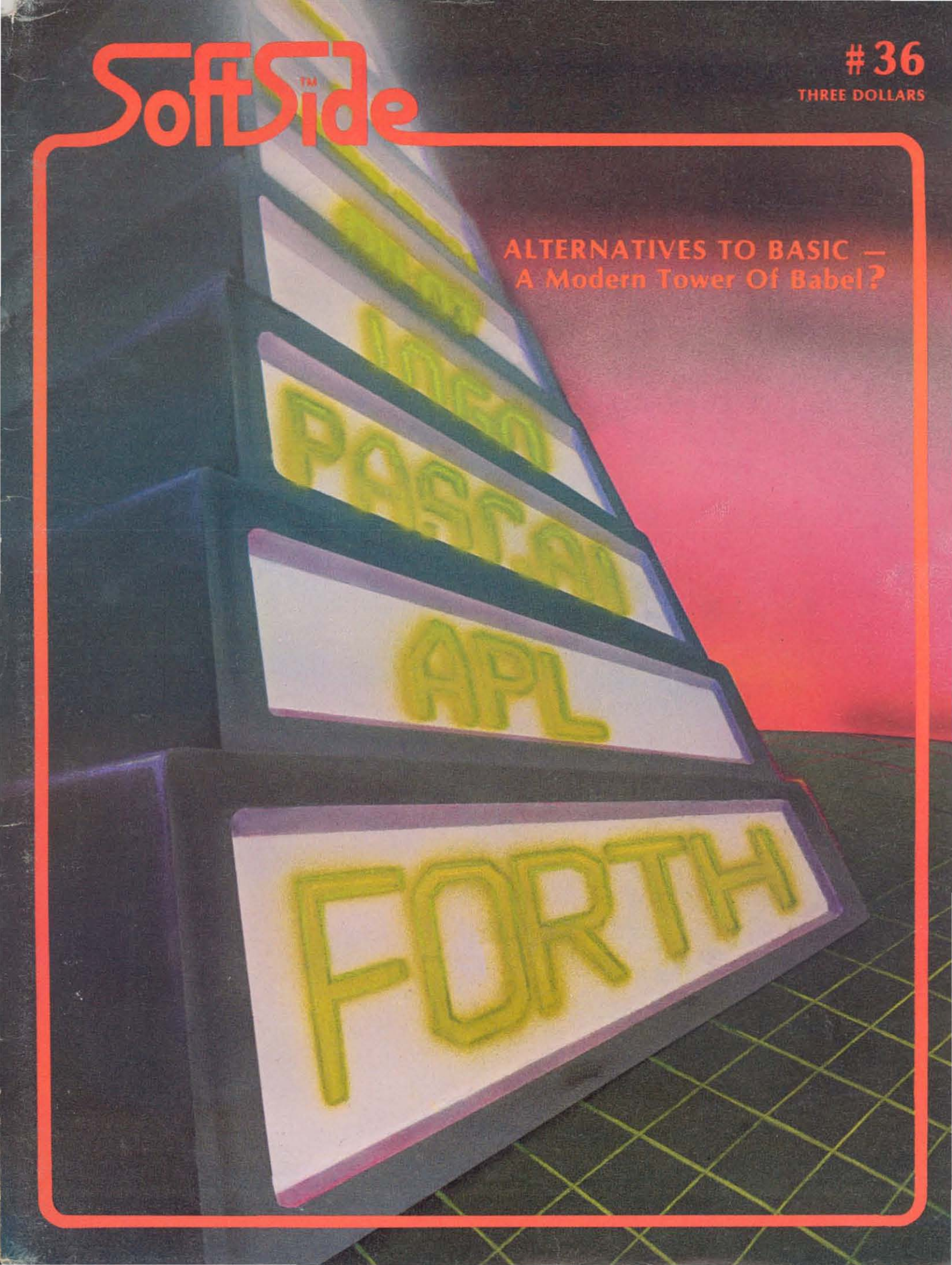
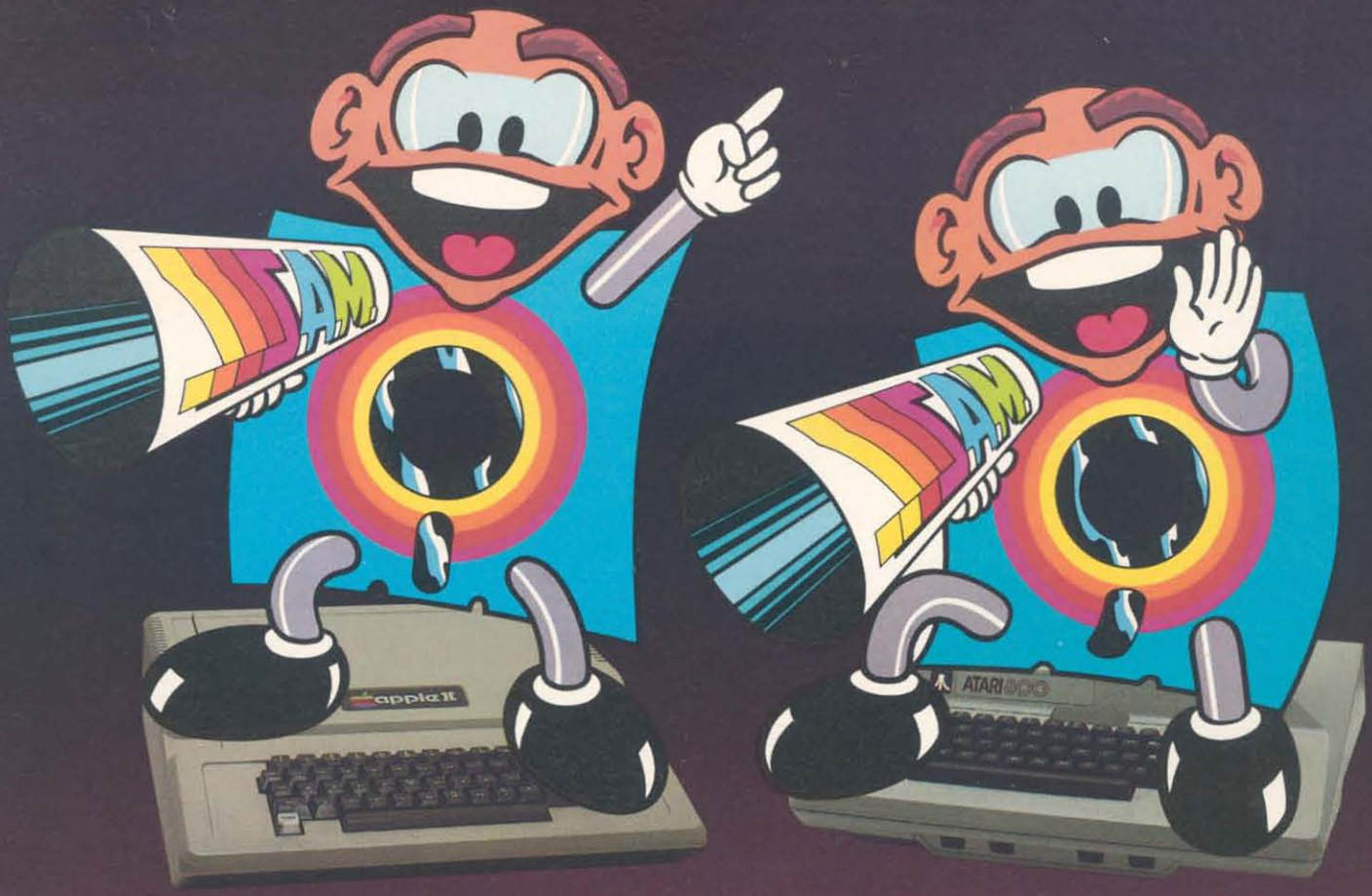
SoftSide™

#36
THREE DOLLARS



ALTERNATIVES TO BASIC — A Modern Tower Of Babel?

A SPEECH SYNTHESIZER ON A DISK!



That's why Apples and Ataris are saying: "Talk Is Cheap"

IT'S CALLED THE SOFTWARE AUTOMATIC MOUTH, S.A.M. FOR SHORT

It's a high quality speech synthesizer created entirely in software. You use it as a software utility, load it into RAM, and then use your machine as usual, except now you can make your programs talk. It generates the speech sounds on demand, so there is no limit to what it can say.

When you hear S.A.M., you'll probably agree that it sounds better than all the hardware speech synthesizers for Apple or Atari computers. And, it has a truly remarkable price.

YOU CONTROL INFLECTION, PITCH AND SPEED

With its user-variable inflection, S.A.M. can accent words on the right syllable and emphasize the important words in a sentence.

You can also make S.A.M.'s speech higher or lower, and faster or slower, over a wide range of settings.

USE EASY PHONETIC INPUT OR PLAIN ENGLISH TEXT

S.A.M. understands a simple phonetic spelling system, not a mysterious alpha-numeric code. S.A.M. helps you learn phonetic spelling by showing you your mistakes, and the owner's manual gets you started with an English-to-phonetics dictionary of 1500 words. So it's easy to make S.A.M. produce exactly the sounds you want.

But suppose you want to type ordinary English, or you want your machine to read a word processor file aloud. The S.A.M. disk comes with RECITER, an English text-to-speech conversion program that lets S.A.M. speak from plain English text.

ADD SPEECH TO YOUR PROGRAMS WITH EASE

In a BASIC program, you add speech with just a couple of commands. In a machine language program, it's just as easy. S.A.M. comes with four demonstration programs to show off its distinctive features and help to write your own talking programs. Write adventure games with talking characters, educational programs that explain aloud, or utilities with spoken prompts – put your imagination to work.

S.A.M.

S.A.M. programmed by Mark Barton.

You can order S.A.M. directly from DON'T ASK. Add $2.00 for shipping and handling to your check or money order (or order C.O.D.)

S.A.M. for the Apple II/II+

Includes an 8-bit digital-to-analog converter and audio amplifier on a board. Only **$124.95**

Requires 48K, disk. (S.A.M. takes up to 9K; RECITER 6K.) You will also need a speaker.

S.A.M. for the Atari 400/800

S.A.M. talks through your television speaker. No additional hardware is required. Only **$59.95**

Requires 32K, disk. (S.A.M. takes up 9K; RECITER 6K.) Note: to produce the highest quality speech, S.A.M. automatically blanks the screen during vocal output; the display is preserved. S.A.M. can talk with the screen on, but the speech quality is reduced.

DON'T ASK INC.
COMPUTER SOFTWARE

2265 Westwood Boulevard, Suite B-150
Los Angeles, California 90064
Telephone: (213) 397-8811

Hear S.A.M. at your favorite computer store today!
Dealer inquiries welcome.

valFORTH™ for Atari* 400/800

Professional Software for the Hobbyist

VALPAR INTERNATIONAL

FORTH has been used for years by ATARI* and others in programming their arcade games. FORTH is *fast*, 15-20 times faster than BASIC, and can make use of every capability for your computer. And it's no longer the province of the professional programmer! With valFORTH and the additional packages described below, you can create programs in an afternoon that would previously have taken weeks of hard work!

WHAT? YOU DON'T ALREADY KNOW FORTH?

Then take advantage of our special offer on *Starting Forth* by Leo Brodie. Widely acclaimed as the best book available on the subject, this entertaining treatment of Forth will make you comfortable with this exciting language quickly and easily. When you order *Starting Forth*, you get a free copy of our "Notes for the valFORTH User," including references, by page, to *Starting Forth*.

These are the utilities developed by Valpar International's software specialists and used to create our commercial software products.

Coming Attractions:
* TARGET COMPILER
* 3D WORLD

For more information *fast* and a 4TH ♥ IF HONK THEN bumpersticker, send 25¢ and a self-addressed, stamped envelope.

valDOS NEW! By Popular Demand!

valDOS. Now you can use your valFORTH system to read and write normal Atari DOS Files created with valDOS or other sources. Also includes valDOS File Editor for creating and reading FORTH Source Code without "screens." (Not a general file editor.)

(Over 25 pages of documentation. Requires valFORTH.)

valFORTH™

Package contains: fig-FORTH kernel with mathematical and stack operations machine-coded for higher speed than normal fig-FORTH; line editor AND screen editor, debugger, sound and graphics commands, floating point, advanced 6502 assembler, diskcopiers, and much more! (Over 110 pages of documentation)

GENERAL UTILITIES AND VIDEO EDITOR

Utilities: 4 array types, 4 case types, text on graphics 8, extensive string manipulation and keyboard input, STICK & PADDLE, randoms, bit manipulation, and much, much more.

Editor: *fast, powerful, complete* valFORTH screen editor 1.1. A professional-quality tool that makes editing a pleasure.

(Over 60 pages of documentation. Requires valFORTH.)

PLAYER-MISSILE GRAPHICS, CHARACTER EDITOR, & SOUND EDITOR

Player-Missile: Create, move, color, change images of and bound players and missiles with high level commands. Full support of 5th player, multicolor players, etc. etc. All critical sections in machine code.

Character Editor: Compose character sets with joystick. Simultaneous display of created characters. Make images for players and missiles.

Sound Editor: Simple independent control of all four voices (one joystick per voice) and audio-control register. Create any single-setting sound with graphical and tabular readout.

(Over 35 pages of documentation. Requires valFORTH.)

DISPLAY FORMATTER

Fast, simple creation of all types of display lists, with automatic 4K boundary jumping. Automatic or user controlled memory allocation, and formatting for horizontal and vertical scrolling, and display list interrupts.

(Over 30 pages of documentation. Requires valFORTH.)

TURTLE & valGRAPHICS, AND ADVANCED FLOATING POINT ROUTINES

All graphics modes supported — even GTIA and "7+". Draw and fill commands *faster* and *smarter* than Basic. "Turn-toward" for "chasing" and vanishing point effects; point labeling, etc. Also SIN, COS, ATN, ATN2, etc. added to floating point.

(Over 35 pages of documentation. Requires valFORTH.)

Text Compression and Auto Text Formatting

A unique, two-part utility!

- Text Compression allows the packing of text into much less space than normally required. Useful for wordy Adventure games, "artificial intelligence," etc.!
- Auto Text Formatting takes both normal and compressed text and routes it to the video screen "windows."

(Over 20 pages of documentation. Requires valFORTH.)

valFORTH alone requires 24K
valFORTH plus one or more packages requires 32K minimum
Memory requirements include 10-12K working space
All products are now on non-protected disks
Over 350 pages of detailed documentation!

valFORTH	$45.00
General Utilities and Video Editor	40.00
Player Missile Graphics, Character Editor, and Sound Editor	40.00
Display Formatter	35.00
Turtle and valGraphics and Floating Point Routines	45.00
Text Compression and Auto Text Formatting	35.00
valDOS and valDOS File Editor	45.00
Heavy-Duty Professional Binder (provided free when ordering 3 or more packages at one time)	13.00
Starting Forth	15.95

Plus Shipping and Handling
VISA and MASTERCARD accepted

VALPAR INTERNATIONAL
3801 E. 34TH STREET
TUCSON, ARIZONA 85713
Call Toll-Free 800-528-7070 In Arizona call (602) 790-7141

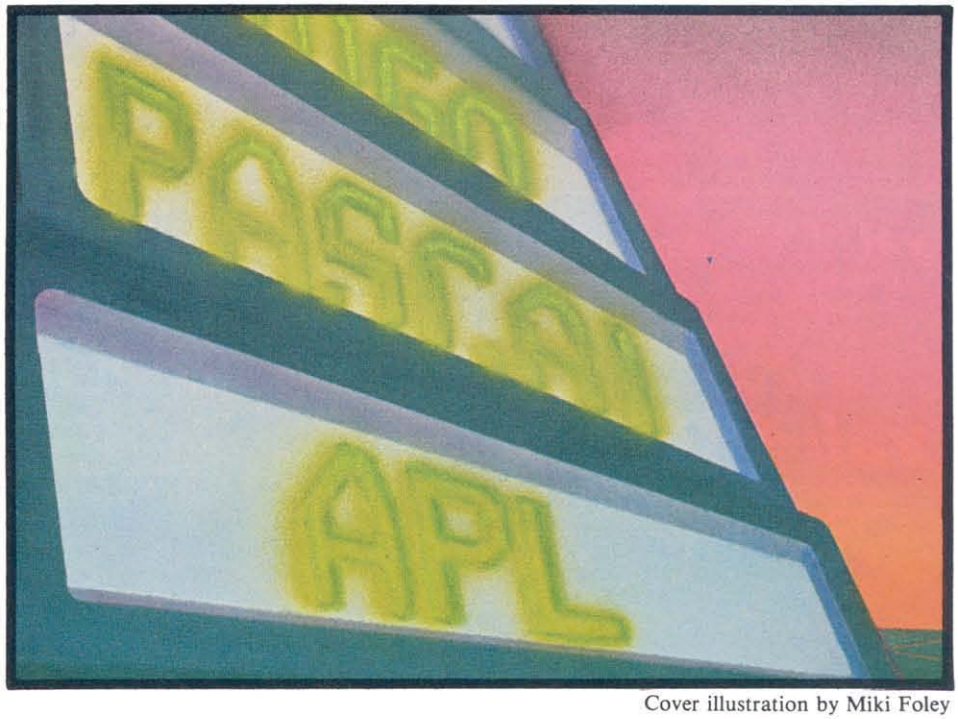
Principal Software Authors: Stephen Maguire and Evan Rosen

SoftSide™

SoftSide Vol. 6, No. 3

SUBSCRIPTION INQUIRIES should be sent to *SoftSide* Publications, 100 Pine Street, Holmes, PA 19043. EDITORIAL AND ADVERTISING CORRESPONDENCE should be sent to *SoftSide* Publications, 6 South Street, Milford, NH 03055. Telephone (603) 673-0585.

SoftSide (ISSN 0274-8630) is published monthly by *SoftSide* Publications, Inc., 6 South Street, Milford, NH 03055. Printed at Wellesley Press, Framingham, MA. Second class postage paid at Milford, NH, Avon, MA, and at additional mailing offices, and applied for at Framingham MA. Subscription rates: US and Canada, $30/12 issues. First Class US, APO, FPO, Mexico, $40/12 issues. Other foreign countries, $62/12 issues. Media subscription rates: US Magazine and Cassette, $75/12 issues. US Magazine and Disk $125/12 issues. APO, FPO, Canada and Mexico, add $20/12 issues. Other foreign add $50/12 issues. All remittances must be in U.S. funds. POSTMASTER: Please send form 3579 to *SoftSide* Publications, 100 Pine Street, Holmes, PA 19043.



Cover illustration by Miki Foley

COVER FEATURE

FEATURES

DEPARTMENTS

CONTENTS

PC/SIDE

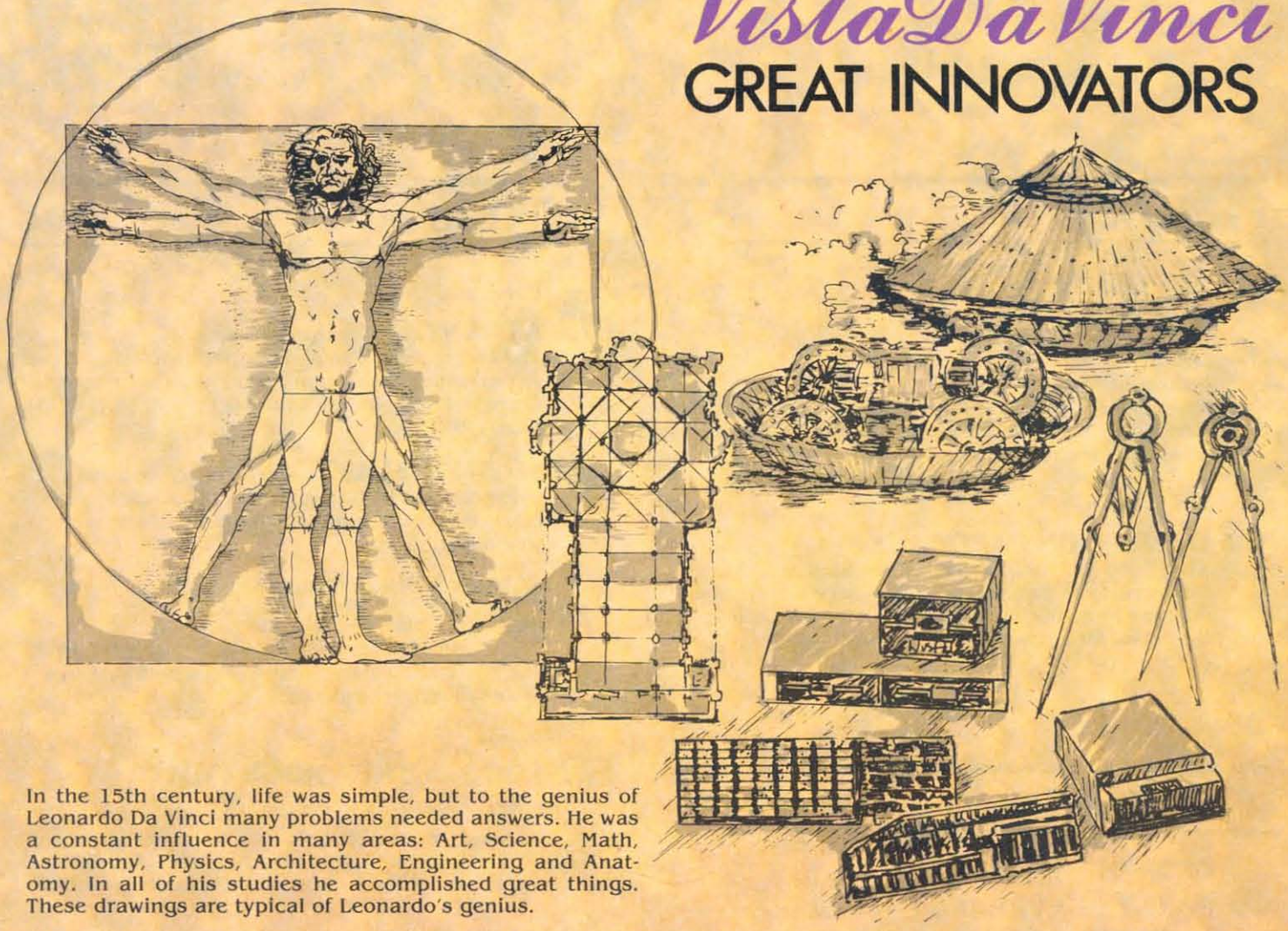
ATARI®/SIDE

APPLE™/SIDE

TRS-80®/SIDE

*Available on *SoftSide* Disk Version (See page 80 for ordering information)
**Available on *SoftSide* Cassette and Disk Versions (See page 80 for ordering information)

VistaDaVinci
GREAT INNOVATORS



In the 15th century, life was simple, but to the genius of Leonardo Da Vinci many problems needed answers. He was a constant influence in many areas: Art, Science, Math, Astronomy, Physics, Architecture, Engineering and Anatomy. In all of his studies he accomplished great things. These drawings are typical of Leonardo's genius.

Vista Computer strives to answer the complex problems of today's computer high technology. Vista excells in many areas also: Apple® Disk Storage, Apple® Drive Subsystems, Vision Series Video Cards, IBM™ Memories and IBM™ I/O.

Vista sees the continuing need for advancement in all these areas and are constantly expanding the product line to meet tomorrow's computer needs.

Vista Family

® Apple II is a registered trademark of Apple Computer Co.

Listed below are products currently manufactured by Vista: • V1000 Dual 8" Disk Drives • V1100 Dual Thinline 8" Disk Drives • V1200 6MB Cartridge Drive • Solo 5¼" 35 Track Disk Drive single sided full height • Duet 5¼" 40 Track Disk Drive double sided half-height • Quartet 5¼" Dual double sided Disk Drive in side-by-side chassis • A800 DMA Double Density Disk Controller for 8" Disk Drives • Apple III™ Time Card • Keyboard Buffer • GB75 Interface • Vision 80 80 x 24 Card • Vision 40 Softscreen Interface • Vision 20 Character Generator • IBM® 576K Maxicard • PC Master • PC Extender • PC Extender Plus • PC Expander • PC Expander Plus • IBM® Multicard.

Continue to watch Vista Computer Co., Inc., as their line expands to meet all your future computer needs.

Contact Your Local Vista Dealer or Call our Vista Hotlines.

Vista COMPUTER COMPANY, INC.

1317 East Edinger / Santa Ana, CA 92705
(714) 953-0523 / (800) 854-8017

DISTRIBUTORS / REPRESENTATIVES

Western - Group 3 Wholesale
(213) 973-7844 (408) 732-1307
South Central - M.P. Systems
(214) 385-8885
UK - Hal Computers Ltd.
(0252) 517175 / TWX: 858404

Northeast - Computers & Peripherals Inc.
(315) 476-6664
Florida - Audio Marketing (REP)
(305) 322-8327
Central - Wyatt & Associates
(317) 773-4791

The Information Utility

by Randal L. Kottwitz

"My phone's quit working."

"The water main broke last week and I'm still driving to my mother's to go to the bathroom."

"That electrical storm knocked out our power for three days!"

How many of the above statements have described your predicament at one time or another? Very few of us manage to make it through this life without having to face the lack of "vital" services at one time or another. Sometimes we are just as glad to leave them behind for "the peace of the country," but most of the time we consider them necessary to even the simplest of lifestyles. It's not easy to consider how we might change our lives should running water, electric power and/or the telephone be permanently removed from them.

A new utility is fast forming on which we may soon be equally dependent. At an ever increasing rate, the stream of information entering and leaving our homes is becoming an assumed luxury. We rely on the weather report to tell us how to dress, we telephone orders for merchandise to be delivered to our door, answering machines click on and off in our absence, informing inquirers of our whereabouts and collecting their messages for later digestion. Most cable companies offer immediate reports from one of the major news wire services and banking from home is an available reality in several major cities. No one has *all* of these inputs influencing their lives, but most of us have enough to keep us thoroughly occupied.

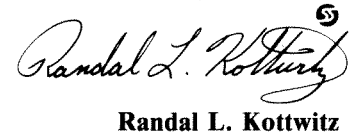
All of these streams of information are converging into a central technology I call "the information utility." You can answer your telephone through your television, access a wide realm of services through a modem and a computer, and interactive cable services are slowly but surely penetrating the nation. Our life revolves more and more around the screen of our television set. The ramifications of all of this are essentially very good — the more we communicate with each other, the better we will understand each other, and the closer we will come to world peace and understanding.

However, we must prepare to answer some serious questions as we become dependent on this new utility. Will communities control the information banks accessed by their citizens, or will it be the county, state or country? Censorship has reared its ugly head again recently as book burnings and pornography issues have garnered headlines. Will influential groups try to seize control of our new utility, telling us what we should or should not know? The issue of *right* to know versus *need* to know will become more serious as we are given the means to search out whatever information we might want.

What about the economic ramifications of our new utility? Will the line between the haves and have-nots become more evident as those who can afford it gain access to data which will make their decisions more profitable and beneficial? What will be the effects on our banking system? First impressions would have you believe that only the physical building of the bank will change as we operate from our home terminals. However, the changes in the way we receive credit and make purchases will reach deeply into the banking system as well. Suppose you received a line of credit based on the value of your home and car, against which you could charge purchases of goods and services — much the way you charge against your credit cards now. Credit cards have gotten many people in "over their heads" as they've learned that credit is not easy to manage. What if these same people could keep charging until they spent away their home and transportation?

As with any utility, the question of service will arise. If we become dependent on our home terminals to educate our children, conduct our personal financial affairs, and even to perform our duties to the workplace, what will be the effects of a failure in the supporting system? I recently had my phone go out of service for a full week. As a result, I made many trips to a local shopping center to use a pay phone to conduct necessary business. I have learned to depend on the telephone to conduct many areas of my life. As we learn to depend on our "information utility" to provide vital support services, what will be our backup system should the utility fail to operate?

I've asked many questions here and not given any answers — intentionally. These issues and many more like them have no definitive answers at this time. However, they are serious topics which we must address as we face the near inevitability of the "information utility." It's easy to paint a rosy picture of the technological society of the future. However, we must recognize that every change we make in our lives affects even more changes in the future. We, as a society, have had a tendency to implement technological advancement without always considering the far reaching ramifications. We've learned some important lessons from our mistakes. Now, as we look forward to the information-rich culture of the future, let's consider those lessons of the past and do our best to speculate and plan for the long term effects of the changes we must make.



Randal L. Kottwitz
Editor-in-Chief

INPUT/OUTPUT

From our readers

INPUT

ZBASIC 2.0

Dear *SoftSide,*

I am very interested in obtaining a copy of the ZBASIC 2.0 compiler which was reviewed in Issue 31. I have not been able to locate a local dealer who is able or willing to supply this item. Will you please publish the mailing address for Simutek, Inc.?

Darrell Rose
Pacific Palisades, CA

Editor's Reply: Our regrets for not publishing the address with the review of ZBASIC 2.0. For your information:

Simutek Computer Products, Inc.
4877 East Speedway
Tuscon, AZ 85712

ATARI® DISKS AND ENVYRN™

Dear *SoftSide,*

I have been a faithful reader of your magazine for over two years. You will be happy to know that I am using your Database, with some minor modifications, published in the December, 1981 issue, to help run a Union representing 1500 people. I have found your Atari® programs to be of excellent quality.

Now come the complaints. First, the matter of tape vs. disk. There are far more cassette users than disk users, although that may not always be the case. I am disappointed when programs that *require* a disk drive are placed in the regular ATARI/SIDE. I feel that if a disk is required it is a program more suitable for the DV version than ATARI/SIDE.

The second problem was the announcement that *Envyrn*™ would be published in Atari Microsoft BASIC rather than Atari 8K BASIC. Even if it significantly delays publication, I beg you to reconsider your decision. On checking informally with members of our users group in the Cleveland area, I found only two who had Atari Microsoft BASIC, and only three who were considering buying it. Several were considering BASIC A+, which, as you know, is upward compatible with Atari 8K BASIC. I talked with about 35 people and believe my findings are significant. The *Envyrn* program is, I believe, the most important you have ever published. The thought of it being available to only a handful is very distressing. To be perfectly honest, the thought of typing it in, translating it to 8K BASIC, and having no way of checking it against a listing or SWATting it is sufficiently intimidating that I probably will not attempt it. Atari Microsoft BASIC is a very major product, but not in wide enough use to warrant its use in your publication. Please reconsider and publish *Envyrn* in 8K BASIC, or at least provide a companion translation and *SWAT* table.

Even if you don't follow my suggestions, I'll keep plunking down my three dollars.

Ronald M. Hopkins-Lutz
Cleveland Heights, OH

Editor's Reply: In our recent survey of *SoftSide* readers, we discovered that more than 50% of Atari owners surveyed owned a disk based system. Obviously, not all of those people subscribe to *DV*. Therefore, we try to offer programs for both disk and tape system owners in every issue of *SoftSide*. As for the problem with Microsoft BASIC, the facts are not yet in. We have had a large response to our request for owners of Microsoft BASIC to inform us of their purchase. However, it has not been substantial enough to justify changing our standard throughout the magazine. We are currently investigating publishing *Envyrn* in Atari 8K BASIC and are encouraged at the possibilities. We hope to be able to fulfill your request.

DISK QUALITY

Dear *SoftSide,*

In Issue 33, you mentioned that you had discovered that some of the disks you were using had limited powers of data retention. As a mass user of disks, it would be a real service to your readers if you were to share your experiences with various brands with us.

Of course, names are essential, and perhaps some toes will be trod on. However, this would be golden information for someone who uses only a couple of dozen new disks per year. With this information, we could evaluate what we are getting for our money and make some educated comparisons. Maybe some trade-offs are acceptable, but it would be nice to know what they are.

Ken Green
Fort Lauderdale, FL

Editor's Reply: Consistency in disk quality is a problem plaguing the computer industry. Over the years, several companies have gained recognition for quality disks, only to have a bad batch reach the marketplace and their reputation come into question. Due to this inconsistency, the best thing we can recommend is that you always purchase disks with a warranty, *always* keep backups, and don't hesitate to return disks to the manufacturer if you have a failure. The company which duplicates *SoftSide DV* is constantly running tests on media from different manufacturers in order to maintain quality control. However, the results on any one manufacturer vary from batch to batch and an attempt to give you a comprehensive analysis of disk quality could only hold true for the disks we tested — not for any future releases from the same manufacturers.

"A CAUSE WITHOUT A REBEL"

Dear *SoftSide,*

In your editorial, "A Cause Without a Rebel," you state that it's time to form an organization as extensive as the PBS to steer the production and application of software for national education. I would like to join you in your call for action. I plan to meet with a vice-president of PBS in Stowe in the near future, and we will certainly discuss this need. In addition, my work with the State Department of Education in Vermont, both as a resource agent

continued on page 8

Integrated, Menu-Driven File And Information System for Home or Office*...with PCHMS™, our Personal Computer Home/Office Management System.

Instantly put your IBM PC to work! PCHMS is a filing and information system that manages all those details that make your office, home and personal life smooth and efficient. Organize personal, household and business details—names and addresses, phone numbers, credit cards, home inventory, zip codes, medical and dental records, insurance policies, recipes, expense accounts, shopping and other lists, area codes, maintenance records and more.

*Note: Over half of our users have PCHMS™ in the office.

PCHMS is an integrated menu-driven system that allows you to add, delete, modify, or print out records in any file by full or partial key search. And you can create 10 full-feature user-defined files for anything you can think of. PCHMS provides all this plus a letter/memo-writer, built in printing calculator, 20 year calendar, constant display of date and time, alarm timer, worldwide time conversion, and metric/English converter. PCHMS files may be individually password protected. PCHMS runs in both monochrome and color.

And . . . that's not all . . .

You can add the PCHMS Auto-Dialer™. Automatically dial any stored phone number in your file with the press of a single key. Auto-Dialer gets you local, long distance, and international phone numbers, as well as networks such as Sprint and MCI.

Want more? Add the PCHMS Mailing List Module, the PCHMS Electronic Mail System, and the PCHMS Budgeting and Home Finance System.

PCHMS™ runs on an IBM Personal Computer with 64K of memory, at least one disk drive (single or double sided). Printer optional. PCHMS Auto-Dialer requires a Hayes Stack Smartmodem and RS232 card.

Ask for PCHMS™ and PCHMS Auto-Dialer™ at your IBM PC dealer or order directly from Arlington Software + Systems. NOW AVAILABLE AT COMPUTERLAND STORES.

Dealer inquires invited.

PCHMS software is supplied on single sided diskettes together with comprehensive user documentation, backup utility and configuration program. Satisfaction Guaranteed!

☐ please send me _____ PCHMS @ $89.95 _____

☐ please send me _____ PCHMS Auto-Dial @ $69.95 _____

☐ please send me _____ PCHMS Mailing List @ $59.95 _____

Shipping and handling $ 3.50 _____

MasterCard and Visa welcome, please add 4% _____

Name: _____

Address: _____

City: _____ State: _____ Zip: _____

Phone: (_____)_____ Card Exp. Date: _____

Charge Card # _____

Signature: _____

97 Bartlett Ave. Arlington, MA 02174
(617) 641-0290

ARLINGTON Software + Systems

continued from page 6

and as a founding director of the state's educational computer users group, gives me a chance to share ideas with a great number of people.

The great challenge for education (when and if the educational establishment takes on the task of retooling itself for the age of information) will be not only to provide needed services and training, but also to provide the leadership in setting the tone and philosophy of the new age. Learning is being redefined in terms of the process of the creative imagination, instead of the traditional notion of the acquisition of skills and information. In the future learning society, the acquisition of knowledge will be secondary to the communication of ideas.

Our focus should therefore be on developing self-directed learners who are at home with creative initiation of quests for knowledge. The new tools of education can extend our capacity for learning as much as the invention of writing did for our ancestors, or they can be used to deliver education as usual. Much depends on our diligence in guiding the growth of the integration of the new technology into the educational system.

I'll look forward to helping you, in whatever way I can, to influence the development of the future of advanced technologies in the nation's education.

David Gibson
Stowe, VT

Editor's Reply: Thanks for your support. The issue of computer awareness in education holds an ongoing interest for *SoftSide*. Watch future issues for reports on developments in this area.

MORE GAMES

Dear ***SoftSide,***

After reading ***SoftSide*** 33, I had only one thing to say: Aaaaaaaaahhhhhhhhh!!!! !!!! Only ONE game for the Apple™, only one for Atari® (DV!!), and none for the TRS-80®!!!! Please, don't let us down. Please publish more games. My fingers need practice, and my brain-juice doesn't run anymore!

Keep up the great work, and thank you for reading.

Leonard Vincent
West Hollywood, CA

Editor's Reply: *SoftSide* continues its commitment to entertainment software for microcomputers. However, we feel that the home computerist's interests reach far beyond games and we are shaping our editorial content to serve as many of those interests as possible.

OUTPUT

by Randal L. Kottwitz

I hope you have a comfortable chair and warm fire to accompany your reading of this issue of ***SoftSide.*** (Sorry, all you people in arid climates — it's January, and in New England that means *cold.*) We've packed our pages with a lot of heavy reading to pass those winter days. Curl up and enjoy.

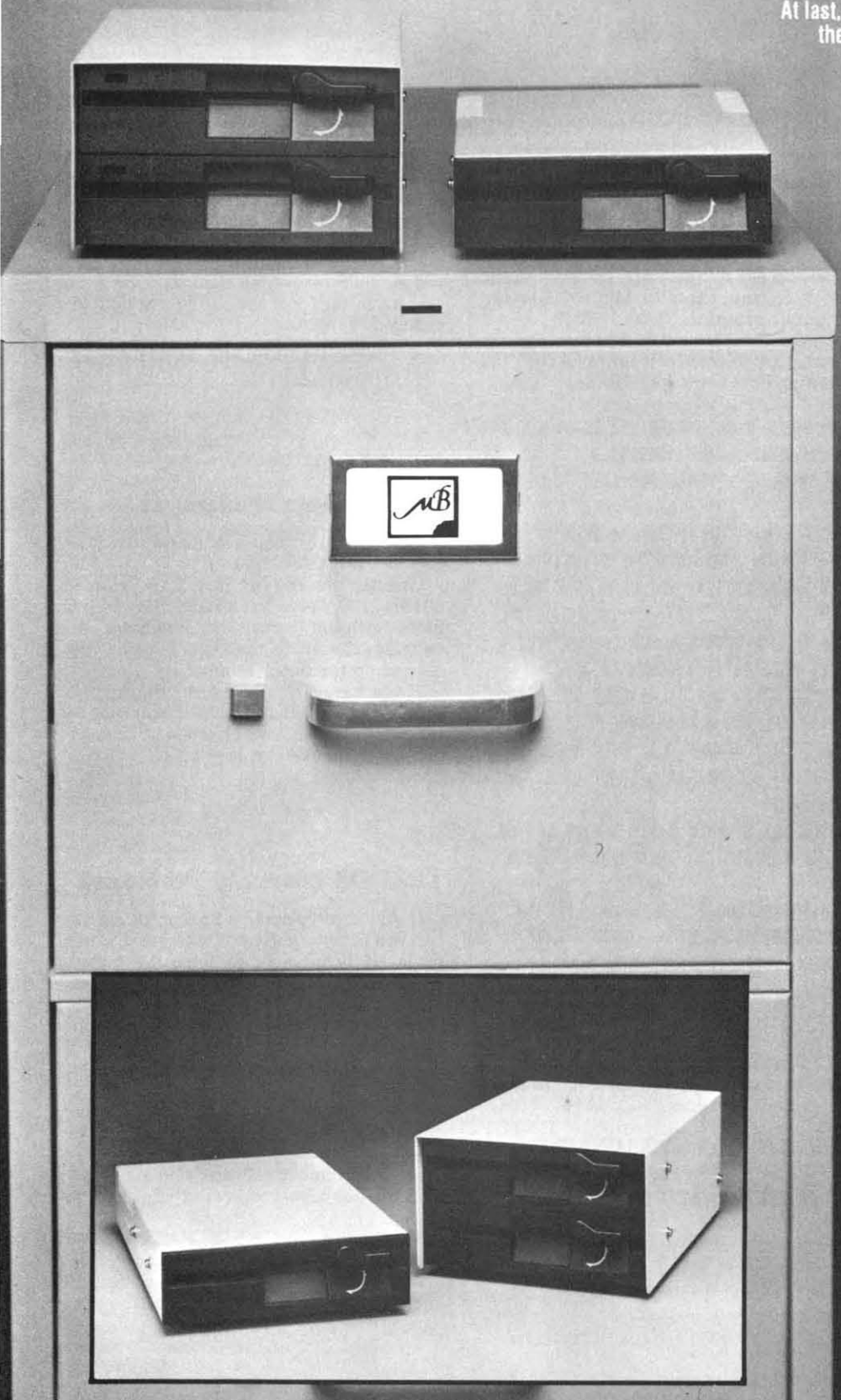
It's a new year, and we'd like to suggest a New Year's resolution. Please resolve that the next time you think of something you'd like to say to ***SoftSide*** or its readers, you'll go to your word processor and write us a letter. Some of the best things published in ***SoftSide*** have originated as suggestions from our readers. In a few cases, the person who wrote the letter actually got involved in carrying out the idea. But, most times he's provided the kernel of a much larger project we've been able to carry out through our network of programmers and authors.

We'd like to offer a word of caution about programs published in ***SoftSide***. We've received several disturbing phone calls indicating that some of the programs we publish are being made available by users groups and bulletin boards. In some cases, members of these organizations have called, requesting permission to distribute our software in this manner. Our answer is always the same — *all* of the material published in ***SoftSide*** is copyrighted, by ***SoftSide*** and its authors. We are able to provide the software very inexpensively through our monthly distribution process, but it does not become public domain when we put it in print, just as the material in a book doesn't become public domain when it's published. In the same manner, the lack of copy protection on the vast majority of our tapes and disks does not imply that we wish to give away the software they contain. We believe that, whenever possible, the user should be able to make backup copies to insure long life for our programs. Please resist the urge to give our software away. It is the means by which we and the authors who write for us make our livings.

So much for messages from the inner chambers of ***SoftSide***. We wish you the happiest of New Years and, as usual, Happy Hacking!

Randal L. Kottwitz

Space Saving Storage



At last, Microbyte has created the perfect storage unit for the Apple II. This new data drive slimline out performs any of its competition. This pint-size drive, works 8 times faster than most other drives, saving you space and time. Compare the features of the new ASAP slimline to what you're using now. We're sure you'll agree, this slimline out performs the others.

- 8 times faster than APPLE II Drives.
- Direct drive motor of extremely high quality.
- Only ½ the size of a regular drive.
- 100% APPLE[1] compatible (including "half track").
- Can be used with IBM PC[2] and other computers by simply disconnecting Interface Board.
- Mechanism and read - write electronics made by TEAC the world leader in Audio recording machines.
- 1 (one) full year warranty at no extra cost!!!!!
- The SLIMLINE is a 35/40 track drive and can take advantage of our ENHANCER Diskette which will give the user 15% more storage capacity & up to 163K Bytes.
- Slimline ½ height 163K.
- Dual Slimline drives ½ height in one box.
- Slimline ½ height (1) Megabyte with controller.

Contact your local dealer for more information.
Dealer inquiries invited.

MICROBYTE

1198 E. Willow, Signal Hill, CA 90806

Call Toll Free (800) 421-7701
In California (213) 595-6431
or (714) 891-2663

[1]APPLE is TM of APPLE Corporation. [2]IBM is TM of International Business Machines.

HINTS & ENHANCEMENTS

From our readers

Atari® *Solitaire* Enhancement

I would like to thank Mr. David Pleacher for his enhancements to Apple™ *Solitaire* (original program, May, 1982; Apple enhancements, Issue 33). I had to make a few changes to convert it for my Atari. The following lines work well for me.

```
1420 GOSUB 1600:? "ARE YOU READY FOR M
Y COMMENTS? (Y/N)";:GET #1,A
1430 IF A<>89 THEN GOSUB 1460:GOTO 133
0
1433 FOR Q=0 TO 6:Q7=Q7+IN(Q):NEXT Q
1434 FOR Q=1 TO 4:Q7=Q7+F(Q):NEXT Q
1435 GOSUB 1600:? "YOU WERE ABLE TO PL
AY ";Q7;" CARDS.":? "  ";:GOSUB 1615
1436 IF Q7>45 THEN GOSUB 1600:? "YOU A
LMOST MADE IT!!!";:GOSUB 1615
1437 IF Q7>30 AND Q7<46 THEN GOSUB 160
0:? "EXCELLENT WORK!";:GOSUB 1615
1438 IF Q7>20 AND Q7<31 THEN GOSUB 160
0:? "THAT IS ONLY AN AVERAGE GAME.";:G
OSUB 1615
1439 IF Q7<21 THEN GOSUB 1600:? "DO YO
U KNOW HOW TO PLAY THIS GAME?";:GOSUB
1615
1440 GOSUB 1600:? "WOULD YOU LIKE TO P
LAY AGAIN? (Y/N)";
1448 GET #1,A:IF A<>78 THEN RUN
1615 FOR PAUSE=1 TO 400:NEXT PAUSE:RET
URN
```

If you would like the computer to tell you when there are only a few cards left to turn, type in these two lines.

```
125 IF IN=49 THEN GOSUB 1475:RETURN
1475 GOSUB 1600:? "YOU HAVE 3 CARDS LE
FT IN THE DECK";:GOTO 1610
```

Henry L. Smith
Newburgh, NY

Atari *Microtext* Enhancement

After reading Randy Rogel's letter (Apple *Microtext* In-line Editing) in Hints and Enhancements, Issue 33, I felt that the same thing was needed for the Atari version. The following line will enable you to select the line to be edited, as before. But now you will be able to use the right arrow key to skip over the text that is to remain unchanged.

```
510 IF C=42 THEN C$=T$(LP(EL-1)+CHAR,LP
(EL-1)+CHAR):GOTO 740
```

Tron Black
Salt Lake City, UT

Disk Snooper changes

Here are two suggested changes to *Disk Snooper* (*SoftSide*, Issue 32).

Change the end of line 2200 to read THEN 2190, instead of THEN 2150. This is more efficient, since it eliminates re-entering the track number if you only messed up the sector number.

If you have 40-track disk drives, the program will read all 40 tracks if you modify the parameters in lines 2160 and 2170 to accept track numbers as high as 39.

Nick Kontis
Valencia, CA

TRSDOS Password Problems?

If you need to make a backup of a disk, and you have forgotten the password, don't panic. It is possible to alter the master password on a disk even if you have forgotten it.

1). Take a disk with a known password and boot it.

2). From DOS, type PROT (PW) and hit ENTER.

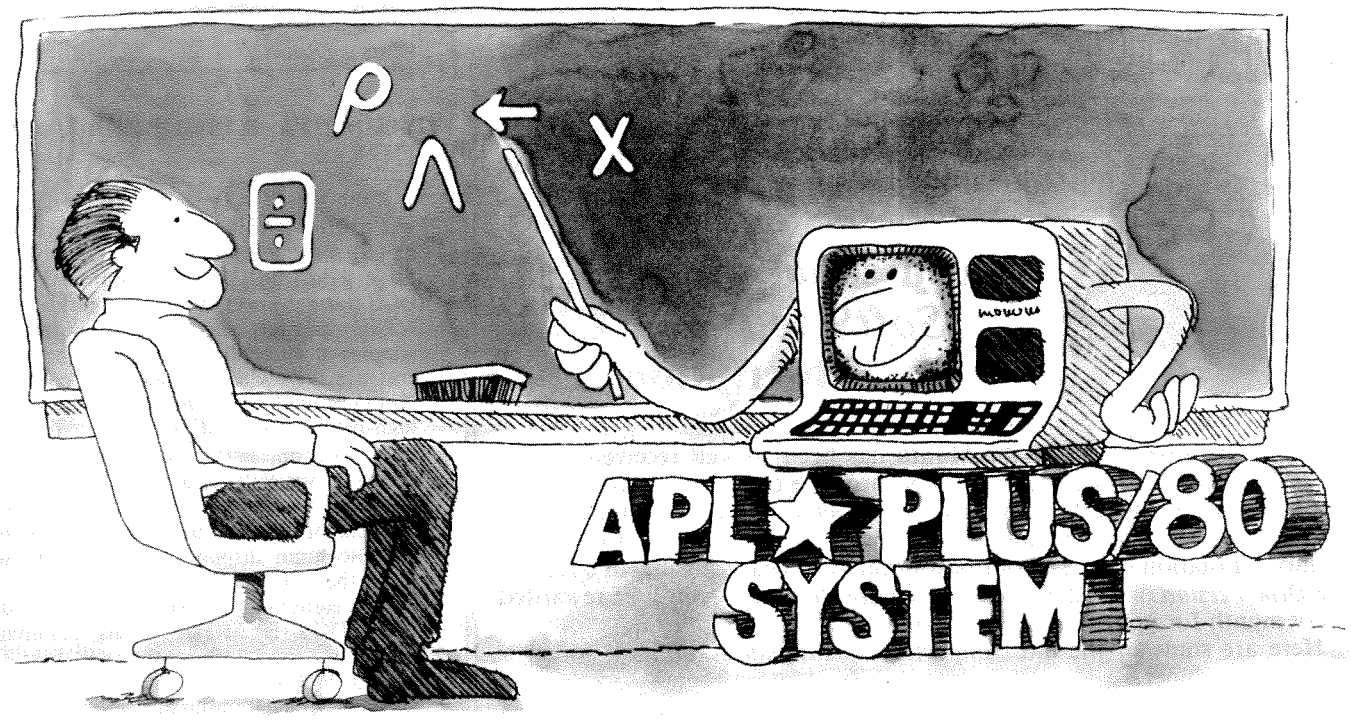
3). Answer the MASTER PASSWORD? query and wait for the disk to stop spinning.

4). Now remove the disk and replace it with the one whose password you want to change.

5). Answer the NEW PASSWORD? query with the password you want and hit ENTER. The new password will be written to the disk. That's all there is to it!

Joseph Goudreau
Malone, NY

continued on page 12



How much do you know about APL?

Q. APL uses "funny symbols." ☐ **TRUE** ☐ **FALSE**
A. TRUE.
Some of the symbols in APL *are* unfamiliar. But many of them are so familiar that you've been using them since grade school. Symbols such as + − × ÷ < > and = . Others, like ; and / , have new uses in APL, but you're familiar with them as symbols. These symbols *and* the "funny" symbols (for example, ρ ι \ and ∈) make APL very concise and, therefore, very productive. One APL symbol often does as much as an entire statement in BASIC.

Our APL★PLUS®/80 System, developed especially for your TRS-80® Model III, offers you the choice between using the traditional APL symbols *or* our English-like keywords. But even if you start out using the mnemonic keywords rather than the "funny" symbols, you'll soon want to make the transition to the APL symbols because they save you time, space, and effort.

Q. APL is hard to learn and to use. ☐ **TRUE** ☐ **FALSE**
A. FALSE.
Like everything else, learning APL takes some concentration, but most users find APL so appropriate to their projects that they can develop their solutions while learning the system. Our simple tutorial, *APL Is Easy!*, leads off the complete documentation package we provide with the APL★PLUS/80—everything you need for building applications whether you're a beginning user or an experienced APL programmer.

APL is easy to use too. You can write useful applications in your first learning session, and soon you can develop and maintain programs in one-fourth to one-tenth the time it would take you in BASIC. APL handles many "housekeeping chores" that other languages saddle you with (from dimensioning arrays to loop control).

APL★PLUS/80—A Complete Application Development System
Would you like to develop and deliver solutions faster and better for yourself or for your clients? With STSC's APL★PLUS/80 you can. Our APL★PLUS system features increase the power and productivity of the APL language, and the APL★PLUS/80 brings these proprietary enhancements to your TRS-80 Model III:

- complete APL language and system features, upwards compatible with our mainframe systems
- powerful output formatter
- array-oriented file system
- access to regular TRSDOS® or LDOS® files and subroutines
- communications features to allow use of TRS-80 as a simple terminal or under APL program control
- traditional APL symbols *or* mnemonic keywords
- utility program libraries
- the most complete set of documentation on the market.

Try the APL★PLUS/80 for yourself. Mail in the coupon below with your payment and we'll send you the APL★PLUS/80 Application Development System—all you need to run APL on your TRS-80 Model III. If you'd like more details, check the box on the coupon and we'll send you our free information package.

We're STSC, Inc., the leading supplier of APL software and services in the United States. Our APL★PLUS systems have been serving the professional and business world for more than 12 years.

APL★PLUS/80 runs under TRSDOS 1.3 or LDOS 5.1 on a 48K RAM TRS-80 Model III with two disk drives. The APL★PLUS/80 comes with a custom APL-character ROM and a self-adhesive keyboard label set to convert your TRS-80 to include the APL character set.

stsc

Attn: APL★PLUS/80 Distribution
STSC, Inc., 2115 East Jefferson Street
Rockville, Maryland 20852 (301) 984-5000 (orders only)

Yes, send me the APL★PLUS/80 System ($295* in U.S. and Canada).
☐ My check is enclosed. (Postpaid in continental U.S.)
☐ Charge my MasterCard Account # _____
MasterCard Bank # _____
☐ Charge my VISA Account # _____
MasterCard/VISA expiration date _____
Credit card customers add $4.00 postage and handling in continental U.S.
*Add applicable state and local sales taxes in CA, CO, CT, IL, MA, MD, MI, NC, NM, NY, PA, TX, WA.

I'd like to know more about the APL★PLUS/80 System.
☐ Send me your free information package.

Name _____
Address _____
City _____ State _____ Zip _____
Phone (___) _____

SS-183

Translation of the Month Contest

You could win a FREE *SoftSide* DV or CV Subscription!

SoftSide's* Translation of the Month** has been so well received by our readers, we're offering a greater author incentive than ever before. No, we can't give you a job at the U.N., but we will award a one-year subscription to ***SoftSide DV or an 18-month subscription to ***SoftSide CV*** for a high-quality translation of one of our past programs. That's a value of $125 for the ***Disk Version*** or $112.50 for the ***Cassette Version*** — you'll be rewarded every month for your translation efforts!

Here are some of the most important qualifications we look for in a translation winner.

Your entry must be a translation of one of the featured programs from a past issue of ***SoftSide*** (We're particularly interested in Apple™, ATARI®, and IBM® PC translations of some of our older TRS-80® only issues. Write for a list of suggested candidates.) In general, we're looking for translations of programs which are a CHALLENGE to translate. Some of the programs we publish are written in more or less "generic" BASIC, which can be typed into another computer with very few changes. Although these programs require the least effort to translate, they are also the least likely candidates for contest winners.

Your translation should be thoroughly tested and completely bug-free. Just converting program lines doesn't automatically ensure a workable translation. Be sure to use-test your translation as carefully as you would test a program you had written entirely from scratch.

Your translation should fully utilize the unique features of the computer for which it is written. The objective of a translation is to "fit" the capability and convention of its host computer, not simply mechanically duplicate the operation of the original program. This is especially true of programs which use graphics, and should be kept in mind for such minor features as keyboard layout (use of such special keys as arrows, ESC, CTRL, CLEAR, etc.). Also be careful with screen formatting; a word that spills over into the next line because of a PRINT statement that wasn't properly rewritten betrays such carelessness that we'll probably reject your translation automatically.

Your entry should incorporate any improvements and enhancements you can add to the original program. Don't feel that you have to limit yourself to the boundaries of the original. (On the other hand, don't go overboard and destroy the character of the original by completely rewriting it!) An enhanced translation is much more likely to catch our attention than a line-for-line duplicate, and it will have more value to our readers.

It's not necessary to include extensive documentation with your translation, only that which is different from the original. If most of the originally published documentation applies to your translation, simply say so. You should, however, include descriptions and explanations of any changes or enhancements you've made.

All **Translation Contest** entries must be submitted on disk, with documentation in printed or typed form. Disks will be returned only if accompanied by a self-addressed, stamped envelope. Send your entries to:

SoftSide™
Translation Contest
6 South Street, Milford, NH 03055

Hints ***continued***

Setting PC screen width from DOS

Although it's simple to set the screen width from BASIC with the WIDTH command, some users may not know you can do it from the PC DOS command level. DOS normally sets the screen width for the NTSC portion of the color/graphics adapter at 40 columns. NTSC is the signal you use if you have a standard color (or monochrome) monitor, as opposed to a high-resolution or RGB monitor. This can be a problem, particularly when, as happened at ***SoftSide***, you want to use WordStar® on a standard monitor. WordStar does not set the screen width, but *assumes* an 80-column display. It cannot be used when the screen width is 40.

The command to set the screen width from DOS is MODE. This command resides in a special file on the DOS disk; it is named MODE.COM. Any disk that will use the MODE command *must* bear this file. If necessary, use the COPY command to put MODE.COM on your disk.

The form of the command is very simple. Just type "MODE 80" or "MODE 40", and hit the Return/Enter key. If you want to make a disk boot up with the screen in a particular mode, do the following at the DOS prompt, A :

```
EDLIN AUTOEXEC.BAT
I
MODE 80
DATE
TIME
Strike Ctrl-Break. (Don't follow this with
Return/Enter.)
E
```

Remember to hit the Return/Enter key after each command.

This will install a batch file on your disk. The special name "AUTOEXEC" tells DOS to execute the commands it contains on boot-up. If you want 40-column mode, replace the "80" above with "40".

Fred Condo
Milford, NH

Bugs, Worms, and Other Undesirables

Atari *Hopper*

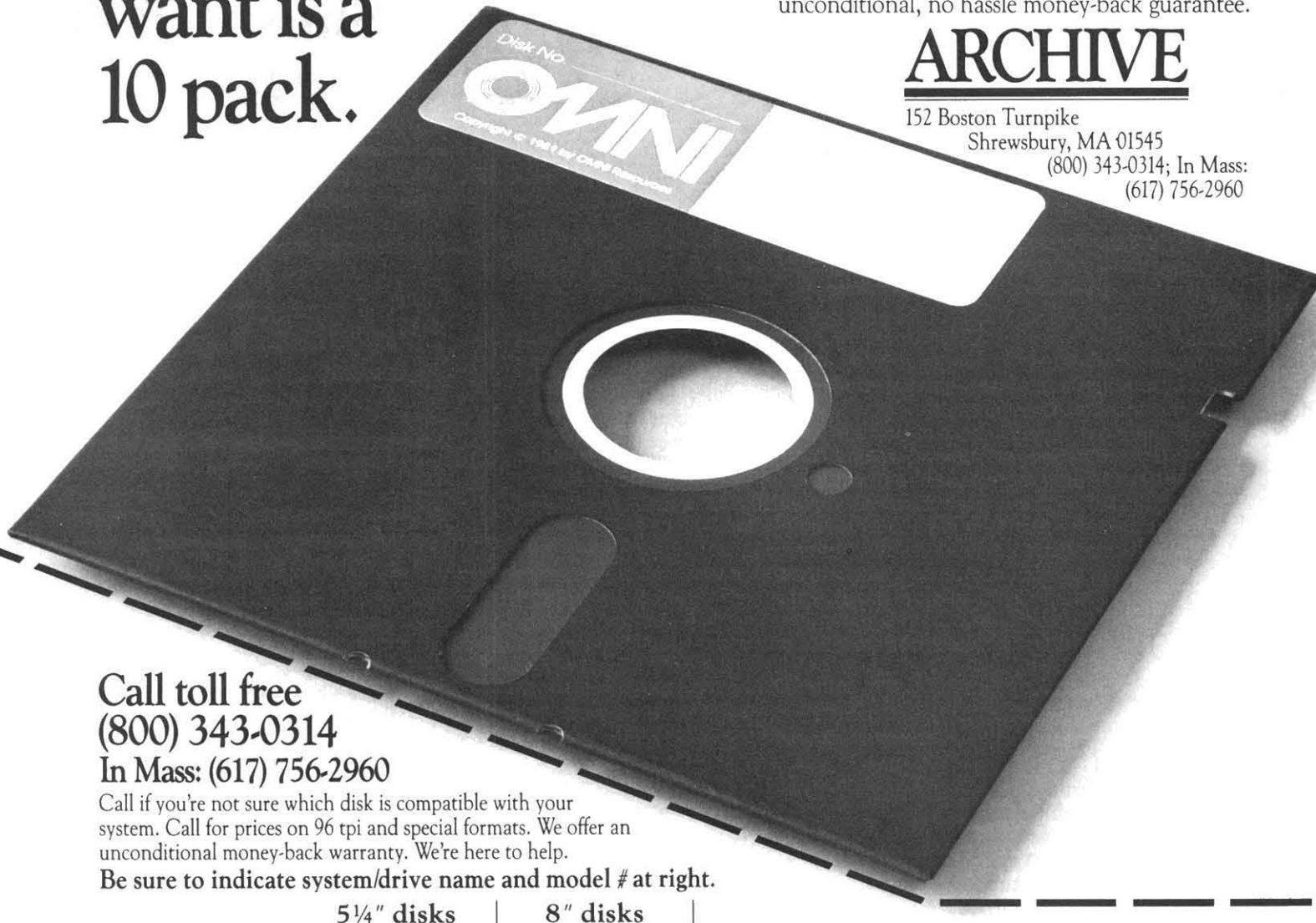
In the Atari version of *Hopper* (issue 35), a slight problem with the end-game screen display may be corrected by changing the keyword POSITION in line 2200 to PLOT.

Get Omni quality for as little as $1.99... even if all you want is a 10 pack.

Call toll-free for great savings on Omni's complete line of 5¼" and 8" premium disks. Each is certified error-free at a minimum of twice the error threshold of your system. Each is rated for more than 12 million passes without disk-related errors or significant wear. And each is precision fabricated to exceed all ANSI specifications with such standard features as reinforced hub rings and Tyvec sleeves. Get same day shipment and an unconditional, no hassle money-back guarantee.

ARCHIVE

152 Boston Turnpike
Shrewsbury, MA 01545
(800) 343-0314; In Mass:
(617) 756-2960



Call toll free
(800) 343-0314
In Mass: (617) 756-2960

Call if you're not sure which disk is compatible with your system. Call for prices on 96 tpi and special formats. We offer an unconditional money-back warranty. We're here to help.

Be sure to indicate system/drive name and model # at right.

	5¼" disks		8" disks		
	Cost per 10 pack	Quantity	Cost per 10 pack	Quantity	Total Cost
Single side/single density	$19.90	_____	$24.90	_____	$_____
Single side/double density	$23.90	_____	$31.90	_____	$_____
Double side/single density	———	_____	$34.90	_____	$_____
Double side/double density	$37.50	_____	$37.50	_____	$_____
Flip/Floppy reversible	$39.90	_____	$39.90	_____	$_____
Plastic library case (in lieu of soft storage box)	$ 2.99	_____	$ 3.49	_____	$_____
Shipping and handling ($2.00 first 10 pack, 40¢ additional 10 packs. Continental U.S. only.)					$_____
5% sales tax (Mass only)					$_____
				Total	$_____

☐ Check ☐ Master Card
☐ COD ☐ VISA

Card # _____ Exp._____

System/drive model # _____

Name _____

Address _____

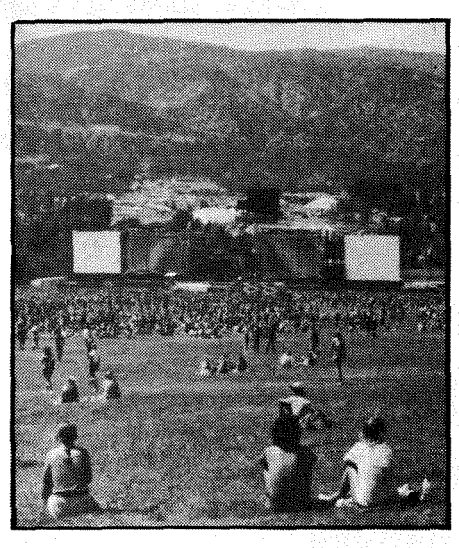
_____

Tel. _____

THE US FESTIVAL

Rock Show OR Technology Fair?

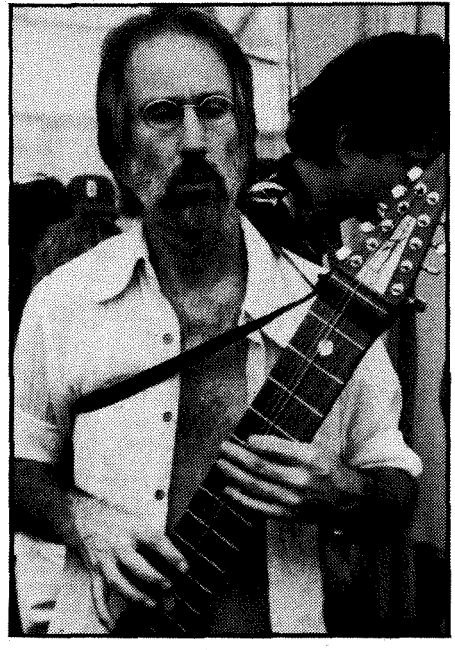
by Virginia Lyons



An estimated 250,000 people gathered in the desert outside San Bernadino, California, on September 3-5, 1982 for the *US Festival.* Bankrolled by computer wiz and co-Apple™ founder, Steve Wozniak, to the tune of $12.5 million, the *US Festival* was billed as the end of the *Me* Generation and the beginning of the *Us* Generation.

The Festival was to have been a blend of a technology fair and top name rock and roll bands in the world's largest outdoor amphitheater. The calibre of bands and quality of musicianship created one of the best rock performances in years. The technology fair, however, was disappointing to both visitors and exhibitors. In a September 5 news conference, Steve Wozniak admitted that the technology fair had been gradually relegated to a much less important place relative to festival site construction, organization and safety precautions. In these aspects, the *Us Festival* was outstandingly successful. The technology fair, however, has much room for improvement.

All technology exhibitors understood there were to be no actual sales. Rather, they were there to expose an unfamiliar public to computer technology. There was to have been an amateur "Home Brew" category of computer products, applications, and contests. Unfortunately, the Home Brew division never happened. UNUSON Corporation, the festival creators, got too bogged down in construction problems and the rock and roll show production.



Personal Computers

Despite the sparcity of exhibitors, 110 degree heat, choking dust and the Home Brew cancellation, there were some exciting exhibits. Hats off to Apple Computer for the most interesting and appropriate personal computer booth. One Apple staffer admitted that there had been some controversy regarding whether to show or not. After all, this was a rock and roll festival — not really Apple's business — and there would be no sales. About twenty Apple employees wanted to attend so badly, however, that they decided to set up a display.

Attractively featured at the Apple booth were educational, business, engineering, scientific, and graphic applications. On the last day, one of the misplaced Home Brewers even attached his Polaroid camera adapter to one of Apple's computers. This particular application was quite popular with the crowds, who waited patiently

for a snapshot of a portrait on the monitor. Finally, an instant way to prove those high game scores! Apple even launched their enormous hot air balloon, which decorated the night sky behind the sound stage.

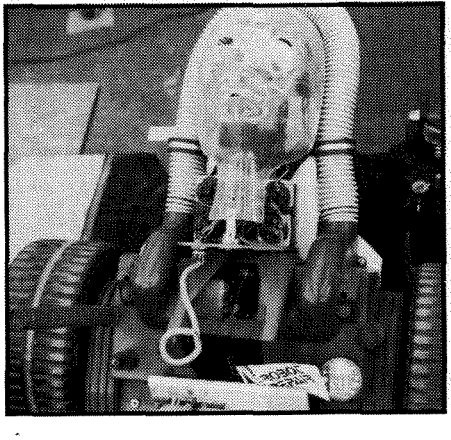
Atari® displayed about 30 arcade games in a 50 foot exhibit — a very popular display with kids and adults alike. It was a great opportunity for the crowds to beat the heat in the air conditioned tents. Unfortunately, Atari did not seem interested in emphasizing any other applications of the personal computer.

Commodore made a half-hearted attempt at an exhibit, showing only the VIC-20 with very little accompanying literature. Atari and Commodore missed a great opportunity to position their products relative to other systems. IBM, Xerox, Sony, Epson, Fortune and other large microcomputer manufacturers did not even bother to show products.

New Inventions

There were some pleasant surprises for those brave souls who could ignore the magnetic pull of the music in the amphitheater and spend time meeting people in smaller exhibits. There was even a robot museum with highly amusing and anachronistic applications garnered from a decade of broken toasters and toys.

One of the more interesting entries came from Eclectic Electric located in Palo Alto, CA. Lucia Grossberger and Harry Vertelney, authors of the *Designer's Toolkit,* demonstrated their new graphic system marketed by Apple Computer. A plastic grid is placed over the Apple Graphics Tablet. Some of the features of the *Designer's Toolkit* include over 20 paint brushes, inverse video, cropping and enlargement of image sections, character input from the keyboard, split screen, and geometric design. The system is a nice synthesis of art and science. Also



featured were Lucia's 5 1/4 inch floppy Art Disks. These images included continuous variations on geometrics and rock heroes.

Numia Institute displayed an Apple Computer controlled hydroponic farm. On a twenty acre farm outside Tucson, Arizona, farm owners Tony Crow and Jenny Starlight set up shrimp and warm water fish "greenhouses" alongside their vegetable fields. The Talapia fish and shrimp are nourished by detritus from poultry manure. When the water from the fish ponds becomes saturated with fish waste, it is dumped on the fields. A fresh water well then replenishes the ponds. Numia Institute is interested in forming an information network with other computer users. They can be reached at Box 532, Marana, Arizona.

For computer users looking for an alternative to the more expensive information networks, CommuniTree Group of San Francisco presented their low cost conference system. The heart of the *CommuniTree Network* is a 48K Apple II Plus. Director Dean Gengle claims that 90 percent of all terminals can actually access the system. After initial purchase of the software, businesses, educational facilities and individuals pay only $15 per year to register their phone number with CommuniTree. Purchasers who elect to keep their phone numbers private may still receive updates from CommuniTree. Nationally, there are about 150 subsystems. Typically, each subsystem has 100 to 150 individual users. CommuniTree plans to expand their software systems for IBM® PCs and TRS-80® Model III computers.

Novation of Tarzana, California, presented their new portable terminal, the *Infone*. The device features a standard Qwerty upper/lower keyboard, with full ASCII character set. It has CPU memory of 8 to 32K, 2 to 8 bytes of text memory, and up to 256 bytes of speech memory. It has three modes of modem operation with a maximum 1200 baud rate. All modes have auto answer. It has a built-in speaker with volume control and an elegant, slim handset. Incredibly, its dimensions are 1 1/2 inches X 7 inches X 10 3/4 inches, and it weighs only 2 1/2 pounds. Novation also provides an acoustic coupler accessory which can be used anywhere. The terminal sells for about $1150.

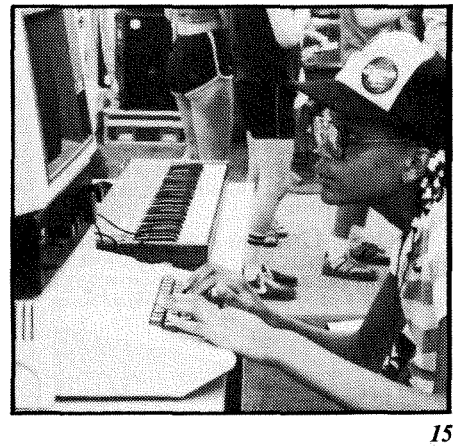
Gordon French, of Square-1 in Menlo Park, California, astonished the crowds with his demonstration of *Floppy Armour*, a cardboard mailer. Volunteers defied the laws of physics by folding 5 1/4 inch floppy disks inside the mailers. The mailer takes all the crease of the fold, leaving the disk unharmed. Quantities of 100 5 1/4 inch mailers sell for $79; 100 8-inch mailers cost $180.

Computer Music

For a music festival, there were surprisingly few music applications. Two groups always had large crowds — Syntauri Corporation, creators of the *Alpha Syntauri* system, and Passport Design, Inc. from Half Moon Bay, California, manufacturers of the *Soundchaser* Music System.

In addition to demonstrating the *Alpha Syntauri* system, Syntauri also featured performers. Emmett Chapman of Stik Enterprises played his stringed invention, the Stik. Simultaneously, youthful groups gathered to work out on Syntauri's new drum interface. A small drum surface and a pair of earphones allowed eight to nine musicians the privacy of their own beat. Best of all, each player could hear how the rhythm would actually sound.

