version) since earlier versions allow negative block numbers for BLOCKNUM. Current versions do not allow negative BLOCKNUM.

The procedure MOVIT in DOSTOBIOS uses the TRANSCODE of 2 to copy tracks 0 through 32 down to 2 through 34. The tracks are copied last to first so that SOURCE and DEST can be the same disk.

BIOS Directories

BIOS directories as defined in these programs have two types of entries: HEAD records and FYLE records. A directory contains one HEAD record and up to 77 FYLE records. The HEAD describes the entire volume while the FYLEs describe each file.

HEAD has seven fields. PSTRTBLK is the integer starting block number of the directory file. This is always zero for HEAD records. PNEXT is the integer starting block number of the next file (i.e. last block + 1). PNEXT - PSTRTBLK = the length of the file in 512 byte blocks. For HEAD entries (with backup directory) PNEXT is 10. PTYP is the type of entry which is zero for HEAD entries. PFILNAM is the volume name. It is a "string[7]" or a "packed arrary[0..7]" with PFILNAM[0] as a length byte.

North Star Pascal Release 1,2 and 3

I recently received a copy of the UPGRADE: diskette for North Star Pascal. One of the programs on North Star's UPGRADE diskette creates a North Star directory reflecting the contents of the Pascal directory. This is the same function as Chris Young's program presented in Listing 2. Since the source code of the North Star program is available on the UPGRADE diskette, I compared Chris's program with the North Star program. I was surprised to find a significant difference in the method used to access the system track (track 0) on the Pascal diskette. The DOSDIR program on the UPGRADE: diskette accesses the system track by using a negative value for the BLOCKNUM parameter in the UNITREAD and UNITWRITE commands. In Chris's program, the physical block number of the desired block is used for the BLOCKNUM parameter, along with NUMBYTES=0 and TRANSCODE=2. Chris refers to this technique as a undocumented method obtained from a staff technician at North Star. I also noticed that Chris documented his Pascal as version 1, release 2.

I contacted North Star to find out what had been done to Pascal version 1, release 1 (my release) to allow Chris's version to work. Initially, all I got was confusion. I contacted Dave Gersen after he returned from vacation and he was able to supply a somewhat rational explanation. Dave Gersen promised to send me a memo he had prepared to clear up the confusion over the North Star Pascal product. Briefly, here is Dave's summary:

There have been three releases of version 1 (UCSD Pascal version 1.5). Release 1 was first sold in early 1979 (before Dave's time at North Star) and was very cheap, costing less than $100 for two diskettes and the manual. The UPGRADE: diskette was released in December of 1979 to provide two programs that were omitted from release 1, LIBRARIAN.CODE and LIBMAP.CODE. The additional utility programs on the UPGRADE: diskette (DOSDIR.CODE, IN.CODE and CD.CODE) were mistakenly included. In early 1980, North Star offered version 1, release 2 for about $199. North Sar felt release 1 was underpriced, and the royalties North Star had to pay did not allow for much profit. Here is where the story gets a bit fuzzy.

During the summer and fall of 1979, North Star hired a contract programmer to review the Pascal BIOS and 'improve' it. This person made changes that North Star was unaware of; namely the handling of the BLOCKNUM parameter in the UNITREAD and UNITWRITE intrinsics. Release 2 was followed quickly by the current release 3. Release 3 is very similar (only four bytes changed) and fixes a bug with quad access. ■

BASEX MEANS SPEED!

BASEX is a fast, easy to learn language for 8080, Z80, or 8085 microcomputers. Its commands resemble BASIC, making translation easy. An interactive compiler permits you to enter, list, edit and run programs up to 10x faster than similar BASIC programs and use half the memory (2K plus program).

Powerful features include:

* Array variables
* 16 Bit Arithmetic/Logic
* Variable name length
* Named subroutines with multiple arguments
* Text strings
* Versatile I/O Functions
* Block memory searches/ transfers
* Custom commands easily added

CHOOSE YOUR BASEX . . .

* 97-page BASEX manual, pub. by Byte Books (Includes source listing. Required for use with disks or tapes below.)	$ 8
* Sorcerer/SOL/Poly 88/Meca Alpha/ Heath H8/cassette	$25
* TRS-80® Level II, 16K tape with graphics commands	$25
* North Star Disk, with disk handler commands	$35
* CP/M® Disk, with disk handler commands	$35
* NEW! My Computer Speaks BASEX, pub. by Hayden (a 128-page primer that contains many examples and detailed explanations).	$10

TRS-80 is a trademark of Tandy Corp.; CP/M is a trademark of Digital Research.

Add $ .75 shipping (special 4th class) or $1.50 special handling or UPS.
See BASEX at your local dealer or order direct from

iwi INTERACTIVE MICROWARE, INC. DEALER INQUIRIES INVITED
P.O. Box 771, Dept. State College, PA 16801
Call (814) 238-8294 for SPEEDY ACTION

TI 820 KSR PACKAGE

FULL OPTIONS

NEW $1737

BRAND NEW IN BOX WITH 112 PAGE MANUAL

Large OEM overstock results in a limited quantity offer of this super reliable, feature packed terminal at a below wholesale price.

FULL (TI item #101) OPTION "PACKAGE": Device forms control, compressed & expanded print, 18 key numeric pad, full ASCII. We count more than 100 user selectable options & features. This virtually assures compatibility with your application.

PARTIAL LIST OF FEATURES

PRINTER: 150 CPS; 9X7 dot matrix; 128 ASCII characters including control chars. (great for debug); forms control; double-width and normal, compressed and standard prts. (4 styles) to 218 chars./line; bidirectional; 1280 char. buffer; V & H tabs; 3"-15" tractor feed; 6 & 8 lines per inch; sgl/dbl spacing; last char. visibility; skip perf.; auto. CR-LF; programmable new-line response.

KEYBOARD: Full ASCII; 18 key num. pad; prog. CR and ENTER keys; control panel with numeric display; auto repeat; N-key rollover; pull-out operator's reference cards, deep dish keys.

GENERAL: Full blown RS-232; 110-9600 baud; EIA & DC3/DC1 busy; Bell 103, 113, 202 & 212A modes; local/standby/off-line modes; non-volatile option/forms/21 char. prog. answerback memory; parity select.; 6' EIA cable; FDX/HDX/HDX w/reverse channel.

PRICING: List price for this configuration is $2420. Our price, brand new in original box, guaranteed working to full specifications, $1737 FOB Morris Plains, NJ. COD and credit cards on approval. NJ add sales tax. Call or write for full specifications.

ELECTROVALUE INDUSTRIAL INC.
P.O. BOX 157
MORRIS PLAINS, NJ 07950
Formerly Electravalue Industrial

201/267-1117

MICROSTAT

Microstat is an advanced statistics package designed for use in research, education and industry. Microstat is a file-oriented statistics package with a Data Management Subsystem (DMS) that creates the data files plus the ability to: edit, list, destroy, delete cases, augment, sort, rank-order, lag, move, merge and transform the data. The data transforms include: add, subtract, multiply, divide, reciprocal, log, natural log and antilog, exponential, linear transformations plus adding any number of variables to create new variables.

Once the file is created, it can be used to produce: Descriptive statistics, Hypothesis tests (mean and proportion), ANOVA (one-way, two-way and random blocks), Scatterplots, Frequency distributions, Correlation analysis, Simple and Multiple regression, Time Series, Nonparametric tests (11 of them), Crosstabs and Chi-square, Factorials, Permutations, Combinations, and 8 Probability distributions.

The price of Microstat is $250.00 and the user's manual is available for $20.00 and includes sample printouts. Since the printouts reference standard statistics textbooks and journal articles, you can compare the accuracy of Microstat to results produced on much larger systems. No other statistics package seems to have the confidence to do that . . . at any price.

Microstat is available for the North Star DOS and Basic, Microsoft's Basic-80™ (5.03 or later) and Compiler Systems' CBasic2™. Please specify 8" SD (soft-sectored) or North Star 5¼" disk when ordering.

ECOSOFT
P.O. Box 68602
Indianapolis, IN 46268
(317) 283-8883

master charge
THE INTERBANK CARD

INTERCHANGE

If you use the CP/M™ operating system, life just got a whole lot easier for you. Interchange is a Z-80™ assembly language program that gives you all of the features that PIP doesn't, plus several unique features. Some of the features of Interchange include:

DIR, in the usual fashion, *plus* listing all files *excluding* those with a specified character. Read/write status is also given.

ERA, as usual plus *exclusive* erases. In addition, a "Q" switch can be used to query on each erase, a "W" allows erases of R/O files without query (normally you are queried), and an "R" switch if system files are to be included.

LIST permits printer listings with formatting controlled by TAB, WIDTH, LINES and WRAP. If you are using the QT Systems Clock Board, listings include the date and time.

COPY including exclusive copies and the optional "Q", "W" and "R" switches plus an "E" switch that queries if the file already exists. It also allows for changing disks in the middle of a copy if either the disk or directory become full. It automatically verifies copies.

STAT, with ambiguous, unambiguous and exclusive listings. It produces an alphabetized listing and includes each file length, total directory entries and space used and unused.

Other commands include RENAME (including ambiguous), HELP, START, END, CLEAR, RESET, DATE, TIME, TAB, WIDTH, LINES, WRAP, QT, SETIT and TYPE. Once you've used Interchange, we doubt that you'll ever use PIP again. The price of Interchange is $59.95 and the manual is available for $10.00. Orders must be accompanied with your CP/M serial number. Interchange is recommended for a 32K or larger system and will not run with an 8080 CPU. At the present time, only User 0 is supported.

CBasic2 is a registered trademark of Compiler Systems.
CP/M is a registered trademark of Digital Research.

FILINVOL is the integer number of FILes IN the VOLume. DATEINIT is the date of the volume was initialized for data disks, and is the current date for system disks. Dates are of the following form:

```
type DATES=packed record
                MON:0..11;
                DAY:0..31;
                YEAR:0..99;
               end;
```

For our purposes, DATES can be declared as integers. TIME is the time of last access in most UCSD Pascal implementations. Because North Star has not implemented a real-time clock, this field should be ignored. The remaining space in HEAD records is unused.

The other kind of entry is called FYLE (because FILE is a reserved word, it could not be used). In FYLE entries, PSTRTBLK, PNEXT, and PTYP are similar to HEAD entries. They are the start block, next start block, and file type of each files. See Table 2 for file types. PFILNAM is the file name which is a "string[15]" or a "packed array[0..15] of char" with PFILNAM[0] as a length byte. LASTBYTES is the integer number of bytes in the last block which is always 512 in our application. DATE is the creation date of the file.

Table 2. *UCSD Pascal File Types.*

Code #	Type
1	Bad disk
2	Codefile
3	Infofile
4	Textfile
5	Datafile
6	Graffile
7	Fotofile
8 up	ILLEGAL

Unlike DOS, there are no blank entries in a BIOS directory. FILINVOL tells exactly the number of files, and likewise the number of FYLE entries in the directory. FYLE entries in BIOS always occur in the same order as the files actually reside on disk. For this reason, the DOS entries are sorted in increasing order with PSTRTBLK as the key. If the value of PASDIR[I+1].PNEXT is not equal to PASDIR[I].PSTRTBLK for some value "I" then there exits a blank space on the disk with a length which is the difference of the two.

The variable definitions for PASENTRY, PASDIRECT, DOSENTRY, and DOSDIRECT use an advanced feature of Pascal which is called "variant record definitions." It allows us to redefine a buffer with several different record formats similar to a "REDEFINES" clause in COBOL. For example, PASDIRECT is a buffer 2048 bytes long. We may refer to it as PASBFR which is "packed array[0..2047] of char," or as PASDIR which is "packed array[0..77] of PASENTRY." Each PASENTRY is one directory entry. Further, we define a PASENTRY as one of two types of records either HEAD or FYLE. To access, for example, the date the volume was initialized use:

```
PASDIRECT.PASDIR[0].HEAD.DATEINIT
```

That is: the variable PASDIRECT, as the array PASDIR, element zero, with HEAD record format, in the DATEINIT field. As another example, the type of the tenth file is:

```
PASDIRECT.PASDIR[10].FYLE.PTYP
```

That is: the variable PASDIRECT, array PASDIR, element 10 FYLE record format, and PTYP field.

The Pascal statement "with" allows us to narrow down the part of a record that we are working with. For example:

```
begin
 ...
  DOSDIRECT.DOSDIR[I].DSTRTBLK:=DOSDIRECT.DOSDIR[I].
  DSTRTBLK+20;
 ...
end;
```

can be replaced by:

```
with DOSDIRECT.DOSDIR[I] do
 begin
  ...
   DSTRTBLK:=DSTRTBLK+20;
  ...
 end;
```

The reason that such powerful constructs as variant records are required is that the UNITREAD and UNITWRITE intrinsics *must* operate on packed arrays of chars. We could pick out sections of the buffers and process them character by character as one might do in Fortran or Basic. However, it is easier to "overlay" arrays of records and access them in a more readable form. For details on use of variant record specifications and the Pascal "with" statement, see *Pascal Users Manual and Report* by Jensen and Wirth, or the UCSD Pascal manual.

After all entries have been converted and copied into the Pascal buffer, the buffer is written into blocks 2 through 5. Then it is written into 6 through 9. These areas are the primary and duplicate directories respectively. Any old BIOS directories on DEST are overwritten. The DOS directory is updated to reflect the new locations of the files. It is written to DEST. If SOURCE is different from DEST, then the directory on SOURCE is not touched. If SOURCE = DEST, the original DOS directory is overwritten.

The process may be verified by entering the F(iler and taking an E(xtended directory listing and comparing it to a DOS directory listing. All files should have starting block numbers which are ten greater than they had in DOS. File lengths are listed in 512 byte blocks so they will appear as half their original DOS values. File names are of the form < filenam >.DOS where < filenam > is the original DOS name.

Using the BIOSTODOS Program

Occasionally the user may wish to create a DOS directory from a Pascal/BIOS directory. The program BIOSTODOS assists in this process. Because BIOS file names can be up to fifteen characters long, while DOS

names are limited to eight, the user must supply DOS names for the files. All other aspects of the conversion are automatic.

To create a DOS directory eX(ecute BIOSTODOS. (See Listing 2.) The user is first prompted by "Unit #" which requests the unit containing the disk to be converted. Unit must be a number; volume names or abbreviations such as "*" or ":" are not allowed. The user is then prompted by:

```
"Type DOS directory file name:"
```

This requests the name of a file which occupies track 0. Usually this file indentifies the name or ID number of the disk. Next, a file containing the Pascal/BIOS directory is created. The user is prompted for the file name by:

```
"Type Pascal directory file name:"
```

Each file name from the BIOS directory is then printed. After each, a type a new DOS name for the file. All file names must be one to eight characters long, and must be unique. After all files have been given names, the user is asked:

```
"Update DOS directory?"
```

A response of "Y" or "y" will write the DOS directory and destroy any previous DOS directory. Any other response aborts the program and leaves everything intact. The BIOS directory is always left intact.

BIOSTODOS uses the same variables and record descriptions as DOSTOBIOS. The special parameters to UNITWRITE are used to access the DOS disk area. A function called GETNAME is used to input the new DOS names and check them for uniqueness. GETNAME returns the integer length of the name. This also doubles as an error flag. In the event of an error in GETNAME, a value of nine is returned. GETNAME has one variable paramter NAME. NAME is a "packed array[0..7]of character." Because the syntax for < parameter list > requires a < type identifier >, a new type PA07OC is defined as "Packed Array [0..7] Of Char." BIOSTODOS creates all file entries as double density type 0 files. The DOS start block number is ten more than in BIOS, and the length is computed by:

```
DFILLEN:=PNEXT - PSTRTBLK;
```

Note that no files are actually moved. The directory is created, written, and that's all.

Using The DOSCAT Program

Another useful program can be found in Listing 3. DOSCAT produces a standard DOS directory catalog listing from the USCD operating system. To use DOSCAT, eX(ecute DOSCAT, and type the unit number the disk is in. As in BIOSTODOS, this must be a number. NUMLINES determines the number of files that will fit on a CRT screen. After the screen is full, type space to continue.

