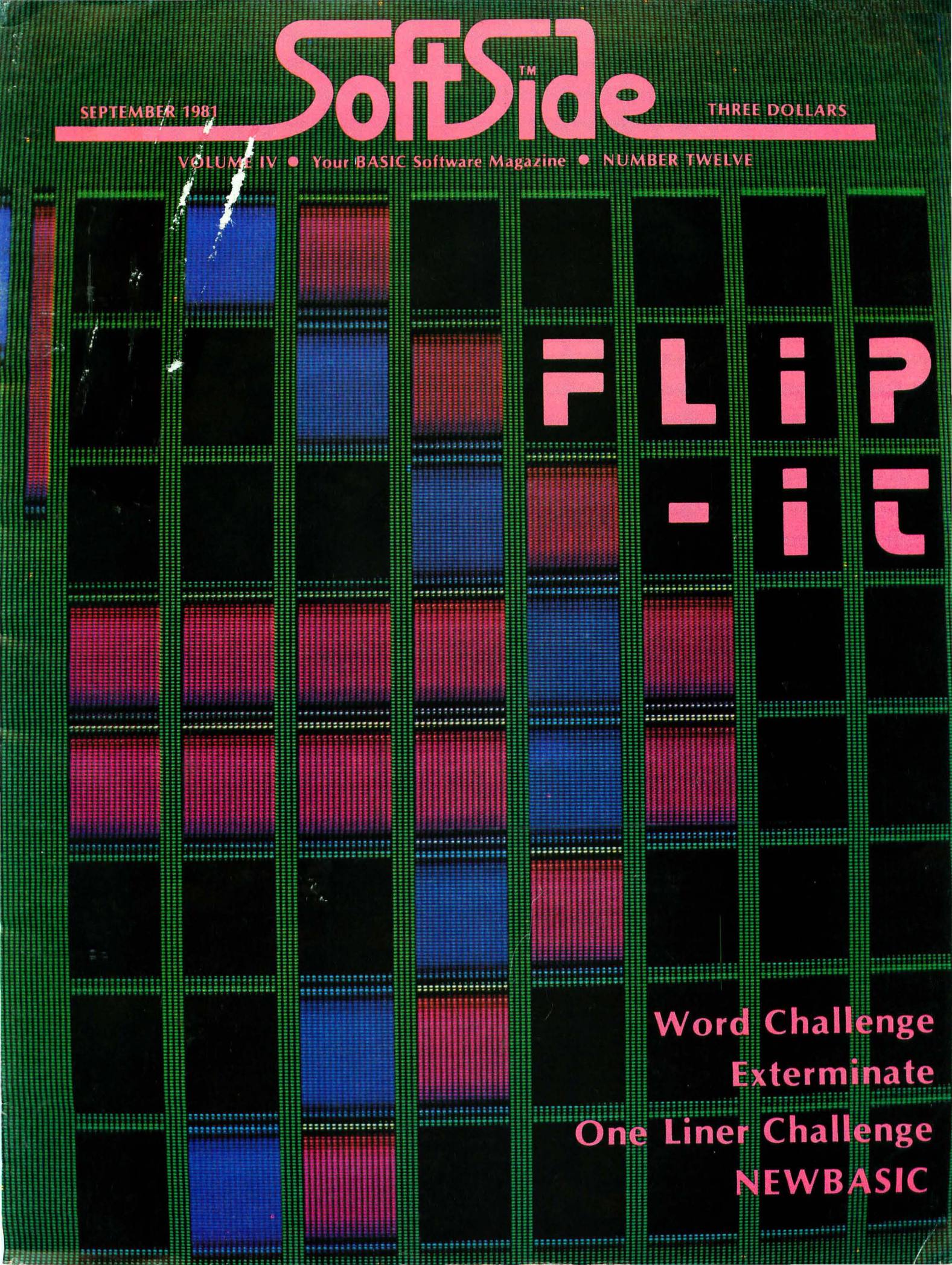
SEPTEMBER 1981

SoftSide™

THREE DOLLARS

VOLUME IV • Your BASIC Software Magazine • NUMBER TWELVE



FLiP-iT

Word Challenge

Exterminate

One Liner Challenge

NEWBASIC

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.

Complete programs of every conceivable type, ongoing data bases, modified languages and I-$tring, the video simulator — all of these are part of **SoftSide DV**. Feel as though you're missing something? You are! But, you needn't any more.

Introductory offer — if you subscribe to **S-80 SoftSide DV** before September 30 at the regular price of $125 per year ($10.42 per issue), you'll receive the September 1981 disk, including NEWBASIC, absolutely FREE! Or, if you'd like to look us over, try the September issue only, for $12.95

For your convenience, we offer an installment payment plan for Visa and Mastercard holders: You pay only $32.50 per quarter (includes a $5 billing charge). For orders outside the USA, please add $18. If you currently receive **SoftSide**, we'll credit the remainder of your subscription to your **SoftSide DV** order.

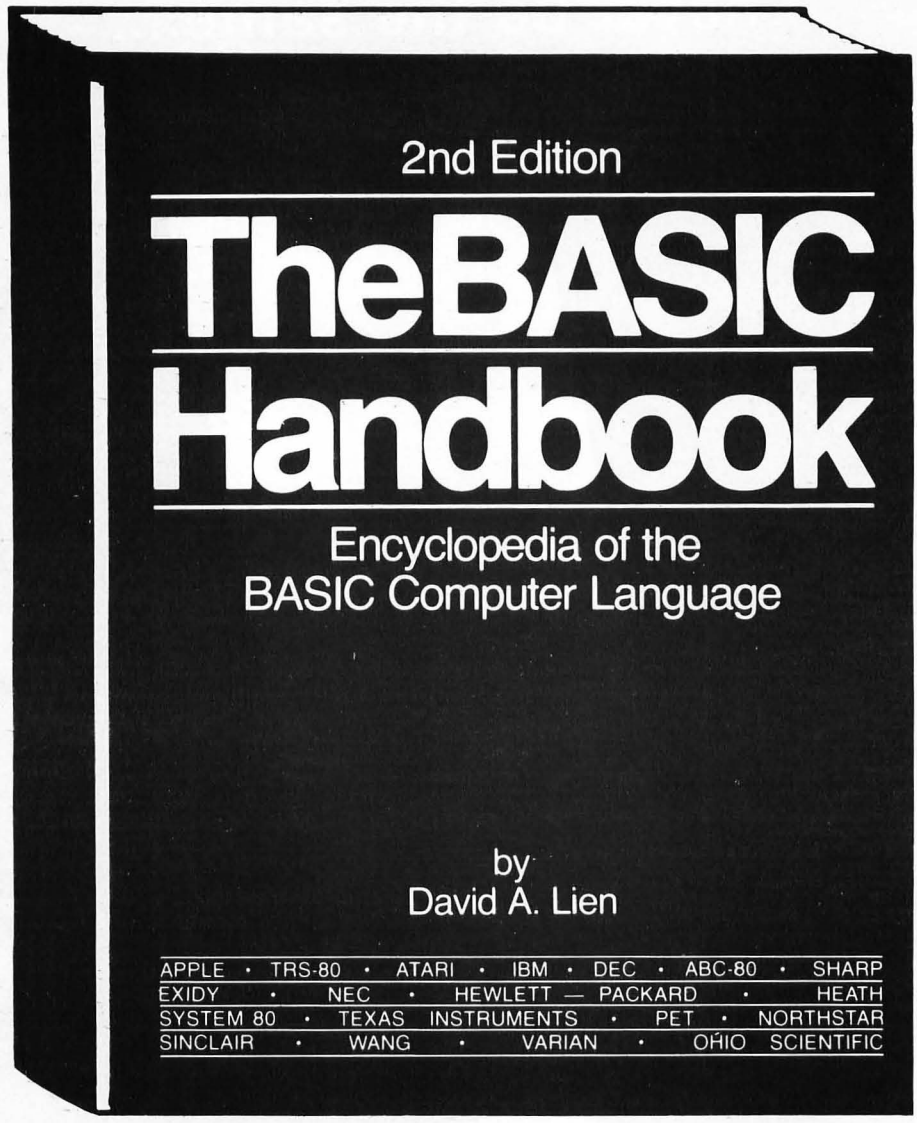
We're developing **SoftSide DV** for one system at a time. The S-80 version is currently available with the Atari and Apple editions soon to follow. In the meantime, new subscribers to the disk version for Atari and Apple will receive a $10 **SoftSide Selections** gift certificate in addition to the monthly magazine and disk containing BASIC program listings.

SoftSide Selections

6 South Street Milford NH 03055

For Orders Only 603-673-0585

Expand Your Computer's Vocabulary!



The BASIC Handbook has never been this complete.

The Expanded Second Edition gives you over twice as much information as the First Edition, explaining nearly 500 BASIC words. The handbook features special sections on Disk BASIC, TRS-80 Extended Color BASIC, Atari BASIC, Acorn Atom BASIC, Tektronix BASIC and "Converting Programs From One Computer For Another."

The computer industry has experienced tremendous change in the last three years. hundreds of new computers have been introduced since **The BASIC Handbook** was released in 1978. The Second Edition meets the challenge head-on, documenting every significant BASIC word used by every BASIC-speaking computer.

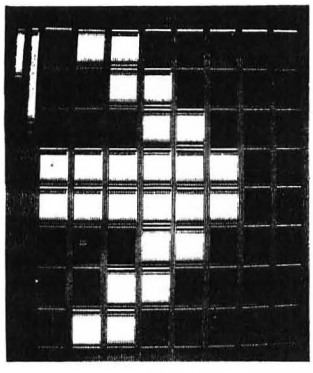
This new Edition makes program conversion easy. Its widely acclaimed feature, "If Your Computer Doesn't Have It" has been expanded. Each BASIC word is alphabetically listed, with **Test Programs** and **Sample Runs. Variations in Usage** combine with **Alternate Spellings** to totally cross-reference each BASIC word.

INTRODUCTORY SPECIAL — 10% OFF!
The BASIC Handbook by David Lien
(Reg. $19.95) $17.95

(Please allow six weeks for delivery)

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585

SoftSide™



This month's cover photo features the Flip-It program as running on an Atari 800 with a Sony monitor. The design was accomplished through the "Set up your own game?" feature of the program. Photo by Mary Locke.

SoftSide is published each month by **SoftSide** Publications, 6 South Street, Milford, New Hampshire 03055. Telephone 603-673-0585. Second class postage paid Milford, New Hampshire and pending at additional mailing offices. ISSN: 0274-8630. Subscription rates: USA, $30.00/12 issues. USA First Class APO, FPO, Canada, Mexico, $40.00/12 issues. Overseas air mail: $62.00/12 issues. Media subscription rates: Magazine and cassette, $75.00/12 months. Magazine and disk, $125.00/12 months. APO, FPO, Canada, Mexico, (add), $36.00/12 months. All remittances must be in U.S. funds. Mail subscription inquiries to **SoftSide** Publications, P.O. Box 68, Milford, New Hampshire 03055.

POSTMASTER:
Send address changes to:
SoftSide Publications
6 South Street
Milford, New Hampshire 03055

ARTICLES

S-80, APPLE, AND ATARI PROGRAMS

S-80 PROGRAM

APPLE PROGRAM

DEPARTMENTS

About This Issue

The color abounds in New England and on the cover of **SoftSide** this time of year. The munchkins are on their way back to school so we won't be hearing from them for a while.

Our staff has been turning somersaults over "Flip-It," a board game to while away your Autumn hours. The Apple original was created by Michael Prescott with S-80 and Atari translations by Alan J. Zett. We doubt you'll ever see the arrow we featured on this month's cover, but we thought we'd point you in the right direction.

For those of you working on your vocabulary before returning to school, we present "Word Challenge." Elrhea Bigham has taken the traditional "Hangman" and added a few new twists to tickle your fancy. That multi-talented Alan Zett translated this one for the Apple and S-80. Maybe we should send him to the UN for Micros!

Robert White ventured out into Micheal O'Keefe's foreign terrain to try his hand at "Orienteering at Jacque's Coulee" (**SoftSide**, May, 1981) and won our translation contest. You Apple owners can get out your compasses and hopefully find your way back in time for the October issue.