Passport Designs demonstrated their monophonic transcription system for the *Soundchaser* — the *Notewriter.* It writes one line of music at a time as it is played on a keyboard. Editing features can move themes, add notes, change meter and keys. *Notewriter* also prints the transposition.



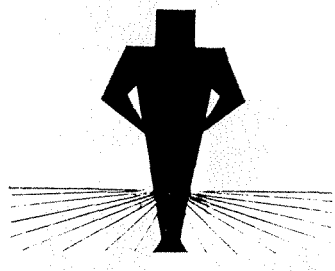
The very last day of the *US Festival* brought the largest crowds to the technology tents, as many visitors decided to take the last opportunity to view the exhibits. Frankly, the tents were not large enough to accommodate the crowds who used the occasion to spend the afternoon in an air conditioned environment. Consequently, there was not much circulation of people. However, readers who have attended other computer fairs and expositions should not be discouraged.

It's a great idea to include new technology as part of a rock festival. Wozniak said that most fairs include some sort of crafts, but after a while all belt buckles look the same. Hopefully, his festivals, now tentatively scheduled for next Memorial Day and Fourth of July, can create a larger role for microcomputer technologies. The facility is now the property of the County of San Bernardino. Because that vast playground is established, more of Wozniak's attention can be turned towards making the computer and technology fair more entertaining and informative.

His corporation, UNUSON, has credibility in terms of creating a large rock event. It still must prove itself in creating a truly unique technology fair. It is appropriate to provide a forum for small computer companies and individual application, but large manufacturers should also be allowed to sell their products. There could be prizes for debugging contests, best spreadsheet templates, graphic art, sound synthesis, and other imaginative applications. Every year thousands more people become microcomputer users, tinkerers, and dreamers. UNUSON needs to expend more effort to attract these users to come and participate.

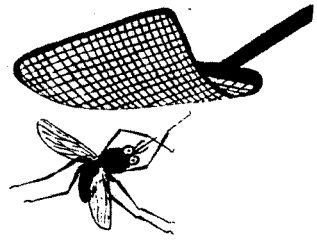
NEW FOR ATARI FROM MMG MICRO SOFTWARE

*NOW — THE **TWO** MOST POWERFUL AIDS FOR ATARI BASIC!!*



BASIC COMMANDER

- **Single key entry file commands**
  - — ENTER "D:
  - — SAVE "D:
  - — LIST "D:
  - — RUN "D:
  - — LOAD "D:
- **Single Key DOS functions from BASIC**
  - — FORMAT a disk
  - — LOCK a file
  - — RENAME a file
  - — UNLOCK a file
  - — DELETE a file
  - — DISK DIRECTORY
- **THREE PROGRAMMABLE KEYS!!**
  - — single keys programmed for your own use, even whole subroutines
- **AUTONUMBER**
  - — automatically generates line numbers for you — speeds program entry 25-75%
- **BLOCK DELETE**
  - — deletes any range of lines instantaneously!
- **RENUMBER**
  - — renumbers lines and all references
  - — extensive error trapping
  - — 3 seconds to renumber 500 lines



MMG BASIC DEBUGGER

- **TRACE through your Basic program**
  - — Single step
  - — TRACE while
  - — TRACE UNTIL
  - — change variables
  - — LIST line numbers executed
  - — examine variables' values
- **Full screen BASIC editing**
  - — scroll up or down by cursor
  - — edit your whole program easily
  - — no more LIST line number ranges
- **Split screen mode**
  - — view two parts of your BASIC program at once, and edit both!
  - — scroll each window independently
- **CROSS REFERENCE**
  - — provides a list of variables and the line numbers in which they are used in your program
- **SEARCH FOR PHRASE**
  - — search your BASIC program for any phrase, command or string of characters; let your computer do the searching for you!

EACH PRODUCT ALONE REQUIRES 16K, AND IS AVAILABLE ON DISK FOR *ONLY $34.95*

Now, the convenience of both powerful utilities together in your Atari *at once*

The Combined Basic Commander and MMG Basic Debugger requires 24K. $74.95

ADDITIONAL PRODUCTS

NECESSITIES

RAM TEST II - The fastest and most thorough memory test available for the ATARI has now been further improved! Tests not only all locations, but also tests the memory addressing system. This all machine language program takes 4 minutes to test 48K. It's the only program that tests the cartridge area of RAM. Good for new 400/800 computer owners, for testing new RAM boards and for use in computer stores to test and pinpoint bad memory locations. Bad memory locations are pinpointed so repair is as simple as replacing a chip!

Requires 8K, Disk or Cassette **$29.95**

DISK COMMANDER II - Just save this program on your BASIC disks and it will autoboat and automatically list all programs for the disk into your screen. Simply run any program by typing a single number.

Requires 16K, Disk Only **$29.95**

TUTORIALS

ASTEROID MINERS - This 50 page book and program provides for a unique intermediate to advanced tutorial. A 32K BASIC game utilizing over 25 players in player-missle graphics, machine language subroutines, a redefined character set, multiprocessing utilizing the vertical blank interrupt interval, and much more! The 50 page book included with the program documents each part of the entire program and contains the fully documented source code for both the BASIC and assembly language parts of the program. Use these routines in your own programs. These examples make it easy!

Requires 32K, Disk or Cassette **$34.95**

GAMES

CHOMPER - An all machine language arcade style game with intelligent monsters. Requires 16K Ram, 1 Joystick and nerves of steel.

Available on Disk or Cassette **$29.95**

BUSINESS/HOME

MAILING LIST - Extremely fast BASIC and machine language program. Each data disk holds over 500 files. Sort on any of 6 fields at machine langugage speed or search on any fragment of a field! Use any size labels or envelopes.

Require 40K, Disk Only **$39.95**

NEW

MMG DATA MANAGER - If you frequently find yourself looking for something, only to find it eventually right under your nose, then MMG DATA MANAGER is for you. Organize virtually anything into a computer-searchable format, and let your ATARI do the hunting for you. MMG DATA MANAGER is the first of a series of business applications from MMG MICRO SOFTWARE, all of which will share the ability to access files created by any of them. This flexible database manager will allow many fields, with machine language sorting, on any field. In addition, you have total control of the structure of your data, allowing you to design a database which you feel most comfortable. A special feature of MMG DATA MANAGER is its ability to select for a given value of any single field, or any combination of values from many fields. You could, for instance, determine who lived in Las Vegas, Nevada, and bought item #3145 from you, and whose last name began with SM, and whose telephone number began with (702)873-4. You'll never lose track of information again! Multiple print options add to the versatility of MMG DATA MANAGER.

Requires 40K, Disk Only **$49.95**

Available At Your Favorite Computer Store

OR Send a Check or Money Order to:

MMG MICRO SOFTWARE

P.O. Box 131 • Marlboro, NJ 07746

OR CALL

(201) 431-3472

For MasterCard, Visa or COD Deliveries

(Please Add $3.00 For Postage & Handling)

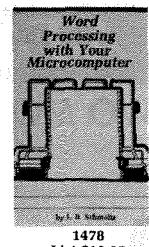
NJ RESIDENTS ADD 5% FOR SALES TAX

The Computer Book Club

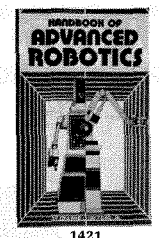
"The ONLY Book Club for micro hobbyists"

Save *time* and money
Keep up with the latest in computer books and software *without leaving your terminal!*

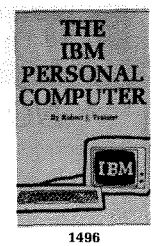
Select 5 fact-filled books for ONLY $2.95 (values to $103.75)



1478
List $19.95



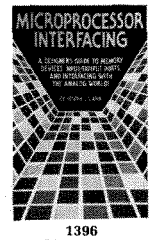
1421
List $21.95



1496
List $17.95

1160
List $13.95

1295
List $16.95

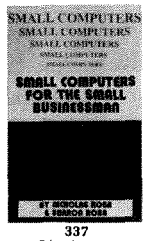


1396
List $13.95

1485
List $21.95

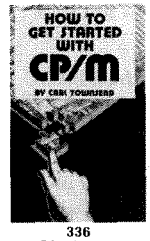
1251
List $16.95

1205
List $16.95



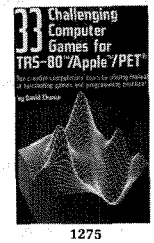
337
List $19.95

1169
List $17.95



336
List $14.95

1277
List $19.95



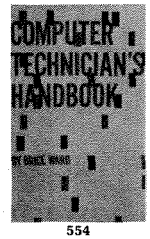
1275
List $14.95

1428
List $17.95

1391
List $17.95

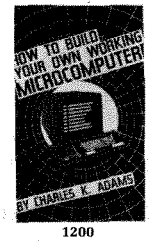
1299
List $16.95

1394
List $15.95

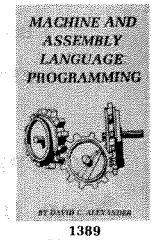


554
List $15.95 (paper)

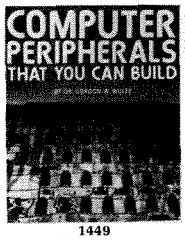
1195
List $13.95



1200
List $16.95



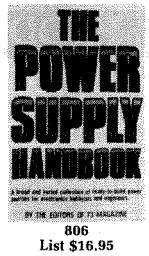
1389
List $15.95



1449
List $19.95

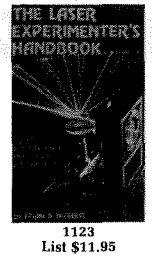
1228
List $13.95

1468
List $18.95



806
List $16.95

1332
List $16.95



1123
List $11.95

1398
List $16.95

1506
List $12.95

7 very good reasons to try The Computer Book Club
Blue Ridge Summit, PA 17214

- **Reduced Member Prices.** Save 20% to 75% on books sure to increase your computer know-how
- **Satisfaction Guaranteed.** All books returnable within 10 days without obligation
- **Club News Bulletins.** All about current selections—mains, alternates, extras—plus bonus offers. Comes 13 times a year with dozens of up-to-the-minute titles you can pick from
- **"Automatic Order."** Do nothing, and the Main selection will be shipped automatically! But . . . if you want an Alternate Selection—or no books at all—we'll follow the instructions you give on the reply form provided with every News Bulletin
- **Continuing Benefits.** Get a Dividend Certificate with every book purchased after fulfilling membership obligation, and qualify for discounts on many other volumes
- **Extra Bonuses.** Take advantage of added-value promotions, plus special discounts on software, games, and more
- **Exceptional Quality.** All books are first-rate publisher's editions, filled with useful, up-to-the-minute information

The Computer Book Club
Blue Ridge Summit, PA 17214

Please accept my membership in The Computer Book Club and send the 5 volumes circled below, billing me $2.95 plus shipping and handling charges. If not satisfied, I may return the books within ten days without obligation and have my membership cancelled. I agree to purchase 4 or more books at reduced Club prices (plus shipping/handling) during the next 12 months, and may resign any time thereafter.

336 337 554 806 1123 1160 1169 1195
1200 1205 1228 1251 1275 1277 1295
1299 1332 1389 1391 1394 1396 1398 1421
1428 1449 1468 1478 1485 1496 1506

Name _____ Phone _____

Address _____

City _____

State _____ Zip _____

(Valid for new members only. Foreign and Canada add 20%. Orders outside U.S. or Canada must be prepaid with international money orders in U.S. dollars.) This order subject to acceptance by The Computer Book Club. SFSD-183

Starting Forth

Reviewed by Peter J. Favaro

by Leo Brodie, FORTH INC. (Prentice-Hall, Inc., Englewood Cliffs, NJ 07632) Price: $15.95

If you have ever attempted to teach yourself a skill, you can appreciate both the pains and the pleasures of such a task. I, like many other game designers, software developers and hobbyists, am a self-taught computer programmer who has driven into many ditches on the road to proficiency.

Learning BASIC was a chore. After breezing through the beginners level with a severely and prematurely inflated ego ("Hey, this stuff is a piece of cake, bring on the hard stuff!"), I stumbled through the more advanced levels with feelings of frustration, incompetence and bewilderment, as the instructional articles and books that I read all assumed that I had an expert's knowledge behind me. Sure, there was a great deal of information on BASIC programming techniques, but none of it ever seemed to suit my present needs or ability level. Much of what I read simply appeared to be "alphabet soup." Remember all of those program listings that did wonderful things once you typed them in and debugged them, and you couldn't understand why? How frustrating! There was no solid text to support what was happening in those listings. What made it worse was that the authors kept telling you how easy this or that application was. Before read-Leo Brodie's *Starting Forth* (Prentice-Hall, $19.95), I had made an unconscious vow to be satisfied with my knowledge of BASIC, Microsoft BASIC and Assembler for my particular computer. Enticed by the irresistible charm and "reader friendliness" of this book, however, I have decided to learn FORTH, in spite of my prior experiences.

Before I begin to summarize both the fine points of Mr. Brodie's book and the advantages of FORTH in general, from a beginner's point of view, let me say that *Starting Forth* is written for everyone. Yes, *everyone* — from the rank amateur to the seasoned computer professional. Mr. Brodie begins his book by assuming that the reader has absolutely no knowledge of FORTH and, moreover, no knowledge of computers whatsoever. He holds his readers' hands and walks them through the basics of FORTH, structured programming philosophy, computer architecture, and much more. What a relief. No pressure to perform, no wondering about how much background the author expects you to have in his area. For those of you who feel comfortable enough without this introduction, my advice is to roll up your sleeves and dive right into the more advanced chapters. I, on the other hand, am one of those people who, in the summertime in the sweltering heat, sit by the steps of the pool "just getting my feet wet." Suffice it to say, I like to take it nice and slow.

In the opening chapter, (written as the introduction to beginners) Mr. Brodie explains that FORTH is a highly structured and very precise language whose power lies in the fact that the FORTH programmer can create user-specific definitions designed to carry out specific tasks within a program. For instance, all Atari® users know that it is difficult to move a player object vertically on the screen. In FORTH, a definition that the programmer might call VMOVE could be stored in the FORTH "dictionary" and used to perform this task. The FORTH word for "definition" is simply "word", so it is said that the user is constantly updating the FORTH "dictionary" with new "words." The fact that I used a game application to demonstrate a FORTH concept was no accident. FORTH promises to be an excellent game development language, largely because of its speed, which is about ten times faster than BASIC (although not quite as fast as Machine Language).

"Starting Forth is written for everyone... From the rank amateur to the seasoned computer professional."

In walking us through the computer architecture, Mr. Brodie explains that FORTH is a "stack-oriented" language with commands, (actually words) such as SWAP, DUP, and ROT, which pertain directly to stack operations. The "stack," as most assembly language programmers know, is a key area in RAM where much of the data manipulation is performed in a "last in, first out" fashion. This means that the first number to be put into the computer goes down on the stack and doesn't come off the stack until all the numbers which follow it are taken care of or operated on. Understanding how all this works can get very confusing. Even simple arithmetic operations are written in RPN (reverse Polish notation), so that even a straightforward operation like "2+2" becomes "22+." However, the patient reader can adapt to FORTH's topsy-turvy world. What would seem to be a syntax that borders on the bizarre eventually becomes very clear. Mr. Brodie explains concept after concept in a logical, orderly progression, using every technique he can muster up, from humorous cartoons (which he illustrates himself) to meaningful comparisons to everyday living.

After this rather important groundwork is laid and the reader becomes familiar with FORTH syntax and structure, it is all downhill. Chapter three teaches the functions of the FORTH editor and the principles of using the magnetic disk as a storage device. Briefly, disk memory is

divided into units called "blocks," each of which holds 1024 characters, or 16 lines of 64 characters each. This sixteen line block is also called a "screen," and FORTH programs are written and organized screen by screen, or block by block. Chapter five presents a nice discussion about the relative advantages of using a fixed point package over a floating point package to do arithmetic and express numbers. Some good arguments are presented here, even though many commercially available FORTHs come with a floating point package. It was when I finished reading this chapter that I began to realize what a good book *Starting Forth* is. Before reading it, if someone had come up to me and asked me if I thought I could be interested in reading about the merits of using a fixed versus floating point package I would have said, "never." Trust me, it's worth taking a look at. The remaining chapters deal largely with programming concepts which include decision-making, branching, looping, an excursion into variables, an introduction to the binary number system, input/output operations and much more. There are also useful examples, sample FORTH screens, and a handful of charts and reference tables listing the FORTH words, which are located in the back of the book. Mr. Brodie leaves absolutely no stone unturned.

"As it stands right now, Starting Forth promises the most success in helping the reader learn FORTH on his or her own."

In summary, the advantages of learning FORTH are its speed, flexibility and power. Its disadvantages are apparent only to those who feel uncomfortable letting go of conventional programming syntax, or put off by the fact that a FORTH program takes a little longer to write than, say, a BASIC program. If you decide that FORTH is for you, the next step is selecting some reading material that will help you learn the language. As it stands right now, *Starting Forth* promises the most success in helping the reader to teach himself FORTH. It is by far the most comprehensive book on the subject available. One word of caution, though. You will never be able to read this book just once. The beginner should read it three times — once to benefit from the general discussions of computer architecture and program structure, once to learn the FORTH words, and once to put everything together.

If You Sell Small Business or Personal Computer Systems, Word Processing, Software, Media & Supplies, or Computer Services...

Have We Got An Offer For You!

Why Prospect?

The answer is obvious. Without prospecting... No Business. But, there's a better way...COMPUTER SHOWCASE EXPO!

This proven series of regional, end-user expositions has brought thousands of prospects, with needs in mind and buying dollars in hand.

THE INTERFACE GROUP, the world's largest producer of computer shows, offers you instant availability to a wide variety of end-users such as:

- **Doctors • Lawyers • Accountants • Retailers**
- **Corporate Managers • Many Other Professionals**

These face-to-face contacts are in businesses such as:

- **Banking • Health Care • Insurance • Education**
- **Government • Real Estate • Manufacturing**
- **Distribution**

If you sell computers, or computer systems, as a solution for a variety of Small Business problems, COMPUTER SHOWCASE EXPO offers you more prospects in a few days than you could possibly see in months!

For more information on how you can reach the market you need, call THE INTERFACE GROUP, toll free at (800) 225-4620.
(In Massachusetts, call 617-879-4502)

COMPUTER SHOWCASE EXPO

SPRING 1983

PHOENIX
Feb. 3-6, 1983

ATLANTA
March 24-27, 1983

CHICAGO
March 24-27, 1983

ST. LOUIS
April 21-24, 1983

SOUTH FLORIDA
April 28-May 1, 1983

BOSTON
April 28-May 1, 1983

ANAHEIM
May 5-8, 1983

WASHINGTON, DC
May 5-8, 1983

SAN DIEGO
May 12-15, 1983

HOUSTON
May 19-22, 1983

SEATTLE
June 2-5, 1983

FALL 1983

DETROIT
Sept. 22-25, 1983

NEW YORK
Sept. 22-25, 1983

SAN FRANCISCO
Sept. 29-Oct. 2, 1983

ATLANTA
Sept. 29-Oct. 2, 1983

PHILADELPHIA
Oct. 6-9, 1983

PITTSBURGH
Oct. 20-23, 1983

SOUTH FLORIDA
Oct. 27-30, 1983

DENVER
Nov. 3-6, 1983

LOS ANGELES
Nov. 10-13, 1983

WASHINGTON DC
Nov. 10-13, 1983

CHICAGO
Nov. 17-20, 1983

THE INTERFACE GROUP
World's Largest Producer of Computer Conferences and Expositions

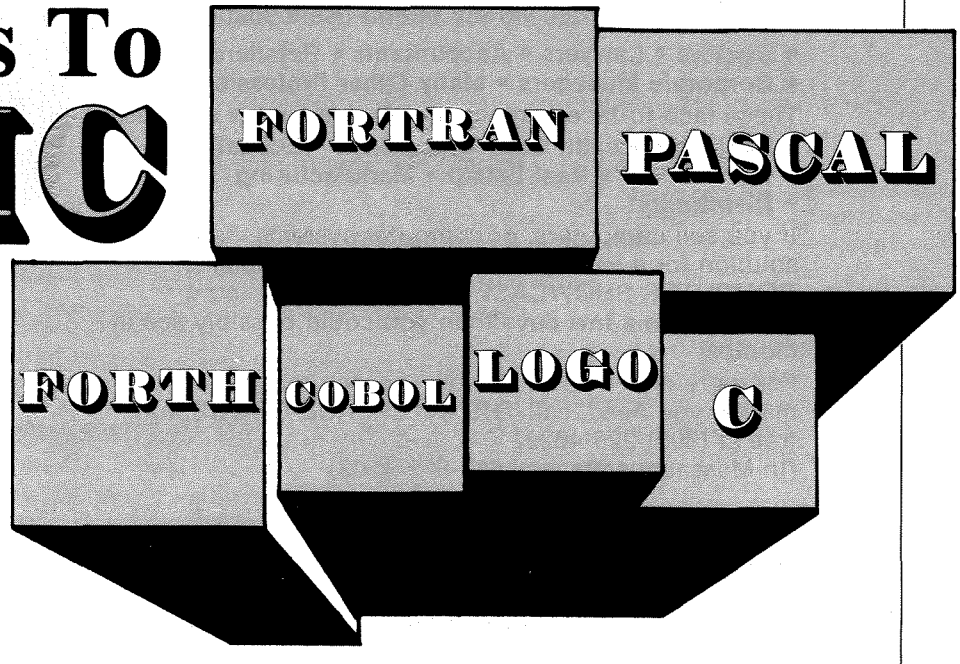
Producers of THE COMPUTER SHOWCASE EXPOS (Nationwide), INTERFACE, FEDERAL DP EXPO, COMDEX/SPRING, COMDEX/FALL, COMDEX/EUROPE

160 Speen Street, Framingham, MA 01701 • (617) 879-4502, Toll Free (800) 225-4620, TELEX - 951176, TWX - 710-380-7645
Regional Office: Suite 121, 4700 North State Road 7, Fort Lauderdale, FL 33319 • (305) 484-6800 Toll Free (800) 327-0100

Alternatives To BASIC

by Allen L. Wold

"Of the two hundred or so languages available, the ones most widely advertised and used on microcomputers, aside from BASIC, are Pascal, C, FORTH, and LOGO. Each of these four languages has its own proponents, its own reasons for being a suitable alternative to BASIC."



Why an Alternative?

There are over two million microcomputers in homes and offices today, and the number is growing minute by minute. BASIC is the programming language most commonly used in these microcomputers, in uncounted dialects and variations. But the ads in any computer magazine reveal that other languages are offered to the microcomputer user, and articles in these and other magazines and technical books suggest that BASIC might not be the best language to use. Indeed, Edsger W. Dijkstra, who is said to have invented the whole idea of unstructed programming, once wrote that, in his opinion, learning BASIC mentally damaged the programmer "beyond hope of regeneration."

Whether this is true or not, a number of people are concerned with finding an alternative to BASIC. BASIC does, after all, have certain severe limitations — it is structured, it takes a lot of memory, it is slow, it is not self-documenting, its file-handling is clumsy, it sometimes requires Machine Language subroutines, it encourages sloppy programming.

Another problem is the wide variety of dialects, or differing forms of BASIC. Some of these are available for microcomputers in general, and some are proprietary to specific brands of micros. Programs written on one machine will probably not run on a different machine. One written with one compiler or interpreter will certainly not run on another, without considerable hand translation. BASIC is not very transportable.

The very variety of BASICs available, each designed to conform to a particular set of prejudices, implies that its limitations are not inherent in the language itself. Some versions are compiled and fast. Others are nearly structured without line numbers. There are versions with graphic and sound capabilities, advanced mathematical functions, and so on. A compiler or interpreter could be written to include almost any kind of function or operation one desired.

BASIC still, in most cases, uses a casual approach to logic, lots of GOTO statements, and in some cases requires the programmer to deal directly with memory locations by means of PEEK and POKE statements. Compensating for this is the fact that BASIC is easy to learn (too easy, some say), and is already, by default, the common tongue.

BASIC was not intended to be the end-all of computer languages. Its name is an acronym for *Beginner's All-purpose Symbolic Instruction Code*. It was originally designed, at Dartmouth, as a very simple language to be used on a time-sharing system to familiarize computer students with a programming language they could learn interactively while waiting for their FORTRAN or COBOL batches to turn around on the main frame. Since then it has, like a fungus, simply grown.

Of the two hundred or so languages available, the ones most widely advertised and used on microcomputers, aside from BASIC, are Pascal, C, FORTH, and LOGO. (COBOL and FORTRAN do not seem to be offered as *alternatives*, though they, too, are widely advertised.) Each of these four languages has its own proponents, its own reasons for being a suitable alternative to BASIC.

The Question of Language

It's unfortunate that the term "language" is used in reference to BASIC, SNOBOL, RPG, and ALGOL, though eventually that term may be more accurate than it is now. Language, as it commonly understood, is a means whereby people communicate with each other. It is a tool for the exchange of ideas, infinitely flexible and extremely complex. Indeed, the rules which govern English are so complex that they are not yet understood. (Whatever those rules are, they are not the ones, derived from Latin, which some of us older folks learned in grade school.) The science of Linguistics is trying hard to discover those rules, and some good generalizations have been made, but every description is full of exceptions and contradictions.

Verbal language, however, follows another kind of logic than that used in mathematics or computers. When we control a computer, what we use is not a language in the above sense, but instead a specialized code which is translated into machine-dependent computer instructions. It resembles human language only because we make it that way. Human language, the way we think if we're being verbal, is not the strict mathematical logic of computers.

We do not really "communicate" with a computer, since communication implies a two-way process, an exchange of information. Interaction between a computer and the user gives the illusion of communications, but it is only illusion. The computer itself, so far, contains nothing new, does not "think" independently, is not creative.

Our communication with a computer is merely an elaboration of our communication with a table of logarithms or a card index. As computers become more powerful and complex, this will become less true. However, it will always be true to some extent.

While we may not be able to actually communicate *with* a computer, we can communicate *through* a computer. This does not mean the use of modems or electronic bulletin boards. Rather, it means interactively programming a computer with ideas which the computer interprets into new ideas (or at least new configurations) which are then "read" by other people.

"When we control a computer, what we use is not a language...but instead a specialized code which is translated into machine-dependent computer instructions."

For example, you might have an idea for aerodynamic airfoils, expressed in mathematical terms and physical concepts. I know nothing of the subject, and think in words and pictures. Using a computer, you can input your ideas, and the computer will give me a literally graphic demonstration — pictures, of what you're thinking about. I respond in words, which the computer translates into mathematical possibilities.

Natural vs. Artificial

There is a difference between programming languages and human languages, of course. It's not just that programming languages are small, elegant (sic), strictly mathematical; and that human verbal languages are large, inelegant, and irrational, though those differences are significant. More importantly, computer languages are artificial constructs, while human verbal languages grew naturally.

In a way, BASIC is more natural than, say, Pascal. One can compare Pascal with classical Latin. Although not what people spoke most of the time, classical Latin was designed by its grammarians to be what they thought the perfect language should be.

There have been a number of attempts to create an artificial verbal language. The objective has been to provide an alternative to English (or French, or Russian, or any other natural language) so that people could communicate with each other, regardless of nationality.

Languages such as Volapuk, Idiom Neutral, Ido, Esperanto, Loglan, and so on, have tried to capture the essence of language communication, using logical and consistent rules of grammar and spelling. Some claim to incorporate the basic elements of every language ever spoken, though they fail utterly. Esperanto, for example, makes little use of Chinese, Cherokee, or any African tongue. Each artificial language was designed to improve communications between people, to be more precise, to be easy to learn.

The lack of success on the part of their creators, however, is illustrated by three facts. First, few people speak any of these languages, there is more than one "universal" language, and even the names of some are unknown to all but a few scholars.

This is due, partly, to simple inertia, and the difficulty of learning languages in general. It takes a dozen or so years, from the moment of birth, to truly learn one's native language, which implies that languages are too complex to be designed by any one person or group of persons.

These artificial languages, in spite of high ideals, do not, in fact, reflect universal language concepts, only those of a particular language family, and the linguistic biases of their designers. In fact, any natural language could serve as the universal language, if only it were taught to

everybody. An artificial language is not needed, except to avoid the question of which natural language to use, and hence avoid the problem of chauvinism and national rivalry.

Such artificial languages, like Latin and classical Greek, are frozen. If they are allowed to change, they will develop dialects of all sorts, and within a short while will be unintelligible to other speakers. Even with mass communications, this is happening with English today. Look at the following sentence: "The dustman left a tin of sweets in his shooting brake on the roundabout and walked through the subway." In "American," that means the trash collector left a box of candy in his station wagon on the traffic circle and walked through the pedestrian underpass.

To Serve Or Be Served

Fortunately, in order to create a computer language, one does not have to develop the full complexity and richness of a verbal language. One only has to design a system which serves two functions — it is readily comprehensible by a human, and it conveys precise instructions to the computer. However, all the points mentioned above in regard to natural vs. artificial verbal languages also apply to computer languages.

With the first computer languages, and even with newer languages like Pascal and FORTH, the guiding principle was to serve the computer. Memory was small. The compiler took up a lot of space, as did the editor. Frequently, both could not be resident in the computer at the same time, and neither could be resident while the program was being run. One had to be kind to the computer if one was to get any work out of it at all. Variables had to be declared to make it easier on the compiler. Names had to be kept short and be related to the variable types. Program logic had to resemble computer logic.

"As the speed and power of the computer and the size of memory grow, and as human time becomes more expensive, it makes less sense to force the user to serve the computer, rather than have the computer serve the user."

Languages like LISP, on the other hand, are specifically designed to be huge and complex interpreters which serve the user, at the expense of memory and time. Languages like LISP, and its offspring LOGO, are now possible since even microcomputers have as much user memory as the main frames of a couple of decades ago. With the ability to use virtual memory, i.e., having most of the program stored on disk, with parts to be called up as necessary, even larger interpreters and compilers are possible. The whole program can exceed the memory capacity of the computer itself (as my word-processing software does).

As the speed and power of the computer and the size of memory grow, and as human time becomes more expensive, it makes less sense to force the user to serve the computer, rather than have the computer serve the user. We have more valuable things to do than convert our instructions into Machine Language, or to follow artificial grammars designed to please a compiler.

Computer languages don't have to serve the compiler any more. Consider the following dialogue, taken from the October, 1982 issue of *Scientific American*. It is part of a run of the program called SHRDLU, an experiment in artifical intelligence, in which a graphics terminal shows a picture of blocks and pyramids of various sizes and colors, in various places on the table. The italicized sentences are "spoken" by the computer, in response to instructions from the operator, working at a keyboard.

Will you please stack up both of the red blocks and either a green cube or a pyramid?
OK.
Which cube is sitting on the table?
The large green one which supports the red pyramid.
Put a small cube onto the green cube which supports a pyramid.
OK.
Put the littlest pyramid on top of it.
OK.
How many things are on top of green cubes?
I'm not sure what you mean by "on top of" in the phrase "On top of green cubes." Do you mean: 1) Directly on the surface, or 2) Anywhere on top of?
2 (anywhere on top)
Three of them.
Had you touched any pyramid before you put the green one on the little cube?
Yes, the green one.
When did you pick it up?
While I was stacking the red cube, the large red block and a large green cube.

continued on page 26

Kelly's Computing

New Toll Free Number
800-572-9215

$2.00 OFF COUPON

ATARI IBM

ATARI

☐ 400 COMPUTER	399.00	Now	299.00
☐ 800 COMPUTER	899.00	Now	660.00
☐ 810 DISK DRIVE	599.00	Now	440.00
☐ 410 CASSETTE RECORDER	99.00	Now	87.50
☐ 850 INTERFACE MOD	219.00	Now	180.00
☐ PAC MAN (CART)	44.95	Now	38.50
☐ CENTIPEDE (CART)	44.95	Now	38.50
☐ CAVERNS OF MARS (DISK)	39.95	Now	33.50
☐ STAR RAIDERS (CART)	49.95	Now	33.50
☐ SPACE INVADERS (CART)	44.95	Now	27.50
☐ ASTEROIDS (CART)	44.95	Now	27.50
☐ MISSILE COMMAND (CART)	44.95	Now	27.50
☐ JOYSTICKS (PAIR) (ACCS)	21.95	Now	19.50
☐ PADDLES (PAIR) (ACCS)	21.95	Now	19.50
☐ GOLD EDITION 1-12 (DISK & CASS) (WHILE THEY LAST)	100.00	Now	87.50
☐ ADVENTURES 1-12 (CASS) 24.95 ea.	24.95	Now	19.50
☐ LUNAR LANDER (CASS)	14.95	Now	13.50
☐ STAR FLITE (DISK)	19.95	Now	16.50
☐ REAR GUARD (DISK)	19.95	Now	16.50
☐ GALACTIC EMPIRE (CASS)	19.95	Now	16.50
☐ GALACTIC TRADER (CASS)	19.95	Now	16.50
☐ PREPPIE (DISK)	29.95	Now	25.50
☐ PREPPIE (CASS)	29.95	Now	25.50
☐ TEMPLE OF APSHAI (DISK & CASS)	39.95	Now	33.50
☐ STAR WARRIOR (DISK & CASS)	39.95	Now	33.50
☐ RESCUE AT RIGEL (DISK & CASS)	29.95	Now	25.50
☐ DATESTONE OF RYN (DISK & CASS)	19.95	Now	16.50
☐ CRUSH, CRUMBLE & CHOMP (DISK)	29.95	Now	25.50
☐ INVASION ORION (DISK & CASS)	24.95	Now	19.50
☐ UPPER REACHES OF APSHAI	19.95	Now	16.50
☐ CURSE OF RA	19.95	Now	16.50
☐ CRYPTS OF TERROR (DISK)	34.95	Now	29.50
☐ CRYPTS OF TERROR (CASS)	29.95	Now	25.50
☐ EMPIRE OF THE OVERMIND (DISK)	35.00	Now	29.50
☐ TANKTICS (CASS)	24.00	Now	19.50
☐ CONTROLLER (DISK)	30.00	Now	25.50
☐ EMPIRE OF THE OVERMIND (DISK)	35.00	Now	29.50
☐ EMPIRE OF THE OVERMIND (CASS)	29.50	Now	25.50
☐ SHAMUS (DISK & CASS)	34.95	Now	29.50
☐ MASTER TYPE (CASS)	39.95	Now	33.50
☐ BASKETBALL (CART)	34.95	Now	29.50
☐ SUPER BREAKOUT (CART)	44.95	Now	33.50
☐ SCRAM (CASS)	24.95	Now	19.50
☐ TOUCH TYPING (CASS)	24.95	Now	19.50
☐ MAILING LIST (CASS)	24.95	Now	19.50
☐ MACRO ASSEMBLER & TEXT EDITOR	89.95	Now	78.50
☐ ASSEMBLER	59.95	Now	50.50
☐ FILE MANAGER 800 (DISK)	99.95	Now	87.50
☐ DISK MANAGER (DISK)	29.95	Now	25.50
☐ MICRO PAINTER (DISK)	34.95	Now	29.50
☐ GRAPHIS MASTER (DISK)	39.95	Now	33.50
☐ SAMMY THE SEA SERPENT (DISK & CASS)	24.95	Now	19.50
☐ PAGE 6 (DISK)	34.95	Now	29.50
☐ DISK DETECTIVE (DISK)	29.95	Now	25.50
☐ PATH FINDER (DISK)	34.95	Now	29.50
☐ SPEED READ PLUS (DISK)	59.95	Now	50.50
☐ SPACE EGGS (DISK)	29.95	Now	25.50
☐ GALAXY (DISK)	25.00	Now	19.50
☐ B-1 NUCLEAR BOMBER (CASS)	16.00	Now	13.50
☐ MIDWAY CAMPAIGN (CASS)	16.00	Now	13.50
☐ NORTH ATLANTIC CONVOY RAIDER (CASS)	16.00	Now	13.50
☐ NUKEWAR (CASS)	16.00	Now	13.50
☐ CONFLICT 2500 (CASS)	16.00	Now	13.50
☐ LORDS OF KARMA (CASS)	20.00	Now	16.50
☐ APPLE PANIC (DISK)	29.95	Now	25.50
☐ RASTER BLASTER (DISK)	29.95	Now	25.50
☐ BUG ATTACK (DISK)	29.95	Now	25.50
☐ CANYON CLIMBER (DISK & CASS)	29.95	Now	25.50
☐ LE STICK (ACCS)	39.95	Now	33.50
☐ PACIFIC COAST HIGHWAY (DISK & CASS)	29.95	Now	25.50
☐ INTRUDER (DISK)	34.95	Now	29.50
☐ INTRUDER (CASS)	29.95	Now	25.50
☐ SHOOTING ARCADE (DISK & CASS)	29.95	Now	25.50
☐ TRACK ATTACK (DISK)	29.95	Now	25.50
☐ CLOWNS & BALLOONS (DISK & CASS)	29.95	Now	25.50
☐ MEGALEGS (DISK & CASS) (WHILE THEY LAST)	34.95	Now	29.50
☐ COMPU-READ (DISK)	29.95	Now	25.50
☐ COMPU-MATH/FRACTIONS (DISK)	39.95	Now	33.50
☐ COMPU-MATH/DECIMALS (DISK)	39.95	Now	33.50
☐ COMPU-MATH/FRACTIONS (CASS)	29.95	Now	25.50
☐ COMPU-MATH/DECIMALS (CASS)	29.95	Now	25.50
☐ MATCH RACERS (DISK)	29.95	Now	25.50
☐ PATHFINDER (AVAIL. FALL) (DISK)	34.95	Now	29.50
☐ DEADLINE (DISK)	49.95	Now	42.50
☐ ZORK I (DISK)	39.95	Now	33.50
☐ ZORK II (DISK)	39.95	Now	33.50
☐ POOL 400 (CART)	39.95	Now	33.50
☐ ACTION QUEST (DISK & CASS)	29.95	Now	25.50
☐ K-RAZY SHOOTOUT (CART)	49.95	Now	42.50
☐ SPEED READ PLUS (DISK)	59.95	Now	50.50
☐ HI-RES ADV. #0- MISSION: ASTEROID (DISK)	24.95	Now	19.50
☐ HI-RES ADV. #2- WIZ & PRINCESS (DISK)	32.95	Now	29.50
☐ CROSSFIRE (DISK)	29.95	Now	25.50
☐ JAWBREAKER (DISK & CASS)	29.95	Now	25.50
☐ THRESHOLD (DISK)	39.95	Now	33.50
☐ THE NEXT STEP (DISK)	39.95	Now	33.50
☐ ALI BABA AND THE FORTY THIEVES (DISK)	32.95	Now	29.50
☐ DELUXE INVADERS (DISK)	34.95	Now	29.50
☐ GALACTIC CHASE (DISK)	29.95	Now	25.50
☐ GALACTIC CHASE (CASS)	24.95	Now	19.50
☐ THE SHATTERED ALLIANCE (DISK)	39.95	Now	33.50
☐ PROTECTOR (DISK & CASS)	34.95	Now	29.50
☐ CHICKEN (DISK & CASS)	34.95	Now	29.50
☐ DODGE RACER (DISK & CASS)	24.95	Now	19.50
☐ WARLOCK'S REVENGE (DISK)	35.00	Now	29.50
☐ WORDRACE (DISK)	24.95	Now	19.50
☐ ALIEN SWARM (DISK)	34.95	Now	29.50
☐ ALIEN SWARM (CASS)	29.95	Now	25.50
☐ SNAKE BYTE (DISK)	29.95	Now	25.50
☐ CYCLOD (DISK)	29.95	Now	25.50
☐ JOYSTICK	29.95	Now	25.50
☐ JOYSTICK DELUXE	39.95	Now	34.50
☐ FAMOUS RED BALL	34.95	Now	29.95
☐ TRACKBALL	69.95	Now	63.50

IBM

☐ LOST COLONY (DISK)	29.95	Now	25.50
☐ TEMPLE OF APSHAI (DISK)	39.95	Now	33.50
☐ GALAXY (DISK)	25.00	Now	20.50
☐ MIDWAY CAMPAIGN (DISK)	21.00	Now	19.50
☐ COMPUTER STOCKS AND BONDS (DISK)	25.00	Now	20.50
☐ VOYAGER (DISK)	25.00	Now	20.50
☐ DRAW POKER (DISK)	21.00	Now	19.50
☐ CHAMPIONSHIP BLACKJACK (AVAIL JULY) (DISK)	39.95	Now	33.50
☐ THE HOME ACCOUNTANT PLUS (DISK)	150.00	Now	125.00
☐ WRITE-ON (DISK)	129.95	Now	110.00
☐ EASy (EXECUTIVE ACCOUNTING SYSTEM) (DISK)	725.00	Now	575.00
☐ DEADLINE (DISK)	49.95	Now	42.50
☐ ZORK I (DISK)	39.95	Now	33.50
☐ ZORK II (DISK)	39.95	Now	33.50
☐ T.I.M. III (DISK) ("Total Information Management")	495.00	Now	399.00
☐ MATHEMAGIC (DISK)	89.95	Now	75.00
☐ EASY SPELLER (DISK)	175.00	Now	149.00
☐ EASY FILER (DISK)	400.00	Now	335.00
☐ EASYWRITER II (DISK)	350.00	Now	280.00
☐ THE TAX MANAGER (DISK)	250.00	Now	199.00
☐ WORDSTAR (DISK)	495.00	Now	399.00
☐ MAILMERGE (DISK)	150.00	Now	99.00
☐ CONQUEST (AVAIL JULY) (DISK)	29.95	Now	25.50
☐ SUPERCALC (DISK)	295.00	Now	225.00
☐ SUPERWRITER (AVAIL JULY) (DISK)	395.00	Now	299.00
☐ IBM JOYSTICKS (ACCS)	64.95	Now	55.00
☐ GRAPHICS HARDCOPY SYSTEM (DISK)	24.95	Now	19.50
☐ VERSAWRITER GRAPHICS TABLET (ACCS)	299.00	Now	250.00
☐ DESKTOP PLAN I (DISK)	300.00	Now	250.00
☐ VISITREND/PLOT (AVAIL. JULY) (DISK)	300.00	Now	250.00

RAM FOR ATARI 400 + 800

48K (Reg. $299.00) *NOW* **$149.00**

32K (Reg. $199.00) *NOW* **$75.00**

COUPON
$2.00 OFF ORDER

To redeem, place a mail order for $75.00 or more and include this coupon. OFFER EXPIRES MARCH 31, 1983.

★ **TO ORDER CALL TOLL FREE 1-800-572-9215. CALIFORNIA RESIDENTS CALL 714-369-8113 OR WRITE TO 3515 BRUCE WAY, RIVERSIDE, CA 92506.** ★

WE PAY SHIPPING on software orders OVER $75 in the Continental U.S. (Foreign & Air extra). ADD $2.50 SHIPPING & HANDLING on orders under $75. **CALIF. RESIDENTS ADD 6% SALES TAX.** We accept **MASTER CARD** and **VISA. C.O.D.'s ADD $5.00.**

NAME _____

STREET _____

CITY _____ **STATE & ZIP** _____

CARD # _____ EXP. DATE _____

SIGNATURE _____

CHECK HERE FOR FREE CATALOG _____

Why?
To get rid of it.
Why did you do that.
To clear off the red cube.
Why did you clear off that cube?
To put it on a large green cube.
Why did you do that?
To stack up the red cube, a large red block, and a large green cube.
Why did you do that?
Because you asked me to.

While SHRDLU is not a programming language, a language compiler or interpreter could be written to permit the kind of dialogue seen above, even in the process of programming.

The Importance Of Purposes

Science fiction stories dealing with advanced computers frequently show the user speaking to the machine in SHRDLU-like language, sometimes even calling it programming. It certainly is user-friendly, but is a long way in the future. Until then, we have to work with what we have. One thing to consider, until the truly universal computer language is designed (or "grown") is the purpose of the language.

This is one of the most important considerations in looking for alternatives. It is fatal to try to handle strings with FORTRAN, which was designed for mathematical and engineering purposes — "number crunching." It is futile to try to use COBOL for quick and dirty programming, since writing the code takes longer than solving the problem by hand. It is hopeless to try to learn programming by studying APL, which has the most compact and unintelligible code of any in existence.

"Right now there is little public awareness of the specific purpose of any particular language. We use BASIC because it is there; or FORTRAN or Pascal because that's what we've been taught..."

Right now there is little public awareness of the specific purpose of any particular language. We use BASIC because it is there; or FORTRAN or Pascal because that's what we've been taught and are familiar with; or assembler because our prime purpose is to control the computer, or because we don't know any better, or because a high level language is not available for that computer, or...

BASIC, of course, was designed for easy use on a time sharing system, though it is much more than that. In its early days, that was all it was good for. A major text, *Progamming Languages: History and Fundamentals,* by Jean E. Sammit, Prentice Hall, 1969, gives detailed descriptions of every computer language which existed at the time, including differences in grammar, structure, uses, and so on. It devotes somewhat less than three pages out of 785 to BASIC. The fact that BASIC has since grown is another matter. Originally, it was almost trivial.

Pascal was designed as a teaching language, to enforce good programming habits in computer science students, according to the theories of structured programming. Once mastered, most other languages should be relatively easy to learn. But computer science students are going to be working on main frames, or at least large minis, not micros, for the most part. Pascal, however, is almost never used in commercial applications outside the micro industry. Also, compare the idealism of Pascal with the real life described in *Soul of a New Machine,* by Tracy Kidder.

FORTH was designed for machine control, originally to operate a large astronomical telescope. As it turns out, it is also a superb graphics language. However, its structure is quite different from any other language. It uses Reverse Polish Notation (like a Hewlett-Packard calculator), and is not at all easy to learn. For the kinds of tasks it was designed to handle, it is quite good. But you couldn't teach a secretary to do business programming with it.

C is a systems language, for writing operating systems, compilers, monitors, etc. It is a "mid-level" language, more nearly like assembler in its logic and syntax than like high level languages such as FORTRAN, which tries to simulate algebraic logic, or COBOL, which tries to resemble English. Its main strength is that it gives the user complete control of the computer. It is easier to use than assembly, and can be extended to suit any task.

LOGO, in spite of the dreams of its writers, is designed to familiarize children with computer logic and power. It's a good language in that it is user-oriented, as BASIC is randomly becoming, rather than machine oriented as are Pascal, C, and FORTH. (Its parent language, LISP, is another language of high power for Artificial Intelligence, but is as difficult to learn as

FORTH.) LOGO lacks a certain degree of power and computational sophistication, though it is not a trivial language, as BASIC once was.

At least at the outset, each language was designed for a specific purpose. The problem arises when such a language is used for, or touted to be useful for, other purposes than those for which it was designed. In selecting an alternative to BASIC, then, one must consider what one wants to do with the computer. One cannot expect an engineering language like APL to perform the same kinds of tasks as a list processing language like SNOBOL.

A New Language?

The first thing, then, is to define the purpose of the language. Let us not, like the designers of PL/I and ADA, decide that we are going to design *the* general, all purpose, for-all-time language. That is not possible, since we have no way of knowing today what need we'll have for such a language tomorrow. Let us instead be less ambitious. Let us concentrate on generality rather than all-inclusiveness.

Currently, the most popular alternative to BASIC is Pascal, which claims to be universal, structured, and powerful. Its universality is weakened by the fact that there are at least five, more or less incompatible, dialects on the market, and more are likely to come. It is structured, forcing good programming habits on the programmer, according to the definitions of the designers of the language, which may not be compatible with the philosophy of the individual programmer. Its power is hampered by a reported weakness in its I/O capability. Still, it is taught in college computer classes, and is widely available on the market for most microcomputers.

To be fair, FORTH, LOGO, and C all suffer from different weaknesses which are equally disqualifying as a truly suitable alternative to BASIC. If we are to have such an alternative, it should be less specialized than any of those other languages. That is, it should be a general, all-purpose language. It should be designed specifically, not just allowed to develop randomly as BASIC has, and should probably have the following features:

"While our general, all purpose, alternative to BASIC should be designed, it should also have the flexibility to grow into new needs and uses as time goes on."

*While our general, all purpose, alternative to BASIC should be designed, it should also have the flexibility to grow into new needs and uses as time goes on. As the general course of computer use evolves, the language should be able to evolve with it, rather than having to be replaced. It should be built on a common kernal, which interfaces the machine and the compiler/interpreter, so that different versions can run on the same machine, and programs written on one machine can be run on another without translation by hand. (FORTH, C, and Pascal are trying to do this, and succeeding to various extents.)

*It should be easy to learn. The student should not have to assimilate the entire language in one lump before being able to program with it. LOGO is like that. Given three or four commands, one can *do* things with LOGO, right now. This encourages further learning. The elements of the language should be linked together in such a way that one leads to the other, naturally.

*It should be easy to use. Questions of format, file management, command and function use should be answered by the compiler/interpreter itself. For example, old FORTRAN requires the programmer to specify every detail of how a print-out will be typed on the page. BASIC assumes much of that task, though it allows the programmer to change the defaults. In other words, don't make the programmer do it if the computer can.

*It should be interactive, which means two things: 1) the programs allow the user to enter data during the run of a program, as BASIC and LOGO do; 2) the interpreter/compiler check on syntax and programming errors as they are being entered, not after the program is compiled or run, as do most languages.

*It should be extensible. It should be possible for the user to add commands, functions, and procedures to the language itself. Possible in FORTH and C, this allows the language to be quite general, but permits a user to make the language suit his or her specific needs. The interpreter/compiler should then be able to incorporate the definitions of such extensions into the program itself, so that it could be run on any other machine which also uses the language.

Our ideal alternative to BASIC will not perform a specific set of complicated tasks. Rather, it will do a large number of elementary tasks easily and quickly, as FORTH and LOGO do, saving the specialized problems for specialized languages. Most of us, after all, do not write data base management systems, solve high order equations of the trajectory of the moon, or investigate the depths of artificial intelligence. Tasks of this sort do need specialized languages, and cannot really be done with BASIC or any reasonable alternative.

An analogy is the personal automobile. It is a general, all-purpose machine. It does not carry many people. It moves pianos or firewood with difficulty. It isn't all that fast, can't float or fly. It can certainly stand improvement, as our day-to-day needs for transportation change. We hire special vehicles for these special purposes, for which an automobile is unsuited. But the car serves us for ninety percent of our around-the-town use. Our ideal alternative to BASIC should be like the ideal car, neither a formula racer nor a dump truck.

Ease of use is not one of the things found in most of the currently available alternatives to BASIC. Pascal *forces* you to write "good code," according to someone else's definition of the term. FORTH is backwards and not at all human-like. C is mid-level, strictly mathematical, and large.

LOGO, as an exception, is certainly easy, and has lovely graphics, but I doubt that its sophistication will encourage its use by serious programmers. One hopes, however, that the philosophy of the LOGO designers, that the language should serve the user, not the computer, will become more popular. But LOGO is not an all-purpose, general language — it is another instructional language.

Generality of function and ease of use cannot be achieved in one step. If they could be, the first version of BASIC would have been what the best current version of BASIC is now, and more. It is only by trial and error, constant use and refining, that the "perfect" language can be developed.

Part of that perfection must be, then, the ability to grow as new uses and needs are perceived, and the ability to drop old functions as they are superceded, improved on, or found superfluous. (Many early versions of BASIC had MAT, or matrix statements. Nested FOR statements accomplish the same thing.) Therefore, rather than designing a whole language, or a specific language, what we need is to design our *goals* for a language.

"There are also good reasons to assume that BASIC is now, and will continue to be, the best language of choice for the general public. It is being improved all the time. It is a living language, not a dead one, and that characteristic should be retained. Each other language meets specific requirements. Only BASIC is general enough..."

Why An Alternative?

It seems, at this moment, that BASIC (as a group of dialects, not as a single specific language) is working in this direction. It is extremely general. New functions are being added all the time. Useless ones are being trimmed out. The very existence of multiple versions means that it is not stable (though perhaps it should be given a more stable kernel), which allows growth and change.

The English language grows and changes, and the best all-purpose computer language should do likewise, in my opinion. Users, by their sheer numbers and economic pressure, force these changes on the computer writers and suppliers.

It is certainly worthwhile, however, to continue to create new languages with new features and abilities, because only in that way can worthwhile new aspects of the "perfect common language" be discovered and tried out in the field. It would also be worthwhile to design that "kernel" for BASIC, so that various dialects could be run on the same machine, without the need for translation by hand.

So, in the search for an alternative to BASIC, we arrive at an interesting question — why do we need one? The version of BASIC in the Commodore SuperPet is reportedly the Danish national programming language. A version at Caltech was designed for management use, and is the preferred language for their non-engineers. Yet, as I've indicated above, there are some good reasons for an alternative, having to do with specificity of purpose.

There are also good reaons to assume that BASIC is now, and will continue to be, the best language of choice for the general public. It is being improved all the time. It is a living language, not a dead one, and that characteristic should be retained. Each other language meets specific requirements. Only BASIC is general enough that it has no specific requirements.

You don't talk math with your landlord, or Latin with the local cop. You use English. BASIC is, in may ways, the equivalent of English among computer languages. Like English, it can be used badly, deceitfully, or unintelligibly. But that is the fault of the user, not the language. Like English, it will grow. If we are intelligent and educated, we can help it, simply by using it.

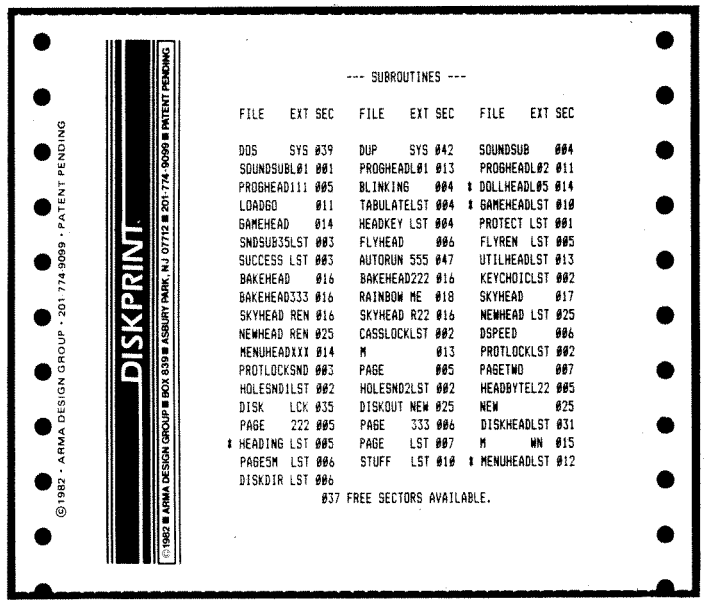
The Arma Design Group Proudly Announces

Diskprint®

If you're using a computer with one or more disk drives, have we got a product for you!

DISKPRINT is beyond a doubt the most useful new package to become available since the computer revolution began.

DISKPRINT will forever free you from having to search for or through that looseleaf binder with all your disk directories listed (if you have one!), or having to spend valuable minutes or hours scanning through each disk directory on-screen looking for that one particular program. DISKPRINT does for you what your computer was supposed to do for you, it eliminates the drudgery of paper shuffling and wasted time.



HERE'S HOW DISKPRINT WORKS...

Simply insert DISKPRINT into one of your disk drives and power up the system. DISKPRINT is self-booting and will load and run automatically. Now load our DISKPRINT custom forms into your printer and switch it on. Insert any disk in your library into your drive and answer the two prompts (which drive and disk title). If you do not choose to title your disk, DISKPRINT will default to its own. Press (RETURN) and DISKPRINT will print out a listing of every program on the disk along with extensions and sectors used as well as the number of free sectors available. This whole process will take about 20 or 30 seconds of your valuable time, so you can run through a one-hundred disk library in about 45 minutes or so. After your form has been printed, advance the next form to the print bar, insert another disc and go again. When you have run through your library, simply remove the tractor strips from the forms and insert them in the jackets with the disks. ***Every disk in your library will have a printed directory in the jacket with it at all times!***

DISKPRINT is delivered with our program on disk, 50 custom DISKPRINT forms and complete instructions. Refill packs of 50 forms each are available at your local dealer or directly from us by mail. (An order form is enclosed).

DISKPRINT is currently availble for:
• ATARI 400/800 & ATARI 825 or EPSON MX Series Printers • IBM-PC & IBM or EPSON MX Series Printers • APPLE II/APPLE II+ & EPSON MX Series Printers • TRS 80 Models I/III & TRS Series Printers • CMB 4032/8032 & CMB/EPSON MX Series Printers

DISKPRINT is now being converted to run on many more makes and models of computers. Call or write for further information.

COMPLETE DISKPRINT KIT FOR ALL MACHINES $24.95
Refill Packs of 50 Custom Forms $8.95
Please add $1.50 per order for P&H

ARMA DESIGN GROUP

"We make the computer revolution a little less revolting." ©

ARMA DESIGN GROUP • P.O. BOX 839 • DEPT SS1182 • ASBURY PARK, N.J. 07712 • 201-774-9099

NAME _____

ADDRESS _____

CITY STATE ZIP _____

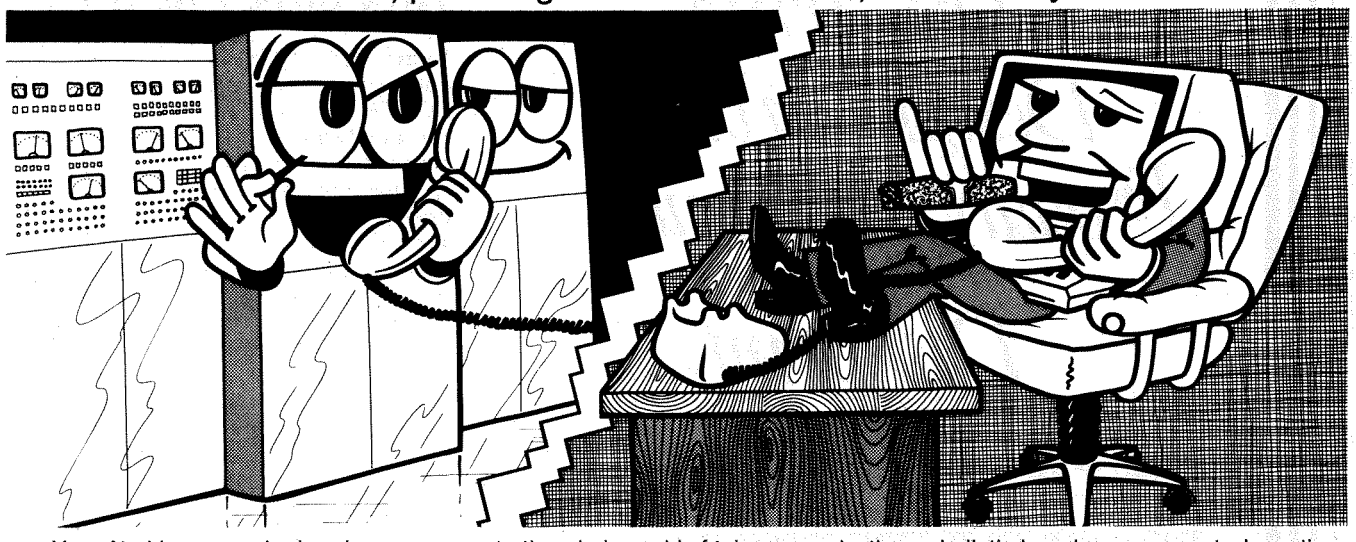
TELEPHONE _____ **COMPUTER SYSTEM:** _____

☐ **MASTERCARD** ☐ **VISA** ☐ **MONEY ORDER** ☐ **CHECK** • **INTERBANK# (M.C.)** _____

CREDIT CARD # _____

DON'T ASK PROVIDES THE MISSING LINKS

⟷ the link between your modem and the outside world. For hassle-free communications, phone right in with TELETARI, The Friendly Terminal.

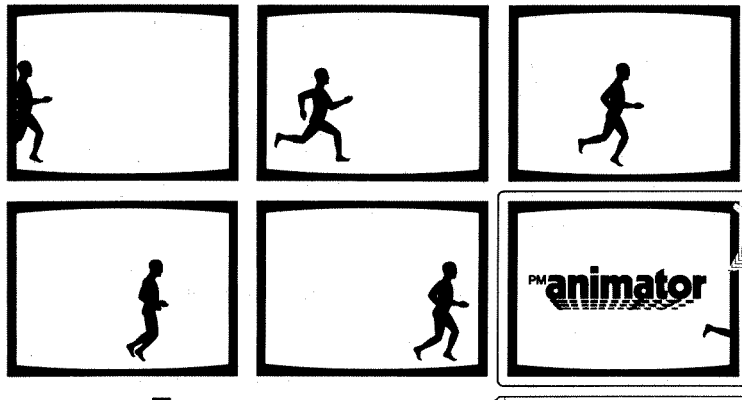


Your Atari has never had such easy access to the whole world of telecommunications – bulletin boards, news reports, large time-sharing computers, the works. Now it's a snap to tap into all these, and it's just as easy to transfer your program or text files to and from a remote computer. Meet TELETARI, The Friendly Terminal. It's just what your modem needs: a powerful, adaptable telecommunications package that's a cinch to use. With TELETARI, you simply choose the desired communications function from a menu. Commonly used terminal parameters are included in the program, but you can change them to suit your needs with a couple of keystrokes, using another handy menu, and store the ones you plan to use again. TELETARI's generous buffer stores up to 20K, so you can review, print, or save received information long after you've hung up the phone. You never knew using a modem could be so convenient. Because it's very flexible, TELETARI is compatible with most modems and a wide variety of computers. And because it works through the RS 232 port, TELETARI is not limited to modem/telephone uses. Put it to work in any RS232 application your imagination can devise – even operating a laser disk!

- buffer of up to 20K
- menu-driven
- highly adaptable
- supports all 850 options
- compatible with 1200 baud modems and BiT 3 Full-view 80™ board
- suitable for any RS232 application

$39.95 Requires Basic, 32K RAM, disk, 850 Interface

⟷ the link between BASIC and arcade-style graphics. Draw and animate pictures for your own BASIC games and other programs with pm ANIMATOR. Create running men, flying rockets, moving figures of all kinds.

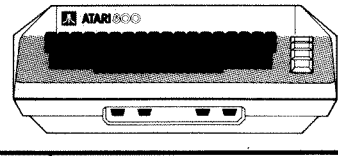


Coming soon from DON'T ASK.

BASIC programmers, pm ANIMATOR puts the power of Player-Missile Graphics at your fingertips.

$34.95 Requires 32K RAM, BASIC, disk.

PM animator



To order direct from Don't Ask, send a check or money order, or call to order COD. Add $2.00 for shipping and handling. California residents add 6% sales tax (6.5% if you reside in L.A. County).

⟷ the link between fast game action and verbal learning:

WORDRACE

Kids and adults, increase your vocabulary while you compete in this exciting word game.
Disk version:
3 levels of play – Beginner, Regular, Challenge
Requires 32K RAM, disk, BASIC. $24.95
Cassette version:
2 levels of play – Beginner, Intermediate
Requires 16K RAM, cassette, BASIC. $19.95
turn WORDRACE into a history game or a famous athletes game, and get more vocabulary words, with the WORDRACE accessory disk: CLAIM TO FAME/SPORTS DERBY. 3 new games in all.
Disk only. Requires WORDRACE disk. $19.95

Atari is a trademark of Atari, Inc. Full-view 80 is a trademark of BiT 3 Computer Corporation.

⟷ the link between you and what your Atari is really thinking:

ABUSE

the insult-exchange program.
Have you cursed out your computer? Now it can understand you and answer back!
Requires 40K RAM, BASIC, disk. $19.95
Release your aggressions! Inflict ABUSE on anyone who's got it coming!

DON'T ASK
⟷ the link between technical excellence and the fun of computing. Why do we give you so much? Don't Ask.

DON'T ASK COMPUTER SOFTWARE INC.

2265 Westwood Bl., Ste. B-150
Los Angeles, CA 90064
(213) 475-4583 or 397-8811

GENERAL INFORMATION

Concerning SoftSide line listings, SWAT & Magnetic Media

Follow these procedures unless otherwise instructed by the documentation in the magazine. Back issues may differ in some details.

SWAT TABLES

At the conclusion of each line listing of a ***SoftSide*** program, we include a *SWAT* (*Strategic Weapon Against Typos*) Table. *SWAT* was published in issue #30 of ***SoftSide*** and is available as a free reprint. Please send a self-addressed, stamped envelope to ***SoftSide*** Publications, Inc., Dept. *SWAT*, 6 South Street, Milford, NH 03055.

APPLE™

Disks are in 16-sector format, created under DOS 3.3. To use, just boot the disk. A cover/menu program will run automatically.

Tapes LOAD in the normal manner. Advance the tape to the beginning of the lead-in tone; stop the tape; insert the plug into the EAR jack; type LOAD; start the tape; and press RETURN. Side two of the tape is a duplicate of side one, unless one or more Integer BASIC programs are included, in which case side two contains the Integer programs.

ATARI®

Line Listings use the following conventions in representing unprintable characters, unless otherwise noted:

Characters (including blank spaces) which are underlined should be typed in inverse video.

When graphics or control characters are to be included in a string (between quotation marks), it will be noted in a nearby REMark. In such cases, graphics characters are represented by the corresponding lower-case letter, and control characters are represented by the corresponding unshifted key symbol. For example: The lower-case letter s represents a graphics cross, entered by holding down the CTRL key and then pressing the S key. The symbol = represents a control-down-arrow, entered by first pressing and releasing the ESC key, then holding down the CTRL key and pressing the = key. (See Appendix F, and the back cover, of the *ATARI® BASIC Reference Manual.)*

The one exception to the above practice is that a clear-screen character (ESC CTRL-↰) is represented in listings by a right-hand brace, which looks like this: }

A shifted = is represented in the listings by a vertical line with a small gap in it: ¦

SWAT — Before appending *SWAT* to a program in memory, the program to be SWATed must first be LISTed to disk or cassette (using LIST "D:FILENAME" for disk or LIST "C:" for tape). Next, turn the computer off, then on again, to clear the system and ENTER the program back into memory (using ENTER "D:filename" for disk or ENTER "C:" for tape). Because of the unique method in which *ATARI® BASIC* stores variables in a program, the variable table must always be in the same order to produce accurate *SWAT* codes. LISTing and ENTERing the program is the only known way to rebuild the variable table in a specific order so that *SWAT* codes can match.

Disks do not contain DOS.SYS files, and are therefore not bootable by themselves. First boot a disk which contains any version of DOS, then insert the ***SoftSide*** disk and RUN "D:COVER" (*Adventure of the Month* — RUN "D:INTRO").

Tapes CLOAD in the normal manner. If you have difficulty, try this procedure:
(1) Type POKE 54018,54 and press RETURN.
(2) Turn up the volume on your TV.
(3) Type CLOAD and press RETURN once.
(4) Press the PLAY button and listen.
(5) When you hear a steady lead-in tone, press RETURN again.
Side two of the tape is a duplicate of side one.

IBM® PC

DV is available on individual order. There is no CV at this time.

TRS-80®

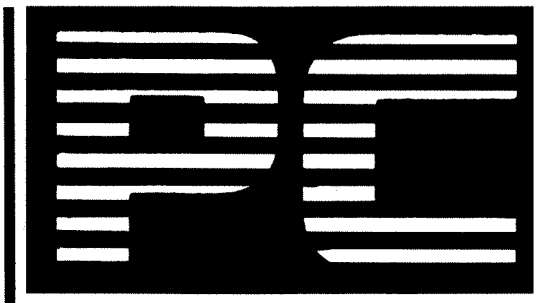
Disks are available in Model I or Model III format. They contain the DOS PLUS operating system, and a cover program which automatically runs upon booting. Back issues prior to May, 1982, are available only in Model I format, and may be converted using the TRSDOS CONVERT utility on a two-drive Model III. Older back issues (with Model I TRSDOS) require you to enter BASIC and then type RUN "COVER".

Tapes CLOAD in the normal manner on Model I's, and at low speed (500 baud) on Model III's. The first program is a cover/menu program. Side two of the tape is a duplicate of side one.

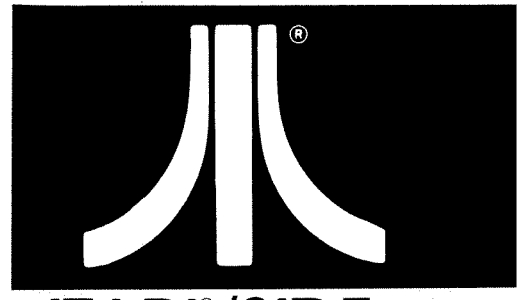
NOTES ABOUT MAGNETIC MEDIA

SoftSide disks and tapes are duplicated by reliable, professional duplication services; bad copies are very rare. However, the trip through the mail occasionally wreaks havoc with sensitive magnetic media. If, after a reasonable number of tries and a careful check and cleaning of your equipment, you are not able to load a program from a tape or disk, please return it to us with an exact description of the problem. If we cannot duplicate the problem on our systems, we will advise you when we send the replacement copy.

We use no copy-protection on our media. We urge you to make a backup copy of every disk or tape as soon as you receive it (and at the same time resist the urge to give copies to friends). Our replacement policy does not extend beyond 30 days.



PC/SIDE page 32



ATARI®/SIDE page 49

APPLE™/SIDE page 72

TRS-80®/SIDE page 100

DATA BASE

by Mark Pelczarski
IBM® PC version by Fred Condo

"The formatter in this Data Base may be one of the most versatile ones around. Although there are still a few things it can't do, it does a lot that many 'professional' databases don't allow."

***Data Base* is a data-management program for an IBM PC with a disk drive. A printer is highly desirable.**

Running the *DATA BASE*

The first choice you will be presented with is whether to have forty or eighty columns on your display. It is suggested that you use eighty columns unless that format is too difficult to read on your monitor.

Next, you must decide whether to initialize a new file or load an existing file. Specify this choice by pressing "L" or "I". The first time you use it, you'll have to initialize a file. Thereafter, when you want to access that data, you will load the file. Any time you want to create a new file with a different type of data, use the initialize option. Several different files will fit on a disk, and you can use as many different data disks as you like. A few examples of files are: a mailing list, (name, address, city, etc.), checkbook list, (to whom, withdrawals, deposits...), and an inventory list (stock number, description number, in stock, on order, etc.) Whatever records you want to keep can usually be stored in this type of database format.

To initialize a new file, give your file a name. This name must be a standard PC DOS file name, *without* an extension. (The program will not accept a name containing an extension.) All data files used with the *Data Base* are assigned the extension ".DAT". Then tell the computer how many headings you want and their names. An example would be a file named "Address," with six headings: Name, Street Address, City, State, Zip Code, and Phone Number. You might want to add an extra heading (or more) for some kind of code. You might use "Computer" for your seventh heading, so you would know what kind of computer a particular person owns.

Your data will be organized into a table. The headings will be your column headings, and each row will have one set of information across those headings. A set of such information is called a record. Once a file has been created, any time that you use the database you only have to give the file name ("Address" is our example) and all of the information will automatically be loaded from the disk.

To load a file, press "L" rather than "I". You will then see a list of all the data files on your diskette. Type the name (omit the extension) of the file you wish to load.

The Main Options

After initialization or loading, you will be given a list of choices for manipulating your database. Here are the choices:

(S) SAVE current data
(P) PRINT data [to screen or printer]
(A) ADD a record
(C) CHANGE a record [such as an address change]
(D) DELETE a record
(T) SORT data
(F) FILES [diskette directory]
(N) NEW data file [equivalent to quitting and re-running program]

(Q) QUIT
(M) MEMORY left [approximate room left for more data]
(L) LITTLE (Compressed) print [for the IBM or Epson printer]
(B) BACK TO Standard print

Adding A Record

This is your logical first choice, since, with no data in memory, the other options aren't too much fun. Choose (A) from the options page and you'll be asked for information to fill each of your headings for one record. After you've filled one record, you'll be returned to the options page. See the note below about searching and sorting numeric fields, if you plan to do such.

Printing A Record

First, a few words about the printer options: (L) will set your IBM or Epson MX-80 for the compressed mode. (B) will set it for the standard print mode.