The internal workings of DOSCAT are rather straightforward. The procedure WRITEHEX is of interest. It takes an integer mod 256 and outputs it as a two "digit" hexadecimal number. WRITEHEX is used to ouput start addresses for DOS type 1 machine code files. ■

Listing 1.

```
     (* Program to write a U.C.S.D. Pascal/BIOS directory *)
     (*  on a DOS disk, and to move data to areas easily  *)
     (*  accessible to Pascal.                            *)
           (* Written Aug. '80                    *)
           (*      by Chris Young                 *)
           (*         3119 Cossell Drive          *)
           (*         Indianapolis IN 46224       *)
           (*            (317)-291-5376           *)
program DOSTOBIOS;
 const DFFTYPE=5;  (* Pascal file type "Datafile" *)
       MAXFILS=77; (* Maximum number of PASENTPYs *)
       PHYSICAL=2; (* Transfer code for physical blocks *)
       LOGICAL=0;  (* Transfer code for logical blocks  *)
 type DOSENTRY=packed record
                DFILNAM:packed array[0..7] of char;
                DSTRTBLK:integer;
                DFILLEN:integer;
                DTYP:packed array[0..3] of char;
               end;
      PASFNTRY=packed record
                case integer of
                 0:(HEAD:packed record
                          PSTRTBLK:integer; PNEXT:integer;
                          PTYP:integer;
                          PFILNAM:packed array[0..7]of char;
                          BLKINVOL:integer;
                          FILINVOL:integer;
                          TIME:integer;
                          DATEINIT:integer;
                         end);
                 1:(FYLE:packed record
                          PSTRTBLK:integer; PNEXT:integer;
                          PTYP:integer;
                          PFILNAM:packed array[0..15]of char;
                          LASTBYTES:integer; DATE:integer;
                         end);
               end;(*PASENTRY*)
 var SOR,DEST,DCOUNT,PCOUNT,NAMLEN,DINDX,I:integer;
     CH:char;
     DOSTAG:packed array[0..3]of char;
     DOSDIRECT:packed record
          case integer of
           0:(DOSDIR:packed array[0..127]of DOSENTRY);
           1:(DOSBFR:packed array[0..2047] of char);
          end;
     PASDIRECT:packed record
          case integer of
           0:(PASDIR:packed array[0..MAXFILS]of PASENTRY);
           1:(PASBFR:packed array[0..2047] of char);
          end;
     (* Function which sorts DOSDIR in increasing order with *)
     (*  DSTRTBLK as a key.  Returns with integer count of   *)
     (*  the number of files in DOSDIR.                      *)
   function SORT:integer;
    var FILCOUNT,I,J,K:integer;
        DNTRY:DOSENTRY;
   begin
    WRITE('Sorting directory');
    with DOSDIRECT do
     begin
      FILCOUNT:=0;
        (* Eliminate blank and zero length files.  *)
      for I:=0 to 127 do
       if (DOSDIR[I].DFILNAM[0]=' ') or (DOSDIR[I].DFILLEN=0)
        then DOSDIR[I].DSTRTBLK:=9999;
      for I:=0 to 127 do
       begin
        K:=I;
        for J:=I to 127 do
         if DOSDIR[J].DSTRTBLK < DOSDIR[K].DSTRTBLK then K:=J;
        DNTRY:=DOSDIR[I];
        DOSDIR[I]:=DOSDIR[K];
        DOSDIR[K]:=DNTRY;
        if DOSDIR[I].DSTRTBLK <> 9999
         then FILCOUNT:=FILCOUNT+1;
        WRITE('.');
        if (I mod 32)=31 then WRITELN;
       end;(*for I *)
     end;(*with DOSDIRECT*)
     WRITELN;WRITELN('Sort complete.');
     SORT:=FILCOUNT;
   end;(*SORT*)
  (* Procedure to move all data from tracks 0 through 32  *)
  (*  to tracks 2 through 34.  Reads unit numbers for     *)
  (*  source and destination disks. Uses special transfer *)
  (*  code in UNITREAD and UNITWRITE so they access       *)
  (*  physical blocks instead of logical blocks.          *)
 procedure MOVIT;
  var TRACK:packed array[0..5120] of char;
      I,J:integer;
  begin
   repeat WRITE('Source unit #'); READLN(SOR);
    until (SOR=4) or (SOR=5);
   repeat WRITE('Destination unit #'); READLN(DEST);
    until (DEST=4) or (DEST=5);
   WRITE('Insert disks.  Are you ready? (Y/N):');
   READLN(CH);
   if (CH<>'Y') and (CH<>'y') then begin
                                    WRITELN('Exiting...');
                                    EXIT(DOSTOBIOS);
                                   end;
   WRITELN('Moving data');
   for I:=32 downto 0 do
    begin
     for J:=0 to 9 do
      UNITREAD(SOR,TRACK[512*J],0,I*10+J,PHYSICAL);
     for J:=0 to 9 do
      UNITWRITE(DEST,TRACK[512*J],0,I*10+J+20,PHYSICAL);
     WRITE('.');
    end;
   WRITELN; WRITELN('Data moved');
```

```
 end;(*MOVIT*)
begin
 WRITELN('DOSTOBIOS version 2.0');
 MOVIT;
 UNITREAD(DEST,DOSDIRECT.DOSBFR,2048,10,LOGICAL);
 UNITREAD(DEST,PASDIRECT.PASBFR,2048,2,LOGICAL);
 DCOUNT:=SORT;
 if DCOUNT > MAXFILS then begin
                           WRITELN('TOO MANY FILES');
                           WRITE('Hit <sp>');READ(CH);
                           EXIT(DOSTOBIOS)
                          end;
 DOSTAG:='.DOS';
 PCOUNT:=1;
 WRITE('Copying directory');
 for DINDX:=0 to DCOUNT do
  with PASDIRECT.PASDIR[PCOUNT].FYLE,
       DOSDIRECT.DOSDIR[DINDX] do
   begin
      (* Output some dots *)
    if (DINDX mod 50)=49 then WRITELN('.')
                         else WRITE('.');
      (* Skip blank and zero length records *)
    if (DFILNAM[0]<>' ') and (DFILLEN<>0) then
     begin
      NAMLEN:=SCAN(8,=' ',DFILNAM);
      MOVELEFT(*to right*)((*source*) DFILNAM[0], (*to*)
                          (*destination*) PFILNAM[1], (*for a *)
                          (*length of *) NAMLEN);
      MOVELEFT(*to right*)((*source*) DOSTAG[0], (*to*)
                          (*destination*) PFILNAM[NAMLEN+1],
                          (*for length of*) 4);
      NAMLEN:=NAMLEN+4;
      PFILNAM[0]:=CHR(NAMLEN);
      PSTRTBLK:=DSTRTBLK+10;
      PNEXT:=PSTRTBLK+DFILLEN;
      PTYP:=DEFTYPE;
      LASTBYTES:=512;
      DATE:=0;
      PCOUNT:=PCOUNT+1;
      (* Update DOS start block *)
      DSTRTBLK:=DSTRTBLK+20;
     end;(*if DFILNAM<>' ' and DFILLEN<>0*)
   end;(*for DINDX; with PASDIRECT,DOSDIRECT*)
      (* Create BIOS directory header record *)
     with PASDIRECT.PASDIR[0].HEAD do
      begin
       PSTRTBLK:=0;
       PNEXT:=10;
       PTYP:=0;
       MOVELEFT(*to right*)((*source*) DOSTAG[1], (*to*)
                           (*destination*) PFILNAM[1],
                           (*for length of*) 3);
       PFILNAM[0]:=CHR(3);
       BLKINVOL:=340;
       FILINVOL:=PCOUNT-1;
       DATEINIT:=0; TIME:=0;
      end;
     (* Create new directory files in DOS directory *)
     with DOSDIRECT.DOSDIR[0] do
      begin
       DFILNAM:='DOSDIR  ';
       DSTRTBLK:=0;
       DFILLEN:=10;
       DTYP[0]:=CHR(128);
      end;
     with DOSDIRECT.DOSDIR[1] do
      begin
       DFILNAM:='PASDIR  ';
       DSTRTBLK:=10;
       DFILLEN:=10;
       DTYP[0]:=CHR(128);
      end;
     (* Blank out unused file entries *)
     for DINDX:=DCOUNT+1 to 127 do
      with DOSDIRECT.DOSDIR[DINDX] do DFILNAM[0]:=' ';
     UNITWRITE(DEST,PASDIRECT.PASBFR,2048,2,0); (* Primary dir. *)
     UNITWRITE(DEST,PASDIRECT.PASBFR,2048,6,0); (* Back-up dir. *)
     (* Rewrite updated DOS directory *)
     for I:=0 to 3 do
      UNITWRITE(DEST,DOSDIRECT.DOSBFR[I*512],0,I,PHYSICAL);
end.
```

Listing 2.

```
                (* Program to write a DOS directory   *)
                (*    on a U.C.S.D. Pascal/BIOS disk. *)
                (*  Written Aug. '80                  *)
                (*      by Chris Young                *)
                (*         3119 Cossell Drive         *)
                (*         Indianapolis IN 46224      *)
                (*            (317)-291-5376          *)
(**)
program BIOSTODOS;
 const MAXFILS=77;  (* Maximum number of BIOS files  *)
       DOSTYPE=128; (* File type for new DOS entries *)
 type PA07OC=packed array[0..7]of char; (* see func. GETNAME *)
      DOSENTRY=packed record
                 DFILNAM:packed array[0..7] of char;
                 DSTRTBLK:integer;
                 DFILLEN:integer;
                 DTYP:packed array[0..3] of char;
                end;
       PASENTRY=packed record
                  case integer of
                   0:(HEAD:packed record
                           PSTRTBLK:integer; PNEXT:integer;
                           PTYP:integer;
                           PFILNAM:packed array[0..7]of char;
                           BLKINVOL:integer;
                           FILINVOL:integer;
```

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

Z80* Disk Software

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 Amidon, 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 5¼" CP/M (soft sectored single density)
for 5¼" North Star CP/M (single density)
for 5¼" North Star CP/M (double density)

BASIC I

A powerful and fast Z80 Basic interpreter with EDIT, RENUMBER, TRACE, PRINT USING, assembly language subroutine CALL, LOADGO for "chaining", COPY 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. An excellent choice for games since the precision was limited to 7 digits in order to make it one of the fastest around. $49.95/$15.

BASIC II

Basic I but with 12 digit precision to make its power available to the business world with only a slight sacrifice in speed. Still runs faster than most other 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, added math functions, and disk file maintenance capability without leaving Basic (list, rename, or delete). $179.95/$25.

ZEDIT

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

ZTEL

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

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

MACRO II

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

LINKER

How many times have you written the same subroutine in each new program? Top notch professional programmers compile a library of these subroutines and use a Linker to tie them together at assembly time. Development time is thus drastically reduced and becomes comparable to writing in a high level language but with all the speed of assembly language. So, get the new CDL Linker and start writing programs in a fraction of the time it took before. Linker is compatible with Macro I & 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 then be made before continuing. It's all under your control. You can even skip displaying a subroutine call and up to seven breakpoints can be set during execution. Use of Debug I can pay for itself many times over by saving you valuable debugging time. $79.95/$20.

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. This 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 I/O ports, hex math, breakpoint, execute, move, fill, display, read and write in Intel or binary format tape, and more! on disk $34.95/$15.

APPLE

8080 version of Zapple $34.95/$15.

NEW! TPM now available for TRS-80 Model II!

TPM*

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

SYSTEM MONITOR BOARD (SMB II)

A complete I/O board for S-100 systems. 2 serial ports, 2 parallel ports, 1200/2400 baud cassette tape interface, sockets for 2K of RAM, 3-2708/2716 EPROM's or ROM, jump on reset circuitry. Bare board $49.95/$20.

ROM FOR SMB II

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

PAYROLL (source code only)

The Osborne package. Requires C Basic 2.
5" disks $124.95 (manual not included)
8" disks $ 99.95 (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" $99.95 (manual not included)
Manual $20.00

GENERAL LEDGER (source code only)

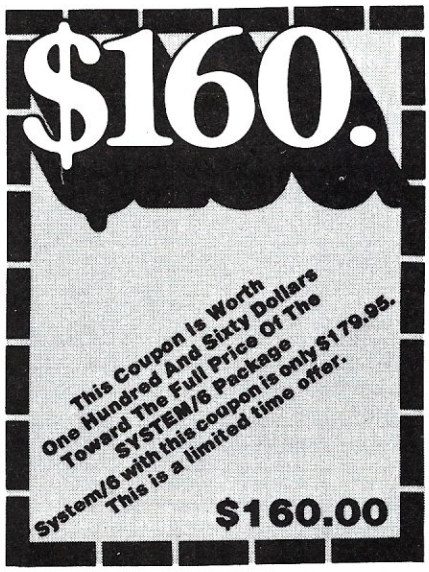
By Osborne. Requires C Basic 2
5" disks $99.95 (manual not included)
8" disks $99.95 (manual not included)
Manual $20.00

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 I)
2. Media (e.g. 8" CP/M)
3. Price and method of payment (e.g. C.O.D.) include credit card info. if applicable.
4. Name, Address and Phone number.
5. For TPM orders only: Indicate if for TRS 80, Tarbell, Xitan DDDC, SD Sales (5¼" or 8"). ICOM (5¼" or 8"), North Star (single or double density) or Digital (Micro) Systems.
6. N.J. residents add 5% sales tax.

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

For information and tech queries call
609-599-2146

For phone orders ONLY call toll free
1-800-327-9191
Ext. 676
(Except Florida)

OEMS

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

* 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*
* CP/M is a trademark of Digital Research
Prices and specifications subject to change without notice.

DEALER INQUIRIES INVITED.

COMPUTER DESIGN LABS

342 Columbus Avenue
Trenton, N.J. 08629

```
                        TIME:integer;
                        DATEINIT:integer;
                       end);
                1:(FYLE:packed record
                        PSTRTBLK:integer; PNEXT:integer;
                        PTYP:integer;
                        PFILNAM:packed array[0..15]of char;
                        LASTBYTES:integer; DATE:integer;
                       end);
            end;(* PASENTRY*)
var UNITNUM,PLEN,DLEN,INDX,NUMPFILES,I:integer;
    CH:char;
    DOSDIRECT:packed record
         case integer of
          0:(DOSDIR:packed array[0..127]of DOSENTRY);
          1:(DOSBFR:packed array[0..2047] of char);
         end;
    PASDIRECT:packed record
         case integer of
          0:(PASDIR:packed array[0..MAXFILS]of PASENTRY);
          1:(PASBFR:packed array[0..2047] of char);
         end;
function GETNAME(var NAME:PA07OC):integer;
(* NOTE:the following header is illegal syntax.....
(* function GETNAME(var NAME:packed array[0..7]of char;):integer
(*                          <= type identifier expected
(* .....that is why there is a type PA07OC *)
 var SNAME:string[20];
     I,LEN:integer;
     MATCH:boolean;
 begin
  READLN(SNAME); LEN:=LENGTH(SNAME);
  if (LEN <= 8) and (LEN > 0)
   then begin
         FILLCHAR(NAME,8,' ');
         MOVELEFT(*to right*)((*source*) SNAME[1], (*to*)
                             (*destination*) NAME[0],
                             (*for length of*) LEN);
         MATCH:=false;
           (* INDX is the number of the file we are working *)
           (*  on.  No MATCH look-up is needed 1st pass.    *)
         if INDX > 0 then
          for I:=0 to INDX-1 do
           with DOSDIRECT.DOSDIR[I] do
             MATCH:=MATCH or (NAME=DFILNAM);
         if MATCH then
          begin
           LEN:=9;
           WRITELN('ERROR Name "',SNAME,'" already used');
          end;(*if MATCH*)
        end(*if LEN<=8 then*)
   else WRITELN('ERROR Name too long or short.');
  GETNAME:=LEN;
 end;(*GETNAME*)
```

```
begin
 WRITELN('BIOSTODOS version 2.0');
WRITE('Unit #'); READLN(UNITNUM);
UNITREAD(UNITNUM,PASDIRECT.PASBFR,2048,2,0);
FILLCHAR(DOSDIRECT.DOSBFR,2048,' ');
with PASDIRECT,DOSDIRECT do
 begin
  NUMPFILES:=PASDIR[0].HEAD.FILINVOL;
  with DOSDIR[0] do
   begin
    repeat
     begin WRITE('Type DOS directory file name:');
           INDX:=0; DLEN:=GETNAME(DFILNAM);
     end until DLEN <=8;
    DSTRTBLK:=0;
    DFILLEN:=10;
    DTYP[0]:=CHR(DOSTYPE);
   end;
   with DOSDIR[1] do
    begin
     repeat
      begin WRITE('Type Pascal directory file name:');
            INDX:=1; DLEN:=GETNAME(DFILNAM);
      end until DLEN <=8;
     DSTRTBLK:=10;
     DFILLEN:=10;
     DTYP[0]:=CHR(DOSTYPE);
    end;
   for INDX:=2 to NUMPFILES+1 do
    with PASDIR[INDX-1].FYLE,DOSDIR[INDX] do
     begin
      PLEN:=ORD(PFILNAM[0]);
      repeat
       begin
        for I:=1 to PLEN do WRITE(PFILNAM[I]);
        for I:=PLEN to 20 do WRITE(' ');
        WRITE(' New DOS name... "........"');
        for I:=0 to 8 do WRITE(CHR(8));
        DLEN:=GETNAME(DFILNAM);
       end until DLEN <=8;
      DSTRTBLK:=PSTRTBLK+10;
      DFILLEN:=PNEXT-PSTRTBLK;
      DTYP[0]:=CHR(DOSTYPE);
     end;(*for INDX with PASDIRECT,DOSDIRECT*)
  end;(*with PASDIRECT,DOSDIRECT*)
 WRITE('Update directory?'); READ(CH);
 if (CH<>'Y') and (CH<>'y') then begin
                                  WRITE('Exiting...');
                                  EXIT(BIOSTODOS);
                                 end;
 for I:=0 to 3 do
  UNITWRITE(UNITNUM,DOSDIRECT.DOSBFR[I*512],0,I,2);
end.(*BIOSTODOS*)
```