If any of you are arthrophobics, we'd suggest you stay away from "Exterminate," an S-80 program by George Geczy guaranteed to be full of bugs. Don't worry, they're supposed to be there. Just be sure you stomp on enough of them to keep from being overcome!

Our One-Liner's go on and on and Bob Howell has given us all a few hints in "The One-Liner Challenge." Bob worked on an Atari, but his techniques can easily be applied to other systems. Read it carefully, check out his six examples and snow us under with your efforts.

John T. Phillipp, M.D. closes his series with "VARPTR, Again," packing machine language subroutines into strings. Jon Voskuil has discovered a technique for mixing text and graphics on page two of the Apple Hi-Res graphics mode. We begin publishing documentation for the **S-80 SoftSide DV** with NEWBASIC, an enhanced BASIC adding commands beyond Level II and Disk BASIC.

Plus, all of the regular features you know and love: One-Liners, K-Byters, Reviews, Programming Hints, The Sensuous Programmer and those awful Bugs, Worms and other Undesirables.

That's all for this month. Next month is our third anniversary and we plan to celebrate it in grand style. I-$tring, the video simulator, will be premiered along with lots of other surprises. Happy hacking!

Say Yoho

by Scott Adams

This month I'll talk about some of my experiences with one of the newer machines, the Atari 800.

Back in October of 1980, I decided it was time to start converting my Adventure series over to the Atari. At this time, disk drives were just coming on the scene along with the new Atari RS232 interfaces, so I figured that the time was right.

I had Adventure running on an Apple II. Since the Apple and Atari both used a 6502 processor as their main chip, I knew this would be the place to start.

My Adventures are all written in assembly using a common interpreter. Once I have an interpreter running on a specific machine, I can then transfer each Adventure data base over. Voila, I then have the entire series done! On the S-80, the interpreter is approximately 6K of code with the same 10K of data base. Eventually I would end up needing 22K of memory on the Atari, but I'm getting ahead of myself.

My first step then was to get the

Atari and the Apple "talking" to each other over the RS232 links. On the Apple there are many choices for RS232 ports, put out by many companies. The one I chose, was the AIO Board by SSM in California. I had previously used this board in getting data bases from the S-80 Model I to the Apple II and I felt comfortable with its performance.

I hooked up the Atari 850 with the RS232 to the Atari disk system and started playing with it trying to transfer just some simple files from the Apple. Immediately I ran into trouble! I found I could get the Atari to read from the Apple only as long as the disk drives were shut off on the Atari!

As Adventure is a fairly long program, I dreaded the thought of having to use the cassette tape drive for all of my assembly and source loads.

Calling the Atari Technical Service Group in California, I spoke for quite a while with their support people who were extremely courteous and knowledgable. Atari has set up this special group just to help outside software houses use their computers better (a tactic that both Tandy and Apple would do well to emulate).

The problem turned out to be in the DOS I was using and next month we will look at the solution. Also, a look at why I used an S-80 Model II in converting Adventures from the Apple II to Atari. Til next month. . . Yo ho. .

ADVENTURE INTERNATIONAL

NEW

THE Adventure Continues

The Scott Adams' Adventure series announces the release of Adventure #11 *"Savage Island,* part 2" and Adventure #12 *"Golden Voyage".*

For The • TRS-80 • APPLE • ATARI • $19.95 tape

Also available on Bonus Triple Pack Disk Adventures #10, #11, and #12.
For the TRS-80 and Apple $39.95

2 maces & magic SERIES

NEW

STONE of Sisyphus

The Stone of Sisyphus carries you to the "thinking man's" dungeon. A subterranean world of monsters, magic, traps and treasures demands brains rather than luck to survive (of course, you might take along your mace just in case. . .).

MACES & MAGIC are fantasy adventures involving you and your computer. Armed only with your wits, a microcomputer, and the software provided, you can become the hero or the meal your destiny dictates. You create a character, equip him (or her) with suitable weapons and armor, and enter the dungeon in search of fame and fortune. Neither is particularly easy to obtain.

If you are successful in avoiding or conquering the various monsters, traps, enchantments and illusions set by our nefarious dungeonmasters, you may escape with riches and glory. Your name and deeds will be recorded for posterity in the records of the dungeon. More importantly, you'll be alive. You may then use the same character in his more experienced and wealthy form when you enter dungeons on later occasions.

The object of the whole exercise is not just to fight the monsters and collect treasure. You have to get out alive to enjoy it. In every dungeon there is at least one exit. It is possible to escape from each and every dungeon with a whole skin. We state that fact here because players often believe this not to be true. We really aren't out to get you. Not really.....

TWO FULL DISKS WORTH OF DATA!!

A compacted data structure makes it possible to run this fantasy on a single disk drive, though 2 full diskettes worth of data are supplied. Expanded use of free-form input puts your ingenuity to an even greater test. The responsiveness of the programs to the individual qualities of the character make this adventure frustratingly enjoyable for hundreds of hours before its secrets can be unlocked.

SCOTT ADAMS SEAL OF APPROVAL

FOR ALL OF THE BELOW
Apple 2 Disk with Applesoft
or Apple 2 Plus 48K
TRS-80 32K Model 1 or
TRS-80 48K Model 3

032-0100
$29.95

A Division of Scott Adams, Inc.
Adventure International • Box 3435 • Longwood, FL 32750
TOLL-FREE ORDER NUMBER: (800) 327-7172
IN FLORIDA (305) 862-6917

SHIPPING AND HANDLING CHARGES ARE EXTRA
PRICES ARE SUBJECT TO CHANGE WITHOUT NOTICE

ai

Connect with ATARI

ATARI HARDWARE

ATARI 400 Computer System, 16K RAM	$369.00	(#36-401)
ATARI 800 Computer System, 16K RAM	$799.00	(#36-800)
16K RAM Module for the ATARI 800	$99.00	(#36-854)
32K RAM Module for the ATARI 800	$139.00	(#36-855)
48K RAM Module for the ATARI 400	$259.00	(#36-856)
ATARI 410 Program Recorder	$89.95	(#36-410)
ATARI 810 Disk Drive	$489.00	(#36-810)
ATARI 822 Thermal Printer	$369.00	(#36-822)
ATARI 825 Printer	$819.00	(#36-825)
ATARI 830 Acoustic Modem	$169.00	(#36-830)
ATARI 850 Interface	$179.00	(#36-850)
MACROTRONICS Printer Interface (36-pin)	$69.95	(#36-936)
MACROTRONICS Printer Interface (40-pin)	$69.95	(#36-940)
VERSAWRITER	$299.00	(#36-1100)
ATARI Joystick Controllers	$19.95	(#36-4004)
LE STICK	$39.95	(#36-3006)
ATARI Paddle Controllers	$19.95	(#36-3004)
Dust Cover for ATARI 400	$8.95	(#36-016400)
Dust Cover for ATARI 800	$8.95	(#36-016800)

ROM Programs

Music Composer	$59.95	(#36-223007)
TeleLink 1	$24.95	(#36-223015)
Asteroids	$39.95	(#36-223013)
Basketball	$29.95	(#36-223004)
Computer Chess	$39.95	(#36-223009)
Missle Command	$39.95	(#36-223012)
Star Raiders	$59.95	(#36-223011)
Super Breakout	$39.95	(#36-223006)
3-D Tic-Tac-Toe	$29.95	(#36-223010)
Video Easel	$29.95	(#36-223005)
Assembler Editor	$59.95	(#36-223003)
PILOT (With "Turtle Graphics")	$89.95	(#36-223405)
ATARI BASIC for ATARI 400	$69.95	(#36-223483)

SOFTWARE on Disk

VISICALC from Personal Software	$199.95	(#36-204010D)
TEXTWIZARD from DataSoft	$99.95	(#36-279010D)
Mailing List from DataSoft	$24.95	(#36-279002D)
Personal Financial Management System	$74.95	(#36-223406D)
Dow Jones Investment Evaluator	$99.95	(#36-223412D)
Microsoft BASIC	$89.95	(#36-223413D)

SOFTWARE on Cassette

3-Dimensional Graphics Package	$29.95	(#36-212001T)
Star Trek 3.5	$19.95	(#36-200025T)
Scram	$19.95	(#36-223123T)
Space Invaders	$19.95	(#36-223111T)
B-1 Nuclear Bomber	$15.00	(#51-237004T)
Conflict 2500	$15.00	(#51-237001T)
Empire of the Overmind	$30.00	(#51-237011T)
Lords of Karma	$20.00	(#51-237003T)
Midway Campaign	$15.00	(#51-237005T)
North Atlantic Convoy Raider	$15.00	(#51-237006T)
Nukewar	$15.00	(#51-237007T)
Planet Miners	$15.00	(#51-237008T)
Stocks and Bonds	$15.00	(#51-237009T)
Tanktics	$24.00	(#51-237012T)

TERMS: Prices and specifications are subject to change. **TSE HARDSIDE** accepts VISA & MASTERCARD, Certified checks and Money Orders. Personal checks accepted (takes 3 weeks to clear). **TSE HARDSIDE pays all shipping charges (within the 48 states) on all PREPAID orders over $100.00.** On all orders under $100 a $2.50 handling charge must be added. COD orders accepted (orders over $250 require 25%deposit), there is a $5.00 handling charge. UPS Blue Label and Air Freight available at extra cost.

ATARI®
A Warner Communications Company

MasterCard VISA

TSE:HARDSIDE
6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

Editorial

by Jon Voskuil

There's a menace abroad in the land. Faster than a MIG fighter; more powerful than television propoganda; able to leap social classes in a single bound: This newest Menace threatens the very fabric of our society, and will leave no one untouched. Compared to this, all the other Menaces in the world today are impotent and inconsequential. It is the Menace of MICROPHILIA.

"Microphilia," of course, is the love of microcomputers. Those who have been duped by this insidious movement are "microphiles," and those who are beginning to oppose the onslaught of the Menace are "microphobes," united together in the common causes of human decency and the fear of computers. The opposing sides are facing off against each other, ready to fight to the last man and CPU chip for their respective beliefs. Brothers and sisters, husbands and wives, parents and children, are skirmishing with one another over the most basic of human rights: Who gets to use the color TV set/video monitor during prime time?

It's odd — I find myself caught in the middle of the fray. Since childhood I've been fascinated by electronic and mechanical gadgets, and microcomputers are for me a fantasy come true. Years ago I spent untold hours and the hard-earned money of a paperboy to make a binary adder out of transistor flip-flop circuits. I never did get the darn thing to work properly, in spite of rebuilding it several times. It always came up with 1 + 1 = 11 (binary). Some years later I found a crazy little book on building a working digital computer out of paper clips, tin cans, and assorted other household goodies (no kidding!) and spent more uncounted hours working on this (very clever) simulation/model of a real digital computer. Eventually, however, that project bombed too.

So I'm prime material for a microphile. After all those years of frustration, I finally found my summum bonum at a computer store a couple of years ago, and my leisure time hasn't been the same since.

Yet I have sympathies with the microphobes, too, for at least two reasons. First, the computerization of vast amounts of information presents a real (not just imagined) threat to security and privacy. Even microcomputer systems have the potential for adding to this danger, as they will become increasingly able to access large time-sharing systems with enormous data-access capabilities. Anyone clever enough to devise a security system to "guarantee" the privacy of certain data will certainly meet his counterpart in the person who is clever enough to defeat that system.

The second reason is more troubling to me than the first, and more directly and immediately affects you and me as microcomputer users. In broad terms, the problem is one of advancing technology leaving human beings out in the cold. We're increasingly aware of the vocational dimension of this problem, with workers being replaced by robots and other microprocessor-based innovations.

But there's a more personal, domestic dimension as well, and that's what concerns me at the moment. We who are the proud owners of home computers, and for whom they may be the fulfillment of lifelong fantasies, need to be careful with our priorities. Because of the practically endless variety of things that computers can do, some of us find it too easy to involve ourselves with them to the exclusion of other interests and — most importantly — other people. We make lots of jokes about computer widows and about computer addicts, but let's not forget that there is a lot of potential for serious problems in precisely those areas.