To see if your data is really there, type "P" to print your record. The program will ask if you want it put in a special format (S), or default format (D). Choose (D) for the moment. After choosing, you'll be asked if you want it on the screen (S) or the printer (P). If you choose the printer, you will also be asked to specify the number of copies of each record you wish to have printed. This is particularly useful for the generation of multiple mailing labels. Then, after that choice, a list of headings will be displayed, followed by the choices "BEGIN" and "Return to Menu." Choose the number next to the word "BEGIN" and press RETURN. Each record that you have in memory will be displayed in sequence. If you're printing them to the screen, pressing any key advances to the next record. The ESC key returns you to the option page. All the other choices mentioned above will be explained under "searching" and "formatting."

Searching

When printing, changing, or deleting records, you have the choice of selecting individual items, subsets of your data, or the entire set of data. This is done through the search routine. When you used the print routine above, you chose to print all of the data by selecting "BEGIN" before any other choice. Each of the headings is also listed at that point, along with "Record Number." By choosing the number next to any of the headings or "Record Number," you elect to do a search under that heading. You are then asked if you want to look for an item that is less than or equal, equal, or greater than or equal, to a value you'll give. After choosing 1, 2, or 3, respectively, you'll be asked for a value for comparison. Example: If you want to search for all records with names starting with A through G, you want NAME, < ,G, where "G" is the value used for comparison. If you want all records from number 20 through the end of the file, you would choose RECORD NUMBER, > , 20.

You also have the option of specifying the beginning of a value for comparison. If you wanted all records from people whose zip code starts with a "60" (as 60185), you can specify ZIP CODE, =,60*. The asterisk says that anything may follow. This is also an easy way to find records knowing exact information. If you can't spell "Pelczarski" (or if you don't like typing) you can try "Pel*" and you'll find the record.

There is also a substring search option. This option works *only* when you select = (choice 2). To use this option, *begin* your specification with an asterisk. Using the example above, you might specify "*cz". This would cause the program to display the records for all people whose names contain "cz" *at any position.*

This feature allows you to have a *single* heading, called, for example, "Category," which can be searched for several different criteria.

For instance, a person's category might look like "work/Christmas/BD Feb 12". This would mean that this is someone from work, to whom you send a Christmas card each year, and whose birthday is on February 12. So each Christmas, you would choose "Category" as the heading to search, choose the = option, and specify "*Christmas" as the search key. Note that your category lines need not be in a rigid format. The search just described would work just as well for a category line that looks like "Christmas/BD Feb 12/work". Also note that the slashes (/) are optional, but do make the category line more legible.

To start the actual search, you must choose "BEGIN." A hidden option here is that you can specify several search criteria. You might, for example, want to find everyone in your list whose zip code starts with a "60" and who owns a PC. You would specify ZIP CODE, =,60*, then specify COMPUTER, =,PC, and tell it to begin. The program will ask if the item must meet *all* of the conditions, or *any* of the conditions. "All" would find only those with zip 60 who also own a PC. "Any" would find everyone with zip 60, plus everyone owning a PC (technically, everyone with zip 60, *or* owning a PC). Up to eight such criteria may be specified, so you may look for everyone whose name starts with D through F (>D and < F), whose zip starts with 9, and who owns an Apple and a PC, etc.

Changing Records

To change a record, choose (C) from the options page. After specifying whatever search criteria you want, the appropriate record(s) will be shown on the screen. The items under each heading will then be shown in sequence, and the program will wait for you to type "K" to keep, "C" to change, or "R" if the remainder of the record is okay. If you type "C", you'll be asked for the information with which to replace the old item.

Deleting Records

After choosing (D) from the options page and going through the search steps, you must specify whether you want to be prompted to verify deletions. If you choose no prompting, then the record(s) in question will be displayed, and will automatically be deleted. If you do choose prompting, you'll be asked to verify that you want each particular record deleted. Type "Y" to delete. Once it's deleted, it's gone, so don't omit prompting unless you're *sure* of what you're doing.

Saving a File

When you want to sign off for the day (or even for a minute), typing (S) from the options page saves your current file on disk. It's an excellent idea, especially with im-

portant information, to do this twice using two separate disks, with one as a backup.

Sorting

The (T) option from the menu allows your items to be sorted in ascending or descending order, under any heading. Alphabetic items are sorted alphabetically, and numeric items are sorted as strings. The latter means that numbers don't always sort the way you want. 125, 34, and 7 will come out in that order because numbers are sorted according to the ASCII code of the first character in each. To get a true numeric sort, add leading zeros to the maximum number of places, such as 007, 034, and 125. This will force proper sequencing.

Files

You can get a directory directly from the options page by typing (F).

Switching Data Files

You can load or create a new file without rerunning the program by selecting (N) from the options. Be sure to verify that the current file has been saved.

Formatting Output

The formatter in this *Data Base* may be one of the most versatile ones around. Although there are still a few things it can't do, it does a lot that many "professional" databases don't allow. You can specify the exact form in which you want each record printed. Each record is printed in sequence, meaning that you cannot mix records across a page. You can specify which headings are to be printed and where, which items are to be printed and where, and what (if any) additional character strings should be printed on the form. You may want to include your company name, an expanded version of a heading instead of the heading itself, or just some lines to separate items.

To create a format, choose the special format option when printing. You'll be asked if you want to load or create one. The first time, you'll have to create it. Draw out exactly what you want printed for your form. You'll be telling the computer, line by line, what it looks like. Your choices are (1) Heading, (2) Item, (3) Tab, (4) Next line, (5) String, and (6) End. Here's one example using the "Address" file I mentioned earlier. The format will print mailing labels like this:

Mark Pelczarski
1206 Kings Circle
West Chicago, IL 60185

Here are the format commands (Numerically, my headings are 1 Name, 2 Address, 3 City, 4 State, 5 Zip, 6 Phone, 7 Computer):

Commands	What To Type
Item, Name	2,1
Next Line, 1	4,1
Item, Address	2,2
Next Line, 1	4,1
Item, City	2,3
Tab, 16	3,16
Item, State	2,4
Next Line, 1	4,1
Tab, 12	3,12
Item, Zip	2,5
Next Line, 3	4,3
End	6

The "1" after the next line means to skip down one line. The "3" at the end skips down three lines before printing the next label. Note that none of the actual headings are used in this format, and neither is the phone number.

Another example is a format that will print a separate little form for each person in the database. For lack of a better example, I'll have the following printed:

```
----------
THE FOLLOWING PERSON OWNS A
                                PC

NAME JOE TATE
PHONE 555-1212

----------
```

Here's the format to do it:

```
String,----------
Next line, 1
String, THE FOLLOWING PERSON OWNS A
Next Line, 2
Tab, 9
Item, Computer
Next Line, 2
Heading, Name (Type "1" for hdg. #)
Tab, 7
Item, Name (Type "1" for item #)
Next Line, 1
Heading, Phone (hdg. #6)
Tab 7
Item, Phone
End
```

I'll let the top line of the next item to be printed be the bottom line for the last, so I can just end the format after printing the last item.

That's all there is to formatting. Play around with it a little to see what it does for you. After a format is created, you'll be asked to name it, and it will automatically be saved to disk. In the future, you'll be able to load it back in when you need it. The names for formats work the same way as the names for data files, except that the format files all have the extension ".FMT".

One last note on formatting: if you wish to intercalate a string into your format (option 5), and this string has leading and/or trailing spaces or consists entirely of spaces, then type it between quotation marks. For example, " ! ".

A Few Final Words

There are a lot of things this *Data Base* program still cannot do, but it is a good introduction for those of you who don't know all of what a database program can be used for. As yet, it has no real numeric capability; it doesn't take advantage of disk capabilities; and some of the routines are slow. On the plus side, it has a lot of the features you should look for in a database, and if you ever decide to shop for one (they're expensive), you'll have an idea of the features to consider. I'm amazed that there are $200 database programs out there that don't even have basic sorting functions; and most only have a rather primitive print formatting.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS        IBM PC BASIC         SS
SS        'Data  Base'         SS
SS   Author: Mark Pelczarski   SS
SS   Translation: Fred Condo   SS
SS     Copyright (c) 1982      SS
SS SoftSide Publications, Inc  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type this program, it is also included in this month's SoftSide DV.

Initialization.

```
25 KEY OFF:TROFF:SCREEN 0,0,0:WIDTH 40:H
EREAGAIN=0
30 DEF FNU$(A$)=CHR$(ASC(A$)+32*(A$>="a"
 AND A$<="z"))
35 FOR K=1 TO 10:KEY K,"":NEXT K
40 DEF FNS$(A$)=MID$(A$,1-(VAL(A$)>=0))
42 IF HEREAGAIN THEN 101
45 WIDTH 40:CLS
50 READ A$:IF A$<>"#" THEN LOCATE VAL(A$
),20-(LEN(A$)-1+(VAL(A$)>9))/2,0:PRINT R
IGHT$(A$,LEN(A$)-LEN(FNS$(STR$(VAL(A$)))
)):FOR DELAY=1 TO 700:NEXT DELAY:GOTO 50
 ELSE FOR DELAY=1 TO 1500:NEXT DELAY
60 X$="":CLS:WHILE X$<>"E" AND X$<>"F" A
ND X$<>"e" AND X$<>"f":LOCATE 12,1,1:PRI
NT"Press F or E for WIDTH Forty or Eight
y.";:X$=INPUT$(1):WEND:X$=FNU$(X$):IF X$
="E" THEN WIDTH 80 ELSE WIDTH 40
```

Change the value of MAX to change the maximum number of records allowed.

```
101 MAX=300
```

More initialization.

```
103 CPF=2
105 DIM C$(7),C1%(7),C2%(7),F$(5)
107 LITTLE=0:REM PRINT SIZE
110 CLOSE:ON ERROR GOTO 0
112 OPEN "SCRN:" FOR OUTPUT AS #2
113 OPEN "LPT1:" FOR OUTPUT AS #3
```

Load/initialize option.

```
115 CLS:LOCATE 1,1,1,0,7:PRINT"(I) Initi
alize a new data set"
120 PRINT"(L) Load a previously saved da
ta set"
130 A$=FNU$(INPUT$(1)):PRINT A$
140 IF A$="L" THEN GOSUB 1000:GOTO 200
150 IF A$="I" THEN GOSUB 1500:GOTO 200
160 GOTO 110
```

Lines 200-410: Main program loop.

```
200 CPF=2:QMAX=1:ON ERROR GOTO 0
```

Display main menu.

```
210 CLS:PRINT"(S) SAVE current data"
220 PRINT"(P) PRINT data"
230 PRINT"(A) ADD a record"
240 PRINT"(C) CHANGE a record"
250 PRINT"(D) DELETE a record"
260 PRINT"(T) SORT data"
270 PRINT"(F) FILES"
280 PRINT"(N) NEW data file"
290 PRINT"(Q) QUIT"
292 PRINT"(M) MEMORY left"
293 PRINT
294 PRINT"'";NF$;"' contains";NI+1;"reco
rd";:IF NI=0 THEN PRINT"." ELSE PRINT"s.
"
295 PRINT:PRINT"Printer Commands:"
296 PRINT" (L) LITTLE (Compressed) print
"
297 PRINT" (B) BACK TO Standard print"
298 LOCATE ,6,1:COLOR 0,7:IF LITTLE THEN
 PRINT"Compressed print";
299 IF NOT LITTLE THEN PRINT"Standard pr
int";
300 COLOR 7,0:PRINT" is on."
```

Accept and act on commands.

```
301 A$=FNU$(INPUT$(1)):PRINT A$:PRINT
303 IF A$="L" THEN GOSUB 20000:GOTO 200
305 IF A$="B" THEN GOSUB 30000:GOTO 200
310 IF A$="M" THEN GOSUB 50000
320 IF A$="S" THEN GOSUB 2000:GOTO 200
330 IF A$="P" THEN GOSUB 3000:GOTO 200
340 IF A$="A" THEN GOSUB 4000:GOTO 200
350 IF A$="C" THEN SB=3:GOSUB 8000:GOTO
200
360 IF A$="D" THEN GOSUB 31000:SB=4:FS=1
:GOSUB 8000:GOTO 200
370 IF A$="T" THEN GOSUB 7000:GOTO 200
380 IF A$="F" THEN GOSUB 600:GOTO 200
400 IF A$="Q" OR A$="N" THEN 500
410 GOTO 200
```

Quit.

```
500 IF SS=1 THEN 540
510 PRINT"Current file is not saved.":PR
INT"Do you still want to quit? (Y/N) ";:
T$=FNU$(INPUT$(1)):PRINT T$
520 IF T$="N" THEN 200
530 IF T$<>"Y" THEN 510
540 IF A$="N" THEN CLEAR:HEREAGAIN=-1:GO
TO 30
550 WIDTH 80:CLOSE:GOSUB 61000:END
```

Display diskette directory.

```
600 FILES:PRINT:PRINT"Hit a key.";:LOCAT
E ,,1:A$=INPUT$(1):RETURN
```

Load a data file.

```
999 REM LOAD SUBROUTINE VERS.1
1000 PRINT:ON ERROR GOTO 1002:FILES "*.d
at":PRINT:LINE INPUT"File name? ";F$:IF
INSTR(F$,".")<>0 THEN 1000 ELSE UP$=F$:G
OSUB 51000:F$=UP$:GOTO 1005
1002 PRINT:PRINT"No data files on this d
iskette.":PRINT"Press any key.";:LOCATE
,,1:A$=INPUT$(1):RESUME 110
1005 NF$=F$
1010 ON ERROR GOTO 1310
1015 OPEN F$+".DAT" FOR INPUT AS #1:CLOS
E #1:REM Verify presence of file.
1020 OPEN F$+".DAT" FOR INPUT AS #1
1040 INPUT #1,NH:INPUT #1,NI
1130 DIM H$(NH),I$(MAX,NH)
1140 FOR I=0 TO NH:LINE INPUT #1,H$(I):N
EXT
1200 IF NI=-1 THEN 1280
1240 FOR I=0 TO NI
1250 FOR J=0 TO NH
1260 LINE INPUT #1,I$(I,J)
1270 NEXT J:NEXT I
1280 CLOSE #1
1300 SS=1:RETURN
1310 PRINT"File not found.":PRINT"Press
any key.";:LOCATE ,,1:A$=INPUT$(1)
1320 RESUME 110
```

Initialize a data file.

```
1499 REM INITIALIZE SUBROUTINE VERS.1
1500 LINE INPUT"Give your file a name: "
;F$:IF INSTR(F$,".")<>0 THEN 1500 ELSE U
P$=F$:GOSUB 51000:F$=UP$
1510 IF F$="" THEN 1500
1515 NF$=F$
1520 INPUT"How many headings";NH
1530 IF NH<1 THEN 1520
1540 NH=NH-1:NI=-1
1560 DIM H$(NH),I$(MAX,NH)
1570 FOR I=0 TO NH
1580 PRINT"Heading #";I+1;:LINE INPUT" :
 ";H$(I)
1590 NEXT I
1600 SS=0:RETURN
```

Save a data file on diskette.

```
1999 REM SAVE SUBROUTINE
2000 PRINT"Use '";F$;"' as name (Y/N)?";
:A$=FNU$(INPUT$(1)):PRINT A$
2050 IF A$="Y" THEN 2090
2060 IF A$<>"N" THEN 2000
2070 LINE INPUT"Name? ";F$:UP$=F$:GOSUB
51000:F$=UP$:NF$=F$
```

```
2080 IF F$="" THEN 2070
2090 ON ERROR GOTO 2290
2100 OPEN F$+".DAT" FOR OUTPUT AS #1:CLO
SE #1:KILL F$+".DAT"
2107 OPEN F$+".DAT" FOR OUTPUT AS #1
2120 PRINT #1,NH:PRINT #1,NI
2130 FOR I=0 TO NH
2140 PRINT #1,H$(I)
2150 NEXT
2220 IF NI=-1 THEN 2270
2230 FOR I=0 TO NI
2240 FOR J=0 TO NH
2250 PRINT #1,I$(I,J)
2260 NEXT J:NEXT I
2270 CLOSE #1
2280 SS=1:RETURN
2290 PRINT"Disk error. Hit any key.":A$=
INPUT$(1):RESUME 200
```

Print records.

```
2999 REM PRINT SUBROUTINE
3000 IF NI=-1 THEN GOSUB 9000:RETURN
3005 PRINT"(S) SELECT format, or (D) DEF
AULT ";:A$=FNU$(INPUT$(1)):PRINT A$
3006 IF A$="S" THEN GOSUB 10000:FS=2:GOT
O 3010
3007 IF A$<>"D" THEN 3005
3008 FS=1
3010 PRINT"(S) SCREEN, or (P) PRINTER ";
:LOCATE ,,1:A$=FNU$(INPUT$(1)):PRINT A$
3020 IF A$="P" THEN SB=2:GOTO 3050
3030 IF A$<>"S" THEN 3010
3040 SB=1:PRINT:PRINT"After each record,
 [ESC] will return to":PRINT"the menu. A
ny other key continues."
3050 PRINT"Press any key.":LOCATE ,,1:A$
=INPUT$(1):IF SB=2 THEN INPUT"How many c
opies";QMAX
3060 IF SB<>2 THEN QMAX=1
3070 GOSUB 8010
3090 IF SB=2 THEN CPF=2
3100 RETURN
3299 REM PRINT ONE RECORD TO SCREEN, VER
S.4
3300 ON FS GOSUB 3700,3800
3310 IF SB=2 THEN 3350
3340 IF SB<>4 THEN LOCATE ,,1:A$=INPUT$(
1):PRINT:IF A$=CHR$(27) THEN RS=1
3350 RETURN
3699 REM PRINT ONE DEFAULT V.1
3700 IF NOT LITTLE AND SB=2 THEN CPF=3:P
RINT #CPF,CHR$(18);
3705 IF LITTLE AND SB=2 THEN CPF=3:PRINT
 #CPF,CHR$(15);
3709 FOR QT=1 TO QMAX:PRINT #CPF,:PRINT
#CPF,"Record";I+1:PRINT #CPF,
3710 FOR J=0 TO NH
3720 PRINT #CPF,H$(J),I$(I,J)
3730 NEXT J:NEXT QT
3740 RETURN
3799 REM PRINT ONE FORMAT V.1
3800 IF LITTLE AND SB=2 THEN CPF=3:PRINT
 #CPF,CHR$(15);
3805 IF NOT LITTLE AND SB=2 THEN CPF=3:P
RINT #CPF,CHR$(18);
3810 FOR QT=1 TO QMAX:J=1:T=0:B$=""
3820 J1=VAL(MID$(F$(T),J,1)):J=J+1
3830 IF J1<5 THEN N=VAL(MID$(F$(T),J,2))
:J=J+2
3840 ON J1 GOTO 3850,3860,3870,3890,3910
,3970
3850 A$=H$(N):GOTO 3950
3860 A$=I$(I,N):GOTO 3950
3870 B$=LEFT$(B$,N-1):IF LEN(B$)<N-1 THE
N FOR J2=LEN(B$) TO N-2:B$=B$+" ":NEXT
3880 GOTO 3960
3890 PRINT #CPF,B$:IF N>1 THEN FOR J2=2
TO N:PRINT #CPF,:NEXT
3900 B$="":GOTO 3960
3910 IF J>LEN(F$(T)) THEN T=T+1:J=1
3920 J2=J
3930 IF MID$(F$(T),J2,1)<>"!" THEN J2=J2
+1:GOTO 3930
3940 A$=MID$(F$(T),J,J2-J):J=J2+1
3950 B$=B$+A$
3960 IF J>LEN(F$(T)) THEN T=T+1:J=1
3965 GOTO 3820
3970 PRINT #CPF,B$:NEXT QT:RETURN
```

Add a record.

```
3999 REM ADD SUBROUTINE VERS.2
4000 SS=0:NI=NI+1
4005 PRINT:PRINT"Record";NI+1:PRINT
4010 FOR J=0 TO NH
4020 PRINT H$(J);:LINE INPUT" ";I$(NI,J)
4030 NEXT J
4040 RETURN
```

Change a record.

```
4999 REM CHANGE SUBROUTINE VERS.2
5000 PRINT:PRINT"(C) CHANGE item, (K) KE
EP item, or":PRINT"(R) KEEP remainder of
 record."
5030 PRINT:PRINT"Record";I+1
5040 CS=1:RS=0:FOR J=0 TO NH
5050 PRINT:PRINT H$(J);" : ";I$(I,J);" "
;:LOCATE ,,1
5055 IF RS=1 THEN PRINT:GOTO 5090
5060 A$=FNU$(INPUT$(1)):PRINT"  ";A$:IF
A$<>"C" AND A$<>"K" AND A$<>"R" THEN 506
0
5070 PRINT A$:IF A$="K" THEN 5090
5075 IF A$="R" THEN RS=1:GOTO 5090
5080 PRINT H$(J);:LINE INPUT" : ";I$(I,J
)
5085 CS=0
5090 NEXT J
5095 RS=0
5100 IF CS=0 THEN SS=0
5110 RETURN
```

Delete records.

```
5999 REM DELETE SUBROUTINE VERS.2
6000 PRINT:IF NOT PROMPTING THEN 6100
6005 PRINT"Delete this record? ":LOCATE
,,1
6070 A$=FNU$(INPUT$(1)):IF A$<>"Y" AND A
$<>"N" THEN 6070
6080 PRINT A$:IF A$="N" THEN 6150
6100 FOR I1=I+1 TO NI
6110 FOR J=0 TO NH
6120 I$(I1-1,J)=I$(I1,J)
6130 NEXT J:NEXT I1
6135 FOR J=0 TO NH:I$(NI,J)="":NEXT
6140 NI=NI-1:SS=0:I=I-1
6150 RETURN
```

Sort data.

```
6999 REM SORT SUBROUTINE VERS.1
7000 IF NI=-1 THEN GOSUB 9000:RETURN
7010 PRINT:FOR J=0 TO NH
7020 PRINT"(";FNS$(STR$(J+1));") ";H$(J)
7030 NEXT J
7040 INPUT"Sort on which heading";J1
7045 J1=J1-1
7050 IF J1<0 OR J1>NH THEN RETURN
7060 PRINT"(A) ASCENDING, or (D) DESCEND
ING ":LOCATE ,,1:A$=FNU$(INPUT$(1)):PRIN
T A$
7070 IF A$="A" THEN A=1:GOTO 7100
7080 IF A$="D" THEN A=2:GOTO 7100
7090 GOTO 7060
7100 FOR I=0 TO NI-1
7110 T=I
7120 FOR I1=T+1 TO NI
7122 PRINT I;" ";I1
7125 ON A GOTO 7130,7140
7130 IF I$(I1,J1)<I$(T,J1) THEN T=I1
7135 GOTO 7145
7140 IF I$(I1,J1)>I$(T,J1) THEN T=I1
7145 NEXT I1
7150 IF T=I THEN 7180
7155 FOR J=0 TO NH
7160 T$=I$(T,J):I$(T,J)=I$(I,J):I$(I,J)=
T$
7170 NEXT J
7180 NEXT I
7200 SS=0:RETURN
```

Search subroutine.

```
7999 REM SEARCH SUBROUTINE VERS.2
8000 IF NI=-1 THEN GOSUB 9000:RETURN
```

```
8010 I1=0:I2=NI:J=0:C1%(0)=-1:BS=1
8015 CLS:PRINT"Search criteria:":PRINT
8020 PRINT" 0) Record Number"
8030 FOR I=0 TO NH:PRINT I+1;CHR$(29);")
 ";H$(I):NEXT I
8035 PRINT:PRINT NH+2;") BEGIN"
8036 PRINT NH+3;") Return to Menu"
8040 LOCATE 21,,1:INPUT"Select";I:IF I<0
 OR I>NH+3 THEN 8040
8045 IF I=NH+2 THEN C1%(J)=-1:GOTO 8150
8046 IF I=NH+3 THEN RETURN
8050 C1%(J)=I-1
8060 LOCATE 22,,1:PRINT"(1) Smaller  (2
) Equal  (3) Larger  ";:A$=INPUT$(1):PRI
NT A$:IF A$<"1" OR A$>"3" THEN 8060
8070 C2%(J)=VAL(A$)
8080 LOCATE 23,,1:PRINT"Compared to : ";
:IF C1%(J)=-1 THEN 8100
8090 LINE INPUT" ";C$(J):J=J+1:IF J>7 TH
EN 8160
8095 GOTO 8015
8100 INPUT I:IF I<1 OR I>NI+1 THEN 8100
8105 I=I-1
8110 IF C2%(J)=1 THEN I2=I
8120 IF C2%(J)=2 THEN I1=I:I2=I
8130 IF C2%(J)=3 THEN I1=I
8140 GOTO 8015
8150 IF J<2 THEN 8200
8160 LOCATE 22,,1:PRINT"1) Item must mee
t ALL conditions":PRINT"2) Item may meet
 ANY condition ";:A$=INPUT$(1):PRINT A$:
IF A$<"1" OR A$>"2" THEN 8160
8170 BS=VAL(A$)
8200 RS=0
8250 I=I1-1:FOR I3=I1 TO I2:I=I+1
8255 AS=0:FOR J=0 TO 7
8260 IF C1%(J)=-1 THEN J=7:GOTO 8345
8270 ON C2%(J) GOTO 8280,8290,8310
8280 IF I$(I,C1%(J))<=C$(J) THEN 8330
8285 GOTO 8340
8290 IF I$(I,C1%(J))=C$(J) THEN 8330
8294 IF LEFT$(C$(J),1)="*" THEN 8410
8295 IF RIGHT$(C$(J),1)<>"*" THEN 8340
8298 T=LEN(C$(J))-1:IF LEN(I$(I,C1%(J)))
<T THEN 8340
8302 IF LEFT$(I$(I,C1%(J)),T)=LEFT$(C$(J
),T) THEN 8330
8305 GOTO 8340
8310 IF I$(I,C1%(J))>=C$(J) THEN 8330
8320 GOTO 8340
8330 IF BS=2 THEN AS=1:J=7
8335 GOTO 8345
8340 IF BS=1 THEN AS=2:J=7
8345 NEXT J
8350 IF AS=0 AND BS=1 THEN 8355
8352 IF AS<>1 THEN 8380
8355 GOSUB 3300
8365 IF SB=3 THEN GOSUB 5000
8370 IF SB=4 THEN GOSUB 6000
8375 IF RS=1 THEN I3=I2
8380 NEXT I3
8390 CPF=2:PRINT:PRINT:PRINT"That's all!
  Hit any key.":LOCATE ,,1:A$=INPUT$(1):
PRINT
8400 RETURN
```

Embedded search feature.

```
8401 REM EMBEDDED STRING SEARCH VER. 1 B
Y F. CONDO
8410 T=LEN(C$(J))-1:EM$=RIGHT$(C$(J),T)
8420 E0=LEN(I$(I,C1%(J)))
8430 IF E0<T THEN 8340:REM NO MATCH -- I
TEM TOO SHORT
8440 EE=E0-T+1
8450 FOR Z0=1 TO EE
8460 IF EM$<>MID$(I$(I,C1%(J)),Z0,T) THE
N NEXT Z0
8470 IF Z0=EE+1 THEN 8340:REM NO MATCH
8480 GOTO 8330:REM MATCH
```

Error handler #1.

```
8999 REM ERROR SUBROUTINE #1
```

continued on page 38

continued from page 37

```
9000 PRINT"No data is in memory."
9010 FOR I=1 TO 1000:NEXT:RETURN
```

Print formatting.

```
9999 REM PRINT FORMATTING, V.1
10000 IF F$(0)="" THEN 10040
10010 PRINT"Same format? ";:A$=FNU$(INPU
T$(1)):PRINT A$
10020 IF A$="Y" THEN RETURN
10030 IF A$<>"N" THEN 10010
10040 PRINT"(L) LOAD format, OR (C) CREA
TE format ";:A$=FNU$(INPUT$(1)):PRINT A$
10050 IF A$="C" THEN 10200
10060 IF A$<>"L" THEN 10040
10090 ON ERROR GOTO 10170
10100 PRINT:ON ERROR GOTO 10102:FILES "*
.FMT":PRINT:LINE INPUT"Format name: ";A$
:IF INSTR(A$,".")<>0 THEN 10100 ELSE UP$
=A$:GOSUB 51000:A$=UP$:GOTO 10105
10102 PRINT:PRINT"No format files on thi
s diskette.":PRINT"Press any key.";:LOCA
TE ,,1:A$=INPUT$(1):RESUME 200
10105 OPEN A$+".FMT" FOR INPUT AS #1:CLO
SE #1:REM Simulate Apple's VERIFY
10110 OPEN A$+".FMT" FOR INPUT AS #1
10130 INPUT #1,NF
10140 FOR J=0 TO NF:LINE INPUT #1,F$(J):
NEXT
10150 CLOSE #1
10160 RETURN
10170 IF ERR=24 THEN PRINT"No printer fo
und.":PRINT"Press any key.";:LOCATE ,,1:
A$=INPUT$(1):RESUME 200 ELSE PRINT"Forma
t not found. Press any key.":LOCATE ,,1:
A$=INPUT$(1):RESUME 200
10200 NF=0:J=0:F$(0)=""
10210 CLS:PRINT"Start in the upper left
corner and work across each line."
10220 PRINT"1:Heading, 2:Item, 3:Tab, 4:
Next Line,":PRINT"5:String, 6:End":INPUT
J1
10230 IF J1<1 OR J1>6 THEN 10220
10240 F$(NF)=F$(NF)+FNS$(STR$(J1)):J=J+1
10250 ON J1 GOTO 10260,10260,10300,10300
,10350,10400
10260 FOR T=0 TO NH:PRINT T+1;") ";H$(T)
:NEXT
10270 INPUT"WHICH";T:T=T-1:IF T<0 OR T>N
H THEN 10270
10280 GOTO 10310
10300 INPUT"How many";T:IF T<1 OR T>99 T
HEN PRINT"Out of range.":GOTO 10300
10310 A$=FNS$(STR$(T)):IF T<10 THEN A$="
0"+A$
10320 F$(NF)=F$(NF)+A$:J=J+2
10330 GOTO 10380
10350 LINE INPUT"String: ";A$:IF LEFT$(A
$,1)=CHR$(34) AND RIGHT$(A$,1)=CHR$(34)
THEN A$=MID$(A$,2,LEN(A$)-2)+"!" ELSE A$
=A$+"!"
10360 IF LEN(A$)+J>255 THEN NF=NF+1:J=0:
F$(NF)=""
10370 F$(NF)=F$(NF)+A$:J=J+LEN(A$)
10380 IF J>252 THEN NF=NF+1:J=0:F$(NF)="
"
10390 GOTO 10220
10400 LINE INPUT"Format Name: ";A$:UP$=A
$:GOSUB 51000:A$=UP$
10405 ON ERROR GOTO 10460
10410 OPEN A$+".FMT" FOR OUTPUT AS #1
10430 PRINT #1,NF:FOR J=0 TO NF:PRINT #1
,F$(J):NEXT
10440 CLOSE #1
10450 RETURN
```

Error handler #2.

```
10460 PRINT"Disk error. Press any key.":
LOCATE ,,1:A$=INPUT$(1):RESUME 10400
```

Set printer for compressed print.

```
20000 LITTLE=-1:RETURN
```

IBM® PC

Set printer for standard print.

```
30000 LITTLE=0:RETURN
```

Set prompting for deletions.

```
31000 LINE INPUT"Do you want prompting (
Y/N)? ";A$:A$=FNU$(LEFT$(A$,1)):IF A$="Y
" THEN PROMPTING=-1:RETURN
31010 IF A$<>"N" THEN 31000
31020 PROMPTING=0:RETURN
```

Subroutine to display free memory.

```
50000 PRINT"You have room for";FRE(X$)-5
0;"more characters.":PRINT"Hit a key.";:
LOCATE ,,1:A$=INPUT$(1):RETURN
```

Subroutine to convert file names to all upper-case letters.

```
51000 IF UP$<>"" THEN FOR CHAR=1 TO LEN(
UP$):MID$(UP$,CHAR,1)=FNU$(MID$(UP$,CHAR
,1)):NEXT CHAR:RETURN ELSE RETURN
```

Data for the title page.

```
60000 DATA "5Data Base","7by Mark Pelcza
rski","8PC version by Fred Condo","20Cop
yright (c) 1982 by","21SoftSide Publicat
ions, Inc.","#"
```

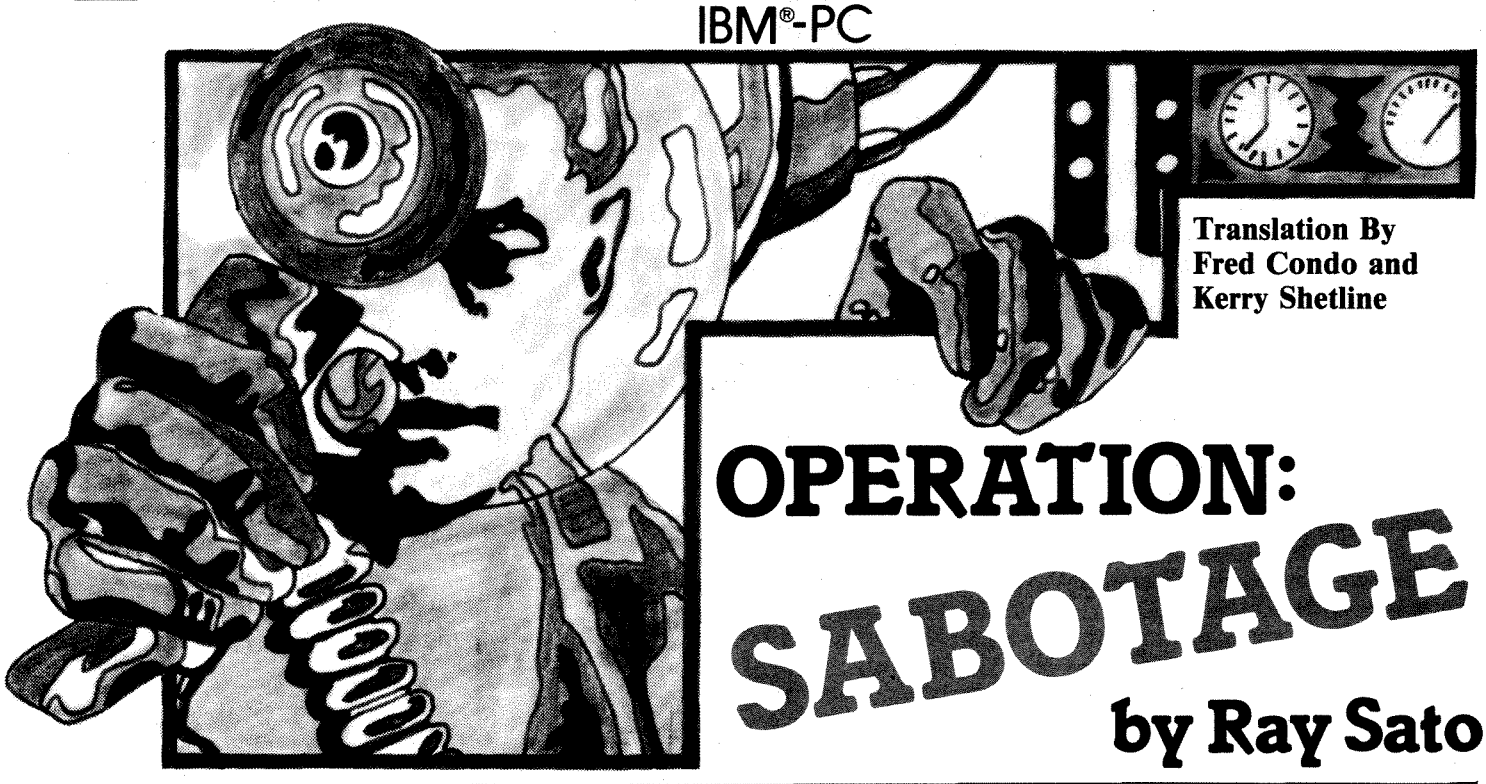
Subroutine to restore the function-key definitions upon exit from the program.

```
61000 KEY ON:KEY 1,"LIST ":KEY 2,"RUN"+C
HR$(13):KEY 3, "LOAD"+CHR$(34):KEY 4, "S
AVE"+CHR$(34):KEY 5, "CONT"+CHR$(13):KEY
 6, ","+CHR$(34)+"LPT1:"+CHR$(13):KEY 7,
 "TRON"+CHR$(13):KEY 8, "TROFF"+CHR$(13)
:KEY 9, "KEY ":KEY 10, "SCREEN 0,0,0"+CH
R$(13):RETURN
```

IBM-PC® SWAT TABLE FOR: DATA BASE

LINES	SWAT CODE	LENGTH	LINES	SWAT CODE	LENGTH
25 - 103	KI	503	5085 - 6110	TK	242
105 - 200	CL	341	6120 - 7045	HB	248
210 - 294	IR	303	7050 - 7135	EU	267
295 - 330	YK	360	7140 - 8015	JG	256
340 - 540	XB	377	8020 - 8090	OF	423
550 - 1140	AB	452	8095 - 8250	MV	321
1200 - 1510	KA	299	8255 - 8310	GR	334
1515 - 2050	BV	265	8320 - 8380	QN	190
2060 - 2230	QU	267	8390 - 8999	LT	363
2240 - 3008	HM	279	9000 - 10100	FD	396
3010 - 3310	BH	402	10102 - 10220	JE	516
3340 - 3800	OS	355	10230 - 10360	YY	448
3805 - 3910	ZD	384	10370 - 30000	AH	305
3920 - 4020	XN	268	31000 - 61000	NF	630
4030 - 5080	LI	370			

IBM®-PC



Translation By
Fred Condo and
Kerry Shetline

OPERATION: SABOTAGE

by Ray Sato

***Operation: Sabotage* is a fantasy/adventure game for an IBM® PC with 16K RAM.**

It is the year 2101 and war has broken out between Earth and the distant planet Zekloke. This alien power has established a large military complex on Mars which will soon become a great danger to Earth. Hidden in the massive installation are several secret documents containing the plans for an incredible defense shield — strong enough to stop an entire fleet of spacecraft.

You are a special agent and have just succeeded in sneaking into the alien complex. Your mission is to destroy this threat to mankind and return with plans for the powerful defense shield. The outcome of this mission will decide the fate of mankind.

Playing Notes

The computer will always give you a brief description of where you are, what objects you can see, and what exits are visible. You move and act by typing in simple commands, generally consisting of a verb and a noun. If the computer tells you that there is a laser pistol in the room, for example, you might want to type in the command "GET PISTOL". Later, you might be able to use it to "SHOOT MONSTER" or for some other purpose. If you no longer want to carry it, you can "DROP PISTOL" whenever you please. Since the computer looks only at the first three letters of the verb and the last three letters of the noun, you may use abbreviations such as "SHO TER" (for "SHOOT MONSTER") if you desire. Movement is accomplished by entering just a single letter rather than a two-word command: N, S, E, or W for north, south, east,or west. Typing the single word "INVENTORY" (or "INV") will display a list of what you are carrying. Typing "STATUS" (or "STA") will give you a readout of your current physical condition.

Part of the challenge of any fantasy/adventure game such as *Operation: Sabotage* is to figure out what you are able to do in a particular situation. Therefore, you will not find a list of all the verbs the computer can understand, or of all the objects you may discover. You might find yourself frustrated by what seem to be dead-ends, and end up getting killed in the process. This is all part of the adventure, and a test of your ingenuity and perseverance.

Program Notes

The most obvious feature of the program listing is that most of it looks like a cryptogram. The BASIC keywords are all in their usual form, but the string assignment statements and DATA lines contain incomprehensible garbage. This is because all of the room descriptions, object names, monsters, and verbs have been encoded. This has been done to preserve the value of the game. Anyone who types an adventure program in from a listing is bound to be disappointed in the game's playability, since he has gained so many clues about the plot. So, even though the typing is made slightly difficult by the scrambled words, this is the only reasonable way of publishing adventure programs in listed form. We have also omitted the usual list of variables for the same reasons. The variable descriptions give away too much information and the encoding of the program reduces the usefulness of a variable list.

SWAT

In order to offset the proofreading problems created by this approach, we have included an expanded *SWAT* Table for this program. (For more details on *SWAT,* see the original article in Issue 35 of ***SoftSide***.) Instead of the normal 12-line/500-byte *SWAT* parameters, we have used 5-line/200-byte parameters. This will provide an expanded *SWAT* Table, enabling you to pinpoint typing mistakes more easily.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS       IBM PC  BASIC         SS
SS    'Operation: Sabotage'    SS
SS      Author:  Ray Sato      SS
SS Translation:  Fred Condo &  SS
SS       Kerry Shetline        SS
SS     Copyright (c) 1982      SS
SS SoftSide Publications, Inc  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type this program, it is also included on this month's SoftSide DV.

Jump to program initialization.

```
1 GOTO 2550
3 GOSUB 4:GOTO 2210
```

Decode and print output.

```
4 IF P$="" THEN RETURN
5 FOR P=1 TO LEN(P$):J=ASC(MID$(P$,P,1))
:PRINT CHR$(ABS(J+155*(J>64 AND J<91)+18
7*(J>96 AND J<123)));:NEXT:PRINT:RETURN
```

Encode input.

```
6 V$="":IF V0$="" THEN RETURN
7 FOR P=1 TO LEN(V0$):J=ASC(MID$(V0$,P,1
)):V$=V$+CHR$(ABS(J+155*(J>64 AND J<91)+
187*(J>96 AND J<123))):NEXT:RETURN
9 GOSUB 4:GOTO 2210
```

Descriptions of individual rooms.

```
10 A$="ZM ZRIOLXP. GSVIV RH Z YOFV YFGGL
M SVIV":S=2:RETURN
20 A$="Z MZIILD XLIIRWLI":N=1:S=3:RETURN
30 A$="Z MZIILD XLIIRWLI":N=2:S=4:RETURN
40 A$="Z MZIILD XLIIRWLI":N=3:S=5:RETURN
50 A$="Z HNZOO ILLN":N=4:S=6:RETURN
60 A$="Z WVXLMGZNRMZGRLM XSZNYVI":B$="GS
VIV RH Z YOFV YFGGLM SVIV":N=5:S=7:RETUR
N
70 A$="Z HNZOO HGLIZTV XSZNYVI":N=6:S=8:
W=12:RETURN
80 A$="Z HNZOO XSZNYVI":N=7:S=9:W=13:RET
URN
90 A$="Z HNZOO VOVXGILMRX OZYLIZGLIB":N=
8:S=10:W=14:RETURN
100 A$="Z YRLOLTRXZO OZYLIZGLIB. GSVIV R
H Z IVW YFGGLM LM GSV DZOO":N=9:W=15:RET
URN
110 A$="Z HGLIZTV XSZNYVI":W=16:RETURN
120 A$="Z OZITV XSZNYVI":B$="GSVIV RH Z
XZYRMVG SVIV":S=13:W=17:E=7:RETURN
130 A$="Z HGIZMTV KFIKOV ILLN. GSVIV RH
Z YOFV  YFGGLM SVIV":N=12:S=14:W=18:E=8:
RETURN
140 A$="Z HNZOO LUURXV":N=13:S=15:E=9:RE
TURN
150 A$="Z HNZOO ILLN DRGS Z XZIW GZYOV R
M GSV   XVMGVI":N=14:S=16:W=20:E=10:RETU
RN
160 A$="Z OZITV LUURXV. GSVIV RH Z WVHP
SVIV":N=15:W=21:E=11:RETURN
170 A$="Z LUURXV DRGS Z OZITV WVHP":S=18
:E=12:RETURN
180 A$="Z HGLIZTV ILLN":N=17:S=19:E=13:R
ETURN
190 A$="Z OZITV SZOO":N=18:S=20:RETURN
200 A$="ZM VMGVIGZRMNVMG ILLN. Z HXIVVM
IVHGH LMGSV DZOO":B$="GSVIV RH Z YOFV ZM
W Z IVW YFGGLM FMWVI  GSV HXIVVM":N=19:S
=21:E=15:RETURN
210 A$="Z WZGZ IVXLIW HGLIZTV ILLN":N=20
:E=16:RETURN
220 A$="IZWZI XLMGILO. GSVIV RH Z HNZOO
HXIVVM  SVIV":S=23:W=27:RETURN
230 A$="Z NVWRXZO HGZGRLM. GSVIV RH Z OZ
ITV     GZYOV SVIV":N=22:S=24:W=28:RETUR
N
240 A$="Z HVXFIRGB HGZGRLM":N=23:RETURN
250 A$="Z IZWRL ILLN":S=26:W=30:RETURN
260 A$="Z HNZOO ILLN.  GSVIV RH Z HZUV R
M GSV   HLFGS DZOO":N=25:W=31:RETURN
270 A$="GSV ILYLG XLMGILO XVMGVI. GSVIV
RH Z    HNZOO XLMGILO XLNKFGVI NLFM6VW R
M GSV   DZOO":B$="":W=32:E=22:RETURN
280 A$="GSV DVZKLMH HGLIZTV ILLN":S=29:W
=33:E=23:RETURN
290 A$="Z ORYIZIB":N=28:S=30:W=34:RETURN
300 A$="Z HVXFIRGB XSVXP ZIVZ":N=29:W=36
:E=25:RETURN
310 A$="Z HNZOO ILLN DRGS Z WVHP. Z HRTM
 IVZWH":B$="'KIVHHFIV GL IVZXGLI.' Z YOF
V YFGGLM RH OLXZGVW FMWVI GSV HRTM":E=26
:RETURN
320 A$="OZFMXS XLMGILO. GSVIV RH Z WVHP
SVIV":S=33:W=37:E=27:RETURN
330 A$="Z HNZOO XLIIRWLI":N=32:S=34:E=28
:RETURN
340 A$="Z HGIZMTV YOFV ILLN. GSVIV RH Z
IVW     YFGGLM SVIV":N=33:S=35:E=29:RETU
RN
350 A$="Z GRMB HGLIZTV ILLN":N=34:W=40:R
ETURN
360 A$="Z HNZOO XSZNYVI. Z WLLI DVHG OVZ
WH GL   GSV MFXOVZI IVZXGLI":E=30:RETURN
370 A$="Z HNZOO, MZIILD XLIIRWLI":N=42:E
=32:RETURN
380 A$="GSV XLNKFGVI XVMGVI. GSVIV RH Z
HNZOO   HOLG RM GSV XLNKFGVI":S=39:RETUR
N
390 A$="GSV XSVNRXZO OZY":N=38:S=40:RETU
RN
400 A$="GSV IVZXGLI XLMGILO XVMGVI.  GSV
IV RH Z YOFV YFGGLM ZMW Z IVW LMV.  Z HR
TM HZBH IVZXGLI XLMGILO - IVW=LM, YOFV=L
UU":N=39:E=35:RETURN
410 A$="GSV MFXOVZI IVZXGLI. Z XLNKFGVI
IVHGH LMGSV DZOO":E=36:RETURN
420 A$="GSV DVHG VMW LU Z OLMT XLIIRWLI"
:S=37:E=43:RETURN
430 A$="GSV VZHG VMW LU Z OLMT XLIIRWLI"
:W=42:E=44:RETURN
440 A$="Z HVXFIRGB XVMGVI":W=43:E=45:RET
URN
450 A$="Z HNZOO OZFMXS ZIVZ":B$="GSVIV R
H Z HNZOO HOLG MVG6 GL GSV OZFMXSTZGV":W
=44:RETURN
```

Extended room descriptions.

```
460 IF A=10 AND (D3=1 OR D3=2) THEN C$="
GSV NLMHGVI XZTV RH LKVM"
470 IF A=12 AND D5=0 THEN C$="GSV XZYRMV
G RH OLXPVW"
480 IF A=12 AND D5=1 THEN C$="GSV XZYRMV
G RH LKVM"
490 IF A=20 AND D6=0 THEN C$="GSV HXIVVM
 RH YOZMP"
500 IF A=20 AND D6=1 THEN C$="Z NLERV RH
 YVRMT KOZBVW LM GSV HXIVVM"
510 IF A=26 AND D9=0 THEN C$="GSV HZUV R
H OLXPVW"
520 IF A=26 AND D9=1 THEN C$="GSV HZUV R
H LKVM"
530 IF A=27 AND E2=0 THEN C$="GSV XLNKFG
VI RH ZXGREV"
540 IF A=27 AND E2=1 THEN C$="GSV XLNKFG
VI RH WVHGILBVW"
550 IF A=36 AND E6=0 THEN C$="GSV IVZXGL
I WLLI RH URINOB OLXPVW"
560 IF A=36 AND E6=1 THEN C$="GSV IVZXGL
I WLLI RH LKVM":W=41
570 IF A=45 AND E9=0 THEN C$="GSV OZFMXS
 TZGV RH XOLHVW"
580 IF A=45 AND E9=1 THEN C$="GSV OZFMXS
 TZGV RH LKVM":E=46
```

Generate the list of visible items and available exits.

```
590 A$=A$+".":IF LEN(B$)>3 THEN B$=B$+".
"
600 IF LEN(C$)>3 THEN C$=C$+"."
610 IF N<>0 THEN E$="MLIGS "
```

```
620 IF S<>0 THEN E$=E$+"HLFGS "
630 IF W>0 THEN E$=E$+"DVHG "
640 IF E>0 THEN E$=E$+"VZHG "
650 IF E$<>"" THEN E$=LEFT$(E$,LEN(E$)-1
)
```

Describe current location, visible items, and available exits.

```
660 CLS:PRINT"YOU ARE IN:":P$=A$:GOSUB 4
:PRINT:IF B$<>"" THEN P$=B$:GOSUB 4
670 IF C$<>"" THEN P$=C$:GOSUB 4
680 PRINT:PRINT"OBJECTS YOU CAN SEE: ":P
$=" ":FOR T=1 TO 16:IF A=I(T) THEN P$=I$
(T):GOSUB 4
690 NEXT:IF P$=" " THEN P$="-NLGSRMT-":G
OSUB 4
700 PRINT:PRINT"EXITS: ":P$=E$:GOSUB 4
```

Print out additional warnings, messages, etc.

```
710 IF (A=40 OR A=35 OR A=30 OR A=31) AN
D I(4)=0 AND F3=0 THEN P$="GSV HNZOO YOZ
XP WVERXV RH YORMPRMT":GOSUB 4
720 IF A=36 AND I(4)=0 AND F3=0 THEN P$=
"GSV HNZOO YOZXP WVERXV RH UOZHSRMT
 YIRTSGOB":GOSUB 4
730 IF F4<>0 THEN F4$=RIGHT$(STR$(F4),2)
:P$="GSV XLNKFGVI HZBH: '"+F4$+" NRMFGVH
 FMGRO WV-HGIFXGRLM'":GOSUB 4
740 IF D3=1 THEN P$="* * * ZORVM NLMHGVI
 ZGGZXPRMT * * *":GOSUB 4
750 IF D7=1 OR E0=1 OR E3=1 OR E7=1 THEN
 P$="* * * HVXFIRGB KZGILO ZGGZXPRMT * *
 *":GOSUB 4
```

Get and interpret command.

```
760 PRINT:INPUT"COMMAND";V0$:GOSUB 6
770 FOR T=1 TO 4:IF V$=LEFT$(V$(T),1) TH
EN V$=V$(T)
780 NEXT T
790 IF LEN(V$)<3 THEN 660
800 V1$=LEFT$(V$,3):V2$=RIGHT$(V$,3)
810 FOR T=1 TO 17:IF V1$=LEFT$(V$(T),3)
THEN V1=T
820 NEXT T:IF V1=0 THEN P$="R WLM'G FMWV
IHGZMW DSZG BLF DZMG.":GOSUB 4:FOR II=1
TO 1000:NEXT:GOTO 2210
830 FOR T=1 TO 16:IF V2$=RIGHT$(I$(T),3)
 THEN V2=T
840 NEXT T
850 ON V1 GOTO 870,910,950,1020,1100,128
0,1340,1360,1530,1590,1630,1760,1830,188
0,1980,2140,2200
860 GOTO 2210
```

Handle commands.

```
870 IF N=0 THEN 1080
880 IF D3=1 THEN P$="GSV NLMHGVI YOLXPH
GSV VCRG":GOTO 3
890 IF D7=1 OR E3=1 THEN GOTO 1070
900 A=N:GOTO 2210
910 IF S=0 THEN 1080
920 IF S=24 AND D8<>0 AND E2<>1 THEN D7=
1:GOSUB 1090
930 IF S=30 AND E4<>0 AND E2<>1 THEN E3=
1:GOSUB 1090
940 A=S:GOTO 2210
950 IF W=0 THEN 1080
960 IF D3=1 THEN P$="GSV NLMHGVI YOLXPH
GSV VCRG":GOTO 3
970 IF E0=1 OR E3=1 OR E7=1 THEN 1070
980 IF W=41 AND F3=0 THEN P$="IZWRZGRLM
UILN GSV IVZXGLI SRGH BLF":GOSUB 4:GOTO
2460
990 IF W=30 AND E4<>0 AND E2<>1 THEN E3=
1:GOSUB 1090
1000 IF W=27 AND E1<>0 AND E2<>1 THEN E0
=1:GOSUB 1090
1010 A=W:GOTO 2210
1020 IF E=0 THEN 1080
1030 IF E0=1 OR E3=1 OR E7=1 THEN 1070
1040 IF E=27 AND E1<>0 AND E2<>1 THEN E0
=1:GOSUB 1090
1050 IF E=44 AND E8<>0 AND E2<>1 THEN E7
=1:GOSUB 1090
1060 A=E:GOTO 2210
1070 P$="GSV HVXFIRGB ZMWILRW YOLXPH GSV
 VCRG":GOTO 3
1080 P$="GSVIV RH ML DZB GL TL GSZG WRIV
XGRLM":GOSUB 4:FOR J=1 TO 1500:NEXT:GOTO
 2210
1090 P$="Z HVXFIRGB ZMWILRW ZDZRGH BLF":
GOSUB 4:RETURN
1100 IF A=1 AND V2$="LXP" THEN P$="GSV Z
RIOLXP LKVMH ZMW BLF ZIV YOLDM LFG RMGL
GSV EZXFFN LU HKZXV":GOSUB 4:GOTO 2460
1110 IF A=12 AND V2$="MVG" AND D5=0 AND
I(2)<>0 THEN P$="BLF QFHG ZIVM'G HGILMT
VMLFTS GL ULIXV  RG LKVM":GOTO 3
1120 IF A=12 AND V2$="MVG" AND D5=0 AND
I(2)=0 THEN P$="GSV XILDYZI SVOKVW.  GSV
 XZYRMVG RH MLD LKVM":GOSUB 4:D5=1:I(5)=
ABS(I(5)):GOTO 2210
1130 IF A=12 AND V2$="MVG" AND D5=1 THEN
 P$="GSV XZYRMVG RH ZOIVZWB LKVM":GOTO 3
1140 IF A=16 AND V2$="VHP" THEN PRINT"OK
":I(6)=ABS(I(6)):GOTO 2210
1150 IF A=17 AND V2$="VHP" THEN PRINT"OK
":I(7)=ABS(I(7)):GOTO 2210
1160 IF A=26 AND V2$="ZUV" AND D9=1 THEN
 P$="GSV HZUV RH ZOIVZWB LKVM":GOTO 3
1170 IF A=26 AND V2$="ZUV" AND D9=0 THEN
 P$="R WLM'G SZEV GSV PVB GL LKVM GSV HZ
UV":GOTO 3
1180 IF A=31 AND V2$="VHP" THEN P$="LP.
BLF URMW MLGSRMT RMHRWV":GOTO 3
1190 IF A=32 AND V2$="VHP" THEN PRINT"OK
":I(14)=ABS(I(14)):GOTO 2210
1200 IF A=36 AND V2$="LLI" AND E6=1 THEN
 P$="GSV WLLI RH ZOIVZWB LKVM":GOTO 3
1210 IF A=36 AND V2$="LLI" AND E6=0 AND
I(6)<>0 THEN P$="BLF WLM'G SZEV GSV PVB
GL GSV WLLI":GOTO 3
1220 IF A=36 AND V2$="LLI" AND E6=0 AND
I(6)=0 AND E5=0 THEN P$="BLF ZIV HFXPVW
RMGL GSV FMKIVHHFIRAVW  IVZXGLI YFROWRM
T":GOSUB 4:GOTO 2460
1230 IF A=36 AND V2$="LLI" AND I(6)=0 TH
EN P$="GSV WLLI RH MLD LKVM":GOSUB 4:E6=
1:GOTO 2210
1240 IF A=41 AND V2$="MVO" THEN P$="GSV
KZMVO RH URINO8 OLXPVW":GOTO 3
1250 IF A=45 AND V2$="LXP" AND E9=1 THEN
 P$="GSV ZRIOLXP RH ZOIVZWB LKVM":GOTO 3
1260 IF A=45 AND V2$="LXP" AND E9=0 THEN
 P$="GSVIV ZIVM'G ZMB ERHRYOV XLMGILOH":
GOTO 3
1270 P$="R XZM'G WL GSZG":GOTO 3
1280 IF V2$="GVI" OR V2$="LRW" THEN P$="
WLM'G YV IRWRXFOLFH":GOTO 3
1290 IF V2=0 THEN P$="R XZM'G WL GSZG":G
OTO 3
1300 IF I(V2)=0 THEN P$="BLF ZOIVZWB SZE
V GSZG":GOTO 3
1310 IF A<>I(V2) THEN P$="R WLM'G HVV RG
 SVIV":GOTO 3
1320 IF P4>=8 THEN P$="HLIIB, BLF XZM'G
XZIIB ZMBGSRMT NLIV":GOTO 3
1330 P4=P4+1:I(V2)=0:PRINT"OK":GOTO 2210
1340 IF V2=0 THEN P$="BLF WLM'G SZEV GSZ
G":GOTO 3
1350 P4=P4-1:I(V2)=A:PRINT"OK":GOTO 2210
1360 IF I(5)<>0 THEN P$="BLF WLM'G SZEV
Z DVZKLM":GOTO 3
1370 IF A=1 AND V2$="LXP" THEN P$="BLF Z
IV YOLDM LFG LU GSV ZRIOLXP RMGL  GSV E
ZXFFN LU HKZXV":GOSUB 4:GOTO 2460
1380 IF A=27 AND V2$="GVI" THEN P$="GSV
XLNKFGVI RH WVHGILBVW":GOSUB 4:E2=1:E0=0
:GOTO 2210
1390 IF A=38 AND V2$="GVI" THEN P$="GSV
HSLG IVUOVXGH LUU LU GSV XLNKFGVI":GOSUB
 4:GOTO 2460
1400 IF A=41 AND V2$="GVI" THEN P$="GSV
DSLOV MFXOVZI IVZXGLI RH VCKOLWRMT":GOSU
B 4:GOTO 2460
```

```
1410 IF V2$="RWH" OR V2$="YLG" OR V2$="I
LO" OR V2$="IWH" OR V2$="ZIW" THEN V2$="
LRW"
1420 IF V2$<>"GVI" AND V2$<>"LRW" THEN P
$="GSV OZHVI HSLG SZH ML VUUVXG":GOTO 3
1430 IF V2$="GVI" AND D3=0 THEN P$="R WL
M'G HVV ZMB NLMHGVI SVIV":GOTO 3
1440 IF V2$="LRW" AND D7=0 AND E0=0 AND
E3=0 AND E7=0 THEN P$="R WLM'G HVV ZMB Z
MWILRWH SVIV":GOTO 3
1450 T=INT(100*RND(1))+1:IF T>P2+P3+50 T
HEN P$="BLF URIV ZMW NRHH":GOTO 3
1460 IF D3=1 THEN P$="BLF SRG GSV NLMHGV
I":GOSUB 4:D4=D4-((10+P2+P3)/2):IF D4<=0
 THEN D3=0:D4=0:P$="BLF SZEV PROOVW RG":
GOTO 3
1470 IF D7=1 THEN P$="BLF SRG GSV ZMWILR
W":GOSUB 4:D8=D8-((5+P2+P3)/2):IF D8<=0
THEN D7=0:D8=0:P$="RG RH WVHGILBVW":GOTO
 3
1480 IF E0=1 THEN P$="BLF SRG GSV ZMWILR
W":GOSUB 4:E1=E1-((5+P2+P3)/2):IF E1<=0
THEN E0=0:E1=0:P$="RG RH WVHGILBVW":GOTO
 3
1490 IF E3=1 THEN P$="BLF SRG GSV ZMWILR
W":GOSUB 4:E4=E4-((5+P2+P3)/2):IF E4<=0
THEN E3=0:E4=0:P$="RG RH WVHGILBVW":GOTO
 3
1500 IF E7=1 THEN P$="BLF SRG GSV ZMWILR
W":GOSUB 4:E8=E8-((5+P2+P3)/2):IF E8<=0
THEN E7=0:E8=0:P$="RG RH WVHGILBVW":GOTO
 3
1510 IF D3=1 THEN P$="RG RH HGROO ZOREV"
:GOTO 3
1520 P$="GSV ZMWILRW RH HGROO UFMXGRLMRM
T":GOTO 3
1530 IF V2=0 THEN P$="R XZM'G WL GSZG":G
OTO 3
1540 IF I(V2)<>0 THEN P$="R WLM'G SZEV G
SZG":GOTO 3
1550 IF V2<>9 AND V2<>14 THEN P$="R XZM'
G WL GSZG":GOTO 3
1560 IF (V2=9 AND A=44) OR (V2=14 AND A=
38) THEN P$="MLGSRMT SZKKVMH":GOSUB 4
1570 IF V2=9 AND A=38 THEN F4=35:P$="GSV
 XLNKFGVI IVKORVH: 'YZHV WVHGIFXG    HVJ
FVMXV HGZIGVW' WVHGIFXGRLM RM 34     NRM
FGVH.":GOSUB 4:P4=P4-1:I(9)=100:GOTO 221
0
1580 IF V2=14 AND A=45 THEN P$="GSV TZGV
 LKVMH":GOSUB 4:E9=1:GOTO 2210
1590 IF V2<>10 THEN P$="WLM'G YV IRWRXFO
LFH":GOTO 3
1600 IF I(10)<>0 THEN P$="BLF WLM'G SZEV
 GSZG":GOTO 3
1610 PRINT"OK":I(10)=50:P4=P4-1:P1=P1+5+
P3:IF P0<P1 THEN P0=P1
1620 GOTO 2210
1630 IF A=1 AND V2$="OFV" THEN P$="GSV Z
RIOLXP LKVMH...  BLF ZIV YOLDM LFG RMGL
GSV EZXFFN LU HKZXV":GOSUB 4:GOTO 2460
1640 IF A=6 AND V2$="OFV" THEN P$="Z HGI
ZMTV, LIZMTV TOLD XLEVIH BLF ZMW   GSVM
UZWVH ZDZB":GOTO 3
1650 IF A=10 AND V2$="IVW" AND D3=1 THEN
 P$="MLGSRMT SZKKVMH":GOTO 3
```

```
1660 IF A=10 AND V2$="IVW" THEN D3=1:P$=
"ZM ZORVM NLMHGVI RH IVOVZHVW.  RG RH
 ZGGZXPRMT BLF!":GOTO 3
1670 IF A=13 AND V2$="OFV" THEN A=34:P$=
"Z UOZHS LU ORTSG GVNKLIZIROB YORMWH BLF
":GOTO 3
1680 IF A=20 AND V2$="IVW" AND D6=0 THEN
 P$="MLGSRMT SZKKVMH":GOTO 3
1690 IF A=20 AND V2$="IVW" THEN D6=0:P$=
"GSV HXIVVM TLVH YOZMP":GOTO 3
1700 IF A=20 AND V2$="OFV" THEN D6=1:P$=
"GSV HXIVVM ORTSGH FK":GOTO 3
1710 IF A=31 AND V2$="OFV" THEN E5=1:PRI
NT"OK":GOTO 2210
1720 IF A=34 AND V2$="IVW" THEN A=13:P$=
"Z UOZHS LU ORTSG GVNKLIZIROB YORMWH BLF
":GOTO 3
1730 IF A=40 AND V2$="IVW" THEN F3=0:PRI
NT"OK":GOTO 2210
1740 IF A=40 AND V2$="OFV" THEN F3=1:PRI
NT"OK":GOTO 2210
1750 P$="MLGSRMT SZKKVMH":GOTO 3
1760 IF A=22 AND V2$="VVM" THEN P$="BLF
XZM HVV MLGSRMT LU RMGVIVHG LM GSV  IZWZ
I":GOTO 3
1770 IF V2=0 THEN P$="R WLM'G SZEV GSZG"
:GOTO 3
1780 IF I(V2)<>0 AND A<>I(V2) THEN P$="R
 WLM'G SZEV GSZG":GOTO 3
1790 IF V2=3 OR V2=13 THEN P$="R HVV MLG
SRMT HKVXRZO":GOTO 3
1800 IF V2=9 THEN P$="HLIIB, LMOB Z XLNK
FGVI XZM IVZW Z       KILTIZN":GOTO 3
1810 IF V2=16 THEN P$="GSV KOZMH ZIV HVZ
OVW...LMOB XLNNZMW XZM LKVM GSVN":GOTO 3
1820 P$="R XZM'G IVZW GSZG":GOTO 3
1830 CLS:P$="* * * KOZBVI'H RMEVMGLIB *
* *":GOSUB 4
1840 PRINT:FOR T=1 TO 16:IF I(T)=0 THEN
P$="- "+I$(T):GOSUB 4
1850 NEXT T
1860 PRINT:INPUT"HIT <RETURN> TO CONTINU
E";T$:PRINT
1870 GOTO 2360
1880 IF V2=0 THEN P$="R XZM'G WL GSZG":G
OTO 3
1890 IF I(V2)<>0 THEN P$="R WLM'G SZEV G
SZG":GOTO 3
1900 IF V2=1 AND A=12 AND D5=0 THEN P$="
GSV XZYRMVG OLXP RH WVHGILBVW":GOSUB 4:D
5=1:I(1)=100:I(5)=ABS(I(5)):P4=P4-1:GOTO
 2210
1910 IF (V2=1 OR V2=15) AND (D3=1 OR D7=
1 OR E0=1 OR E3=1 OR E7=1) THEN I(V2)=10
0:P4=P4-1:GOTO 1460
1920 IF (V2=1 OR V2=15) AND A=1 THEN P$=
"GSV ZRIOLXP RH WVHGILBVW...BLF ZIV YOLD
MLFG RMGL GSV EZXFFN LU HKZXV!":GOSUB 4:
GOTO 2460
1930 IF (V2=1 OR V2=15) AND A=36 AND E6=
0 AND E5=0 THEN P$="GSV WLLI RH WVHGILBV
W... BLF ZIV HFXPVW RMGL GSV FMKIVHHFIRA
VW IVZXGLI YFROWRMT":GOSUB 4:GOTO 2460
1940 IF (V2=1 OR V2=15) AND A=36 AND E6=
0 AND F3=0 THEN P$="GSV WLLI RH WVHGILBV
W. BLF ZIV YLNYZIWVWDRGS IZWRZGRLM":GOSU
B 4:GOTO 2460
1950 IF (V2=1 OR V2=15) AND A=36 AND E6=
```

```
0 THEN P$="GSV WLLI RH WVHGILBVW":GOSUB
4:E6=1:I(V2)=100:P4=P4-1:GOTO 2210

1960 IF V2=1 OR V2=15 THEN P$="GSV "+I$(
V2)+" SZH ML VUUVXG":GOSUB 4:I(V2)=100:P
4=P4-1:GOTO 2210

1970 GOTO 1340
1980 IF V2=0 THEN P$="R XZM'G WL GSZG":G
OTO 3

1990 IF I(V2)<>0 THEN P$="R WLM'G SZEV G
SZG":GOTO 3
2000 IF V2=5 AND D3=1 THEN V2$="GVI"

2010 IF V2=5 AND (D7=1 OR E0=1 OR E3=1 O
R E7=1) THEN V2$="LRW"
2020 IF V2=5 THEN 1360

2030 IF V2=4 AND F3=0 AND (A=40 OR A=35
OR A=30 OR A=31) THEN P$="GSV YOZXP WVER
XV RH YORMPRMT":GOTO 3

2040 IF V2=4 AND F3=0 AND A=36 THEN P$="
GSV YOZXP WVERXV RH UOZHSRMT YIRTSGOB":G
OTO 3

2050 IF V2=4 THEN P$="GSVIV ZIVM'G ZMB E
RHRYOV XLMGILOH LM    GSRH WVERXV":GOTO
3

2060 IF V2=12 THEN I(12)=A:P4=P4-1:I$(12
)="ZINVW KSLGLM YLNY":F2=35:P$="GSV YLNY
 DROO VCKOLWV RM 35 NRMFGVH":GOTO 3

2070 IF V2=2 AND A=12 AND D5=0 THEN D5=1
:P$="GSV XZYRMVG RH MLD LKVM":GOSUB 4:I(
5)=ABS(I(5)):GOTO 2210

2080 IF V2=2 AND A=12 AND D5=1 THEN P$="
GSV XZYRMVG RH ZOIVZWB LKVM":GOTO 3

2090 IF V2=7 AND A=26 AND D9=0 THEN D9=1
:I(16)=ABS(I(16)):P$="GSV HZUV LKVMH":GO
TO 3

2100 IF V2<>11 THEN P$="DSZG WL BLF DZMG
 NV GL WL DRGS GSV      "+I$(V2)+"?":GOT
O 3

2110 IF I(8)<>0 THEN P$="GSVIV ZIVM'G ZM
B YZGGVIRVH ULI GSV IZWRL":GOTO 3


2120 IF F2<>0 THEN F2$=STR$(F2):PRINT:P$
="Z ELRXV HZBH 'YLNY HGZGFH:":GOSUB 4:P$
=F2$+" NRMFGVH FMGRO WVGLMZGRLM'":GOTO 3

2130 P$="GSV IZWRL RH HROVMG":GOTO 3
2140 CLS:P$="* * * KOZBVI'H HGZGFH * * *
":GOSUB 4:PRINT

2150 P1$=STR$(P1):P$="XFIIVMG SRG KLRMGH
 = "+P1$:GOSUB 4

2160 P2$=STR$(P2):P$="WVCGVIRGB ZGGIRYFG
V = "+P2$:GOSUB 4

2170 P3$=STR$(P3):P$="OFXP ZGGIRYFGV
 = "+P3$:GOSUB 4

2180 PRINT:PRINT"HIT <RETURN> TO CONTINU
E";:T$=INPUT$(1):PRINT
2190 GOTO 2360

2200 CLS:P$="TZNV LEVI":GOSUB 4:GOTO 247
0
```

Update player status. Conduct combat if appropriate.

```
2210 IF F2<>0 THEN F2=F2-1:IF F2<=0 THEN
 2380
2220 IF F4<>0 THEN F4=F4-1:IF F4<=0 THEN
 2420

2230 IF P1<P0 THEN P5=P5+.5:IF P5=1 THEN
 P5=0:P1=P1+1
2240 IF D3=0 AND D7=0 AND E0=0 AND E3=0
AND E7=0 THEN 2350
2250 T=INT(RND(1)*100)

2260 IF D3=1 THEN P$="GSV NLMHGVI ZGGZXP
H... ":GOSUB 4
2270 IF D3<>1 THEN P$="GSV HVXFIRGB ZMWI
LRW HSLLGH... ":GOSUB 4
2280 IF T>80-(P2+P3) THEN P$="RG NRHHVH"
:GOSUB 4:GOTO 2350


2290 P1=P1-(INT((5*RND(1))+1)+((5*RND(1)
)+1)+((5*RND(1))+1)+((5*RND(1))+1)+15-P3
)

2300 IF D3<>1 THEN P1=P1+5
2310 IF P1<0 THEN 2460
2340 P$="BLF ZIV SRG!":GOSUB 4

2350 IF V1=0 OR V1>4 OR D3+D7+E0+E3+E7>0
 THEN FOR ZZ=1 TO 3500:NEXT
```

Initialize for new turn. Jump to appropriate room description.

```
2360 V$="":V1$="":V2$="":V1=0:V2=0:A$=""
:B$="":C$="":D$="":E$="":N=0:S=0:W=0:E=0

2370 ON A GOSUB 10,20,30,40,50,60,70,80,
90,100,110,120,130,140,150,160,170,180,1
90,200,210,220,230,240,250,260,270,280,2
90,300,310,320,330,340,350,360,370,380,3
90,400,410,420,430,440,450,2500:GOTO 460
```

Evaluate end-game conditions and display appropriate messages.

```
2380 CLS:IF A=46 THEN F4=-1:GOTO 2500
2390 P$="GSV KSLGLM YLNY VCKOLWVH...GSV
VMGRIV   XLNKOVC RH WVHGILBVW":GOSUB 4

2400 P$="BLF SZEV YVVM PROOVW YB GSV ULI
XV LU GSVYOZHG":GOSUB 4

2410 PRINT:PRINT:GOTO 2470
2420 CLS:IF A=46 THEN 2500

2430 IF A=38 THEN P$="GSV XLNKFGVI UOZHS
VH YIRTSGOB, VNRGGRMT HKZIPH RM ZOO WRIV
XGRLMH":GOSUB 4

2440 P$="GSV XLNKOVC HFWWVMOB VCKOLWVH R
MGL NRO- ORLMH LU KRVXVH":GOSUB 4

2450 P$="BLF ZIV PROOVW YB GSV UZOORMT W
VYIRH    ZILFMW BLF":GOSUB 4:PRINT:PRINT
:GOTO 2470

2460 P$="BLF ZIV WVZW!":GOSUB 4

2470 INPUT"DO YOU WANT TO PLAY AGAIN? (Y
/N) ";A$
2480 IF LEFT$(A$,1)="Y" OR LEFT$(A$,1)="
y" THEN 2550
2490 PRINT:KEY ON:WIDTH 80:CLS:END
2500 P$="GSV HKZXV HSRK HFWWVMOB ORUGH R
MGL LIYRGZILFMW GSV KOZMVG":GOSUB 4

2510 IF ((F2=0) OR (F2<>0 AND I(12)<>41)
) AND F4=0 THEN P$="BLF WRWM'G WVHGILB G
SV YZHV.  BLF SZEV  UZROVW BLFI NRHHRLM.
":GOSUB 4:PRINT:GOTO 2470

2520 P$="UILN Z WRHGZMXV, BLF XZM HVV GS
V ZORVM  YZHV VCKOLWV":GOSUB 4

2530 IF I(16)<>0 THEN P$="BLF WRWM'G IVX
LEVI GSV HVXIVG KOZMH     MVVWVW YB HGZI
 XLNNZMW":GOSUB 4:PRINT:GOTO 2470

2540 P$="NRHHRLM RH Z HFXXVHH!":GOSUB 4:
GOTO 2470
```

Initialize workspace. Read in items and verbs.

```
2550 TROFF:CLS:KEY OFF:CLEAR:WIDTH 40:DI
M I$(16),I(16),V$(17)
2560 CLS:PRINT TAB(5);CHR$(34);"OPERATIO
N: SABOTAGE,";CHR$(34);" BY RAY SATO":PR
INT TAB(13);"IBM PC VERSION BY":PRINT TA
B(7);"FRED J. CONDO & KERRY SHETLINE"
```

continued on page 46

COLOR CODED: Multi-color DiscSavers™ are designed for easy recognition of individual disks with your own color-keyed filing system. Ideal for office or home use.