Listing 3.

```
            (* Program to list a DOS directory     *)
            (*    from U.C.S.D. Pascal.            *)
            (*  Written Aug. '80                   *)
            (*      by Chris Young                 *)
            (*         3119 Cossell Drive          *)
            (*         Indianapolis IN 46224       *)
            (*            (317)-291-5376           *)
program DOSCAT;
 const NUMLINES=23; (* Number of lines on screen - 1 *)
       DIRSIZ=127;  (* Number of DOSENTRYs           *)
       PHYSICAL=2;  (* Transfer code for UNITREAD    *)
 type DOSENTRY=packed record
                DFILNAM:packed array[0..7] of char;
                DSTRTBLK:integer;
                DFILLEN:integer;
                DTYP:packed array[0..3] of char;
               end;
 var UNITNUM,INDX,I,TYP:integer;
     CH:char;
     DOSDIRECT:packed record
          case integer of
           0:(DOSDIR:packed array[0..127]of DOSENTRY);
           1:(DOSBFR:packed array[0..2047] of char);
          end;
 (* Procedure to write a hexidecimal VAL to OUTPUT *)
 (*  Takes VAL mod 256 and writes 2 hex "digits"   *)
 procedure WRITEHEX(VAL:integer);
  var HEXDIG:packed array[0..15] of char;
  begin
   VAL:=VAL mod 256;
   HEXDIG:='0123456789ABCDEF';
   WRITE(HEXDIG[VAL div 16]);
   WRITE(HEXDIG[VAL mod 16]);
  end;(*WRITEHEX*)
 begin
  WRITELN('DOSCAT version 2.0');
  WRITE('Unit #'); READ(UNITNUM);
  for I:=0 to 3 do
   UNITREAD(UNITNUM,DOSDIRECT.DOSBFR[I*512],0,I,PHYSICAL);
  with DOSDIRECT do
   for INDX:=0 to DIRSIZ do
    with DOSDIR[INDX] do
     begin  (* Ignore blank entries *)
      if DFILNAM[0]<>' ' then
       begin
        WRITELN;
        WRITE(DFILNAM,DSTRTBLK:6);
        TYP:=ORD(DTYP[0]);
       if TYP>127  (* Test density bit *)
        then begin
              TYP:=TYP-128; WRITE(DFILLEN*2:6,' D');
             end
        else begin
              WRITE(DFILLEN:6,' S');
             end;
        WRITE(TYP:4);
        (* Test for type 1 machine code file *)
        if TYP=1 then begin WRITE(' ');
        (* High order byte *) WRITEHEX(ORD(DTYP[2]));
        (* Low order byte  *) WRITEHEX(ORD(DTYP[1])); end;
        (* Test for full screen *)
        if (INDX mod NUMLINES)=(NUMLINES-1) then
         begin
          WRITE((* Home cursor *) CHR(11),'Hit <sp> to continue');
          READ(KEYBOARD,CH);
          WRITELN((* Clear screen *) CHR(12));
         end;
       end;(*if DFILNAM[0]<>' '*)
     end;(*with DOSDIRECT; for INDX; with DOSDIR[INDX]*)
end.
```

The Godbout Spectrum Color Graphics & Sublogic 3D Software Package

by Joe Secondo

I recently purchased a moderately priced hi-resolution graphics board called the "Spectrum" by Godbout Electronics. Included with the hardware was a 3-D graphics software package by Sublogic. The result: moderately fast dynamic 3-D displays that are a pleasure to use.

Using Basic and the Sublogic software, I developed displays depicting my home, including the porch we're planning to add to the house. I then had the opportunity to review the result in 3-D from all angles, including from inside the house (see photos). I may never add the porch, but what a ball I'm having manipulating the display and adding additional detail to the scene (doors, windows, shed, etc.)!

The Spectrum Hardware

The Spectrum contains 8K hi-speed memory, of which 6K is used for the display, and two parallel I/O ports, one for controlling and monitoring the display and the other for general use. It also has a number of features such as IEEE-696 compatibility and extended addressing. The board took less than four easy hours to build. This is typical of the Godbout "Unkits" along with their well thought through and laid out designs. The inclusion of a good RF Modulator is essential for color if you don't have a color monitor and have to use a color TV set. I picked up an assembled SUP'R'MOD II, by M&R Enterprises ($30-35), plugged it into the waiting matching receptable on the Spectrum and into a color TV set, and then turned the system on. Everything worked immediately, including my normally cantankerous television.

The Spectrum permits ten modes of operation controlled via the control port. The first two modes are for use as either an 8K RAM board or as a 32X16 alphanumeric display using 512 bytes (basically not suitable for a terminal) with an eight color semigraphics (64X32) capability. The remaining eight modes are divided into two sets, one with color and one without. The four color modes can only display four of the eight colors at a time. (The colors are divided into two sets: green, yellow, blue, and red; or gray, cyan, magenta, and orange.) The resolution of the four color modes increases from 64X64 using 1K of memory to 128X192 using 6K. The resolution of the non-color goes from 128X64 using 1K of memory to 256X192 using 6K.

Joe Secondo, 18 Brookside Dr., Bath, PA 18014.

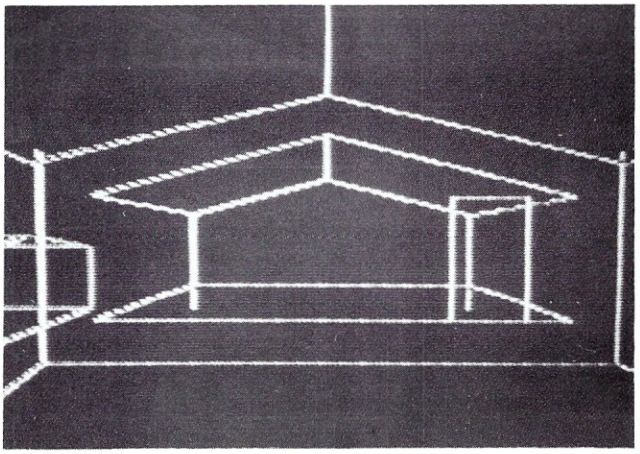
The 3-D Graphics Software

The software from Sublogic consists of two programs, 2-D and 3-D. Both programs, used as subroutines for a mainline program, are interpreters. The user defines a series of X-Y coordinates and drawing characteristics, such as point, line, rectangle, ellipse, character (right side up or sideways), etc. The 2-D program, as the basic line drawing capability, interprets this data and develops the screen display. The 3-D interpreter takes the user's series of X-Y-Z coordinates, and characteristics such as starting point, continuation line, or ray (a line that doesn't continue), and develops it as input for the 2-D interpreter.

The approach is in common use and is very clever because, as Sublogic points out in its literature, the "input files" are portable from display device to display device and from machine to machine. More important to the user is the fact that Sublogic also has interpreters for plotters. Hence a display developed on a video graphics board can be printed.

Using The System

To use the system, I developed a mainline program in Microsoft 5.0 Basic. Fundamentally, the program communicates with the interpreters using the POKE function to create the "input files." The use of Basic does not slow down the action as much as you would think because the functions in the mainline program, which are used to call on the interpreters to rotate or move the display around, only have to adjust viewing location parameters.



3-D representation of the interior of the author's home.

THE NATIONAL COMPUTER SHOWS

HAVE WE GOT A PROGRAM FOR YOU IN '81 & '82

Attend the biggest public computer shows in the country. Each show has 100,000 square feet of display space featuring over 50 Million Dollars worth of software and hardware for business, industry, government, education, home and personal use.

You'll see computers costing $150 to $250,000 including mini and micro computers, software, graphics, data and word processing equipment, telecommunications, office machines, electronic typewriters, peripheral equipment, supplies and computer services.

All the major names are there including; IBM, Wang, DEC, Xerox, Burroughs, Data General, Qantel, Nixdorf, NEC, Radio Shack, Heathkit, Apple, RCA, Vector Graphic, and Commodore Pet. Plus, computerized video games, robots, computer art, electronic gadgetry, and computer music to entertain, enthrall and educate kids, spouses and people who don't know a program from a memory disk.

Don't miss the Coming Of The New Computers – Show Up For The Show that mixes business with pleasure. Admission is $5 for adults and $2 for children under 12 when accompanied by an adult.

Ticket Information

Send $5 per person with the name of the show you will attend to National Computer Shows. 824 Boylston Street. Chestnut Hill. Mass. 02167. Tel. 617 739 2000. Tickets can also be purchased at the show.

THE MID-WEST COMPUTER SHOW

CHICAGO
McCormick Place
SCHOESSLING HALL
23rd & THE LAKE
THURS-SUN
SEPT 10-13, 1981
11 AM TO 7PM WEEKDAYS
11 AM TO 6PM WEEKENDS

THE MID-ATLANTIC COMPUTER SHOW

WASHINGTON, DC
DC Armory/Starplex
2001 E. CAPITAL ST. SE
(E CAP. ST. EXIT OFF I 295 -KENILWORTH FRWY)
ACROSS FROM RFK STADIUM
THURS-SUN
SEPT 24-27, 1981
11 AM TO 7PM WEEKDAYS
11 AM TO 6PM WEEKENDS

THE NORTHEAST COMPUTER SHOW

BOSTON
Hynes Auditorium
PRUDENTIAL CENTER
THURS-SUN
OCT 15-18, 1981
11 AM TO 7PM WEEKDAYS
11 AM TO 6PM WEEKENDS

THE SOUTHEAST COMPUTER SHOW

ATLANTA
Atlanta Civic Center
395 PIEDMONT AVE NE AT
RALPH McGILL BLVD
THURS-SUN
OCT 29-NOV 1, 1981
11 AM TO 7PM WEEKDAYS
11 AM TO 6PM WEEKENDS

THE SOUTHERN CALIFORNIA COMPUTER SHOW

LOS ANGELES
LA Convention Center
1201 SOUTH FIGUEROA
THURS-SUN
MAY 6-9, 1982
11 AM TO 7PM WEEKDAYS
11 AM TO 6PM WEEKENDS

Godbout & Sublogic cont'd...

The basic drawback experienced in using the system was insufficient memory space. The three interpreters, Basic, and the 3-D and 2-D Sublogic subroutines required 25K, 4.5K, and 6K respectively, for a total of 36K. Add a 7K CP/M and you are left with about 6K of a 48K machine for the Basic program. Since the non-display 2K on the Spectrum was used for the "input files," the only problem was excessive length variable names and REMs. Before the program was finished these were eliminated, making the program illegible, and leaving 56 bytes to spare (they were also used).

Conclusion

The Spectrum is somewhat slower than other units, i.e. Matrox and Microangelo. However, it is considerably less expensive ($339 unkit, $399 assembled and $449 qualified under Godbout's CSC program) and just as soul-satisfying. The Sublogic graphics software (normally $35, but only $25 when purchased with the Spectrum) is well written and well documented. In fact, it was because the software was available for it that I selected the hardware. The software, which performed better than advertised, used techniques to speed up the displays I had never heard of (my ignorance is showing), techniques such as "Stack Blasting" and timed sensitive loops rather than testing a device condition. It also has many user-adjustable parameters. This software package does a very professional job and is well worth the money.

Notes On The Mainline Program

The program consists of a mainline and nine subroutines. The line numbers for MAINLINE are in the one hundred series. The line numbers for the subroutines are in the thousands starting with the number used in the menu, e.g. the MENU subroutine is in the menu as "1", hence the MENU subroutine is in the 1000 series of numbers. The subroutines are:

1000 — MENU

2000 — CONSTANTS
initializes constants and addresses

3000 — INITIALIZE SPECTRUM
contains the data and routine for constructing the "input files"

4000 — PRINT CONTENTS OF ADDRESSES
there are lots of them

5000 — WHAT'S IN MEM
a generalized poke routine

6000 — a generalized peek routine
POKE MEM
a generalized poke routine

7000 — CALL GRAPHICS
that's all it is, a call to the interpreters

8000 — DRAW 2D
temporarily contains a print free memory routine.

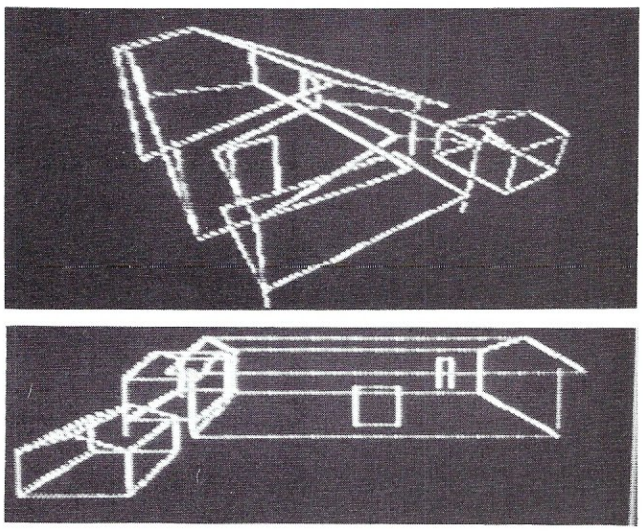
9000 — VIEWER LOCATION CONTROL
contains two routines:
1-user x,y, & z location
2-user pitch, bank, and heading

This type of application calls for a "hot" keyboard, so 1000-MAINLINE and 9000-VIEWER LOCATION CONTROL routines use "INKEY$". Location control requires more keys than a numeric keypad contains (4/6 for +X/-X; 8/2 for +Y/-Y; 5/0 for +Z/-Z; and 8/2, 7/9, and 1/3 for + & - pitch, bank, and heading), so the two location subroutines exit to each other on "D" for direction and "L" for location. "C" executes 4000-PRINT CONTENTS OF ADDRESSES subroutine. This is used while dynamically manipulating the display to "see" where you are. The routine shows the values of the X, Y and Z coordinates as well as pitch, bank and heading. "Q" quits to the mainline.

3000-INITIALIZE SPECTRUM actually contains two displays—a test display and the working display. The purpose of the test display (a 2-D line segment at a 45 degree angle) is to provide visual evidence that the 2-D interpreter is working. Manually creating and changing displays is so prone to error that when a fouled-up display occurs, you don't know if it's you or the interpreter (especially since the poke function is used so much). The test display, which only comes on for a short time after the working file has been created or changed, lets you know that at least the 2-D interpreter is working.

The working display is actually a series of DATA statements containing the drawing characteristic (call it an "OP CODE") and the X, Y and Z coordinates for each line or point. The 3-D interpreter expects the data to be in hexadecimal and in 8080/Z80 byte-swapped fashion. To avoid that problem and to stay in decimal, all coordinates are kept under 256 units and the hi-order bytes are zeroed out in the routine which creates the "input files" (lines 3160-3240). This also has the effect of reducing the "cramped for space" problem. Only one half the number of bytes used in the "input files" are stored in the DATA statements. (Later on I plan to change the program to store the working file on disk.) Using decimal values of one unit per foot in the working display, and starting the display at X, Y and Z coordinates of 100,0,100, plus the elimination of the hi-order byte in the DATA statements, reduced the manual input error substantially. ■

For more information on these products, contact: Godbout Electronics, Building 725, Oakland Airport, CA 94614; phone: (415) 562-0636.