Although I approach computers as one who is an irredeemable gadgeteer, I approach them also as one who believes strongly in the worth of human beings and the importance of interpersonal relationships. There are times when the former overshadows the latter — in friendships and in marriage and family life — and I have to be reminded of the priorities that are built into the universe. People are immeasurably more important than things, however ingenious the things may be.

Input

Dear **SoftSide**,

I would enjoy it very much if you were to publish some Z-80 Machine Language programs for the S-80. I am just getting started in Machine Language programming, and I don't want to get it from a book alone. It would be nice to have it every month. I could make a couple of games, and I have a friend who is working on a big space game. I am sure that there are others who feel the same way.

About the covers, I don't see what is wrong. Mine comes through the mail in perfect condition. The only thing wrong with them is that the address sticker comes on the front. I think someone wrote to you about this before, and you said that you would try to fix it.

At first I couldn't stand that there were two other computers in my **S-80** magazine. And then in Dave Albert's Outgoing Mail, in the June, 1981, issue of your magazine, he put "primitive" S-80. I think that all of the people in **SoftSide**, including the readers, should agree not to discriminate against any computers, unless they have a good reason to. The only reason he had was to try to say why **SoftSide** gets more S-80 programs. I would not call a system "primitive" just because there are more programs written for it. I very much doubt that if a computer is "primitive" that there would be more programs written for it.

John Lauro
Owosso, MI

Editor's Reply: With the expanded SoftSide S-80 Disk Version we now have a place for Machine Language programs. Let's see some submissions!

Dear **SoftSide**,

Regarding all the complaints you've received about issues being torn up etc. before they reach the subscribers, all of my issues arrive in perfect condition most of the time. I think that the complaints and the responsibility should not be put upon **SoftSide**, but at the local postal service. Why should **SoftSide** have to put covers on its magazines when most other magazines don't? I suggest that anyone who receives a damaged issue direct their complaints at the postal service and not at **SoftSide**.

Also regarding a letter in the June issue, how could you print a letter from someone who refers to the TRS-80 as the TRaSh-80. It seems that TRaSh-80 has become the accepted term used by Apple users. Anyone who wants their letter printed shouldn't use derogatory terms for other computers. When you print a letter like this, you're promoting bad feelings between Apple users and S-80 users. It's going to be hard putting out a magazine for three different groups that hate each others' guts. I'm sick of reading letters from readers who like the old format better because they can't stand looking through other material for another computer. Most of these readers feel this way because they are afraid that their computer is inferior to the other.

All in all I think that the new format is just wonderful. Don't change it because there are many of us who don't mind looking through other material and appreciate the effort you put into this magazine.

Andre Chen
Potomac, MD

Dear **SoftSide**,

I am generally happy with **SoftSide** and always look forward to each new issue, but when I received the June issue last week, I was very annoyed by the condition it arrived in. The cover and the first couple of pages were torn off and the middle section was missing. I know many of your readers have already written to you about this very problem. I have a suggestion and hope you will seriously consider it. Mail the magazines out in the brown paper wrappers many other magazines use. If the cost is the reason why this method hasn't been tried before, I'm sure most SoftSide readers would pay a little more ($1 more per year) just to receive the magazine intact.

Finally, to all the Apple II+ owners who had trouble running the "Moonlanding" (10/80) program. The following changes should solve the problem.

```
Del 1000
Add 1037 HGR
Change 1980 J = ABS (J) : HPLOT J,B1:POKE
7912+J,B1:RETURN
```

And if have not done so:

```
Add 5006 RETURN
```

S. Wong
Brooklyn, NY

Editor's Reply: It's the Post Office, not SoftSide that delivers your magazine in less than perfect condition. When we can find a viable mailing alternative that can satisfy both the need to get your magazine to you in tip-top shape and the need to keep the costs (ours and yours) down, then we will give it serious consideration.

Dear **SoftSide**,

How do I rate a magazine? Well, if by the time I'm finished reading it, the cover is gone, that's a darn good magazine. Anyway, I just received TWO issues of your October issue and BOTH of the magazines' covers are gone. I have hence subscribed.

Another thing, I'm in total agreement with Larry Eiss in the June issue.

Steve Harris
Gladstone, MO

Dear **SoftSide**,

I am an Apple owner who believes your magazine is one of the best. I too, like many subscribers, believe we should have separate magazines for owners of different computers. I know this is a big request, but I suggest a different plan such as making translations for each program you print each month. This is also cumbersome and would probably raise the price of your fine magazine. A final solution for **SoftSide** would be to have the readers vote on a plan which the majority of subscribers would like. You (**SoftSide**) can do what you feel is appropriate about this issue.

David Buckley
Littleton, CO

Editor's Reply: We plan to continue our format of offering programs and articles of interest to users of all three systems that we support.

Dear **SoftSide**,

This letter is to inform you of a new computer club I am forming in Rhode Island. The National Apple Newsletter will be for owners and users of Apple II or Apple II+ computer systems.

The club will print anything our members demand from Pascal programs to Home-Control, computerized alarm systems, and other practical applications. Member input and questions will be printed in the newsletter also. We hope to hold monthly meetings in members' homes. Club dues have not yet been set as we must first find out how many people would be interested in the club. We need ambitious, eager people to help organize this club and welcome all inquiries.

Scott Summer
The National Apple Newsletter
27 Leicester Way
Pawtucket, RI 02860

Dear **SoftSide**,

I enjoy your magazine and find the Apple programs refreshing. However, I hate more than anything to spend time typing in the programs and then not have them run because of program errors - not syntax errors.

I think your programs that you print should be thoroughly tested. The "Battle at Sea" in the April, 1981, issue was excellent. Then I type in "Galaxia" in the May issue. What a mess. So many values are exceeding the graphics limits (0 to 279) that I gave up trying to fix them. The more common default lines are 250, 390, and 450. In 250, if x (from line 230) is 16 and N2 is anything but 0, then ERROR.

Anyway, I would appreciate you error checking your programs.

Montgomery A. Lee
A.P.O. Miami, FL

continued on page 10

SoftSide™ INPUT POLICY

SoftSide Magazine welcomes your comments and thoughts on both the magazine and the field of microcomputing. We try to publish as many of our readers' letters each issue as we can.

For the sake of clarity and legibility, all letters should be typwritten and double-spaced. Send your letters to:

SoftSide Publications,
Input
6 South St.,
Milford, N.H. 03055

We reserve the right to edit any letters prior to publication.

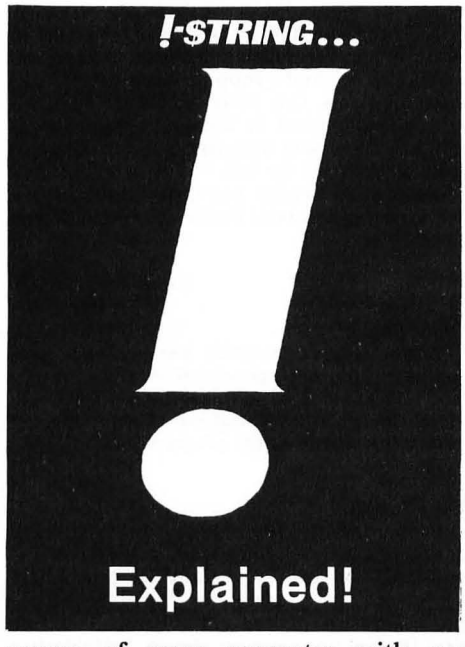
Outgoing Mail

by Randal L. Kottwitz

Greetings! The excitement is almost unbearable here in the halls of **SoftSide** as multitudes of projects which have been in the planning stages for months are coming to fruition. With this issue, we are initiating the accompanying S-80 Disk Version. As I've talked to a few of you in my first week as editor, I've sensed a bit of fear that **SoftSide**, as you know it, will be no more. As you can see from this issue, all of the BASIC program listings, programming hints, one-liners and reviews remain. We simply now have a method of speaking to those of you who want more. Let's face it, we're in the middle of an information revolution and we at **SoftSide** would be guilty of playing the proverbial ostrich with our combined head stuck in the sand were we not to step into the forefront and create the magazine of the future, today. Ultimately, I'm sure we will look back and see this as only an elementary step into a vast new method of information distribution.

The third anniversary issue of **SoftSide** is already in the preparatory stages. In it, we plan to unveil the mysterious I-$tring we've been hinting about for several months. Our intention has not been to be mysterious. I-$tring simply has so many implications, we have not known quite how to begin explaining it to you. Imagine, if you will, a thick road atlas. Now, open that atlas to a single page and look down on it. Transfer the map to the screen of your computer with a blinking cursor indicating your location. Press a couple of keys and the area covered by that cursor enlarges to the entire size of the screen, showing a great deal more detail, with that cursor still there, blinking away in its original size. Press a couple more keys and the area covered by the cursor again explodes to the size of the screen showing the most minute detail. Step back up to the original map and you can move the cursor around, character space by character space, with the option of stepping back down to examine any of the locations in the other two "resolutions" previously described AND from any of eight different perspectives in the highest mode of resolution. As you approach the edge of the map, the cursor crossing into the boundary area will trigger yet another map to load from your disk, turning the page of the atlas, in effect.

Impressive, no? But, that's not all. I-$tring not only gives you a method of looking at things differently, but one of experiencing them as well. As you "walk" your cursor around the map, the features of the map will react just as though they were real. You cannot walk through walls or on water. Closed doors must be opened before you can enter. Don't try to swim when you're carrying too much weight or you'll surely drown. By the way, can you swim? Your wildest fantasies or most practical mistakes may be lived on the



screen of your computer with no danger to life, limb or finances.

"But," you say, "what if I'm not interested in the set of maps you choose to put in your atlas." No problem. With the "Map Maker" utility of I-$tring, you have the option of designing your own maps in any sequence, again, with the option of stepping down into the two more detailed resolutions and designing the detail contained therein, as well. You also design the "intelligence" of the environment. In other words, if you don't want the user to walk through walls, design them that way. If a bridge will only withstand a certain weight limitation, put it in the characteristics of the bridge's format. Your only limitation will be the amount of disk space you have available to store the "pages" of your "atlas."

The technique of I-$tring can be applied to Adventures, Architectural Design, Community Planning and multitudinous other fields. Data bases of pre-designed map-making characters may be stored for applications by other users and made to be interactive. An example would be a set of characters to be used in designing a house. What would happen if plumbing and wiring were improperly intertwined? I-$tring has the capability of assessing the combinations of situations and warning the user of the possible ramifications. Many of the problems is this world are dependent on physical interrelationships. I-$tring is a tool for examining them. That's all I'll say about I-$tring for the moment. Next month you'll start to see some of the actual programming goodies that make all of its capabilities possible.

Submissions for "The Adventure of the Month" are starting to roll in, but not yet at the pace we would like to see in order to give you the quality you expect from **SoftSide Publications** without a great deal of gnashing of keyboards and substantial enhancements. I cannot over emphasize the importance of complete line-by-line documentation. We thoroughly recognize that most of you only have access to one system on which to compose your programming brilliance. However, the more complete your documentation and explanation of variables, the more likely that we'll be able to translate your program for the other two systems we support.

Whew! I've finished my first communication with you, the faithful readers of **SoftSide**. I feel as though I know many of you from the long conversations we had when I was travelling the country with the "Micro-Faire" computer shows. I'm anxious to hear from you more and more as all of our various new projects become reality. Your input is highly valued and no suggestion passes by without serious consideration. We are all truly the pioneers of a new frontier. It's a tough road because we never know what lurks around the corner. If you've already been around that corner and can inform us about what awaits, please do so. We'll do our very best to pay you the same courtesy.

Input

continued from page 8

Dear **SoftSide**,

I am writing this letter to let all Model I and III owners of S-80s know that they can also get Hi-Res and color with graphics up to 256*192. This is done by buying the Electric Crayon made by Percom for under $300. It's been on the market long enough!!! Yet no one has been smart enough to develop some software on it. The problem is everyone who has craved for Hi-Res bought either an Apple (sour) or Atari. Let's get going S-80 owners. I know I will be submitting a couple programs (mostly games in Hi-Res) to this great magazine **SoftSide**. That will be soon, once I learn more on programming in Level II.