PROTECTIVE: Custom grain vinyl provides added protection for magnetic disks by guarding against common handling hazards.

ATTRACTIVE: DiscSavers provide a handsome and professional method of single disk storage and enhance the look of your hardware while protecting your valuable software.

DURABLE: Rigid vinyl construction protects against constant handling to ensure long wear and tear.

PORTABLE: DiscSavers are the only portable vinyl disk sleeves for use with a single diskette that bear the RockRoy mark of quality.

Contact your Dealer or Distributor.

ROCKROY INC.
Computer Products Division

7721 E. Gray Road
Scottsdale, Arizona 85260
(602) 998-1577
Toll-Free 800-528-2361

DiscSavers is a trademark of RockRoy Inc.

continued from page 44

```
2570 FOR T=1 TO 16:READ I$(T),I(T):NEXT
2580 FOR T=1 TO 17:READ V$(T):NEXT
2590 FOR T=1 TO 40:P0=P0+INT(2*RND(1))+1
:NEXT T
```

Establish player-attribute points. Jump to first room.

```
2600 P1=P0
2610 FOR T=1 TO 10:P2=P2+INT(2*RND(1))+1
:NEXT T
2620 FOR T=1 TO 10:P3=P3+INT(2*RND(1))+1
:NEXT T
2630 FOR T=1 TO 50:D4=D4+INT(2*RND(1))+1
:D8=D8+INT(2*RND(1))+1:E1=E1+INT(2*RND(1
))+1:E4=E4+INT(2*RND(1))+1:E8=E8+INT(2*R
ND(1))+1:NEXT T
2640 A=1:P4=1
2650 GOSUB 10:GOTO 460
```

Item and verb data.

```
2660 DATA KOZHGRX VCKOLHREV,0,XILDYZI,7,
XZOVMWZI,8,HNZOO YOZXP WVERXV,9,OZHVI KR
HGLO,-12,HVXFIRGB PVB,-16,VOVXGILMRX XLM
GILO YZGLM,-17,YZGGVIRVH,18,XLNKFGVI WVH
GIFXG KILTIZN,21,HROEVI KROO,23,KLIGZYOV
 IZWRL,25,OZITV KSLGLM YLNY,28
2670 DATA TZOZXGRX XSZIG,32,OZFMXS HBHGV
N XZHHVGGV,-32,MRGILTOBXVIRM,39,HVXIVG K
OZMH,-26
2680 DATA MLIGS,HLFGS,DVHG,VZHG,LKVM,TVG
,WILK,HSLLG,RMHVIG,VZG,KFHS,IVZW,RMEVMGL
IB,GSILD,FHV,HGZGFH,JFRG
```

IBM®-PC SWAT TABLE FOR: OPERATION: SABOTAGE

(Modified Parameters: NU = 5, B = 200)

LINES	SWAT CODE	LENGTH	LINES	SWAT CODE	LENGTH
1 - 6	BK	154	780 - 820	YN	182
7 - 20	UA	202	830 - 870	MP	153
30 - 70	HT	237	880 - 920	VU	148
80 - 110	NM	212	930 - 970	ZA	153
120 - 150	CA	274	980 - 1020	JN	183
160 - 200	YW	322	1030 - 1070	EW	177
210 - 240	IC	230	1080 - 1100	RY	226
250 - 270	FL	221	1110 - 1120	DI	223
280 - 310	WQ	262	1130 - 1160	JI	241
320 - 350	NN	229	1170 - 1200	LI	266
360 - 380	NC	202	1210 - 1220	DY	210
390 - 410	ZM	241	1230 - 1250	ZC	211
420 - 450	WM	235	1260 - 1290	HS	209
460 - 500	GK	259	1300 - 1340	HI	235
510 - 550	LF	244	1350 - 1380	CJ	262
560 - 600	CB	224	1390 - 1410	UJ	237
610 - 650	LR	139	1420 - 1440	RT	224
660 - 700	GS	227	1450 - 1470	ZC	291
710 - 730	UU	285	1480 - 1490	PE	222
740 - 770	SL	212	1500 - 1520	YT	201
			1530 - 1560	FL	202
			1570 - 1580	JX	206
			1590 - 1630	HL	264
			1640 - 1660	RN	246
			1670 - 1690	WD	203
			1700 - 1730	SE	227
			1740 - 1780	DY	254
			1790 - 1810	CD	201
			1820 - 1860	XC	176
			1870 - 1900	JJ	205
			1910 - 1930	ZN	340
			1940 - 1950	FB	214
			1960 - 2000	MI	210
			2010 - 2040	RB	236
			2050 - 2070	CW	273
			2080 - 2100	UX	218
			2110 - 2130	OU	210
			2140 - 2170	LA	206
			2180 - 2220	WM	161
			2230 - 2270	DL	217
			2280 - 2340	XC	191
			2350 - 2370	YD	334
			2380 - 2420	WD	197
			2430 - 2450	TP	236
			2460 - 2500	AK	205
			2510 - 2530	OX	293
			2540 - 2560	ZV	214
			2570 - 2610	WE	145
			2620 - 2660	VG	414
			2670 - 2680	VL	186

Attack at EP-CYG-4®

a space adventure from BRAM Inc.

For the ATARI® 400/800

You have just been revived from four years of stasis. Your burst-drive interstellar transport is now orbiting the fourth planet of Epsilon Cygnus. Your enemy, the Tartillians, are a brutal machine race that have destroyed their humanoid creators and have sworn the destruction of all humanoids. Your mission is to destroy the Tartillians and their production facilities. It won't be easy as the Tartillians fight back with cleverly placed ground batteries, cruise ships and deadly interceptors. How will you choose to face these brutal enemies? Board a gravitron attack ship/Uni-class and your fate is your own. You will have complete control of weapons, shields and the ship itself. Or will the mission demand the gravitron attack ship/Bi-class with your skills as a pilot guiding the ship and controlling the shields as your gunnery officer fires the lyso-blast weapon, and shares your fate.

No matter what ship type or mission you choose, Attack at EP-CYG-4 offers arcade action on a new level.

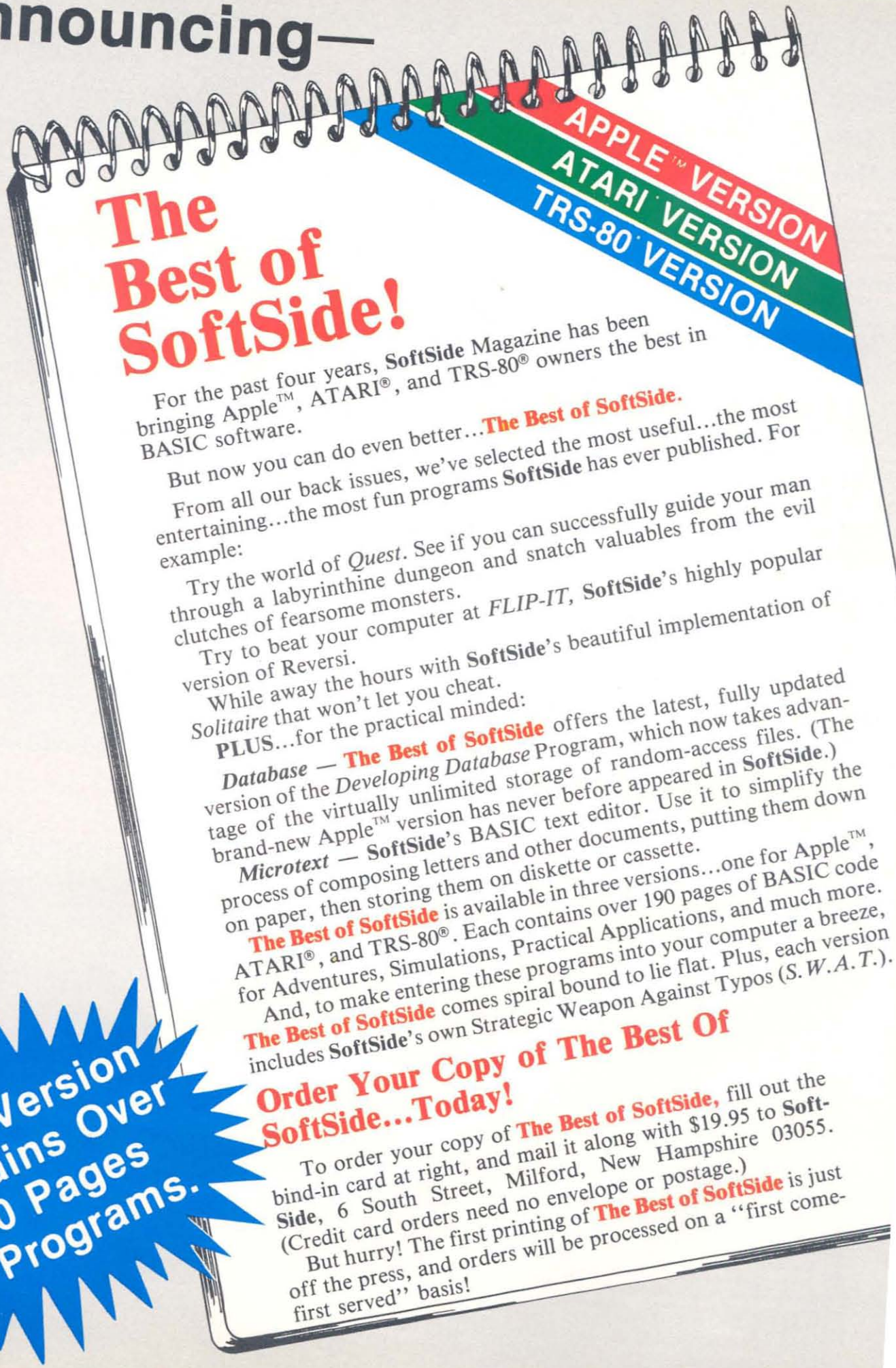
100% machine language: HI-RES graphics with sound
One player or co-operative two player operation
2 different missions on cassette (16K)
3 different missions on diskette (24K)

3 levels of difficulty
Advanced joystick control capabilities

Available at your local dealer
Direct orders add $1.50 postage and handling
Mastercharge Visa Personal Check
Diskette $32.95 Cassette $29.95

18779 Kenlake Place N.E.
Seattle, Washington 98155
(206) 644-3425

Announcing—



The Best of SoftSide!

For the past four years, **SoftSide** Magazine has been bringing Apple™, ATARI®, and TRS-80® owners the best in BASIC software.

But now you can do even better…**The Best of SoftSide.**

From all our back issues, we've selected the most useful…the most entertaining…the most fun programs **SoftSide** has ever published. For example:

Try the world of *Quest*. See if you can successfully guide your man through a labyrinthine dungeon and snatch valuables from the evil clutches of fearsome monsters.

Try to beat your computer at *FLIP-IT*, **SoftSide**'s highly popular version of Reversi.

While away the hours with **SoftSide**'s beautiful implementation of *Solitaire* that won't let you cheat.

PLUS…for the practical minded:

Database **— The Best of SoftSide** offers the latest, fully updated version of the *Developing Database* Program, which now takes advantage of the virtually unlimited storage of random-access files. (The brand-new Apple™ version has never before appeared in **SoftSide**.)

Microtext **— SoftSide**'s BASIC text editor. Use it to simplify the process of composing letters and other documents, putting them down on paper, then storing them on diskette or cassette.

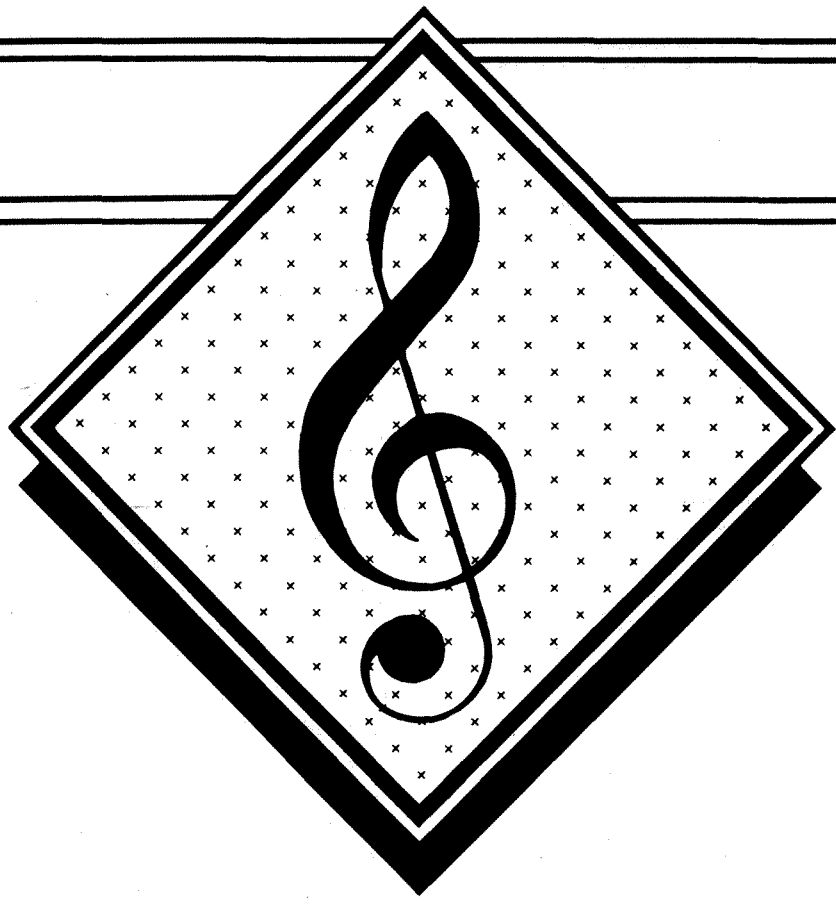
The Best of SoftSide is available in three versions…one for Apple™, ATARI®, and TRS-80®. Each contains over 190 pages of BASIC code for Adventures, Simulations, Practical Applications, and much more. And, to make entering these programs into your computer a breeze, **The Best of SoftSide** comes spiral bound to lie flat. Plus, each version includes **SoftSide**'s own Strategic Weapon Against Typos (*S.W.A.T.*).

Order Your Copy of The Best Of SoftSide…Today!

To order your copy of **The Best of SoftSide,** fill out the bind-in card at right, and mail it along with $19.95 to **SoftSide**, 6 South Street, Milford, New Hampshire 03055. (Credit card orders need no envelope or postage.)

But hurry! The first printing of **The Best of SoftSide** is just off the press, and orders will be processed on a "first come-first served" basis!

Each Version Contains Over 190 Pages Of Programs.



POKEY PLAYER II

by Craig Chamberlain

***Pokey Player II* is a music/editing/playing utility for the Atari® 400/800 with 32K and Atari BASIC.**

The POKEY chip inside your Atari computer is capable of making many sounds, but it takes some work to make them sound like music. Doing this work in BASIC means there's no time to do other things, such as graphics, while the music is playing. But what if the programming were done in Machine Language? Then the music could be played at incredibly high speeds, too fast for any listening pleasure. Delay loops could be inserted to slow it down, but this would mean that much computing time would be wasted just waiting.

The solution? Have the note processing occur as part of the system's normal vertical blank interrupt routine. The vertical blank interrupt (VBLANK) happens every sixtieth of a second, just before the television starts drawing the next frame. Every time VBLANK happens, the computer puts aside its main task, be it a BASIC program or other operation, and reads the joysticks and paddles, looks at the keyboard, goes into the "attract" mode, etc. When done with the VBLANK processing, the computer returns to its main task as if nothing had happened. VBLANK is invisible to the user.

The vertical blank interrupt would be an excellent time to do note processing. Music would appear to be running simultaneously with another program.

Presenting... *Pokey Player II*

With just a few modifications to the *Player* program published in issue 34 of ***SoftSide***, you will have updated *Pokey Player* to the second version. *Pokey Player II* plays the music during the vertical blank, and in fact, can be merged with other Atari BASIC programs. Now, you can add arcade music effects to your own games, or listen to Bach as you balance your checkbook.

How To Update *Pokey Player I*

Make the following changes and additions to the *Player* program mentioned above. Be careful to make no mistakes, as it is very easy to crash the system. (Messing with the vertical blank interrupt is a very sticky operation.)

```
1080 P=P+84:K=INT(P/256):POKE 1540,P-2
56*K:POKE 1541,K
1090 P=P+206:K=INT(P/256):POKE 1542,P-
256*K:POKE 1543,K
1500 FOR K=1564 TO 1656:READ P:POKE K,
P:NEXT K
1950 P=USR(1647)
1985 GOTO 1985
2000 DATA 173,0,6,240,76,216,169,3,141
,50,2,141,15,210,173,59,6,141,0,210,17
3,63,6,141,1,210,173,60,6,141,2,210
2002 DATA 173,64,6,141,3,210,173,61,6,
141,4,210,173,65,6,141,7,210,173,62,6,
141,6,210,173,66,6,141,5,210,173,67
2004 DATA 6,141,8,210,162,0,32,94,6,23
2,32,94,6,232,32,94,6,76,98,228,189,16
,6,133,203,189,19,6,133,204,188,28
2006 DATA 6,185,8,6,133,205,185,12,6,1
33,206,222,31,6,240,27,189,40,6,208,18
,189,31,6,221,46,6,176,10,189,63,6
2008 DATA 41,15,240,3,222,63,6,96,32,9
7,6,32,100,6,133,207,41,248,201,128,20
8,18,165,207,41,7,168,185,68,6,24,125
2160 DATA 104,169,7,162,6,160,91,76,92
,228
```

At line 1970 you may want to insert some commands to produce an interesting display while the music is playing. If you have a GTIA chip in your computer, try this:

```
1960 GRAPHICS 10
1970 POKE 705+INT(RND(0)*8),PEEK(53770
):COLOR RND(0)*16:DRAWTO RND(0)*79,RND
(0)*191
1980 IF PEEK(1536) THEN 1970
```

If for some obscure reason you have not treated yourself and your computer to the wonders of a GTIA chip, the following may be of small interest:

```
1960 POKE 53261,255:POKE 53262,255:POK
E 704,66:POKE 705,24:POKE 623,1
1970 POKE 53248,PEEK(1595):POKE 53249,
PEEK(1596):GOTO 1970
```

After making these changes and additions correctly, the *Player* program will play music just as before, with one exception. Control will be returned to BASIC right after the USR call is executed, whereas before, control was not returned until after the music had stopped playing. Therefore, lines after the USR call will be executed with the music being played concurrently.

Danger! Warning! Use Extreme Caution!

The *Player* program stores the vertical blank interrupt service routine in a string (PP$). When Atari BASIC stops (by STOP, END, or error), strings get moved around in memory. Because of this feature, it is very important that the BASIC program (the *Player*, or your own program that contains the *Player* routines) never stop after the USR call. This is true even after the music has ended! Press SYSTEM RESET to clear the vertical blank interrupt patch. Failure to heed this suggestion could result in machine lockup, requiring a cold start. It will not, however, damage your computer.

Merge

To add music to an Atari BASIC program, just append the entire *Player* program. The first few lines of *Player*, which print identification messages, can be deleted. There is also room free from 1970 to 1999, as well as before line 1000 and after the last DATA line for the third voice. (3400 and on should be free.) To start the music, execute the USR call.

Controlling Location "Active"

It is possible to freeze the music, then cause it to continue, by doing a POKE 1536,0 or POKE 1536,1. Memory location 1536 is labeled ACTIVE and is monitored for zero/nonzero values.

Final Notes

The original purpose of *Pokey Player* was to provide a means to add music to BASIC programs. That goal has been achieved. Please remember, however, that permission must be obtained before *Pokey Player* can be used in commercial programs. Contact *SoftSide* for further details.

Capriccio

SoftSide's issue 34 CV and DV included three *Pokey Player* demonstration pieces: *Happy Birthday,* Handel's March from *Scipio,* and *Capriccio.* Space limitations allowed us to publish only the first two in the magazine. This month we present the third piece, which can be found immediately following this article. It is listed separately as a data file running from lines 3100 on up. To use the data file, first LOAD the *Player* program into memory, then ENTER in the data file. Be sure the data file is in the LIST format. After *Player* has loaded, type ENTER "D:CAP" (or ENTER "C:" for cassette) to merge in the *Capriccio* music data. A simple RUN command will now start the music.

SPECIAL NOTE!

Craig Chamberlain and Harry Bratt, authors of the *Pokey Player* programs, are most anxious to receive any music files created using *Pokey Player.* Please send all correspondence to *SoftSide.*

```
SS SS SS SS SS SS SS SS SS SS SS
SS         Atari BASIC        SS
SS         "Capriccio"        SS
SS      Music by: Handel      SS
SS  Arranged by: Harry Bratt  SS
SS     Copyright (c) 1982     SS
SS SoftSide Publications, Inc SS
SS SS SS SS SS SS SS SS SS SS SS
3100 DATA 744,32,4,8,16,32,64,128,0,2,
132,82,18,82,26,90,58,98,58,132,34,10,
74,26,90,50,90,58,132,26,18,82,34,98
3102 DATA 58,90,59,132,27,19,83,27,35,
107,128,99,19,132,19,137,201,137,201,1
0,74,68,3,27,84,3,35,84,3,27,75,27
3104 DATA 35,83,26,132,34,18,82,26,90,
58,98,58,132,34,10,74,26,90,128,98,90,
11,35,108,68,12,74,98,18,82,34,98,58
3106 DATA 90,58,132,66,10,74,26,90,50,
90,58,132,66,18,82,26,90,58,98,59,91,1
15,91,83,35,59,83,60,3,128,99,75,99
3108 DATA 107,91,128,83,91,115,91,128,
83,91,107,99,128,75,99,107,91,2,82,18,
82,26,106,18,82,34,82,18,10,18,106
3110 DATA 18,82,26,90,18,82,26,90,58,9
8,58,132,66,18,82,34,98,58,90,50,132,6
6,18,82,34,98,58,90,58,132,66,10,74
3112 DATA 26,90,58,98,58,132,66,18,82,
26,90,58,98,58,132,34,10,74,128,82,132
,66,18,10,58,132,34,10,74,128,82,132
3114 DATA 66,18,10,58,132,34,10,74,128
,82,132,26,18,82,128,74,132,26,18,82,1
28,82,132,34,18,82,128,82,132,66,18
3116 DATA 82,26,90,18,10,75,128,67,99,
91,19,128,67,99,91,20,84,84,84,75,128,
99,91,107,59,99,107,91,128,82,132,66
3118 DATA 18,82,26,90,58,98,58,132,34,
10,74,26,90,50,90,58,132,26,18,82,34,9
8,58,90,58,132,26,18,82,26,90,58,98
3120 DATA 58,132,34,10,74,26,90,58,98,
58,132,26,18,82,58,132,34,10,74,128,82
,132,26,18,82,58,132,34,10,74,52,76
3122 DATA 60,4,132,28,4,19,99,75,27,83
,163,68,70,68,3,99,76,4,2,82,18,82,26,
90,58,98,58,132,66,18,82,34,98,58,90
3124 DATA 50,132,66,18,82,34,98,58,90,
58,132,66,10,74,26,90,58,98,58,132,66,
18,82,26,90,58,98,59,75,84,60,4,2,106
3126 DATA 10,74,42,106,128,98,90,147,6
8,27,84,108,60,84,106,90,10,74,26,106,
18,10,20,4,68,4,67,83,76,128,83,83
3128 DATA 76,84,84,60,108,60,107,11,20
,84,109,69,90,66,10,74,26,82,26,90,18,
90,10,18,123,128,67,107,83,75,128,67
3130 DATA 83,83,76,83,83,84,76,12,76,1
2,75,11,75,11,19,19,83,59,108,4,132,27
,91,107,99,128,76,12,76,12,75,11,75
3132 DATA 11,19,19,83,59,106,98,18,82,
34,98,58,90,58,98,18,82,34,98,58,90,51
,132,35,59,132,27,58,132,98,18,82,74
3134 DATA 128,98,18,18,124,59,107,27,1
```

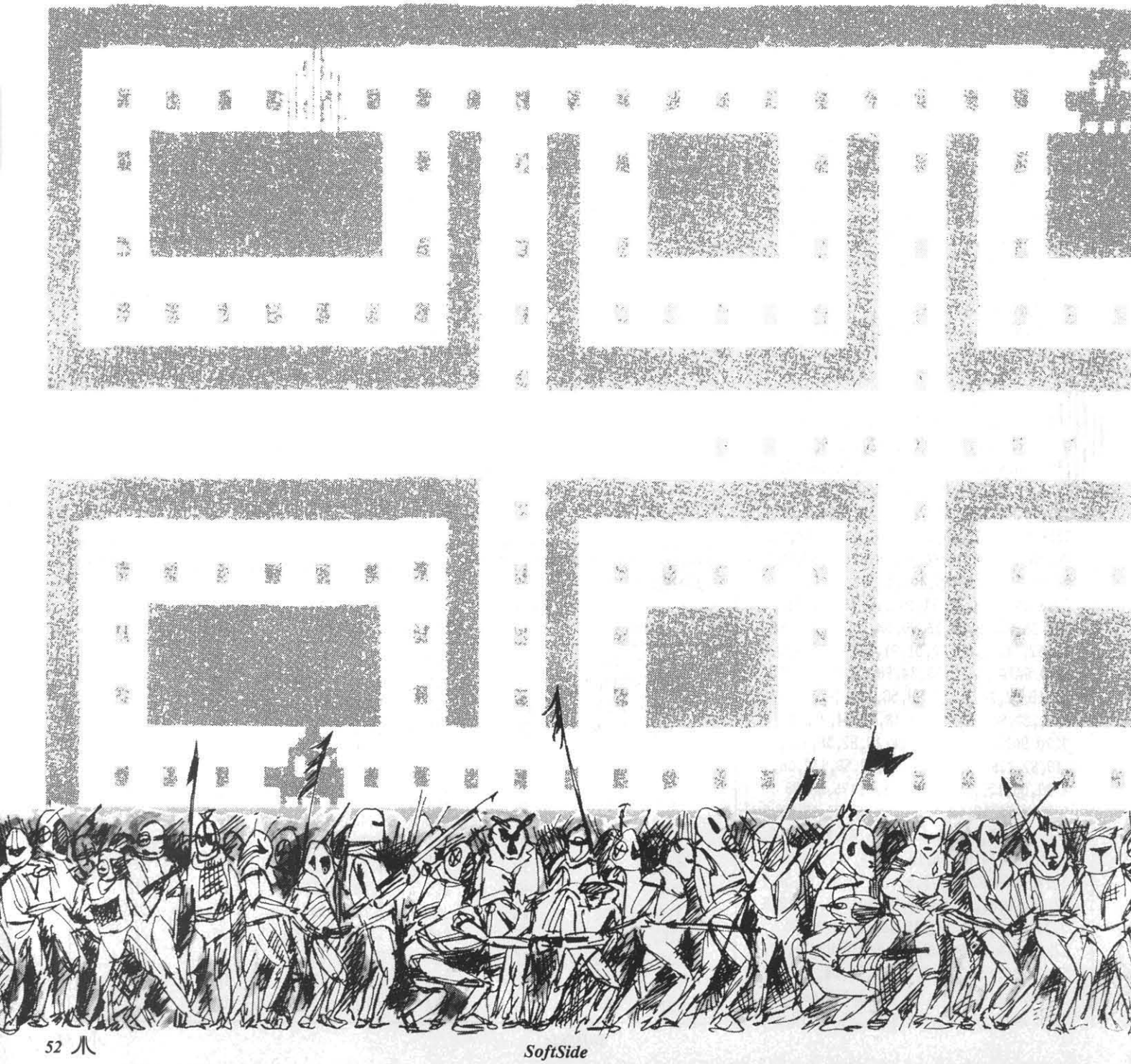
```
07,28,76,28,84,44,76,84,83,75,84,132,2
8,4,128,67,35,83,91,11,35,83,91,11
3136 DATA 35,83,91,12,3,128,99,76,3,91
,83,91,12,107,128,67,91,27,75,99,107,9
1,128,82,132,66,18,82,26,90,58,98,58
3138 DATA 132,66,18,82,34,98,58,90,50,
132,66,18,82,34,98,58,90,58,98,18,82,3
4,98,58,90,58,132,66,10,74,26,90,50
3140 DATA 90,58,132,26,18,82,34,98,58,
90,58,132,26,18,82,26,90,58,98,58,132,
34,10,74,26,90,58,98,58,132,26,18,82
3142 DATA 58,132,34,10,74,58,132,26,18
,82,58,132,26,18,82,58,132,34,10,74,50
,132,34,18,82,59,83,84,76,128,68,84
3144 DATA 68,98,34,82,82,74,82,82,74,1
28,84,68,98,34,82,82,74,82,82,74,83,59
,44,132,110,120,6
3200 DATA 977,6,6,6,2,132,122,10,74,26
,90,58,98,58,132,26,10,74,34,98,58,90,
58,132,26,18,82,26,90,58,98,59,132
3202 DATA 35,128,83,132,27,75,43,83,91
,84,3,67,36,42,132,26,18,82,34,98,58,9
0,128,82,132,66,18,82,26,90,18,82,26
3204 DATA 90,58,98,58,132,66,18,82,34,
98,58,90,50,132,66,18,82,34,98,58,90,5
8,132,66,10,74,26,90,50,90,58,98,74
3206 DATA 128,98,146,67,90,10,90,18,82
,26,90,58,98,58,132,66,18,82,34,98,58,
90,58,132,66,10,74,26,90,50,90,58,132
3208 DATA 66,18,82,26,90,58,98,58,132,
66,18,82,34,98,58,90,52,76,68,12,75,99
,91,27,35,107,99,91,128,83,107,99,91
3210 DATA 128,83,107,91,99,128,18,132,
90,10,74,26,90,50,90,58,132,26,18,82,1
28,74,132,66,10,18,58,132,26,18,82
3212 DATA 128,74,132,66,10,18,58,132,2
6,18,82,128,74,132,26,18,82,128,82,132
,34,10,74,128,82,132,26,18,82,128,74
3214 DATA 132,66,10,74,26,90,10,18,82,
90,58,98,128,90,90,10,18,82,90,58,98,1
28,90,98,18,18,82,106,18,82,26,106
3216 DATA 18,82,34,106,10,74,26,106,18
,82,26,90,18,82,26,90,58,98,58,132,66,
18,82,34,98,58,90,51,91,83,91,58,132
3218 DATA 26,18,82,34,98,58,90,50,132,
34,18,82,26,90,58,98,58,132,34,10,74,2
6,90,50,90,58,132,26,18,82,34,98,58
3220 DATA 90,58,132,26,18,82,58,132,26
,18,82,128,74,132,26,18,82,58,132,26,1
8,82,58,122,18,18,12,76,4,196,66,98
3222 DATA 18,18,10,90,18,82,82,18,82,3
4,82,26,18,82,74,10,74,26,82,90,18,82,
82,18,82,34,82,26,18,82,74,10,74,26
3224 DATA 82,90,18,82,26,90,58,98,58,1
32,66,18,82,34,98,58,90,50,132,66,18,8
2,34,98,58,90,58,132,66,10,74,26,90
3226 DATA 58,98,58,132,66,18,82,26,90,
58,98,58,132,66,18,82,34,98,58,90,58,1
32,66,10,74,26,90,50,90,58,132,66,18
3228 DATA 10,145,209,145,209,18,18,66,
42,18,82,26,90,58,98,60,4,2,106,82,74,
212,66,66,18,10,202,67,82,156,66,66
3230 DATA 106,18,18,98,18,82,34,106,10
,18,82,34,10,74,26,106,18,10,74,98,18,
82,34,106,10,18,82,106,18,82,42,90
3232 DATA 18,10,76,4,52,76,68,12,68,14
8,66,90,10,18,2,132,34,10,18,18,98,18,
82,34,82,26,90,18,66,10,74,26,82,26
3234 DATA 90,18,122,18,82,34,82,26,90,
18,98,18,18,12,18,132,66,18,82,42,90,1
8,10,76,4,2,82,18,82,42,90,10,18,82
3236 DATA 90,18,82,42,106,18,82,26,90,
18,82,42,106,18,82,26,90,42,106,26,90,
42,106,58,122,128,98,132,34,58,122
3238 DATA 58,114,76,4,2,106,10,74,42,9
8,18,18,82,90,10,74,42,106,10,74,26,90
,10,74,42,106,10,74,26,90,42,106,26
3240 DATA 90,42,106,58,114,58,132,34,5
8,122,58,114,58,90,18,82,26,90,58,98,5
8,90,10,74,26,90,58,98,58,132,66,18
3242 DATA 82,82,18,18,10,18,106,18,82,
26,90,18,10,18,132,66,18,82,82,18,82,3
4,106,10,74,26,106,128,66,18,82,82
3244 DATA 18,82,34,114,18,82,34,106,10
,74,42,122,18,82,58,132,26,18,82,128,9
0,132,18,10,74,128,82,132,67,83,128
3246 DATA 209,201,137,201,10,74,66,98,
18,82,74,10,74,128,98,10,132,34,18,82,
74,10,74,128,98,11,75,68,2,122,18,82
3248 DATA 74,10,74,128,98,10,90,18,82,
58,132,18,18,10,74,98,18,82,34,98,58,9
0,106,10,18,82,35,10,18,66,90,18,82
3250 DATA 26,90,58,98,58,132,66,18,82,
34,98,58,90,51,91,107,99,128,74,132,66
,10,74,26,90,58,98,58,132,66,18,82
3252 DATA 26,90,58,98,58,90,18,82,26,9
0,58,98,58,132,66,18,82,34,98,58,90,50
,132,34,18,82,26,90,58,98,58,132,34
3254 DATA 10,74,26,90,50,90,58,132,26,
18,82,34,98,58,90,58,132,26,18,82,58,1
32,26,18,82,58,132,34,10,74,50,132
3256 DATA 34,18,82,58,132,26,18,82,58,
132,26,18,82,58,132,34,10,74,42,98,18,
18,82,42,74,42,82,82,74,128,98,82,34
3258 DATA 82,82,74,82,82,74,2,50,74,82
,82,18,18,82,26,34,82,82,74,82,82,74,2
,50,74,82,82,18,18,82,27,76,11,38,6
3300 DATA 330,16,5,64,48,6,0,3,115,75,
99,76,3,67,28,128,84,20,68,6,6,4,2,132
,106,11,76,4,6,6,129,68,84,60,108,108
3302 DATA 4,5,16,5,64,53,6,0,2,106,10,
74,26,106,18,82,27,107,20,16,4,64,46,6
,0,4,5,2,82,18,18,10,90,18,82,26,90
3304 DATA 58,98,2,90,10,18,84,76,4,2,4
2,18,82,82,18,82,34,82,26,18,82,74,10,
74,26,82,90,18,82,82,18,82,34,84,4
3306 DATA 5,6,6,5,2,132,26,18,18,67,11
,76,6,2,106,10,18,12,2,34,234,67,26,82
,74,12,76,4,128,90,98,18,82,34,106
3308 DATA 10,18,82,34,10,74,26,106,18,
10,76,3,99,2,132,18,18,82,42,90,10,18,
2,18,18,82,26,106,18,82,34,82,18,10
3310 DATA 18,106,18,82,26,74,10,18,18,
98,18,18,12,132,28,76,4,2,42,18,82,34,
82,26,90,5,3,124,11,5,2,114,18,82,42
3312 DATA 90,10,18,84,4,35,27,3,19,3,8
3,3,19,6,132,26,90,18,82,26,90,58,98,5
,3,19,3,19,3,83,3,19,6,132,108,4,5
3314 DATA 6,128,50,66,18,82,74,10,74,2
6,106,18,82,26,107,35,68,60,84,68,68,1
00,76,3,83,19,27,107,128,91,3,99,67
3316 DATA 107,26,106,18,82,74,10,74,12
8,98,11,75,68,84,3,59,108,4,195,68,11,
16,7,64,50,6,0,5,84,11,18,18,132,125
3318 DATA 128,68,27,18,18,132,125,66,1
8,10,74,28,128,86,6
```

ATARI® SWAT TABLE FOR: CAPRICCIO (Modified Parameters: NU = 2, B = 500)

LINES	SWAT CODE	LENGTH	LINES	SWAT CODE	LENGTH
3100 - 3102	AU	214	3218 - 3220	LM	214
3104 - 3106	FZ	215	3222 - 3224	LW	214
3108 - 3110	BP	213	3226 - 3228	OM	215
3112 - 3114	DU	215	3230 - 3232	HU	213
3116 - 3118	HT	215	3234 - 3236	IC	213
3120 - 3122	GE	215	3238 - 3240	KZ	214
3124 - 3126	GD	215	3242 - 3244	IP	213
3128 - 3130	FV	215	3246 - 3248	OB	215
3132 - 3134	GP	214	3250 - 3252	OL	213
3136 - 3138	IW	215	3254 - 3256	NC	214
3140 - 3142	HP	215	3258 - 3300	SO	216
3144 - 3200	XA	195	3302 - 3304	MG	213
3202 - 3204	NJ	215	3306 - 3308	LW	213
3206 - 3208	TC	217	3310 - 3312	ON	214
3210 - 3212	IZ	214	3314 - 3316	TK	215
3214 - 3216	LG	214	3318 - 3318	GA	53

MUNCHKIN ATTA

by David N.



ACK

Plotkin

Munchkin Attack is an arcade-style game for an Atari® with 16K RAM (24K disk) and a joystick.

Munchkin Attack is similar to *TRS-Man*, which appeared in the January, 1982 issue of ***SoftSide***. It makes use, however, of the Atari's unique features, including Player/Missile (PM) Graphics and nine colors on the screen at once. Thus, this is more of an adaptation than a *translation*.

The player controls a little mouth as it moves around the screen, getting points for consuming dots while avoiding one to four computer-controlled creatures that try to munch on the mouth. If the creatures catch the mouth three times, then the game ends. There are, however, four special, yellow dots on the screen. When the mouth eats one of these, all the creatures turn blue for a short period of time, during which the mouth may eat the creatures for some extra points and a bit of revenge. A bell sounds to warn the player just before the creatures return to their normal color.

Variables

A, I, N, W, WW: Loop and miscellaneous variables.
AA: Memory base of PM graphics in pages.
C0: Counter for setting and resetting creature colors.
D, DI(i): Direction of mouth (D) and creatures — 1 = up, 2 = down, 3 = left, 4 = right.
DP: Temporary direction variable for mouth's direction.
L: Used to determine program branch target location.
NEW: Memory location (vertical) of creatures.
NUMC: Number of creatures chasing you.
SC: Score.
ML: Mouths left.
NUMD: Number of dots eaten.
PB: *Memory base of PM graphics* in bytes.
X0, Y0: X and Y co-ordinates (PM) of mouth.
X(i), Y(i): X and Y co-ordinates (PM) of creatures.
XD, YD: Distance between mouth and creatures.
XR, YR: Temporary variables for storing the X and Y co-ordinates of mouth and creature in graphics co-ordinates.
Z, Z1, Z2, Z3, Z4: LOCATE variables; test for a collision.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS        Atari BASIC          SS
SS     'Munchkin Attack'       SS
SS   Author: David Plotkin     SS
SS     Copyright (c) 1982      SS
SS SoftSide Publications, Inc SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type in this program, it is available on this month's SoftSide CV and DV.

```
1 GOTO 10
```

Subroutine to change horizontal position register of creatures. Placed at front of program to run faster.

```
2 POKE 53248+N+3*(N=4),X(N):IF N=4 THE
N POKE 53254,X(4)+2:POKE 53253,X(4)+4:
POKE 53252,X(4)+6
3 RETURN
```

Initialization.

```
10 GOSUB 1300:GOSUB 1000
20 DIM DI(5),X(5),Y(5):DI(0)=0:X(0)=0:
Y(0)=0:FL=0:C0=0
30 SC=0:NUMC=0:ML=3
40 NUMD=0:NUMC=NUMC+1:POKE 53278,1:IF
NUMC>4 THEN NUMC=4
45 FL=0:C0=0
50 ST=7:DI(1)=4:DI(2)=3:DI(3)=3:DI(4)=
4:X0=124:Y0=52:X(1)=72:Y(1)=20:X(2)=17
6:Y(2)=84:POKE 53248,X0
60 X(3)=176:Y(3)=20:X(4)=72:Y(4)=84:R=
USR(1536,PB+512+Y0,300)
70 FOR N=1 TO NUMC:POKE 53248+N+3*(N=4
),X(N):R=USR(1536,PB+512*(N(4)+384*(N=
4)+128*N*(N(4)+Y(N),260):NEXT N
80 IF NUMC=4 THEN POKE 53254,X(4)+2:PO
KE 53253,X(4)+4:POKE 53252,X(4)+6
```

Test joystick position.

```
100 XR=(X0-48)/2:YR=(Y0-16)/2:STO=ST:S
T=STICK(0)-4:GOTO 120
```

If joystick was trying to move the mouth in a direction that would cause it to collide with a wall, then ignore the joystick direction, and, if possible, continue in the current direction.

```
110 ST=STO
```

Go to the appropriate direction subroutine.

```
120 ON ST GOTO 190,140,310,110,190,140
,240,110,190,140,110
```

Move mouth up if possible. Award points for eaten dots.

```
140 LOCATE XR+2,YR-1,Z:LOCATE XR,YR-1,
Z1:LOCATE XR+3,YR-1,Z2:LOCATE XR+2,YR-
2,Z3
150 IF (Z=1 OR Z1 OR Z2) AND ST<>STO T
HEN 110
155 IF Z=1 OR Z1 OR Z2 THEN 670
160 DD=USR(1536,PB+512+Y0,316)
170 Y0=Y0-4:R=USR(1536,PB+512+Y0,284):
IF Z=2 OR Z3=2 THEN COLOR 0:PLOT XR+2,
YR-(Z=2)-2*(Z3=2):SC=SC+10:GOTO 620
180 GOTO 670
```

Move mouth down if possible. Award points for eaten dots.

```
190 LOCATE XR+2,YR+4,Z:LOCATE XR,YR+4,
Z1:LOCATE XR+3,YR+4,Z2:LOCATE XR+2,YR+
5,Z3
200 IF (Z=1 OR Z1 OR Z2) AND ST<>STO T
HEN 110
205 IF Z=1 OR Z1 OR Z2 THEN 670
210 R=USR(1536,PB+512+Y0,316)
220 Y0=Y0+4:R=USR(1536,PB+512+Y0,292):
IF Z=2 OR Z3=2 THEN COLOR 0:PLOT XR+2,
YR+4*(Z=2)+5*(Z3=2):SC=SC+10:GOTO 620
230 GOTO 670
```

Move mouth left if possible; test for wrap-around. If the mouth eats a blue dot, then award points. If the mouth eats a yellow dot, then award points, turn creatures blue, and start the timer.

```
240 LOCATE XR-1,YR+2,Z:LOCATE XR-1,YR,
Z1:LOCATE XR-1,YR+3,Z2:LOCATE XR-2,YR+
2,Z3
250 IF (Z1 OR Z2) AND ST<>STO THEN 110
255 IF Z1 OR Z2 THEN 670
260 R=USR(1536,PB+512+Y0,300):X0=X0-4:
IF X0<=50 THEN X0=196
270 POKE 53248,X0:IF Z=2 OR Z3=2 THEN
COLOR 0:PLOT XR-(Z=2)-2*(Z3=2),YR+2:SC
=SC+10:GOTO 620
280 IF Z=0 AND Z3=0 THEN 670
290 COLOR 0:PLOT XR-(Z=1)-2*(Z3=1),YR+
2:SC=SC+100:FL=1:CO=20:POKE 705,115:PO
KE 706,115:POKE 707,115:POKE 711,115
300 GOTO 620
```

Same as lines 240-300, but for rightward motion.

```
310 LOCATE XR+4,YR+2,Z:LOCATE XR+4,YR+
3,Z1:LOCATE XR+4,YR,Z2:LOCATE XR+5,YR+
2,Z3
320 IF (Z1 OR Z2) AND ST<>STO THEN 110
325 IF Z1 OR Z2 THEN 670
330 R=USR(1536,PB+512+Y0,308):X0=X0+4:
IF X0>=198 THEN X0=52
340 POKE 53248,X0:IF Z=2 OR Z3=2 THEN
COLOR 0:PLOT XR+4*(Z=2)+5*(Z3=2),YR+2:
SC=SC+10:GOTO 620
350 IF Z=0 AND Z3=0 THEN 670
360 COLOR 0:PLOT XR+4*(Z=1)+5*(Z3=1),Y
R+2:SC=SC+100:FL=1:CO=20:POKE 705,115:
POKE 706,115:POKE 707,115:POKE 711,115
370 GOTO 620
```

Eating sound.

```
620 SOUND 0,180,12,8:FOR W=1 TO 2:NEXT
 W:SOUND 0,150,12,8:FOR W=1 TO 2:NEXT
W:SOUND 0,0,0,0
```

"Dots eaten" counter. Check if all the dots have been ingested.

```
630 NUMD=NUMD+1:IF NUMD<151 THEN GOTO
660
```

When all the dots have been eaten, erase the creatures, mouth, dots, and board; then redraw them.

```
640 DD=USR(1536,PB+512+Y0,316):DD=USR(
1536,PB+640+Y(1),316):DD=USR(1536,PB+7
68+Y(2),316)
650 DD=USR(1536,PB+896+Y(3),316):DD=US
R(1536,PB+384+Y(4),316):GOSUB 1070:GOT
O 40
```

Print new score.

```
660 POKE 656,1:POKE 657,12:? SC;
```

The creatures' logic for chasing the mouth.

```
670 FOR N=1 TO NUMC:XD=X0-X(N):YD=Y0-Y
(N):XR=(X(N)-48)/2:YR=(Y(N)-16)/2
```

Wrap-around for creatures.

```
680 IF DI(N)=3 AND X(N)<=50 THEN X(N)=
196:GOSUB 2:NEXT N:GOTO 935
690 IF DI(N)=4 AND X(N)>=198 THEN X(N)
=52:GOSUB 2:NEXT N:GOTO 935
695 LET NEW=PB+512*(N<4)+384*(N=4)+N*1
28*(N<4)
700 L=750*(YD=0 AND XD>0)+730*(YD=0 AN
D XD<0)+830*(XD=0 AND YD>0)+800*(XD=0
AND YD<0):GOTO L*(L<>0)+860*(L=0)
```

Creature chases mouth left, if possible.

```
730 LOCATE XR-1,YR,Z1:LOCATE XR-1,YR+3
,Z2:IF Z1 OR Z2 THEN ON DI(N) GOTO 870
,885,895,905
740 X(N)=X(N)-4:DI(N)=3:GOSUB 2:NEXT N
:GOTO 935
```

Creature chases mouth right, if possible.

```
750 LOCATE XR+4,YR+3,Z1:LOCATE XR+4,YR
,Z2:IF Z1 OR Z2 THEN ON DI(N) GOTO 870
,885,895,905
760 X(N)=X(N)+4:DI(N)=4:GOSUB 2:NEXT N
:GOTO 935
```

Creature chases mouth up, if possible.

```
800 LOCATE XR,YR-1,Z1:LOCATE XR+3,YR-1
,Z2:IF Z1 OR Z2 THEN ON DI(N) GOTO 870
,885,895,905
820 R=USR(1536,NEW+Y(N),316):Y(N)=Y(N)
-4:DI(N)=1:R=USR(1536,NEW+Y(N),260):NE
XT N:GOTO 935
```

Creature chases mouth down, if possible.

```
830 LOCATE XR,YR+4,Z1:LOCATE XR+3,YR+4
,Z2:IF Z1 OR Z2 THEN ON DI(N) GOTO 870
,885,895,905
850 R=USR(1536,NEW+Y(N),316):Y(N)=Y(N)
+4:DI(N)=2:R=USR(1536,NEW+Y(N),260):NE
XT N:GOTO 935
860 ON DI(N) GOTO 870,885,895,905
```

Creature continues up, if possible.

```
870 LOCATE XR,YR-1,Z1:LOCATE XR+3,YR-1
,Z2:IF Z1 OR Z2 THEN DI(N)=INT(RND(0)*
4+1):NEXT N:GOTO 935
880 GOTO 820
```

Creature continues down, if possible.

```
885 LOCATE XR,YR+4,Z1:LOCATE XR+3,YR+4
,Z2:IF Z1 OR Z2 THEN DI(N)=INT(RND(0)*
4+1):NEXT N:GOTO 935
890 GOTO 850
```

Creature continues left, if possible.

```
895 LOCATE XR-1,YR,Z1:LOCATE XR-1,YR+3
,Z2:IF Z1 OR Z2 THEN DI(N)=INT(RND(0)*
4+1):NEXT N:GOTO 935
900 GOTO 740
```

Creature continues right, if possible.

```
905 LOCATE XR+4,YR+3,Z1:LOCATE XR+4,YR
,Z2:IF Z1 OR Z2 THEN DI(N)=INT(RND(0)*
4+1):NEXT N:GOTO 935
910 GOTO 760
```

Decrement timer. When timer reaches zero, reset the creature colors.

```
935 IF FL=1 THEN CO=CO-1:IF CO<=0 THEN
 FL=0:POKE 705,85:POKE 706,100:POKE 70
7,195:POKE 711,52
```

Warning bell.

```
937 IF CO=5 THEN SOUND 0,50,10,8:FOR W
=1 TO 5:NEXT W:SOUND 0,0,0,0
```

continued on page 56

Lyco Computer Marketing & Consultants

TO ORDER CALL US

TOLL FREE 800-233-8760

In PA 1-717-398-4079

ATARI SPECIALS

810 Disk Drive ... $429.00
32K RAM ..... $ 79.00
400 32K RAM ... $179.00

800 48K... $539.00

ATARI®
A Warner Communications Company

PERCOM : In Stock

Single Drive	CALL
Dual Drive	CALL

(Read all Atari Disks)

PRINTERS

Okidata 82A	$479.00
Okidata 83A	$719.00
Okidata 84	$1089.00
Citoh	CALL
Prowriter I	$499.00
Prowriter II	CALL
SMITH CORONA TP-1	$625.00
NEC	CALL

(Interfacing Available)

JOYSTICKS : In Stock

Atari CX-40	$18.00
LeStick	$34.00
Wico Command Control	$24.00
WICO RED BALL	$27.95
STICK STAND	$ 6.75

Computer Covers

800	$6.99
400	$6.99
810	$6.99

DISKETTES : In Stock

Maxell MD1 . . .(10)	$34.00
Maxell MD2 . . .(10)	$44.00
Elephant . . .(10)	$21.00

THIRD PARTY SOFTWARE ATARI PROGRAM EXCHANGE

Eastern Front 1941	$25.50
Avalanche	$15.50
Outlaw/Howitzer	$15.50
Dog Daze	$15.50
Wizard of War	$31.00
Gorf	$31.00
Frogger	$26.00

BUSINESS SOFTWARE : In Stock

Atari Word Processing	$109.00
Letter Perfect	$129.00
Test Wizzard	$ 89.00
Datasam/65	$125.00
Interlisp	$125.00
Monkey Wrench	$ 42.00
Utility Disk	$ 36.50
Ultimate Renumber	$ 15.50

ATARI HARDWARE

410 Cassette Recorder	$75.00
825 Printer	$585.00
830 Phone Modem	$149.00
850 Interface	$164.00

PACKAGES

CX481 Entertainer	$69.00
CX482 Educator	$125.00
CX483 Programmer	$49.00
CX494 Communicator	$325.00

SOFTWARE

CXL4012	MISSILE COMMAND	$28.75
CXL4013	ASTEROID	$28.75
CXL4020	CENTIPEDE	$32.75
CXL4022	PACMAN	$32.75
CXL4011	STAR RAIDER	$34.75
CXL4004	BASKETBALL	$26.75
CXL4006	SUPER BREAKOUT	$28.75
CXL4008	SPACE INVADER	$28.75
CX8130	CAVERNS OF MARS	$31.75
CX4108	HANGMAN	$12.75
CX4102	KINGDOM	$12.75
CX4112	STATES & CAPITALS	$12.75
CX4114	EUROPEAN COUNTRIES	$12.75
CX4109	GRAPHIT	$16.75
CX4121	ENERGY CZAR	$12.75
CX4123	SCRAM	$19.75
CX4101	PROGRAMMING I	$19.75
CX4106	PROGRAMMING II	$22.75
CX4117	PROGRAMMING III	$22.75
CXL4015	TELELINK	$21.75
CX4119	FRENCH	$39.75
CX4118	GERMAN	$39.75
CX4120	SPANISH	$39.75
CX4120	SPANISH	$39.75
CXL4007	MUSIC COMPOSER	$33.75
CXL4002	ATARI BASIC	$45.75
CX8126	MICROSOFT BASIC	$65.75
CXL4003	ASSEMBLER EDITOR	$45.75
CX8126	MACROASSEMBLER	$69.75
CXL4018	PILOT HOME	$65.75
CX405	PILOT EDUCATOR	$99.75
CX415	HOME FILING MANAGER	$41.75
CX414	BOOKEEPER	$119.75

NEW RELEASES

CHOP LIFTER	$27.75
APPLE PANIC	$23.75
PREPPIE	$19.95

THIRD PARTY SOFTWARE for atari 800 or 400

K-BYTE

KRAZY SHOOTOUT	$35.00
K-DOS	$65.00
K-STAR PATROL	$37.75
K-RAZY ANTICS	$37.75
K-RAZY KRITTERS	$37.75
Q-BALL JOYSTICK KIT	$6.75

AUTOMATIED SIMULATIONS

Star Warrior	$28.00
Crush, Crumble & Chomp	$23.00

WE CARRY MANY OTHER THIRD PARTY PRODUCTS YOU CAN CALL FOR PRICES ON AND ASK FOR YOUR FREE ATARI PRODUCT CATALOG.

commodore

VIC-20 ..... $189.00

VIC1530 DATASSETTE	$67.00
VIC1540 DISK DRIVE	$499.00
VIC1515 PRINTER	$355.00
VIC1210 3K RAM	$35.00
VIC1110 8K RAM	$52.00
VIC1211A SUPER EXPANDER	$53.00

VIC-20 SOFTWARE

VIC1212 PROGRAMMER AID	$45.00
VIC1213 VICMON	$45.00
VIC1906 SUPER ALIEN	$23.00
VIC1914 ADVENTURE LAND ADVENTURE	$35.00
VIC1915 PRIVATE COVE ADVENTURE	$35.00
VIC1916 MISSION IMPOSSIBLE	$35.00
VIC1917 THE COUNT ADVENTURE	$35.00
VIC1919 SARGON II CHESS	$35.00

THIRD PARTY SOFTWARE

ALIEN BLITZ	$21.00
Omega Race	$35.00
Gorf	$32.00
16K RAM/ROM	$99.00
AMOK	$21.00
SUPER HANGMAN	$16.00
SPIDERS OF MARS	$45.00

MasterCard POLICY VISA

In-Stock items shipped within 24 hours of order Personal checks require four weeks clearance before shipping. PA residents add sales tax. All products subject to availability and price change. Add 4 % for Mastercard and Visa.

TO ORDER CALL TOLL FREE
800-233-8760
In PA 1-717-398-4079
or send order to
Lyco Computer
P.O. Box 5088
Jersey Shore, PA 17740

Munchkin Attack *continued*

Test for a hit between a creature and the mouth.

```
940 IF PEEK(53259)=1 OR PEEK(53256)=1
THEN GOTO 945
941 IF PEEK(53260)=0 THEN POKE 53278,1
:GOTO 100
```

If creatures are blue, then branch around the next section of code to line 965.

```
945 IF FL=1 THEN GOTO 965
```

Decrement the number of mouths left. Make the sound signifying the loss of a mouth. If all the mouths are gone, jump to the end-of-game routine, otherwise erase the creatures and mouth, and place them in their starting positions.

```
950 ML=ML-1:FOR W=10 TO 60 STEP 10:SOU
ND 0,W,10,8:FOR WW=1 TO 50:NEXT WW:NEX
T W:SOUND 0,0,0,0:IF ML=0 THEN GOTO 13
50
955 FOR N=1 TO NUMC:R=USR(1536,PB+512*
(N<4)+384*(N=4)+128*N*(N<4)+Y(N),316):
NEXT N:R=USR(1536,PB+512+Y0,316)
960 POKE 53278,1:GOTO 50
```

Erase an eaten creature, and place it back in its starting location. Increment score.

```
965 IF PEEK(53260)<>0 THEN GOTO (970+5
*(PEEK(53260)=4)+10*(PEEK(53260)=8))
967 IF PEEK(53259)=1 OR PEEK(53256)=1
THEN GOTO 985
970 R=USR(1536,PB+640+Y(1),316):Y(1)=2
0:X(1)=72:POKE 53249,72:R=USR(1536,PB+
640+Y(1),260):GOTO 992
975 R=USR(1536,PB+768+Y(2),316):Y(2)=8
4:X(2)=176:POKE 53250,176:R=USR(1536,P
B+768+Y(2),260):GOTO 992
980 R=USR(1536,PB+896+Y(3),316):Y(3)=2
0:X(3)=176:POKE 53251,176:R=USR(1536,P
B+896+Y(3),260):GOTO 992
985 R=USR(1536,PB+384+Y(4),316):Y(4)=8
4:X(4)=72:POKE 53255,72:POKE 53254,72:
POKE 53253,74:POKE 53252,76
990 R=USR(1536,PB+384+Y(4),260)
992 POKE 53278,1:SC=SC+INT(RND(0)*4+1)
*200:POKE 656,1:POKE 657,12:? SC;
995 SOUND 0,10,10,8:FOR W=1 TO 50:NEXT
 W:SOUND 0,0,0,0:GOTO 100
```

Initialize PM graphics. Draw board.

```
1000 AA=PEEK(106)-12:PB=256*AA
1010 FOR A=PB TO PB+1024:POKE A,0:NEXT
A
1020 FOR A=1536 TO 1560:READ I:POKE A,
I:NEXT A
1030 DATA 104,104,133,204,104,133,203,
104,133,207,104,133,206,160,0,177,206,
145,203,200,192,8,208,247,96
1040 FOR A=260 TO 323:READ I:POKE A,I:
NEXT A
1050 DATA 24,60,126,90,219,255,255,170
,24,60,126,90,219,255,255,85
1060 DATA 24,60,126,255,255,126,60,24,
68,70,199,239,239,126,60,24
1065 DATA 24,60,126,239,239,199,70,68,
56,124,254,31,7,30,252,56,28,62,127,24
8,224,120,63,28,0,0,0,0,0,0,0,0
1070 GRAPHICS 5:POKE 710,0:POKE 752,1:
POKE 559,46:POKE 54279,AA
1080 POKE 53277,3:POKE 623,17
1090 POKE 704,18:POKE 705,85:POKE 706,
100:POKE 707,195:POKE 709,165:POKE 711
,52
1100 COLOR 1:PLOT 0,0:DRAWTO 79,0:PLOT
 0,1:DRAWTO 79,1:PLOT 0,39:DRAWTO 79,3
9:PLOT 0,38:DRAWTO 79,38
1110 PLOT 0,0:DRAWTO 0,17:DRAWTO 25,17
:DRAWTO 25,6:PLOT 1,0:DRAWTO 1,16:DRAW
TO 24,16:DRAWTO 24,6
1120 PLOT 0,39:DRAWTO 0,22:DRAWTO 25,2
2:DRAWTO 25,33:PLOT 1,39:DRAWTO 1,23:D
RAWTO 24,23:DRAWTO 24,33
1130 PLOT 79,0:DRAWTO 79,17:DRAWTO 54,
17:DRAWTO 54,6:PLOT 78,0:DRAWTO 78,16:
DRAWTO 55,16:DRAWTO 55,6
1140 PLOT 79,39:DRAWTO 79,22:DRAWTO 54
,22:DRAWTO 54,33:PLOT 78,39:DRAWTO 78,
23:DRAWTO 55,23:DRAWTO 55,33
1150 PLOT 30,6:DRAWTO 30,17:DRAWTO 49,
17:DRAWTO 49,6:PLOT 31,6:DRAWTO 31,16:
DRAWTO 48,16:DRAWTO 48,6
1160 PLOT 30,33:DRAWTO 30,22:DRAWTO 49
,22:DRAWTO 49,33:PLOT 31,33:DRAWTO 31,
23:DRAWTO 48,23:DRAWTO 48,33
1170 FOR Y=6 TO 11:PLOT 6,Y:DRAWTO 19,
Y:PLOT 36,Y:DRAWTO 43,Y:PLOT 60,Y:DRAW
TO 73,Y:NEXT Y
1180 FOR Y=28 TO 33:PLOT 6,Y:DRAWTO 19
,Y:PLOT 36,Y:DRAWTO 43,Y:PLOT 60,Y:DRA
WTO 73,Y:NEXT Y
1200 COLOR 2:FOR X=4 TO 76 STEP 3:PLOT
 X,4:PLOT X,20:PLOT X,36:NEXT X:COLOR
0:PLOT 4,20:PLOT 76,20:COLOR 2
1210 FOR X=4 TO 22 STEP 3:PLOT X,14:PL
OT X+54,14:PLOT X,26:PLOT X+54,26:NEXT
 X
1220 FOR X=34 TO 46 STEP 3:PLOT X,14:P
LOT X,26:NEXT X
1230 Y=7:GOSUB 1250:Y=11:GOSUB 1250:Y=
29:GOSUB 1250:Y=32:GOSUB 1250
1240 GOTO 1260
1250 PLOT 4,Y:PLOT 22,Y:PLOT 28,Y:PLOT
 34,Y:PLOT 46,Y:PLOT 52,Y:PLOT 58,Y:PL
OT 76,Y:RETURN
1260 PLOT 28,14:PLOT 28,17:PLOT 28,23:
PLOT 28,26:PLOT 52,14:PLOT 52,17:PLOT
52,23:PLOT 52,26
1265 COLOR 1:PLOT 13,14:PLOT 67,14:PLO
T 13,26:PLOT 67,26
1270 POKE 656,0:POKE 657,12:? "MUNCHKI
N ATTACK":POKE 656,1:POKE 657,5:? "SCO
RE":POKE 656,1:POKE 657,12:? SC;
1280 RETURN
```

Beginning of game graphics.

```
1300 GRAPHICS 2+16
1310 POSITION 2,5:? #6;"MUNCHKIN ATTAC
K"
1320 FOR W=1 TO 25:SETCOLOR 0,RND(0)*1
5,7:SOUND 0,W*20,12,6:FOR WW=1 TO 50:N
EXT WW:NEXT W:SOUND 0,0,0,0
1330 RETURN
```

End of game.

```
1350 POKE 656,2:POKE 657,1:? "PRESS FI
RE TO PLAY AGAIN...."
1360 IF STRIG(0)=1 THEN GOTO 1360
1370 Y(0)=Y0:FOR N=0 TO 4:R=USR(1536,P
B+512+N*128*(N<4)+384*(N=4)+Y(N),316):
NEXT N:GOSUB 1070:GOTO 30
```

ATARI® SWAT TABLE FOR: MUNCHKIN ATTACK

LINES	SWAT CODE	LENGTH
1 - 50	EG	662
60 - 120	UP	568
140 - 205	CM	507
210 - 270	QU	537
280 - 340	HJ	547
350 - 640	AT	541
650 - 700	QU	622
730 - 830	RN	588
850 - 905	BR	593
910 - 950	PE	532
955 - 975	EH	648
980 - 995	HM	580
1000 - 1070	RJ	548
1080 - 1120	XD	601
1130 - 1160	ZS	588
1170 - 1230	FP	584
1240 - 1300	UT	520
1310 - 1370	JO	462

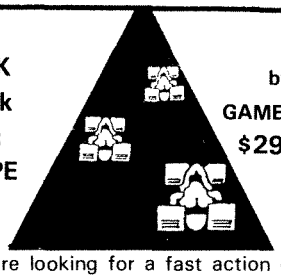
Royal Software

WE DEAL EXCLUSIVELY IN PRODUCTS
FOR THE ATARI (THE BEST) COMPUTER

YOUR MARKETPLACE FOR:

* ATARI

BAJA BUGGY



16K Disk OR TAPE

by GAMESTAR $29.60

If you're looking for a fast action driving game then we recommend this one. Good graphics and sound.

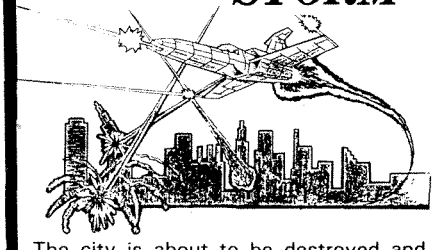
HARDWARE

Prices listed are the CASH discount price, charge prices are slightly higher.

800 Computer 16K	$600
800 Computer 48K	$678
400 Computer (used)	$199
810 Disk Drive	$448
850 Interface	$178
410 Recorder	$ 78
16K Ram module	$ 69
32K Ram module	$ 89
Percom Disk Drive	$648
48K Ram module for 400	$168
WICO JOYSTICK	$ 26
Trac Ball controller	$ 63
800/400 Dust cover	$ 9
10 Blank Disks	$ 28
Mark II Modern	$ 99
Alien Group Synthesizer	$153
Epson MX-80FT/Plus	$588
Amdek Color 1 monitor	$387
12' Joystick Extension Cord	$ 10
Computer Paper 1000 8½ x 11	$ 17

Trade in your 400 for a NEW 800 Computer. Call for details.

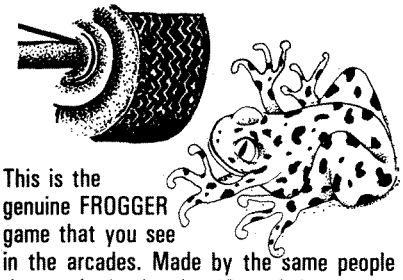
METEOR STORM



The city is about to be destroyed and ONLY you can save its people from destruction. FAST, arcade type action game.

24K Disk
16K Tape $29.95

FROGGER



This is the genuine FROGGER game that you see in the arcades. Made by the same people that made Jawbreaker (One of the top ten sellers.)

by ON-LINE

32K Disk 16K Tape $31.40

SOFTWARE

ZAXXON

Now you can get that powerful 3-D game for the Atari. Made by one of the BEST game makers, Datasoft. Order early to get yours. D $35.10

WIZARD & THE PRINCESS

A terrific HI-RES graphics adventure that is one of the best for the Atari 40K Disk $29.90

EASTERN FRONT

The MOST popular of all the APX programs and truly one of the best war games for the Atari. D,T $27.90

SAMMY THE SEA SERPENT

A must for the little computerists ages 4-7. The kiddies LOVE this one and "Sammy" too. Tape $15.90

WAR

A colorful war simulation for either one or two players. If you enjoy wargaming, then this great game is certain to please. 32K Disk $22.50

THRESHOLD

Superb graphics makes this one of the BEST space games for the Atari. Fast paced and arcade quality make this a must. 40K Disk $35.10

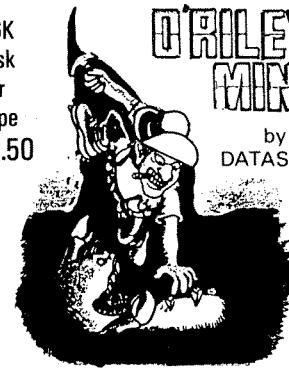
DE RE ATARI

This manual is a favorite among the serious programmers. Learn the many secrets of the Atari. $17.90

FREE
ATARI PRODUCTS CATALOG
(includes program descriptions)
with any order or
send $1.00 refundable with order
Send your name to get our
FREE monthly flyers

O'RILEY'S MINE

16K Disk or Tape $31.50



by DATASOFT

A SUPER graphic game in which you are the miner and must watch out for mine flooding and monsters of the deep.

GORF

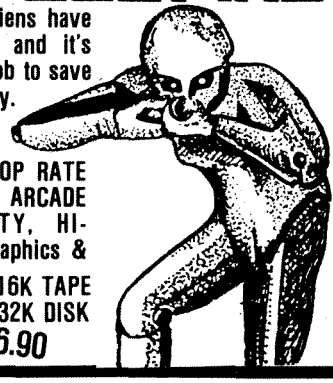
★ASTRO BATTLES
★LAZER ATTACK
★SPACE WARP
★FLAG SHIP
by
ROKLAN
MIDWAY

Just like the arcade game that has been so popular. This is one of our BEST sellers for all ages.

$35.10 24K Disk $39.90 16K Cart.

CROSSFIRE

The aliens have landed and it's your job to save the city. (If you can!) TOP RATE GAME, ARCADE QUALITY, HI-RES Graphics & Sound.



16K TAPE
32K DISK
$26.90

VISA master charge

TO ORDER CALL (503) 683-5361

HOW TO ORDER: Send check or money order or phone your order using your charge card. Ask about our 48 hour delivery service. Shipping on programs is $2.00 per order in USA or $3.90 for the 48 hour AIR service. Call for Hardware shipping costs. Prices subject to price change without notice.

Store Hours
8 am – 6 pm
Mon. – Sat.