Front and side view of author's home.

```
10 REM GRAPHICS FOR THE SPECTRUM MBASIC 5.2
       GRAPHICS.ASC 2/8/81
  JOE SECONDO - LEHIGH VALLEY COMPUTER GROUP, LEHIGH VALLEY PA.
              - AMATEUR COMPUTER GROUP OF NJ
20 ' NOTE "<<<---" INDICATES HARD CODED ADDRESSES
30 '
40 ' *** INITIALIZATION ***
50 GOSUB 2030:' INITIALIZE CONSTANTS
60 GOSUB 3010:' INITIALIZE SPECTRUM
70 GOSUB 1020:' PRINT MENU
80 '
90 ' *** MAINLINE ***
100 PRINT "CMD ";
110 CMD$=INKEY$:IF LEN(CMD$) = 0 THEN 110
120 IF CMD$="Q" THEN END
140 CMD=ASC(CMD$)-48
150 IF (CMD<2) THEN CMD=1
160 IF CMD>9 THEN CMD=1
170 ON CMD GOSUB 1020,2030,3010,4010,5010,6010,7020,8020,
       9020
180 GOTO 100
190 '
1000 ' *** MENU ***
1010 PRINT "       *** MENU ***"
1020    PRINT"1=MENU":PRINT "2=RE-INIT CONST & ADDRS':
        PRINT"3=RE-INIT SPECTRUM":PRINT"4=PRINT CONST &ADDRS'":
        PRINT"5=PEEK MEM":PRINT"6=POKE MEM":
        PRINT"7=CALL GRAPHICS":PRINT"8=DRAW 2D":
        PRINT"9=3D GRAPHICS CONTROL":
        PRINT"Q=QUIT"
1030 RETURN
1040 '
2000 ' *** CONSTANTS ***
2010 '
2020 ' *** 3D ADDRESSES ***
2030 BASE.3D=&H7B00:' <<<--- 3D GRAPHICS BASE ADDRESS
2040 ENTRY.3D=BASE.3D+&H60
2050 IBP.3D=BASE.3D+&H52:' INPUT BUFFER POINTER
2060 OBP.3D=BASE.3D+&H54
2070 XV=BASE.3D+&H3C:YV=BASE.3D+&H3E:ZV=BASE.3D+&H40
2080 PV=BASE.3D+&H42:BV=BASE.3D+&H44:HV=BASE.3D+&H46
2090 AXR=BASE.3D+&H48:AYR=BASE.3D+&H4A:AZR=BASE.3D+&H4C
2100 SCRH=BASE.3D+&H4E:SCRW=BASE.3D+&H50
2110 MTXC=BASE.3D+&H56
2120 FILLSW=BASE.3D+&H58
2130 WINDC=BASE.3D+&H5A
2140 '
2150 ' *** 2D ADDRESSES ***
2160 '
2170 BASE.2D=&H8D00:'<<<--- 2D GRAPHICS BASE ADDRESS
2180 ENTRY.2D=BASE.2D+&H2C5
2190 IBP.2D=BASE.2D+&H256
2200 IB.2D=(PEEK(IBP.2D+1)*256)+(PEEK(IBP.2D))
2210 ERR.IND.2D=BASE.2D+&H258
2220 LAST.BYTE.LOOKED.AT=BASE.2D+&H254
2230 RETURN
2240 '
3000 ' *** INITIALIZE SPECTRUM ***
3010 POKE IBP.2D,&H0:POKE IBP.2D+1,&HFC:' <<<--- RESTORES TEST FILE ADDRESS
3020 IB.2D=(PEEK(IBP.2D+1)*256)+(PEEK(IBP.2D))
3030 POKE (IB.2D),&H47: 'RESTORES TEST FILE
3040 POKE (IB.2D+1),0
3050 POKE (IB.2D+2),1
3060 POKE (IB.2D+3),6
3070 POKE (IB.2D+4),0
3080 POKE (IB.2D+5),0
3090 POKE (IB.2D+6),&H45
3100 POKE (IB.2D+7),&H45
3110 POKE (IB.2D+8),4:REM EOF
3120 CALL ENTRY.2D
3130 '
3140 ' *** PREP WORKING FILE ***
3150 POKE IBP.3D,&H0:POKE IBP.3D+1,&HF8:' <<<--- INPUT FILE LOC
3160 FOR X=&HF800 TO &HFA00 STEP 7:' <<<--- INPUT FILE ADDRESSES
3170 IF OP$="4" THEN 3240
3180 READ OP$:IF OP$="4" THEN POKE X,4:GOTO 3240
3190 READ XX$,YY$,ZZ$
3200 OP=VAL(OP$):XX=VAL(XX$):YY=VAL(YY$):ZZ=VAL(ZZ$)
3210 POKE X,OP:POKE X+1,XX:POKE X+2,0:POKE X+3,YY:
            POKE X+4,0:POKE X+5,ZZ:POKE X+6,0
3220 PRINT HEX$(X);"=";OP;XX;YY;ZZ
3230 PRINT PEEK (X);PEEK(X+1);PEEK(X+2);PEEK(X+3);PEEK(X+4);
       PEEK (X+5);PEEK(X+6)
3240 NEXT X
3250 RESTORE
3260 '
3270 ' *** WORKING DATA: HOUSE AND SHED (FORMAT=OP CODE, X, Y, & Z) ***
3280 DATA 37,100,0,100,38,152,0,100,39,152,11,100,38,152,0,124,
       39,152,11,124,38,100,0,124,39,100,11,124,
       38,100,0,100,39,100,11,100:REM WALLS
3290 DATA 37,98,11,98,38,154,11,98,38,154,15,112,39,98,15,112,
       38,154,11,126,38,98,11,126,38,98,15,112,38,98,11,98:
       REM ROOF
3300 DATA 37,152,2,102,38,162,2,102,39,162,0,102,38,162,2,122,
       39,162,0,122,38,152,2,122,38,152,2,102:REM PORCH
3310 DATA 37,100,2,100,38,152,2,100,38,152,2,124,38,100,2,124,
       38,100,2,100:REM FLOOR
3320 DATA 37,152,2,104,38,152,9,104,38,152,9,107,38,152,2,107:
       REM PORCH DOOR
3330 DATA 37,155,0,129,38,165,0,129,39,165,5,129,38,165,0,139,
       39,165,5,139,38,155,0,139,39,155,5,139,38,155,0,129,
       38,155,5,129,38,160,7,129,39,160,7,139,38,165,5,129,
       38,165,5,139,38,160,7,139,38,155,5,139,38,155,5,129:
       REM SHED
3340 DATA 37,106,2,100,38,106,9,100,38,109,9,100,38,109,2,100:
       REM BACK DOOR
3350 DATA 37,123,2,124,38,129,2,124,38,129,8,124,38,123,8,124,
       38,123,2,124:REM BOW WINDOW
3360 DATA 4:REM END OF "INPUT FILE"
3370 POKE OBP.3D,&H3:POKE OBP.3D+1,&HFC:' <<<--- OUTPUT BUFFER ADDRESS
3380 POKE MTXC,&H0
3390 POKE WINDC,0
3400 CALL ENTRY.3D
3410 CALL ENTRY.2D
3420 RETURN
3430 '
4000 ' *** PRINT CONTENTS OF ADDRESSES ***
4010 PRINT "BASE.3D=";HEX$(BASE.3D);" ";
4020 PRINT"ENTRY.3D=";HEX$(ENTRY.3D);" ";
4030 PRINT "IBP.3D=";HEX$(IBP.3D);" ";HEX$(PEEK(IBP.3D));" ";
       HEX$(PEEK(IBP.3D+1));" ";
4040 PRINT "OBP.3D=";HEX$(OBP.3D);" ";HEX$(PEEK(OBP.3D));" ";
       HEX$(PEEK(OBP.3D+1))
4050 PRINT "XV= ";HEX$(XV);" ";HEX$(PEEK(XV));" ";
       HEX$(PEEK(XV+1));" ";
4060 PRINT "YV= ";HEX$(YV);" ";HEX$(PEEK(YV));" ";
       HEX$(PEEK(YV+1));" ";
4070 PRINT "ZV= ";HEX$(ZV);" ";HEX$(PEEK(ZV));" ";HEX$(PEEK(ZV+1))
```

```
4080 PRINT "PV= ";HEX$(PV);" ";HEX$(PEEK(PV));" ";
     HEX$(PEEK(PV+1));" ";
4090 PRINT "BV= ";HEX$(BV);" ";HEX$(PEEK(BV));" ";
     HEX$(PEEK(BV+1));" ";
4100 PRINT "HV= ";HEX$(HV);" ";HEX$(PEEK(HV));" ";
     HEX$(PEEK(HV+1))
4110 PRINT "AXR= ";HEX$(AXR);" ";HEX$(PEEK(AXR));" ";
     HEX$(PEEK(AXR+1));" ";
4120 PRINT "AYR= ";HEX$(AYR);" ";HEX$(PEEK(AYR));" ";
     HEX$(PEEK(AYR+1));" ";
4130 PRINT "AZR=";HEX$(AZR);" ";HEX$(PEEK(AZR));" ";
     HEX$(PEEK(AZR+1))
4140 PRINT "SCRH=";HEX$(SCRH);" ";HEX$(PEEK(SCRH));" ";
     HEX$(PEEK(SCRH+1));" ";
4150 PRINT "SCRW=";HEX$(SCRW);" ";HEX$(PEEK(SCRW));" ";
     HEX$(PEEK(SCRW+1))
4160 PRINT "MTXC=";HEX$(MTXC);" ";HEX$(PEEK(MTXC));" ";
4170 PRINT "FILLSW=";HEX$(FILLSW);" ";HEX$(PEEK(FILLSW));" ";
4180 PRINT "WINDC= ";HEX$(WINDC);" ";HEX$(PEEK(WINDC))
4190 PRINT
4200 PRINT "BASE.2D=";HEX$(BASE.2D);" ";
4210 PRINT "ENTRY.2D=";" ";HEX$(ENTRY.2D);" ";
4220 PRINT "IBP.2D=";HEX$(IBP.2D);" ";
4230 PRINT "ERR.IND.2D=";HEX$(ERR.IND.2D);" ";
     HEX$(PEEK(ERR.IND.2D))
4240 PRINT "LAST.BYTE.LOOKED.AT=";HEX$(LAST.BYTE.LOOKED.AT)
     ;" ";HEX$(PEEK(LAST.BYTE.LOOKED.AT+1))
     ;" ";HEX$(PEEK(LAST.BYTE.LOOKED.AT))
4250 PRINT "IB.2D=";HEX$(IB.2D)
4260 START.ADDR=IB.2D
4270 BYTES=10
4280 GOSUB 5060
4290 RETURN
4300 '
5000 ' *** WHAT'S IN MEM ***
5010 PRINT:PRINT "PEEK MEM"
5020 INPUT "START ADDRESS (HEX)",START.ADDR$
5030 IF LEN(START.ADDR$)=0 THEN RETURN
5040 START.ADDR=VAL("&H"+START.ADDR$)
5050 INPUT "NO. OF BYTES",BYTES
5060 FOR X=START.ADDR TO START.ADDR+BYTES
5070 PRINT HEX$(X);"=";HEX$(PEEK(X));" ";
5080 NEXT X
5090 RETURN
5100 '
6000 ' *** POKE MEM ***
6010 PRINT:PRINT "POKE MEM"
6020 INPUT "START ADDR (HEX)",START.ADDR$
6030 IF LEN(START.ADDR$)=0 THEN RETURN
6040 PRINT "INPUT BYTE (Q=QUIT)"
6050 START.ADDR=VAL("&H"+START.ADDR$)
6060 PRINT HEX$(START.ADDR);"=";HEX$(PEEK(START.ADDR));" ";
6070 INPUT BYTE$
6080 IF BYTE$="Q" THEN RETURN
6090 IF LEN(BYTE$)=0 THEN GOTO 6120
6100 BYTE=VAL("&H"+BYTE$)
6110 POKE START.ADDR,BYTE
6120 START.ADDR=START.ADDR+1
6130 GOTO 6060
6140 '
7000 ' *** CALL GRAPHICS ***
7010 CALL ENTRY.3D
7020 CALL ENTRY.2D
7030 RETURN
7040 '
8000 ' *** DRAW 2D ***
```



```
8010 ' TEMP USAGE
8020 PRINT FRE(0)
8030 RETURN
8040 '
9000 ' *** VIEWER LOCATION CONTROL ***
9010 ' *** X,Y, & Z CONTROL; D=GOTO VIEWER DIRECTION CONTROL ***
9020 X$=INKEY$:IF LEN(X$)=0 THEN 9020
9030 IF X$="D" THEN GOTO 9160
9040 IF X$="C" THEN GOSUB 4010
9050 IF X$="Q" THEN RETURN
9060 IF X$="5" THEN ZVL=PEEK(ZV):ZVL=ZVL+1:POKE ZV,ZVL
9070 IF X$="0" THEN ZVL=PEEK(ZV):ZVL=ZVL-1:POKE ZV,ZVL
9080 IF X$="4" THEN XVL=PEEK(XV):XVL=XVL+1:POKE XV,XVL
9090 IF X$="6" THEN XVL=PEEK(XV):XVL=XVL-1:POKE XV,XVL
9100 IF X$="8" THEN YVL=PEEK(YV):YVL=YVL+1:POKE YV,YVL
9110 IF X$="2" THEN YVL=PEEK(YV):YVL=YVL-1:POKE YV,YVL
9120 GOSUB 7010:GOTO 9020
9130 '
9140 ' *** VIEWER DIRECTION CONTROL ***
9150 ' *** PITCH, BANK AND HEADING; L= GOTO VIEWER LOCATION CTL ***
9160 D$=INKEY$:IF LEN(D$)=0 THEN 9160
9170 IF D$="Q" THEN RETURN
9180 IF D$="C" THEN GOSUB 4010
9185 IF D$="L" THEN GOTO 9020
9190 IF D$="8" THEN PVD=PEEK(PV+1):PVD=PVD+1:POKE PV+1,PVD
9200 IF D$="2" THEN PVD=PEEK(PV+1):PVD=PVD-1:POKE PV+1,PVD
9210 IF D$="7" THEN BVD=PEEK(BV+1):BVD=BVD+1:POKE BV+1,BVD
9220 IF D$="9" THEN BVD=PEEK(BV+1):BVD=BVD-1:POKE BV+1,BVD
9230 IF D$="1" THEN HVD=PEEK(HV+1):HVD=HVD+1:POKE HV+1,HVD
9240 IF D$="3" THEN HVD=PEEK(HV+1):HVD=HVD-1:POKE HV+1,HVD
9250 GOSUB 7010:GOTO 9160
```

Managing Your Magazine File

by Lou S. Davis

Although you may have subscribed to only two or three computer magazines in the last few years, your stacks of reference material must be, as mine are, exceeding manageable proportions. A yearly index by subject, author and title offered by many magazines, while helpful, is quite limited in scope and still requires much pulling, tugging, and page flipping to find needed articles. Even *Microsystems*, with its relatively few issues, is beginning to present these same difficulties.

Your text editor can help ease these information retrieval problems. By following a few simple formatting rules, the text editor will build and maintain reference files on your home computer that can be searched by the program described in this article. This program, written in Digital Research's PL/I-80, will process multiple files in looking for combinations of key words and phrases in references created with your text editor.

For example, one might code a reference to a *Microsystems* article as:

S809010 Epstein, J: Intro to CP/M, Part 4 tutor, Utilities, BIOS, Interrupts, IOBYTE, ASM pgm examples

where the S809010 stands for *S-100 Microsystems* (the "S-100" prefix in use until January 1981), 1980, September, page 10. Coding is relatively easy. Key words can be chosen by quickly scanning the article. Other words, such as tutor (for a tutorial article) or pgm (for an article containing a program listing), can be added to help to identify articles in future searches.

The publication reference file format required by the retrieval program is simple. For my files, each article (and each interesting letter to the editor) is coded as follows:

```
Pyymppp#title line
##key word,key word,...          as many detail lines
##key word,key word,...          as necessary or none

as necessary or none
```

where:

P	is a Publication Identifier (e.g. M = *Microsystems*)
yy	year, e.g., 80
m	month, e.g., 2 = Feb, A = Oct, B = Nov, C = Dec
ppp	page number, e.g., 010
#	are mandatory spaces (blanks)
title	is any combination of author name, article title, key words, etc.

and the key words are those words appearing in the articles which are important for future retrieval considerations. Figures 1 and 2 show several examples of coded articles.

However, the only critical requirements are that the first character of the first line of a group of lines for one article should be alphabetic and upper case (e.g., the M for *Microsystems* or S for *S-100 Microsystems*), and that the first character of each of the rest of the lines representing one article be blank.

Files can contain one or more years worth of references for any given publication, depending mostly on how well your text editor handles files larger than memory size.

Requests are made by specifying words, either from the title line or detail lines, separated by logic operators and/or parentheses. The allowed operator set is as follows:

- ∧(left parenthesis
- ∧) right parenthesis
- ∧| logical or
- ∧& logical and
- ∧~ logical not

Figure 1 shows the complete interaction between the computer and a requester. This request was motivated by the need to find a Basic program for doing a quicksort. It asks for articles that have been coded with the words:

QUICK and SORT and PROGRAM and BASIC.

Three separate files, as named in Figure 1, were searched for this request:

S01.REF S-100 Microsystems, 80 Jan - 80 Oct
K01.REF kilobaud Microcomputing, 77 Mar - 79 Dec
K02.REF kilbaud Microcomputing, 80 Jan - 80 Oct

A detailed explanation of the output in Figure 1 follows. Line 1, in Figure 1, is the CP/M call for the retrieval program KWIRP (Key Work Information Retrieval Program). Line 2 is the program output to the console asking for the retrieval string. Line 3 is the console echo of the requester's typed-in string. Line 4, output to the list device, is a printed repeat of the request string as input. Line 5, to the console, is an explicit chance to double check the exact request, e.g., no blanks before or after any given word. At this point the program does a rigorous check of the string and will print out any appropriate error messages (shown in a later example). Line 6, displayed on the console, asks for the name of the first file to be scanned. In this case, the requester response is b:s01.ref. Line 7, on the list device, shows the name of the file searched, followed by any hits on that file and

Lou S. Davis, 38 Krohn Lane, Oakland, CA 94611.

```
{ 1}  A>b:kwirp
{ 2}  Type Retrieval String
{ 3}  -> quick^&sort^&basic^&pgm
{ 4}  quick^&sort^&basic^&pgm
      1    -2    "QUICK"
      2     4    "^&"
      3    -2    "SORT"
{ 5}  4     4    "^&"
      5    -2    "BASIC"
      6     4    "^&"
      7    -2    "PGM"
{ 6}  Specify  1st  File Name, e.g., [B:]Test[.ref]
      Type File Name ----------> b:s01.ref
{ 7}  ******** B:S01.REF      ********
      S805035 is your cmpr out of sorts?
        test and compare algorithms,bubble,heap,shell metzner,tree,quick,
        basic pgms
      ********************************* Number of Hits on File B:S01.REF   =   1
{ 8}  Specify   F to read next FILE,   S to enter new STRING, or  D if DONE
      Enter F, S, or D ----------> f
{ 9}  Specify  Next  File Name, e.g., [B:]Test[.ref]
      Type File Name ----------> b:k01.ref
{10}  ******** B:K01.REF      ********
      K794096 quicksort
        shellsort,basic pgms
      ********************************* Number of Hits on File B:K01.REF    =   1
{11}  Specify   F to read next FILE,   S to enter new STRING, or  D if DONE
      Enter F, S, or D ----------> f
{12}  Specify  Next  File Name, e.g., [B:]Test[.ref]
      Type File Name ----------> b:k02.ref
{13}  ******** B:K02.REF      ********
      ********************************* Number of Hits on File B:K02.REF    =   0
{14}  Specify   F to read next FILE,   S to enter new STRING, or  D if DONE
      Enter F, S, or D ----------> d
{15}  End of Execution
```