I was wondering if anyone has made a couple games that they could submit to **SoftSide** or me that work through the S-80 for the Crayon.

I do know that Programma has made a couple pieces of software for their Hi-Res board, and there is someone in Florida that has been selling some software for the Crayon. I don't know why they never advertise in **SoftSide** or any other kind of magazine. . . they would make a bundle of money.

P.S. The sour after the word Apple is for all those people who call the TRS-80 the TRaSh-80!!!

Marc Fraioli
St. Amford, CT

Dear **SoftSide**,

I've had my Atari 800 for about six or seven months now and I'm a lover of games. I enjoy reading through magazines about programming hints or tricks. I've decided to share something I've just discovered, something new. For all you people out there with the game "Star Trek III .5", this is for you.

Whenever you fire a photon torpedo at anything other than a klingon, you get court-martialled and the game ends. I've discovered that if you type in the command CONT that the game will continue as if nothing had happened. Sometimes my computer replaces the quadrant with a pulsar or something else.

I think Lance Micklus has done a great job with the game. My favorite part is the phasers and how you can have multiple shots. Even when I have photon torpedoes, I use phasers just to watch. I highly recommend this game to people with an Atari or Apple. Great job, Lance!

Scott Macfarlane
Pennington, NJ

Dear **SoftSide**,

In the January, 1981, issue Phillip Case bowed to and walked away from the Sublogic "Flight Simulator" loader. I, however, did not. The reason for me to be decoding this in the first place was the need to make it work on the Model III. A teacher at one of our elementary schools had recently sold his Model I and bought two Model IIIs. To his dismay, the screen came up "Cass?" when he would start the loader. He called to ask if I could find and fix the error. After a weekend I had converted the program for the Model III. There was no need to change the program code itself, just the loader and tape format.

Now to the point. If any of our Model III readers have found out the hard way that the Sublogic package didn't work on their computer, or would like to be able to buy it now, I would gladly send instructions on how to convert this wonderful program. Note that the change requires access to a Model I Level II machine with an Editor Assembler (a local Radio Shack store may be willing to offer their services). I just hope that this doesn't increase the pirating rate and that any request to me is a legitmate one. I can't say whether or not I like the new format of **SoftSide** since I never saw the old one!

David Yon
R.D.#1-168
Richmond, VT 05477

Dear **SoftSide**,

I am a Middle School teacher who finds your magazine invaluable, as a source of programs and programming aids which are extremely effective with school-age children. I do, however, find your magazine lacking in its recognition of the achievements and the abilities of these same school-age children. For example, I can only recall seeing a handful of programs which were published that had been written by school children. May I suggest that you devote a few pages of **SoftSide** magazine (or perhaps publish a mini-**SoftSide**) which would deal exclusively with programs written by school students.

I feel that the students who have the ability to write quality programs need and deserve our recognition and support for their accomplishments. One program, written by a student and published by a national magazine, is bound to motivate numerous other students to put their skills to the test.

Remember! In a few years, you and I could be working for these same students!! They deserve a chance!

Karen Ann Golubic
Edinboro, PA

Editor's Reply: SoftSide has published programs written by high school students. We do not take the age of the programmer into consideration when evaluating our submissions, but rather the quality of the program itself.

Dear **SoftSide**,

In the April, 1981, issue, Jon Voskuil's editorial listed a number of suggestions for polishing the style of a particular program. I agree with these suggestions, all of which improve a program's readability. I wish, however, that the S-80 program listings in **SoftSide** followed suggestions 4b and 4c (regarding long lines of coding and lack of spaces). I own an Apple II and occasionally enjoy translating an interesting-looking S-80 program into Applesoft. Long lines of code with no space make this task much more difficult than it would be if the listings were more readable.

If the problem results from trying to fit long programs into 16K memories, perhaps a solution would be to list the programs with spaces but advise 16K S-80 owners that they should eliminate the spaces as they key in the programs.

I enjoy your publication, but think that the advantages of supporting the Apple, Atari, and S-80 in one magazine are lessened if unnecessary barriers are placed in the way of those of us who wish to try to adapt programs written for one computer to another.

By the way, you could make translation easier by publishing a picture of the screen if a program uses graphics. For example, the "Volcano" program in a previous issue looked interesting, but it apparently started with a graphic depiction of an eruption. The difficulties of translating this into either high or low resolution Apple graphics would have been considerably lessened if some indication of how this was supposed to look had been provided.

Kristin Jensen
Huntington Beach, CA

Dear **SoftSide**,

Several people here in Houston have organized "Houston Atari Computer Enthusiasts" (HACE). HACE is a group of people with a common interest in learning more about and making more effective use of their computers. Members also help each other solve computer problems and freely exchange information and non-copyrighted programs, through the HACE software exchange library.

Some of the members have asked if they may give programs obtained from your magazine to non-subscribers. Our general feeling is that unless a person is a subscriber to your magazine, he does not have permission to copy programs from it. Are these programs protected by copyright laws? We would appreciate your clarification of our rights to the material in your publication with regards to the giving or trading of such material with non-subscribers.

Gerald Manegold
Secretary, HACE
Houston, TX

Editor's Reply: SoftSide's programs are all copyrighted and meant for the exclusive use of its subscribers. To allow programs to be copied and distributed freely cuts into the author's royalties. As a potential author, how would you feel if your programs were circulated without any compensation to you?

Dear **SoftSide**,

You published my January 16, 1980, letter in the Input column of the March, 1980, issue of **SoftSide: Apple Edition** which was only the third issue for the Apple II. In that letter, I made some observations about relative advertising space in **SoftSide** vs the Biggie computer magazine — **SoftSide** won (i.e., having less).

In that letter I complained about the wisdom of publishing programs that must be laborously typed in with the expected typos and the near impossibility to debug without a printer. I was wrong!

The Apple One Liners caught my eye and I keyed in a few without much problem, then I ventured into "Meteor Storm." I do enjoy the luxury of a printer and found debugging typos not too bad. But there were hidden advantages.

I learned the touch keyboard at a tender age, but never commanded the top row of keys. The need to type long data statements and other numbers forced me to make the commitment to master that top row which I did with the next program. Advantage No. 1.

More importantly I found myself interested in how the author "did it" and have found and borrowed techniques that have added a new dimension to my own programming. Big Advantage No. 2.

Some of the programs are utilitarian in nature but most of them are games and very entertaining. Advantage No. 3

I have followed the reader criticism of bugs in programs and have been a little frustrated in that regard myself. In partial offset however, debugging has FORCED me into the program logic and has been the source of a better understanding of program architecture. I hope you can solve the problem of missing code falling to the cutting room floor.

Although I feel that **SoftSide** gives short measure to Apple, relative to the probable

number of Apple users, I think **SoftSide** is a very worthy publication. Keep up the good work.

Earl E. Marchand
Houston, TX

Dear **SoftSide**,

It seems a lot of people out there are rooting for your old S-80 edition. I own a Model III S-80, but only started subscribing a few months ago (and am very pleased with what I can use from your magazine). If these pro S-80 letters continue coming in, and bitchy Apple and Atari letters continue, I suggest that your staff consider changing your magazine into the way it used to be. (S-80 exclusively). I mean, there are over 250,000 S-80s out there! More than any other! Why not serve the majority? Hurray to J.J. O'Malley, Jr. in the June, 1981, Input!...Boo to Larry Eiss in the same Input. What point was Larry trying to make ???? Tandy is a "small crumb of the computer cake"?? Ha! Send that to "Laughter, the Best Medicine"! It seems that some people let their worlds rotate around "Apples" sometimes! Open up, world!

I agree with D. Scott Harper's suggestion to stop "squashing" programs. Also, it would be great if you could take advantage of the S-80's AUTO command (or rather, let us take advantage of it) by making sure that published programs have line numbers like "10, 20, 30, 40" instead of "1, 5, 17, 39, 2938". It's a pain doing all of those weird numbers....bug city!

LAST, BUT NOT LEAST — When you do a program like "Old Glory", at least tell us if it's for a disk system or not! I typed in it on my Model III 48K cassette system, and got down (after some work) to line 30010. SUPRISE! "CMD'T", "DEFUSR", etc. I ran the thing just to see if all of my work would work,....nope. I'm going to type in "Divide and Conquer" tonight. I notice some "NAME's" in there, but look! — It says: "Divide and Conquer" is an S-80 program which requires at least 16K RAM...I guess it does work (sound and all) with my cassette computer.. It also says "compatible with Level II and Disk BASIC", also! We'll see...

Tim Knight
Moraga, CA

Dear **SoftSide**,

I own a S-80 and have subscribed to your magazine since November. Your magazine has several good points: 1) All regular features are terrific. 2) I have experienced none of the problems in typing reported; all of the programs I have typed in have worked fine. 3) Congratulations to the two Chris Lights for what must be the most hilarious piece of writing I have seen in some time, "Big Apple Adventure." I have shown it to at least 20 others who could have no way of knowing of the allusions to the Apple keyboard and Colossal Cave, and they still enjoyed it. 4) I am glad that Mr. Elsasser's suggestion (February) about segregating the programs has been used. Now I know where the stuff for my computer is.

The constant letters from Apple owners on including the S-80 programs and, more recently, the lack of Apple Adventures and not renewing subscriptions make a bad addition to your magazine. Some people just want a cheaper and simpler computer, and so they buy S-80s. That is the simple reason why more people have the S-80 than the Apple, and because of this, **SoftSide** publishes more programs for it. The subject is further discussed in Outgoing Mail (June).

Unfortunately, your magazine has several bad points as well. They are: 1) The reviews are terrific, but how about saying which computers these programs are for? At present, I must "read between the lines" for such clues as "Hi-Res", "Applesoft", and "The S-80's terrible graphics." 2) In the paragraph on top, which says which computer and how many K it is for, add whether it requires a disk drive and/or line printer. Four articles in the June issue ("Race of Sorts," "Word-Search Puzzle Generator," "Divide and Conquer," and "Anallist") require at least one of these. For us ordinary mortals who cannot spend $800 for a disk drive and $300-up for a line printer, articles assuming we can are confusing. 3) I agree with Mr. Wagner (June). There SHOULD be a series on converting programs. There must be someone in your staff who knows more than one computer. If there is not, one of your readers might. 4) In response to another point of Mr. Wagner's concerning ordering back issues, you said "See page 45." On page 45 is just an advertisement explaining about cassette and disk subscriptions. Is this a misprint?

Otherwise, your magazine is terrific. I will probably renew my subscriptions again and again.

Randall Rose
Plainview, NY

Editor's Reply: The ad for back issues was pulled at the last minute and the reference to it in one of the Input letters was overlooked.

Dear **SoftSide**,

Once a month I receive a newspaper from the Computer Information Exchange called "The CIE People's Software News." It usually contains several articles about the software they sell. The unusual part is that they give a very detailed description of their main software packages instead of the usual skimpy overview.

Two months ago I noticed an ad for a program called NEWBASIC which would add 33 new commands to your TRS-80. NEWBASIC is sold by Modular Software Associates for $19.95 for the Level II version, $29.95 for the disk version on tape and $35.90 for the disk version on disk. The following month a very large and detailed explanation of NEWBASIC was on the front page. After reading it over a few times, I decided that I just had to have NEWBASIC and phoned in my order.

Seven days later it arrived and I have not been disappointed at all. It came on a Memorex disk with a 44 page manual that explains how to easily use NEWBASIC.

First is the creator program that will allow you to use any or all of the 33 commands available. Each time a command is listed, a short explanation of the function is given with an approximate size of the code needed to add each command. After responding (Y/N) the next command is shown until all commands have been entered and then a total size is displayed, indicating the total byte count NEWBASIC uses once it is executed.