Royal Software
2160 West 11th Ave., Eugene, Oregon 97402

ATARI® fig-FORTH

"This article is the result of an attempt to write an interactive tutorial on using this version of fig-FORTH. Read the text and try the examples on your Atari as we go along. Above all, have fun."

by H. E. Striepe
Documentation by H. E. Striepe and Dave Flory

***Atari fig-FORTH* requires an Atari 400/800 with 32K RAM and disk drive. It is included as the bonus program on issue 36 Atari DV. See the Bind-in Card elsewhere in this issue to order this month's disk.**

Welcome to the world of *Atari fig-Forth*. We are trying to make FORTH easy for you to learn and use. Since we are learning too, we would appreciate any feedback you may have on this version of *fig-FORTH*, and the included documentation. Please send any comments to Team Atari, 4029 Payne Avenue, San Jose, CA 95117, or leave E-mail on Compuserve for [70525,434]. We'll try to answer all correspondence.

To start, boot the FORTH disk. When you get the prompt "oK", type 30 LOAD and press RETURN. Be prepared to wait. Various messages will be displayed — ignore them for now. After you get the "oK" prompt again, you can get a list of all the currently defined FORTH verbs by typing VLIST. CTRL 1 will freeze the display just as in BASIC. You may notice that there are many BASIC commands included in FORTH. After you've seen enough, type 0 UE.

UL, LL, T, N, P, INDEX

FORTH works with blocks of information called screens. You should be looking at the upper half of screen zero right now. *Atari fig-FORTH* saves the last two screens EDITed, LOADed, or LISTed in memory. One screen is displayed, while the other is stored in a buffer. This is especially handy when comparing screens from different areas of the disk. Since each screen is too large to be shown on the display at one time, only half of the screen is shown. You can choose between the upper half and the lower half by using the UL and LL verbs. UL selects the Upper Limit of the currently displayed screen, while LL selects the Lower Limit. Try looking at the entire contents of screen 0 by using the UL and LL verbs. The verb T allows you to Toggle between the two available screens. Since there is no other screen in the buffer yet, type 1 UE to get the upper half of screen 1. Now try the T, as well as the UL and LL verbs. The P verb will fetch the screen number that is one less than the currently displayed screen. Type P and then use the T verb to get back. The N verb does the opposite. It loads the next higher numbered screen. Another handy verb to learn is xx yy INDEX. (xx and yy are the starting and ending screens, respectively.)

INDEX is the verb you use to find out what is on your FORTH disk. It shows you the first line of each screen on the disk. It's a good practice to make the first line of any screen you write a short description of what is on the screen. Then, when you use INDEX, you can get a quick list of the contents which can also be sent to the printer. To use the INDEX verb, you must specify a starting and ending screen number between $00 and $4D. To see the entire contents of a FORTH disk, type 0 4D INDEX.

LPOPEN, LPINDEX, LIST, LISTLP, SHOW, SHOWLP

To print the INDEX, you need to open the printer device for output. LPOPEN is the verb to do this. It need only be executed once. The INDEX verb works only on the screen. To get a printed INDEX, use the xx yy LPINDEX verb. This verb functions exactly like INDEX. To get an INDEX of the entire FORTH disk, type 0 4D LPINDEX. However if you wish to see an entire screen at once, there are two verbs that will do this: xx LIST and xx LISTLP (xx being the screen number). The first will LIST to the display; the second to the printer. Optionally, two screens may be shown at once. The xx yy SHOW verb displays all screens between xx and yy consecutively on the display, while xx yy SHOWLP prints them two at a time, side by side, on a page.

SYS, BACKUP, DoFORget, SETPHYS, RESPHYS, FORMAT, MAKEBOOT, SETSYS

You may have wondered how to duplicate this disk. The normal DOS copy routines will not work, as there is no directory on a FORTH disk. You can use SUPERDUP, ARCHIVE, or any other sector copying utility. Or you can use the BACKUP verb. Since FORTH does not use the Atari DOS, it has a limited set of verbs for disk operations that can be accessed through the SYS verb. After SYS has loaded, you will be given a menu of what DOS operations can be performed. The BACKUP verb allows you to make a copy on a one disk system. DoFORget allows a protected dictionary verb to be erased, whereas the normal FORGET verb will only erase verbs defined after booting. xx SETPHYS sets the starting location of screen 0 on the disk. RESPHYS changes the start location of screen 0 to the previous value. To modify the RESPHYS location, simply specify SETPHYS twice — the first time for the RESPHYS location, the second for SETPHYS. xx FORMAT will format the disk in drive xx with the standard FORTH format. MAKEBOOT generates a compiled listing of all currently defined verbs for loading in when you boot. These verbs will then be part of the runtime system, erasable only with the DoFORget verb. SETSYS will store all boot-up parameters pertaining to display colors and margins, and FORTH options such as error messages, stack display, etc. Note: *Do not* move any of the screens 1 through 9 and F!

STON, STOF, WARNON, WARNOFF, GS, WS, BS, NS, HX, DX, BX

If a display of the stack is desired, STON and STOFF will turn the stack display on and off respectively. To get long error messages, type WARNON. To turn them off and get only message numbers, type WARNOFF. There are a number of verbs currently defined for altering the display colors: GS = green screen, WS = white screen, NS = normal screen, and BS = black screen. Give them a try and see which ones you like. The number base for numeric entries can be altered to best suit the needs of the user: *HX = hexidecimal, DX = decimal,* and *BX = binary.* Note that the stack display only shows the lowest four digits of the items currently on the stack. This means that the stack will show 2345 after entering the line DX 12345, even though all digits are still there. The color of the display border will change to remind you which number base you are in.

General Information

Now that we have some basic tools with which to find our way around the disk, let's turn to some general information on how things are stored. This month's ***SoftSide*** DV contains a minimal subset of the available verbs in standard *fig-FORTH* when it is first booted. Many other verbs are stored on the disk in the form of source screens that can be loaded as desired. When you type 30 LOAD, a special screen is executed that causes all of the other verb screens to be loaded and added to the verb list. You got a list of all available verbs when you typed VLIST earlier. It is recommended that you make a copy of the disk with all the verbs loaded, and another with only the minimal subset in memory. If a smaller system is desired after making a copy, use the FORGET and DoFORget verbs. The more verbs compiled in RAM, the less room you'll have to work within your own applications. Incidentally, a faster form of the LOAD verb may be used on screens known to be error free. xx L& shuts of the screen and inhibits error messages normally generated with xx LOAD. This verb does not exist in the minimal system, but is active after typing 30 LOAD.

FORTH, EDITOR, ASSEMBLER, L#ON, L#OFF, DOIT, WIPE

There are three different modes of operation in the *Atari fig-FORTH* system: FORTH, EDITOR, and ASSEMBLER. The format you have been using so far is the EDITOR format. There are two different ways of viewing screens while in the EDITOR. There is a split screen editing window accessible with the verbs xx UE and xx LE. UE selects the upper half of screen xx while LE selects the lower. The other major format is the xx LIST or xx L. format. In this format, you have to use the CTRL 1 key to stop and start the scrolling of the screen. By typing L#ON you will get line numbers on the editing screen to facilitate use of line edit verbs. If you want to get rid of them later, just type L#OFF and they will go. This works in either format.

Screen 0D is blank so you can try some screen editing. Type 0D UE to go there and then type on the upper edit window. When done, use the cursor arrow keys to move down to the command window, (the area containing the word DOIT) and press the RETURN key. The DOIT verb takes the information in the edit window and puts it into a screen. Then if you type FH, the entire screen will be written to disk. To erase it, use the xx WIPE verb (or just W while in the editor).

REV.F has a number of bug fixes for REV.A OS. It also intercepts the DOSINI vector during BOOT, and at WARMSTART (RESET). This makes it possible to set the screen colors and margins for boot up and reset. Simply set your margins and screen colors before calling SYS, and select STACKON or STACKOFF. Then use the verbs SETSYS, and MAKEBOOT. Your new disk will now boot in your colors, and have your margins.

You can also hook in a Machine Language subroutine at this point. It must have been defined prior to calling SYS. After calling SYS, use the the verb HOOK xxxx (where xxxx is your CODEd assembler verb). If you think you've made a mistake, UNHOOK reverses the process. Note: Your routine is called as a machine subroutine, and must end with an assembler RTS.

The screen editor now has a keyboard intercept, so you can't accidentally hit RETURN while editing the upper window. To override this while editing, hit CTRL (or SHIFT) RETURN. Note that many screens have been changed from the previous revision, so full compilation requires staying with one group of source screens. The enhanced SYS verbs should not be used with kernels of previous revisions.

Player/Missile Demo

Now, let's take a look at the player/missile demo. To do this, type 3A L& (or 3A LOAD). It's also a good idea to print out this screen for reference, as we will be using it in the comments that follow. To do this, type LPOPEN if you have not done so already, and type 3A LISTLP (or 3A LIST for those without printers). Here is a listing of the new verbs:

0VP: P/M vertical position.
0HP: P/M horizontal position.
0VPOLD: P/M old vertical position.
GETPS: get player shape and place in P/M memory space.
SPLAY: show player.
RUNIT: demo program code.
B/H: converts a binary number to hex and decimal.

The 2A in front of COLPM! is the color of the player in HEX. The first digit indicates the color, and the second is the luminance. The first number in front of SIZE! is the P/M size,

the second is the P/M number. The numbers for GETPS in SPLAY are:

1. The number of screen holding P/M shape data.
2. The offset into the screen where the P/M data starts.
3. The number of bytes for P/M data.
4. The horizontal position to show player.

If you are using this routine to store P/M data on disk, you must be careful to place the data correctly on the data screen. Count the bytes carefully and enter the data in ATASCII characters. This is done by using BDUMP to enter the hex numbers into the RAM buffer position corresponding to the screen. The B/H verb is very useful in converting a P/M bit pattern in binary to its hex and decimal equivalents. This verb was added when you loaded the P/M demo screen (3A). To become more familiar with its use, try converting a P/M box shape such as the one that follows. (Note: The portions in italics are computer generated responses.)

BX

11111111 B/H ***HEX=FF DEC. =255***
10000001 B/H ***HEX =81 DEC. =129***
10000001 B/H ***HEX =81 DEC. =129***
11111111 B/H ***HEX =FF DEC. =255***

To use B/H you must be in BX or binary base. Then, enter the binary shape line, type B/H and press RETURN. FORTH will type the HEX and DEC. numbers for you. To put these numbers onto the disk without having to look up the Atari control keys in the table, use the BDUMP verb. It works this way. You type the following: 0F BLOCK 3E0 + DUP BDUMP. This gives you eight character positions of P/M data on screen number 0F displayed on the monitor. You then count in the number of spaces you want the data and enter it on the screen, using the cursor and pressing RETURN on the same line. It looks similar to this:

HX 0F BLOCK 3E0 + DUP BDUMP
0CE0 99 5A 3C FF 7E 24 3C 81
oK

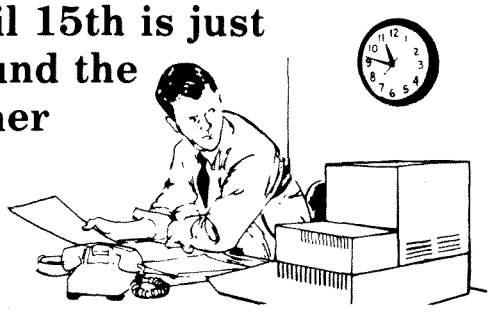
The xx BLOCK verb locates the start of screen xx in memory. The 3E0 + DUP make the starting and ending locations for the xx yy BDUMP verb. xx is the starting memory address, and yy is the ending.

In this instance, the player's current location is displayed at memory location $0520. (Note that your address may be different.) You can alter the memory locations displayed in the BDUMP. Just move the cursor over the line you wish to change. By typing over the hex values displayed with the values obtained from the B/H verbs, and typing return, you can alter the P/M data. This is a temporary change, however. To permanently save the new shape, you must edit the screen (0F UE) or mark it as updated (UPDATE), and store the new buffer on the disk (DOIT and FLUSH).

To see the shape and run the Player/Missile demo, type SPLAY and RUNIT. This will display the shape under the control of joystick 0. You will notice that the shape "hides" behind the text window. You can draw something on the playfield and experiment with the P/M priority register (GPRIOR) to create other obstacles.

The graphics verbs in this version of *Atari fig-FORTH* are very like those you use in BASIC except that, as you have noted here, the numbers come before, not after, the commands.

xx C. = COLOR xx
xx GR. = GRAPHICS xx
xx yy POS. = POSITION xx, yy
xx yy DR. = DRAWTO xx, yy
xx yy zz SE. = SETCOLOR xx, yy, zz
ETC.



The fill verb is xx XIO18. It requires the same setup as the BASIC version, with xx being the fill color register. Try this example:

DX 8 GR. 1 C. 159 0 PL.
239 80 DR. 79 80 DR.
158 1 POS. 1 XIO18

Stack Display

One of the nice features of this system is the stack display. To use it, just type STACKON or STON. Then type a few numbers and press RETURN. You will see the numbers on the stack display at the top of your monitor screen. To turn it off when unwanted, type STACKOFF or STOF. You will find the stack display very useful as you start defining your own verbs. You can go through each step in the word, one at a time, and watch what it does to the stack. The stack is the single most difficult thing for most beginning FORTH users to understand, and this display is used by even the most experienced programmers.

Put some numbers on the stack and try some of the math verbs such as +, /, -, and *. Use SWAP, ROT, ., DROP, etc., and observe their effects. Try mixing the HX (hexadecimal), DX (decimal), and BX (binary) modes. Watch carefully how each is displayed on the stack. Base conversions are a snap when you alternate between number bases. For example, type:

HX B6 BX .
10110110 oK

DECOMPILER, DCP, ZZ

Now we come to one of the most useful and powerful verbs in this version of the FORTH vocabulary — DECOMP. To use it, just type DECOMP xxxx, DCP xxxx, or ZZ xxxx; where xxxx is the word you want to decompile or take apart. This word works on even the primitive words in the *fig-FORTH* kernel. Try a few of the simple words first, like GS or NS. You will find that the numbers above 0,1, and 2 are all prefixed by the verb LIT. The first three are used so much that they are predefined as FORTH words. The verb LIT tells FORTH that the numbers are to be taken as literal values. To give you an idea how this works, try typing in the following verb definition and then decompling it.

DX
: AS 26 709 C! 18 710 C!
16 712 C! ; HX

You now have another word available to you — AS, which stands for amber screen. Try to decompile it. Study the display. (Try decompiling the verbs that make up AS, like C! or ;S.) If you like, you can make this a permanent part of your disk by using MAKEBOOT to write out all the compiled verbs you have in RAM. To remove it from memory, type FORGET AS.

Texts On FORTH

If you are new to FORTH, this should give you enough to think about for quite awhile. We haven't tried to make you a programmer, but only to introduce you to the features of this version of the language and help you over a few of the initial rough spots.

There are several good books for people new to FORTH. The one we found most helpful was *Discover FORTH*, published by McGraw Hill. Another excellent book, and probably the best choice if you can afford to buy only one, is *Starting Forth* by Leo Brodie. (See review elsewhere in this issue.) published by FORTH, Inc.

We wish you the best of luck and hope that you will keep us up-to-date on your extensions to this language. We will try to act as a clearing house for *Atari FORTH* people. As we come up with more features, we will add them to the package. We will try to upload the new additions onto Compuserve ACCESS area so you can get free updates. We suggest that you keep the original source disk unchanged, so that you can recompile your working kernel as you get more updates. If someone wishes to write a terminal program in *Atari fig-FORTH*, we will make it available immediately, so that we can all communicate FORTH material more easily, without typing in the screens. Note that *Atari fig-FORTH* is public domain. Please feel free to copy the FORTH system on your ***SoftSide DV*** and pass it around to everyone you know. The following is a listing of some of the specialized *Atari fig-FORTH* verbs. Most are active only after a full load (screen 30 LOAD).

Editor Commands and Functions

EDITOR: VOCABULARY name.
xx EDIT: Enter editor on screen xx.
L, N, P: List current, next, or previous screens.
LL: List lower half of current edit screen.
UL: List upper half of current edit screen.
DOIT: Take top 16 lines of screen and place them into the top or bottom half (LL or UL) of the edit screen.
xx yy COPY: Copy screen xx to screen yy. No change to screen xx.
xx LIST: Set screen number to xx and list it.
xx yy SHOW: List screens xx to yy inclusive.
xx yy INDEX: List first line of screens xx through yy.
FLUSH: Return to FORTH and write out all updated screens.
UPDATE: Mark screen as updated.

Fast Edit Commands

EDT: Same as EDITOR.
FORTH: Exits EDITOR without action.
xx UE: Same as xx EDIT UL.
xx LE: Same as xx EDIT LL.
N: Edit next upper screen.
N.: Edit next lower screen.
P: Edit previous upper screen.
P.: Edit previous lower screen.
T: Edit other upper screen in buffer.
T.: Edit other lower screen in buffer.
FH: Same as FLUSH.
xx WIPE: Clear screen xx.
W: Clears screen being edited. Requires a response of RETURN to work, or N to cancel.
xx LOAD: FLUSH screen being edited, and LOAD xx.
L#ON: Turn on line number display.
L#OFF: Turns off line number display.
SOUNDOFF: Eliminate audible cue (the beep heard for various editing commands).
SOUNDON: Restore audible cue.

Editor Line Editing Commands

xx TL: Type line xx into PAD and display it.
xx HL: Type line xx into PAD but don't display it.
xx DL: Delete line xx and put into PAD.
xx IL: Insert line in PAD after line xx.
xx RL: Replace line xx with line in PAD.
xx SL: Spread lines at xx (insert blank line).
xx BL: Blank out line xx (erase line).
xx $: Text following $ will replace line xx and go into PAD.
xx %: Text following % will be inserted after line xx and go into PAD.
xx yy CL: Copies line xx of screen yy into PAD.

fig-FORTH 1.4S Commands

xx yy zz COPIES: Move screens xx through yy to the screen zz on up. zz=xx, zz+1, ..., zz+nn=yy.
xx yy DUPLICATE: Duplicate screens xx through yy onto another disk in a single drive system.
SYS: Loads BOOTMAKER and other disk related words.
STACKON: Turn on stack display.
STACKOFF: Turn off stack display.
STON: STACKON.
STOF: STACKOFF.
WARNON: Display long error messages.
WARNOFF: Display error number codes only.
DRAIN: EMPTY-BUFFERS.
xx yy LZERO: Clears screens xx though yy.
xx SETPHYS: Set PHYSOFF to xx.
RESPHYS: Reset PHYSOFF to original value.
DCP: DECOMP.
ZZ: DECOMP.
xx yy BDUMP: Dump hex bytes in memory locations xx through yy.
xx yy CDUMP: Dump ATASCII characters in memory locations xx through yy.

Verbs by Bob Gonsalves, *Antic* Magazine

xx >< : Swap the MSB and LSB of xx.
xx MSBYTE: Use only the MSB of xx. Mask off LSB and shift down.
xx LSBYTE: Use only the LSB of xx. Mask off MSB.
xx VAR yyyy: Defines the variable yyyy with an initial value of xx. Can be used in place of the xx VARIABLE yyyy verb. Variables defined with VAR will return the value of the variable, *not* its memory location. To put a value into a VAR variable, use the TO verb.
xx TO yyyy: Assigns the value xx to the variable yyyy. This pertains to VAR declared variables, not VARIABLE.
xx MSB: shortform MSBYTE.
xx LSB: shortform LSBYTE.

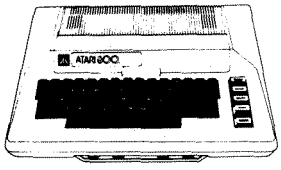
Verbs by R. Mansfield, *Compute!* and *HES*

xx yy FIND zzzz: Search for text zzzz, from screen xx to screen yy. Search all screens on disk as a default. Abort by pressing START button.
NS, GS, WS, BS: Change screen colors.
xx U.: Prints the unsigned value of byte xx. Normally integers range from -32767 to 32767. U. displays integers as 0 to 65535.
VERIFY, NOVERIFY: Change disk I/O verify on write command.
xx SNDOFF: Turn off voice xx (e.g. xx 0 0 0 SOUND).
THERE: Returns the address of the top of available user memory (MEMTOP).
FREE: THERE HERE -
BINARY: Change BASE to 2.
HX, DX, BX: Shortforms of HEX, DECIMAL, and BINARY.
PON: All screen I/O is echoed to the printer.
POFF: Turns off printer echo.

And many, many more ...

A.S.C., INC
Authorized Service Center

"Call us for discount prices, availability, or advice. Remember, service is our middle name." Joe Guzzo, President, A.S.C. Inc.



ATARI 800

800 16K $654.95
800 32K $745.95
800 48K $815.95

ATARI 400

400–16K $264.95
400-32K $364.95
400-48K $430.95

410 Recorder $76.00
810 Disk Drive $444.95

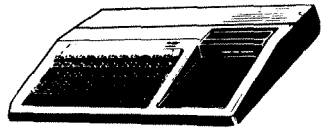
Kits

481 Entertainer $83.00
482 Educator $129.95
483 Programmer $56.95
484 Communicator . . . $339.95

Software

We carry the full line of Atari hardware and software products backed by our Authorized Service Department.

Please call for special prices on all Atari software and APX software.



TEXAS INSTRUMENTS

All models . . . . . Call for prices

MONITORS

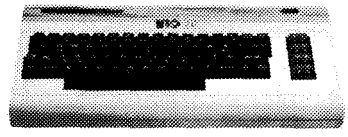
Amdex 12-in. B&W . . . $129.00
Amdex 12-in. Green . . $139.00
Amdex 13-in. Color . . . $349.00
BMC--BM 12A $114.95
NEC 12-in. B&W $169.95
NEC 12-in. Color $339.00
TI 10 . . . . . $344.95
Corvus . . . . . Call for prices
Zenith Z19 12-in. $115.00

DISKS

Elephant . . . . . 10/$21.95
Elephant DD . . . . . 10/$24.95
Dysan 5¼-10/16 . . . . . 10/$38
Verbatim 8-in.SD--SS . . . 10/$32
Verbatim . . . . . 100/$245
Scotch 3M . . . . . 10/$25.95

DISK DRIVES

Corvus . . . . . Call
Percom . . . . . Call
Paradynamics . . . . . Call



VIC-20

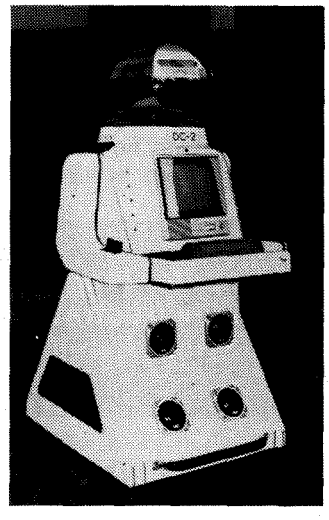
VIC-20 Personal Computer $175

GAMES/ PROGRAMS

Unlimited Availability . . . . . Call

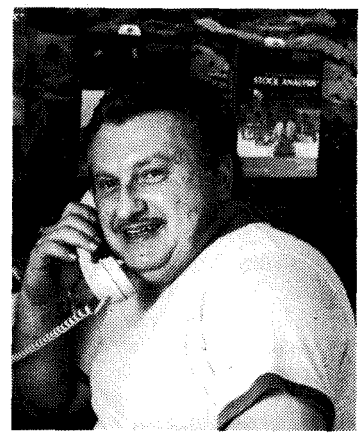
COMPUTER DESKS

Send for catalog and photos.



DC-2 ROBOT

Operates on any home computer.
Stands 4 feet high.
Call for details and prices.



TERMINALS

Televideo $569.00
910 . . . . . $569.00
912C $694.95
920C $744.95
950 . . . . . $934.95

Call for Computers

MODEMS

Lexicon $149.00
U.D.S. 1030ALP $169.00
Racal Vadic VA 3212 . . $749.00
Prentice Star $179.95

PRINTERS

Brother HR-1 Parallel . . $829.95
Brother HR-1 Serial . . . $880.00
Smith-Corona TP-1 Parallel $665
Smith-Corona TP-1 Serial . $665
C. Itoh Starwriter $1,350
Diablo 630 . . . . . $1,999.95
Epson printers . . . . . Call
Centronics printers . . . . . Call
All name brands . . . . . Call

NEW•NEW

Coleco-vision, Mattel Intellivision, and Atari VCS game machines and game cartridges. Call us for information on the latest, improved models and best prices.

Limited space prevents a full listing of the lines we carry. If you don't see it listed, we can get it for you, for less.

TO ORDER: Phone orders accepted via Mastercard, Visa, or bank wire transfer. Visa and MC service charge of 2%. Mail orders may send charge card number (include expiration date), cashiers check, money order, or personal check (allow 10 business days for personal or company checks to clear). Please add 3% ($5.00 minimum) for UPS shipping, handling, and insurance. APO and FPO include 5% ($7.00 minimum) for postage. California residents add 6% sales tax. Please include phone number on all orders. FOREIGN ORDERS include 1% handling. Credit cards not accepted on foreign orders. All equipment is in factory cartons with manufacturer warranty. Opened products not returnable. Restocking fee for returned merchandise. Equipment subject to price change and availability. Retail prices differ from mail order prices. WE SHIP THE SAME DAY ON MOST ORDERS.

Call Collect (714) 284-5615 7 Days/Week

Monday through Saturday 8am–8pm
Sundays and Holidays 12n–6pm

After 11/6/82 , Call (619) 284-5615

A.S.C. Inc., 7436 University Ave.
La Mesa, CA 92041

valFORTH

From Valpar Corporation, 3801 East 34 St., Suite 105, Tucson, AZ 85713. System requirements: Atari® 400/800 with 24K and disk drive. Use of one or more of the various extension packages available requires a 32K system. The suggested retail price for the basic package (*valFORTH* kernel and fig Editor) is $45.00.

Trying to describe any computer language is difficult, but with FORTH it is especially frustrating, because it defies attempts to place it within familiar categories. Because so few people are acquainted with FORTH, however, it seems appropriate to make a few general observations on the nature of the language before reviewing this specific implementation. It is difficult to say if FORTH is an interpretive or compiled language, as it has elements of both. Like a compiled language, source code is entered through a terminal or from a disk file, and is then converted into a form which can be used internally without having to be re-interpreted. But like interpreted languages, it is interactive. You can sit down and create a new module, test it in a controlled setting, change it and test it again, without having to change all of your source code.

"Trying to describe any computer language is difficult, but with FORTH it is especially frustrating, because it defies attempts to place it within familiar categories."

If this sounds somewhat mysterious, it is because of the unique way in which the language operates. Each FORTH command is called a "word." The typical implementation of FORTH contains a more or less standard group of these words, which is referred to as the "kernel," or core FORTH. These commands include simple looping structures, conditional structures, mathematical and logical operations, memory and stack operations, and input/output functions. The way you build an application out of these commands is to use groups of simpler commands to define new, more complex commands. These new commands then become part of the language, and can be used in exactly the same way that you use the core commands. By building more and more sophisticated words, you finally get one word that actually is your complete application.

An example related to BASIC programming may clarify this somewhat. In a BASIC program, if you wish the program to pause and do nothing for a while, you might put in an empty loop to delay the program. The statements FOR I=1 TO 10000:NEXT I would cause such a delay, and allow the user time to read some text, for example. Every time you wished the delay to occur, you would have to insert this same statement into the program. In FORTH, the equivalent function could be assigned to a newly-defined word called DELAY. The code to accomplish this might look like

```
: DELAY 10000 1 DO LOOP ;
```

This could be read as "Start definition of the word DELAY. Set an initial index to 1, and loop, increasing this index by 1 until it reaches the maximum value of 10000. End of definition." The word DELAY is defined in terms of the core words DO and LOOP. Once defined, it becomes a valid FORTH command. This means that you could use DELAY in defining other FORTH words, and when these words were executed, they would include the built-in delay. If you wanted to test the BASIC statements to see how long a delay they caused, you could execute them by typing both statements in and hitting RETURN. Likewise, if you wanted to test the new word DELAY, you could execute it in the immediate mode by typing in DELAY on the keyboard, and hitting RETURN. One interesting difference you would discover if you tried testing these two programs is that the BASIC delay loop takes about 23 seconds to execute, while the FORTH loop is finished in a little over a second!

One of the main advantages of FORTH is speed of execution. Depending on the situation, it can range from five times to 100 times faster than BASIC. Where extra speed is needed, FORTH words can also be defined directly in Machine Language using a FORTH assembler. This type of speed is especially useful for graphics animation, and fast-action games, which are almost impossible to program effectively in BASIC. There are other advantages as well. A well-coded FORTH program is very compact, and requires little memory. The modular structure of FORTH programs encourages good programming habits. It allows the programmer to isolate each small element of the overall problem, and deal with those elements one at a time. As mentioned above, the interactive nature of the language allows easy testing of each module. An added bonus is the portability of these

Reviewed by Sheldon Leemon

modules. Once you have defined a word which performs a specific task, you can use that same word in future programs. If you improve the word later, earlier programs can be improved just by substituting the improved definition of the word. Creating a specialized vocabulary for commonly performed tasks can really speed program development time.

Of course, along with these advantages go some drawbacks. Although FORTH programs execute faster, it may take you longer to code them. There are many reasons for this. First of all, FORTH is a stack-oriented language. This means that the arguments used by FORTH operators (like the variables and constants you manipulate in BASIC) are usually kept on a pop-up stack while you are working with them. The last number that is pushed down onto the stack is the first one that gets pushed off. Therefore, it is important to keep track of just what is on the stack at all times. BASIC programmers aren't used to attending to this kind of detail. Because of this stack orientation, FORTH uses Reverse Polish Notation (RPN). In FORTH, if you want to add 3 and 5, the correct syntax would be 3 5 +, with the addition operator coming after the arguments. This notation is generally conceded to be clearer than the more commonly used infix notation, where the operator comes between the arguments. For example, using infix notation, you need parentheses to tell whether 3+5/4 means (3+5)/4 or 3+(5/4). The RPN statement 3 5 4 / + does not suffer from this ambiguity. Unfortunately, beginners may find this unambiguous statement difficult to comprehend. The problems that some beginners may have with postfix notation and the heavy stack orientation could be magnified by the modular nature of the language; if you build a high-level word from lower-level words, you must keep track of what arguments each component word requires, and what happens to the stack as a result of each word. In other words, while FORTH is the kind of language you can make up as you go along, you have got to remember your own rules! This may not sound too difficult, but it requires planning ahead. While you can achieve results with sloppy BASIC programming, in FORTH it will only get you into trouble.

"...every new command becomes part of the language, the sheer number of them might be overwhelming. Atari BASIC has only about 80 keywords, but the FORTH kernel could have several times that many."

Another problem that a beginner might have with FORTH stems from its extensible nature. Because every new command becomes part of the language, the sheer number of them might be overwhelming. Atari BASIC has only about 80 keywords, but the FORTH kernel could have several times that many. When you get into graphics extensions, editor and assembler, disk utilities and the like, you are faced with literally hundreds of commands. This makes the language more powerful, but somewhat harder to control. To the dedicated FORTH enthusiast, however, these problems are minor in relation to the limitless horizons it presents to the programmer who is willing to work with it.

valFORTH is based on the standard model of the FORTH Interest Group (fig), with most of the core words coded in Machine Language for extra speed. It is a straightforward implementation, with very thorough documentation. The documentation deals mostly with the particulars of this implementation though; for a generalized introduction to FORTH, it refers the user to the book *Starting FORTH*, by Leo Brodie. (See review on page 19.) Mr. Brodie's work is a clear and amusing treatment of what could otherwise be a confusing subject. While not a step-by-step introduction to FORTH, the documentation is carefully laid out with an eye to clarity, and to accommodating the FORTH novice.

Besides the fig-FORTH kernel, a number of very useful extension packages are available. There are two editors with which the user can create source code "screens," as they are called. One is the fig-FORTH editor, (included in the basic package) which is command oriented. The other is a special *valFORTH* editor which is more oriented to the Atari operating system's method of editing lines of text. The novice, used to the latter, will find the valFORTH editor much easier to use. The assembler included with the package allows the user to add Machine Language subroutines, or define FORTH words in terms of Machine Language instructions. This assembler is a superset of two popular assemblers, the Ragsdale assembler, and the APX version, and is compatible with both. Extensions are also included which support the full range of Atari graphics and sound, as well as I/O opera-

continued on page 67

A feast of computing ideas.

If you work with a 6502/6809-based system, you're probably hungry for the facts and ideas that will help you understand the inner workings of your computer. You want to go beyond canned software—use your computer for more than games—learn the advanced programming techniques that enable you to get the most out of your 6502/6809 system.

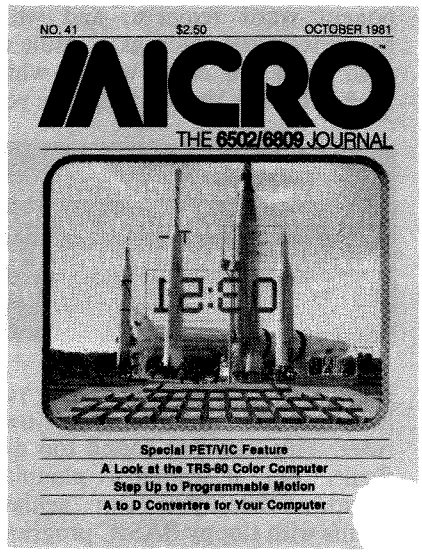
MICRO, The 6502/6809 Journal, gives you page after page, month after month, of solid information to sink your teeth into. **MICRO** is the premier how-to magazine for serious users of the Apple, PET/CBM, OSI, Atari, AIM, SYM, KIM, and all 6809 based systems including the TRS-80 Color Computer. It's a resource journal internationally respected by professionals in business, industry, and education. Every issue of **MICRO** keeps you informed with up-to-the-minute data on new products and publications:

- **hardware catalog** with organized, concise description
- **software catalog** in an easy-to-use format
- **new publications** listed and annotated
- **reviews and evaluations** of significant products

And there's much more:

- **In-depth hardware tutorials** bring expert advice into your home or office.
- **Detailed discussions of programming languages** deepen and broaden your programming ability.
- **Complete program listings** enable you to increase your machine's capabilities.
- **Bibliography of 6502/6809 information** helps you to find pertinent articles in a timely manner.
- **Special monthly features** with in-depth treatment of one subject or system increase your knowledge of the field.
- **Balanced mix of machine-specific and general articles** for your everyday use as well as long-range reference needs.
- **Informative advertising** focused specifically on 6502/6809 machines keeps you abreast of latest developments.
- **Reader feedback** puts you in touch with other micro-computerists.

MICRO is the magazine you need to get the most from your own 6502/6809 system!



You'll love every byte.

To order, send your check or international money order (payable to MICRO) and the order form at left, to:

Subscription Fulfillment
MICRO, Dept. MI
34 Chelmsford Street
P.O. Box 6502
Chelmsford, MA 01824

Or, for your convenience, call our toll-free number:

1-800-345-8112
(In California, 800-772-3545, Ext. 564)

and charge your subscription to your MasterCard or VISA. (All orders must be prepaid in U.S. dollars or charged to your MasterCard or VISA.)

YES! I want to get more from my microcomputer. Please send me ___ year(s) of MICRO at $_____/year. (Outside U.S. and Canada, please indicate via ☐ surface or ☐ air mail.)

Name _____

Company _____

Street _____

City _____ State _____ Zip Code _____

☐ Check enclosed for $ _____
☐ Charge my credit card account
☐ VISA ☐ MasterCard

Signature _____

Card number _____

Expiration date _____

SUBSCRIPTION RATES (U.S. dollars)
Yearly subscription (ISSN 027-9002) saves 20% off the single-issue price.

U.S. $24*

Canada $27

Europe $27 ($42 by air mail)

Mexico, Central America, Mideast, North and Central Africa $27 ($48 air)

South America, Far East, South Africa, Australasia $27 ($72 air)

* SPECIAL OFFER—U.S. ONLY:
Save even more—30% off single-issue price: 2 years, $42

Dept. S S

continued from page 65

tions. These are very similar to the Atari BASIC commands. Some disk utilities are provided for copying source code, formatting disks, and backing up programs. (The FORTH language is very disk-bound, and is available only on disk.) Some program debugging aids are included. This package even has floating point words implemented. (FORTH is usually restricted to two-byte integer math.) A very nice feature of this package is the inclusion of words for creating an auto-booting application package out of FORTH programs you have written. Valpar International gives permission to users to distribute programs created with this system, as long as credit is given. Programs so created do not require that the user have the valFORTH system, and can be run on any system. It is even possible to create bootable cassettes!

If this sounds like a lot for the money, it is. This comprehensive FORTH system presents enough material for learning and programming to occupy anyone for quite a while. Once you are ready to go further with FORTH, Valpar International offers a great number of extension packages, covering a very broad spectrum of programmers' needs. These include an advanced editor, with several utilities (including one for handling string data, a rarity for FORTH systems), a Player-Missile graphics package, a "turtle" graphics extension, a display formatter, and a text-compression utility. All of these packages come with extremely thorough documentation, including a cardboard quick-reference guide, a tutorial on the use of the extensions, sample programs, and complete, heavily commented source-code listings. Future packages will include a DOS interface that will let you use disk files in Atari DOS format, (which FORTH normally does not use) and a target compiler. This latter package, which will be in the $300 price range, will allow you to produce very compact object code for commercial-quality software.

This is not to suggest that it is necessary to buy all of these packages. Though the Player-Missile graphics and Editor packages are ones which I could recommend to everybody, some of the others are quite specialized, and would be useful for only certain applications. Moreover, FORTH is the type of language which most users like to "customize," suiting the means of performing specific tasks to their own personal preferences, which may or may not coincide with the way in which *valFORTH* handles them. The important point here is that Valpar International has shown a great amount of dedication to supporting a wide variety of applications, and to producing a product line which expands and improves in response to the needs of its customers. Although FORTH is not the easiest language to learn, they have worked very hard at making it as accessible as possible. They have succeeded to the degree that anybody who feels confident programming in BASIC and wants to develop higher-performance type programs should certainly investigate this alternative.

ATARI® Pascal Language System

Reviewed by Jeannine Giffee

from APX (Atari Program Exchange), 155 Moffett Park Drive, B-1, P.O. Box 427, Sunnyvale, CA 94086. System requirements: Atari 400/800 with 48K and two disk drives. Suggested retail price: $49.95.

Pascal is a high level language developed by Professor Niklaus Wirth and Kathleen Jensen in the early 1970's with a very specific goal in mind. They believed that, with the explosion of computers into society, people needed a powerful programming language designed with the rank amateur in mind — an introductory language. From this intention developed the very powerful and highly structured language, Pascal, — powerful, because with a minimum vocabulary of 35 reserved words, you can develop highly sophisticated levels of organization; structured because the language requires a particular format and a top down design flow. For example, your program must declare and define a particular Function or Procedure prior to implementing it. Pascal forces the programmer to operate in a systematic and organized fashion. The benefits of such a language are that it makes programming relatively easy to code and debug, and it is most certainly more readable than a number of other languages. The difficulties in learning Pascal are in knowing the protocols for developing a program structure that the compiler will understand and *knowing* Pascal's syntax rules (Pascal will not tolerate any syntax errors).

Atari Pascal

To evaluate this system, it is best to treat it in two ways: 1) as an implementation of Pascal, and 2) comparing it to other programming languages available on the Atari. With this approach, you can realize the benefits of the system, and also see its disadvantages, in proper perspective. Although I find a number of annoyances in using *Atari Pascal*, they are primarily because of the nature of the language. In fact, this is a very good implementation of Pascal. If you've already made the hardware investment and can meet the system requirements, $49.95 is an excellent price for a programming language.

The *Atari Pascal Language System* is based on Jensen and Wirth's definition of the language and adheres to ISO standards (International Standards Organization). Although it is not UCSD based, it is a superset of the Pascal described by Jensen and Wirth. An important thing to realize is that Pascal is very standardized. The *Atari Pascal Language System* simply takes best advantage of the features and structure of your Atari computer. Many features are offered that support "micro" technology, i.e. being able to do bit and byte manipulations, and *Atari Pascal* supports the highly desirable Atari sound and graphics capabilities.

System Requirements

The APX package includes a reference manual and two diskettes which contain the Pascal Monitor, Compiler, Linker, Interpreter, and run-time subroutine Library. This system should not be considered a tutorial, since it assumes some familiarity with Pascal. The reference manual defines the language features specific to *Atari Pascal* and helps you understand how to implement them. It also provides instructions on how to use the compiler and linker options and provides some insight into their operation. If you are not familiar with Pascal, you will definitely need a programming manual. I can highly recommend George Cherry's *Pascal Programming Structures* and the *Pascal User Manual and Report* by Jensen and Wirth. Cherry's book assumes that you are starting from scratch, but advances quickly. It reveals, through great explanations and specific examples, the complete Pascal language. Jensen and Wirth's book, while a bit more cut and dried, provides an understanding of the implementation of the language as well as offering a functional description.

Operation

The three programs that come in this package are used to develop a Pascal program. An editor is also required, but is not part of the package. There are basically four stages in program development. First, you create a source file. Second, you compile your source file. Pascal source code is not compiled to machine code, but rather to an intermediate language called P-Code (a familiar term for FORTH programmers). P-Code is relocatable code, meaning that once you have compiled your source module, it is ready to be linked with other necessary modules, like subroutines in the Pascal library. Third, you link together compiled modules and store your executable file on disk. Hopefully, in the fourth and final stage, you run your program. Since your object file resides as P-Code, it requires the Pascal Interpreter at run time in order to be executed.

Each program is called from the Pascal Monitor, a menu which will load the desired program. The system also supports at least two disk drives. Drive one is designated for system software, and drives two through four can be used for your program work space.

This particular configuration leads to a couple of restrictions. First of all, you are disk bound, which leads to slow development time. Each program must be loaded every time you wish to use it. Debugging a program, therefore, can take a long time. Each time you make a change you have to load the editor, exit, load the compiler, compile and, if successful, link and run the program. Also, control for each phase must pass through the Pascal Monitor. Since the Pascal Compiler occupies a major portion of RAM,

you cannot create large modules which might exceed available memory. You can easily compensate for this restriction, however, by writing small modules and linking them together. It also makes debugging source files faster. With the ability to declare External functions and procedures, you can maintain complete communication between modules. The bottom line is that RAM restrictions are easily managed, but without significant hardware investment, development time cannot be greatly enhanced. I think it is fair to state that such restrictions are not due to a poor implementation, however, but rather to the nature of the language.

Features

With these restrictions in mind, the developers of *Atari Pascal* provided some nice features to help you cope. The compiler tells you how long your compiled module is, giving you a sense of your RAM limitations. It also gives you the option of creating compiled files on the same drive as source files or on another drive. There are also a number of compiler and linker options which detect error types, thereby aiding you in debugging.

The documentation offers descriptions of a number of very powerful procedures and parameters in the categories of bit, byte, word, and string operations, direct disk access, access to the operating systems I/O control blocks (i.e. Open and Close procedures), and more.

A major advantage of using a language like Pascal is realized at run time. It is significantly faster than BASIC. My crude comparisons showed about a 300% increase in speed of execution. The end result is a fast program without the necessity of getting your hands dirty with assembler.

Summary

Atari Pascal is a powerful tool for only $49.95, but it requires a full blown ATARI computer system. Development time is slow, yet the instruction set is extremely powerful — a superset of the ISO's. As an interpreted language, programs are not universally portable, yet execution is speedy. There are many more pluses and minuses to this package. As a firm believer in the old adage, seeing is believing, however, I have provided two listings of a player missile program entitled "JAWS," one in BASIC and one in Pascal. See for yourself!

```
   SS SS SS SS SS SS SS SS SS SS SS
   SS                            SS
   SS        ATARI  BASIC        SS
   SS           'JAWS'           SS
   SS  AUTHOR:  JEANNINE GIFFEE  SS
   SS                            SS
   SS SS SS SS SS SS SS SS SS SS SS

10 GRAPHICS 21
15 REM
16 REM ***THE BACKGROUND***
17 REM
20 POKE 708,45:POKE 712,148:COLOR 1:PL
OT 1,12:DRAWTO 12,0:DRAWTO 0,0:POSITIO
N 0,12:POKE 765,1
30 XIO 18,#6,0,0,"S:"
40 PLOT 79,12:DRAWTO 79,0:DRAWTO 67,0:
POSITION 78,12:XIO 18,#6,0,0,"S:"
50 COLOR 1:PLOT 12,47:DRAWTO 0,35:POSI
TION 0,47:POKE 765,1:XIO 18,#6,0,0,"S:
"
60 PLOT 79,47:DRAWTO 79,35:POSITION 67
,47:POKE 765,1:XIO 18,#6,0,0,"S:"
70 COLOR 1:PLOT 32,18:DRAWTO 28,15:DRA
WTO 23,15:POSITION 18,18
80 POKE 765,1:XIO 18,#6,0,0,"S:"
85 REM
86 REM ***THE TREE***
87 REM
90 POKE 709,240:COLOR 2:PLOT 27,15:DRA
WTO 27,7:PLOT 28,15:DRAWTO 28,7:POKE 7
10,198
91 PLOT 28,15:DRAWTO 28,7:POKE 710,198
95 COLOR 3:PLOT 27,6:DRAWTO 23,3
100 COLOR 3:PLOT 27,6:DRAWTO 33,3:PLOT
 27,6:DRAWTO 42,3:PLOT 27,6:DRAWTO 16,
3:PLOT 27,6:DRAWTO 16,9
101 PLOT 27,6:DRAWTO 42,9
105 REM
106 REM **INITIALIZE P/M**
107 REM
510 PMBASE=54279:GRACTL=53277
520 SDMCTL=559:RAMTOP=106
530 HPOS0=53248:HPOS2=53250:HPOS3=5325
1
540 PCOLR0=704:PCOLR2=706:PCOLR3=707
560 REM
570 X=100:Y=60:Z=150
580 A=PEEK(RAMTOP)-24:POKE PMBASE,A
600 MYPMBASE=256*A
620 POKE GRACTL,3:POKE SDMCTL,46
630 POKE HPOS0,100:POKE HPOS2,150:POKE
 HPOS3,158
640 POKE PCOLR0,8:POKE PCOLR2,88:POKE
PCOLR3,88
650 REM
660 REM **CLEAR PM AREA**
670 REM
680 FOR I=MYPMBASE+512 TO MYPMBASE+102
4:POKE I,0:NEXT I
690 REM Create Player
700 REM
710 FOR I=MYPMBASE+512+Y TO MYPMBASE+5
21+Y:READ A:POKE I,A:NEXT I
711 DATA 224,112,56,60,62,62,63,63,127
,255
714 FOR I=MYPMBASE+768+Y TO MYPMBASE+7
77+Y:READ A:POKE I,A:NEXT I
715 DATA 0,0,64,67,102,125,123,61,30,1
5
716 FOR I=MYPMBASE+896+57 TO MYPMBASE+
908+57:READ A:POKE I,A:NEXT I
717 DATA 240,255,255,112,64,224,240,56
,248,252,188,124,248
718 REM
719 REM **MOVE SHARK**
720 REM
723 IF X=Z THEN GOTO 2000
725 FOR X=100 TO 150:POKE HPOS0,X:FOR
J=1 TO 5:NEXT J:NEXT X
730 IF X=Z THEN GOTO 2000
2000 REM
2010 REM **EAT DUCK**
2020 REM
2090 FOR H=1 TO 14:POKE MYPMBASE+777+Y
,0:POKE MYPMBASE+905+Y,0
2091 FOR I=10 TO -3 STEP -1:POKE MYPMB
ASE+768+Y+I,PEEK(MYPMBASE+767+Y+I)
2092 POKE MYPMBASE+896+Y+I,PEEK(MYPMBA
SE+895+Y+I)
2093 NEXT I
3000 FOR I=1 TO 30:SOUND 0,100,4,10:NE
XT I:SOUND 0,0,0,0
3010 NEXT H
3020 POKE PCOLR0,88
3025 FOR I=1 TO 200:NEXT I
3030 FOR I=1 TO 20:SOUND 0,45,12,10:NE
XT I:FOR J=45 TO 40 STEP -2:SOUND 0,I,
12,10:FOR K=1 TO 10
3040 NEXT K:NEXT J:SOUND 0,0,0,0
3050 POKE HPOS0,0
3060 GOTO 3060
```

continued on page 70

Atari Pascal *continued*

```
            SS SS SS SS SS SS SS SS SS SS SS
            SS                           SS
            SS       ATARI PASCAL        SS
            SS          'JAWS'           SS
            SS  AUTHOR:  JEANNINE GIFFEE  SS
            SS                           SS
            SS SS SS SS SS SS SS SS SS SS SS



PROGRAM JAWS(INPUT,OUTPUT);
TYPE
  SCRN_TYPE=(FULL_SCREEN,SPLIT_SCREEN);
  CLEAR_TYPE=(CLEAR_SCREEN,DO_NOT_CLEAR_SCREEN);
VAR
  A,I,J,K,L,N,X,Z,PMBASE:INTEGER;

(*PROCEDURES*)

EXTERNAL PROCEDURE INITGRAPHICS(MAX_MODE:INTEGER);
EXTERNAL PROCEDURE GRAPHICS(MODE:INTEGER; SCREEN:SCRN_TYPE;
                        CLEAR:CLEAR_TYPE);
EXTERNAL PROCEDURE SOUND(VOICE,PITCH,DISTORTION,VOLUME:INTEGER);
EXTERNAL PROCEDURE COLOR(COLOR_VALUE:INTEGER);
EXTERNAL PROCEDURE FILL(X,Y:INTEGER);
EXTERNAL PROCEDURE PLOT(X,Y:INTEGER);
EXTERNAL PROCEDURE POSITION(X,Y:INTEGER);
EXTERNAL PROCEDURE DRAWTO(X,Y:INTEGER);
EXTERNAL PROCEDURE POKE(ADDR,VAL:INTEGER);
EXTERNAL FUNCTION PEEK(ADDR:INTEGER):INTEGER;

PROCEDURE P1;
BEGIN
  INLINE(224/112/56/60/62/62/63/63/127/255);
END;
PROCEDURE P2;
BEGIN
  INLINE(0/0/0/0/0/64/67/102/125/123/61/30/15);
END;
PROCEDURE P3;
BEGIN
  INLINE(240/255/255/112/96/224/240/56/248/252/188/124/248);
END;

PROCEDURE MOVERIGHT;
VAR X,N:INTEGER;
BEGIN
 FOR X:=100 TO 150 DO
 BEGIN
  FOR N:=1 TO 80 DO N:=N; (*DELAY*)
  POKE(53248,X);
 END;
END;

      (************************)
      (*****MAIN PROGRAM*****)
      (************************)
BEGIN
INITGRAPHICS(5);
GRAPHICS(5,FULL_SCREEN,CLEAR_SCREEN);

    (*CREATE BACKGROUND*)

POKE(755,1); POKE(708,45); POKE(712,148); COLOR(1);
PLOT(1,12); DRAWTO(12,0); DRAWTO(0,0); POSITION(0,12);
POKE(765,1); FILL(0,12);
PLOT(79,12); DRAWTO(79,0); DRAWTO(67,0); POSITION(78,12);
FILL(78,12); PLOT(12,47); DRAWTO(0,35); POSITION(0,47);
POKE(765,1); FILL(0,47); PLOT(79,47); DRAWTO(79,35);
POSITION(67,47); POKE(765,1); FILL(67,47); PLOT(32,18);
DRAWTO(28,15); DRAWTO(23,15); POSITION(18,18); POKE(765,1);
FILL(18,18);

      (*CREATE TREE*)

POKE(709,240); COLOR(2); PLOT(27,15); DRAWTO(27,7);
PLOT(28,15); DRAWTO(28,7); POKE(710,198); PLOT(28,15);
DRAWTO(28,7); POKE(710,198); COLOR(3); PLOT(27,6);
DRAWTO(23,3); PLOT(27,6); DRAWTO(33,3); PLOT(27,6);
DRAWTO(42,3); PLOT(27,6); DRAWTO(16,3); PLOT(27,6);
DRAWTO(16,9); PLOT(27,6); DRAWTO(42,9);

     (*INITIALIZE P/M*)

POKE(559,46); POKE(53277,3);
A:=PEEK(106)-24; POKE(54279,A);
PMBASE:=256*A;
POKE(53248,100); POKE(53250,150); POKE(53251,158);
POKE(704,8); POKE(706,88); POKE(707,88);

   (*SETTING UP PLAYERS*)

FOR I:=PMBASE+512 TO PMBASE+1024 DO POKE(I,0);
FOR I:=1 TO 10 DO
  POKE(PMBASE+512+60+I,PEEK(ADDR(P1)+(I+4)));
FOR I:=1 TO 13 DO
  BEGIN
    POKE(PMBASE+768+57+I,PEEK(ADDR(P2)+(I+4)));
    POKE(PMBASE+896+57+I,PEEK(ADDR(P3)+(I+4)));
  END;
MOVERIGHT;

   (***EAT DUCK***)

FOR K:=1 TO 14 DO
 BEGIN
  POKE(PMBASE+780+57,0);
```

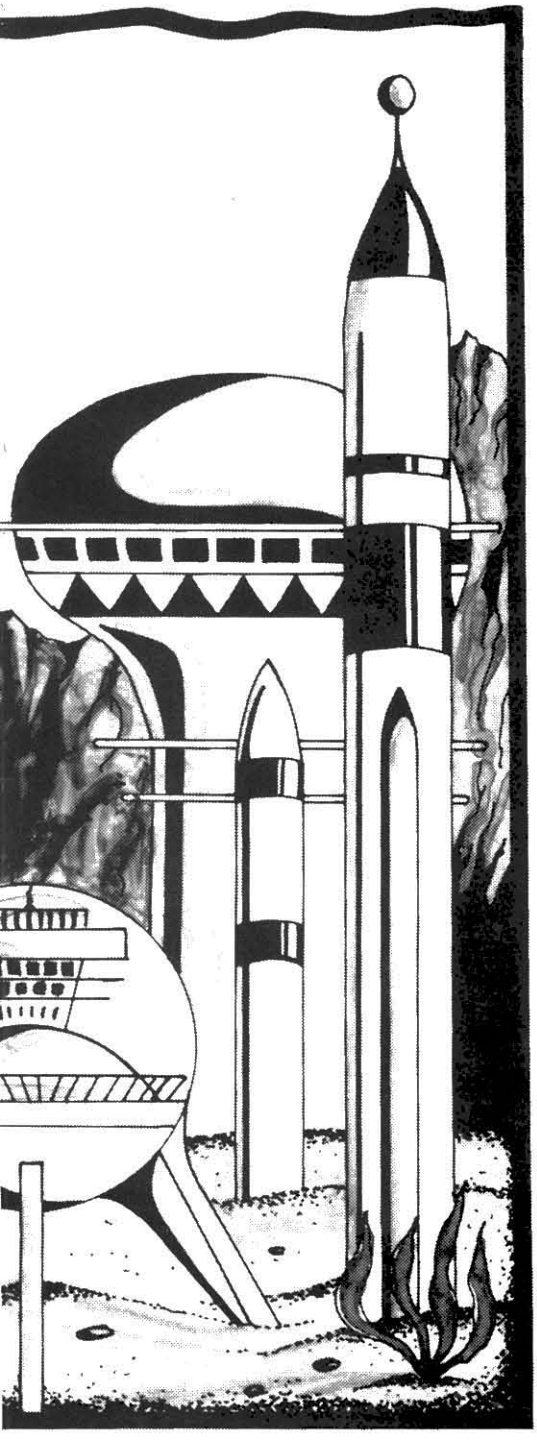
```
   POKE(PMBASE+908+57,0);
    N:=13;
    FOR J:=14-K DOWNTO 1 DO
    BEGIN
     POKE(PMBASE+768+57+N,PEEK(ADDR(P2)+(J+2)));
     POKE(PMBASE+896+57+N,PEEK(ADDR(P3)+(J+2)));
     N:=N-1; (*DECREMENT BYTE TO FILL*)
    END;
   FOR L:=1 TO 150 DO SOUND(0,100,4,10);
   SOUND(0,0,0,0);
 END;
POKE(704,88);
FOR I:=1 TO 50 DO I:=I; (*DELAY*)
FOR I:=1 TO 70 DO
  SOUND(0,45,12,10);
FOR J:=45 DOWNTO 40 DO
  BEGIN
   SOUND(0,J,12,10);
   FOR N:=1 TO 20 DO N:=N;  (*DELAY*);  END;
SOUND(0,0,0,0);
POKE(53248,0);

WHILE 4>2 DO
  BEGIN
  END;
END.
```

```
MODULE PEEKPOKE;
(* This module performs  the  BASIC like functions,  PEEK  &
POKE*)


PROCEDURE POKE(ADDR,VAL:INTEGER);


VAR
 PTR:^CHAR;
BEGIN
 PTR:=ADDR;    (*Set PTR to point at the desired address*)
 PTR^:=CHR(VAL);  (*Poke new address *)


END;


FUNCTION PEEK(ADDR:INTEGER):INTEGER;
VAR
 PTR:^CHAR;
BEGIN
 PTR:=ADDR;
 PEEK:=ORD(PTR^);
END;


MODEND.
```

ATLANT

TIS

by Michael Newman



***Atlantis* is an arcade-style game for an Apple™ with 16K RAM (32K disk), and an optional disk drive.**

Defend Atlantis! Long ago, the lost civilization of Atlantis fought a desperate battle against marauding aliens. The last hope of the ancient, yet technically advanced civilization lay in three "neutralizers," defensive weapons that nullified the energy blasts of the enemy.

In *Atlantis*, you are the gunner manning the neutralizers. Only one of the neutralizers may be deployed at a time. Paddle 1 controls the position of the neutralizer along the bottom of the screen. Immediately below the neutralizer is the city of Atlantis. Above you is the calm blue sky. Soon, deadly energy missiles will descend upon you and the millions you defend. Pressing button 1 will fire the neutralizer. Be warned, however, that there are smart missiles that will try to avoid your fire.

The city has a defensive energy shield that can absorb two energy missiles at any one point on its surface. A third strike at the same location causes the city to explode.

Each missile you neutralize gains you points. If a missile hits any part of your neutralizer, the neutralizer is demolished. If you lose all three of your neutralizers, the city explodes. (If you lose your third neutralizer while your score is between 950 and 1000 points, missiles will continue to drop onto the city's shield.) Each impact of a missile on the outer layer of the shield gains you five points, and each strike on the inner layer gains you ten, so a bit of luck will bump your score to 1000 points, earning you an extra neutralizer.

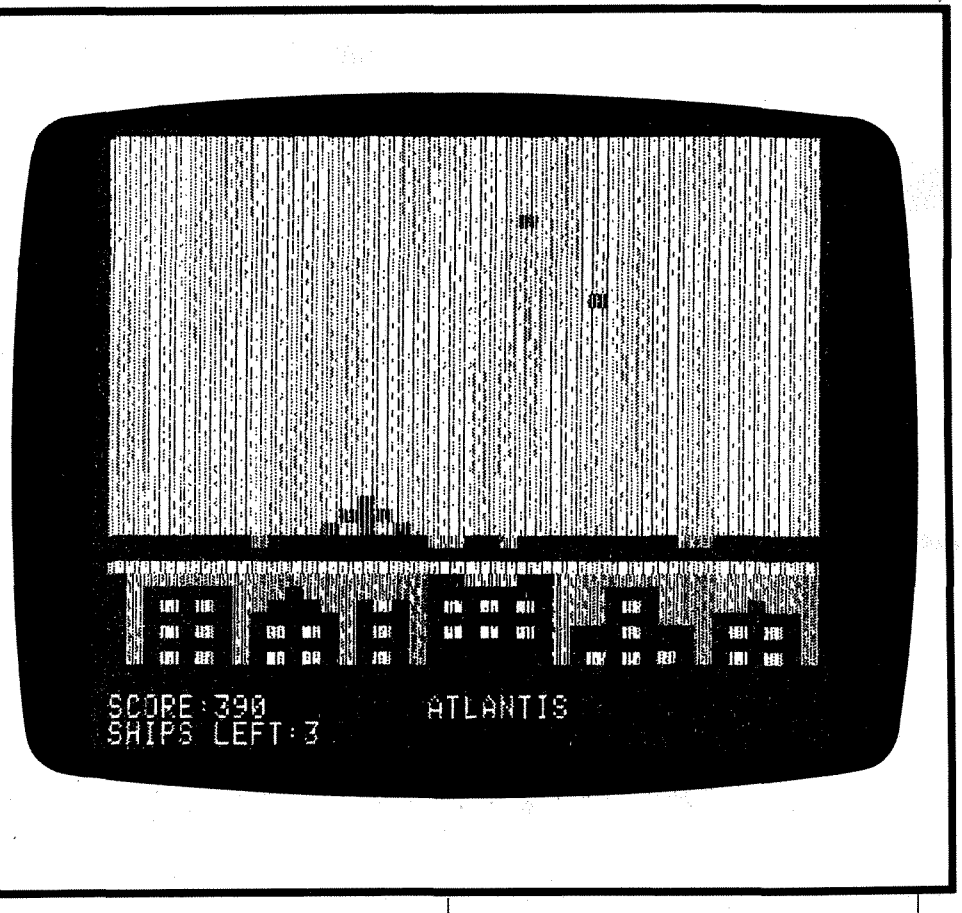
Atlantis has the ability to save the top fifteen scores on diskette. Your high score and up to 11 characters can be stored. If you do not have a disk system, do not include lines 140-220 and 2110-2400.

Variables

A: Movement of ship.
A1: Pitch of tone.
B(*): Status of each position of outer shield.
B1: Duration of tone.
D$: CHR$(4) for Apple disk operations.
D(*): Status of each position of inner shield.
H(*): Horizontal position of missiles.
L: Status of paddle button 1.
M: Number of missiles on screen.
N: Number of normal missiles on screen.
N1: Maximum number of normal missiles allowed on screen.
N2: Number of smart missiles.
P: Indicates which missile on the screen is a smart missile.
Q: Vertical movement of missiles.
R: Maximum number of smart missiles.
R1: Score needed for bonus neutralizer.
S: Score.
S(*): Speed of each missile.
S1: Number of ships left.
U(*): High scores.
U$(*): Names of players with high scores.
U1: Horizontal movement of missiles.
V(*): Vertical position of missiles.
W: Flag to indicate if you have a free ship.
X,Z: Loop variables.
X$: Used to achieve proper spacing in high score list.
Y(*): High scores on disk.
Y$(*): Names of players with high scores on disk.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS      APPLESOFT BASIC        SS
SS         'ATLANTIS'          SS
SS   AUTHOR: MICHAEL NEWMAN    SS
SS     COPYRIGHT (C) 1982      SS
SS SOFTSIDE PUBLICATIONS, INC  SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

If you do not wish to type in this program, it is available on this month's SoftSide CV and DV.



Initialize display and variables, and poke sound routine into memory.

```
10  TEXT : NOTRACE : SPEED= 255: HOME

20  POKE 768,173: POKE 769,48: POKE
    770,192: POKE 771,136: POKE
    772,208: POKE 773,4: POKE 77
    4,198: POKE 775,7
30  POKE 776,240: POKE 777,8: POKE
    778,202
40  POKE 779,208: POKE 780,246: POKE
    781,166
50  POKE 782,6: POKE 783,76: POKE
    784,0
60  POKE 785,03: POKE 786,96
70  SPEED= 255
80  DIM Y(15),Y$(15),D(39),B(39),
    K(4)
90 R1 = 1000
100  DIM V(4),H(4),S(4),U(15),U$(
    15)
110 M = 1:S1 = 3:A = 15:D$ =  CHR$
    (4):R = 10
120 U = R1:N = 30:N2 = 1
130 R = 10
```

Retrieve high scores from disk.

```
140  HOME : ONERR  GOTO 230
150  PRINT D$;"OPEN HIGH"
160  PRINT D$;"READ HIGH"
170  FOR X = 1 TO 15
180  INPUT Y(X)
190  INPUT Y$(X)
200  NEXT X
210  PRINT D$;"CLOSE HIGH"
220  ONERR  GOTO 3000
```

Print high scores.

```
230  PRINT : PRINT " ATLANTIS"
240  PRINT "BY MICHAEL NEWMAN"
250  PRINT : PRINT "     15 BEST
    SCORES SO FAR": PRINT
260  FOR X = 1 TO 15:X$ = ""
270  POKE 7,3: POKE 6,X * 6: CALL
    768
280  IF Y(X) = 0 THEN Y$(X) =  CHR$
    (95) +  CHR$ (95) +  CHR$ (9
    5)
290  IF X < 10 THEN X$ = " "
300  PRINT X$;X;"     .......
    ";Y(X); SPC( 5 - (Y(X) > 9)
     - (Y(X) > 99) - (Y(X) > 999
    ) - (Y(X) > 9999));Y$(X)
310  NEXT X
320  VTAB (23)
330  PRINT "HIT ANY KEY TO BEGIN"
    ;: GET KEY$
```

Draw the background.

```
340  HOME
350  GR : COLOR= 14
360  FOR X = 0 TO 39
370  IF X > 32 THEN  COLOR= 0
380  IF X = 32 THEN  COLOR= 13
390  HLIN 0,39 AT X
400  NEXT X
```

Draw Atlantis.

```
410  COLOR= 13
420  FOR X = 1 TO 39 STEP  + 2
430  PLOT X,35
440  PLOT X,37
450  PLOT X,39
460  NEXT X
470  COLOR= 10
480  HLIN 1,17 AT 33
490  VLIN 33,39 AT 1
500  VLIN 39,34 AT 7
510  PLOT 8,35: PLOT 9,34: PLOT 8
    ,34
520  PLOT 38,35: PLOT 12,35: PLOT
    37,34
530  VLIN 34,39 AT 13
540  HLIN 11,17 AT 34
550  VLIN 39,34 AT 17
560  HLIN 20,22 AT 33
570  HLIN 25,39 AT 33
580  VLIN 39,34 AT 25
590  VLIN 36,34 AT 26
600  VLIN 36,34 AT 27
610  VLIN 39,34 AT 33
620  HLIN 31,39 AT 34
630  VLIN 34,39 AT 39
640  HLIN 31,35 AT 35
650  HLIN 31,34 AT 36
660  PLOT 38,35
670  PLOT 37,35
680  COLOR= 0
690  PLOT 34,36
700  PLOT 31,37: PLOT 27,37: PLOT
    9,35: PLOT 11,35
710  FOR X = 19 TO 23 STEP  + 2
720  PLOT X,39
730  NEXT X
```

Draw shields.

```
740  COLOR= 0
750  HLIN 0,39 AT 31
760  HLIN 0,39 AT 30
```

Print statistics.

```
770  VTAB (22): PRINT
780  PRINT "SCORE:";S," ATLANTIS
     "
790  PRINT "SHIPS LEFT:";S1
```

Action starts here.

```
800  GOSUB 2460
810  IF N = 3 THEN 830
820 N = N - 1: GOTO 920
830 N1 = N1 + 1
840  IF N1 < H THEN 920
850 N2 = N2 + 1:N1 = 0
860  IF N2 = 5 THEN 880
870  GOTO 920
880 M = M + 1:N2 = 1:N = 30:R = R
     - 2
890 H = H + 1
900  IF R < 1 THEN R = 1
910  IF M > 4 THEN M = 1
```

Set starting position of missile.

```
920  FOR X = 1 TO M
```

Make a "smart" missile.

```
930 Z =  INT ( RND (1) * R) + 1
940 K(X) = 0
950  IF Z = 1 THEN K(X) = 1
960 S(X) =  INT ( RND (1) * 3) +
     1
970 H(X) =  INT ( RND (1) * 30) +
     5
980 V(X) = 0
990  NEXT X
1000 W = S1 + 1:I = 0
1010  IF W = 1 THEN 1040
1020  IF A = U1 THEN 1100
1030  IF A = 100 THEN A = 15
```

Award an extra neutralizer.

```
1040  IF S < R1 THEN 1100
1050 R1 = R1 + U:S1 = S1 + 1
1060 W = 0
1070  VTAB (23): PRINT "SHIPS LEF
     T:";S1: VTAB (20)
1080  FOR X = 1 TO 6: PRINT  CHR$
     (7);: NEXT X
1090 A = U1
```

Fire laser.

```
1100 L =  PEEK ( - 16286)
1110  IF W = 1 THEN L = 0
1120  IF L < 120 THEN 1260
1130  FOR X = 1 TO M
1140  COLOR= 0
1150  VLIN 26,0 AT A
1160  FOR X = 1 TO M
1170  IF V(X) = 100 THEN 1240
1180  IF L < 120 THEN 1260
1190  IF A <  > H(X) THEN 1240
1200  FOR P = 5 TO 14: COLOR= P: POKE
     7,1: POKE 6,P *  INT ( RND (
     1) * 3): CALL 768: PLOT H(X)
     ,V(X): NEXT P
1210 V(X) = 100:I = I + 1
1220 S = S + ((S(X) * 5) + (N2 *
     5) + (K(X) * 5))
1230  VTAB (22): PRINT "SCORE:";S
1240  NEXT X
1250  COLOR= 14: VLIN 26,0 AT A
1260  FOR X = 1 TO M
1270  IF V(X) = 100 THEN 1540
1280  PLOT H(X),V(X)
1290  POKE 7,2: POKE 6,V(X) * 3: CALL
     768
1300  COLOR= 14: PLOT H(X),V(X)
1310 V(X) = V(X) + S(X): IF K(X) =
     1 THEN 1390
1320 P =  INT ( RND (1) * N) + 1
1330  IF P = 1 THEN H(X) = H(X) + N2
1340  IF P = 2 THEN H(X) = H(X) -
     N2
1350  IF H(X) > 39 THEN H(X) = 0
1360  IF H(X) < 0 THEN H(X) = 39
1370  COLOR= 2: PLOT H(X),V(X): GOTO
     1490
1380  PLOT H(X),V(X): GOTO 1490
1390 Q =  INT ( RND (1) * (N2 * 2
     )) + 1
1400  IF A < H(X) THEN Q =  - Q
1410  IF V(X) = 28 AND  ABS (A -
     H(X)) < (N2 * 2) + 3 THEN H(
     X) = A + 1
1420  IF V(X) = 27 AND  ABS (A -
     H(X)) < (N2 * 2) + 3 THEN H(
     X) = A + 1
1430 H(X) = H(X) + Q
1440  IF A = H(X) THEN H(X) = H(X
     ) -  INT ( RND (1) * (4 * N2
     )) - (2 * N2)
1450  COLOR= 1
1460  IF H(X) > 39 THEN H(X) = 0
1470  IF H(X) < 0 THEN H(X) = 39
1480  PLOT H(X),V(X)
1490  IF V(X) > 38 THEN 1970
1500  IF V(X) < 27 OR V(X) > 28 THEN
     1520
1510  IF  ABS (H(X) - A) < 3 THEN
     1700
1520  IF V(X) = 30 THEN 1850
1530  IF V(X) = 31 THEN 1880
1540  NEXT X
1550  IF I = M THEN 810
1560  IF W = 1 THEN 1040
```

Erase position of neutralizer.

```
1570  COLOR= 14
1580  VLIN 29,27 AT A
1590  HLIN A - 1,A + 1 AT 28
1600  HLIN A - 2,A + 2 AT 29
```

Plot new position of neutralizer.

```
1610 A =  INT ( PDL (1) / 5)
1620  IF A > 36 THEN A = 36
1630  IF A < 3 THEN A = 3
1640 U1 = A
1650  COLOR= 4
1660  VLIN 29,27 AT A
1670  HLIN A - 1,A + 1 AT 28
1680  PLOT A - 2,29: PLOT A + 2,2
     9
1690  GOTO 1040
```

Neutralizer explodes.

```
1700 I = I + 1:S1 = S1 - 1
1710 V(X) = 100:W = 1
1720  FOR Z = 1 TO 14
1730  COLOR= Z
1740  POKE 7,1: POKE 6,Z * 3: CALL
     768
1750  VLIN 29,26 AT A
1760  HLIN A - 2,A + 2 AT 29
1770  HLIN A - 3,A + 3 AT 28
1780  HLIN A - 2,A + 2 AT 27
1790  NEXT Z
1800  VTAB (23): PRINT "SHIPS LEF
     T:";S1
1810 U1 = A
1820 A = 100
1830  IF S1 < 1 AND (S < 950 OR S
      > = 995) THEN 1970
1840  GOTO 1540
```

Missile hits shield at screen line 30.

```
1850  IF B(H(X)) = 1 THEN S(X) =
     1: GOTO 1540
1860 B(H(X)) = 1:S = S + 5
1870  GOTO 1900
```

Missile hits shield at screen line 31.

```
1880  IF D(H(X)) = 1 THEN 1540
1890 D(H(X)) = 1:S = S + 10
1900  FOR Z = 15 TO 10 STEP  - 1
1910  COLOR= Z: POKE 7,1: POKE 6,
     20 + Z * 2: CALL 768
1920  PLOT H(X),V(X)
1930  NEXT Z
1940 V(X) = 100:I = I + 1
1950  VTAB (22): PRINT "SCORE:";S
1960  GOTO 1540
```

Atlantis explodes.

```
1970  FOR Z = 0 TO 10 STEP  + 2
1980  COLOR= Z
1990  FOR X = 33 TO 39
2000  HLIN 0,39 AT X
2010 SOUND =  PEEK ( - 16336) +  PEEK
     ( - 16336) +  PEEK ( - 16336
     ) +  PEEK ( - 16336)
2020  NEXT X
2030 Z = Z - 1: IF Z > 4 THEN 198
     0
2040  COLOR= 1
2050  FOR Z = 39 TO 0 STEP  - 1: HLIN
     0,39 AT Z: POKE 7,1: POKE 6,
     Z * 2: CALL 768: NEXT Z
2060  COLOR= 10
2070  FOR Z = 0 TO 39: HLIN 0,39 AT
     Z: POKE 7,1: POKE 6,Z * 2: CALL
     768: NEXT Z
2080  SPEED= 5
2090  VTAB (23): PRINT "GAME OVER
          "
2100  SPEED= 255
```

Print final results.

```
2110 O = 0: TEXT : HOME
2120  FOR X = 1 TO 15
2130  IF S < Y(X) THEN 2230
2140  IF O = 1 THEN 2260
2150 O = 1:U(X) = S
2160  PRINT : PRINT : PRINT
2170  PRINT "C O N G R A T U L A
     T I O N S"
2180  PRINT "YOU SCORE IS ONE OF
     THE BEST SO FAR."
2190  PRINT "PLEASE TYPE YOUR NAM
     E!": INPUT U$(X): IF U$(X) =
     "" THEN U$(X) =  CHR$ (95) +
      CHR$ (95) +  CHR$ (95)
2200  IF  LEN (U$(X)) > 11 THEN  PRINT
     "NO MORE THAN 11 LETTERS PLE
     ASE.": GOTO 2190
2210  PRINT "YOU NOW ARE ONE OF T
     HE 15 BEST PLAYERS!"
2220  GOTO 2240
2230 U(X) = Y(X):U$(X) = Y$(X)
2240  NEXT X
2250  GOTO 2280
2260 U(X) = Y(X - 1):U$(X) = Y$(X
      - 1)
2270  NEXT X
2280  IF O = 0 THEN 2370
2290  PRINT D$;"OPEN HIGH"
2300  PRINT D$;"WRITE HIGH"
2310  FOR X = 1 TO 15
2320  PRINT U(X)
2330  PRINT U$(X)
2340  NEXT X
2350  PRINT D$;"CLOSE HIGH"
2360  GOTO 2410
2370  PRINT : PRINT : PRINT
2380  PRINT "WELL YOU DIDN'T MAKE
      HIGH SCORE."
2390  PRINT : HTAB 10
2400  PRINT "KEEP TRYING!"
```

Prompt for next game.

```
2410  VTAB (23)
2420  PRINT "DO YOU WANT TO PLAY
     AGAIN?";
2430  CLEAR : GET A$: PRINT A$: HOME
2440  IF A$ = "Y" THEN 10
2450  END
```

Sound routine.

```
2460  FOR X = 1 TO 36
2470  READ A1,B1
2480  IF  PEEK (49152) > 127 THEN
      POKE 49168,0: GOTO 2510
2490  POKE 7,A1 * 3: POKE 6,B1: CALL
     768
2500  NEXT X
2510  FOR X = 1 TO 1000: NEXT X: RETURN
2520  DATA 40,181,40,181,40,181,2
     0,230,10,152,40,181,40,181,4
     0,181,20,230,10,152,40,181,2
     0,230,10,152,40,181
2530  DATA 20,230,10,152,40,181
2540  DATA 40,181,40,181,40,122,4
     0,122,40,122
2550  DATA 20,117
2560  DATA 10,152
2570  DATA 40,181,40,181,40,181
2580  DATA 20,230,10,152,40,181,2
     0,230,10,152,40,181,20,230,1
     0,152
2590  DATA 60,181
3000  END
```

APPLE™ SWAT TABLE FOR:
ATLANTIS (CASSETTE VERSION)

LINES	SWAT CODE	LENGTH
10 - 120	LS	331
130 - 330	GB	322
340 - 450	PX	127
460 - 570	QV	171
580 - 690	NJ	152
700 - 810	WA	183
820 - 930	WZ	172
940 - 1050	EN	183
1060 - 1170	ZQ	177
1180 - 1290	JT	256
1300 - 1410	SR	279
1420 - 1530	SR	265
1540 - 1650	PX	147
1660 - 1770	SY	179
1780 - 1890	WI	209
1900 - 2010	BJ	213
2020 - 2430	OZ	230
2440 - 2550	UX	307
2560 - 3000	RE	117

APPLE™ SWAT TABLE FOR:
ATLANTIS

LINES	SWAT CODE	LENGTH
10 - 120	LS	331
130 - 240	VM	177
250 - 360	LN	297
370 - 480	TM	127
490 - 600	MW	185
610 - 720	XP	167
730 - 840	FF	167
850 - 960	KY	179
970 - 1080	QM	198
1090 - 1200	JI	202
1210 - 1320	AF	237
1330 - 1440	QD	311
1450 - 1560	TP	198
1570 - 1680	RX	162
1690 - 1800	TR	181
1810 - 1920	BU	216
1930 - 2040	UU	189
2050 - 2160	EF	215
2170 - 2280	MG	361
2290 - 2400	SY	192
2410 - 2520	OV	290
2530 - 3000	QS	195

LOGO: The Programmer–Friendly Language

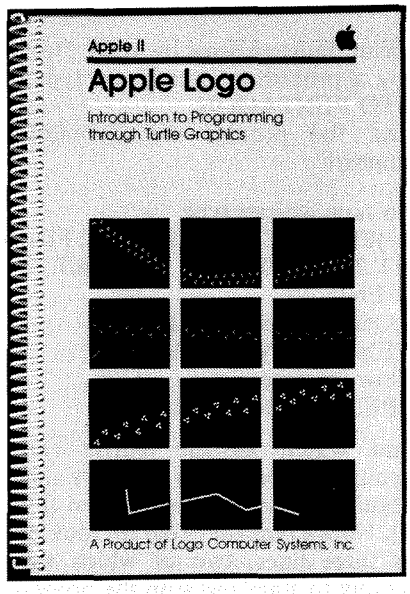
Reviewed by Steve Birchall

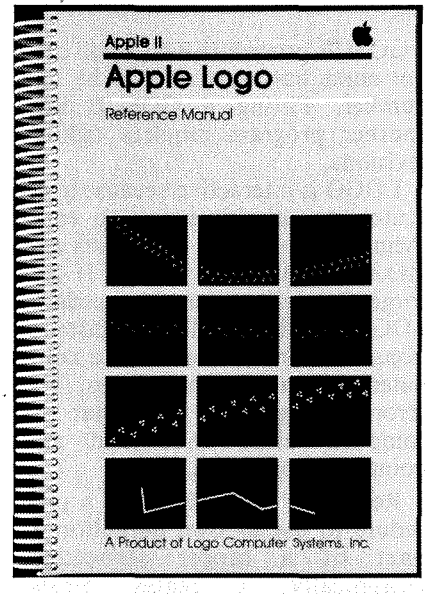
LOGO. Turtle graphics. Kids and computers. These are the first impressions most people have of this wonderful language, newly adapted for microcomputers. But LOGO has much more to offer than these immediately visible features. The other half of the language is a sophisticated set of word and number manipulation commands. Buried in all of this is a fascinating point of view about the interface between computers and the human mind, and how this interaction can enhance learning and creativity.