Figure 1: *Detail of Retrieval Program Output (see text for explanation) Interspersed Console and List Device Output.*

```
^(Sort^|merge^)^&^~^(History^|graphic^|business^|TRS80^|PET^|Apple^)

******** B:S01.REF      ********
S805035 is your cmpr out of sorts?
  test and compare algorithms,bubble,heap,shell metzner,tree,quick,
  basic pgms
S805043 no more waiting for sorts
  binary tree,basic pgm
********************************* Number of Hits on File B:S01.REF       =    2

******** B:K01.REF      ********
K774034 sorting routines
  tutor,common sort techniques,basic pgm
K785100 5 minutes or 5 hours, sorting techniques compared
  basic pgm,benchmark timing
K794094 let's have some order
  alphabetization basic pgm,swtp,sort
K794096 quicksort
  shellsort,basic pgms
********************************* Number of Hits on File B:K01.REF       =    4

******** B:K02.REF      ********
K803084 an operator oriented database management system
  sort, print the data,basic pgm
K807120 sortit, a sort pgm
  basic pgm
K80A040 Looney,C: Address List Pgm
  6800 asm pgm,bubble sort
********************************* Number of Hits on File B:K02.REF       =    3

******** B:CS01.REF     ********
C71C147 martin,w: sorting
  sorting,bubble,shell,quick,radix exchange,sample,item at a time,merge,
  comparison, bibliography
  5.31
C74C261 knuth,d: structured pgmg with go to statements
  language design,event indicators,recursion,boolean variables,iteration,
  optimization of pgms,pgm transformations,
  pgm manipulation systems searching,quicksort,efficiency
  4.0, 4.10, 4.20, 5.20, 5.5, 6.1, 5.23, 5.24, 5.25, 5.27
********************************* Number of Hits on File B:CS01.REF      =    2
```

Figure 2: *List Device Output of More Complicated Request.*

```
^)Key1^*^~Key2^(^(Ke y3^|^&Key4 ^+^&^(

A  ")"  BEFORE  a  "("

Illegal Logic Operator:  *

Illegal Logic Operator:  +

Parentheses Do Not Match
    1    2  "^)"
    2   -2  "KEY1"
    3    6  "^*"    **** ERROR
    4    5  "^~"
    5   -2  "KEY2"
    6    1  "^("
    7    1  "^("
    8   -4  "KE Y3"
    9    3  "^|"
   10    4  "^&"
   11   -4  "KEY4 "
   12    6  "^+"    **** ERROR
   13    4  "^&"
   14    1  "^("

Illegal Sequence: start^)
Illegal Sequence: ^)KEY1
Illegal Sequence: KEY1^*
Illegal Sequence: ^*^~
Illegal Sequence: KEY2^(
Illegal Sequence: ^|^&
Illegal Sequence: KEY4 ^+
Illegal Sequence: ^+^&
Illegal Sequence: ^(end

Run Aborted -- Respecify Request String

Type Retrieval String

->
```

Figure 3: *Output Diagnostics for a (Very) Bad Request Interspersed Console and List Device Output.*

Before you spend $200 — or $2000 — on a computer system that won't meet your needs, read this.

Straight Talk About Buying A Small Computer

A personal computer. If you don't already have one you're probably thinking about it.

The Plunge

Spending a couple of thousand or even a few hundred dollars is not something you do lightly. Making a decision about which computer to buy is not easy. Deciding about peripherals is even harder. And making intelligent decisions about software is nearly impossible.

But there's a way. *Creative Computing*. It's the magazine with the toughest evaluations in the industry.

In an industry prone to dazzling sales pitches filled with technical doubletalk, some straight advice from an unbiased source could make a lot of difference. Some of the newest computers on the market will be extinct before the first orders are filled; others will achieve astounding popularity. Which would you rather own?

Why Buy One?

The uses of a personal computer would fill several books. Some people have a specific use in mind, others just a general desire to join the computer age. In general, it's advisable to have a good idea of at least one or two things you want a computer to do for you. If you want to analyze stock options, for example, you'll want a computer for which a stock option package is available. If you want to work with color graphics your choice of computer is narrowed to those units with high resolution color output. Even after you've chosen some applications, the choice of a machine isn't easy. Here are a few things that might help.

Hard and Soft

Besides encountering hard and soft sells, you'll also encounter the terms "hardware" and "software." Hardware refers to the electronic parts of a computer system, the circuit boards, chips, peripherals, and other components.

An important part of hardware is memory. The amount of memory a computer has determines how much it can do. Most people start out with systems having 16K (K is short for kilobyte, which means 1024 bytes, or characters like a letter or number) of memory. Later, many people decide to expand their systems, and buy more memory.

Some of the new computers can be purchased with as little as 1K of memory. They can usually be expanded, but the upper limit varies. If you want to play games and write short programs, 16K is adequate. If you want to add a disk drive, or do word processing, you'll probably need 48K or more.

By themselves, these parts are rather dumb. The programs that instruct the computer how to do the more interesting applications (stock option analysis, animation, play a game) are usually contained on magnetic tape or disks. This is software.

Both hardware and software are important. A system with the wrong hardware can be as worthless to you as a sports car would be for a six-member family. Some computers cannot be connected to a printer. Others can't be expanded without a great deal of additional expense. You may not need these extras now, but if you anticipate needing them later, you'll want to select an appropriate system now. And a computer without good software is just an expensive dust catcher.

The Software Cycle

When a computer first hits the market, the only software available will be from the manufacturer. This limits the uses of the machine. As soon as a computer becomes popular, new software pours from dozens of sources. But there is a catch. People won't buy a computer until there is plenty of software available. And vendors won't produce the software until people start buying the computer. Where does this leave you? You can go with one of the established computers, or take a chance on a new machine.

The Newcomers

New computers are appearing almost monthly. One might be right for you. Can anyone tell for sure which will survive? Probably not. But the new machines can be compared against the old. If a computer does everything and more than another does, and costs less, it has a good chance of catching on. If it does less than existing computers, and costs about the same, it is probably doomed.

The Survivors

A few computers currently have the majority of the market. They all have good points and disadvantages. One costs less to start with, but costs more to expand. Another has great graphics but no lowercase letters—a bit of a problem if you want to do word processing. The limitations of any machine can be overcome, for a price. But it is better to get what you want at the start. If you know what you plan to use the computer for, the first step is to determine what that use requires. Do you want to play games? Then you have to decide how important joysticks, paddles, and other controls are. Some computers are supplied with these attachments. Some companies sell these attachments as extras. Do you want to use your own television? Or would you prefer a computer that comes with its own monitor? Will you demand complicated math capabilities from your computer? Certain computers can only handle integers.

Your Choice

Where does this leave you? If you've read this far, you are probably concerned about making the right choice. The following hints could be a good starting place. Take your time, and don't let your first impression of any computer prevent you from taking an honest look at its good and bad points. All computers seem impressive at first. Once you've looked at a few, you'll find that the initial awe is replaced by cold comparison. If you have a specific application in mind, ask to see the computer perform that application. If the salesman starts talking in technical terms while assuring you the machine will do what you want with only a few modifications, find another store. Ask about warranties. Will it be repaired at the store or sent out? Will they provide a loaner during repairs? Is the dealer authorized by the manufacturer?

We Don't Sell Computers

We have good reason to hope you buy a computer. Every new owner represents a potential reader for *Creative Computing*, the number one magazine of applications and software. We give the beginner a wealth of useful articles, tutorials, games, and ideas for his computer. And when the beginner becomes an expert, we still have a lot to offer; in-depth articles on programming, reviews of the latest products, highlights of important events in computer community and much more.

One beginner who became an expert, David Gerrold of *Star Trek* fame, had this to say, "*Creative Computing* with its unpretentious, down-to-earth lucidity encourages the computer user to have fun. *Creative Computing* makes it possible for me to learn basic programming skills and use the computer better than any other source."

Why not join over 90,000 readers and subscribe? One year (12 issues) costs only $20 and saves you $10 compared to the newsstand price. To subscribe, call toll-free from 9 AM to 6 PM **800-631-8112**. In New Jersey, call 201-540-0445. Or write to Creative Computing, Morris Plains, NJ 07950. We accept Visa, MasterCard, and American Express.

First Get the Facts

The first step to making intelligent buying decisions about computers, peripherals and software is arming yourself with the unbiased facts. You'll find these facts presented in a down-to-earth style in *Creative Computing*. Take the first step and subscribe today.

creative computing

the number of hits. Here, the single hit is interpreted as *S-100 Microsystems*, 1980, May, page 35. Line 8 is the program console output asking what the requester wants to do next. The 'f' reply asks to continue with another file for the same search string. (At this point, a new diskette may be inserted in the appropriate drive.) Line 9, on the console, asks for the next file name—the requester response is for file b:k01.ref. Line 10 is printed on the printer, and so on. Line 15 is PL/I-80's response when the requester indicates he is done.

A more complicated request is shown in Figure 2. This request was motivated by the need to answer a general sort/merge problem without having to look at articles with particular applications. The following Boolean request string was generated:

(SORT or MERGE) and not
(HISTORY or GRAPH or BUSINESS or
TRS-80 PET or APPLE)

The files searched also included a reference file for *Computing Surveys* (CS01.ref), one of the journals of the Association for Computing Machinery (ACM). *Computing Surveys* is easy to code since each article is headed by a list of key words and phrases. Also, following the key words in *Computing Surveys* are *Computing Reviews'* (another ACM journal) numerical classification categories under which the articles are reviewed. These numbers can also be used as key words.

The search found a total of eleven hits. A request for just SORT or MERGE got seventeen hits. Adding delimiters can help reduce the number of articles you have to examine individually.

The last example, Figure 3, shows the output of the diagnostics routine for a bad request specification. The words "start" and "end" are place markers to help the interpretation of the illegal sequence messages. Note that "Key3" and "Key4" have blanks which may or may not be desired. It is "Key 3" and "Key4 " that would have been searched if the requested string had been valid otherwise.

As is obvious in the examples, certain common words were abbreviated (e.g., pgm = program, cmpr = computer). This was originally started to conserve space in the files. In retrospect, it was probably a bad idea; one of these days I will edit most of these back to the full spelling. To avoid the problems of arbitrary hyphens in some words and acronyms (and thereby missing hits because of inclusion or exclusion of hyphens), all hyphens were eliminated in coding (e.g., S100, not S-100). Also, all references were originally entered on the files in lower case to simplify search key word specifications. This is no longer necessary, as discussed below.

Finally, some miscellaneous information about the program. The program was originally coded and the logic debugged in Microsoft Basic 4.51. Since the compute time (on my Z80A S-100 bus system under CP/M) for a comprehensive sample search was 16 minutes 19 seconds, the program was rewritten in S.S.S. FORTRAN where the compute time was only 2 minutes 15 seconds. (S.S.S. was the company that wrote the original TDL FORTRAN.)

With the availability of Digital Research's PL/I-80, the program was again rewritten, taking advantage of the TRANSLATE built-in function and the ability to specify the buffer size for reading records from the files. The TRANSLATE function allows the use of both upper and lower case in the references, making them more readable. During the search, all words are translated and all comparisons between search words and reference words are made in upper case. Although this added burden of translating all reference words internally during search should connote significantly more processing, the PL/I-80 search time for the same sample case was just 2 minutes 16 seconds.

The program could be further expedited, possibly, by early recognition of a satisfied search string. For example, if for a given article reference, "Key1" was found by the request "Key1 OR Key2", there would be no need to look for "Key2" in that particular reference. A hook to insert a subroutine to do this is signalled by comments in the Execute subroutine in the program listing.

The strategy to use the '^' sybmol preceding the operators was primarily to make the string scan a little simpler in conjunction with reserving only one character out of the entire ASCII set for coding reference words. This character, the '^', is the only one that cannot be used in the reference files. If it is desirable to use some other symbol instead, the '^^' symbol in the program line following statement 'S1:' in the Separate subroutine can be replaced. Also, different operator symbols can be specified by substitution in the first argument of the INDEX function of the Assign subroutine.

The retrieval program requires 22K bytes of memory in addition to CP/M requirements. ■