Now for the goodies. Just what will NEWBASIC do? Some of the commands can be toggled on by typing in the command and then toggled off by typing in the command again. These are:

BLINK - turns on or off blinking block cursor
REPEAT - activates repeating keyboard and beep
LCASE - toggles lowercase display for Scripsit and Electric Pencil
QUICKEY - allows entry of entire BASIC keywords

Four new graphic commands:

WTS - whites the entire screen instantly
FILL - turns on graphic blocks between two points
RECT - draws a rectangle between two points
LINE - draws a line between two points

SPOOLON, SPOOLOFF and DESPOOL -three spooling commands

Now you can control your RS232 from BASIC with three more commands - RS232, RSOUT and RSIN. Create your own DO/UNTIL loops, locate a "string" or string within your own program, echo your screen to the line printer even with printer driver routines in memory, assign labels to your routines, call a machine language program from BASIC without USR(X) or POKES, get a directory from BASIC, set memory size within BASIC and a sound command that lets you have almost any frequency or duration that you want. And that is not all.

The nice part of it all is that these commands are really useful, not a bunch of functions that you will never use. Also, the price is right. If you were to buy a software package for the LCASE command alone, it could cost more than this whole package.

Since NEWBASIC occupies RAM below normal BASIC, it moves BASIC programs higher in RAM than usual. Some programs that use PEEK and POKE don't allow for this and may not run properly. All NEWBASIC commands must be at the start of a BASIC statement.

I highly recommend this program to anyone with a disk system. I cannot comment on the tape version, but if it is anything like the disk version, it is well worth the price. I am anxious to see what Modular Software Associates are up to now.

Terry Hazelett
Parkersburg, WV

Editor's Reply: This is the very same NEWBASIC featured in this issue of SOFTSIDE! You'll find it lives up to these hurrahs and much, much more.

Do You Have What it Takes to be a

SoftSide Editor?

Are you computer literate? Are you just plain literate? Are you both? If so, then you may be the person **SoftSide** is looking for. We need people who like to write both in English and in BASIC to work on our editorial staff.

As a leader in the field of software publishing, **SoftSide** is committed to providing its readers with clear, concise, and easy to understand articles and programs for the TRS-80™, Apple, and Atari microcomputers. Our growth rate is such that our staff is constantly expanding and we are always on the lookout for those with the special skills required to put out a quality magazine.

While New Hampshire wages are not the highest around, we offer a better scale of pay than most of our competitors. Furthermore, our fringe benefits are excellent. Another bonus is that New Hampshire has no personal income or sales tax.

Located in a beautiful area of southern New Hampshire, Milford is within an hour's distance of Boston, the Atlantic Ocean, and the White Mountains. Nashua and Manchester, New Hampshire's two largest cities, are but 15 minutes from our offices. Ski slopes abound in the region, as do fine restaurants, arts and crafts centers, and countryside the likes of which must be seen to be believed.

If you are a hard-working, skilled person with experience in the fields of computers and magazines, and if you enjoy the relaxed country lifestyle with the convenience and cultural opportunities of major urban centers nearby, drop us a line. We might be good for each other.

If you feel that you fit the bill, please write to us and let us know who you are and what you can do. A formal resume isn't a necessity, just send a letter outlining your skills, qualifications, and experience to:

Randal Kottwitz,
SoftSide Publications,
6 South Street,
Milford, New Hampshire
03055

BUGS, WORMS, and other undesirables

The Apple version of "Miner" (February, 1981) needs to have corrections made to lines 730 and 740. They should read:

```
730  POKE 34,20: HOME : PRINT "GP
     $ ";GP;: PRINT  TAB( 16): PRINT
     GC;" OZS."
740  PRINT "CASH $ ";M;: IF M > 7
     000 THEN  GOTO 790
```

A couple of bugs have been reported in the S-80 version of "Strategy Strike" (March, 1981). The following two lines should be changed as indicated:

```
870 FOR I=1TO10: IF C(C1,C2)=10 THEN
PRINT@ E(C1-AD,C2), etc.

1030: the variable C9 should be C1.
```

A minor bug in "Word Wars" (July, 1981) can cause a "subscript out of range" error under certain circumstances. Changing line 1490 as shown will correct the problem:

```
1490 FOR I=0 TO NP-1: CS!(I)=0: NEXT I:
NL=13: GN=1
```

Two more bugs have been discovered in "Math Decathlon" — neither of them fatal, fortunately. In part three (May) line 6070 contains a variable which was changed everywhere else in the program; it should read:

```
6070 C =  INT ( RND (1) * 3 * (S(
     P) ^ 2 + 1)) + 1: IF C = B THEN
     6070
```

In part four (June), a last-minute attempt at a small enhancement created a potential hangup with the Penny Battle game. Line 9155 should be added, and line 9160 changed as follows:

```
9155  IF N2 = 0 THEN N2 = 1
9160  IF ( RND (1) * (S(P) + 1) <
     .4 AND NUM >  = MAX * 2) THEN
     N2 =  INT ( RND (1) * MAX +
     1)
```

CALENDAR

September 1-3

Computerized Office Equipment Exposition
Civic Center, Atlanta, GA
A three-day expo displaying new equipment for data/word processing, record storage/retrieval, information management and micrographics. Business executives can benefit from this forum on office automation.
Contact: Cahners Exposition Group, 222 W. Adams St., Chicago, IL 60606, (312)263-4866

September 10-12

Personal Computer World Show
Cunard Hotel, Hammersmith, London, England
Various demonstrations and discussions will be held on small computer systems.
Contact: Timothy Collins, Personal Computer World Show, 11 Manchester Sq., London W1E 2QZ, England

September 14

Invitational Computer Conference
Boston Marriott, Newton, MA
This will be a seminar geared to the needs of quantity buyers of computer and peripheral equipment.
Contact: B.J. Johnson & Assoc., 2503 Eastbluff Dr., Suite 203, Newport Beach, CA 92660, (714)644-6037

September 14-17

Productivity — An Urgent Priority
Capitol Hilton, Washington, D.C.
Conference is designed to provide a wide focus on productivity in the computer industry.
Contact: Compcon Fall '81, POB 639, Silver Spring, MD 20901, (301)589-3386

September 14-17

Software Info '81
Merchandise Mart Expocenter, Chicago, IL
Lectures and demonstrations will be conducted on increasing productivity through software packages.
Contact: Software Info, 1730 N. Lynn St., Suite 400, Arlington, VA 22209, (703)521-6209

September 15-17

Wescon '81
Brooks Hall, Municipal Auditorium and Hilton Hotel, San Francisco, CA
Exhibits and conferences will feature computer equipment and related products. Offerings will include discussions on data communications and computer hardware and software. Topics will also include aerospace avionics, medical electronics, consumer electronics and office automation.
Contact: Electronic Conventions Inc., Suite 410, 999 N. Sepulveda Blvd., El Segundo, CA 90245, (213)772-2965

September 16-18

Diagnostic Software Planning and Design
Boston, MA
This course for design, test and diagnostic engineers and managers is being offered at a cost of $495.00. It will feature design examples, lectures and information sessions. Individual and group programming sessions will also be included in the program.
Contact: Prof. Donald D. French, Institute for Advanced Professional Studies, One Gateway Center, Newton, MA 02158, (617)964-1412

September 21-24

Robotics Course
Centennial College, Ontario, Canada
This is the first course in a series of ten-week courses which will run through November. The first, on September 21, will cover the application of robots in industry. September 22 will center on digital logic and microprocessors. The history of programmable controllers will be held on September 23 and September 24 will be designed around Intel 8080 and 8085 microprocessors.
Contact: Coordinator of Technical Programs, Centennial College of Applied Arts and Technology, Box 631, Station A, Scarborough, Ontario, Canada M1K 5E9

September 24-25

Microprocessors: Hardware, Software and Applications
Worcester Polytechnic Institute, Worcester, MA
Courses will be given on the basics of hardware and software, selection/evaluation of microprocessors, memory, input/output systems, circuit testing and debugging.
Contact: Ginny Bazarian, c/o Office of Continuing Education, Worcester Polytechnic Institute, Worcester, MA 01609, (617)753-1411, Extension 517

If you or your organization are sponsoring or know of an event you think would be of interest to **SoftSide** readers, please send complete information to:

SoftSide Publications
6 South Street
Milford, NH 03055

Be sure to include complete information concerning dates, location, subject matter and a contact name, address and phone number.

Programming Hint

For those who are just beginning a study of Machine Language programming, here is a short program to be used with a printer that will print out a complete table of hexadecimal and decimal equivalents (from 0 to 255 — or 00 to FF). It pauses halfway through the printout for you to change to a second sheet of paper.

```
10 DIMA$(16):FORI=48TO57
:A$(I-48)=CHR$(I):NEXTI
20 FORI=65TO70:A$(I-55)=
CHR$(I):NEXTI
30 FORJ=0TO255:J1=INT(J/16):
J2=JAND15:LPRINTTAB(
(JAND3)*16);
40 LPRINTUSING"###=%%H"
;K;A$(J1)+A$(J2);
50 IF(JAND3)=3THENLPRINT" "
60  IFJ=71THENINPUT"PRESS
ENTER TO CONTINUE";Q$
70 K=K+1:NEXTJ
```

THE NATIONAL COMPUTER SHOWS

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

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

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

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

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

Ticket Information

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

THE MID-WEST COMPUTER SHOW

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

THE MID-ATLANTIC COMPUTER SHOW

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

THE NORTHEAST COMPUTER SHOW

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

THE SOUTHEAST COMPUTER SHOW

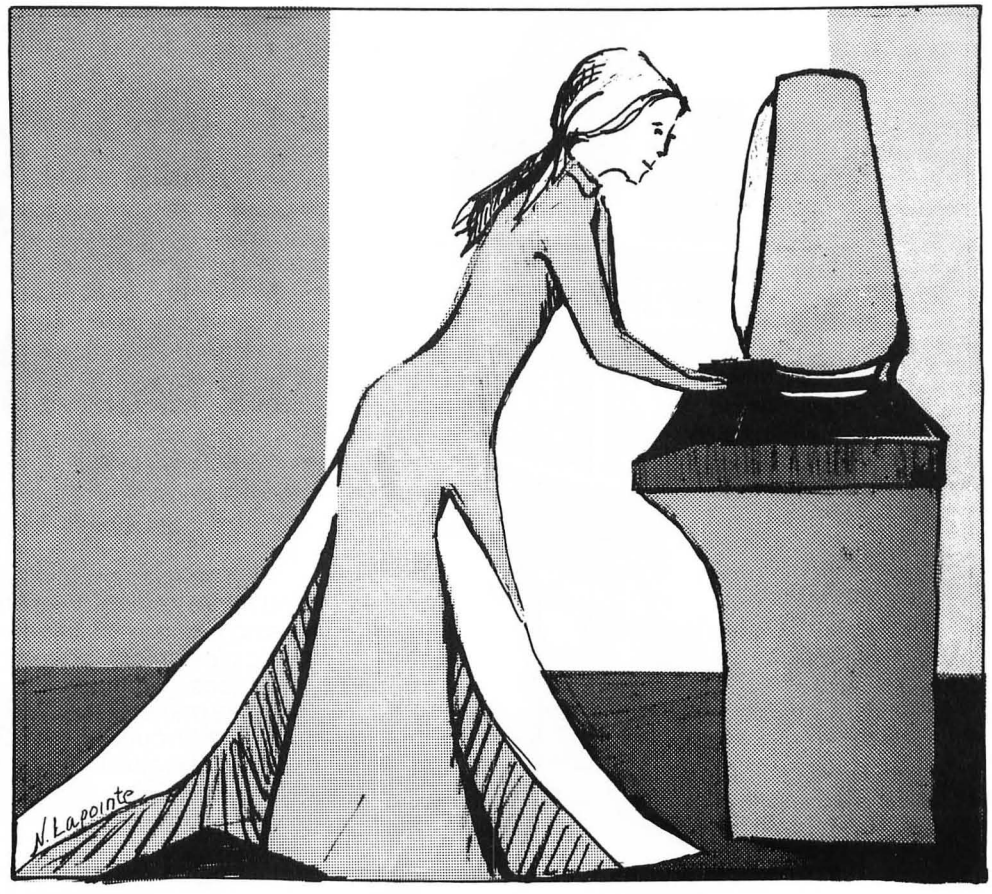
ATLANTA
Atlanta Civic Center
395 PIEDMONT AVE NE AT RALPH McGILL BLVD
THURS-SUN
OCT 29-NOV 1, 1981
11 AM TO 7PM WEEKDAYS
11 AM TO 6PM WEEKENDS

THE SOUTHERN CALIFORNIA COMPUTER SHOW

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

The Sensuous Programmer

by "J"



The fifth intimate interlude

J's Theorem: "All computer users are idiots."