Growing out of Seymour Papert's work with artificial intelligence, LOGO has become an interactive programming environment suitable for very young children, severely handicapped persons, college students and adults. Make no mistake, LOGO is not a toy language; it is as sophisticated as its user. Papert was a student of the famed French psychologist Piaget, who made the study of human development a major, serious, branch of psychology. Anyone who has taken a course in educational psychology is familiar with Piaget's contributions to our understanding of the development of the human mind. In his book, *Mindstorms: Children, Computers And Powerful Ideas* (Basic Books, Harper Colophon CN 5077, NY 1980, 230pp., $6.95), Papert discusses this interaction between the computer and the human mind as it relates to young children and the learning process. The artifical intelligence project at MIT and at Bolt, Beranek and Newman (the famed acoustical consulting and research firm in Cambridge), led to the development of another language, LISP (LIST PROCESSING) which is a larger, more complex language — the parent of LOGO. The name LOGO itself was coined by Wallace Feurzeig at BB&N and is not an acronym, but takes its meaning from the Greek for thought or word: logos. Presently, LOGO exists in four implementations for the Apple™, plus versions for Texas Instruments, and Radio Shack Color computers.

(1)Apple LOGO, written by LOGO Computer Systems in Quebec and sold by Apple. $175.

(2)MIT LOGO, sold by Krell Software. $150.





(3)MIT LOGO, sold by Terrapin, Inc. $150.

(4)Cyberlogo, a less versatile version which requires only 48K, sold by Cybertronics International, 999 Mount Kemble Ave., Morristown, NJ 07960. $99.

(5)Texas Instruments LOGO has additional graphics concepts called Sprites and some animation capability, but lacks some of the word handling commands of the Apple versions.

(6)Radio Shack's Color Computer version is the most limited (it only takes 32K), providing turtle graphics and some arithmetic capabilities. All have differences, but do substantially the same things, except as noted.

More are on the way, and we should soon see LOGO for the Atari® and IBM® PC, among others.

Structure

Overall, LOGO is a language which is procedural, interactive, and recursive. You write short program modules (called procedures) which do particular things, give them descriptive names, and chain them together for larger tasks. It "learns" procedures which you "teach" it and can use these later in more complex program structures. If the language lacks a command (called primitives in LOGO) you would like to have, simply define it as a procedure and it will always be available to you by name. Try that with BASIC, and you must resort to

Figure 1

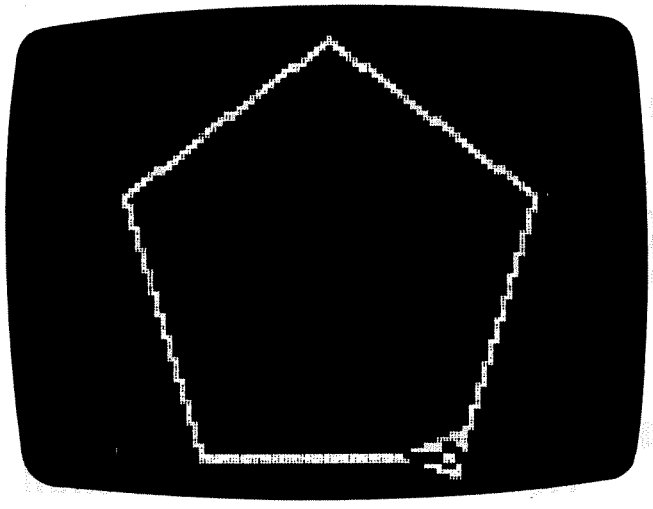
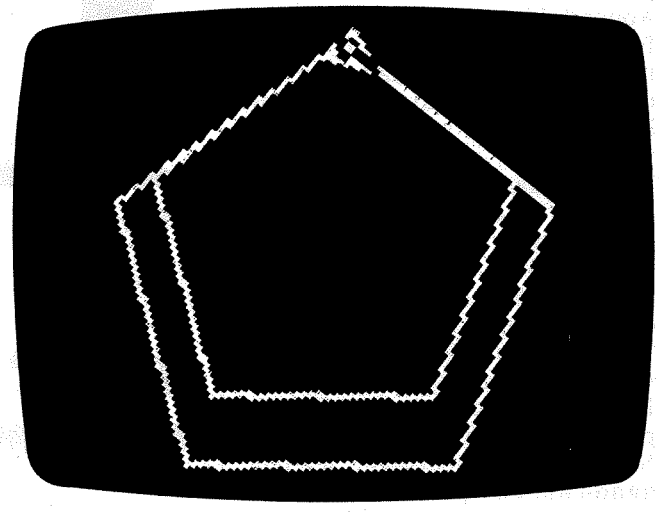


Figure 2



a GOSUB procedure which you cannot name but must refer to by line number. Fitting a GOSUB into another program requires tedious revisions.

LOGO is interactive because it executes the primitives as you enter them so you can see the results and make corrections as you go. If you forget to define a new procedure, LOGO simply asks for the instructions, and from then on the computer remembers what to do. The error messages tell you what the computer can't understand and point to a correction.

Recursion is the ability of a procedure to use its own definition as part of a larger structure. Graphically, it creates spirals, mazes, and other designs. Used in number and word operations, recursion produces amazing patterns, even pseudo-poetry.

Turtle Graphics

To give you a feeling for how easily LOGO draws you into its environment, I will "crawl through" some short examples of LOGO programs using the Turtle Graphics for which it is well-known. The screen says "WELCOME TO LOGO" and below that is the prompt character, a "?". Since everybody starts explaining Turtle Graphics by drawing a square, let's make it just a little more interesting and ask the Turtle to draw a pentagon, which is a difficult figure to draw by hand. Defining a procedure is easy: we type in

```
?TO PENTAGON
```

and the prompt for definitions appears

So now we can type

```
>)FD 50 RT 72 FD 50 RT 72 FD 50 RT 72 FD 50 RT 72 FD 50 RT 72
>END
```

LOGO responds by saying

```
PENTAGON DEFINED
```

Now we type

```
?PENTAGON
```

and watch as the Turtle races around the boundary of a five sided regular polygon on our screen (Fig. 1). Notice that the program took only three lines, two of which were simply to start and stop the process. How much thinking and cleverness were required? We had to do nothing more than tell the Turtle how far to move forward, which direction and how far to turn, and repeat the process five times. Knowing that pentagons have an interior angle of 72 degrees (i.e. 360/5) presumably came from earlier experimentation with squares and other polygons and was acquired intuitively and interactively. Try doing this in BASIC, Pascal or (shudder) FORTRAN. Line two is rather tedious. You wouldn't want to do that for a 500 sided figure. So LOGO has a shorthand notation:

```
REPEAT 5 [FD 50 RT 72]
```

To make pentagons of any size, we could make the instructions more general and write the program

```
?TO PENTAGON :SIDE
  REPEAT = [FD :SIDE RT 72]
  END

?PENTAGON 60
```

If we had typed in just the primitive PENTAGON, LOGO would have replied "NOT ENOUGH INPUTS" to remind us that we have to specify the length of a side in this revision of the program. Now the Turtle has drawn another, smaller pentagon over the first one (Fig. 2). Drat! Forgot to clear the screen. But this looks interesting anyway, so we start filling the screen with pentagons of different sizes. We begin to wonder what would happen if we did this

```
?REPEAT 10 (PENTAGON 45 RT 36)
```

Here the Turtle has drawn 10 pentagons 36 degrees apart. This is more like it (Fig.3)! Wow, let me see if I could make it...

There you see the essential spirit of LOGO. It turns you on intellectually. LOGO leads you to ask more and more questions about abstract ideas and their relationships. LOGO gives you the answers, leading you to more questions. Within a short time you are writing complicated programs easily. Because the programs can use (by means of a one word primitive-label) already established procedures as building blocks for more elaborate structures, LOGO programs are easy to understand. For the last program, the comparable listing in BASIC

Figure 3

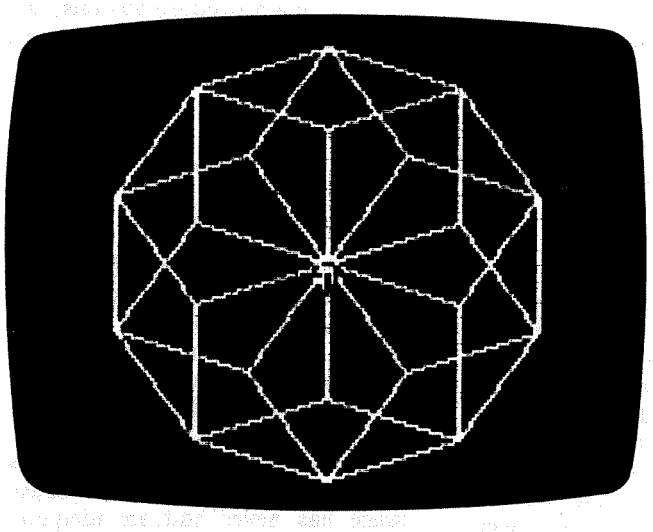
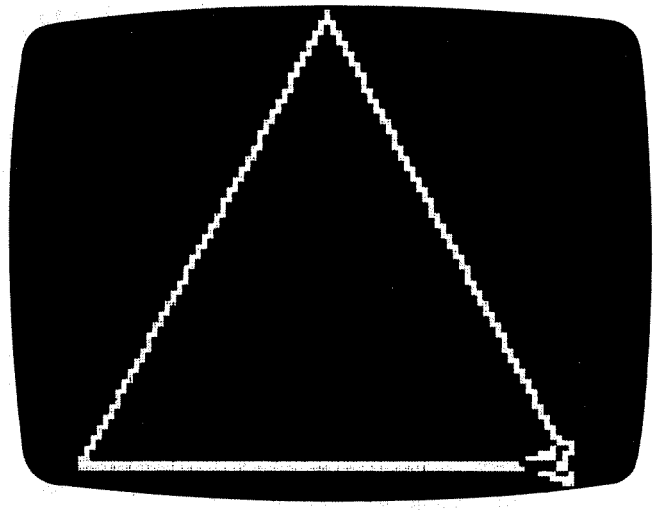


Figure 4



would be indecipherable as to what it does (as well as longer and more difficult to write and debug). You can save all your work on disk. Tomorrow when you come back to the computer, you can start where you left off. A good example of what might take place in the next day's work is writing a program to generate polygons with any number of sides — triangles, squares, or 25-sided figures. With only a small amount of trial and error, you will find out how to derive the proper angle and generalize the instruction set.

```
?TO POLYGON :SIDES
:LENGTH
  REPEAT :SIDE [FD :LENGTH
RT 360/SIDES]
END
```

To draw a triangle, we type in

```
?POLYGON 3 100
```

and the Turtle draws a triangle with sides 100 units long (Fig. 4). You wonder how you could draw a circle? Thinking of a circle as a polygon with an infinite number of sides opens the door to further thought. This is only the beginning of learning much more interesting things to do. Writing more sophisticated programs for drawing a house or a portrait of a friend is a practical possibility.

Words And Lists

In word and list operations, LOGO can be quite useful. To start, LOGO has some primitives to take care of routine operations.

WORD assembles letters, numbers, or groups of them into a single entity.

```
?WORD [C OMP U TE R]
COMPUTER

?WORD (12 56)
1256

?WORD (1 25 6) + (54 34)
6690
```

A list is a group of words set off with square brackets.

```
[BLUE FOX RUNS]
```

SENTENCE puts words together. FIRST, BUTFIRST, BUTLAST and LAST give you particular elements of a list or word, much as LEFT$, MID$, and RIGHT$ do in BASIC

```
?FIRST [BLUE FOX RUNS]
BLUE

?BUTFIRST [BLUE FOX RUNS]
FOX RUNS

?BUTLAST [BLUE FOX RUNS]
BLUE FOX

?FIRST [12 36 48 144]
12
```

Some interesting word games are possible with these primitives. You could make a list of computer buzz phrases very easily by using the PICKRANDOM procedure.

```
?TO BUZZPHRASE
 >MAKE "WORD1 (STATIC
FLOPPY DOUBLE.DENSITY
MICRO PERSONAL INTERNAL
DIGITAL MODULAR)
 >MAKE "WORD2 (BIT ROM
ALGORITHM BYTE HEX AR-
RAY CHARACTER ALIEN DOS)
 >MAKE "WORD3 (GEN-
ERATOR RAM CONTROLLER
FIRMWARE SPREADSHEET
SOFTWARE INTERFACE
GRAPHICS)
 >WRITE
 >END


?TO WRITE
 >PRINT SENTENCE
(SENTENCE (PICKRANDOM
:WORD1) (PICKRANDOM
:WORD2))(PICKRANDOM
:WORD3)
 >WRITE
 >END


?BUZZWORD
INTERNAL ALIEN GRAPHICS
INVERSE DOS SPREADSHEET
FLOPPY ALGORITHM
GENERATOR
PERSONAL HEX
CONTROLLER
MICRO ALIEN SOFTWARE
INVERSE ALIEN FIRMWARE
FLOPPY ARRAY
CONTROLLER
DOUBLE.DENSITY ALIEN
INTERFACE
FLOPPY BYTE
```

REGISTER 16BIT CHARACTER INTERFACE

The period between DOUBLE and DENSITY is a LOGO convention which tells the interpreter to consider it as one word, while permitting you to see it as two. While you can have fun playing with procedures like this, you can also see the possibilities for extending them into more complex structures by using them as building blocks for larger programs which manipulate words and lists.

Since LOGO is interactive, you can test and debug as you go, avoiding problems before they become serious. If you want to get really fancy, you can use title pages, menus, instructions and other helps, just as you are accustomed to doing in BASIC. Once you have written procedures for these kinds of main program utilities, you can save the entire group of procedures and use them whenever you write other large programs. Once you teach LOGO a procedure, you can use it as a module in larger and more difficult programs. You don't have to remember and specify in detail how to do it each time. Just use the name of the procedure. If you need to know how the procedure works, LOGO will give you a listing. Also, consider how "user friendly" programs written in LOGO can be. More importantly, consider how "programmer friendly" LOGO itself is. This is one of the most important aspects of this lanuage: it makes writing a program easy by making the function of a procedure obvious, and by letting you give it a name providing instant recognition of its' function.

"Since LOGO is interactive, you can test and debug as you go, avoiding problems before they become serious."

Notation For Choreography

At a meeting of the Boston Computer Society's LOGO Users Group, I observed an unexpected application for LOGO. Michael Grandfield, a student at Lesley College, had discovered that LOGO could form the basis for an Interactive Dance Notation. This is a truly revolutionary idea, because dance has never had an adequate notation, (The existing systems such as Labanotation are woefully incapable of conveying essential information.) and, traditionally, dance has been passed on from person to person as an oral and demonstrative tradition. Grandfield had conjured

continued on page 82

SOFTSIDE ORDERING INFORMATION

FORM OF PAYMENT

USA

VISA, MasterCard, certified checks, money orders and personal checks are accepted.

Canada/Mexico

The preferred method of payment is by VISA or MasterCard. A bank check is acceptable if it has been preprinted for payment in U.S. dollars. No personal or company checks accepted.

Other Foreign Orders

Payment must either be by a bank check drawn on a U.S. bank payable in U.S. dollars or by affiliated bank credit cards of VISA or MasterCard.

GUARANTEE

All software is guaranteed to load and run. If you experience difficulties with the product within 30 days, it may be returned for replacement. Send your properly protected tape or disk to the attention of the Customer Service Representative and include your name, address, and the reason it is being returned.

LIABILITY

All software is sold on an as-is basis. **SoftSide** assumes no liability for loss or damage caused or alleged to be caused directly or indirectly by products sold or exchanged by them or their distributors, including, but not limited to, any interruption in service, loss of business or anticipatory profits or consequential damages resulting from use or operation of such software.

PRICES

Prices are subject to change without notice. We are not responsible for typographical errors.

Unless otherwise noted in a published advertisement, the following prices are in effect as of this issue:

	USA/Canada APO/FPO	USA/Canada FIRST CLASS Mexico	Other Foreign
SoftSide Magazine (yr)	$30	$40	$62
SoftSide Magazine (6 mo.)	$15	$20	$31

	USA APO/FPO	Mexico Canada	Other Foreign
CV (year) & magazine	$75	$95	$125
(6 mo.)	$40	$50	$70
DV (year) & magazine	$125	$145	$175
(6 mo.)	$70	$80	$90

Adventure of the Month	**Disks**	**Cassettes**
3-Mo. Trial	$29	$19
6-Mo.	$49	$29
12-Mo.	$89	$49
24-Mo. Charter	$169	$89

BACK ISSUES

Minimum order for magazines only — 3 issues. There is no minimum order for magazine/media combinations.
Price includes shipping to the 48 states only. Alaska, Hawaii, Puerto Rico, APO/FPO, and ALL foreign orders — postage is additional.

ALL Foreign orders and all magazine/media combination orders — Order directly from **SoftSide, 6 South St., Milford, NH 03055.**

The Adventure is Waiting for You...

How would you like to go back in time to 19th century London to match wits with Jack the Ripper? Out into space to brave the swirling vortex of a black hole? Into the depths of the ocean, or on a quest to rescue a beautiful princess from the clutches of evil monsters?

You never know where **SoftSide Magazine's Adventure of the Month Club** might take you. But you can be sure that each month you will experience new delights and new challenges as you receive an original adventure on tape or disk, ready to load into your computer.

And now it's even easier for you to join **Adventure of the Month.** A Trial Membership (3 months, 3 different Adventures), costs only $29 for Disks, $19 for Cassettes.

Or choose a Charter Membership, available on disk or tape, for 6 months, 12 months, or 24 months. Your choice of a 24 month Charter Membership will bring you a savings of almost half on the individual price of the Adventures you'll receive.

Super Disks, each containing three Adventures, are also available for $26 each.

Adventure #19 — Alaskan Adventure

Brrrrr, it's **cold.** You're in Alaska, standing in the snow. Around you, fifteen treasures are hidden. Some are buried in the snow, others are cached in nearby igloos. Some are in even stranger places, and you will need considerable ingenuity to get them. Everything has a purpose, so use your imagination. And get moving. There's a nice, warm parka nearby, but you'll turn blue if you don't find it soon.

Adventures Available:

June 1981
Arabian Adventure

July 1981
Alien Adventure

August 1981
Treasure Island

September 1981
Jack The Ripper

October 1981
Crime Adventure

November 1981
Around the World in Eighty Days

December 1981
Black Hole

January 1982
Windsloe Mansion

February 1982
Klondike Adventure

March 1982
James Brand

April 1982
Witches' Brew

May 1982
Titanic

June #13
Arrow One

July #14
Robin Hood

August #15
The Mouse That Ate Chicago

September #16
Menagerie

October #17
The Deadly Game

November #18
The Dalton Gang

For more detailed information on prices, and to enter your membership, see the handy postage-free bind-in card found facing page 80 of this issue.

LOGO *continued*

up a screenful of animated figures leaping and running around an imaginary stage quite realistically. Moreover, the primitives to do all of this were simple words which almost anyone could learn quickly (TURN 30 LEAP 10). For the first time in the history of dance, a choreographer has the opportunity to notate his dance and actually see the patterns of movements — the visual rhythms — displayed in real time. He can make changes and perfect the dance composition's organization and structure before teaching it to an ensemble of dancers (which is time-consuming and expensive). Because the notation conveys the movements and stage positions to the dancers so quickly and graphically, the process of learning the choregraphy is faster (and perhaps will become cheaper). You can repeat particular segments, freeze the display, slow it down, or even have the figures draw lines on the screen to show their movements and make understanding easier. What remains is to teach the fine points of the interpretation. This is what musicians are accustomed to doing with their combination of notation and the unwritten tradition of style and interpretation passed on from teacher to student, generation to generation.

How Is It Done?

Besides the Turtle, LOGO can have figures called Sprites. These have user-definable shapes and can be animated. (The Texas Instruments version has some Sprite capabilities built in.) To do it with Apple LOGO requires an extra board with 16K more memory and an auxiliary video microprocessor. The results are astonishing: the Sprites move completely independently of each other in speed and direction, have different shapes, and can change those shapes while moving — all against a background drawn by the Turtle. The background could range from a simple perspective view of a bare stage to a complete set. (Since the sets can be changed instantly onscreen, some interesting challenges to existing theatre stagecraft result.) The first step in Grandfield's process is to use the shape editor to create outlines of dancers in various positions. A total of 55 Sprite shapes is possible. The editor is very easy to use. You select the shape you wish to edit, and it presents an expanded view divided into a gridwork of squares. You move the cursor to the square you

"...the Sprites move completely independently of each other in speed and direction, have different shapes, and can change those shapes while moving..."

continued on page 85

JOURNEY TO THE PLANETS™

A space, adventure, and arcade action game for your ATARI® 400/800™ personal computer.

NEW

What is your favorite type of game; space, arcade, or adventure? "Journey To The Planets" presents an intriguing combination of all three as you find yourself on a strange planet in a strange universe. Luckily, the local gods are friendly and supply you with energy, a spaceship, and weaponry. In turn, you agree to search the universe for treasures for the gods. Board your ship, take off, accelerate through the upper atmosphere and out into space. Your flight should take you past many other inviting planets. With a slow approach and skilled maneuvering, you drop down through the planet's sky to a soft landing on its surface. Disembark and wander through several TV screens full of mystery and excitement. A different adventure awaits you on each planet.

Planetary adventures are designed to exercise your puzzle solving intellect, with arcade action thrown in to enhance the excitement. Although you are given as much time as necessary to solve each adventure, your score, which is based on many factors, favors those who are speedy.

After you capture all the treasures this universe has to offer, return to your adopted planet. Who knows, the gods may be so happy that you can convince them to send you back to your real home. You can't get there without their help!

Available from your local Atari retailer or send $29.95 in check or money order (California residents add 6½% sales tax) to JV Software Inc.

Atari® and 400/800™ are trademarks of ATARI, Inc.

A 32K assembly language program written for your ATARI® 400/800™ computer.

Other products by JV Software include Action Quest and Ghost Encounters, both 16K real time adventure games. Available on cassette or diskette for $29.95.

JV SOFTWARE, INC

3090 MARK AVE. SANTA CLARA, CA 95051

Franklin's ACE1000 Runs With The Best!

VisiCalc®, DB Master®, Desktop Plan®—they are all running on the Franklin ACE 1000. Cash flow, budgets, word processing or data base management, business or pleasure, the ACE 1000 runs with the best.

The Franklin ACE 1000 is hardware and software compatible with the Apple® II. Franklin users can choose from an enormous selection of programs —programs that run better on the ACE because it includes 64K of RAM, upper and lower case, VisiCalc keys, a numeric pad and an alpha lock key.

Run with the best. Call or write today for the name of your local authorized Franklin dealer.

FRANKLIN
COMPUTER CORPORATION

7030 Colonial Highway
Pennsauken, NJ 08109
609-488-1700

want to change, and turn it on or off, continuing until you have the shape you want. When reduced to normal size, its outlines become smoother and more rounded. Grandfield used his 55 available Sprite shapes to create several series of figures in slightly different positions and postures. (They would animate if placed in a flipbook format.) Each series contains the movements for a leap or jete or whatever dance movement the choreographer wants to include in his working vocabulary.

Dress Rehearsal

Next, he wrote LOGO procedures to present these Sprite shapes in the proper sequence to make his dancers walk, leap, and turn as they move across the stage. When you say LEAP, LOGO executes the primitives. Grandfield gave his dancers whimsical names (like Fred and Ginger) so he could identify each one easily onscreen and in the instruction sets. A complete dance sequence becomes nothing more difficult than writing a list of instructions using readily-recognizable words, rather than PEEK, POKE, HPLOT, etc. You could write:

```
?TELL FRED (SETSPEED 35
TURNLEFT 90 WALK 3
SETSPEED 50 LEAP 10)

? TELL GINGER (SETSPEED 20
TURNRIGHT 45 WALK 2
SETSPEED 20 JETE 45)
```

and watch the two figures move according to your instructions.

To make all the dancers move in synchronization, you write another short LOGO procedure which executes the first command in each dancer's list, then the second, and so on, until the lists are exhausted. A common clock rate or beat applies to all the dancers, regardless of the speed set, so you can plan all the movements. The wonderful thing about all of this is that LOGO permits you to write a program and accomplish an extraordinarily useful task with ease and elegance. To write a similar program in BASIC would be time-consuming, tedious, boring, and so intimidating most people would never attempt it. Grandfield, whose education is in dance, not computer engineering, did this after only a few months' exposure to LOGO.

In the Krell and Terrapin versions, LOGO has musical capabilities. Not only can music and dance be synchronized but new kinds of audiovisual arts become possible, created directly for the medium (using all the visual transformations of the Sprites), or translated into live performance music and dance.

Grandfield worked with LCSI (developers of Apple LOGO) in Montreal this past summer, helping them work out their Sprite board. It should be available late in 1982 or early 1983. No doubt, it will include some of his choreographic work in its documentation. A dancer himself, he said, "I don't think there is a more flexible or appropriate language to work out dance."

This interactive, user-friendly, animated dance notation is one of the most revolutionary developments in the history of choreography. The impact of this invention on the future of dance will be profound, and Michael Grandfield has become a revolutionary figure for having the imagination and foresight to put these ideas together so beautifully and effectively. His application is an excellent example of what Seymour Papert had in mind when he began developing LOGO.

Each successive generation of computer languages has made the computer more available and friendly to non-experts. BASIC certainly has made that claim, and has also succeeded admirably in making computers accessible to more people. BASIC has its faults, but the marriage of BASIC to a microprocessor was the genesis of the personal computer industry. LOGO offers possibilities beyond BASIC, in the same way that BASIC was an improvement over Fortran, and Fortran an improvement over machine code, etc. No software written in LOGO has appeared on the market. However, that will change rapidly. For the moment, memory is the biggest limitation, because LOGO occupies nearly all of 64K, leaving very little for input and subsequent processing. However, when LOGO is available for a computer with larger memory, like an Apple III, or an IBM PC, we should see some ingenious software developed for these machines. The graphics capability is awesome and specialized problem-solving programs would be a natural outcome. Michael Grandfield's Interactive Dance Notation System is but the tip of the iceberg. Complex new games with amazing graphics, sophisticated word and list handling, and interactive techniques could develop. Think of games combining the strategy of chess with the word manipulation of Scrabble and player interactions of Monopoly. Imagine what an amazing Word Processor could be written in LOGO. Animated Business Graphics, with the Turtle tracing the curve, would be great. All of these projects and more are waiting to be explored.

Weak and Strong Points

LOGO does have some problems. The most obvious limitation is that it consumes nearly all of the available memory on a 64K machine. To use Sprites mandates an additional memory and auxiliary microprocessor. Right now, all this is relatively expensive. But the price of memory is coming down and a sudden surge of demand from many LOGO users could drive it down very rapidly. Also, making LOGO the resident lanuage, instead of BASIC, would change the

"LOGO offers possibilities beyond BASIC, in the same way that BASIC was an improvement over Fortran, and Fortran an improvement over machine code, etc."

picture considerably. Another problem is the process of storing and retrieving data. You must save it in the form of a list. For BASIC users, this requires some adjustment and looks like a potential problem. However, once you get the feel for this system, you begin to appreciate the extra potential of being able to use LOGO's word and list primitives on data.

I have heard comments on the fact that LOGO has no provisions for comments in programs. That sounds like a very serious criticism, because, as everyone knows, REMS are important in writing a good program. They help both you and the user to remember what various sections do. Moreover, they are enormously helpful in the debugging process. The answer is that LOGO does not need REMs because they are built into the language. The names you give to your procedures are also REMs, and the advantage of this approach is that you can't avoid using REMs. As a result, you always know what to expect when you ask for a procedure; its function is implicit in the name you gave it.

"LOGO takes us one step further in making computers more useful to people, so that they can tackle problems previously too difficult or time-consuming."

LOGO users generally have an active interest in debugging their programs. A good attitude toward debugging is built into the language because its interactive style gives you quick feedback — you know right away whether a procedure will work or not. With languages like BASIC or Pascal, you must write long sections of code before you have something which you can run to see if it works. In LOGO, you debug and perfect as you go, which engenders an entirely different attitude toward programming. No one in his right mind is going to be satisfied watching the turtle make the wrong movements. LOGO also has good error messages which help to locate the source of a bug and what kind of problem the program has.

What other exciting applications are waiting for someone to discover? LOGO takes us one step further in making computers more useful to people, so that they can tackle problems previously too difficult or time-consuming. Already, LOGO is stimulating the demand for home computers in the same way VisiCalc® did just a few years ago. This significant improvement in the man/machine interface will unleash enormous amounts of human creative potential when more people start using computers in their daily lives.

Apple™ Logo

Reviewed by Steve Birchall

by Harold Abelson from BYTE/ McGraw-Hill, Peterborough, NH, 1982, 224 pages, $14.95. (This edition is intended for users of Apple LOGO, and includes an appendix for Texas Instruments. Another edition of the same book, *LOGO For The Apple II,* is almost exactly the same, but is for users of MIT LOGO from Krell or Terrapin. Includes the Texas Instruments appendix.)

This is an excellent tutorial on LOGO, written by one of its original developers at MIT. If you are at all interested in LOGO, read through this book before you buy the package, because it will give you an excellent overview of how the language works, what it can do, and how to use it. Abelson writes in an easy, conversational style, taking each of the principal topics in turn. Being more adventurous than most, and perhaps impatient to cover the field rapidly, I tried out the Turtle Graphics at the beginning of the book, and then dipped into the later chapters at random, wherever my interest led me. I fumbled a little, and had to backtrack a few times. Because of the nature of LOGO and the clarity of Abelson's writing, however, I was able to learn the new primitives and procedures easily and without undue frustration.

All the forms of LOGO come with thorough documentation, so why is a book like this necessary? Abelson shows how to use the language. The supplied manuals simply explain the primitives in a general sense, without drawing conclusions about practical uses and relating them to your needs in writing programs. Abelson gives you that practical, working knowledge usually passed along verbally by a skilled teacher. Much of the "basic folklore" of working with LOGO is in the book, including the famous *Polyspiral* procedure, the *Doctor* program, and a LOGO version of *Animals*. Lots of useful routines and working knowledge, the kinds of things you would normally acquire by long hours of experimentation, are presented. Always, he gives examples and explains how and why the procedures work. If you go through the book from cover to cover, working the examples as you go, you will certainly end up as a skilled practitioner of LOGO. In particular, his exposition of the word and list manipulation primitives is excellent. Far too many people are distracted by the Turtle Graphics and forget to go any further.

As good as the book is, it has some faults. One important thing he overlooks is how to get printouts of Turtle Graphics. Neither his book nor the supplied manuals tell you how. Understandably, this varies considerably depending on the printer, controller card, and software you use, but some general approaches would be helpful. The book is explicit in other areas, so why not here? Not everyone can figure this out without help. Frequently, part of this process involves storing the design on the screen as a picture file on disk, and this also is not covered in the manuals or the book. However, a brief item in the September, 1982 *POLYSPIRAL,* the newsletter of the Boston Computer Society's LOGO Users Group, provides the key. Put an initialized DOS 3.3 disk in the disk drive and type

?.PRINTER 6

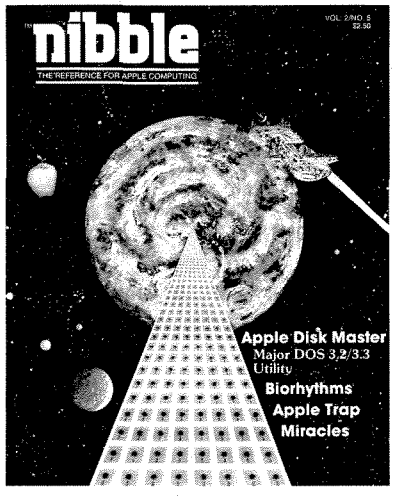
This loads the disk (assuming your drive is on Slot 6), but wipes out LOGO, and any procedures in the workspace not already saved. But the graphics buffer is still OK, so now you can type from BASIC

BSAVE < filename >, A$2000,L$2000

and this saves the picture onto the disk. From there you can run your screen dump program, and print the picture. They also give routines for using the *Silentype* printer and the *Grappler* card, referring to Terrapin's new technical manual for more details. Other possible interesting uses for pictures stored on disk include presenting a sequence of prepared pictures to illustrate a story or a lecture or to present business graphics at a meeting.

Finally, Abelson doesn't give enough attention to the process of writing a larger program. This could be the subject of a companion volume, and would be a welcome addition. The present book excels at explaining short procedures, but leaves the structure of larger programs nearly untouched. His chapters on interactive programs and the two examples (*Doctor* and *Animals*) really only hint at this topic. What are the idiomatic ways of assembling larger programs? Do you just string a bunch of procedures together, or try to assemble them into intermediate sized subprocedures which you can shuffle around? Should you start at the level of building a library of procedures to draw on, or work from the end back to the beginning of the problem, inventing new procedures as needed? How do you make title pages, instructions to the user, or menus? How do you construct the timing loops to make them stay on the screen and fade out? How do you organize your screen displays? (LOGO has no tabs, but you could invent them easily.) All of these things may be obvious to some, but certainly not to all, and they need to be presented in an organized way, so that LOGO programmers can make effective use of the language's "user friendliness."

"NIBBLE® IS TERRIFIC" (For Your Apple)



NIBBLE IS: *The Reference for Apple computing!*

NIBBLE IS: One of the Fastest Growing new Magazines in the Personal Computing Field.

NIBBLE IS: Providing Comprehensive, Useful and Instructive Programs for the Home, Small Business, and Entertainment.

NIBBLE IS: A Reference to Graphics, Games, Systems Programming Tips, Product News and Reviews, Hardware Construction Projects, and a host of other features.

NIBBLE IS: A magazine suitable for both the Beginner and the Advanced Programmer.

Each issue of NIBBLE features significant new Programs of Commercial Quality. Here's what some of our Readers say:

- *"Certainly the best magazine on the Apple II"*
- *"Programs remarkably easy to enter"*
- *"Stimulating and Informative; So much so that this is the first computer magazine I've subscribed to!"*
- *"Impressed with the quality and content."*
- *"NIBBLE IS TERRIFIC!"*

In coming issues, look for:

☐ Stocks and Commodities Charting ☐ Assembly Language Programming Column
☐ Pascal Programming Column ☐ Data Base Programs for Home and Business
☐ Personal Investment Analysis ☐ Electronic Secretary for Time Management
☐ The GIZMO Business Simulation Game

And many many more!

NIBBLE is focused completely on the Apple Computer systems.

Buy NIBBLE through your local Apple Dealer or subscribe now with the coupon below.

Try a NIBBLE!

nibble

We accept Master Charge & Visa

Box 325, Lincoln, MA. 01773 (617) 259-9710

I'll try nibble!
Enclosed is my $19.95 (for 8 issues)
(Outside U.S., see special note on this page.)

☐ **check** ☐ **money order**

Your subscription will begin with the next issue published after receipt of your check/money order.

Card # _____ Expires _____

Signature _____

Name _____

Address _____

City _____

State _____ Zip _____

NOTE:

— Domestic U.S. First Class subscription rate is $36.50
— Canada Air Mail subscription rate is $42.50
— Outside the U.S. and Canada Air mail subscription rate is $47.50

All payments must be in U.S. funds drawn on a U.S. bank.

MASTER BLASTER

BY STEVEN WONG

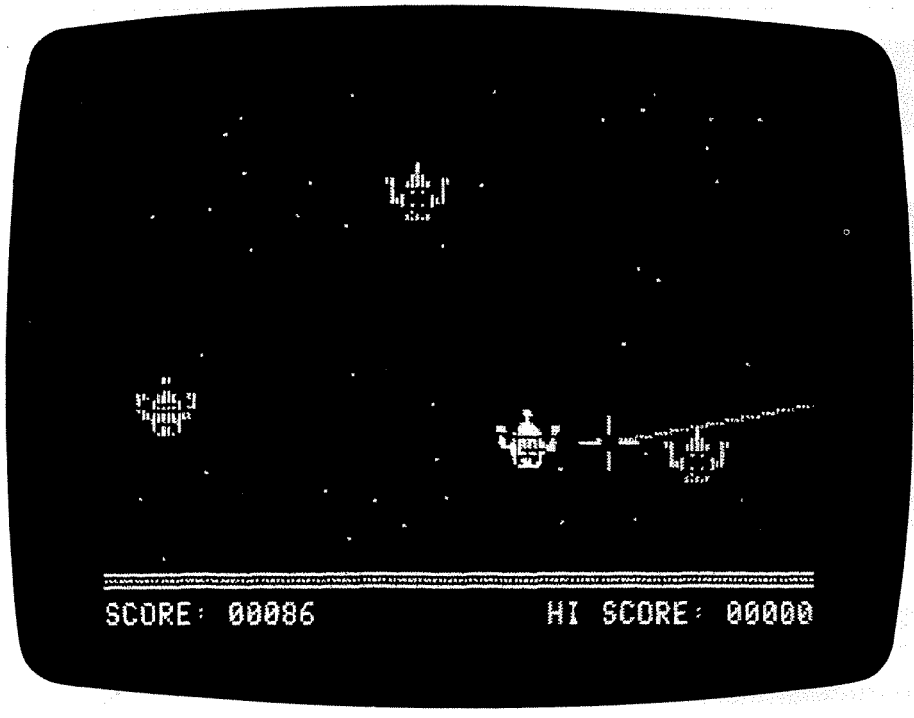
Master Blaster **is an arcade-style game for the Apple™, requiring 16K (32K with disk), Applesoft, and paddles or a joystick.**

The object of this game is to stop the attacking aliens before they break through your deflector screens and destroy your planet. Each alien that manages to reach the screens will weaken them, until they are no longer able to repel the onslaught.

You have at your disposal a powerful laser to defend against the invading horde. Aim the laser at the descending enemy with the joystick or game paddles, and use either button to fire. The game gets increasingly more difficult, because after every fourth alien is destroyed, the rate of descent increases.

Variables

A,B: Loop counters.
CO(*): Colors of aliens.
DF: Flag to indicate if alien has broken through; yes if DF = 1.
FB: Number 127.
H: Rate of alien descent.
KB,KR: Keyboard read and clear (-16384 and -16368).
E,F,J,K,L,LL,M,N,O,Q: Loop variables.
P0,P1: Paddle buttons (-16286 and -16287).
PA: Delay loop variable.
P0(*): Used in determining new random vertical position for an alien.
SC: Score.
W(*): Position of deflector screens.
X(*),Y(*): Coordinates of aliens.
XP,YP: Coordinates of crosshair.
Z0: Number 255.
Z4,Z5: Numbers 259 and 143, used to compute paddle range.




```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS      APPLESOFT BASIC        SS
SS      'MASTER BLASTER'       SS
SS    AUTHOR: STEVEN WONG      SS
SS     COPYRIGHT (C) 1982      SS
SS SOFTSIDE PUBLICATIONS, INC SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type this program, it is also included on this month's SoftSide CV and DV.

Initialization.

```
10  GOSUB 6000: GOTO 8000
```

Erase crosshair, get new paddle reading and draw crosshair.

```
30  SCALE= 3:XP =  INT ( PDL (0) /
    Z0 * 259 + 11):YP =  INT ( PDL
    (1) / Z0 * 138 + 3): HCOLOR=
    0: DRAW 1 AT X1,Y1: HCOLOR=
    3: DRAW 1 AT XP,YP: SCALE= 1
    :X1 = XP:Y1 = YP: RETURN
```

Draw laser and make laser sound.

```
40  HCOLOR= 5: HPLOT 0,100 TO XP -
    3,YP: HPLOT 279,100 TO XP +
    3,YP: FOR S = 5 TO  RND (1) *
    75 + 75 STEP 5: & TS,2: NEXT
```

Test for a hit.

```
50  FOR B = 1 TO 4
60  IF  ABS (X(B) - XP) < 11 AND
     ABS (Y(B) - YP) < 9 THEN 10
    0
70  NEXT
```

Erase laser.

```
80  HCOLOR= 0: HPLOT 0,100 TO XP -
    3,YP: HPLOT 279,100 TO XP +
    3,YP
90  RETURN
100  HCOLOR= 0: HPLOT 0,100 TO XP
     - 3,YP: HPLOT 279,100 TO XP
     + 3,YP: DRAW 2 AT X(B),Y(B)
```

Explosion routine.

```
110  FOR K = 1 TO 2: POKE  - 1629
     8,0: FOR M = 100 TO 200 STEP
     50: & TM,5: NEXT : POKE  - 1
     6297,0: FOR M = 200 TO 100 STEP
      - 50: & TM,5: NEXT
115  NEXT K
```

Scoring routine.

```
120 SC = SC + (160 - Y(B)):SC =  INT
    (SC):X(B) =  INT ( RND (1) *
    46 + PO(B)):Y(B) = 5:CC =  INT
    ( RND (1) * 7 + 1):CO(B) = C
    C - (CC = 4)
122  VTAB 21: HTAB 8 + (SC < 9999
    ) + (SC < 999) + (SC < 99): PRINT
    SC
```

Draw some stars and increment rate of descent.

```
125  FOR L = 1 TO 5: HCOLOR= L: HPLOT
     RND (1) * Z4 + 5, RND (1) *
    Z5 + 5: NEXT
130 J = J + 1: IF J > 5 THEN H =
    H + 1.5:J = 0
140  POP
```

Erase alien, move it, check if near deflector screens, draw alien.

```
160  FOR A = 1 TO 4: SCALE= 1
170  HCOLOR= 0: DRAW 2 AT X(A),Y(
    A)
180 Y(A) = Y(A) + H: IF Y(A) =  >
    W(C) - 8 THEN 1000
190  HCOLOR= CO(A): DRAW 2 AT X(A
    ),Y(A)
200  GOSUB 30
205  NEXT
```

Check paddle buttons.

```
210  FOR LL = 1 TO 6: GOSUB 30: IF
     PEEK (P0) > FB OR  PEEK (P1
    ) > FB THEN LL = 6: GOSUB 40
220  NEXT LL
```

Go back and move aliens again.

```
230  GOTO 160
```

An alien has landed. Cause alien to change color.

```
1000  FOR E = 1 TO 7
1020  HCOLOR= E: DRAW 2 AT X(A),W
    (C) - 8
1030  & TE * 35,5: & T255 / E,5
1040  FOR PA = 1 TO 150: NEXT : NEXT
    E
```

Erase alien and one wall.

```
1050  HCOLOR= 0
1055  IF C > 3 THEN DF = 1: GOTO
    1090
1060  FOR D = 0 TO 270 STEP 9
1070  HPLOT D,W(C) TO D + 9,W(C)
1080  FOR E = 100 TO 200 STEP 25:
     & TE,2: & T200 - E,2: NEXT
    : GOSUB 30: HCOLOR= 0: NEXT
1090  DRAW 2 AT X(A),W(C) - 8: IF
    DF THEN Y(A) = W(A) - 8: GOTO
    2000
1100 C = C + 1
1110 X(A) =  INT ( RND (1) * 46 +
    PO(A)):Y(A) = 5
1120  GOTO 210
```

Aliens have broken through.

```
2000  SCALE= 3: HCOLOR= 0: DRAW 1
      AT XP,YP: SCALE= 1:Q = Q +
      1: IF Q > 7 THEN 2110
2005 Q = Q + 1: IF Q > 7 THEN 211
      0
2010  IF Q = 4 THEN 2005
```

Have all aliens change color.

```
2020  FOR A = 1 TO 4
2040  HCOLOR= 0: DRAW 2 AT X(A),Y
      (A)
2050  FOR L = 100 TO 200 STEP 25:
      & TL,5: & T255 - L,5: NEXT
      L
2060  HCOLOR= Q: DRAW 2 AT X(A),Y
      (A)
2070  FOR L = 200 TO 100 STEP  -
      25: & TL,5: & TL + 50,5: NEXT
      L
2080  NEXT A: GOTO 2005
```

Destruction of planet.

```
2110  FOR K = 1 TO 3
2120  POKE  - 16298,0: FOR L = 1 TO
      8: & T200 - L * 10,3: & T160
      ,5: NEXT L
2130  POKE  - 16297,0: FOR L = 8 TO
      1 STEP  - 1: & T160 - L * 10
      ,5: & T100,3: NEXT L
2140  NEXT K
```

Clear Display.

```
2150  POKE  - 16298,0
2160  HCOLOR= 0: FOR B = 1 TO 4: DRAW
      2 AT X(B),Y(B): NEXT : SCALE=
      3: DRAW 1 AT XP,YP
```

Return to hi-res and display stars. Ask if player would like to play again.

```
2161  POKE  - 16297,0
2170  IF SC > HS THEN HS = SC: VTAB
      21: HTAB 36 + (HS < 9) + (HS
      < 99) + (HS < 999) + (HS <
      9999): PRINT HS
2175  VTAB 23: PRINT "DO YOU WANT
      TO"
2180  VTAB 24: PRINT "PLAY AGAIN?
      ";: HTAB 13: FLASH : PRINT
      " ";: NORMAL
```

Make stars blink and wait for answer.

```
2190  IF  RND (1) > .5 THEN  HCOLOR=
      0: GOTO 2200
2195 J =  INT ( RND (1) * 7 + 1):
      HCOLOR= J + (J = 4)
2200 K =  INT ( RND (1) * 10 + 1)
      : ON K GOTO 2310,2320,2330,2
      340,2350,2360,2370,2380,2390
      ,2400
2310 M = 20:N = 20: GOTO 2450
2320  HPLOT 50,130: GOTO 2460
2330 M = 200:N = 145: GOTO 2450
2340  HPLOT 175,100: GOTO 2460
2350 M = 120:N = 75: GOTO 2450
2360  HPLOT 250,40: GOTO 2460
2370 M = 140:N = 100: GOTO 2450
2380  HPLOT 140,80: GOTO 2460
2390 M = 45:N = 125: GOTO 2450
2400  HPLOT 210,90: GOTO 2460
2450  HPLOT M,N TO M + 1,N TO M +
      1,N + 1 TO M,N + 1 TO M,N
2460 LL =  PEEK (KB): POKE KR,0: IF
      LL < 128 THEN 2190
2480  IF LL = 206 THEN 9999
2490  IF LL = 217 THEN 2500
2495  GOTO 2460
```

Clear variables for a new game.

```
2500  VTAB 22: CALL  - 958:SC = 0
      :C = 1:H = 8:B = 0:K = 0:Q =
      0:O = 0:N = 0:R = 0:F = 0:M =
      0:J = 0:A = 0:CC = 0:LL = 0:
      DF = 0:D = 0:L = 0: GOTO 800
      0
```

Display title page. Poke in sound routine.

```
6000  TEXT : HOME : INVERSE
6010  FOR A = 6 TO 12 STEP 6: VTAB
      A: HTAB 10: FOR B = 1 TO 22:
       PRINT "*";: NEXT : NEXT : FOR
      A = 7 TO 11: VTAB A: HTAB 10
      : PRINT "*";: HTAB 31: PRINT
      "*": NEXT
6020  NORMAL : VTAB 8: HTAB 14: PRINT
      "MASTER BLASTER": VTAB 10: HTAB
      14: PRINT "BY STEVEN WONG"
6025  DIM CO(7),X(4),Y(4),PO(4),W
      (4)
6030  FOR K = 7680 TO 7800: READ
      L:D1 = D1 + L: POKE K,L: NEXT
6040  IF L OR D1 <  > 7671 THEN  TEXT
      : HOME : VTAB 8: PRINT "ERRO
      R! CHECK DATA LINES 7000-702
      2": END
6050  FOR K = 1 TO 4: READ CO(K),
      PO(K),W(K): NEXT
```

continued on page 92

MOVING?

If you're planning to move, please let us know at least six weeks in advance. This will help us to change your address insuring you with prompt and accurate service on your subscription. Attach your current mailing label filling in your name and NEW address in the space provided.

Attach
old label
here

Name _____

New Address _____

City _____ **State** ____ **Zip** _____

Send old label with your name and NEW address to:

SoftSide
100 Pine Street
Holmes, PA 19043

APPLE™

Master Blaster *continued*

```
6070  POKE 232,0: POKE 233,30
6080 A$ = "201,084,208,015,032,17
     7,000,032,248,230,138,072,03
     2,183,000,201,044,240,003,07
     6,201,222,032,177,000,032,24
     8,230,104,134,003,134,001,13
     3,000"
6090  FOR K = 1 TO 35: POKE K + 7
     67, VAL ( MID$ (A$,K * 4 - 3
     ,K * 4 - 1)): NEXT
6100 A$ = "170,160,001,132,002,17
     3,048,192,136,208,004,198,00
     1,240,007,202,208,246,166,00
     0,208,239,165,003,133,001,19
     8,002,208,241,096"
6110  FOR K = 1 TO 33: POKE K + 8
     02, VAL ( MID$ (A$,K * 4 - 3
     ,K * 4 - 1)): NEXT
6120  POKE 1013,76: POKE 1014,0: POKE
     1015,3
```

Initialize variables.

```
6125 Z0 = 255
6130 KB =  - 16384:KR =  - 16368
6140 P0 =  - 16286:P1 =  - 16287
6150 FB = 127:H = 8:C = 1
6155 Z4 = 259:Z5 = 143
6160  VTAB 22: HTAB 9: PRINT "< H
     IT ANY KEY TO BEGIN >"
6170 A =  PEEK (KB): IF A > 127 THEN
      POKE KR,0: TEXT : HOME : RETURN
6180 M =  INT ( RND (1) * 100 + 1
     50): & TM,8: FOR PA = 1 TO 1
     00: NEXT PA: GOTO 6170
```

Shape table data.

```
7000  DATA 2,0,6,0,17,0,59,63,72,
     9,100,146,45,213,219,54,0,35
     ,63,54,54,45,36,76,44,53,54,
     62,39,228,192,193,57,63,63,5
     4,62,62,63,36,39,36
7010  DATA 37,63,55,53,54,46,54,4
     5,45,48,46,192,193,49,54,54,
     46,46,45,45,45,37,37,36,36,1
     48,42,36,45,37,44,36,60,44,4
     5
7020  DATA 62,54,54,55,62,63,39,6
     0,36,60,63,63,60,63,55,44,44
     ,45,45,53,53,60,63,39,60,55,
     44,45,53,63,36,36,39,45,190,
     146,146,146,146,218,53,45,44
     ,0
7050  DATA 1,13,152,2,83,154,3,15
     2,156,5,222,158
```

Set graphics. Draw stars and walls.

```
8000  HGR : FOR K = 1 TO 100:M =
     RND (1) * Z4 + 5:N =  RND (
     1) * Z5 + 5:O =  RND (1) * 7
     + 1: HCOLOR= O - (O = 4): HPLOT
     M,N: NEXT
8005  ROT= 0
8007  FOR E = 152 TO 156 STEP 2
8008 K =  INT ( RND (1) * 7 + 1):
      IF K = 4 THEN 8008
8010  HCOLOR= K: HPLOT 0,E TO 279
     ,E: NEXT
8020  FOR F = 1 TO 4:X(F) =  INT
     ( RND (1) * 46 + PO(F)):Y(F)
     = 5 * F: NEXT
8025  COLOR=  INT ( RND (1) * 14 +
     1)
8030  FOR K = 0 TO 39: HLIN 0,39 AT
     K: NEXT
8035  POKE  - 16297,0
8036  FOR K = 1 TO 3: FOR L = 100
      TO 1 STEP  - 3: & TL,4: NEXT
     : FOR L = 1 TO 100 STEP 3: &
     TL,4: NEXT : NEXT
8038  VTAB 21: HTAB 1: PRINT "SCO
     RE: 00000": IF  NOT HS THEN
      VTAB 21: HTAB 26: PRINT "HI
      SCORE: 00000";
8040  GOTO 160
```

End of program.