Note: *CP/M annd PL/I-80 are registered trademarks of Digital Research.*

Listing 1.

```
/*      [FILE:  KWIRP.pli]
        [DATE:  1980 OCT 10]
        [VERSION:  1.0  1980 Oct 18]
        metaDOON Computer
        Microcomputer Program Series    */

/* PROGRAM KWIRP.pli     KeyWord Index Retrieval Program
   PURPOSE:                                  */
/*  A program to retrieve indices of articles from simple formatted files   */
/*  which are created and maintained by any good text editor program.       */
/*  Search is specified by a boolean string based on keywords or phrases.   */

KWIRP:  PROC OPTIONS(MAIN);

  %REPLACE
     MaxLinesNo BY 25,      /* Max lines allowed in one article index        */
     MaxLineLength BY 120,  /* Max length of each index line                 */
     MaxOps BY 30,          /* Max number of operands and operators allowed  */
                            /*     in boolean retrieval string               */
     MaxOpsV BY 31,         /* Must be one more than MaxOps                  */
```

```
      MaxKeyWord BY 30,     /* Max length of Keyword (Operand) allowed      */
      NumberOfDrives BY 2,  /* Change to agree with number of disk drives   */
                            /*      on system, B:=2, C:=3, etc.             */
      True BY '1'B,  False BY '0'B;

   DCL
      LstFCB FILE,    /*  PRINT                                             */
      OldFCB FILE;    /* STREAM INPUT                                       */

   DCL
      FlTitle CHAR(15) VARYING,
      Drive BIN(7),
      FlName CHAR(8),
      FlType CHAR(3);

   DCL
      BString  CHAR(MaxLineLength) VARYING,
      (Line(MaxLinesNo),TransLine(MaxLinesNo)) CHAR(MaxLineLength) VARYING,
      SepOp(0:MaxOps) CHAR(MaxKeyWord) VARYING,
      Temp CHAR(MaxKeyWord) VARYING,
      Adj CHAR(10) VARYING,
      Upper CHAR(26) STATIC INIT('ABCDEFGHIJKLMNOPQRSTUVWXYZ'),
      Lower CHAR(26) STATIC INIT('abcdefghijklmnopqrstuvwxyz'),
      (Operator,PubIdent,Next) CHAR(1),
      Buzz CHAR(1) STATIC INIT('^G'),
      OpVector(0:MaxOps) BIN,
      (RPNList(MaxOpsV),Stack(MaxOpsV)) BIN,
      (NextOp,StackOp,nList,nStack,nLine,HitCount) BIN,
      (SLength,NumOps,ParenNo,SColPtr,FlagLoc,KWLength,OperatorNo,i,j,k) BIN,
      TFStack(MaxOpsV) BIT(1),
      (AbortSw,LoopSw) BIT(1),
      Table(-2:6,-2:6) BIT(1) STATIC INIT
          /*    opernd begin  end    (     )     +     *     ~    illegal     */
       /*opd*/  (False,False,True,  False,True,  True,  True,  False,False,
       /*beg*/   True, False,False,True,  False,False,False,True, False,
       /*end*/   False,False,False,False,False,False,False,False,False,
       /* ( */   True, False,False,True,  False,False,False,True, False,
       /* ) */   False,False,True,  False,True,  True,  True,  False,False,
       /* + */   True, False,False,True,  False,False,False,True, False,
       /* * */   True, False,False,True,  False,False,False,True, False,
       /* ~ */   True, False,False,True,  False,False,False,False,False,
       /*ill*/   False,False,False,False,False,False,False,False,False);

/*  Main Program Starts Here                                               */
/*  NOTE:  Numbered ERRORS, #1,#2,#3 are unrecoverable errors and require  */
/*         either program modification or possible hardware repair         */

   AbortSw = True;
   OPEN FILE(LstFCB) PRINT ENV(B(128)) TITLE('$LST');

   DO WHILE (AbortSw = True);
      AbortSw = False;
      PUT SKIP(2) LIST('Type Retrieval String');
      PUT SKIP(2) LIST('-> ');
      GET EDIT(BString)(A);      /*  Read In Boolean Search String (BString) */
      SLength = LENGTH(BString);
      PUT FILE(LstFCB) SKIP(4) LIST(BString);  /* Output it to list device  */
      CALL Separate;  /* Separate the operands(Key Words) and operators     */
      CALL Legal;     /* Check operand/operator sequences in string         */
      IF AbortSw = True THEN
         PUT SKIP(2) LIST('Run Aborted -- Respecify Request String');
      ELSE
         DO;
            CALL RPNSequence;  /* Convert string to RPN sequence            */
            CALL GetFile;      /* Scan files for hits                       */
         END;
   END;
   STOP;

/*  END of Main Program -  SubRoutines follow -                             */

Assign:  PROC; /* Assign number to each operator, track parentheses        */
  OperatorNo = INDEX('()|&~',Operator);  /* 1,2,3,4, or 5 respectively      */
  IF OperatorNo > 2 THEN  RETURN;        /*  Was a  |, &, or ~              */
  IF OperatorNo = 2 THEN                 /*  Was a  )                       */
     DO;
        ParenNo = ParenNo - 1;
        IF ParenNo < 0 THEN
           DO;
              PUT FILE(LstFCB) LIST(Buzz);
              PUT SKIP(2) LIST('A  ")"  BEFORE  a  "("');
              ParenNo = 0;
              AbortSw = True;
           END;
        RETURN;
     END;
 IF OperatorNo = 1 THEN  ParenNo = ParenNo + 1;  /* Was a  (               */
 ELSE
     DO;
        PUT FILE(LstFCB) LIST(Buzz);
        PUT SKIP(2) LIST ('Illegal Logic Operator: ',Operator);
        OperatorNo = 6;  /* Set to 6 for subsequent legal check in Table   */
        AbortSw = True;
     END;
 RETURN;
 END Assign;

GetFile:  PROC;  /*  Scan Files for Hits                                   */
     Adj = '1st';
     ON ENDFILE(OldFCB)
        GO TO PrelimEnd;
     ON UNDEFINEDFILE(OldFCB)
        BEGIN;
           PUT FILE(LstFCB) LIST(Buzz);
           PUT SKIP(2) LIST('File: ',FlTitle,' Not Found.   Try Again');
           GO TO GetTitle;
        END;

GetTitle:  /*  Get File Name, Open File, Find first line with valid        */
           /*  Publication Identifier                                      */
     CALL GFlRef();
     OPEN FILE(OldFCB) TITLE(FlTitle) STREAM INPUT LINESIZE(MaxLineLength)
                              ENV(B(1024));
     PUT FILE(LstFCB) SKIP(2) EDIT('******** ',FlTitle,' ********')(A,A(14),A);
     HitCount = 0;
     PubIdent = ' ';
     DO WHILE(PubIdent < 'A' | PubIdent > 'Z');
        GET FILE(OldFCB) EDIT(Line(1))(A);
        PubIdent = SUBSTR(Line(1),1);  /* Get Publication Index or a blank */
     END;

     /*  Read File, get one complete indexed article, and go check for hits */
     DO nLine = 2 REPEAT (nLine + 1);
        GET FILE(OldFCB) EDIT(Line(nLine))(A);
        PubIdent = SUBSTR(Line(nLine),1);
        IF PubIdent > '@' & PubIdent < '[' THEN
           DO;
              CALL Execute;
              IF AbortSw = True THEN RETURN;
              Line(1) = Line(nLine);
              nLine = 1;
           END;
        ELSE
           IF nLine = MaxLinesNo THEN
              DO;
                 PUT FILE(LstFCB) LIST(Buzz);
                 PUT SKIP(2) EDIT('ERROR - #3  ',SUBSTR(Line(1),1,7),
                           ' Reference has more lines than allowed.',
                       '  Recompile with bigger MaxLinesNo or change file')
                   (A,A,A,SKIP,A);
                 AbortSw = False;
```

```
                RETURN;
              END;
        END;

    PrelimEnd:    /* Finish last indexed article, output no. of hits          */
        IF PubIdent > '@' & PubIdent < '[' & SUBSTR(Line(1),3) ~= 'END'  THEN
           DO;
              CALL Execute;
              IF AbortSw = True THEN RETURN;
           END;
        CLOSE FILE(OldFCB);
        PUT FILE(LstFCB) SKIP EDIT('*******************************',
                        ' Number of Hits on File ',FlTitle,'  = ',HitCount)
                         (A,A,A(14),A,F(4));
        PUT SKIP(2) EDIT('Number of Hits on File ',FlTitle,'  = ',HitCount)
                         (A,A,A,F(4));
        DO WHILE (True);  /*  Done with this file, find out what to do next    */
           PUT SKIP(2) LIST('Specify   F to read next FILE,',
                            '          S to enter new STRING, or  D if DONE');
           PUT SKIP LIST('Enter F, S, or D ----------> ');
           GET SKIP LIST(Next);
           IF Next = 'F' | Next = 'f' THEN
              DO;
                 Adj = 'Next';
                 GO TO GetTitle;
              END;
           IF Next = 'D' | Next = 'd' THEN RETURN;
           IF Next = 'S' | Next = 's' THEN
              DO;
                 AbortSw = True;
                 RETURN;
              END;
        END;

     END GetFile;

    GFlRef:  PROC;
    /*   This program gets a file reference from the console in the form of     */
    /*   a character string  [d:]FlName[.Fltype].  If 'd:' is not                */
    /*   specified,the logged-in drive is assumed.  If 'FlType' is not          */
    /*   specified, 'REF' is  assumed.                                          */

    /*   The program disassembles the FlTitle character string into its         */
    /*   components.                                                            */
    /*   The output line is qualified by 'Adj'                                  */
      Entry:
        PUT SKIP LIST ('Specify ',Adj,' File Name, e.g., [B:]Test[.ref]');
        PUT SKIP LIST ('Type File Name ----------> ');
        GET LIST (FlTitle);
        FlTitle = TRANSLATE(FlTitle,Upper,Lower);
        i = INDEX(FlTitle,':');
        IF i = 0 THEN Drive = 0;
        ELSE
           IF i = 2 THEN
              DO;
                 Drive = INDEX(Upper,SUBSTR(FlTitle,1,1));
                 IF Drive = 0 | Drive > NumberOfDrives THEN
                    DO;
                       PUT SKIP LIST('Bad Drive Specification - Try Again');
                       GO TO Entry;
                    END;
              END;
           ELSE
              DO;
                 PUT SKIP LIST('Illegal : position, Try Again');
                 GO TO Entry;
              END;
        j = INDEX(FlTitle,'.');
        IF j = 0 THEN
           DO;
              FlTitle = FlTitle || '.REF';
              j = INDEX(Fltitle,'.');
           END;
        FlName = SUBSTR(FlTitle,i+1,j-i-1);
        FlType = SUBSTR(FlTitle,j+1,3);
     END GFlRef;
    Execute:  PROC;  /*  Look for operands (key words) in the references      */
        nStack = 0;
        DO i = 1 TO nLine - 1;   /* Change letters to upper case               */
           TransLine(i) = TRANSLATE(Line(i),Upper,Lower);
        END;
        DO i = 1 TO NumOps + 1;
           NextOp = RPNList(i);  /* Get [next] operand/operator from RPN List  */
           IF NextOp > 5 THEN      /* It's an operand (key word)               */
              DO;
                 NextOp = NextOp - 5;
                 nStack = nStack + 1;
                 LoopSw = False;
                 Temp = SepOp(NextOp);
                 DO j = 1 TO nLine - 1 WHILE (LoopSw = False);
                    IF INDEX(TransLine(j),Temp) > 0 THEN
                          LoopSw = True;   /* This operand found in article   */
                 END;
                 TFStack(nStack) = LoopSw;
                   /* Next two lines of code allow a hook to a routine to      */
                   /* check for early satisfaction of string -- to be          */
                   /* implemented later but only if majority of searches have  */
                   /* lots of hits                                             */
    /*             IF LoopSw = True THEN                                       */
    /*                CALL Speed;                                              */
              END;
           ELSE     /*  Have an operator                                       */
              DO;
                 IF NextOp = 3 THEN     /*  '|'   OR                           */
                    DO;
                       TFStack(nStack-1) = TFStack(nStack-1) | TFStack(nStack);
                       nStack = nStack - 1;
                    END;
                 ELSE
                    IF NextOp = 4 THEN      /*  '&'  AND                       */
                       DO;
                          TFStack(nStack-1) = TFStack(nStack-1) & TFStack(nStack);
                          nStack = nStack - 1;
                       END;
                    ELSE
                       IF NextOp = 5 THEN    /*  '~'  NOR                     */
                          TFStack(nStack) = ~ TFStack(nStack);
                       ELSE
                          IF NextOp = 0 THEN  /* End of RPN List              */
                             DO;
                                IF TFStack(nStack) THEN
                                   DO;  /* It's an article HIT                */
                                      HitCount = HitCount + 1;
                                      DO k = 1 TO nLine-1;
                                         PUT FILE(LstFCB) SKIP LIST (Line(k));
                                      END;
                                   END;
                                RETURN;
                             END;
                          ELSE  /* Should have only had 0,3,4,5 in stack      */
                             DO;
                               PUT FILE(LstFCB) LIST(Buzz);
                               PUT SKIP(2) LIST ('ERROR - #1',
                                                 ' Illegal Operator Code');
                               AbortSw = True;
                               RETURN;
                             END;
     END;
```

```
      IF nStack <= 0 THEN
         DO;
            PUT FILE(LstFCB) LIST(Buzz);
            PUT SKIP(2) LIST('ERROR - #2',
                        ' True/False Stack Reduced to 0');
            AbortSw = True;
            RETURN;
         END;
   END;
  RETURN;
END Execute;


RPNSequence:  PROC;  /* Put operands and operators in RPN sequence        */
   DO i = 1 to NumOps + 1;
      RPNList(i) = 0;      /* Final vector of operands/operators in RPN    */
      Stack(i) = 0;        /* Temp hold stack for lower priority operators */
   END;
   nList = 1;
   nStack = NumOps;
R1:  DO i = 1 TO NumOps +1;
      NextOp = OpVector(i); /* operands <= -2, end signal = 0, ( = 1,      */
                            /* ) = 2, | = 3, & = 4, ~ = 5                  */
      IF NextOp >= 0 THEN
         DO;        /* Have end signal or operator                         */
GetStackOp:
          StackOp = Stack(nStack);
          IF NextOp > StackOp THEN
             IF StackOp = 1 & NextOp = 2  THEN
                nStack = nStack + 1; /*  matching parens - kill both      */
             ELSE
                DO;  /* put lower priority operator on stack              */
                   nStack = nStack - 1;
                   Stack(nStack) = NextOp;
                END;
          ELSE
             IF NextOp = 1 THEN
                DO;  /* always put ( on stack                             */
                   nStack = nStack - 1;
                   Stack(nStack) = NextOp;
                END;
             ELSE
                IF NextOp = 0 & nStack = NumOps THEN  RETURN;
                ELSE
                   DO;
                      RPNList(nList) = StackOp;  /* Put Op in RPN List  */
                      nStack = nStack + 1;       /* and Pop Stack       */
                      nList = nList + 1;
                      GO TO GetStackOp;
                   END;
         END;
      ELSE  /* Have operand (key word)                                     */
         DO;
            RPNList(nList) = 5 + i;
            nList = nList + 1;
         END;
   END R1;
   RETURN;
END RPNSequence;

Separate:  PROC;  /* This program separates the operators and operands    */
    ParenNo = 0;
    SColPtr = 1;       /* String Column Pointer                           */
    NumOps = 0;        /* Number of Operators and Operands Counter        */
S1: DO WHILE(SColPtr <= SLength);
       FlagLoc = INDEX(BString,'^^^');  /* Find [next] flag (^)          */
       IF FlagLoc = 0 THEN         /* No [more] flags, so                 */
          FlagLoc = SLength + 1;  /* set Flag Location past line end      */
       KWLength = FlagLoc - SColPtr;          /* Key Word Length          */
       IF KWLength ~= 0 THEN  /* If not zero, have operand                */
          DO;   /* SepOp = Vector of separated operators and operands     */
             NumOps = NumOps + 1;
             SepOp(NumOps) = TRANSLATE(SUBSTR(BString,SColPtr,KWLength),
                Upper,Lower);      /*  Make operand all upper case        */
             OpVector(NumOps) = -2 - ParenNo;
          END;
       SColPtr = FlagLoc + 2;  /* Process following operator               */
       IF FlagLoc <= SLength THEN
          DO;
             NumOps = NumOps + 1;
             SepOp(NumOps) = SUBSTR(BString,FlagLoc,2);
             Operator = SUBSTR(SepOp(NumOps),2,1);
             CALL Assign();   /* Change operator character to number     */
             OpVector(NumOps) = OperatorNo;
             SUBSTR(BString,FlagLoc,1) = ' '; /* Erase flag              */
          END;
    END S1;
    OpVector(0) = -1;             /* Start signal                        */
    OpVector(NumOps + 1) = 0;     /* To be End signal                    */
    SepOp(0) = 'start';
    SepOp(NumOps + 1) = 'end';
    /*  Take care of errors                                               */
    IF ParenNo ~= 0 THEN
       DO;
          PUT FILE(LstFCB) LIST(Buzz);
          PUT SKIP(2) LIST('Parentheses Do Not Match');
          AbortSw = True;
       END;
    DO i = 1 TO NumOps;  /*  Print out separated string and number vectors */
        IF OpVector(i) = 6 THEN  Adj = '**** ERROR';
        ELSE   Adj = ' ';
       PUT SKIP EDIT(i,OpVector(i),'"',SepOp(i),'"',Adj)
                   (2F(5),X(2),A,A,A,X(3),A);
    END;
    RETURN;
 END Separate;

Legal:  PROC;  /*  Check legality of string - uses True/False Table      */
             /*  Check operand/operator against following operand/operator */
    PUT SKIP;
L1: DO i = 0 TO NumOps;
       j = OpVector(i);  /* Set up subscript for 1st operand/operator     */
       IF j < -2 THEN  j = -2;
       k = OpVector(i + 1);  /*  Set up subscript for 2nd operand/operator */
       IF k < -2 THEN  k = -2;
       IF Table(j,k) = False THEN
          DO;
             PUT SKIP EDIT('Illegal Sequence: ',SepOp(i),SepOp(i+1))(A,A,A);
             AbortSw = True;
          END;
    END L1;
 END Legal;

END KWIRP;
```

Book Review

Hal Chamberlin's 'Musical Applications of Micro-Processors'

by Jon Bondy

Musical Applications of Micro-Processors, by Hal Chamberlin. Hayden Publishing Co., 1980.

Although there have been many articles on "computer music" in the various personal computer magazines, often they have been inaccessible to some readers, perhaps because of their attempts to speak to readers with too wide a range of backgrounds. Some articles are so technical that the average reader cannot follow them, while others are completely lacking in substance. Until recently, there has been no single source to which one could turn in order to both get a general overview of the subject, and also learn enough of the details to create interesting music software. Hal Chamberlin, the earliest and most prolific creator of personal microcomputer music software and systems, has written a book which bridges this information gap, allowing people with almost any background to learn about computer music.

Chamberlin has written extensively in personal microcomputer publications in the past, and his book published by Hayden (1980), is a comprehensive guide to a wide variety of both analog and digital computer musical applications. It is almost 650 pages long, and just a quick scan through the table of contents shows you how much ground he covers: music synthesis principles, sound modification methods, voltage control methods (analog synthesizer), direct computer synthesis methods, an overview of microprocessors, basic analog modules, digital-to-analog and analog-to-digital conversion, signal routing in analog and analog/digital synthesizers, organ keyboard interfacing, control sequence graphic display and editing, digital tone generation, digital filtering, percussive sound generation, source signal analysis, digital hardware for music synthesis, and music synthesis software (whew!). There must be something here for everyone.

As I read the book, I searched for a way to show just how much could be learned by reading it, especially from the first few sections that introduce the principles of music synthesis (very rapidly, but also very clearly). I decided to write down all the terms which were defined in the first hundred pages—so here goes: frequency, pitch, amplitude (peak to peak and root mean square (RMS)), decibel (dB), sine, timbre, fundamental frequency, overtone/harmonic, spectrum, white noise, frequence modulation (FM), vibrato, amplitude envelope, attack, decay, tremolo, echo, reverb, preverb, nonlinear systems, clippers, filters, gain, phase shift, high pass filter, low pass filter, band pass filter, band reject filter, center frequency, Q, cutoff frequency, cascaded filters, formants, comb filter, flanging, ring modulator, spectrum inverter, envelope modification, chorusing, formant tracking, voltage controlled synthesizer, exponential versus linear control voltages in analog synthesizers, voltage controlled oscillator (VCO), voltage controlled amplifier (VCA), voltage controlled filter (VCF), envelope generator, portamento, sequencer (for analog synthesizer), sample and hold, and noise generator.

It is amazing not only that all of this is covered, but that it is covered in a readable, but not over-simplified manner. Chamberlin's book explains without getting so technical that the reader is scared off. On the other hand the book *does* get very specific when necessary. There are discusions about—and Basic program listings for—the following functions: analysis of interpolation noise for various interpolation schemes, conversion of Fourier coefficients to wave forms (with phase randomization to minimize peak amplitude!), slow Fourier transform, Fast Fourier Transform (FFT), and a Transversal Filter. In addition, there are 6502 Assembly language listings for a state-variable digital filter and a program to transform Fourier coefficients to a wave form

The book is not only informative in the sense of covering the topics thoroughly, but it also gives some interesting asides which provide insight into the synthesis techniques used to create some musical effects—whose sources I, at least, did not understand. For instance, on page 47 Chamberlin describes how one can use tape echo with a gain which is manually varied near unity to create whooshing "saucer landing" effects. On page 63, he points out that the "voiced portions of human speech (vowels and dipthongs) may be simulated with a harmonically rich tone and two to four variable bandpass filters;" he indicates that this is a bit oversimplified, but it does give the reader a feel for the utility of the concepts discussed.