J's Corollary: "All computer programmers who don't believe that all computer users are idiots, are greater idiots."

Idiot-proofing programs is one of the staples of the sensuous programmer's art. If mere misspellings, grammatical errors, and sloppy screen formatting make for "tacky" programs (as I argued in my third seductive installment), then what grim label must be applied to mid-program crashes accompanied by system error messages? Would "Nasty" be adequate? "Disreputable"? "Repugnant"? "Horrific"? Alas, one is tempted to invoke unprintable vulgarities to describe such behavior in programs. And all because the programmer neglected a few simple precautions and instead constructed a time bomb waiting for the right dummy to set the timer.

Idiot Countermeasure #1: Get all user input in the form of strings. All of it. Always. No exceptions. (Well, none that occur to me at the moment.) None of the systems that we're covering — Apple, Atari, or S-80 — take very kindly to receiving a non-number when they're expecting a number. The S-80 and Apple don't go so far as to bomb, but they do give you messy error messages like "?REDO" and "?REENTER". The Atari doesn't mess around; it not only gives one of *its* usual cryptic messages ("ERROR-8 AT LINE nn"), but goes all the way with a total program crash.

Preventing such things is simplicity itself, and should become second nature to a BASIC programmer. Get all user input in the form of strings, and then convert them to numeric form if needed. Instead of

```
100 INPUT A
```

use

```
100 INPUT A$
110 A = VAL(A$)
```

The VAL function gives the arithmetic value of everything that can be interpreted as a numeric character prior to the first non-numeric character in the string. For example, the VALue of the string "279AB34" is 279, and the VALue of the string "+3.61 NL" is 3.61.

The Apple and S-80 are a bit more tenacious in trying to interpret characters as parts of numbers than is the Atari. The Atari will ignore leading spaces, but after encountering a number will then stop at any subsequent space; both Apple and S-80 ignore all spaces. Atari, then, gives VAL(" 12 3") as 12, while the others will give 123. All will try to interpret "+", "-", ".", and "E" as elements of a number, so that the string "+4.65E-02" will successfully be converted by VAL into the positive real number .0465 (4.65 times 10 to the -2 power).

One important consideration with the Atari is that if the first character of a string cannot be interpreted as a part of a number, using the VAL function on that string will cause the program to crash. This is not a problem with the Apple or S-80, which will simply return a VALue of zero under those circumstances. Therefore, to really idiot-proof the Atari's input, you must first check the initial character of the string by adding a statement such as this to the above program:

```
105 A = ASC(A$,1,1): IF A > 57 OR (A < 48 AND A < >46 AND A < > 45 AND A < > 43) THEN 100
```

This checks to make sure that the first character of A$ is either a digit, a plus sign, a minus sign, or a decimal point; if not, it goes back for another try.

Another safeguard must be included on the Atari and on the S-80. If the user just presses RETURN or ENTER in response to the INPUT prompt, the string A$ retains whatever value it had previously. That may occasionally be useful, but it can wreak havoc with your program if you're not anticipating it. It's a good idea, then, to precede the INPUT in line 100 above with the statement A$ = "", assigning a null value to it. (The Apple, logically enough, will automatically assign it the null value if the user presses only RETURN.) In the Atari's case, however, if you use the suggested line 105, you will need to add another check before it to see if A$ has a null value. Otherwise, you again risk being

the victim of a crash when line 105 looks at the first character of a non-existent string.

The complete idiot-proofed routines, then, turn out like this for the three computer systems:

Apple:

```
100 INPUT A$
110 A = VAL(A$)
```

S-80:

```
100 A$ = "": INPUT A$
110 A = VAL(A$)
```

Atari:

```
100 A$ = "": INPUT A$
102 IF A$ = "" THEN 100
105 A = ASC(A$,1,1): IF A > 57 OR
(A < 48 AND A < > 46 AND A <
> 45 AND A < >43) THEN 100
110 A = VAL(A$)
```

Idiot Countermeasure #2: Once you've managed to get any input that the user might have chosen to type in, without error messages or program crashes, then make sure that the input is within the limits that your program can handle. If you're looking for a number between 1 and 5, then put in a line that will check for that:

```
120 IF A< 1 OR A > 5 THEN 100
```

And if you're looking for an INTEGER between 1 and 5, you'd better make sure that it IS one. The following addition will round off the input to the the nearest integer, but you could also just lop off any decimal value if that suits your purposes (by omitting the "+ .5").

```
120 A = INT(A + .5): IF A < 1 OR
A > 5 THEN 100
```

Incidentally, in the event that the input is unacceptable, you may well want to go back to a line prior to the actual INPUT statement. For example, if you took the pains to have the user's input displayed at a particular place on the screen with a PRINT@ or POSITION or VTAB/HTAB statement, then you'll probably want to return to those lines to prevent messing up the display. You may also want to include some kind of user prompt to tell the idiot what he's doing wrong and why he has to re-enter his response.

Similar checks need to be made on string input which is not to be converted to a numeric value. If you ask someone to type in his name, and you later use that name in a carefully formatted PRINT statement, it's imperative to make sure that it's not going to be too long. Decide what a reasonable maximum length would be, allow for that many characters in your PRINT statement, and use a string function to hack off any extra characters. For example, a statement such as this could be used on the Apple or S-80:

```
210 IF LEN(N$) > 12 THEN N$ =
LEFT$(N$,12)
```

or this on the Atari:

```
210 IF LEN(N$) > 12 THEN N$ =
N$(1,12)
```

Another common reason for checking string inputs is to make the exact response less crucial. Rarely (unless you have some special devious reason) should you require the user to type in the exact words "yes" or "no", for example. Since it's a nice accident of colloquial English that lots of positive words start with "y" (yes, yep, yessir, yeah, etc.) and lots of negative ones start with "n" (no, nope, no way, never, not a chance, etc.), all that's really needed is a check for the initial letter. A typical input and evaluation routine might look like this:

Apple:

```
500 VTAB 22: HTAB 1: INPUT
"WANT TO GO FIRST? "; X$
510 IF X$ = "" THEN 500
520 X$ = LEFT$(X$,1): IF X$ = "Y"
THEN 800
530 IF X$ < > "N" THEN 500
540 GOTO 700
```

S-80:

```
500 X$ = "": PRINT @640, "WANT
TO GO FIRST ";: INPUT X$
(510-540 same as for Apple)
```

Atari:

```
500 X$ = "": POSITION 2,22: PRINT
"WANT TO GO FIRST ";: INPUT X$
510 IF X$ = "" THEN 500
520 X$ = X$(1,1): IF X$ = "Y"
THEN 800
(530-540 same as above.)
```

Idiot Countermeasure #3: In especially crucial situations, include a routine to bypass normal system error processing. This is generally quite unnecessary with your average, run-of-the-RAM program such as "Space Shoot-Em-Up #44" or "Blackjack 2001". But if it has taken you seventeen hours to get past ghouls, killer spiders, and various other creepy-crawlies into room #108 of the Cave of Deep Terrors, you want to make sure that some little glitch somewhere doesn't leave you staring at a system error message *without the* chance to save the game first.

Actually, if you're programming stuff like that you're probably wasting your time reading this article. But I think that any program which accesses the disk drive during execution can benefit from an error-trapping-and-processing routine. Disk errors such as "DISK FULL", "FILE NOT FOUND", "FILE LOCKED", and their ilk can really mess up an otherwise nice program. All three of the systems in question have provisions for you to detect such errors and process them to suit your purposes.

You let the computer know that you want to do the error-handling yourself by using the BASIC statement appropriate to your system: "ON ERROR GOTO" for the S-80, "ONERR GOTO" *for the Apple, and* "TRAP" for the Atari. This statement is always accompanied by the line number where your error-handling routine begins: ON ERROR GOTO 10000 or TRAP 22000, for example. This routine can be stuck in anywhere, but the trapping statement itself must come BEFORE an error happens, in order to intercept it — usually as one of the first lines of the program.

When an error of any kind is encountered, which would normally generate an error message and often cause a program crash, control is instead given to your routine. This routine will normally do the following things:

A. Determine what error has taken place. This is done by locating the error code number, which the computer stores in a special memory location. The S-80 has a built-in function to help with this, such that the quantity (ERR/2 + 1) gives the code. On the Atari, a PEEK(195) will return the code, and on the Apple, PEEK(222). These codes and their corresponding errors are different for each system, of course, and are listed in the appropriate reference manuals.

B. If needed, determine the line number where the error occurred. Again, the S-80 has a BASIC function which helps out: The quantity ERL will be the line number. The Atari will yield the number with a PEEK(186) + 256 * PEEK(187). And the Apple — well, it must be in there somewhere; if one of you knows where, write in and let me know.

continued on next page

continued from previous page

C. Using the information you now have about the error type (and location, if needed), give the computer and user some instructions on how to get out of this pickle. How complicated this will be depends on how many different types of errors you're trying to anticipate. Let's say you're only concerned about a disk being full during a program-controlled SAVE on your Apple. In that case your error-handling routine might look like this:

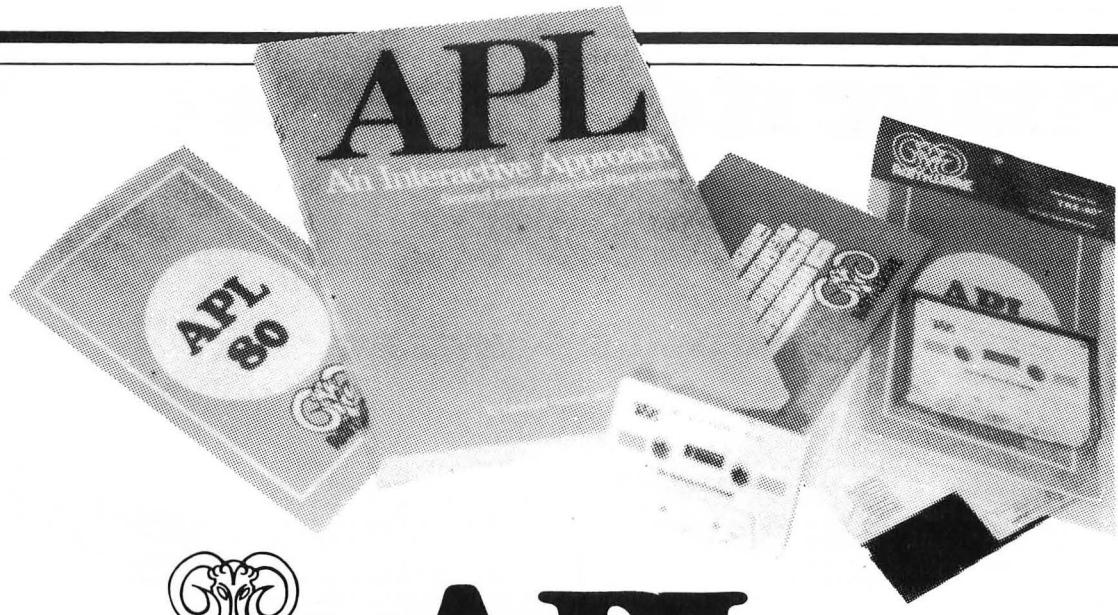
```
10000 E = PEEK(222): REM
ERROR CODE
10010 IF E<>9 THEN 10100:
REM 9 IS "DISK FULL"
10020 PRINT "YOUR DISK
DOESN'T HAVE ENOUGH
ROOM."
10030 PRINT "PLEASE INSERT
ANOTHER AND PRESS
RETURN.";
10040 INPUT ANY$
10050 RESUME
10100 PRINT "SORRY, THE PRO-
GRAM JUST CRASHED."
10110 PRINT "ERROR CODE: "; E
10120 STOP
```

With small modifications, this routine would work on the S-80 or Atari as well. Note that it can only process a DISK FULL error, and all others will cause the program to end. This is a bit on the skimpy side for error-handling, but you get the idea.