```
9999  TEXT : HOME : VTAB 10: PRINT
     "WE'LL BE BACK!!!"
10000  IF  PEEK (KB) > 127 THEN 1
     0100
10010  GOTO 10000
10100  POKE KR,0: HOME : END
```

APPLE™ SWAT TABLE FOR: MASTER BLASTER

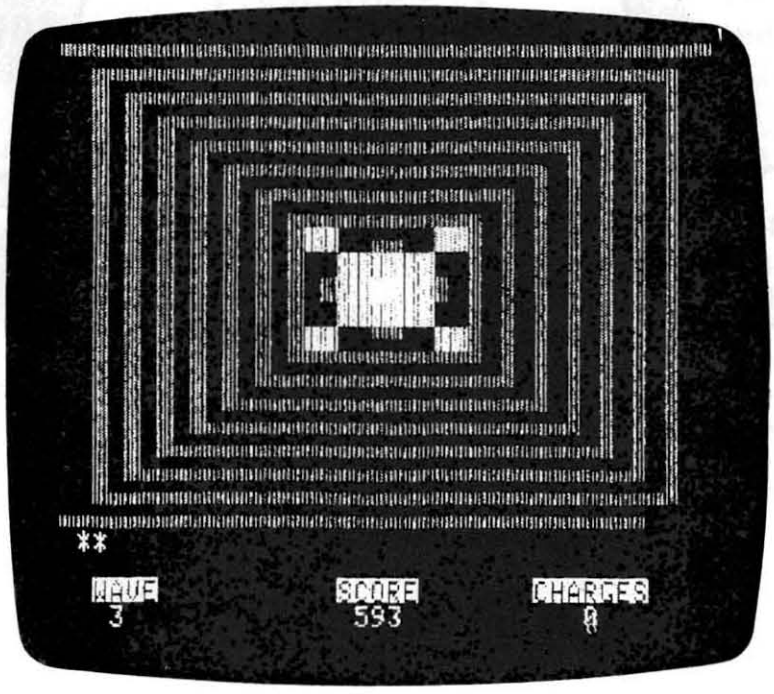
LINES	SWAT CODE	LENGTH
10 - 120	LV	493
122 - 220	AE	271
230 - 1100	DN	243
1110 - 2110	SO	263
2120 - 2200	BH	418
2310 - 2460	PV	262
2480 - 6070	JB	457
6080 - 6160	YZ	530
6170 - 7050	DI	511
8000 - 8040	FL	357
9999 - 10100	FV	78

FORTRESS

by Ronald Azuma

***Fortress* is an arcade-style game program for an Apple™ with Applesoft, 48K RAM, and a disk drive. A joystick is optional. It is included as the bonus program on the issue 36 Apple DV. See the Bind-in Card elsewhere in this issue to order this month's disk.**

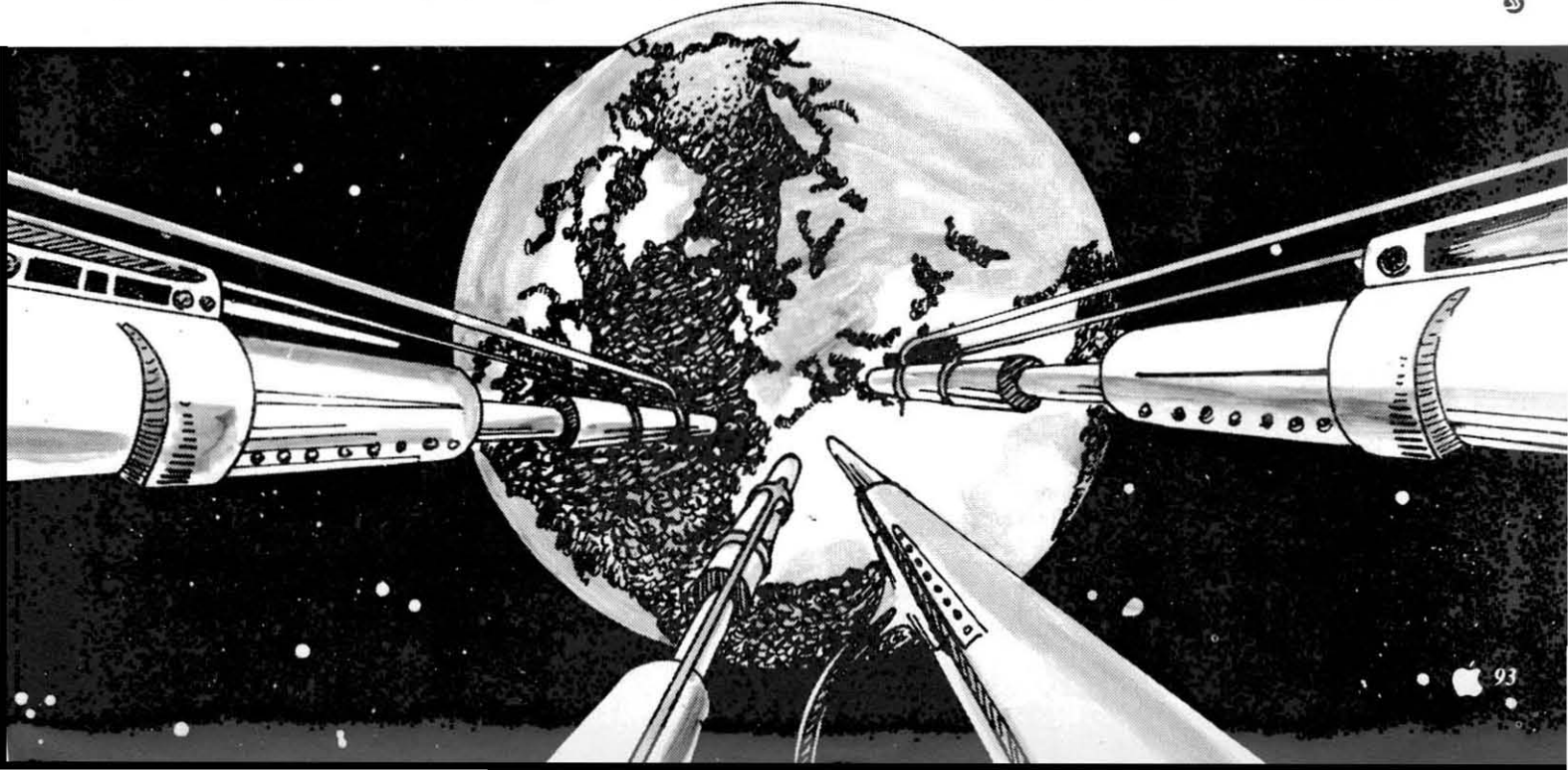
Earth is under attack. You are the commander of the last surviving Star Fortress in the Mars defense line. Your Star Fortress is armed with four laser cannons, one for each compass direction, and each capable of annihilating any enemy ship with a single shot. In addition, you have a limited amount of Smart Bombs that destroy everything on the screen. The enemy is a formidable opponent. He will attack in "waves," fire his nuclear missiles at you, and leave the screen. If one of his missiles hits your Star Fortress, you will be destroyed. Later in the game, the enemy gets faster ships (Wave 4, green ships) and faster missiles (Wave 7, purple ships).



To play *Fortress*, run FORTRESS HELLO. The program initially places you in the demo mode. When you are ready to play, hit RETURN. You can use either the keyboard or a joystick to play the game. If you want keyboard control, use the I-J-K-M keys for firing the lasers, and the space bar for using a Smart Bomb. For those of you who are not familiar with the I-J-K-M pattern, I will fire up, J will fire left, K will fire right, and M will fire down. If you prefer using a joystick, pushing the stick in the correct direction will fire the lasers, while pushing button 0 will activate a Smart Bomb. It would be wise to adjust the joystick before playing the game, so that the program can adapt to your particular joystick. (The program is initially set for a TG Super Joystick.)

You start with two Smart Bombs and a limited supply of laser charges. If you use up all of your charges, your lasers will be useless. You receive an extra Smart Bomb every 5000 points. The number of asterisks in the upper left-hand corner of the text screen indicates how many Smart Bombs remain. You get points for each ship destroyed and for each laser charge remaining at the end of each attack wave. The game ends when your Fortress is destroyed. The program saves the top ten scores, so keep your disk in the drive!

Good luck, Commander! Mankind is depending on you!



Heartland Software
1983 Specials

Tank Arcade
16K Cassette
$13.95

Sea Fox
48K Disk
$23.95

Picnic Paranoia
16K Cassette
$27.95

We Stock

Broderbund	APX
Synapse	PDI
Avalon Hill	JV Software
On-Line	Strategic Simulations
Datasoft	Automated Simulations
Big-Five	Adventure International
Sirius	**New Games Arriving Every Day!**
	Call For Information

Happy New Year from The Heartland

apple® ATARI® TRS-80®

Title	List	Our Price	Title	List	Our Price	Title	List	Our Price
Action Quest	29.95	23.95	Frogger	34.95	25.95	Ricochet	19.95	15.95
Adventure 12-Pack	129.95	100.00	Galaxy Invasion	15.95	12.95	S.A.G.A. 1-12	39.95	29.95 ea.
Andromeda Conquest	23.00	18.95	Genetic Drift	29.95	23.95	Sammy The Sea Serpent	16.95	13.95
Apple Panic	29.95	23.95	Graphics Composer	39.95	29.95	Scarfman	19.95	15.95
B-1 Nuclear Bomber	16.00	12.95	Ghost Encounter	29.95	23.95	Serpentine	34.95	27.95
Battle of Shiloh	39.95	29.95	Invasion Orion	24.95	18.95	Sea Fox	29.95	23.95
Claim Jumper	34.95	27.95	K-Razy Shootout	49.95	37.95	Shamus	34.95	27.95
Commbat	24.95	19.95	Labyrinth	29.95	23.95	Shooting Arcade	29.95	23.95
Convoy Raider	16.00	12.95	Legionnaire	35.00	27.95	Slime	34.95	27.95
Cosmic Balance	39.95	29.95	Micropainter	34.95	27.95	Softporn Adventure	29.95	23.95
Crossfire	29.95	23.95	Mission: Asteroid	24.95	18.95	Space Eggs	29.95	23.95
Curse of Ra	19.95	15.95	Moon Base IO	29.95	23.95	Star Blazer	31.95	24.95
Cyclod	29.95	23.95	Morloc's Tower	19.95	15.95	Star Warrior	39.95	29.95
Datestones of Ryn	19.95	15.95	Mousekattack	34.95	27.95	Stellar Escort	15.95	12.95
Deadly Secrets	34.95	27.95	Midway Campaign	16.00	12.95	Stellar Shuttle	29.95	23.95
Deadline	49.95	37.95	Nautilus	34.95	27.95	Super Nova	15.95	12.95
Defense Command	15.95	12.95	OK Galaxy	20.00	14.95	Tanktics	24.00	18.95
Dnieper River Line	30.00	24.95	Pacific Coast Hwy	29.95	23.95	Temple Of Apshai	39.95	29.95
Dodge Racer	34.95	27.95	Protector	34.95	27.95	Text Wizard	99.95	75.95
Dog Daze	22.95	19.95	Preppie	29.95	23.95	Tigers In The Snow	39.95	29.95
Eastern Front	29.95	23.95	Reptilian	34.95	27.95	Track Attack	29.95	23.95
Fort Apocalypse	34.95	27.95	Rescue At Rigel	29.95	23.95	Visicalc	250.00	199.95

Indicate type of computer, disk or cassette.
For fast delivery, send Certified Check or Money Order.
Personal checks require two weeks to process.
C.O.D. orders add $1.50.
Master Charge or Visa orders add 3%
(Include all embossed information on card)
Add $2.00 shipping and handling.
Prices subject to change.

Make Checks and Money Orders payable to:
HEARTLAND SOFTWARE DISTR.
P.O. Box 25517
Cleveland, Ohio 44125
Ohio Residents add 6.5% Tax
(216) 641-5055
Order Lines Open 10 a.m. - 9:00 p.m. Mon.-Fri.
Noon - 6:00 p.m. Sat.

Send $1.00 for complete catalog

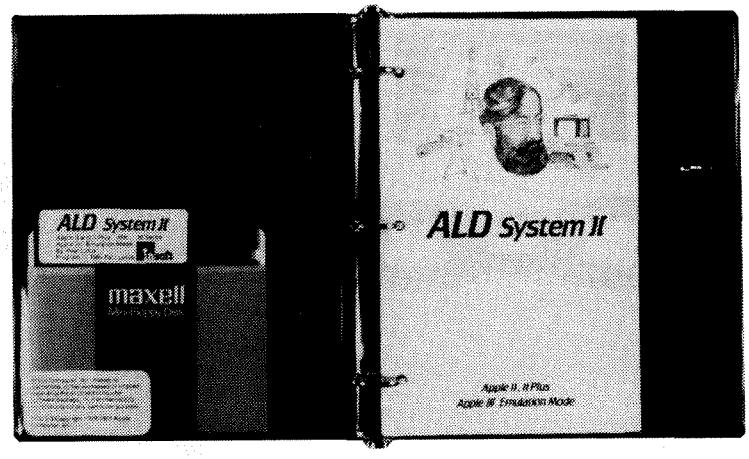
641-5055

(216)

ALD System II

Reviewed by Cary W. Bradley

by Paul Lutus (Insoft, 10175 Southwest Barbur Blvd., Suite 202B, Portland, OR 97219). System requirements: 48K Apple II™ or Apple II Plus, Language Card optional. Suggested retail price: $75.00.



Sometime ago, I read about a fellow who wrote a word processor for the Apple II, working in a cabin in the woods of Oregon, with power supplied by a 1300-foot extension cord. As it turns out, the word processor was the best-selling *Apple Writer*, and the fellow was Paul Lutus. This same man is the author of a number of other successful programs for both the Apple II and the Apple III.

His name started popping up in ads in several magazines, selling a language called TRANSFORTH. Recently, ***SoftSide*** published a review of *Electric Duet* (May, 1982), which produces two musical voices simultaneously through the Apple speaker. The program was written by — you guessed it, Paul Lutus!

During the month or so I spent working with *ALD System II*, for the purpose of this review, an article appeared in the *Wall Street Journal* about a "mountain hermit," who programs computers. Paul Lutus again!

Part of the publicity stems from the fact that Mr. Lutus is an interesting personality. Beyond that, from everything I've read and seen about his creations, this much is clear: Paul Lutus is an exceptionally efficient and proficient programmer, and we Apple users are fortunate to have him working for us. It's easy to see why I had high expectations for *ALD System II.*

ALD (Assembly Language Development) System II is an enhancement of *ALDS I*, also a Lutus production, which is marketed by Hayden Book Company. *ALD System II* is compatible with *ALDS I*, in that source files created on the earlier system can be converted easily to System II by following step-by-step instructions given in the manual.

For those who know nothing about assembly language, let me explain. Assembly language is the way Machine Language programs are written. Machine Language for the 6502 microprocessor consists of instructions which occupy one to three bytes of memory each, and are executed directly by the microprocessor. Everything your Apple does is ultimately done in Machine Language, regardless of what computer language you are using.

Machine Language programs are, for all intents and purposes, unintelligible to human beings. An assembler is a program which allows people to represent the cryptic Machine Language instructions by three-letter mnemonic codes, numeric operands, and symbolic labels. Programs written in this form are then translated into the Machine Language form by the assembler. There are several different assemblers for the Apple on the market. About the only common feature is the set of mnemonic codes, defined by the designers of the microprocessor. If you want to learn to write Machine Language programs, you should talk to people who know about assembly language and can show you the features of the various assemblers available. Compare these features to the ones described in this review to determine which assembler you should buy. *ALD System II* should definitely be a candidate.

ALD System II consists of the two essential elements of an assembly language system, an editor and an assembler. Both are resident in RAM at the same time. This means that you can go back and forth between them instantly without loading anything from the disk. This is one of many features of *ALD System II* which contribute to its speed. The editor is used to create the source file, the one readable by people. The Machine Language file created by the assembler is called the object file.

The *ALD System II* editor is cursor-based. The cursor always appears between two adjacent characters in the source file. If you are accustomed to a line editor, this will take some getting used to, but the advantages of this editor quickly become apparent. Normally, the cursor is the familiar blinking white block. Hit the ESC key once, and it becomes a blinking up-arrow, which means that any character you type will be an inverse character. Hit the ESC key again, and the cursor is a blinking plus sign, which allows you to position the cursor within the file.

continued page 97

ALD *continued*

The I, J, K, and M keys move the cursor up, left, right and down, one line for each keypress. Pressing any other key brings back the normal cursor and allows you to continue editing.

The editor includes an amazingly fast string locating and replacement feature, invoked by CTRL-S. You can also instantaneously position the cursor at the beginning or the end of the file with CTRL-B and CTRL-E. Other CTRL key functions make it simple to delete words or lines, and give you an alternative method for entering inverse characters. Still others allow you to save portions of your source file to the disk, insert saved portions into your source file, display the amount of editor memory remaining, and directly access DOS commands.

Although the editor is not a line editor, line numbers are recognized, but not shown. If an error is flagged during assembly, a line number is given, and when you return to the editor you can jump directly to that line in the source file by using CTRL-J and entering the line number.

Characters are erased from your source file by using the left-arrow key to backspace over them. A 256-byte input buffer is maintained, and characters you delete in this fashion are stored in this buffer on a last-in, first-out basis. These characters are retrieved by using the right-arrow key. This is a tremendously useful feature of the editor. You can move a portion of your file to any other place in the file by positioning the cursor at the end of the portion you want to move, backspacing through the characters, moving the cursor to the new position, and using the right-arrow key to retype the characters you just deleted — simple.

The only thing about the editor which could cause a problem is that, every once in a while, the cursor moves below the portion of the file that is visible on the screen. This happens when you are in the "blinking plus sign" mode, and are using the M key to jump the cursor forward in the file. The cursor always reappears on the next keypress, however, and your editing capabilities are not adversely affected. It's just a little disquieting the first time it happens.

The editor supports the conventional fields used in assembly language; one each for the label, operator, operand and comment. The editor tabs to the next field each time you use the space bar. Thus, it requires only one byte in your source file for each time you tab to the next field in a line. The space between the operator and the operand is not required, but the amount of memory saved by not using it is insignificant, compared to the loss of readability in your source listing. Comments are always preceded by a semicolon, and may appear anywhere they are needed; i.e., you may insert pure comment lines at any place you want them in your source file. The semicolon disables the automatic tab feature, so that your comments appear correctly.

continued on page 98

ALD System II ***continued***

The assembler supports all of the standard 6502 opcode mnemonics. Only one nonstandard mnemonic is used, the GLB operator. It allows the simple construction of two-byte address tables, commonly used as branch destinations for instructions using one of the 6502's indexed addressing modes.

If you've used an assembler with its own set of nonstandard mnemonics for various assembler functions, you may wonder how you'll get along without them. It's easy. Most notably, the use of a nonstandard mnemonic for equates has been replaced by a simpler and more natural procedure. You just enter the label you wish to assign, tab to the operator field, and enter an equals sign followed by the constant the label will represent. Character strings are defined using this same simple method — enclosing the desired string in quotes.

A set of thirteen system operators handles other assembler functions. These require the "@" sign (SHIFT-P) as a prefix, and appear in the label field. With these operators, you can select the address at which the object file will be created, the address at which it will run, the amount of memory set aside for the assembler symbol table, several printing options, or a file to be inserted or appended during assembly, to name just a few of the available options.

Constants in *ALD System II* are all entered in hexadecimal form. Even when an assembler allows me to enter constants in hex, decimal, octal, base 87, etc., I never use anything but hex, because it's the natural form for 6502 instructions. The advantages gained by doing away with alternative numbering systems are a savings in the length of the assembler program itself, and the elimination of the need to prefix hex numbers with a dollar sign. It's acceptable to the assembler if you feel compelled to use the dollar sign, and it is required when specifying a zero-page constant operand in order to prevent the assembler from interpreting it as a two-byte address.

Paul Lutus has some strong opinions about computer language structure, and the *ALD System II* assembler reflects these feelings. It uses a three-level-hierarchy of labels — local, global and universal. Universal and global labels are designated by "&" and "*" prefixes, and local labels have no special prefix. During assembly, the lower-level labels are sought only between the first next higher-level labels appearing before and after the current statement. This creates distinct program segments, allowing you to use simple labels, such as LOOP, NEXT, BRANCH, HERE, THERE, etc., over and over again within a program, provided that each is unique within the program segment to which it belongs.

This structural form, along with the capabilities for saving, recalling, inserting and concatenating source files and source file segments, give the assembly language programmer tremendous advantages. With *ALD System II*, it is simple to create an extensive library of assembly language routines, and to insert them in programs or string them together at will, without concern over label conflicts or file compatibility.

The other point on which Lutus is a real stickler is user-friendliness. *ALD System II* demonstrates this, as well, with fast, simple menus and commands, making the operation of a lot of other software appear to be just so much red tape. If you've ever been driven to distraction by the incessant standard Apple "beep," which most programs use to get your attention, you'll be pleased to know that it is totally absent from this system. I won't tell you any more about this, so that you can be as pleasantly surprised as I was on my first venture into *ALD System II*.

For easy identification, by the editor, the assembler and you, source file names are given the prefix "SRC.", and object files, the prefix "OBJ.". This is all done automatically by the software; you can ignore it while using the system. Just assign a name that's meaningful to you. Additionally, each time you are asked to provide a file name, the one most recently entered is displayed, so you can use it again if you want to. Just use the right-arrow and REPT keys to retype the name. It's just as easy to change a few characters in the file name, to

update a version number, for example, or to enter a brand new file name. The proper file name prefix is automatically attached, or searched for, depending on the context of your situation.

Another important part of user-friendliness is the manual, and this one is excellent. First, you're given a description of the system and its capabilities. Then you're told that the manual will not teach you assembly language programming; it would be unreasonable to expect that it should. So much for page one.

The rest of the manual is designed to be used as a reference. I like that. There is very little material of a tutorial nature, although examples are given where necessary. I understand that Lutus believes a good piece of software should not require a manual at all. This package comes as close to meeting this requirement as I believe an assembler could, due to the special nature of such a system. After you master the features you'll use most frequently, you will seldom refer to the manual. When you need a feature that you don't often use, it is easily found and clearly explained.

I read through the entire manual twice (only 32 pages long) before I put the disk into my drive, to get a general feel for how the system worked. I'm not sure this was absolutely necessary, but it was helpful. By doing so, I learned that a short sample program was located on page 23, with complete instructions for entering and assembling it. If you've ever used an assembler, this example is all you need to start writing your own programs.

Paul Lutus has a reputation for being somewhat adamant about his opinions on the way things should work, and authors like to point out that he has ruffled some feathers because of this. A couple of statements in the otherwise straightforward and matter-of-fact style of the manual stood out as apparent attempts by Lutus to get in a couple of editorial licks. I'll give two examples.

Lutus has expressed a distaste for BASIC, and in the manual he refers to it as a "dinosaur language." At another point, he says that Apple DOS could have included one of the features of *ALD System II's* file-handling capabilities, adding sarcastically, "...but that would have required DOS to have been written competently." At first, I interpreted such statements as arrogance, but in time I came to view them as a source of amusement. Besides, if arrogance is ever justifiable, it can only be justified by truly superior ability on the part of the writer. Mr. Lutus has amply demonstrated such ability.

I'll comment on the DOS feature without taking sides on the issue Lutus raises. The statement was made in reference to the fact that *ALD System II* reads text files via a "speed-reader," both when loading and assembling the files for the editor. This is not a unique accomplishment. Other software vendors have used similar routines to make their programs load more quickly. However, it is an important feature of *ALD System II*. When assembling long source files, much of your time is spent waiting. A lot of time can be wasted, because even a one-character change in your program requires you to go through the entire assembly process.

Although this is not intended to be a comparative review, I did assemble a fairly lengthy source file on both *ALD System II* and another assembler that reads text files in the normal DOS manner. Total assembly time (without a printout) with *ALD System II* was less than half the time required by the other assembler — impressive.

There's no denying that *ALD System II* is an excellent assembler for the Apple. I would have expected no less from one of the most capable and successful Apple programmers. The major innovation is the relatively high degree of program structure the system allows. This, along with the extensive file management features, makes the system suitable for the creation of virtually any conceivable piece of Machine Language software. The user has great flexibility and control over the utilization of memory during editing and assembly. The system bears the Lutus hallmark of compactness, ease of use, and speed. If you're considering the purchase of your first assembler, or think you might do better than one you're already using, take a good look at this one.

TRS-80

SPACE FIRE

by Bruce Forstall
and David Henderson

***Space Fire* is an arcade style game for the TRS-80® Model I/III with 16K RAM.**

The object of *Space Fire* is to shoot the maximum number of invading space ships with a limited number of steerable missiles. You start with fifteen missiles to shoot the space intruders. The base where the missiles originate is in the bottom center of the screen. On the left side of the base is a message telling you how many ships you have hit. On the right side of the base is a message telling you how many points you have scored.

The intruders' ships move from the left side of the screen to the right side, except for the second ship, which moves from right to left. There will always be four ships on the screen at one time. Each ship moves at a randomly-determined speed which ranges from 1 to 5. The number of points scored for each ship is equal to the speed, so go for the fast ones to get the best score.

When the program starts, it spends a few seconds setting up the sound routine. The screen is then drawn and the game begins.

Missiles are fired and controlled by pressing the following keys:

spacebar..... — straight up
left arrow or Z — left and up
right arrow or / — right and up

You may control the path of your missile with these keys after it has been fired. If you are *very* good, you can get all four ships on the screen with one shot.

When the game is over, a tally will be printed showing how many ships you hit, and your final score.

Variables

A$(*): First ship's design.
B$(*): Second ship's design.
C$(*): Third ship's design.
D$(*): Fourth ship's design.
D$: String of remaining missiles.
I$: "Y/N" input string.
X$: Player's missile base.
A1-A4: Positions of ships. 1 = top ship, 4 = bottom ship.
B1-B4: Speed of ships, 1 = top ship, 4 = bottom ship.
F: Missile direction flag. 1 = left and up, 2 = right and up, 3 = straight up.
F9: Flag which indicates which ship to print.
HI: High score from current run.
S: Position of player's missile.
SC: Player's score.
SH: Number of ships hit.

```
SS SS SS SS SS SS SS SS SS SS SS
SS                             SS
SS          TRS-80 BASIC       SS
SS          'SPACE FIRE'       SS
SS  AUTHORS:  BRUCE FORSTALL   SS
SS     AND  DAVID HENDERSON    SS
SS      COPYRIGHT (C) 1982     SS
SS SOFTSIDE PUBLICATIONS, INC SS
SS                             SS
SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type this program, it is also included on this month's SoftSide CV and DV.

Set up the sound routine (November, 1981 **SoftSide**)

```
10 Z=0:FORX=1TO158:READY:Z=Z+Y:NEXT:IFZ<>15204THENCLS:PRINT"DATA
 BASE ERROR IN LINES 70-170, CHECK LISTING.":PRINT:LIST70-170ELS
EY=86:X=255:POKE-1,0:IFPEEK(-1)<>0THENX=191:POKE-16358,0:IFPEEK(
-16385)<>0THENX=127
20 POKE16562,X:POKE16561,Y:CLEAR80:A1=PEEK(16561)+2:A2=PEEK(1656
2):A=A1+A2*256:Z=A-1:FORX=1TO158:Z=Z+1:Z=Z+65536*(Z>32767)
30 READY:IFY<0THENY=A1+ABS(Y):POKEZ,Y+256*(Y>255):Z=Z+1:POKEZ,A2
-(Y>255):NEXTELSEPOKEZ,Y:NEXT
40 IFPEEK(16396)=201POKE16526,A1:POKE16527,A2ELSECMD"T":DEFUSR=A
1+(A2+256*(A2>127))*256:POKE14308,0
50 IFPEEK(16807)+PEEK(16808)*256<>A+24THENA=USR(0)
60 SOUND11,11
70 DATA58,166,65,50,-164,42,167,65,34,-165,62,195,50
80 DATA166,65,33,-24,34,167,65,201,245,123,254,2,40,4,254
90 DATA16,32,79,229,213,42,230,64,126,183,32,4,35,35,35,35
100 DATA215,6,5,17,-156,26,190,32,104,19,35,16,248,43,215
110 DATA43,34,230,64,241,241,241,241,197,213,215,205,55,35
120 DATA229,205,127,10,42,33,65,34,-167,225,215,43,34,230,64
130 DATA35,205,55,35,43,229,205,127,10,42,33,65,58,-167,60
140 DATA183,87,24,4,24,48,24,44,66,62,1,211,255,16,252,66,62
150 DATA2,211,255,16,252,58,64,56,230,4,32,7,124,181,40,3,43
160 DATA24,228,175,50,154,64,225,209,193,215,195,30,29,83,79
170 DATA85,78,68,209,225,241
```

Pack the strings to be used in the program.

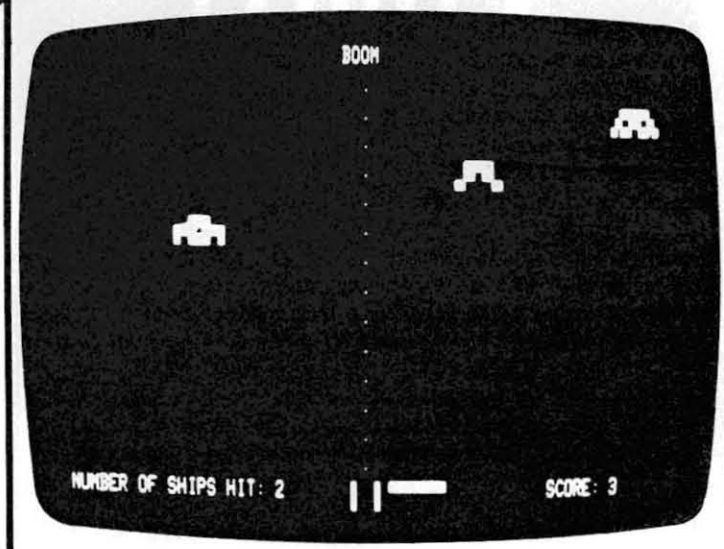
```
180 X$=CHR$(149)+" "+CHR$(170)
190 A$(0)=CHR$(166)+CHR$(183)+CHR$(183)+CHR$(183)+CHR$(132)
200 A$(1)=CHR$(174)+CHR$(187)+CHR$(187)+CHR$(187)+CHR$(132)
210 B$(0)=CHR$(152)+CHR$(179)+CHR$(143)+CHR$(179)+CHR$(164)
220 B$(1)=CHR$(168)+CHR$(163)+CHR$(143)+CHR$(147)+CHR$(148)
230 C$(0)=CHR$(176)+CHR$(143)+CHR$(131)+CHR$(143)+CHR$(176)
240 C$(1)=CHR$(140)+CHR$(179)+CHR$(131)+CHR$(179)+CHR$(140)
250 D$(0)=CHR$(156)+CHR$(174)+CHR$(179)+CHR$(157)+CHR$(172)
260 D$(1)=CHR$(141)+CHR$(174)+CHR$(179)+CHR$(157)+CHR$(142)
```

Initialize program variables.

```
270 CLS:D$=STRING$(15,131):GOSUB540:GOSUB550:GOSUB560:GOSUB570:P
RINT@993,D$;
280 F=0:S=990:F9=1
```

Start of main loop.

```
300 PRINT@0,;:SOUND100,5
```



Check for keyboard input.

```
310 IFPEEK(14656)=32ORPEEK(15119)=4THENF=1
320 IFPEEK(14656)=64ORPEEK(14880)=128THENF=2
330 IFPEEK(14912)=128THENF=3
```

Print the trail of player's missile.

```
340 PRINT@S,".";
```

Move player's missile according to input.

```
350 IFF=1THENS=S-67ELSEIFF=2THENS=S-61ELSEIFF=3THENS=S-64
```

Print missile base.

```
360 PRINT@989,X$;
```

If missile goes off the screen, reset its initial position and subtract one from missiles remaining.

```
370 IFS<0THENCLS:F=0:S=990:D$=LEFT$(D$,LEN(D$)-1):IFLEN(D$)=0THE
N580  ELSEPRINT@993,D$;
```

Print player's missile and score. Beep.

```
380 PRINT@S,CHR$(131);:PRINT@1010,"SCORE:";SC;:IFS<>990THENPRINT
@0,;:SOUND11,10
390 PRINT@960,"NUMBER OF SHIPS HIT:";SH;
```

Toggle between ship designs based on value of F9.

```
400 F9=1-F9
```

Test to see if a ship is hit. If it is, print "BOOM", beep, add the score, and reset ship's position.

```
410 IFS>A1-3ANDS<A1+5THENPRINT@A1,"BOOM ";:PRINT@0,;:SOUND255,75
:PRINT@A1,"     ";:SC=SC+B1:SH=SH+1:GOSUB540
```

TRS-80®

```
420 IFS>A2-3ANDS<A2+5THENPRINT@A2,"BOOM ";:PRINT@0,;:SOUND255,75
:PRINT@A2,"     ";:SC=SC+B2:SH=SH+1:GOSUB550
430 IFS>A3-3ANDS<A3+5THENPRINT@A3,"BOOM ";:PRINT@0,;:SOUND255,75
:PRINT@A3,"     ";:SC=SC+B3:SH=SH+1:GOSUB560
440 IFS>A4-3ANDS<A4+5THENPRINT@A4,"BOOM ";:PRINT@0,;:SOUND255,75
:PRINT@A4,"     ";:SC=SC+B4:SH=SH+1:GOSUB570
```

Move ships and print them.

```
450 PRINT@A1,"     ";:A1=A1+B1:PRINT@A1,A$(F9);
460 PRINT@A2,"     ";:A2=A2-B2:PRINT@A2,B$(F9);
470 PRINT@A3,"     ";:A3=A3+B3:PRINT@A3,C$(F9);
480 PRINT@A4,"     ";:A4=A4+B4:PRINT@A4,D$(F9);
```

Test to see if a ship has reached the edge of the screen.

```
490 IFA1>62THENPRINT@A1,"     ";:GOSUB540
500 IFA2<128THENPRINT@A2,"     ";:GOSUB550
510 IFA3>318THENPRINT@A3,"     ";:GOSUB560
520 IFA4>446THENPRINT@A4,"     ";:GOSUB570
```

Return to start of main loop.

```
530 GOTO300
```

Set the initial position and speed of ships.

```
540 A1=0:B1=RND(5):RETURN
550 A2=191:B2=RND(5):RETURN
560 A3=256:B3=RND(5):RETURN
570 A4=384:B4=RND(5):RETURN
```

Print end of game message.

```
580 PRINT@472,"OUT OF BULLETS!!";
590 IFSC>HITHENHI=SC
600 PRINT@540,"SCORE:";SC;
610 PRINT@601,"HIGH SCORE:";HI;
620 PRINT@665,"NUMBER HIT:";SH
630 PRINT@920,"PLAY AGAIN? (Y/N)"
640 I$=INKEY$:IFI$<>"Y"ANDI$<>"N"THEN640
650 IFI$="Y"THENSC=0:SH=0:GOTO270ELSEEND
```

TRS-80® SWAT TABLE FOR:
SPACE FIRE

LINES	SWAT CODE	LENGTH
10 - 70	DL	525
80 - 160	VV	512
170 - 280	LH	479
300 - 410	WA	439
420 - 510	JQ	513
520 - 630	TL	277
640 - 650	WZ	60

Fliptag

By Thomas G. Haulin III

***Fliptag* is a chase/maze game program for a 16K RAM TRS-80® Model I or III.**

Fliptag is a graphics game based on a popular children's game. Two players start off on opposite sides of a giant maze. The player with the arrows next to his score is the "chaser;" his aim is to "tag" the other player by running into him. The "chasee" attempts to maintain his distance by adroitly maneuvering through the maze. To spice things up a little, the computer randomly switches the players' roles: the arrows at the top of the screen switch sides, the bottom of the screen flashes a warning, and suddenly "chaser" becomes "chasee!"

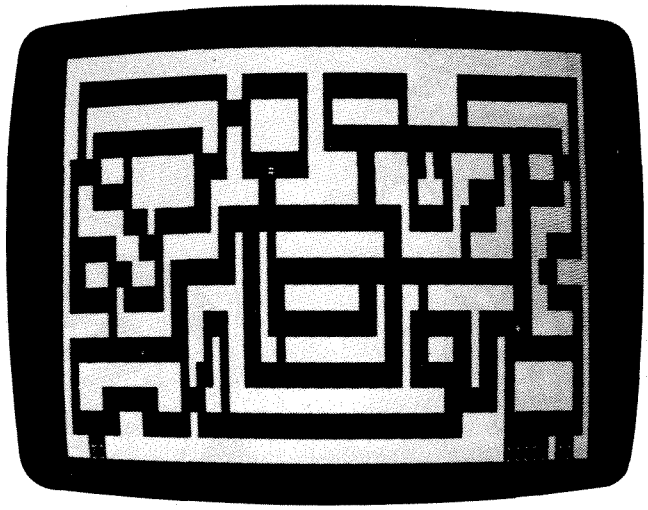
Each "tag" gives the current "chaser" one point. Ten points are required to win the game. Players can move only up and down, not diagonally. The player on the left uses the up and down arrow keys to move vertically, "W" to move left, and "E" to move right. The player on the right uses the right and left arrows to move horizontally, "P" to move up, and ";" to move down.

For extra speed, a USR function is used to determine which keys are being pressed. If the function's argument is 0, it returns a value for the left player's keys; if 1, the value is for the right player's keys. The value returned for "up" is -64, "down" is 64, "left" is -1, and "right" is 1. If none of those keys are pressed, a 0 is returned. The reason for this arrangement may not be readily apparent, but the purpose is simple enough: the value returned is the value that should be added to the current screen position of the player token to move it in the correct direction. That is, to move a player "up," you add -64 to his screen position.

Try out the game; you'll like it!

Variables

A$(0-1): Pointer arrows for each player.
A(0-1): Where to put A$(0-1) on the screen.
B: 32 (ASCII blank).
C: "Chaser" player number.
D: Delay loop.
F: 15360 (memory location of first screen position.)
I(0-1): Initial player locations.
K: Contents of location to which player is trying to move.
L(0-1): Player locations.
L: General purpose loops.
M: Player's move.
N: Number of moves done.
P(0-1): Points.
P: Player to be checked for move.
R: 191 (graphics rectangle).
S(0-1): Where to print scores.
S$: Two-digit score.
S: "Moves until" switch.
SL: Length of blank string to put on screen.
SP: Screen position of blank string.
T$(0-1): Player tokens.
U,UD,UU: USR routine set-up.
W: 1.
WP: Winning number of points.
Z: 0.



```
SS SS SS SS SS SS SS SS SS SS SS SS
SS                                SS
SS          TRS80 BASIC           SS
SS           'FLIPTAG'            SS
SS AUTHOR: THOMAS G. HANLIN III   SS
SS      COPYRIGHT (C) 1982        SS
SS  SOFTSIDE PUBLICATIONS, INC.   SS
SS                                SS
SS SS SS SS SS SS SS SS SS SS SS SS
```

If you don't wish to type this program, it is also included on this month's SoftSide CV and DV.

Initializer.

```
10 CLEAR1E3:RANDOM:DEFINTA-Z:Z=0:W=1:R=191:DIMA(W),A$(W),I(W),L(
W),P(W),S(W),T$(W):GOSUB90:T$(Z)="▮":T$(W)="▮":WP=10:I(Z)=65:I(W
)=126:L(Z)=I(Z):L(W)=I(W):A(Z)=5:A(W)=53:A$(Z)=STRING$(5,60):A$(
W)=STRING$(5,62):GOSUB80:PRINT@A(C),A$(C);
```

Check for key press.

```
20 IFPEEK(14463)=ZTHEN20
```

Translate keypress into move.

```
30 M=USR(P):IFM=ZTHEN50ELSEIFN>STHENFORL=WTO3:PRINT@960,STRING$(
63,B);:PRINT@960,STRING$(63,R);:NEXT:PRINT@A(C),STRING$(5,R);:C=
W-C:N=Z:S=RND(140)+10:PRINT@A(C),A$(C);:GOTO20
```

Check results of move.

```
40 K=PEEK(F+L(P)+M):IFK=BTHENPRINT@L(P)," ";:N=N+W:L(P)=L(P)+M:P
RINT@L(P),T$(P);ELSEIFK=42ORK=35THEN60
```

Next player.

```
50 P=W-P:GOTO30
```

Score, increment points and check for win.

```
60 P(C)=P(C)+W:S$=RIGHT$(STR$(P(C)),2):IFLEFT$(S$,W)=" "THENS$="
0"+RIGHT$(S$,W)
70 PRINT@S(C),S$;:IFP(C)=WPTHENFORL=ZTOWSTEPZ:PRINT@L(C)," ";:PR
INT@A(C),"▮WIN▮";:FORD=ZTO22:NEXT:PRINT@L(C),T$(C);:PRINT@A(C),S
TRING$(5,R);:IFPEEK(14400)=WTHENRUNELSENEXTELSEN=Z:GOSUB80:PRINT
@A(C),A$(C);:GOTO20
```

Switch players' roles.

```
80 PRINT@A(C),STRING$(5,R);:FORL=ZTOW:PRINT@L(L)," ";:L(L)=I(L):
PRINT@L(L),T$(L);:NEXT:C=W-C:M=Z:S=RND(140)+10:RETURN
```

Set up screen.

```
90 CLS:C=RND(2)-1:F=15360:B=32:S(Z)=2:S(W)=59:FORL=WTO4:PRINTSTR
ING$(255,R);:NEXT:PRINTSTRING$(3,R);:POKE16383,R:FORL=WTO111:REA
DSP,SL:PRINT@SP,STRING$(SL,B);:NEXT:PRINT@S(Z),"00";:PRINT@S(W),
"00";:IFSL-19THENPRINT"▮▮▮ DATA ERROR ▮▮▮":STOP
```

USR routine.

```
100 U$="01234567890123456789012345678901234567890123456789012345
67890123456789012345678901234567890123456789"
110 U=VARPTR(U$):UU=PEEK(U+W)+PEEK(U+2)▮256:FORL=UUTOUU+92:READU
D:POKEL,UD:NEXT:IFUD-218THENPRINT"▮▮▮ DATA ERROR ▮▮▮":STOPELSEIF
PEEK(16396)=201THENPOKE16526,PEEK(U+W):POKE16527,PEEK(U+2):RETUR
NELSECMD"T":DEFUSR0=UU:POKE14308,Z:RETURN
```

Screen data.

```
120 DATA65,9,79,32,112,6,121,6,129,1,137,1,139,6,171,1,174,3,180
,7,190,1,193,1,201,1,203,2,206,7,214,19,235,1,237,2,240,1,254,1,
257,9,267,2,270,2,275,2,278,2,292,1,295,2,299,1,301,1,304,15,321
,1,327,6,334,7,342,2,345,13,359,2,363,3,368,2,377,1
130 DATA385,4,391,2,403,1,406,2,409,2,420,2,423,2,432,2,435,5,44
1,6,452,2,455,31,487,2,490,8,499,1,503,3,509,2,513,4,519,2,526,1
,534,2,549,1,551,2,554,2,563,3,569,6,577,1,583,2,587,4,592,5,598
,22,621,7,629,3,638,1,641,1,643,6,650,4,656,1,660,1
140 DATA665,2,677,2,685,2,695,5,702,1,705,3,711,2,714,1,720,2,72
3,2,729,2,737,8,747,4,759,1,763,4,769,1,771,28,801,1,808,1,811,1
,814,14,829,1,833,1,846,1,853,1,862,1,865,1,872,4,893,1,897,14,9
17,10,929,8,939,19
```

USR routine data.

```
150 DATA205,127,10,125,183,32,52,58,64,56,230,8,40,5,33,192,255,
24,37,58,64,56,230,16,40,6,33,64,1,37,24,24,58,4,56,230,128,40,5
,33,255,255,24,12,58,1,56,230,32,33,255,255,35,40,1,35,195,154,1
0,58,64,56,230,32,32,229,58,4,56,230,1,32,197
160 DATA58,32,56,230,8,32,202,58,64,56,230,64,33,255,255,35,40,2
21,24,218
```

TRS-80® SWAT TABLE FOR: **FLIPTAG**

LINES	SWAT CODE	LENGTH
10 - 70	VU	634
80 - 110	NP	526
120 - 140	HZ	690
150 - 160	HJ	306

SoftSide DV, the magazine of the future, is here!

If your computer could pick a magazine, wouldn't it prefer one in its own language? Now there's one available. **SoftSide DV** is an enhancement of the **SoftSide** you have in your hands.

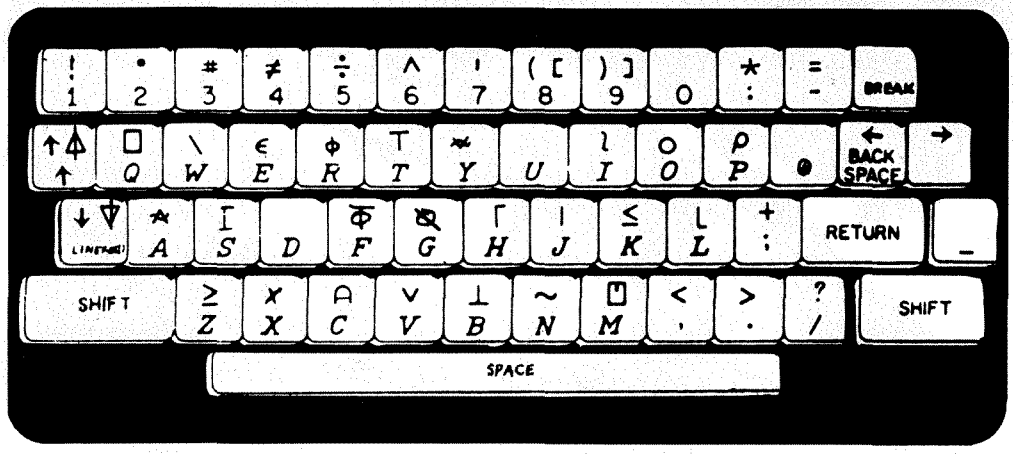
SoftSide DV contains not only the complete programs listed in every month's issue of **SoftSide,** but additional programs of every conceivable type, as well — multiple and Machine Language programs, modified languages, ongoing modular programs and software so extensive, it would take an entire issue of **SoftSide** just to print the code. Only the documentation for these programs will appear in **SoftSide** Magazine, **NOT** the code.

Feel as though you're missing something? You are! But, you needn't miss out on another issue. **SoftSide DV** is now available for Apple™, ATARI® and the TRS-80® . The cost to you — $125 for 12 magazines and 12 disks, packed with some of the best software available, all delivered to your home in the next year. For orders outside the USA, please add $36. For your convenience, we offer an installment payment plan for VISA and MasterCard holders: You pay only $32.50 per month for four months (a total of $130, which includes a $5 billing charge). Please use the special DV-CV bind-in card in this issue to order.

Computerists are offered the rare opportunity of marching into a new frontier. Advance to the front of the parade by subscribing to **SoftSide DV**, the magazine of the future, available today!

APL80

by Phelps Gates



Symbol APL	Symbol APL80	Monadic / Dyadic	Name
|	j	M	Absolute Value
+	+	D	Add
∧	&	D	AND
←	←		Assign
→	→		Branch
,	,	D	Catenate
⌈	h	M	Ceiling
	#		Chr$ / ASC
○	o	D	Circular
!	!	D	Combinatorial
⍝	c		Comment
/	/	D	Compress
?	?	D	Deal
⊥	b	D	Decode
÷	%	D	Divide
↓	↓	D	Drop
⊤	t	D	Encode
=	=	D	Equal
\	w	D	Expand
*	*	M	Exponential
!	!	M	Factorial
⌊	l	M	Floor
⍒	↓	M	Grade Down
⍋	↑	M	Grade Up
>	>	D	Greater Than
≥	z	D	Greater or Equal
⍳	i	M	Index Generator
[ ]	();	D	Indexing
⍸	I	D	Index of
.	.	D	Inner Product
:	:		Label
<	<	D	Less Than
≤	k	D	Less Than or Equal
⍟	@	D	Log to a Base
⌈	h	D	Maximum
∈	e	D	Membership
⌊	l	D	Minimum
×	x	D	Multiple
⍲	a	D	NAND
⍟	@	M	Natural Log Negative
⍱	y	D	NOR
~	n	M	Not
≠	$	D	Not equal
∨	v	D	OR
∘.	"	D	Outer Product
⍤	q		Quad
	m		Quote Quad
?	?	M	Random
,	,	M	Ravel
⌹	%	M	Reciprocal
/	/	M	Reduction
⍴	p	D	Reshape
⌽	j	D	Residue
⊖	r	M	Reversal
⍉	r	D	Rotation
*	p	M	Shape
×	x	M	Sign
⍳	s	M	System
−	−	D	Subtract
↑	↑	D	Take
⍉	f	M	Transposition
⍉	g	M	Transposition
⍕	g	D	Format

***APL80* is a large subset of the APL language adapted for the TRS-80® Model I/III with 32K RAM and disk drive. It is included as the bonus program on issue 36 TRS-80 DV. See the Bind-in Card elsewhere in this issue to order this month's disk.**

Editor's Note: These instructions assume that you know APL or have access to a manual which describes it. Only the differences and limitations of *APL80* are described here.

On the Model I, *APL80* uses arrow keys to perform certain functions. The Model III uses the following characters in place of arrows.

↑ [←]
↓ \ → ^

Another note on the Model III version of *APL80*: The shifted @ key is used as a function, while the unshifted @ is used to exit the insert/replace submodes. This is the opposite of how the Model I *APL80* operates.

Loading *APL80*

To load *APL80* from disk, just type *APL80* from the DOS READY mode.

Character Set

APL80 redefines certain keys to get characters not normally available. With older Model I's, you can type all four arrows by pressing the SHIFT key, together with the appropriate arrow. If you have a Model III or late revision Model I, hold the shift, down arrow, and Z to get the down arrow. Also, you can type an underscore by pressing the CLEAR key.

The *APL80* keyboard diagram shows the various APL characters that can be simulated using the TRS-80 keyboard. Most are simulated by holding the SHIFT key down and then depressing another key. Of course, a shifted 4 is still displayed on the screen as a dollar sign, for example but it stands for "not-equals" in *APL-80.*

Shifted letters are used to represent many *APL80* functions. These instructions will use lower case letters to represent shifted letters. Since an unmodified TRS-80 Model I doesn't have the ability to display lower-case letters, *APL80* automatically prints a graphics dot at the left of a shifted letter to distinguish it from an unshifted letter.

Note that round parentheses are used for indexing (as in BASIC). Also, arrows are used for Grade Up/Down. Since these are monadic and Take/Drop are dyadic, no ambiguity arises.

Format

The f operator establishes a format. It must come first in a statement, followed by a vector with an even number of elements, specifying the field size and number of decimals for all following numerical output. Note the difference in notation:

APL: 6 3 ⍕ OUTPUT
APL80: f 6 3
OUTPUT

The format continues in effect until 1) a new f statement, 2) an error or system command, or 3) any character output.

The d operator chooses the dimension along which a matrix manipulation is performed (square brackets in APL).

APL: +/[1] MATRIX

APL80: +/1dMATRIX ("plus-reduction on the first dimension")

This applies to reduction, compression, expansion, scan, reversal, and rotation.

Getting Started

The disk contains five workspaces which explain some of the ways you can use *APL80.* They are: LESSON1/APL, LESSON2/APL ... LESSON5/APL. To run the first lesson, just type)LOAD LESSON1/APL. The disk also contains a workspace called CUSTOM/APL, which provides a number of modifications to *APL80.*)LOAD CUSTOM/APL will load this workspace, which is self-documenting.

System Commands

The following commands correspond to APL, with minor differences:
)OFF)CLEAR)FNS)VARS)SI)ERASE)RESET

In the variable list, any active local or dummy variables are marked with an asterisk.

)ERASE name — will erase a variable or function.

)CLEAR — erases everything...you start fresh with an empty workspace.

)SI ("State Indicator") — lists any functions which have been stopped by an error or BREAK with the number of the line which was about to execute.

)RESET — clears the SI. Numerous interrupted functions use up memory and slow down execution, and will eventually cause a depth error.

)SAVE filespec — saves the current workspace. (Its previous contents are lost.)

)LOAD filespec — loads a workspace.

)KILL filespec — will delete a file from the disk. Any file, not just APL80-created ones, can be killed.

)COPY filespec — merges the workspace named into the current workspace. Duplicated objects replace the corresponding items in the current workspace; individual objects cannot be copied.

)DOS — returns to DOS READY. You can then execute DOS commands (such as DIR or KILL). To return to *APL80* with the workspace intact, type RETURN.

)AUTO expression — puts a latent expression into the workspace. If the workspace is then saved and loaded, this expression will automatically execute, unless you override this feature by holding down the space bar during loading.

)EXEC — executes the latent expression, if any, in the current workspace.

)TRON — turns on trace. Function name and line number are printed just before each statement is executed.

)TROFF — turns off trace.

)PS ("print single") — makes *APL80* print numbers with 6-digit precision. Only output is affected; calculations and variables continue to have 15-digit precision. This will considerably speed up output, expecially if format is used.

)PD ("print double") — restores 15-digit precision in output.

)RAM — enables the PEEK, POKE, and CALL functions. (See below under "RAM interaction".)

The READ, WRITE, RESTORE, UPDATE and QUERY commands are described below, under "File Handling."

Function Definition

The syntax of functions in *APL80* is identical with APL, but the mechanics of function definition differ. Some functions are like programs — you just type their name and they execute. These are called "Niladic." Other functions are like operators, and require arguments. A "Monadic" function has one argument; a "Dyadic" function has two. To create a function, use the)DEF command, followed by the header of the function (line 0). Examples:

```
)DEF NILADIC
)DEF RES←LEFTARG NAME
RIGHTARG; LOC1; LOC2
```

The function name must not be in use already, or a DEFN ERROR results.

To change or display a function which already exists, use)EDIT, followed by the function name.

```
)EDIT NILADIC
)EDIT RES
```

The)DEF and)EDIT commands put you into definition mode, and *APL80* will prompt you for the first unused line of the function (line 1 for a new function). Type the statement, press ENTER, and it will ask for the next line.

To leave definition mode, press BREAK.

To display the function, type)? (and press ENTER).

To type the function on a printer, type)H.

You can avoid screen scrolling in long functions by typing)P followed by a line number; *APL80* will display 14 lines, starting with that number. Example:)P21 prints lines 21-34.

To replace a line, type) followed by the line number which you want to replace. For example, typing)2 will replace line 2.

To insert a line, type)I followed by the line number where you wish to insert a new line. For example, typing)I2 will insert a new line between lines 1 and 2.

To delete a line, type) and the line number. Then, when *APL80* asks you for a replacement, leave definition mode (with BREAK) or ask for a display with)? or)P.

Whenever you insert or delete a line, *APL80* immediately renumbers the lines in sequence. If you ever get confused about the numbers, just type)? for a display of the function with the current line numbers.

As in APL, errors are normally not detected until a function is executed. Exception: syntax errors involving "-" (such as 3-3) or single number domain errors (1E99) will be detected during function definition, and will cause an exit from definition mode.

You may also edit a function line by typing)E followed by the number of the line which you want to edit. *APL80* will display the line at the bottom of the screen, with an edit pointer above it. Note that in edit mode, the graphics dots used to represent lower case letters on unmodified Model I's are displayed below the letters, rather than at the left.

To move the edit pointer, press the left or right arrows. To insert text at the pointer, press I. To delete text at the pointer, press D (hold down to repeat). To replace text at the pointer, press R. To cancel changes and start fresh, press A. To complete the edit session and actually change the function line, press ENTER. Pressing I or R puts you into insert/replace submode. To exit from this submode and return to edit mode, press shifted @.

If a line exceeds 64 characters, a length error results.

Since *APL80* replaces double apostrophes ("''") in function lines with single apostrophes, you must replace the second one whenever you edit the line they are contained in with)E.

The header cannot be edited.

Limitations

Transcendental functions, including *, are accurate only to six digits. Matrix inversion, lamination, and the diagonal case of dyadic transposition are not implemented. Encode and decode are limited to vector arguments. The arguments of ! must be integers. Hyperbolic functions are not implemented. Multiple specifications (X←Y←9) must be split into two statements. This also applies to implied multiple specifications, such as % x←9. A quad can't be typed in response to another quad. (Quote-quad is OK.) No more than 32 functions can be defined in a workspace, and a function can't have more than 25 lines.

The # and s Operators

The operator # converts numeric arguments to characters and vice versa, like CHR$ and ASC in BASIC. Try, for example, #28 31 150 or #'ABC'. In addition to the Level II control characters (except 14 and 15), *APL80* recognizes:

#3	begin 16-zone print mode
#5	begin printing hard copy
#6	randomize workspace (like RANDOM in BASIC)
#7	begin 8-zone print mode
#9	stop printing hard copy
#11	restart random link (?100 will give 77)
#15	begin 4-zone print mode
#16	stop real-time clock
#17	start clock
#18	reset clock to zero
#255	cancel zone printing
#-1 to #-1023	are equivalent to PRINT @ 1 to PRINT @ 1023.

Once begun, zone printing continues in effect for all numeric output until cancelled by #255. If you call for hard copy with #5 with no printer connected, nothing will happen.

The monadic s operator has these uses:

s0: the time since sign-on (in 25ms increments on the Model I, 33 1/3ms increments on the Model III).

s1: the "read pointer:" the number of the record which will be read by the next READ command (see File Handling).

s2: the "update pointer:" next record to be updated.

s3: number of records in the most recently accessed file.

s4: "line counter:" the number of the line currently executing. To continue execution of a function after pressing BREAK, type s4.

s5: number of free bytes in the workspace.

Miscellaneous

Function and variable names may contain "." and "-": THIS-IS-A-NAME AND.SO.IS.THIS. Watch out: M minus N is "M_N". "M-N" is a reference to a variable named "M-N".

All comparison operators may be used with character arguments (>< $k2 z); also Grade Up/Down, but not reduction.

?0 yields a random number between 0 and 1. ?10 yields a random integer from 1 to 10.

ENTER always terminates a quoted string. To include carriage returns in a string, use line-feed (unshifted down arrow).

Pressing BREAK will terminate function execution, even if the function is awaiting input from a quad or quote-quad.

You can temporarily freeze output by holding the space bar down.

The operands of comparison operators must be of the same type ('A'=2 gives a domain error).

Statements in function listings and error messages may differ slightly from what you typed. Usually the difference is just in spacing, buy try, for example, .00001%0.

The semicolon used to index multi-dimensional arrays does not function as a strong delimiter. This means that parentheses must be used around compound expressions used as indices: MAT((1+2);2).

Indices need not be integers: *APL80* uses the integer value of the indices.

When syntax errors occur, *APL80* sometimes detects them at a different point than APL, and behaves differently. Example: If FUNC does not return a value, the statement FUNC+FUNC will result in an error before the function is executed. *APL80* also differs from APL in its treatment of duplicate function/label/local variable names.

RAM Interaction Functions

APL80 includes PEEK, POKE, and CALL functions. To avoid unfortunate accidents, these functions must first be enabled with the)RAM command. Monadic u is PEEK: the right argument is a RAM location (0 to 65535). The function returns the contents of that location. Dyadic u is POKE: the left argument is put into the location given by the right argument (example: 65u16000 puts an A onto the screen). It returns a value: the contents of the location after the POKE (the same as the ROM argument, unless you're POKEing ROM!). Dyadic # is CALL: the right argument is put in the A register and the address given by the left argument is CALLed. It returns the value of the A register after the call.

File Handling

You can move the contents of variables to and from files using the)WRITE,)READ,)RESTORE,)UPDATE, and)QUERY commands. You can include these commands in a function by spacing once before typing the command. (This prevents the initial right parenthesis from being interpreted as a function definition command.)

)WRITE VARIABLE filespec

The contents of VARIABLE are added to the end of the file. If the file does not exist, it is created, and the contents of VARIABLE become the first record in the file.

)READ VARIABLE filespec.

One record (as written by one WRITE command) is read from the file into the variable. Which record gets read is determined by the current value of the "read pointer" (s1). When *APL80* first loads, the read pointer is set to 1, and each READ command advances it by 1.

)RESTORE

Resets the read pointer to 1. The next READ command will then read the first record in its file.

)QUERY filespec

Tells you how many records are in a file, and how big each one is.

A record containing numeric data requires 4+2d+8e bytes, where d is the number of dimensions in the stored variable and e is the total number of elements in it. Character data requires 4+2d+e bytes.

Since each WRITE or READ command requires considerable overhead, it is more economical to write large arrays

to disk, rather than many small variables. Files are limited to 127 records, with a total length of less than 65535 bytes.

If the "read pointer" is beyond the last record in a file, an empty vector will be read (you can use this as a test for end of file). Notice that RESTORE doesn't affect writing; the)WRITE command always adds records at the end of the file.

You can change the value of the read pointer (to access records at random) by using s as a dyadic operator with 1 as the second argument. For example: 3s1 followed by READ will read the third record. Dyadic s returns the previous value of the pointer — if you don't want it printed, just assign it to a variable: DUMMY←3s1.

)UPDATE VARIABLE filespec

Writes the contents of VARIABLE to the file, not at the end of the file, but as a replacement for the record pointed to by the "update pointer" (s2). The update pointer is advanced by one. The new record must be the same size as the old one. You can change the value of the update pointer with dyadic s, with 2 as the right argument. For example: 4s2 prepares to update the fourth record. RESTORE also resets the update pointer: it's exactly equivalent to 1s1 2. If the update pointer is beyond the end of the file, or if the file doesn't exist, UPDATE is equivalent to WRITE.

The following example shows how file handling works:

```
A←3
B←1 2 3
C←'HELLO'
)WRITE A FILE/DAT          (file created, first object
                            is 3)
)WRITE B FILE/DAT
)WRITE C FILE/DAT
)RESTORE
)READ X FILE/DAT           (X contains 3)
3s1                        prepare to read third record
)READ X FILE/DAT           (X now contains 'HELLO')
)READ X FILE/DAT           (X contains an empty
                            vector-end of file)
UP←4 5 6
2s2                        (prepare to update second
                            record)
)UPDATE UP FILE/DAT
2s1                        (now read updated record)
)READ Y FILE/DAT           (Y contains 4 5 6)
```

Using APL

The simplest way to use *APL80* is to type an expression, followed by ENTER. *APL80* will evaluate it and print the result. For example, try:

```
2 + 2
2 % 3
3 x 4 (use the shifted X key)
7 _ 5 (press CLEAR for underscore)
2 * 10
```

APL80 will also operate on groups of several numbers, called "vectors." For example, try:

```
2 3 4 + 10 11 12
5 % 1 2 3
2 * 0 1 2 3 4 5
```

Use a space to separate the members of a vector. Note what happens if you type 2 3 + 1 2 3 4. The vectors must be the same length, or one must be a single number.

One slightly tricky thing about *APL80* is the way it handles minus signs. To subtract numbers, use the underscore sign (press CLEAR key): 5_3, for example. Use an ordinary minus sign to indicate negative numbers:

```
-5 + 10 20 30
10 _ -30
3 + -10.
```

This takes a little getting used to.

Like BASIC, *APL80* can store data in variables. Variable names may be any length (up to the length of the input line). To assign a value to a variable, use the left-arrow (shifted←key). Try, for example.

```
VAR←1 2 3 4 5
VAR*2
VAR_VAR
```

A value error will result if you reference a variable name which has not been assigned a value.

Operators

We've already met + /_ (Watch out for those minus signs!), x (shifted x) and * (exponentiation). Addition, subtraction, etc., are called "Dyadic" operations: they involve two numbers, one before and one after the operator. *APL80* also has "Monadic" operators, which use only one argument, after the operator: for example, !5 equals 120 (factorial 5). % can also be monadic (%5 equals .2, the reciprocal of 5). + and _ may also be used as monadic functions (identity and negative) and x (sign). Monadic * is the exponential function (e to the nth).

Order

In APL, there is no hierarchy of operations — it doesn't do multiplication before addition, for example. Expressions are evaluated from the right. For example, 2x3+10 equals 26 — two times (three plus 10). This takes a little getting used to, but it beats worrying about whether "and" is performed before "or," etc. In APL, you only evaluate from the right: 2x3+10_9 is 2x(3+(10_9)). If you want, you can force any order by using parentheses — (2x3)+10 is 16.

Index-Gen

The monadic operator i (Index Generation) produces a vector of numbers. i3 is 1 2 3, for example. i0 is a vector of 0 numbers, called an "empty" vector.

Reshape

The reshape operator (p) constructs an array whose dimensions are given by the first argument, and whose elements are taken from the second argument. The result will have as many dimensions (up to 64) as there are numbers in the first argument. If the second argument isn't long enough, *APL80* goes back to the beginning and starts over.

Reverse

Reverse (r) is an easy one — note that the reversal occurs on the last dimension (the columns).

Product

The notation A".xB ("outer product of A and B") indicates an array generated by multiplying every element of A by every element of B. Any of the operators described in LESSON1/APL may be used: A".+B will add each element; A".=B tests to see if they are equal, etc. The notation A+.xB ("inner product") of two vectors means "multiply each element of A by the corresponding element of B, and sum the resulting vector." Again, any of the basic operators may be used — 441 different combinations are possible.

Quads

The symbol q ("Quad") corresponds to BASIC "INPUT" — try typing: 20+q. You don't have to type a number in response — you can type an expression (2+2), a variable name (must exist), a function which returns a value, or a character string, in single quotes. m ("Quote-quad") is similar, except that (1) no prompt is displayed, and (2) what you type is treated as a character string.

Semicolon

You can use ";" to print several things on the same line. This is in addition to the use of ";" for multi-dimensional indexing.

Take-Drop

A↑B ("A take B") selects A elements from the vector B. If B isn't long enough, it's padded out with zeroes, or blanks for a character string. The elements are taken from the start if A is positive, from the end if A is negative. A↓B ("A drop B") is the opposite — it drops A elements from the beginning or end. Take and drop also work with multidimensional arrays. The left argument must be a vector, with one number for each dimension of the array, telling how many to take or drop from that dimension.

Encode

Encode (t) switches from one number system to another. The left argument is a vector containing as many digits as we want in the answer, the right argument is the number which we want to convert: 2 2 2 2 2 2 t 21 yields 0 1 0 1 0 1 (21 base 2). If the left argument is too short, any overflow will be lost. You can put 0 as the first element in the left vector, in which case any overflow is put into the first element of the result. The numbers of the left argument need not be the same — to convert 100 inches to yards/feet/inches, try 0 3 12 t 100.

Decode

Decode (b) works like this:

Think of the right argument as a vector of digits in a number system whose base is given by the left argument. The result is the value of the number with those digits:

16 b 7 15 15 15 gives 32767

For a mixed number system, use a vector on the left, with the same number of elements as the vector on the right. (The first may be a dummy.) How many seconds are there in 2 weeks, 3 days, 4 hours, 7 minutes and 12 seconds? 1 7 24 60 60 b 2 3 4 7 12.

Catenate

A comma adds one vector onto the end of another — this is called "Catenation." Try: 1 2 3 , 4 5 6

Index

Just as in BASIC, you can use parentheses to extract elements from an array. If an array has more than one dimension, use a semicolon to separate the dimensions: MAT(2;3). A notation like MAT(;4) means "all the items in the fourth column." You can use indexing on the left of an assignment arrow to change individual elements of any array.

Rotate

Dyadic r rotates the right argument left as many places as specified by the left argument. Try:

3 r i10

-3 r i10

Multi-dimensional arrays are rotated along their last dimension (columns).

Grade

↑ (grade up) and ↓ (grade down) are monadic functions which tell the order in which you would need to select the elements of a vector in order to sort them into ascending or descending order. They are usually used in conjunction with indexing. Try:

A←5 4 6 2 3 1

↓A

A(↑A) (sort the vector!)

Shape

Shape (p) is a monadic function. It yields a result which tells how big an array is — a vector of one number for each dimension.

p1 2 3 4, for example, yields 4.

To find the number of dimensions in an array, just use p twice: p p MAT. A slightly tricky detail — APL distinguishes between a single number, called a "Scalar," which has no dimensions (p3 yields an empty vector), and a vector of one element, which has one dimension — you can construct a vector of one element with reshape or ravel.

Ravel

The comma can be used as a monadic function to convert an array of any number of dimensions into a vector. If the argument is a scalar (0 dimensions), it is converted into a vector of one element.

Index-of

Function i can also be used as a dyadic function, to tell where the second argument occurs in the first argument. (This is called "Index-of".) 9 3 4 6 i 4 yields 3 because 4 occurs in the third position in the first argument. If the second element doesn't occur in the first, it yields a value one greater than the length of the first argument.

If the second argument occurs more than once, only the first occurrence is found.

Membership

For each element in the left argument, e checks to see if it is found in the right argument. If it is, it yields 1; if not, 0. The result has the same number of elements and dimensions as the left argument. Try:

1 e 1 2 3 1

Branching Functions

One of the aspects of APL which can be confusing at first is the way it handles branching. The right arrow (→) is equivalent to GOTO in BASIC. For example:

```
)DEF INFINITELOOP
1: 'PRESS BREAK TO STOP'
2:→1
```

Or you can use labels:

```
)DEF SQUAREROOT;X
1: GETMORE:'ENTER NUMBER'
2: X←q
3: 'THE SQUARE ROOT OF';X;
'IS';X*.5
4:→GETMORE
```

Note the use of X as a local variable in this function. This allows us to call the function without affecting any value which we may have stored in a variable called X. The shifted q (for "quad") prints a prompt and waits for input (APL □).

Note that it is preferable to use labels for branching, since the numbers of lines within a function will change when function lines are inserted or deleted, but the branch statement will not.

Conditional branching is a little trickier, since APL doesn't have operators which correspond directly to IF or THEN in BASIC (or FOR... NEXT). Of course, you don't have to use branching as much in APL as in BASIC. Since APL can operate directly on arrays, a single APL line can often do the work of a whole program in BASIC. Sometimes, however, you do have to branch conditionally. Consider the

following function, which generates Pascal's triangle of binomial coeffecients:

```
)DEF TRIANGLE
1: K←-1
2: ANOTHER: K←K+1
3: 1,(iK)!K
4: →ANOTHER
```

This will print Pascal's triangle, but it won't stop — it just keeps printing lines until the numbers get too big and you get a DOMAIN ERROR. We need to test K and loop back only if K is less than 10. To do this, we have to know the rules for branching in APL. There are three cases:

1. The line number (or label) exists in the current function. Control simply passes to that line.

2. The line number doesn't exist in the function (→0 or →999 or →-1 or just →). The function terminates: this is equivalent to RETURN in BASIC.

3. The expression to the right of the arrow is an empty vector (i0). No branch occurs; control just passes to the next line.

In the TRIANGLE function, for example, we could change line 4 to:

```
4: ANOTHER x K<10
```

Now if K is less than 10, the logical expression K<10 will have the value 1 (true), and the expression in line 4 will have the value ANOTHER x 1, which equals ANOTHER. On the other hand, if K is not less than 10, the expression K<10 has the value 0 (false), and the function branches to ANOTHER x 0, which equals 0, and so the function terminates, by rule 2.

We could also write line 4 as:

```
4: ANOTHER x i K<10
```

If K is less than 10, this branches to ANOTHER x i 1, and since i 1 equals 1, this is equivalent to →ANOTHER. If K is not less than 10, we get ANOTHER x i 0, and since multiplying an empty vector by anything still gives an empty vector, this is equivalent to →i 0. No branch occurs (rule 3), but since there aren't any more lines in the function, execution terminates anyway.

This works fine mathematically, but it doesn't make your programs any easier to read. One solution would be to define a function, to figure it out for us. For example:

```
)DEF BRANCH←LABEL IF
CONDITION
1: BRANCH←LABEL x i
CONDITION
```

Now we can write line 4 of the TRIANGLE function as

```
4: ANOTHER IF K<10
```

which makes a little more sense. The function IF computes the proper destination for the branch, depending on the value of K. Note that we don't have to call it IF; you could call the function PROVIDED.THAT or SI or whatever. It's a dyadic function, requiring both a left operand (the label) and a right operand (the condition), and it returns a value (the appropriate destination).

You can also do it backward, and define a function called UNLESS:

```
)DEF BRANCH←LABEL UNLESS
CONDITION
1: BRANCH←LABEL x i n
CONDITION
```

and rewrite the TRIANGLE function as:

```
4: ANOTHER UNLESS K>9
```

Instead of using multiplication, you can define the IF function using the compression operator:

```
1: BRANCH ←CONDITION/
LABEL
```

*Editor's note: An enhanced version of APL-80, called APL*80/PLUS, is available for the TRS-80 Model III from STSC, Inc., of Rockville, MD., for $295.00*

*APL*80/PLUS is one of a family of APL implementations offered by STSC. The interpreter is partially RAM resident, with overlays on the disk to be called in as needed. The system comes with a custom character set in ROM which provides the full APL character set.*

Some of the extensions implemented by STSC include support for parallel printer output, full terminal mode with RS-232 support, FMT report formatter, 16 digit floating point math, "HELP" system for on-line documentation of system features, STOP and TRACE debugging facilities, and native graphics support.

The system supports TRSDOS v1.3 and LDOS v5.1, and allows access to native DOS files.

Further information is available from:

STSC, Inc.
2115 East Jefferson St.
Rockville, MD 20852
(301) 984-5000

Reviewed by J. B. Harrell, III

PASCAL-80

An Excellent Alternative

from New Classics Software, 239 Fox Hill Road, Denville, NJ 07834. System requirements: TRS-80® Model I or III with 48K memory and one disk drive. Suggested retail price: $99.00/disk.

Over the several years that I have been involved in computers, I have seen the development of many computer languages to ease the programmer's burden. Languages such as FORTRAN and COBOL were developed and evolved into the early "work-horses" of the computer society. During this formative era, BASIC was originally devised as a "quick-and-easy" language for both student and teacher alike.

No language has ever approached the easy, friendly, and interactive nature of BASIC until *Pascal-80* was developed. Essentially every compiler currently on the market for the TRS-80 requires extensive disk access to edit, compile, and run a program. Repeated compilations and trial runs during program development cause the programmer to lose time just moving among the system modules to change and recompile his program.

Pascal is a fully structured language that has been developed for a wide range of computers, from the largest mainframes to micro-computers. Several versions are now available for the TRS-80, but most require the CP/M (Control Program for Microcomputers) operating system and cost hundreds of dollars. Many of these have limited features and appear to be a "force-fit" to make a 56K system fit into the TRS-80.

Pascal-80 — System Overview

The *Pascal-80* system loads entirely into memory and provides many of the features of a BASIC interpreter. It allows you to edit, compile, and run programs rapidly with a few key-strokes from the system monitor. This simple ability removes much of the frustration that you would have with any other compiler system.

The *Pascal-80* system is supplied on a single-density, 35 track disk for Model I users. This disk comes with TDOS (Tiny version of DOSPLUS) and is ready to operate when received. *Pascal-80* is compatible with the popular current operating systems written for the TRS-80. All you do is copy the file PASCAL/CMD (and any desired source code files) over to your current operating system disk.

From the system monitor, you can easily execute any of the following commands (displayed in the system menu) using a single key-stroke:

E - Enter full screen editor mode
Q - Quit and return to the Disk Operating System
K - Kill the current source file in memory

C - Compile and list code to video screen
R - Run the program, compile it if necessary
S - Save the source code to the named disk file
L - Load source code file from disk file
A - Append disk source code to in-memory text
W - Write object code to disk, compile if necessary
X - Execute object code from disk

Once the *Pascal-80* system is loaded into memory, you are left with a source program text buffer of approximately 23K bytes. This will hold about 1,100 lines of source code, assuming an average Pascal line length of 20 bytes and the text-compression techniques used in *Pascal-80*. Note that source programs are not limited by the text buffer size but by the object p-code work space. This is because the source program may be developed in several segments and compiled from the disk by using the INCLUDE facility of the compiler.

Use of the in-memory text buffer accounts for the high compilation rate for *Pascal-80*. Using a standard TRS-80 Model I (1.77 MHz clock), I clocked an average compilation rate of 1,000 lines per minute listing to the video screen and 1,500 lines per minute with screen listing secured. Compiling from the disk using the INCLUDE option slows the compiler down considerably, as it is limited by the disk operating system data transfer rate.

to BASIC

The System Editor

The text editor used by the *Pascal-80* system is a full-screen text editor invoked from the monitor. Fifteen lines of source code are displayed on the video screen at one time, and the non-destructive cursor is moved on the screen by the arrow keys. Unlike many text editors, each line is separate and may be no more than 64 characters long. Text lines and corrections to them are permanently stored in the text workspace only after ENTER has been pressed. One very nice feature of the editor is the *auto-tab* function which allows easy formatting of Pascal source code with proper indentation.

The editor commands are invoked by the control key sequence of SHIFT DOWN-ARROW plus the first letter of the command. The editor command menu is displayed on the bottom line of the video screen by pressing BREAK. The following editor commands are allowed:

- Block move source code
- Cancel all changes in the current line
- Delete the character at the current cursor location
- Erase the current line from the workspace
- Format control toggle for automatic tab feature
- Line insert one blank line at the current cursor
- Next page (15 lines) of source code
- Open the line one space at the current cursor location
- Previous page of source code
- Quit and return to monitor menu
- Top of source code text workspace
- Write source code workspace to the line printer

The System Compiler

The *Pascal-80* compiler implements a subset of Standard Pascal with several extensions. It is a one-pass compiler which generates Pascal pseudo code (p-code). Since the p-code must be executed by an interpreter, *Pascal-80* will run somewhat slower than a system which generates the native machine code for the processor.

The reference manual that accompanies the disk is a description of the limitations and extensions of the implementation of Pascal sold by New Classics Software. It is not a tutorial for the Pascal language nor does it provide any extensive programming examples. If you desire additional knowledge, purchase a tutorial text such as *Programming in PASCAL* by Peter Grogono (Revised Edition, Reading, MA: Addison-Wesley, 1980). This book provides an excellent description of the standard implementation of Pascal and uses in-depth examples where applicable.

Many extensions have been added to the *Pascal-80* compiler. The most important additions are the ability to "include" source text from a disk file in the current program being compiled, plus graphics and randomizing functions. Judicious use of the powerful INCLUDE facility allows you to create a source program much larger than the capability of the text buffer and then compile it from the disk. This allows the maximum available space for the generated p-code. The complete list of extensions is presented below:

- Arrays of characters may be printed using a single WRITE statement. This somewhat makes up for the absence of true STRINGs.
- The procedures READ and WRITE may be used with non-text files in place of the missing procedures GET and PUT (see limitations listed below).

When using string constants in an assignment statement and in logical comparisons with character arrays, the constant on the right of the assignment operator or the logical operator may be shorter than the item on the left. It will be padded with blanks as necessary to make the length of the two strings equal. If the variable STRING is defined to be of type ARRAY(.1..10.) OF CHAR then STRING:= "NAME" is a valid assignment statement and STRING "NA" is a valid logical comparison and will evaluate to TRUE. The string constant must have at least two characters to be considered a string and not a single character.

- REAL variables have 14 digits of precision and require 8 bytes to store. REAL6 have six digits of precision and require four bytes to store. No time is saved, since all calculations are double-precision. REAL6 variables have other usage limitations, and are best saved for situations where space is critical.
- The program name may be omitted from the PROGRAM header statement. In addition, none of the standard file designators (INPUT and OUTPUT) are required

declared in the PROGRAM header. These files are always available, as is the file LP for line printer output. INPUT, OUTPUT and LP are declared to be type "FILE OF TEXT".

The following intrinsic procedures and functions are added to those implemented in the standard definition of the language:

1. CALL(address,value) — type INTEGER function which places an 8-bit *value* in the Z-80's A-register and calls the routine located at *address*. The function's return value is the 8-bit *value* left in the A-register when the Z-80 return instruction is executed from the routine called.
2. CLOSE — close all open disk files.
3. CLS — clear the video screen.
4. EX(real-expression) — type INTEGER function which returns the exponent of the numerical result of evaluating the real-expression. For example, EX(10.1234) = 2.
5. FP(real-expression) — type REAL function which returns a real value in the range of 0 to 1 corresponding to the mantissa of the *real-expression* as its function value. For example, FP(10.1234) = 0.101234.
6. INKEY — type CHAR function which returns as its function value the 8-bit *value* of the keyboard. CHR(0) is returned if no key is pressed. This is identical to the operation of the BASIC INKEY$ function.
7. MEM — type INTEGER function returning the value corresponding to the amount of free memory available.
8. PEEK(address) — type INTEGER function returning the 8-bit *value* from memory at *address* as its function value.
9. POKE(address,value) — normally, a procedure to place the 8-bit *value* in memory at *address*. This procedure has been extended to provide rapid access to the TRS-80 graphics capability. If *address* is in the range of 0 to 127, then a SET operation is performed interpreting *address* as the x coordinate and *value* as the y coordinate of the pixel. If *address* is in the range of 128 to 255, then a RESET operation is performed at the pixel defined by (*address-128, value*). If *address* is in the range of 256 to 383, then the pixel at the pixel defined by (*address-256, value*) is tested and location 21458 is set non-zero to reflect that the pixel is set, zero otherwise.
10. SEEK(expression,filename) — procedure to position the file specified by *filename* to the record in the file pointed to by the integer part of *expression*.

A compile-time function allows you to INCLUDE source code in the current program from the disk by specifying the filename of the source file as follows: (*$ filename *). Source code is compiled into the program at the location of the INCLUDE statement.

The CASE statement has two extensions from Standard Pascal: 1) An optional ELSE clause may be specified and will be executed if none of the specified conditions are satisfied. 2) If the ELSE clause is not specified and none of the CASE conditions are satisfied, execution flows to the next executable statement rather than raising an error condition.

The most significant limitations of the *Pascal-80* compiler appear to be the omission of pointer variables, variant records, and the operators for direct manipulation of the "heap" such as NEW and DISPOSE. Listed below are the deviations from standard Pascal implementation:

- Variant records are not implemented.
- WITH ... DO is not implemented.
- Pointer variables, including NEW and DISPOSE, are not implemented.
- File window (buffer) variables, including GET and PUT, are not implemented.
- PACK and UNPACK are not implemented. All data structures are already allocated based on the byte data structure of TRS-80 memory.
- PAGE is not implemented, but WRITE(LP,CHR(12)) works the same.
- Structures of files (i.e., ARRAY OF FILE, etc.) are not implemented.
- Sets are limited to 256 members and numeric elements must be in the range of 0 to 255.
- A procedure or function identifier may not be passed as a parameter.
- No expression passed as a value parameter to a procedure or function may exceed 510 bytes unless it is passed as a VAR parameter.
- Integer variables used to reference array elements in a record must be global variables.
- Model I users may use "(." for the left bracket and ".)" for the right bracket.
- Spaces are significant in some isolated cases where they would not be significant in Standard Pascal.

Additional Features

The *Pascal-80* master diskette contains source code for INCLUDE procedures to link to the graphics capabilities, random numbers, and position the video cursor. Several demonstration programs are included to illustrate features of the language.

Two utility programs are provided for manipulation of *Pascal-80* source code files. Due to the internal space-compression algorithm, *Pascal-80* source code files are not compatible with any other ASCII file processors (such as *Scripsit* or *Electric Pencil*). The file ASCII/CMD will read a *Pascal-80* file and write a standard ASCII character file from the compressed source code, terminating each line with the character 0DH. The second file, TEXT/CMD, reads an ASCII file and converts it to the proper compressed mode for use by *Pascal-80*. I find that the editor in *Pascal-80* is not powerful enough to meet all demands. These programs allow you to use any of the powerful word processors currently sold to edit the source code. One note: *Electric Pencil* requires that the ASCII file end with a 00H byte.

The remaining files, AUTHOR/SRC and AUTH CODE/CMD, provide you with the capability to convert any *Pascal-80* source program into a CMD file which will execute directly from the "DOS READY" prompt. No royalty fee is required! New Classics Software grants a license to all registered owners of *Pascal-80* to distribute any compiled program provided that you meet the very lenient requirements specified in the *Pascal-80* User's Manual. A minimal Pascal program of just a BEGIN END block requires 9 granules of disk space as a CMD file due to the size of the p-code interpreter. Do not be too distressed by this size until you compare this to the space required by the program in Figure 1 (10 granules) as opposed to the same program compiled using FORTRAN (12 granules)!