On page 81, he describes how one can derive the accuracy requirements of the control voltages (and their generation and processing circuitry) for an analog synthesizer. At the lowest frequencies which can be heard by man, the allowable frequency deviation for a musical tone is 0.2 Hz; when this is expressed as "full-scale

Jon Bondy, Box 148, Ardmore, PA 19003.

accuracy," this becomes 0.2/20,000 Hz, or 0.0001 percent. For a synthesizer with a control voltage range of 10 volts, this would amount to deviations smaller than 100 microvolts, an unrealistic requirement for signals which will, among other things, be run through patch cards for a number of feet. Chamberlin points out that this kind of accuracy is almost impossible to obtain, except in very expensive laboratory instruments, so it would be uneconomical for this kind of accuracy to be required by musical synthesizers that are produced in large quantities. The way to get around this problem is not to use a linear control voltage, but rather to use an exponential one; this increases the minimal deviation to 14.5 millivolts, a decrease in noise succeptibility of two and one half orders of magnitude! I know that my brief description of this topic is not a clear one, but Chamberlin's discussion is, and it does indicate the kinds of interesting discussions that fill the book.

I program mostly in Pascal, and generally have no interest in assembly language "tricks," but I was intrigued by Chamberlin's observations on interpolation. Consider the problem of determing F(X) when F(X1) and F(X2) are known. Normally I think of this interpolation as:

$$F(X) = F(X1) + (X - X1) * [(F(X2) - F(X1)) / (X2 - X1)]$$

and I consider that the division is a requirement; Chamberlin points out a number of things which make this much simpler than it appears. He says that since the value (X2 -X1) is a constant for interpolation of tables of evenly spaced values, it becomes a constant and need not be computed each time. Also, if the number of entries in the table is a power of two, then the division reduces to a simple shifting operation. I would have needed one addition, three subtractions, a multiplication, and a division; Chamberlin only requires an addition, two subtractions, and a multiplication.

Other topics of interest which are covered include: how to determine sound fidelity (frequency response and signal to noise (S/N) ratio) for digitally sampled audio signals, analysis of techniques for transitioning between two wave forms in terms of noise due to mismatched phases in the wave forms, windowing of signals for spectral analysis, generation of waveforms using FFTs and time varying frequency spectra, digital filter analysis and design, waveform interpolation techniques, and pitch measurement and auto-correlation.

Lest it seem that I am too enthusiastic, let me point out that in at least one area, that of analysis of source signals, Chamberlin doesn't provide enough information to actually allow one to perform the signal windowing required to start the analysis. Moreover, he mentions only in passing another technique, resampling, which can be easily applied to the windowing problem. He does provide, however, a list of references from which the details of these techniques can be obtained.

In conclusion, I found Chamberlin's book to be well written, with enough introductory material that people unfamiliar with either digital computers or musical synthesis techniques could build up their background to render the rest of the book understandable. I believe that this book will become a classic in the next few years.

SOFTWARE DIRECTORY

Program Name: Milestone
Hardware System: CP/M-86 & 80 x 24 display.
Minimum Memory Size: 56K
Description: Project management software package based on critical path network analysis techniques. Useful for any project that can be broken into a series of distinct tasks, each with a duration, a level of manpower and a cost. Automatically lays out each job against a time scale showing which tasks are critical and which can be delayed. Also displays manpower and expenses versus time, as well as totals and project completion data. Original plan can even be altered during the course of a project to reveal impact of any scheduling changes.
Release: June 1981
Price: $295
Included with price: Disk and Manual
Where to purchase it:
Organic Software
1492 Windsor Way
Livermore, CA 94550
(415)455-4034

Program Name: TCS Business Accounting Package
Hardware System: Any system using Microsoft Basic, CP/M
Minimum Memory Size: 48K
Language: Microsoft Basic
Description: A fully integrated business software arsenal including General Ledger (provides immediate financial information for your company by keeping thorough records of all financial transactions); Accounts Payable (maintains complete vendor/voucher history including check writing capabilities); Accounts Receivable (instant customer accounts information - current and aged - with complete invoicing and statement capabilities); Payroll (calculates payroll for every type employee while maintinaing monthly, quarterly and yearly totals for reporting purposes in multiple states. User modifiable tax tables. W-2, 941's, checks, etc.); Inventory Management (detailed inventory records, allows multiple item location and dept. ID., simplified posting and new easier-to-read reports.)
Release: GL,AP,AR,PR - 1978; IM - 1981
Price: Inventory: $400.00; GL,AP,AR,PR: $500.00 (for Microsoft Basic 4.5); IM: $400.00 (MBasic 5. x); GL,AP,AR,PR: $600.00 (for Microsoft Basic 5.X) GL,AP, AR, PR: $850.00 (for compiled version running on Microsoft Compiler.)
Included with price: Program disk, 600 page user manual, sample data and source code.
Author: TCS Software
Where to puchase it:
TCS Software
5582 Peachtree Road
Atlanta, GA 30341

Program Name: Master Disk Catalog
Hardware System: Micropolis 5 1/4" Drives or Single Density 8" CP/M
Minimum Memory Size: 32K
Language: Assembly—8080
Description: The program maintains a record of the files from your disks on a single catalog disk. As you work on programs or files, comments and dates can be put into them which may then be recorded on the catalog disk. This information may be searched for all occurences of particular file names, for text in your comments, or for dates. File names and disk names may include CP/M "wildcard" characters.
Release: May 1981
Price: $35 & postage
Included with price: Disk, Manual
Where to purchase it:
Mendocino Software
P.O. Box 1564
Willits, CA 95490
(707)459-9130

Program Name: Micro Link
Hardware System: 8080/Z80 System & Modem
Minimum Memory Size: 16K
Description: The Micro Link program enables microcomputer users to communicate over telephone lines. Files transmitted automatically. Readable word-wrapped display fitted to any screen width, a host of options with convenient default settings, and simple, fast user commands are Micro Link features. Micro Link supports originate and answer mode, full- and half-duplex and operates at equipment baud rate. Files may be transmitted in character, line or memory block protocol. The program may be used with others in Basic, assembly or other languages.
Release: June 1981
Price: $89
Included with price: Object code and manual. Supplied on 16 sector, 77 track, 5 1/4" disk (Micropolis); 8" CP/M disk.
Where to puchase it:
Wordcraft
c/o Microcomputer Software Associates
1122 B St.
Hayward, CA 94541
(415)534-2212

DISK DRIVE WOES? PRINTER INTERACTION?
MEMORY LOSS? ERRATIC OPERATION?

Don't Blame The Software!

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.

Clear up Software and System problems with an ISOLATOR!

ALL ISOLATORS: • 125 VAC, Standard 3-prong plug • 1875 W MAX Loabd — 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-1	-3 individually filtered sockets	$ 62.95
ISO-4	-6 individually filtered sockets	106.95
ISO-2	-2 filtered banks; 6 sockets	62.95
ISO-5	-3 filtered banks; 9 sockets	87.95

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

ISO-6

ISO-6	-3 switched, filtered sockets	$141.95
ISO-8	-5 switched, filtered sockets	178.95
ISO-3	-3 super filtered sockets	94.95
ISO-7	-5 super filtered sockets	154.95

*SUPER ISOLATORS — Cure for severe interference problems. Useful for Industrial applications and heavy duty controlled equipment or peripherals.

ISO-7

- Dual Balanced Pi Filtered sockets
- Spike/Surge Suppression — 2000 Amps, 8/20 usec

*CIRCUIT BREAKER any model (add-CB) . . . . . ADD 8.00
*CKT BKR/SWITCH/PILOT any model (CBS) . . . ADD 16.00

Master-Charge, Visa, American Express
TOLL FREE ORDER DESK 1-800-225-4876
(except AK, HI, MA, PR & Canada)

ESP Electronic Specialists, Inc.
171 South Main Street, Natick, Mass. 01760
Technical & Non-800 1-617-655-1532

INTRODUCTORY OFFER!

*Turn your Micro-Computer into a Mini-Computer. Try the world's #1 programming language-**COBOL!** Finally at a price you can afford and with no risk!*

Introducing...

NPS-MICRO-COBOL

This is the Naval Post Graduate School Cobol that you've heard so much about. Designed to pass the stringent government Hypo-COBOL tests used by GSA in their Compiler Certification Program. This is the first public release of version 2.1. This is an elaborate ANSI-COBOL subset. Comes complete with users manual in DeLuxe three-ring binder.

- **Perfect for learning COBOL.**
- **Perfect for teaching COBOL**
- **FREE sample programs included**
- **Runs in 24K**
- **Requires 8080, Z-80®, or 8085 and standard CP/M® system**
- **Provided on standard 8" disk or Northstar Double Density CP/M 5"**

Only $69.95!

FREE ALGOL INCLUDED!

FREE BONUS. All purchasers receive a FREE copy of NPS-ALGOL at no extra cost. A favorite language in Europe, ALGOL is the original structured language. Comes with FREE sample programs.

MONEY-BACK GUARANTEE. *If you're not completely satisfied with this software. You may return it within fifteen days for any reason and get a full refund.*

Send Check, Money Order or Credit Card information and order a copy today! Please add $2.50 shipping and handling on all orders.

Credit Card buyers: For Extra Fast service call (415) 527-8717

Order from: The Software Review
704 Solano Avenue, Albany, CA 94706

Yes, I want to run COBOL on my system! Enclosed find $69.95 plus $2.50 shipping/handling (California residents please add appropriate sales tax). I will receive the NPS-COBOL system plus a FREE copy of NPS-ALGOL. I understand that I may return the software within 15 days if not completely satisfied for a full refund.

NAME _____
COMPANY _____
STREET _____
CITY _____
STATE _____ **ZIP** _____
AMOUNT ENCLOSED *$* _____
Disk size desired: 5" 8"

☐ Check Enclosed ☐ VISA
☐ UPS C.O.D. ☐ Mastercharge
Card number _____
Expiration Date _____
Signature _____

☐ Check here for more information

CP/M is a trademark of Digital Research and Z-80 is a trademark of Zilog.

Software Shops

—Arizona—

Creative Software Systems: Systems integration and custom software (BASIC, PASCAL, Z-80 assembler). Small business and word processing systems. 632 Camelot Dr., Sierra Vista, AZ 85635.
Phone (602)458-6063.

—California—

Clear Systems
309 Santa Monica Blvd., # 404
Santa Monica, CA 90401.
(213)394-7740.
(making complexity serve simplicity)

—Colorado—

Random Factors LTD.: Industrial test, control & data acquisition — Hi speed & accuracy for S100 & STD-BUS. From software to complete systems. W.K. Borsum, P.E., Random Factors LTD. Castle Rock, CO 80104. (303) 688-5338.

Nelson Engineering: We write applications software for all micro-based systems in Asembly language, Basic, and Pascal. (213) 390-2963; 13450 Maxella Ave. G185 Suite 142, Marina Del Rey, CA 90291.

—Massachusetts—

MICROFT INC.: Customization of CP/M-80, MP/M, CP/M-86 and other operating systems. Full range of consulting services in microsystems software (systems, utilities applications), product selection, hardware. Contact: Tom Campbell, Chief of Technical Staff, P.O. Box 128, E. Falmouth, MA 02536. Phone (617)563-3807.

—New Jersey—

New Jersey Software Services: Full range of CP/M, S-100 services.
- System design
  - Business applications
  - Real-time systems
  - Mathematical analysis
- Software creation/customizing
  - 8080 assembly language
  - Z-80 assembly language
  - BASIC
  - FORTRAN
- Product evaluation/selection
  - Hardware Software
- In house training
- Telecommunication service
- Voice I/O applications

Contact C. A. Ryan, 6 Village Circle, Westfield, N.J. 07090 (201) 233-9297.

—Washington—

CHI ENERGY: Custom programs and package modification in Assembler, Basic & C languages; CP/M and real time systems. Contact: Mark A. Carlson, P.O. Box 55145, Seattle, WA 98155. (206)364-5463

Software Directory cont'd...

Program Name: Micro Link
Hardware System: CP/M 1.4 or Micropolis with serial port & modem
Minimum Memory Size: 16K
Description: Micro Link program enables microcomputer users to communicate with each other, large computers and terminals over telephone lines. Files may be prepared in advance and transmitted automatically. The entire two-way record of communication may be recorded in memory and on disk. Features include readable word-wrapped display fitted to any screen width, a host of options with convenient default settings, and simple, fast user commands. Micro Link scans The Source, other data bases and bulletin boards quickly, recording segments that interest the user for review off line. The Micro Link hosts another computer or a terminal. Micro Link supports originate and answer mode, full- and half-duplex and operates at equipment baud rate. Files may be transmitted in character, line or memory block protocol. The program may be used with others in Basic, assembly or other languages.
Release: April 1981
Price: $89
Included with price: 8" or 5 1/4" disk and manual
Where to purchase it:
Wordcraft
c/o Microcomputer Software Associates
1122 B St.
Hayward, CA 94541
(415)534-2212

Program Name: Information Master
Hardware System: CP/M Operating Sytem
Minimum Memory Size: 32K
Language: Object program only
Description: Information Retrieval Program handling a large body of static information requiring flexible access. This is accomplished by creating a compact index to the text files based on key words or phrases designated by the user. Retrieved data files may be created with any CP/M compatible text editor or user program in a free form format. Main program maintains a dictionary of all key words indexed, and rapidly searches the index on Boolean (AND & OR) combinations of key words. The program directs the user to the disk containing the data. Retrieved data can be displayed, printed or written to another file. The system is ideal for handling abstracts from scientific literature, product literature, record and book collections, correspondence, recipes, and applications where data is not frequently modified but a large base is required.
Release: 1979
Price: $37.50 plus $1.50 shipping and handling
Included with price: Instruction manual, 8" or 5 1/2" disk containing program & sample data base.
Author: William B. Brogden, Island Cybernetics
Where to purchase it:
Elliam Associates
24000 Bessemer Street
Woodland Hills, CA 91367 ■

NEW PRODUCTS

Data Entry System for CP/M

Southern Computer Systems, Inc. of Birmingham, Alabama announces the release of "RADAR," the first truly general purpose data entry system for CP/M environments. RADAR (Random Access Data Acquisition & Retrieval) will convert any CP/M computer into a powerful data entry station with complete key-to-disk capabilities.

RADAR provides full check digit verification, double key verification, sixteen totalling accumulators, date check, data type verification by masks, controlled access, etc. RADAR is designed and optimized for the highest possible throughput in demanding data entry environments. Written entirely in machine language, RADAR is extremely fast, guiding but never obstructing the operator. RADAR may be used to edit and/or update existing data. It supports access to individual data records by relative position or record content. Records may be added, deleted, inserted, or appended easily with full editing capabilities.

RADAR is the ideal replacement for key punch machines and 3741 type key-to-disk systems, combining the speed of key punch with the convenience of CP/M's virtual file access. RADAR's verification mechanisms insure the most accurate entry possible.

A single user license is $495, and special multi-machine license arrangements are available. The user's manual is available separately for $25. For more information, contact Southern Computer Systems, Inc., P.O. Box 3373A, Birmingham, AL 35255; (205)933-1659.

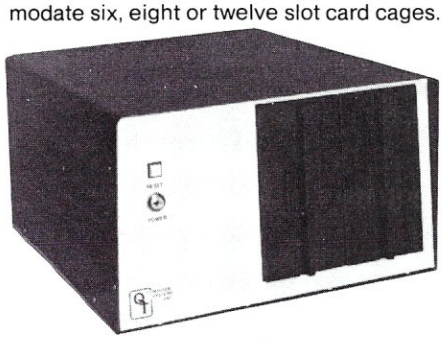
S-100 8048 In-Circuit Emulator

An in-circuit emulator for 8048 family of one chip microcomputers is now S-100 bus compatible. MICE-48 is capable of emulating the Intel (or equivalent) 8035/8039/8048/8049/8748 parts, the National 8040/8050 parts, plus the new 80C48/80C35 CMOS versions from NEC and others. Contained on one standard S-100 board, MICE-48 runs under the CP/M operating system on any 8080 or Z-80 based CPU. A ribbon cable with a buffer assembly connects the emulator to the user system by replacing the user's 8048 processor. Emulation is done in real time using user's clock or crystal at speeds up to 11 Mhz.

MICE-48 also features trace, unlimited number of breakpoints and display/modify of program memory, external RAM, registers, I/O ports and flags. A mapping command allows the user to map program memory to user's PROM/EPROM or to emulator's RAM. It comes with all supporting software including a powerful 8048 Macro Assembler. Pricing: $950.00. Signum Systems, 726 Santa Monica Blvd., #217, Santa Monica, CA 90401; (213)451-5382.

S-100 Mainframe

QT Computer Systems has introduced a new series of mainframes. The Mainframe+ Disk Drive is one of the most versatile dual 8" mainframes on the market. It will accommodate six, eight or twelve slot card cages.



The MF+DD accepts Shugart, Remex, Siemens and other standard 8" disk drives. Other features include IEEE S-100 compatibility, steel cabinet, 25-amp power supply, keyed power switch, reset switch on front panel, line filter for electrical noise suppression and a circuit breaker for safe operation. Other QT mainframes include the MF+MD, which accepts two 5 1/4" disk drives with remaining space for either a six, eight or twelve slot Silence+ motherboard, and the MF+IND6, a rugged mainframe for industrial control applications. QT Computer Systems, Inc., 15620 South Inglewood Avenue, Lawndale, CA 90260; (800)421-5150.

Nine-Track IBM/ANSI Standard Format Tape Drive

The new nine-track tape system from Cromemco offers microcomputer users an excellent means of data backup, long-term recordkeeping (or archival storage), and data exchange with other IBM/ANSI standard systems using the 1600 BPI format. The tape drive and intelligent controller, which can store over forty million bytes on a reel of tape, are able to read, write, or search at tape speeds of 25 and 100 IPS (inches per second) without intervention from the host CPU. The system includes self-test and diagnostic capabilities. The tape drive can operate in either a start/stop mode, where the reels are brought to rest in the inter-record gaps, or in the streaming mode where data and gaps are written "on-the-fly" (e.g. hard disk backup).