The Apple's RESUME statement (line 10050) returns control of the program to the beginning of the line in which the error occurred. The S-80 uses RESUME to continue with the STATEMENT in which the error occurred (even if that statement was not the first one in its line), and in two additional ways. Using "RESUME NEXT" will cause the program to continue at the statement following the one in which the error happened; or, substituting a line number for "NEXT" will cause execution to resume at the specified line. The Atari has no equivalent of RESUME, so you must use a GOTO along with the line number that you found in step B if you want to continue where the program broke off. On the Atari, you must also reset the TRAP once it has been used to process an error.

Idiot Countermeasure #4: Keep reading "The Sensuous Programmer." It's especially written to make programmers out of idiots. Or is it the other way around?

RESUME NEXT month.



RAMWARE

APL

by Phelps Gates

Now a high-level, scientific programming language for the home computer that doesn't cost $200 or $300. The power of this language is in its strong mathematical operations, especially with regard to matrices and vectors. Programs requiring matrix multiplication or other matrix problem solving that would require hours of programming time in BASIC are solved quickly and with minimal effort in APL.

To aid in learning APL, lessons are included on the disk. Starting from the basics, you are brought step by step through the various programming techniques involved with APL. These lessons act as a tutor which will have you "talking APL" in no time. Also available is the book, "APL: An Interactive Approach," which reinforces many of the examples given in the lessons and provides additional insight into APL programming.

FEATURES

APL-80 on disk contains the following features:)SAVE and)LOAD workspace on disk;)COPY other workspaces into current ones; Return to DOS for directory or commands without losing your workspace; Send output to lineprinter; Five workspaces of lessons included; Sequential and random files; 15 digit precision; Monadic and dyadic transposition; Easy editing within FUNCTION lines; Latent expressions (FUNCTION can "come up running" when loaded); Tracing of function execution; Real-time clock; User-control of random link; Workspace is 25587 bytes (in 48K machine); Arrays may have up to 63 dimensions.

COMMANDS APL-80

APL-80 supports the following commands; Absolute value, add, and assign, branch, catenate, ceiling, chr$/asc, circular, combinational, comment, compress, deal, decode, divide, drop, encode, equal, expand, exponential, factorial, floor, format, grade down, grade up, greater, greater/equal, index generator, indexing, index of, inter product, label, less, less/equal, logarithm, maximum, member, minimum, multiple, nand, negate, nor, not, not equal, or, outer product, peek, poke, quad, quote quad, random, ravel, reciprocal, reduction, reshape, residue, reverse, rotate, scan, shape, sign, system, subtract, take, transposition.

SPECIFICATIONS

Minimum system requirements: 32K disk system (48K recommended) includes APL-80, Five workshapes of lessons, instruction manual. . . . . $39.95 on disk

Reduced feature: 16K Level II tape version, no lessons.
Transpositions, format, and inner product not implemented. Reduced domain for some functions, 6 digit accuracy.
. . . . . $14.95 on cassette

LIMITATIONS

Due to the absence of the special APL character set on the TRS-80 , APL-80 uses shifted letters to represent the various APL characters. In addition to the keyboard limitations, lamination, domino, and matris inverse are not implemented but can be derived with user-defined functions. Multiple specifications must be split into two statements unless the left-hand assignment is to a quad. This also applies to implied multiple specifications. Reduction and reshape (p) are not permitted for empty arguments; the argument of add/drop may not be scalar; empty indices are not permitted. A quad (q) can't be typed in response to a quad (nor can the name of a function which itself gets input from a quad). Quote-quad (m) is permitted. No more than 32 user functions can be defined in a single workspace and a function may not contain more than 255 lines.
A comment (c) must occupy a separate line: a comment can't follow a function statement on the same line.
In the tape version, arrays are limited to five (5) dimensions.

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585

We make it easy to see what numbers really say.

PERSONAL SOFTWARE INC.™

VISIPLOT

The VisiPlot™ program adds a striking new visual dimension to all your financial analysis, forecasting, budgeting and business planning.

VisiPlot takes all those rows and columns of hard-to-grasp numbers and automatically turns them into easy-to-comprehend charts and graphs, right before your eyes.

You get "presentation quality" graphs and charts that'll aid you in understanding business information, and add clarity to reports and proposals.

And you don't need to know any computer programming to put VisiPlot to work for you. A simple "menu" selection helps you through the easy steps, step-by-step. You enter data from the keyboard, your personal computer performs all the calculations and displays your graph or chart. You can save your charts for future display, or print them out on your system's graphic printer.

VisiPlot makes it fast and easy to produce all kinds of charts and graphs:

Line charts, single or multiple.
Bar charts, single, side-by-side or stacked.
Area graphs.
High-Low graphs.
Scatterplot charts.
Pie charts.

With VisiPlot, you can "overlay" graphs for more than one data series. You can use foreground and background shading for differentiation (or color, if your system offers it). You can "split" bar graphs with VisiPlot for side-by-side comparisons; such as this year and last year, same months. You can even display two charts or graphs at the same time, each with one or more data series. For example, display a high-low plot of your favorite stock along with a bar chart of trading volume.

VisiPlot automatically generates plotting symbols and legends, automatic x, y axis dimensions, horizontal and vertical grids, and the flexibility of both fixed and movable titles.

VisiPlot turns your computer into a powerful tool for perceiving relationships—not just calculating them. Complex business charts and graphs become easy to produce, easy to revise.

VisiPlot can also interchange data with other Personal Software programs, such as VisiCalc, multiplying your calculating, planning, and graphic display capabilities.

VisiPlot is available for Apple II and II Plus personal computers.

Ask your Personal Software dealer to show you how easily VisiPlot can show you "the big picture."

Visi Plot
#47-204001D $179.95

VISITREND/VISIPLOT

If you want sophisticated trend forecasting and statistical analysis—and you know how hard it is to create them by hand—you'll want to get your hands on the VisiTrend/VisiPlot™ program!

VisiTrend/VisiPlot responds to the demanding needs of the sophisticated manager, planner, investor, analyst, statistician, consultant. You'll be able to accomplish more because you decide on your time series analysis, and VisiTrend/VisiPlot performs it for you. So you can spend your time examining various alternatives, not figuring them out.

You can perform regressions on data series to look for relationships. Make transformations on the data like lags, leads, percent change, moving averages and exponential smoothing. Or write arbitrary formulas to calculate new series based on several existing ones. With one simple command, you can get descriptive statistics on any data series including mean, variance, standard deviation, minimum and maximum.

And with VisiTrend/VisiPlot's powerful graphics capability, you'll be able to easily produce charts and graphs that make it so much easier to see how one series relates to another. You get everything VisiPlot provides —line charts, area charts, bar charts, high-low graphs, scatter graphs, and pie charts, too.

And you don't need to know any computer programming to put VisiTrend/VisiPlot to work for you. A simple "menu" selection helps you through entering your data, modifying it, displaying it as a chart or graph, or calculating new series from other series.

You can save all your time series data, charts and graphs for future display, or print them out on your system's printer for reports and proposals.

VisiTrend/VisiPlot can also exchange data with other Personal Software programs. Develop a product sales projection with VisiCalc, do a regression analysis with VisiTrend and then create a graph of the industry and product figures with VisiPlot. Or put the data together in other ways only you can think of.

VisiTrend/VisiPlot is available for Apple II and II Plus personal computers.

To see just how very useful VisiTrend/VisiPlot could be for you in sophisticated analysis and projection, ask your Personal Software dealer to give you a demonstration.

Visi Trend/Visi Plot
#47-204002D $259.95

VISITERM

With the VisiTerm™ program, you and your personal computer can communicate with larger computers, with other personal computers, and send and receive disk file information over the phone.

You'll be able to tap into time-sharing services, newswires, data bases, stock reports and more. You'll be able to access your company's own dp computer, if there is one, for data entry, proprietary software, and internal communications.

#47-204003D Visi Term $149.95

VISICALC

The VisiCalc™ program is the most widely-used personal computing program in the world today!

That's because VisiCalc is the easiest to use, most versatile way for you to "run the numbers" when evaluating any financial or business decision.

#47-207010D Visicalc $199.95

VISIDEX

The VisiDex™ program makes it a snap to remember anything, and everything, you don't want to forget.

VisiDex imposes no restrictions on the format or content of your data. So you can file away any kind of information you want to: names and addresses, important numbers and dates, ideas, things-to-do, lists, notes, report and memo highlights, stock, tax, and personal information.

#47-204004D Visi Dex $199.95

DESKTOP/PLAN II

You can produce a high-quality, professionally organized print-out of your plan or model with Desktop/PLAN II. The format is much like an accountant's columnar analysis pad—it can be up to 18 columns wide or up to 300 rows deep. Pagination, headers which carry your model's title, and other report features are handled automatically by the program.

And Desktop/PLAN II can receive information from VisiCalc—an interchange capability giving you greater combined usefulness.

#47-207011D Desktop/Plan II $199.95

TERMS: Prices and specifications are subject to change. **TSE HARDSIDE** accepts VISA & MASTERCARD, Certified checks and Money Orders. Personal checks accepted (takes 3 weeks to clear). **TSE HARDSIDE pays all shipping charges (within the 48 states) on all PREPAID orders over $100.00.** On all orders under $100 a $2.50 handling charge must be added. COD orders accepted (orders over $250 require 25%deposit), there is a $5.00 handling charge. UPS Blue Label and Air Freight available at extra cost.

6 South St., Milford, NH 03055 (603) 673-5144

TOLL FREE OUT-OF-STATE
1-800-258-1790

FLIP-IT

BY MICHAEL PRESCOTT

S-80 and Atari Translation by ALAN J. ZETT

FLIP-IT is an Apple (Integer BASIC) and S-80 program requiring 16K memory; Atari program requires 24K memory.

This is a computerized board game, in which you and the computer match wits trying to outflank and capture one another's pieces on an eight-by-eight board. The computer is a formidable opponent, and you won't find it a trivial matter to win.

The game begins with a square arrangement of four chips in the center of the board, two belonging to you and two belonging to the computer. You are allowed to choose your color and who will play first. When it is your turn, your object is to pick an unoccupied square such that putting one of your chips there will "outflank" one or more of the computer's chips. This means that the computer's chips would be sandwiched in between one of your existing chips and the new one you're playing, in a straight line. (You might think of it as your chips being bookends at each end of a row of your opponent's pieces.)

When you do this, all the computer's pieces which you have outflanked will become yours. This can happen in more than one direction, so that in any given turn you might capture pieces horizontally, vertically, and diagonally. This can result in a substantial shift in the relative number of chips in one turn, and the outcome of the game is rarely certain until the last few moves.

The method of entering your chosen move is different for each of the three computers. The Apple original uses Paddle 0 to position a cursor on the board, and pressing the button on the paddle registers your move. The Atari version uses the joystick to move the cursor horizontally, vertically, or diagonally from its current location, and again pressing the button enters the move. The S-80 version has all the squares numbered on the screen, and you simply type in the number of the square you want. The computer always checks to see that your move is legal, and will not allow cheating!

During your turn, you may ask the computer to recommend your best move by pressing "B." If you see no move that you can make, you must press "N;" the computer will then check to see if you really have no possible move, and if so it will continue with its play. (Before pressing "B" or "N," you should move the cursor to an already occupied square.) If the computer has no move, it will pass after searching all the open squares (the Atari version will indicate "I pass" with what might be described as a "sighing" sound). If neither of you has any possible move (which occasionally happens even before the board is filled), then the game is over and the player with the greater number of chips is the winner. You may also press "Q" to quit at any time.

The original Apple version has a provision to save a game in progress if you should want to do so. The necessary data is stored in memory locations 300-364 hex. This information can be saved to tape following the Quit command by entering the monitor with a CALL 151, typing in 300.364W, starting the tape recorder in the recording mode, and pressing RETURN. Following the beep, the data has been saved. The information can be reloaded at the beginning of a new game by choosing the manual set-up option. You then enter the monitor, type 300.364R, start the tape in the play mode, and hit RETURN. After the beep, type CTRL-C and RETURN, then GOTO RESTART. The data can also be saved on disk using BSAVE (filename), A$300, L$65, and recalled using BLOAD (filename).