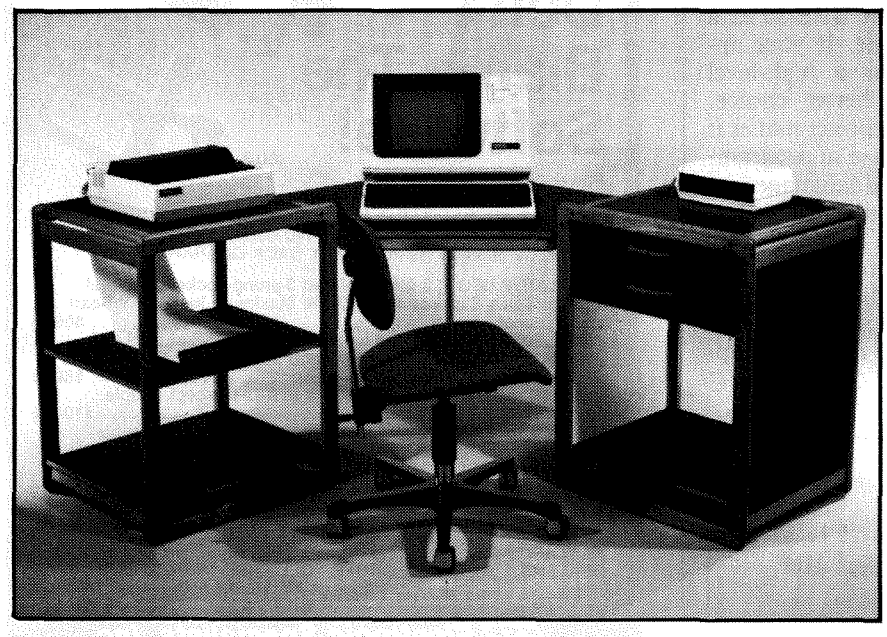
Conclusion

I have attempted to portray *Pascal-80* for what it is — the first real attempt to provide a useful implementation of this powerful and popular language for the TRS-80 at a reasonable price. Pascal is not for everyone. If your applications require speed and a need for a high-level language, then FORTRAN is a much better choice. *Pascal-80* is not slow, but the use of an interpreter makes it slower than native Z-80 machine code. If ease in programming using the elegance of a well-structured language in a friendly, interactive environment is your objective, then buy *Pascal-80.*

Mr. Blank and Mr. Koch have done a great job of improving the superlative system written by Mr. Gates. With the support demonstrated by New Classics Software, it would not surprise me to see a derivative of *Pascal-80* supplant BASIC as the language of the future.

Figure 1

```
program PRIME;

(*Benchmark program taken from ""A High-Level Language
   Benchmark'' by Jim Gilbreath in BYTE, Sept 1981,
   page 182 *)

  const
    size = 8190;

  var
    flags : array(. 0..size .) of boolean;
    i, prime, k, count, iter : integer;

  begin
    writeln('1 iteration');
    for iter := 1 to 1 do
      begin
        count := 0;
        for i := 0 to size do
          flags(.i.) := true;
        for i := 0 to size do
          if flags(.i.)
            then
              begin
                prime := i + i + 3;
                k := i + prime;
                while k <= size do
                  begin
                    flags(.k.) := false;
                    k := k + prime
                  end;
                count := count + 1;
                write(prime:8)
              end
      end;
    writeln;
    writeln(count,' primes')
  end.
```

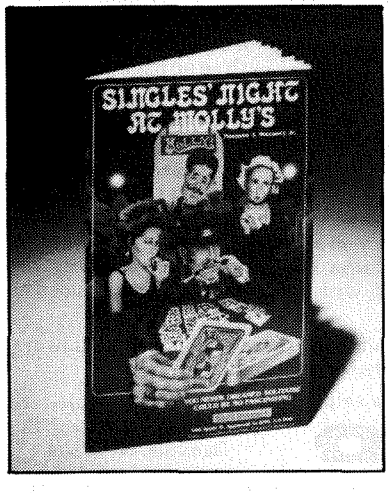


A Corner For Your Computer

JOHN JAMES FURNISHINGS
9015-A Meadow Vista Boulevard
Houston, TX 77064
(713) 469-4508

Are your computer and assorted peripherals threatening to overun your household? *Comp-U-Corner*™ utilizes normally wasted corner space, enabling the user to have a home computer center without having to dedicate an entire room to it.

Comp-U-Corner features a choice ot wood finishes and laminates, printer paper slots, adjustable shelves and a five year guarantee. There are also many options available — a drawer conversion kit, a mounted surge protector/outlet run, and the Cable Roundup®, which organizes the many wires required by today's equipment. As the need arises for more work and storage space, additional components provide limitless opportunities to customize your electronic environment, and are easily added to the basic package. The price of the *Comp-U-Corner* is $475.00.



Play Cards With Your Apple™

SOFT IMAGES
200 Route 17
Mahwah, NJ 07430
(201) 529-1440

Singles' Night At Molly's consists of two challenging solitaire card games — Royal Flush and Sly Fox. Both games feature Hi-Res graphics and various difficulty levels requiring a considerable amount of strategy and playing skill. No matter how many times either game is played, no two games will ever be exactly alike. Games can be played by one person, or any number of players.

The *Singles' Night* package includes a diskette and a 28 page instruction booklet. It requires a 48K Apple II or II Plus with Applesoft in ROM or a Language Card and DOS 3.3. It is priced at $29.95.

New Text On Microsoft BASIC

DILITHIUM PRESS
11000 S.W. 11th Street
Suite E
Beaverton, OR 97005
(503) 646-2713

MICROSOFT BASIC, 2nd Edition by Ken Knecht starts with an introduction to programming in BASIC, and a glossary of the computer terms that are used throughout the book. Knecht then covers such important topics as branching and loops, arithmetic in BASIC, strings, editing, arrays and files, the disk and additional useful features.

This book describes the latest version of Microsoft BASIC, release 5.0. While this version of BASIC-80 is the newest version of Microsoft BASIC, some popular microcomputers may use a slightly different version. Differences, however, are small, and easily changed by adaptations shown in the book.

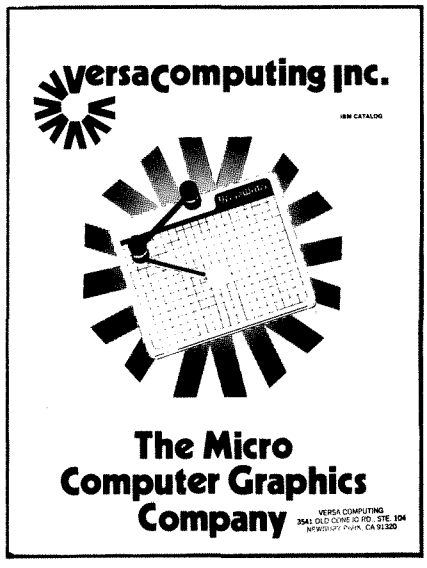
MICROSOFT BASIC, 2nd Edition requires only a basic understanding of computer fundamentals. Through examples that actually run, the reader is shown how this powerful version of BASIC can save valuable programming time and effort. The book is available for $14.95.

New Speed For Your Apple™

DIVERSIFIED SOFTWARE RESEARCH, INC.
5848 Crampton Court
Rockford, IL 61111
(815) 877-1343

Diversi-DOS is a new Apple DOS 3.3 compatible operating system for the Apple computer. It loads and saves BASIC, Binary, and Text files two to five times faster than standard DOS 3.3. All DOS commands, including INIT, are preserved. In addition, two utility programs are included. The keyboard buffer utility allows rapid typing without missing characters, even during disk operations. The print buffer utility temporarily saves characters on a standard 16K RAM card until the printer is ready. A simple, menu-driven, installation program is included on the unprotected disk.

Diversi-DOS requires a 48K Apple II or II Plus with DOS 3.3, and may be ordered directly from Diversified Software Research for $30.00.



Graphics For The IBM® PC

VERSA COMPUTING, INC.
3541 Old Conejo Road
Suite 104
Newbury Park, CA 91320
(805) 498-1956

The *VersaWriter Drawing Tablet* plugs directly into the IBM PC Game Control Adapter's connector and allows immediate entry of graphics to the IBM PC 320 x 200 and 640 x 200 screens. No expansion slot space is required. Over 30 graphics commands and more than 100 color options are available. Graphics produced with the *VersaWriter* can be saved on disk, and printed as hardcopy to an Epson MX80 FT Printer.

The *VersaWriter Drawing Tablet* is available now at a retail price of $299.00.

New Logo Newsletter Available

THE NATIONAL LOGO EXCHANGE
Attn: KTL
Box 5341
Charlottesville, VA 22905

The National LOGO Exchange is a monthly newsletter providing teachers with practical suggestions for implementing LOGO in the classroom. Published September through May, it serves as a forum for the exchange of ideas, philosophies, and techniques of teaching and using LOGO. Columns by well-known professional educators are featured each month, as well as practical articles submitted by subscribers.

The National LOGO Exchange is available for $25.00/year in the U.S., and $30.00 elsewhere.

TRS-80® Disk Management System

HEXAGON SYSTEMS
P.O. Box 397, Station A
Vancouver, B.C. Canada V6C 2N2
(604) 682-7646

HEXMAN performs disk management chores that previously had to be done manually. It consists of a series of interlinked programs with an easy-to-use menu of functions.

HEXMAN does automatic daily backups of amended files, detects the creation of new files, automatically catalogs them in its file index, and will retrieve and load a file or group of files as needed.

HEXMAN requires a 48K TRS-80 Model I or III with two disk drives and the LDOS Operating System. It is available from Hexagon Systems for $169.00.

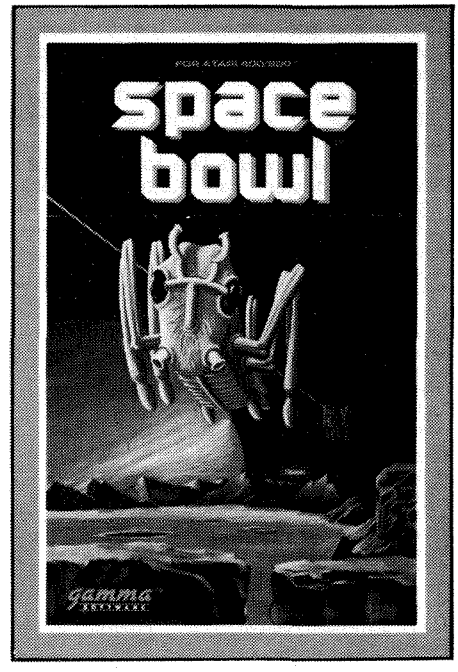
TRS-80® C-Language Compiler

MISOSYS
P.O. Box 4848
Alexandria, VA 22303
(703) 960-2998

LC is a C-language compiler for the TRS-80 Model I/III. It is an integer-only compiler that supports all statements except "struct", "union", and "typedef". *LC* supports all operators except "- ", ".", "sizeof", and "typename".

LC supports I/O redirection, command line arguments, dynamic memory management, and sequential files for read, write, and append. Floating point routines in ROM are accessible via function calls supported in FP/LIB. An extensive installation library supports graphics routines, string routines, DOS calls, port I/O, and more. *LC* generates ROMable Z-80 assembler source code compatible with EDAS IV.

LC requires a TRS-80 Model I or III with two disk drives and LDOS 5.1.x. It comes with more than 200 pages of documentation and is priced at $175, plus $4 shipping.



Enjoy Sports In Space On Your ATARI®

GAMMA SOFTWARE
P.O. Box 25625
Los Angeles, CA 90025
(213) 473-7441

Space Bowl™ is a high speed Atari computer game which introduces a new genre — a combination space game and sports game.

Imagine a championship sporting event for a species of extraterrestrials called Denebs. The field of battle is black space. Goals orbit the field and serve as moving targets for the fierce play action. The Denebs undergo reincarnation when zapped by lasers, and can make themselves invisible, making *Space Bowl* unlike any sporting event on earth.

Space Bowl is for two players and requires an Atari 400/800 with 16K RAM. It is available on both cassette and disk for a retail price of $29.95.

Submissions may be sent to:
***SoftSide* Magazine**
New Products Editor
6 South Street
Milford, NH 03055

Where To Buy SoftSide.....

On this and the next few pages you'll find over 1,000 retail stores — computer shops and book stores — that sell **SoftSide** Magazine each month. And the list gets longer with each issue we publish.

We invite you to patronize these fine stores for all your computer product needs...including **SoftSide** Magazine.

And, if you would like to become a **SoftSide** dealer, call Kathie Maloof, (603) 673-0585, for information.

ALASKA

B Dalton Books	Anchorage
Walden Books	Anchorage

ALABAMA

Anderson Computers	Huntsville
B Dalton Bookseller	Birmingham
Madison Books	Madison
Olonsky Brothers	Mobile
Trade N Books	Montgomery
Village Green Bookstore	Gardendale

ARIZONA

All Systems Go	Tempe
B Dalton Bookseller	Phoenix
B Dalton Bookseller	Tucson
B Dalton Bookseller	Flagstaff
B Dalton Bookseller	Little Rock
B Dalton Bookseller	Fayetteville
Computer Mart	Phoenix
Computerland Tempe Mesa	Mesa
Copper State Cash Register	Phoenix
Expotek	Phoenix
Heathkit Electric Center	Phoenix
Personal Computer Place	Mesa
Royal Bookstore	Phoenix
Walden Books	Phoenix
Walden Books	Mesa
Walden Books	Scottsdale
Walden Books	Tucson

ARKANSAS

B Dalton Bookseller	Little Rock
B Dalton Bookseller	Lafayette
Bits N Bytes	Fort Smith
Publishers Bookshop	Little Rock
Vaughn Electronics Corp	Rogers
Walden Books	North Little Rock

CALIFORNIA

A Clean Well Lighted Place	Cupertino
Apple of Orange	Orange
Atari Service Center	La Mesa
Avenue Books	Stockton
B Dalton Bookseller	Hollywood
B Dalton Bookseller	Los Angeles
B Dalton Bookseller	Rolling Hills
B Dalton Bookseller	Santa Monica
B Dalton Bookseller	Whittier
B Dalton Bookseller	Carson
B Dalton Bookseller	Northridge
B Dalton Bookseller	N Hollywood
B Dalton Bookseller	Carlsbad
B Dalton Bookseller	Manhatt Beach
B Dalton Bookseller	Indio
B Dalton Bookseller	Costa Mesa
B Dalton Bookseller	El Toro
B Dalton Bookseller	Westminster
B Dalton Bookseller	Mission Viejo
B Dalton Bookseller	Oxnard
B Dalton Bookseller	Santa Barbara
B Dalton Bookseller	Visalia
B Dalton Bookseller	Lancaster
B Dalton Bookseller	Fresno
B Dalton Bookseller	Salinas
B Dalton Bookseller	Pacific Gr
B Dalton Bookseller	Mountain View
B Dalton Bookseller	Sunnyvale
B Dalton Bookseller	Walnut Creek
B Dalton Bookseller	Cupertino
B Dalton Bookseller	Sacramento
B Dalton Bookseller	Redding
Bay Co Books	Santa Cruz
Byte Shop Computer Store	Citrus Heights
Capital Computer Systems	Sacramento
Computer Age Inc.	San Diego
Computer Emporium	San Jose
Computer Mart	Diamond Bar
Computer Palace	Glendale
Computer Post	Newark
Computer Store	Santa Monica
Computer Wave, Inc.	Westminster
Computerland Fresno	Fresno
Computerland Newhall	Newhall
Computerland of Laguna Hills	Laguna Hills
Computerland of South Bay	Lawn Dale
Computerland of West L.A.	W Los Angeles
Computerland Sacramento	Sacramento
Computerland San Bernardino	San Bernardino
Computerland San Diego	San Diego
Computerland San Leandro	Hayward
Computerland San Maria	San Maria
Computerland Santa Barbara	Santa Barbara
Computerland Stockton	Stockton
Computerland Thousand Oaks	Thousand Oaks
Computerland Woodland Hills	Woodland Hills
Computertime	Citrus Heights
Dimensional Software	San Diego
3-E Software	Hayward
Family Computing, Inc.	Encinitas
Futurevision	Napa
Game Gallery	Carmel
Harding Way News	Stockton
Leavity Distributors	N Hollywood
Malibu Microcomputing	Malibu
Micro Age Computer Store	Mountainview
National Computer Center	Fresno
Net Profit Computers	Anaheim
On Line Computer Center	Sacramento
On Line Micro Center	Fresno
Opamp Technical Books	Los Angeles
Paradise Computer Systems	San Luis Obispo
PC Computers	El Cerrito
Printers Ink Bookstore	Palo Alto
Quement Electronics	San Jose
Rainbow Computing	Northridge
Software Center	San Francisco
Software Center	Santa Ana
Software Central	Pasadena
Software City	San Jose
Software Emporium	Los Altos
Software Plus	Citrus Heights
Software Store	Los Angeles
Sunset Computers	San Francisco
Tower Books	Sacramento
Walden Books	Los Angeles
Walden Books	Culver City
Walden Books	Downey
Walden Books	Santa Monica
Walden Books	Torrance
Walden Books	Buena Park
Walden Books	Cerritos
Walden Books	Lakewood
Walden Books	Pasadena
Walden Books	Glendale
Walden Books	Canoga Park
Walden Books	Northridge
Walden Books	Thousand Oaks
Walden Books	Woodland Hills
Walden Books	Panorama City
Walden Books	Sherman Oaks
Walden Books	City of Industry
Walden Books	Upland
Walden Books	West Covina
Walden Books	Chula Vista
Walden Books	El Cajon
Walden Books	National City
Walden Books	San Diego
Walden Books	Hemet
Walden Books	San Bernadino
Walden Books	Brea
Walden Books	Huntington Beach
Walden Books	Laguna Hills
Walden Books	Westminster
Walden Books	Mission Viejo
Walden Books	Anaheim
Walden Books	Ventura
Walden Books	Salinas
Walden Books	Monterey
Walden Books	Burlington
Walden Books	Mountainview
Walden Books	San Bruno
Walden Books	San Francisco
Walden Books	Palo Alto
Walden Books	San Mateo
Walden Books	Concord
Walden Books	El Cerrito
Walden Books	Fairfield
Walden Books	Fremont
Walden Books	Hayward
Walden Books	Newark
Walden Books	San Leandro
Walden Books	Berkeley
Walden Books	San Rafael
Walden Books	Cupertino
Walden Books	San Jose
Walden Books	Modesto
Walden Books	Santa Rosa
Walden Books	Citrus Heights
Walden Books	Sacramento
Zackit Electronics	Monterey

COLORADO

B Dalton Bookseller	Aurora
B Dalton Bookseller	Littleton
B Dalton Bookseller	Denver
B Dalton Bookseller	Boulder
B Dalton Bookseller	Ft. Collins
B Dalton Bookseller	Grand Junction
Video World	Englewood
Walden Books	Aurora
Walden Books	Westminster
Walden Books	Littleton
Walden Books	Denver
Walden Books	Ft. Collins
Walden Books	Colorado Springs
Walden Books	Pueblo
Wholelife Distributors	Denver

CONNECTICUT

Aetna Life Club Store	Hartford
B Dalton Bookseller	Bloomfield
B Dalton Bookseller	Farmington
B Dalton Bookseller	Westport
Computer Center	Old Saybrook
Computerifics	E. Hartford
Computerworks	Westport
Marks Stationary Store	Greenwich
Walden Books	Enfield
Walden Books	Hartford
Walden Books	Willimantic
Walden Books	Ansonia
Walden Books	Meriden
Walden Books	Southbury
Walden Books	New Haven
Walden Books	Bridgeport
Walden Books	Trumbull
Walden Books	Waterbury
Walden Books	Danbury
Walden Books	Greenwich
Walden Books	Westport
Walden Books	Wilton
Walden Books	Stamford

DELAWARE

Micro Products	Wilmington
Walden Books	Wilmington
Walden Books	Dover

DISTRICT OF COLUMBIA

Program Store	DC
Walden Books	DC

FLORIDA

American/Software Tech Inc.	N. Bay Village
B Dalton Bookseller	Daytona Beach
B Dalton Bookseller	Tallahassee
B Dalton Bookseller	Panama City
B Dalton Bookseller	Pensacola
B Dalton Bookseller	Mary Esther
B Dalton Bookseller	Orlando
B Dalton Bookseller	Miami
B Dalton Bookseller	Boca Raton
B Dalton Bookseller	Pinellas Park
B Dalton Bookseller	Tampa
B Dalton Bookseller	Naples
Book & Variety Store	Selung
Byte Shop of Miami	Miami
City News Stand	Melbourne
Clarks Out of Town News	Ft. Lauderdale
Computech	Tallahassee
Computer Haven Inc.	W. Melbourne
Computer Image	Miami
Computer Room	W. Palm Beach
Computer Scene	Miami
Computer Scene	N. Miami Beach
Computerland	Boca Raton
Computerland	Clearwater
Computerland Lakeland	Lakeland
Computerland Tallahassee	Tallahassee
Computerland W Palm Beach	W. Palm Beach
Corner Book Shop	N. Miami Beach
Extra Extra	Tampa
Goerings Book Center	Gainesville
Grice Electronics	Pensacola
H.I.S. Computermation	Melbourne

Home & Professional Comp.	Miami
IBARS Electronics Corp.	Hialeah
Ives Book Shop	Fort Meyers
Lafayette Radio	Ft. Lauderdale
Plaza Bookstore Inc.	Hollywood
Rainbow Computers	Orlando
Rays Computer Center	Clearwater
Software Store	Tampa
Sound Ideas Microcomputer	Gainesville
Sunnys At Sunset Inc.	Sunrise
The Logical Choice	St. Petersburg
Walden Books	Orange Park
Walden Books	Tallahassee
Walden Books	Pensacola
Walden Books	Gainesville
Walden Books	Altamonte Springs
Walden Books	Orlando
Walden Books	Merritt Island
Walden Books	Hialeah
Walden Books	Hollywood
Walden Books	Miami
Walden Books	Fort Lauderdale
Walden Books	W. Palm Beach
Walden Books	Bradenton
Walden Books	Clearwater
Walden Books	Tampa
Walden Books	Fort Myers

GEORGIA

Atlanta Computer Mart	Atlanta
B Dalton Bookseller	Decatur
B Dalton Bookseller	Atlanta
B Dalton Bookseller	Augusta
Computerland Athens	Athens
Oxford Books Inc	Atlanta
Software-To-Boot	Savannah
The Bookstore	Valdosta
Walden Books	Roswell
Walden Books	Atlanta
Walden Books	Athens
Walden Books	Augusta
Walden Books	Savannah
Walden Books	Columbus
Walden Books	Morrow

HAWAII

Walden Books	Alea
Walden Books	Honolulu

IDAHO

B Dalton Bookseller	Idaho Falls
B Dalton Bookseller	Nampa
Bookpeople of Moscow Inc.	Moscow
Idaho Peoples Computer Store	Buhl
Pioneer Bookstore Inc.	Idaho Falls
Walden Books	Chubbuck
Walden Books	Moscow

ILLINOIS

B Dalton Bookseller	Waukegan
B Dalton Bookseller	Bloomingdale
B Dalton Bookseller	St. Charles
B Dalton Bookseller	Schaumburg
B Dalton Bookseller	Champaign
B Dalton Bookseller	Springfield
Complete Computing Corp.	Lombard
Computerland	Champaign
Computerland	Mundelin
Computerland	Naperville
Computerland	Schaumburg
Computerland Arlington Heights	Arlington Heights
Computerland of Oakpark	Oakpark
Computerland Peoria	Peoria
Data Domain	Schaumburg
Digitalworld	Addison
Godfrey Bookstore	Godfrey
Krochs & Bretanos	Chicago
Krochs & Bretanos	Oak Park
Krochs & Bretanos	Oakbrook
Krochs & Bretanos	Rockford
Krochs & Bretanos	Skokie
Libertyville Video & Computer	Libertyville
Midwest Microcomputers	Lombard
Nabih's Inc.	Evanston
Oak Brook Computer Center	Oak Brook
Prairie News Agency	Chicago
Shadids Book Mart	Springfield
Taylors Book Shop	Centralia
The Computer Store/Rockford	Rockford
TZ Computers	Bloomington
Video Etc.	Deerfield
Videomat	Chicago
Walden Books	Arlington Heights
Walden Books	Northbrook
Walden Books	Waukegan
Walden Books	Dundee
Walden Books	Lombard
Walden Books	Villa Park
Walden Books	Schaumburg
Walden Books	Joliet
Walden Books	Bolingbrook
Walden Books	Orland Park
Walden Books	Park Forest
Walden Books	Aurora
Walden Books	North Riverside
Walden Books	Chicago
Walden Books	Niles
Walden Books	Sterling
Walden Books	Rockford
Walden Books	Moline
Walden Books	Ottawa
Walden Books	Normal
Walden Books	Urbana
Walden Books	Mattoon

INDIANA

A Computer Store	Indianapolis
B Dalton Bookseller	Greenwood
B Dalton Bookseller	Indianapolis
B Dalton Bookseller	Ft. Wayne
B Dalton Bookseller	Bloomington
B Dalton Bookseller	Evansville
B Dalton Bookseller	Terre Haute
Bauer Electronics	Lawrenceburg
Bytrex	Ft. Wayne
Computer Store	Marion
Computerland	Anderson
Computerland	Ft. Wayne
Computerland Mishawaka	Mishawaka
Computerland	Indianapolis
Computerland Evansville	Evansville
Data Domain	Bloomington
Emprise Computer System	Indianapolis
Game Preserve	Indianapolis
Hoosier Electronics	Terre Haute
Indiana News	Indianapolis
Twin City News	Lafayette
Vons Bookshop	W. Lafayette
Walden Books	Indianapolis
Walden Books	Merrillville
Walden Books	Mishawaka
Walden Books	South Bend
Walden Books	Fort Wayne
Walden Books	Kokomo
Walden Books	Evansville
Walden Books	Terre Haute
Walden Books	Lafayette

IOWA

B Dalton Bookseller	Des Moines
B Dalton Bookseller	Mason City
B Dalton Bookseller	Sioux City
B Dalton Bookseller	Spencer
B Dalton Bookseller	Iowa City
B Dalton Bookseller	Cedar Rapids
B Dalton Bookseller	Muscatine
Interstate Bookstore	Davenport
Memory Bank	Bettendorf
Walden Books	Ames
Walden Books	Marshalltown
Walden Books	West Des Moines
Walden Books	Des Moines
Walden Books	Waterloo
Walden Books	Sioux City
Walden Books	Dubuque
Walden Books	Iowa City
Walden Books	Cedar Rapids
Walden Books	Davenport

KANSAS

Amateur Radio Equipment	Wichita
B Dalton Bookseller	Overland Park
B Dalton Bookseller	Topeka
B Dalton Bookseller	Wichita
Computerland	Topeka
Crossroads Inc.	Hutchinson
Micro Age Computer	Salina
Mr. Computer	Emporia
Walden Books	Overland Park
Walden Books	Wichita

KENTUCKY

B Dalton Bookseller	Florence
Computer Emporium	Louisville
Walden Books	Louisville
Walden Books	Lexington

LOUISIANA

B Dalton Bookseller	New Orleans
B Dalton Bookseller	Lafayette
Computerland/Louisiana	New Orleans
Sidneys Newsstand Uptown	New Orleans
The Book Shelf	Slidell
Walden Books	Metairie
Walden Books	Gretna
Walden Books	New Orleans
Walden Books	Houma
Walden Books	Lafayette
Walden Books	Lake Charles
Walden Books	Baton Rouge

MAINE

B Dalton Bookseller	Bangor
Bookland of South Portland	South Portland
Harper Electronics	Portland
Walden Books	South Portland

MARYLAND

B Dalton Bookseller	Bethesda
B Dalton Bookseller	Gaithersburg
B Dalton Bookseller	Kensington
B Dalton Bookseller	Cockeysville
B Dalton Bookseller	Columbia
B Dalton Bookseller	White Marsh
B Dalton Bookseller	Baltimore
B Dalton Bookseller	Annapolis
B Dalton Bookseller	Frederick
Book House	Ocean City
Bowes Books	Lexington Park
Business Computer Center	Ellicott City
Chafitz	Rockville
Computer Age Inc.	Silver Springs
Computer Unlimited	Towson
Greetings & Readings	Towson
Layhill News Stand	Silver Springs
The Computer Forum	Severna Park
Walden Books	Forestville
Walden Books	Hillcrest Heights
Walden Books	Bethesda
Walden Books	Laurel
Walden Books	Kensington
Walden Books	Wheaton
Walden Books	Cockeysville
Walden Books	Baltimore
Walden Books	Parkville
Walden Books	Rosedale
Walden Books	Annapolis
Walden Books	La Vale
Walden Books	Frederick
Walden Books	Hagerstown

MASSACHUSETTS

B Dalton Bookseller	Framingham
Computer/Source	Pittsfield
Custom Electronics Inc.	Chicopee
Ninis Corner	Cambridge
Norwood News & Bookshop	Norwood
Voyager Bookshop	Brockton
Walden Books	Hadley
Walden Books	Holyoke
Walden Books	Auburn
Walden Books	Andover
Walden Books	Billerica
Walden Books	Methuen
Walden Books	Saugus
Walden Books	Swampscott
Walden Books	East Walpole
Walden Books	Boston
Walden Books	Somerville
Walden Books	Medford
Walden Books	Newton Centre
Walden Books	Lexington
Walden Books	Plymouth
Walden Books	North Dartmouth
Walden Books	Swansea

MICHIGAN

B Dalton Bookseller	Novi
B Dalton Bookseller	Rochester
B Dalton Bookseller	Southfield
B Dalton Bookseller	Sterling Heights
B Dalton Bookseller	Troy
B Dalton Bookseller	Warren
B Dalton Bookseller	Ann Arbor
B Dalton Bookseller	Taylor
B Dalton Bookseller	Flint
B Dalton Bookseller	Essexville
B Dalton Bookseller	Okemos
B Dalton Bookseller	Lansing
B Dalton Bookseller	Portage
B Dalton Bookseller	Benton Harbor

B Dalton Bookseller	Houghton
Binary Corp.	Berkley
Book Lovers Shop	Sterling Heights
Book Nook	Allen Park
Computer Horizons	Livonia
Computer Mart	Ann Arbor
Computer Mart	E. Lansing
Computer Mart	Flint
Computers And More	Grand Rapids
Computerland	Southfield
Computerland Grosse Pt.	St. Clair Shores
Family Computer Center	Berkley
Heathkit Electronic	Detroit
Lyceum Microcomputers	Warren
Merit Book Center	Detroit
Metro News Center	Birmingham
Micro Computer Diversified	Alma
Micro Station	Southfield
New Horizon Book Shop	Roseville
Northland Computer	Traverse City
Paper Tiger	Livonia
Plymouth Book World	Plymouth
Rainbow Computers	Troy
Rochester Book Center	Rochester
Simtec	Birmingham
Thompson News	Traverse City
Tri City Computer Mart	Saginaw
Walden Books	Novi
Walden Books	Rochester
Walden Books	Sterling Heights
Walden Books	Ann Arbor
Walden Books	Dearborn
Walden Books	Detroit
Walden Books	Saginaw
Walden Books	Kalamazoo
Walden Books	Grand Rapids
Walden Books	Traverse City

MINNESOTA

B Dalton Bookseller	St. Paul
B Dalton Bookseller	Maplewood
B Dalton Bookseller	W St. Paul
B Dalton Bookseller	Minnetonka
B Dalton Bookseller	Minneapolis
B Dalton Bookseller	Golden Valley
B Dalton Bookseller	Brooklyn Center
B Dalton Bookseller	Edina
B Dalton Bookseller	Duluth
B Dalton Bookseller	Mankato
B Dalton Bookseller	St. Cloud
B Dalton Bookseller	Fergus Falls
Fine Print Inc.	Northfield
Read More Book Store	Minneapolis
Schaak Electronics	St. Cloud
Spectrum Computers	Lathrup Village
Walden Books	Maplewood
Walden Books	Roseville
Walden Books	Burnsville
Walden Books	Eden Prairie
Walden Books	Minneapolis
Walden Books	St. Louis Park
Walden Books	Blaine
Walden Books	Duluth
Walden Books	Rochester
Wizard's Work	New Hope
Zim Computers	Brooklyn Center

MISSISSIPPI

Walden Books	Jackson
Walden Books	Laurel

MISSOURI

B Dalton Bookseller	Chesterfield
B Dalton Bookseller	St. Louis
B Dalton Bookseller	Creve Coeur
B Dalton Bookseller	Kansas City
B Dalton Bookseller	Springfield
Back To BASICS Computer	Florissant
Computerland West City	Manchester
Cowley Distributing	Jefferson City
Discount Books	Webster
Fredericks Computer Products	Frederick
House of Computer	Blue Spring
Instant Replay Ltd.	Chesterfield
Micro Age Computer	St. Louis
Walden Books	Independence
Walden Books	Kansas City
Walden Books	Jefferson City
Walden Books	Springfield
Walden Books	Landover
Walden Books	Belair

MONTANA

B Dalton Bookseller	Bozeman
B Dalton Bookseller	Missoula
Computerland Billings	Billings
Walden Books	Billings
Westgate News	Billings

NEBRASKA

B Dalton Bookseller	Omaha
B Dalton Bookseller	Lincoln
Hatch Radio Shack	Alliance
Hobbytown	Lincoln
2001 Inc.	Norfolk
Micro Age	Omaha
Nebraska Bookstore	Lincoln
Walden Books	Omaha
Walden Books	Lincoln
Walden Books	Grand Island

NEVADA

Century 23	Las Vegas
Computerland Reno	Reno
Home Computers	Las Vegas
PCS Computers	Las Vegas
Walden Books	Las Vegas
Walden Books	Reno

NEW HAMPSHIRE

B Dalton Bookseller	Manchester
Computer Mart of NH	Merrimack
Computer Shoppe	Nashua
Computer Town	Salem

NEW JERSEY

Apple Core Computer	Basking Ridge
B Dalton Bookseller	Wayne
B Dalton Bookseller	Morristown
B Dalton Bookseller	Moorestown
B Dalton Bookseller	Lawrenceville
B Dalton Bookseller	Toms River
B Dalton Bookseller	Edison
Computer Corner of NJ	Pompton Plains
Computer Dimensions	Westfield
Computer Discount of NJ	W Milford
Computer Mart of NJ	Iselin
Computer Port	Northfield
Computerland Cherry Hill	Cherry Hill
Computerland New Castle City	Newark
Computerland of Morristown	Morristown
Earthrise Computer Center	Madison
Felice's Follies	Red Bank
Garden State News	Fort Lee
Gemini	Cedar Knolls
Micro Concepts	Morris Plains
Midas Data Systems Inc.	Marlton
Software/Asylum	Carteret
Software City	Fairview
Software City	Montvale
Software City	Summit
Software City	Teaneck
Unicorn Book Store	Hoboken
Vista Computers	Matawan
Walden Books	Livingston
Walden Books	Woodbridge
Walden Books	Wayne
Walden Books	Hackensack
Walden Books	Eatontown
Walden Books	Rockaway
Walden Books	Voorhees
Walden Books	Deptford
Walden Books	Lawrenceville
Walden Books	Toms River
Walden Books	East Brunswick

NEW MEXICO

B Dalton Bookseller	Albuquerque
Computerland of Santa Fe	Santa Fe
Computer/Tech Assoc.	Albuquerque
East West Enterprises	Albuquerque
Micro Age Computer Store	Albuquerque
The Book Company	Albuquerque
Walden Books	Albuquerque
Walden Books	Las Cruces

NEW YORK

ASD Office Systems Computer	Poughkeepsie
B Dalton Bookseller	New York
B Dalton Bookseller	Yonkers
B Dalton Bookseller	Great Neck
B Dalton Bookseller	New Hyde Park
B Dalton Bookseller	Flushing
B Dalton Bookseller	Lake Grove
B Dalton Bookseller	Massapequa
B Dalton Bookseller	Glens Falls
B Dalton Bookseller	Clay
B Dalton Bookseller	Cheektowaga
B Dalton Bookseller	Horseheads
BC Communications Inc.	Huntington Station
Castle Computers Inc.	Latham
Computer Center	Rochester
Computer Emporium	Middletown
Computer Resource	Williamsville
Computer Room Ltd.	Albany
Computer System Spec Inc.	Wappingers Falls
Computer Tree Inc.	Endwell
Computerland	Messina
Computerland of Ithaca	Ithaca
Computers Etc.	Dewitt
Compuworld	Rochester
Comtec	Brooklyn
Datascan Computer Systems	Farmingville
Digbytes	New York
Future Visions Computer	Melville
Hurst Video/Village	Massapequa Park
J & J Coins	Orchard Park
Little Professor Book Center	Great Neck
Micro Age Computer Store	Rochester
Omnifax Computer Store	Syracuse
Parkway Video	Brooklyn
Software Emporium	Roslyn Heights
Software Supply Intl.	Niagara
Syosset Video & Electronics	Syosset
Tastos Video Fair	Brooklyn
The Computer Shoppe	Patchogue
The Long Island Computer	Lynbrook
Video Connection	Clifton
Village Green Bookstore	Rochester
Walden Books	Staten Island
Walden Books	Portchester
Walden Books	Scarsdale
Walden Books	White Plains
Walden Books	Nanuet
Walden Books	Manhasset
Walden Books	Brooklyn Heights
Walden Books	Brooklyn
Walden Books	Forest Hills
Walden Books	Cedarhurst
Walden Books	Garden City
Walden Books	Greenvale
Walden Books	East Northport
Walden Books	Massapequa
Walden Books	Latham
Walden Books	Albany
Walden Books	Kingston
Walden Books	Fishkill
Walden Books	Poughkeepsie
Walden Books	Fayetteville
Walden Books	Syracuse
Walden Books	New Hartford
Walden Books	Utica
Walden Books	Johnson City
Walden Books	Buffalo
Walden Books	Williamsville
Walden Books	Cheektowaga
Walden Books	Victor
Walden Books	Rochester
Walden Books	Horseheads
Walden Books	Ithaca
World Wide Sales	Rochester

NORTH CAROLINA

B Dalton Bookseller	Winston Salem
B Dalton Bookseller	Cary
B Dalton Bookseller	Raleigh
B Dalton Bookseller	Durham
B Dalton Bookseller	Rocky Mount
B Dalton Bookseller	Gastonia
B Dalton Bookseller	Wilmington
B Dalton Bookseller	Asheville
Computer Alternatives	Asheville
Computer Room	Charlotte
Computerland Charlotte	Charlotte
K & S Newsstand Inc.	Winston Salem
North Carolina News Co.	Durham
Walden Books	Winston Salem
Walden Books	Goldsboro
Walden Books	Raleigh
Walden Books	Durham
Walden Books	Concord
Walden Books	Charlotte
Walden Books	Fayetteville
Walden Books	Kinston
Walden Books	Jacksonville

NORTH DAKOTA

B Dalton Bookseller	Fargo
B Dalton Bookseller	Grand Fork
B Dalton Bookseller	Bismarck
Computerland Grand Fork	Grand Fork
Computerland Bismarck	Bismarck

OHIO

Abacus Computer	Cincinnati
B Dalton Bookseller	Columbus
B Dalton Bookseller	Toledo
B Dalton Bookseller	Northwood
B Dalton Bookseller	Cleveland Heights
B Dalton Bookseller	Beachwood
B Dalton Bookseller	Akron
B Dalton Bookseller	Niles
B Dalton Bookseller	Youngstown
B Dalton Bookseller	Tiffin
B Dalton Bookseller	Cincinnati
B Dalton Bookseller	Dayton
Computerland Cleveland East	Mayfield Heights
Computerland Warren	Warren
Data Link	Dayton
Eric Martins Inc.	Maple Heights
Fine Print Books	Xenia
Hudson Computers	Hudson
Jerrys Computer	Cleveland
Micro Center	Columbus
Microconcepts	Akron
News Reader	Hillsboro
News Reader	Fairborn
Northcoast Computers	Bay Village
P.J.'s Bookstore	Dayton
Program Store	Columbus
The Computer Store	N Canton
The Home Computer Store	Westerville
The Ninth House Book Shoppe	Brecksville
The Open Book	Tiffin
The Open Book	Toledo
Video Computer World	Oregon
Video Depot	Cleveland
Walden Books	Columbus
Walden Books	Marion
Walden Books	Zanesville
Walden Books	Elyria
Walden Books	Mentor
Walden Books	North Olmsted
Walden Books	Cleveland
Walden Books	Fairview
Walden Books	North Randall
Walden Books	Euclid
Walden Books	Akron
Walden Books	Niles
Walden Books	Canton
Walden Books	Mansfield
Walden Books	Dayton
Walden Books	Springfield
Wilkie News	Dayton

OKLAHOMA

B Dalton Bookseller	Norman
B Dalton Bookseller	Oklahoma City
B Dalton Bookseller	Tulsa
Bartlesville Bookland	Bartlesville
Bettys Book Rack	Bartlesville
Caravan Books	Stillwater
Computerland of Oklahoma	Oklahoma City
Computers Plus	Duncan
Micro-com Radio Shack	Tonkana
On Line Micro Centers	Oklahoma City
Steves Sundry Books	Tulsa
Walden Books	Norman
Walden Books	Midwest City
Walden Books	Oklahoma City
Walden Books	Lawton
Walden Books	Tulsa
Walts News & Bookstore	Lawton

OREGON

B Dalton Bookseller	Portland
B Dalton Bookseller	Salem
B Dalton Bookseller	Eugene
Computer Solutions Inc.	Eugene
Computer Specialties	Salem
Marketplace Bookstore	Eugene
Micro Age Portland	Portland
Norvac Electronics	Beaverton
Universal Video	Oregon City
Walden Books	Portland
Walden Books	Salem
Walden Books	Eugene

PENNSYLVANIA

Abronovick Assoc. Inc.	Kittanning
AV Computer	Philadelphia
B Dalton Bookseller	W. Mifflin
B Dalton Bookseller	Monroeville
B Dalton Bookseller	Pittsburgh
B Dalton Bookseller	Butler
B Dalton Bookseller	Meadville
B Dalton Bookseller	State College
B Dalton Bookseller	Camp Hill
B Dalton Bookseller	Carlisle
B Dalton Bookseller	Harrisburg
B Dalton Bookseller	York
B Dalton Bookseller	Muncy
B Dalton Bookseller	Philadelphia
B Dalton Bookseller	Exton
B Dalton Bookseller	North Wales
B Dalton Bookseller	Plym. Mtg.
B Dalton Bookseller	Reading
Bookarama	Bethlehem
Bradd Alan Bookstore	Philadelphia
Central Wholesale	Pittsburgh
City Software Center	Philadelphia
Computerland	Feasterville
Computerland	Paoli
De Bug	Indiana
Doc's Computer Supply Ctr.	Ardmore
Erie Computer Co.	Erie
General Computer Store	Allenstown
Global Books Inc.	Feasterville
Integral Computer Service	Rochester
Newsy	Philadelphia
One Stop Computer Shop	Leymore
Pittsburgh Computer Store	Pittsburgh
Second Childhood	Beaver Falls
Star Gate	Mckeesport
Stevens Electronics	Phoenix
Television Parts Co.	New Brighton
The Computer Forum	Frazer
Walden Books	Monaca
Walden Books	Pittsburgh
Walden Books	Butler
Walden Books	Hanover
Walden Books	Muncy
Walden Books	Selingsgrove
Walden Books	Pottsville
Walden Books	Cornwells Heights
Walden Books	Langhorne
Walden Books	Media
Walden Books	Philadelphia
Walden Books	Center Square
Walden Books	North Wales
Walden Books	Plymouth Meeting
Walden Books	Pottstown
Walden Books	Reading

RHODE ISLAND

Walden Books	Newport
Walden Books	Lincoln
Walden Books	Walefield
Walden Books	Warwick
Walden Books	Cranston

SOUTH CAROLINA

B Dalton Bookseller	Columbia
B Dalton Bookseller	Spartanburg
B Dalton Bookseller	Charleston
B Dalton Bookseller	Greenville
News Stand	Clemson
Walden Books	Sumter
Walden Books	Columbia
Walden Books	North Charleston
Walden Books	Charleston
Walden Books	Greenville
Walden Books	Greenwood

SOUTH DAKOTA

B Dalton Bookseller	Sioux Falls
B Dalton Bookseller	Rapid City
Computer Terminal	Sioux Falls
Computerland	Sioux Falls
Computerland of Rapid City	Rapid City
Walden Books	Brookings
Walden Books	Sioux Falls
Walden Books	Huron
Walden Books	Pierre

TENNESSEE

B Dalton Bookseller	Kingston
B Dalton Bookseller	Memphis
Card N Book	Nashville
Computerland	Kingston
Computerland Of Memphis	Memphis
Educational Computing Sys.	Oak Ridge
Tobacco Corner	Memphis
Walden Books	Memphis

TEXAS

Austin News Agency	Austin
B Dalton Bookseller	Richardson
B Dalton Bookseller	Sherman
B Dalton Bookseller	Mesquite
B Dalton Bookseller	Dallas
B Dalton Bookseller	Arlington
B Dalton Bookseller	Ft Worth
B Dalton Bookseller	Waco
B Dalton Bookseller	Early
B Dalton Bookseller	Houston
B Dalton Bookseller	Lk Jackson
B Dalton Bookseller	Beaumont
B Dalton Bookseller	Victoria
B Dalton Bookseller	Laredo
B Dalton Bookseller	San Antonio
B Dalton Bookseller	Corpus Christi
B Dalton Bookseller	Austin
B Dalton Bookseller	Del Rio
B Dalton Bookseller	Midland
B Dalton Bookseller	El Paso
Baps	San Antonio
Books, Etc.	San Marcos
Century Bookstore	Lubbock
Compushop Inc.	Richardson
Computer Tech Associates	El Paso
Computerland Brownsville	Brownsville
Computerland Houston Bay	Houston
Computerland of Dallas	Dallas
Computerland of Tyler	Tyler
Computerland SO W Houston	Houston
Computers To Go	Austin
Congress Avenue Booksellers	Austin
Dabney's Bookland	Waco
Expensive Toys For Big Boys	San Antonio
Fultz News Agency	Denton
Homing Pigeon	Elgin
Lone Star News	Houston
M - 6 Distributing	Houston
Micro Age Computers	Hurst
R L Coles Computer Ctr.	San Antonio
Simtec	Dallas
Software Concepts	Dallas
Softwares	Richardson
Total Computer Store	Beaumont
Twig Bookshop	San Antonio
Walden Books	Plano
Walden Books	Richardson
Walden Books	Dallas
Walden Books	Marshall
Walden Books	Arlington
Walden Books	Hurst
Walden Books	Forth Worth
Walden Books	Houston
Walden Books	Baytown
Walden Books	Friendswood
Walden Books	San Antonio
Walden Books	Corpus Christi
Walden Books	Mcallen
Walden Books	Austin
Walden Books	Lubbock
Walden Books	Abilene
Walden Books	Midland
Walden Books	Odessa
Westheimer Newstand	Houston

UTAH

B Dalton Bookseller	Orem
B Dalton Bookseller	W Valley City
B Dalton Bookseller	Ogden
Personal/Business Comp. Inc.	Salt Lake City
Quality Technologies	Salt Lake City
The Software Hut	Salt Lake City
Walden Books	Salt Lake City

VIRGINIA

B Dalton Bookseller	Springfield
B Dalton Bookseller	Williamsburg
B Dalton Bookseller	Newport News
Walden Books	Fairfax
Walden Books	Mclean
Walden Books	Manassas
Walden Books	Springfield
Walden Books	Arlington
Walden Books	Alexandria
Walden Books	Charlottesville
Walden Books	Richmond

Walden Books	Portsmouth
Walden Books	Petersburg
Walden Books	Roanoke
Walden Books	Chesapeake
Walden Books	Virginia Beach
Walden Books	Norfolk
Walden Books	Newport News
Walden Books	Hampton

WASHINGTON

Alpha Computer Center	Richland
B Dalton Bookseller	Bellevue
B Dalton Bookseller	Overlake
B Dalton Bookseller	Lynnwood
B Dalton Bookseller	Seattle
B Dalton Bookseller	Bremerton
B Dalton Bookseller	Tacoma
B Dalton Bookseller	Chehalis
B Dalton Bookseller	Yakima
Butlers Mobile TV	Federal Way
City News	Bellevue
Computer Connection	Silerdale
Computer Craft	Ellensburg
Computerland Of Vancouver	Vancouver
Data Borne Computers	Renton
Elliot Bay Book Co.	Seattle
Kennewick Computer Co.	Kennewick
Micro West Dist.	Bellingham
Take A Byte	Kennewick
Walden Books	Bellevue
Walden Books	Lynnwood
Walden Books	Seattle
Walden Books	Everett
Walden Books	Tacoma
Walden Books	Olympia
Walden Books	Vancouver
Walden Books	Spokane

WEST VIRGINIA

B Dalton Bookseller	Beckley
Computer World	S Charleston
Computerland	Clarksburg
Computerland Parkersburg	Parkersburg
Computerland S Charleston	S Charleston
Computers Plus	S Charleston
Market Street News	Wheeling
Walden Books	Bluefield

WISCONSIN

B Dalton Bookseller	Racine
B Dalton Bookseller	Madison
B Dalton Bookseller	Green Bay
B Dalton Bookseller	Superior
Byte Shop Greenfield	Greenfield
Chalet Computers	Monroe
Magic Lantern	Madison
Readmore	La Crosse
Team Elect	Eau Claire
Walden Books	Brookfield
Walden Books	Greendale
Walden Books	Glendale
Walden Books	Milwaukee
Walden Books	Racine
Walden Books	Janesville
Walden Books	Madison
Walden Books	Green Bay
Walden Books	Eau Claire
Walden Books	Oshkosh
Walden Books	Fond Du Lac

WYOMING

Computer Concepts	Cheyenne

PUERTO RICO

B Dalton Bookseller	Carolina

CANADA

Micro West	North Vancouver
Micron	Toronto, Ontario
Minitronics	Vancouver

FOREIGN

Worldwide Media Service, Inc.	New York

MARKET/SIDE

Advertiser's Index

***SOFTSIDE* PUBLICATIONS**

Advertising Representatives
Sue Rowland, Bob Mackintosh (603) 673-0585

National Representative
Christopher Smith, Inter-Marketing Associates
(603) 827-3976

Compatible color . . . monitors

. . . for demanding graphics or text display.

For high resolution (560H X 240V) color graphics, you can't beat the Amdek Color-II Monitor. And if you're looking for economy, the Color-III Monitor with 260H X 300V resolution is a superb buy.

Both monitors feature RGB video input for computer controlled color . . . and Amdek's easy-to-install Digital Video Multiplexor (DVM) board permits interface with the most popular 80 column boards.

Just call, or write for full details.

- Color-II Monitor has RGB input and 560(H) X 240(V) line resolution.
- Color-III Monitor has RGB input and 260(H) X 300(V) line Resolution.
- Digital Video Multiplexor (DVM) assures color graphics interface with most popular 80 column boards, such as: Videx "Videoterm", Advanced Logic "Smart-term", M&R "SUP 'R' TERMINAL", Bit-3 Computer Corporation "FULL VIEW-80", and the "Doublevision" boards.

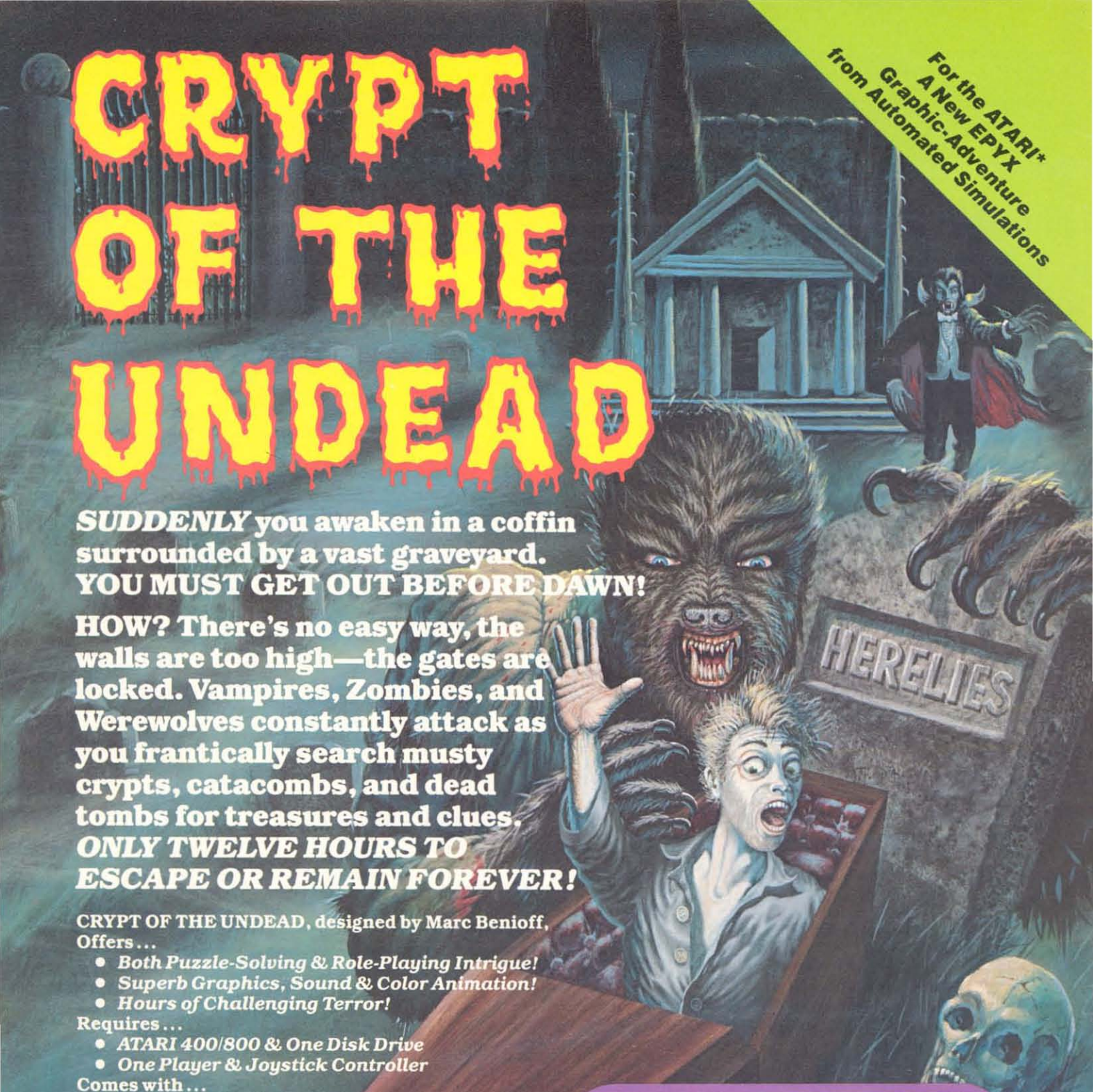
2201 Lively Blvd. • Elk Grove Village, IL 60007
(312) 364-1180 TLX: 25-4786

AMDEK CORP.

Amdek . . . your guide to innovative computing!

CRYPT OF THE UNDEAD

For the ATARI* A New EPYX Graphic-Adventure from Automated Simulations



SUDDENLY you awaken in a coffin surrounded by a vast graveyard. YOU MUST GET OUT BEFORE DAWN!

HOW? There's no easy way, the walls are too high—the gates are locked. Vampires, Zombies, and Werewolves constantly attack as you frantically search musty crypts, catacombs, and dead tombs for treasures and clues. *ONLY TWELVE HOURS TO ESCAPE OR REMAIN FOREVER!*

CRYPT OF THE UNDEAD, designed by Marc Benioff, Offers...

- *Both Puzzle-Solving & Role-Playing Intrigue!*
- *Superb Graphics, Sound & Color Animation!*
- *Hours of Challenging Terror!*

Requires...

- *ATARI 400/800 & One Disk Drive*
- *One Player & Joystick Controller*

Comes with...

- *Game Program & Complete Instructions*
- *EPYX 30/FOREVER WARRANTY*

Now Available At Your Favorite Dealer...$29.95
For the name of your nearest EPYX dealer write:
"CRYPT OF THE UNDEAD"
EPYX/Automated Simulations, Inc.
P.O. Box 4247, Mountain View, CA 94040

EPYX
COMPUTER GAMES THINKERS PLAY

EPYX Temple of Apshai was the very first computer game ever to win the Hobby Industry award for excellence. EPYX pledges you that same excellence in every game you purchase from us ... the VERY BEST in entertainment!

EPYX 30/FOREVER WARRANTY

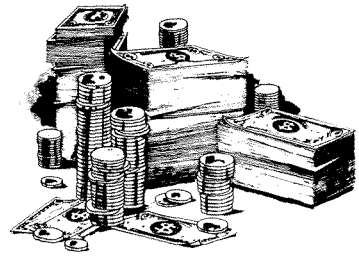
*Our 30-day Unconditional Guarantee: If your EPYX Game has any defect whatsoever within 30 days of purchase, return it to us or your dealer and we will replace it free.

*Our Forever Warranty: If anything happens to your disk at any time after 30 days, for any reason, just send it back with $5.00 for shipping, and we will send you a replacement.

*ATARI 400/800 is a trademark of ATARI, INC.

LIMBER UP YOUR COUNTING FINGERS!
***SoftSide* 37 will have as its focus:**

Personal Finance



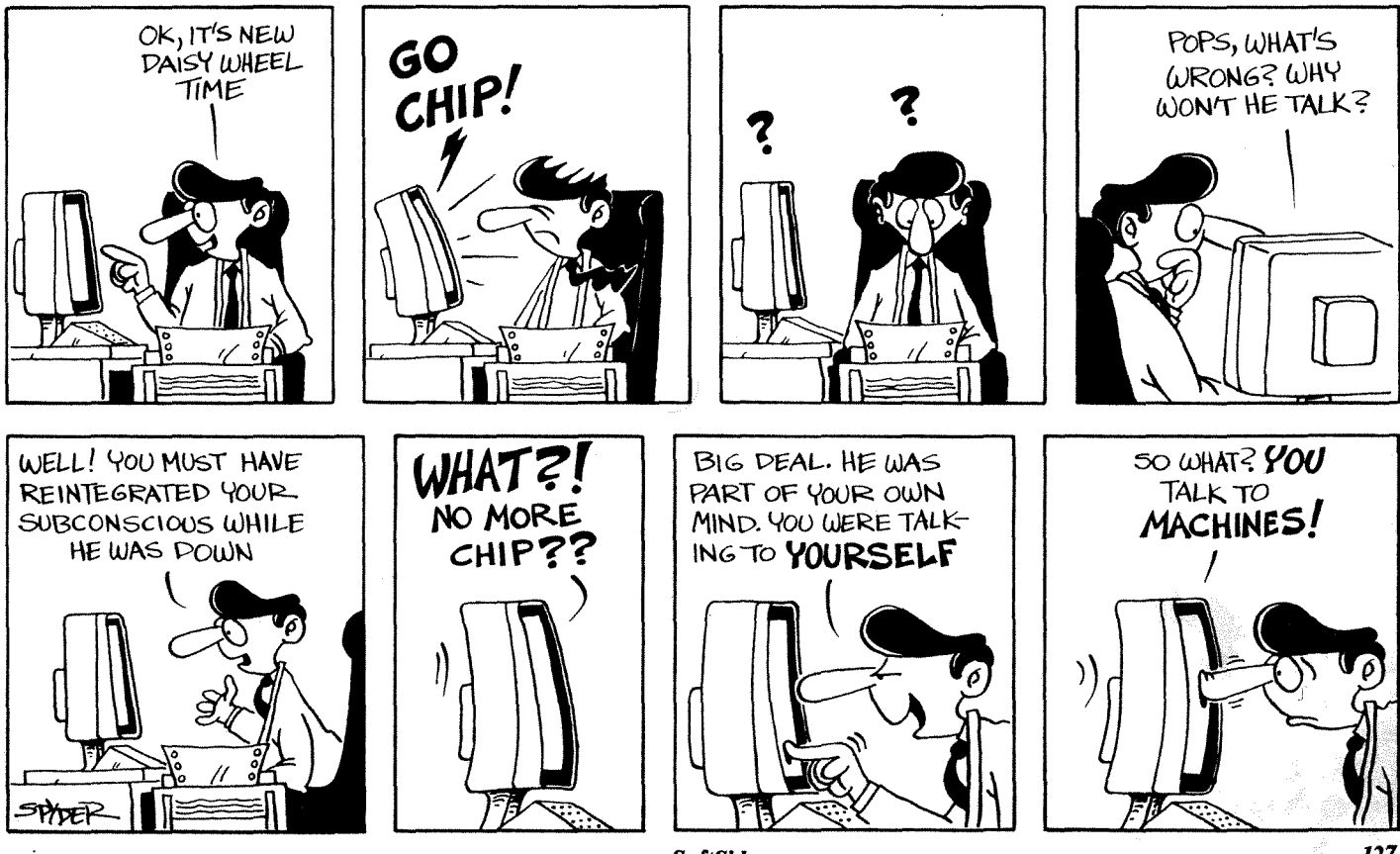
Few of us are capable of understanding the complex workings of accounting systems. At the same time, our personal financial matters would be much better under control if we could only manage our lives like a small business.

The next issue of ***SoftSide*** will offer some easy-to-understand solutions to this problem.

We'll have a Front Runner to help you keep track of where your money is going and reviews of some of the dynamic new personal financial management packages on the market. We'll even teach you how to use your word processor more effectively to let those creditors who've messed up your last three bills know about it.

Start polishing your green plastic visor — it's time to get your money matters under control!

MACHINE HEAD BY SPYDER



ANNOUNCING THE NEW ACCESS UNLIMITED MICRO SHOPPING CENTER

OVER $150,000 IN NEW GAMES & BUSINESS SOFTWARE

WE SUPPORT "TI", "ATARI", "COMMODORE", "IBM", "APPLE", "PERCOM", "WANG", "iBEX", "TRS-80", "EPSON", "OSBORNE"!

MANUFACTURER OVER-PRODUCTION

LESS THAN DISTRIBUTOR COST—BEST BUYS IN U.S.
"iBEX" 64K Business System/Word Processor
Model 2200
- ▶ Hi Resolution 9" Green Phosphor Screen.
- ▶ 2 Ea. 80KB Floppy Drives, Includes CP/M* Operating System Basic
- ▶ Basic Interpreter and "C'ITOH" Letter Quality Full Size Daisywheel Printer.

YOU SAVE $3408.00—Reg. Retail $6621.00—Sale Price: **$2960.00**

"iBEX" SYSTEM as above w/Billing Quality Printer instead—
Sale: $1938.00.

Model 7202 Reg. Price $9070 **Now Only $3995.**
Once in a lifetime. 64K Ram iBEX
12" Green Phosphor Screen with 2 8" Floppy Disk Drives
2.4 Megabyte Storage

Look what you get for $3995

- Z80
- CP/M compatible
- 64 kilobyte RAM, expandable to 192 kilobytes
- Dual 8" floppies (1.2 megabytes each) switchable to IBM format
- 12" green phosphor monitor (80 x 24 characters)
- Centronics compatible printer interface
- Serial interface
- Full function keyboard
- Clock timer and calendar (with battery)

NEW! 5¼" SOFT SECTOR DISKETTES W/Hub Ring

"BASF" 5¼" Lifetime Limited Warranty **$24.90** SSDD Box of 10

ATHANA 1 Yr. Limited Warranty **$19.90** SSSD Box of 10

8" Diskettes From $22.75

OTHER DISKETTES AT EQUAL SAVINGS

Flip 'N File Boxes—Holds 50 Diskettes		
5¼"	Reg. $34.95	**Sale $19.95**
8"	Reg. $49.95	**Sale $28.95**

LIBRARY CASES—Holds 10 Diskettes. Sale **$2.95** ea.

"IBM PC" COMPUTERS IN STOCK
Bare Drives (without Controller) For "IBM PC"
—Internal or External—**$269.95 ea.**

OVER $1,000,000 IN NEW COMPUTERS, PERIPHERALS AND SUPPLIES!
NEW PRINTERS IN STOCK: "CENTRONICS", "C'ITOH", "OKIDATA", "STAR", "TALLY", "RICOH", "SMITH-CORONA TP", "EPSON".
VALUES TO $3500.00
SALE: $345 to $1995

Call for pricing and fast delivery on other hard-to-get micro peripherals and software. **1-800/527-3475**
Top 100 Software programs normally maintained in stock and sold at our special Access Unlimited Micro Shopping Center prices.

" " Reg. Trademarks

☐ Please send me a FREE catalog. I'm not ready to order at this time.
☐ YES, I'm taking advantage of your Sales prices.

Name _____
Company Name _____
Address _____
City _____ State _____ Zip _____
Phone Number (_____) _____

Quantity	Item	Unit Price	Subtotal

Subtotal
State Sales Tax (Texas residents only)
handling charge
Total

Check one:
☐ payment enclosed ☐ Visa ☐ MasterCard*

*If MasterCard, numbers above name:

Expiration Date: ☐☐ — ☐☐

Authorized signature, if charged

ACCESS UNLIMITED

DEPT. D1/401 N. Central Expwy. #600/Richardson, Texas 75080
Tel. 1-800/527-3475 214/340-5366
214/690-0207 Sat. and Evenings Only

Explore the Frontiers of Intelligence

THE MIND OF MAN

Chess, **Checkers**, and **Odin** (as played by U.S. Othello Association rules) are classic games of the intellect. They evolved over the centuries as a way to understand complex situations and achieve mastery in action.

Now, by interacting with the intelligence embodied in these programs, you can participate in the exciting challenge of the **Mind of Man.**

FOR ALL GENERATIONS— A NEW GENERATION OF INTERACTIVE INTELLIGENCE

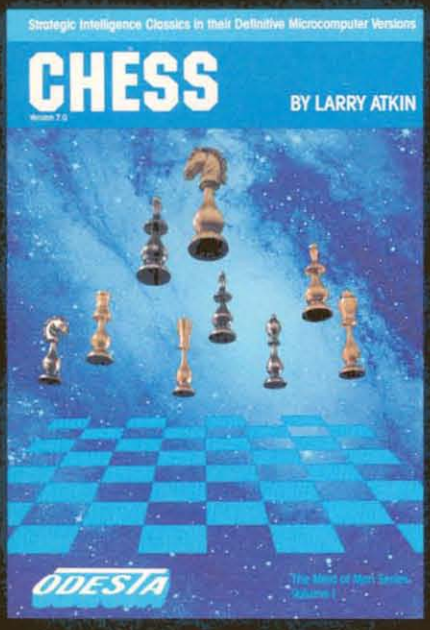
Chess, **Checkers**, and **Odin** are unique—in playing strength and in what they let **you** do. You can try out any idea—and even get ideas from the programs themselves. Plus, it is easy and fun to play the games and use their many features:

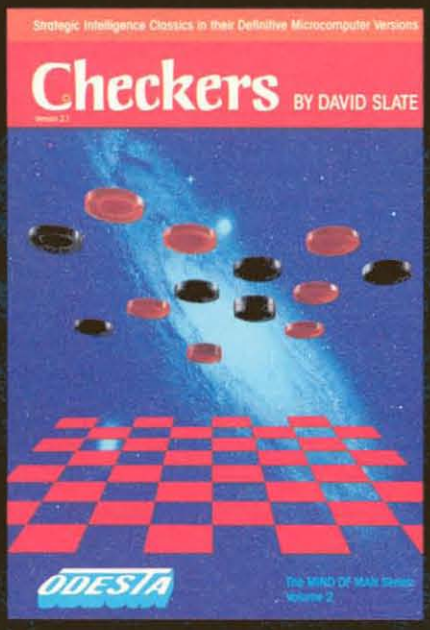
• Different levels of play, from beginner to expert • Advice on best move • Take back and replay moves • Auto and manual modes • Instant replay of games • "Change" feature adds or subtracts pieces • Sophisticated opening libraries • Clear graphics • Instructions include chapter on skillful play.

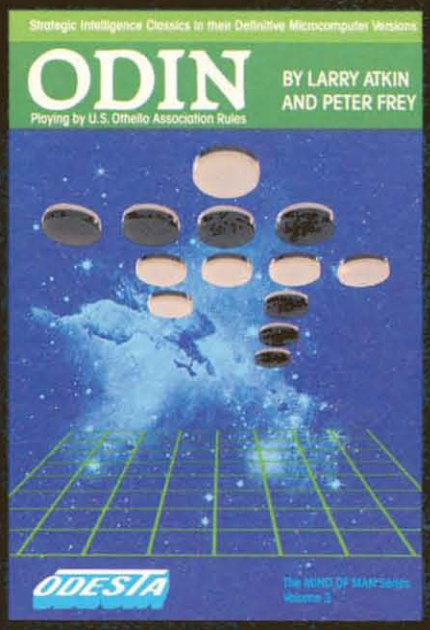
THE PEOPLE BEHIND THE PROGRAMS

LARRY ATKIN AND DAVID SLATE
Authors of the Northwestern University 4.7 Computer Chess program; Winners of the World Computer Chess Tournament, 1977-1980; Winners of 8 North American Computer Chess Championships between 1970 and 1979; Two of the world's authorities on machine intelligence.

PETER FREY
Professor at Northwestern University, teaching courses in Psychology and Computer Science. Editor of and contributor to the definitive text on computer chess: **Chess Skill in Man and Machine.** One of the U.S. Othello Association's top-ranked players.







CHALLENGES THE PROFICIENT; INSTRUCTS THE BEGINNER.

A new microcomputer standard for what many consider to be the ultimate game of the intellect. In addition to its superior playing strength, **Chess** from Odesta introduces a new generation of interactive intelligence—with 27 cursor-controlled features, including:

• Advice and prediction of best moves • Save games to disk • Graphic illustration of attacks and defenses • Variations of blind-fold chess • 17 levels of play, including postal and mate-finder modes • Enactment of over 30 classic human and computer chess games • Plus—an opening library of over 7000 moves.

For those who want the best.

YOU'VE NEVER KNOWN CHECKERS LIKE THIS!

Learn the complexities of this surprisingly sophisticated strategy game by interacting with **Checkers'** 24 user features:

• Play against 16 levels of difficulty • Watch **Checkers** play against itself—one level against another • Switch to "Give-away" mode, where the object is to make your opponent take **your** pieces • Watch the **Checkers** movie—an instant replay of a whole game • For those interested in the inner-workings of "programs that think", adjust 58 program parameters, so that **you** can experiment with the way **Checkers** itself thinks, and how it plays.

An ideal introduction to artificial intelligence.

WHY ARE 20 MILLION PEOPLE PLAYING THIS GAME?

A classic board game, where the object is to entrap your opponent's pieces—but only at the right time. The rules of play are simple and the game fast. At your disposal are the full range of features found in all of Odesta's **Mind of Man** series. You will need them, along with your keenest insight and deepest perception, to master the secret of **Odin** (playing by U.S. Othello Association rules). Join the growing number of strategists exploring the subtleties of a game that may change the way you think.

ODESTA

930 Pitner
Evanston, IL 60202
(U.S.A.)

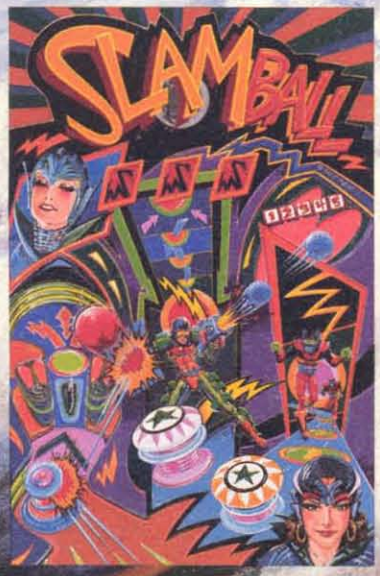
Chess: $69.95
Checkers: $49.95
Odin: $49.95

See your local software dealer, or order (Mastercard or Visa):
800-323-5423
(in Illinois, call 312-328-7101)

For **Apple II, Apple II Plus** 48K disk systems, and **Atari** 48K disk systems. Odin is also available for **TRS-80 Model 1 & 3** 32K disk systems.

Why let some good times slip through your fingers?

Corner the latest Synapse Software games at your local computer store.



Available in disk, cassette, and cartridge for the Atari 400/800 computers.

Other titles soon available for the VIC 64, IBM-PC and the TI 99/4.

synapse
SOFTWARE

5327 Jacuzzi St., Suite J, Richmond, CA 94804