As many as eight drives may be daisy-chained from a single interface card, and up to sixteen such interface cards may be connected to each I/O processor. Multiple processors also may be included in a single main system. The expansion bus used to interconnect the I/O processor card and interface cards is called the C-bus, and it is common to other intelligent peripherals offered by Cromemco.

The tape system, including transport, formatter, intelligent controller and all cables sells for $7995.00, (Model TDS). Cromemco, Inc., 280 Bernardo Ave., Mountain View, CA 94043; (415)964-7400.

Symbolic Debugging System

Southern Computer Systems, Inc. of Birmingham, Alabama announces an enhanced version of its powerful symbolic debugging system, "RAID," which permits RAID to be used for debugging programs created by upper level language compilers including Fortran, PL/M, PL/MX, Basic, etc. RAID may, of course, also be used for debugging assembly language programs.

RAID is a fully symbolic debug program that operates in two modes: as an interpreter, and as a real-time monitor. The user may switch back and forth between the two modes at will. The interpretive mode pro-

New Products cont'd...

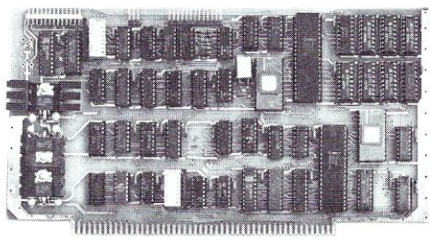
vides a totally secure (crash-proof) environment especially useful to those who are not yet fluent in assembly language, providing the same sort of interactive capabilities associated with Basic interpreters, but at the assembly language level.

RAID's command arsenal is over seventy strong, and includes memory searches for bytes, words, or strings; up to seven simultaneous breakpoints; tracing by single instruction, subroutine nesting, or prime path; memory protection; programmable data dumps at breakpoints; user specifiable iteration through n breakpoints; direct I/O to ports; direct access to disks by track and sector; loading and saving memory images and symbol tables; and much more.

RAID is available to run on either Digital Research's CP/M operating system or Intel's ISIS operating system. The standard CP/M version of RAID is $250.00 for a single user license. The typeset users manual is available separately for $25.00. A four page brochure is also available from Southern Computer Systems, 586 Shades Crest Road, P.O. Box 3373A, Birmingham, AL 35255; (205)933-1659.

Smart S-100 Video I/O Interface

The VIO-X S-100 Video I/O Interface board is an intelligent video controller for the S-100 bus. It uses the Intel 8275 CRT Controller with an 8085 processor on board, and features a port-mapped interface. This allows operation without use of the host's memory and without firmware overhead on the system processor. Port-mapped I/O also allows compatibility with most S-100 processors. The VIO-X features two pages of memory allowing the use of protected fields or forward/reverse scrolling. Firmware is available to emulate the Soroc 120 or ADM 3A control/escape command set. Full



ASCII upper and lower case alphanumeric character set are included in the 2716/2732 character generator. Screen display is a full 80 character by 24 line format. Both block and conversational mode are supported, providing on-screen editing. The 8275 supports several video attributes including character flash, underline, inverse video, and dim. An audio ouput port for the ASCII BEL signal, a lightpen port, and an 8-bit parallel keyboard input port are provided on the board. The price is $295.00. WW Component Supply Inc., 1771 Junction Ave., San Jose, CA 95112; (408)295-7171.

Software Package Brings UNIX Features To CP/M

Unica is a package of software instruments which offers many of the facilities and commands of UNIX to the CP/M user. All

Microsystems — the CP/M* and S-100 User's Journal

CP/M is the software bus!+ S-100 is the hardware bus for sophisticated microcomputer users!

If you are a CP/M user, on any system—S-100, Apple, TRS-80, Heath, Ohio Scientific, Onyx, Durango, Intel MDS, Mostek MDX, etc—after all CP/M is the Disk Operating System that has been implemented on more computer systems than any other DOS—then *Microsystems* magazine is the "only" magazine published specifically for you!

Or, if you use an S-100/IEEE-696 based computer—and the most sophisticated microcomputer systems available use the S-100/IEEE-696 hardware bus—then *Microsystems* magazine is the "only" magazine published specifically for you!

We started publishing *Microsystems* almost two years ago to fill the void in the microcomputer field. There were magazines catering exclusively to the TRS-80, Apple, Pet, Heath, etc. system users. There were also broad based publications that cover the entire field but no one system in depth. But no magazine existed for CP/M users—nor did one exist for S-100 users.

The why and what of a software bus

First of all what is a "bus?" And why do we call CP/M "the software bus?"

A "bus" is a technique used to interface many different modules. Examples are the "S-100/IEEE-696 Bus" and the "IEEE-488 Bus." These are hardware buses that permit a user to plug a bus-compatible device into the bus without having to make any other hardware modifications and expect the device to operate with little or no modification.

CP/M is a Disk Operating System (DOS). It was first introduced in 1974 and is now the oldest and most mature DOS for microcomputer systems. CP/M has now been implemented on over 250 different computer systems. It has been implemented on hard disk systems as well as floppy disk systems. It is supported by two user groups (CP/M-UG and SIG/M-UG) that have released over 80 volumes containing over 2,000 public domain programs that can be loaded and run on systems using the CP/M DOS. Add to this another 1,500 commercially available CP/M software packages and you have the largest applications software base in existence.

CP/M is the only DOS for micros that has stood the test of time (seven years) with the highest level of compatibility from version to version. And over the years this compatibility has been maintained as new features have been added.

This is why we say "CP/M is the software bus" and why *Microsystems* magazine is vital to providing CP/M users with technical information on using CP/M, interfacing to CP/M, new CP/M compatible products and for CP/M users to exchange ideas.

Why support the S-100 bus?

S-100 is currently the most widely used microcomputer hardware bus. It offers advantages not available with any other microcomputer system. Here are a few of the advantages:

S-100 is processor independent. There are already thirty different S-100 CPU cards that can be plugged into an S-100 bus computer. Nine 8-bit microprocessors are available: 6502, 6800, 6802, 6809, 2650, F8, 8080, 8085 and Z80. Eight 16-bit microprocessors are available: 8086, 8088, 9900, Z8000, 68000, Pascal Microengine, Alpha Micro (similar to LSI-11) and even the AMD2901 bit slice processor. Take your pick from the incredible offerings.

S-100 has the greatest microcomputer power. What other microcomputer system has direct addressing of up to 16 megabytes of memory, up to 65,536 I/O ports, up to 10 vectored interrupts, up to 16 masters on the bus (with priority) and up to 10 Mhz data transfer rate? You will have to go a long way to use up that computing power.

S-100 is standardized. The S-100 bus has been standardized by the IEEE (Institute of Electrical and Electronic Engineers) assuring the highest degree of compatibility among plug-in boards from different manufacturers. And, *Microsystems* has published the complete IEEE S-100/696 standard (all 26 pages).

S-100 has the greatest hardware support. There are now over sixty different manufacturers of about 400 different plug-in S-100 boards. Far greater than any other microcomputer system.

With all these advantages is it any wonder that S-100 systems are so popular with microcomputer users who want to do more than just play games?

For the serious computer user.

Each issue of *Microsystems* brings you the latest in the CP/M and S-100 world. Articles on applications, tutorials, software development, product reviews, and lots more, to keep you on top of the ever changing microcomputer scene.

And if you are an S-100 system user using other operating systems (e.g. North Star) *Microsystems* also supports you.

Get your copy today

Order your susbscription to *Microsystems*. Send $10 for one year (6 issues), $18 for two years (12 issues) or $24 for three years (18 issues). If you prefer, call our toll-free number, **800-631-8112** (in NJ 201-540-0445) to put your subscription on your MasterCard, Visa or American Express card. Canadian/Mexican and other foreign surface subscriptions are $15 and $25, respectively, per year and must be prepaid. We guarantee that you will be completely satisfied or we will refund your subscription.

Join thousands of *Microsystems* subscribers like Jim Johnstone of Los Altos CA, who said "*Microsystems* has lived up to your promises and my expectations. Congratulations."

the CP/M* and S-100 user's journal
MICROSYSTEMS

a Creative Computing publication

39 East Hanover Avenue
Morris Plains, NJ 07950, USA
Toll-free 800-631-8112
(In NJ, 201-540-0445)

* registered trademark Digital Research + registered trademark Lifeboat Associates

News & Views cont'd...

Unica commands support redirection of standard I/O, connection of subsequent commands via "pipes," extended filenames with user numbers, and wildcard filenames based on pattern-matching rather than character-masking. The UNIX-like Unica commands include sr, which searches multiple files for a pattern in a grep-like fashion; sp, a spelling error detector with a 20,000 word dictionary; ln, which forms links (aliases) to files; sc, a source file comparator with resynchronization; srt, a file sorter; ls, an intelligent directory lister; cat and hc, which do vertical and horizontal file concatenation; dm, a disk map utility; and several others.

The Unica commands are written in XM-80, which allows the versatility of assembly language while providing high-level procedure constructs. The XM-80 library contains over forty "software components," subroutine packages which implement the UNIX features as well as providing device-independent, buffered I/O; numeric/string conversion; heap space management; string manipulation; output formatting; intelligent error message publications; and program chaining, among other services. The XM-80 translator produces assembler source code, and requires Microsoft's MACRO-80 assembler system.

The Unica with XM-80 and extensive documentation is priced at $195. An X80 system running CP/M version 2 or later is required. Contact Knowlogy, P.O. Box 283, Wilsonville, OR 97070; (503)635-5701.

Tarbell Offers 1 to 20 Megabyte System

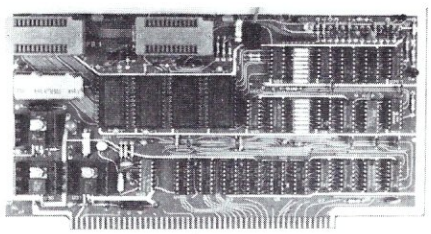
The Tarbell Empire System includes a Z80 4 MHz CPU with memory management, timer and full interrupt capability, two RS-232 serial ports with handshaking, 64KB of random-access memory (expandable to 256 KB), double density floppy disk interface, two double density floppy disk drives, cabinet, power supply and cables. The timer, interrupts and memory management make it easier to build multi-user systems.

The system software includes: Digital Research CP/M 2.2 DOS, Tarbell Disk Basic, Tarbell Database System, and all manuals and documentation. The applications software available includes word processing, inventory control with bill of materials, mailing lists, general ledger, payables, receivables, payroll with cost accounting and order entry.

Tarbell also offers the Digital Research MP/M Multi-User Operating System and four additional RS-232 serial ports. The Tarbell Empire Series is delivered assembled, tested and with a full six-month warranty on parts and labor. For further information or the name of your nearest dealer, contact Tarbell Electronics, 950 Dovlen Place, Suite B, Carson, CA 90746; (213)538-4251.

New S-100 Programming And EPROM Memory Board

SSM Microcomputer Products has introduced a dual-service board that combines programming and memory. Called PB1, it has four on-card sockets for either 4KB or 8KB of memory, depending upon whether 2708 or 2716 EPROMs are used. Two Textool programming sockets are also provided, one for a 2708, another for a 2716—both are DIP-switch programmable to any 4K boundary.



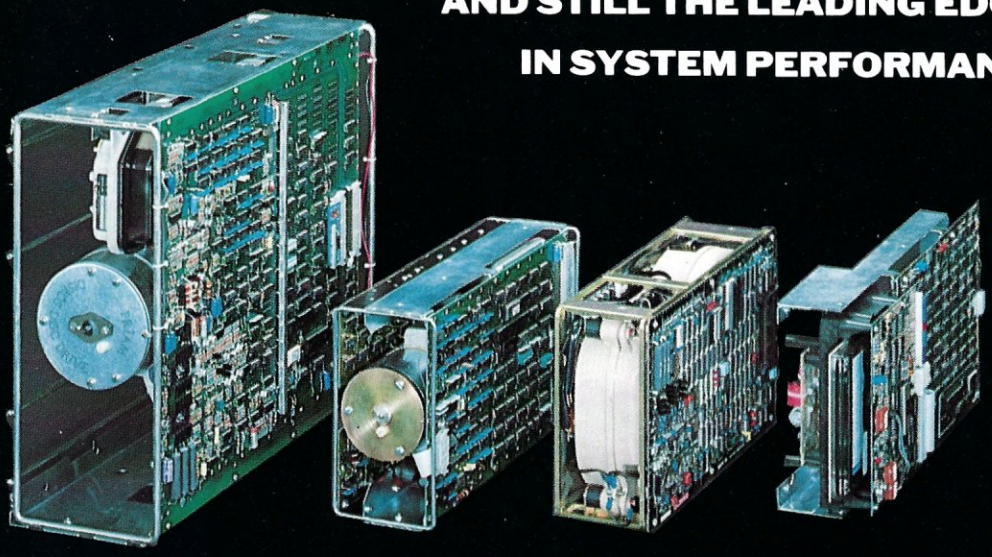
The PB1 board, with software for programming and verifying EPROMs is $265, A & T. The kit version is $179. SSM Microcomputer Products Inc., 2190 Paragon Drive, San Jose, CA 95131; (408)946-7400. ■

Advertiser Index

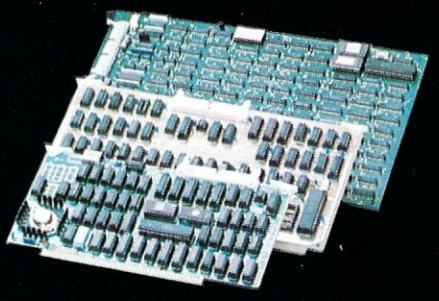
THE HARD EDGE

AND STILL THE LEADING EDGE . . .

IN SYSTEM PERFORMANCE



ADES

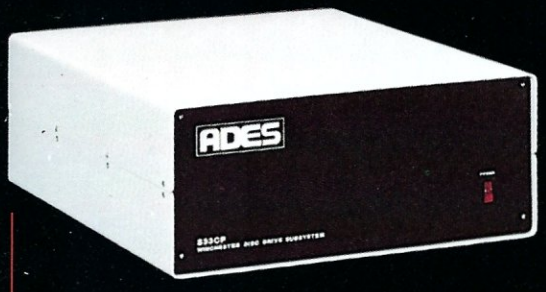


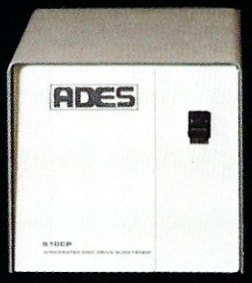
Hard Disk and Streaming Tape

reliability and versatility that can't be beat

OEM CONTROLLERS

- Multibus
- General Purpose
- S100





S33

- 31 or 62 MByte formatted

S10

- 10.5 or 32 MByte formatted

SYSTEM 8

- 8" Winchester 10 or 32 MByte formatted
- Integrated streaming cartridge backup
- Streaming backup at 5 MB/min
- Selective file backup under CP/M
- Versatile parallel I/O or DMA interface

- Lowest cost/MByte from any manufacturer
- CP/M* compatible "drop in" BIOS
- Single S100 card controller
- Fully expandable up to four drives, 8" or 14"
- Reliable high performance Priam Winchester Disks
- Immediate delivery

ADES

ADAPTIVE DATA & ENERGY SYSTEMS

2627 Pomona Boulevard • Pomona, CA 91768

Phone: (714) 594-5858

*CP/M is a trademark of Digital Research Corp.

THE BIGGEST NAME IN S-100 MEMORY PRESENTS THE BIGGEST S-100 MEMORY: 128K RAM 21

RAM 21 isn't just big, it's versatile. For 8 bit systems, it's a 128K X 8 board; for 16 bit systems, a 64K X 16 board. Addressing conforms to IEEE 696/S-100 24 bit extended address protocol.

RAM 21 is faster than today's needs to protect your memory investment tomorrow. High speed memory devices allow no-wait-states operation with today's 5/6 MHz CPUs, as well as with the coming generation of 10 MHz CPUs.

RAM 21 is fully static. Forget about the reliability and DMA problems associated with dyamic memory.

RAM 21 has the lowest current consumption in the industry, bar none. RAM 21 draws half the power (1.6A typical) of an equivalent amount of dynamic memory. The result? Less heat build-up, less strain on the power supply, lower energy costs, and greater reliability.

RAM 21 is built to work and keep on working. All RAM 21 boards are qualified under the Certified System Component high reliability program, with 200 hours of burn-in and extensive quality control testing. *

RAM 21 is not only fast, low power, dense, and versatile: it's affordable. At **$2495, RAM 21** represents exceptional quality whose value will not diminish when expanding to faster 8 bit systems or more powerful 16 bit systems.

The biggest name in memory has dropped the big one:

RAM 21 is here.

CompuPro™ division GODBOUT ELECTRONICS

OAKLAND AIRPORT, CA 94614 (415) 562-0636

How to Order: Call **415-562-0636** for the name of the authorized **CompuPro** sales center nearest you, or for placing factory direct VISA® /Mastercard® orders.

Prices shown do not include tax or shipping charges.

* **2 year limited warranty.**