You will soon find that winning a game involves more than just capturing as many pieces as you can on any given turn. Much more important to the eventual results will be the position of your chips on the board. Capturing edge and (especially) corner squares, and preventing the computer from doing the same, will pay off in the long run, even if it means outflanking only one square when you could get more elsewhere on the board.

VARIABLES

Apple and Atari versions:

A: Paddle position
A(*): Decision-making array for computer's moves.
A$: Data manipulator.
B: Decision pointer.
C: Capture pointer.
CHIP: On-the-fly token.
COLR: Sampler variable.
COMMD: Key input variable
D: Loop variable.
HOME: Apple's clear-screen address.
IP: 'I pass' flag.
OC: Token for computer's chip.
OPPONENT: Color of computer's chips.
P1: Number of chips captured.
P2: Number of chips on board.
P4: Sum of chips captured.
Q: Substitution variable.
RESTART: Re-entry line number.
SY: Number of your chips on board.
X,Y,Z: Loop variables.
X1,Y1: Point under consideration.
X2,X3: Check loop variables.
XX,YY: Plot variables.
XXX: Capture loop variable.
YC: Token for your chip.
YOUR: Color of your chips.
YP: 'You pass' flag.

S-80 Version:
Note that some of the PRINTs in the S-80 version include lower-case characters. These are for the benefit of those who may have lower-case available; if you don't have this modification, then simply treat all the characters as upper-case.

A(*): Decision-making array for computer's moves.
A$: General purpose (mainly INPUTs).
B(*): Array representing board layout. (Contents: 9 represents off-board, 0 is unoccupied square, 1 is occupied by white, -1 is occupied by black.)
B$: Graphics string for a Black piece.
BL: Black piece value in board array (=-1).
BP: Number of black pieces on board.
C: Value of the color (black=-1, white=1) calling the 'legal check' routine.
F: At start of game, F=1 if you choose to go first. During game, used as a flag to flip pieces.
FL: If FL=0 then legal check will not flip pieces even if a legal square was found.
IP: 'I pass' flag for computer.
LG: Indicates legal move if =1.
OP: Holds the value of the computer's color (-1 or 1).
OP$: Graphics string for the computer's pieces.
P: Temporary holding variable for PDL.
P2: Total number of pieces on the board.
PDL: Position of a square under consideration (1st digit is row, 2nd is column).
RF: Return flag; if =1 then RETURN from a normal (non-GOSUB) routine.
SC: Value of the color being searched for in the 'legal check' routine.
W$: Graphics string for a white piece.
WH: White piece value for use in the board array (=1).
WP: Number of white pieces on the board.
X,Y,Z: Miscellaneous.
YO: Holds the value of a player's color (-1 or 1).
YO$: Graphics string for your pieces.
YP: 'You pass' flag.

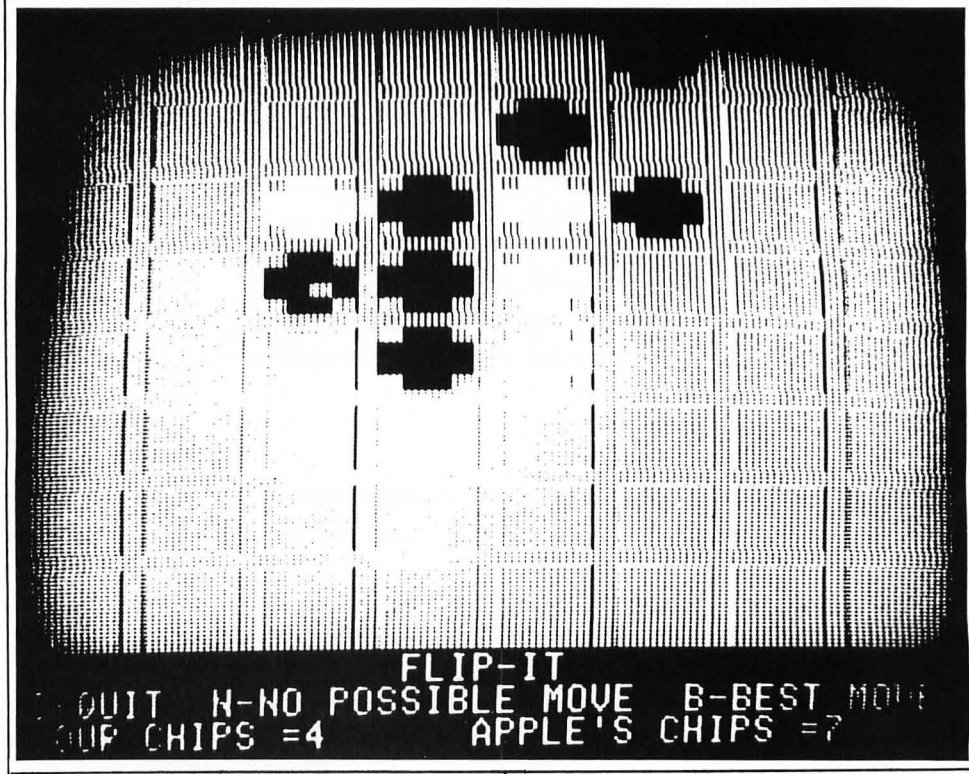
Apple Version

Initialization.

```
100 DIM A(60),A$(100):Z=1: GOSUB
    6020: FOR X=1 TO LEN(A$) STEP
    2:A(Z)= ASC(A$(X,X))-176+( ASC(
    A$(X+1,X+1))-176)*10:Z=Z+1:
     NEXT X: GOSUB 5000
110 GOSUB 6030: FOR X=1 TO LEN(
    A$) STEP 2:A(Z)= ASC(A$(X,X)
    )-176+( ASC(A$(X+1,X+1))-176
    )*10:Z=Z+1: NEXT X
120 POKE -16368,0:B=C=D:HOME=-936
    :YOUR=0:OPPONENT=15:P2=4:SY=
    2:RESTART=4030
```

Graphic introduction.

```
150 TEXT : GR : CALL HOME: TAB
    (17): PRINT "FLIP-IT": POKE
    34,21
160 FOR Y=0 TO 22: COLOR=Y: IF
    Y MOD 2=0 THEN 180
170 FOR X=0 TO 19: VLIN 0,39 AT
    X: VLIN 0,39 AT 39-X: NEXT
    X: GOTO 190
180 FOR X=0 TO 19: HLIN 0,39 AT
    X: HLIN 0,39 AT 39-X: NEXT
    X
```




```
190 NEXT Y: COLOR=12: FOR X=1 TO
    7: VLIN 0,39 AT 5*X-1: HLIN
    0,39 AT 5*X-1: NEXT X
```

Get choice of color, and clear portion of memory which keeps track of game progress.

```
200 CALL HOME: INPUT "WHICH COLOR (W
    ,B)",A$: IF A$="" THEN 210:
     IF A$(1,1)#"W" THEN 210:YOUR=
    15:OPPONENT=0:YC=32:OC=48: GOTO
    220
```

```
210 YOUR=0:OPPONENT=15:YC=48:OC=
    32
220 D=768: FOR X=0 TO 1: GOSUB
    6000+X*10: FOR Y=1 TO LEN(A$
    ) STEP 2: POKE D, ASC(A$(Y,
    Y))-176+10*( ASC(A$(Y+1,Y+1
    ))-176):D=D+1: NEXT Y,X
230 XX=2:YY=2: COLOR=9: PLOT XX,
    YY:COLR=6
```

Prompt for type of game setup.

```
240 CALL HOME: INPUT "WANT TO SET UP
     YOUR OWN GAME?",A$: IF A$=
    "" THEN 600: IF A$(1,1)="Y"
     THEN 4000
```

Set up new game.

```
600 COLOR=0:X=45:CHIP=48: GOSUB
    1160:X=54: GOSUB 1160: COLOR=
    15:X=44:CHIP=32: GOSUB 1160
    :X=55: GOSUB 1160
```

Print text header.

```
610 POKE 34,21: CALL HOME: PRINT
    "Q-QUIT  N-NO POSSIBLE MOVE  B-B
    EST MOVE": POKE 34,22
```

Choose who goes first.

```
620 CALL HOME: INPUT "WANT TO GO FIR
    ST?",A$: IF A$="" THEN 630:
     IF A$(1,1)="N" THEN 840
```

If board is filled then end; otherwise update the chip selector cursor.

```
630 IF P2=64 THEN 2010: GOSUB 7000
```

Get command, or update chip-selector cursor.

```
640 IF PEEK (-16384)<128 THEN 800
    :COMMND= PEEK (-16384): POKE
    -16368,0: IF COMMND<>209 AND
    COMMND<>206 AND COMMND<>194
     THEN 800
650 IF COMMND=209 THEN 660: IF
    COMMND=206 THEN 680: IF COMMND=
    194 THEN GOSUB 700: GOTO 800
```

End-game command.

```
660 POKE 34,22: CALL HOME: INPUT
    "WANT TO END GAME?",A$: IF
    A$="" THEN 800: IF A$(1,1)#
    "Y" THEN 800: TEXT : CALL HOME:
     VTAB 8
670 PRINT "IF YOU WANT TO SAVE THIS
    GAME SET UP": PRINT : PRINT
    "USE A DISK OR CASSETTE TO SAVE"
    : PRINT "DATA AREA A$300,L$65 OR
     300.364W": END
```

No-possible-move command.

```
680 PRINT : PRINT "LET ME CHECK."
    ;: GOSUB 700: IF YP=1 THEN
    840: PRINT : PRINT "I FOUND ONE!
    ";: FOR X=1 TO 800: NEXT X:
     CALL HOME: GOTO 800
```

Find best possible move.

```
700 FL=1:Q=YOUR:YOUR=OPPONENT:OPPONE
    NT=Q: GOSUB 2000: IF YP=1 THEN
    710
705 FOR XXX=1 TO 30: IF XXX MOD
    2=0 THEN COLOR=9: IF XXX MOD
    2 THEN COLOR=6: IF XXX MOD
    5 THEN GOSUB 1160: NEXT XXX
710 POKE 768+X,0:Q=YOUR:YOUR=OPPONEN
    T:OPPONENT=Q:FL=0: RETURN
```

Position and display player's move.

```
800 IF A=63* PDL (0)/255 THEN 810
    :A=63* PDL (0)/255: COLOR=COLR:
     PLOT XX,YY: COLOR=9:XX=5*(
    A MOD 8)+2:YY=A/8*5+2:COLR=
     SCRN(XX,YY-1): PLOT XX,YY
810 IF PEEK (-16287)<128 THEN 640
    : IF COLR<>6 THEN 640
820 P3=SY:CHIP=YC: COLOR=YOUR:P4=
    0:D=0: FOR XXX=1 TO 8:X=A MOD
    8+1+10*(A/8+1):P1=0: GOSUB
    980+XXX*20:P4=P4+P1: IF P1 THEN
    D=D+1
830 NEXT XXX: IF D=0 THEN 640:SY=
    SY+P4-D+1:P2=P2+1
```

Do computer's move.

```
840 GOSUB 7000: GOSUB 2000: FOR
    XXX=1 TO 300: IF XXX MOD 2=
    0 THEN COLOR=9: IF XXX MOD
    2 THEN COLOR=6: IF XXX MOD
    21=0 THEN GOSUB 1160: NEXT
    XXX: COLOR=OPPONENT
900 CHIP=OC:P2=P2+1:Q=YOUR:YOUR=
    OPPONENT:OPPONENT=Q:P4=0:D=
    0: FOR XXX=1 TO 8:X=A(B):P1=
    0: GOSUB 980+20*XXX: IF P1 THEN
    D=D+1:P4=P4+P1
910 NEXT XXX:SY=SY-P4+D
920 Q=YOUR:YOUR=OPPONENT:OPPONENT=
    Q:COLR= SCRN(XX,YY-1): GOTO
    630
```

Capture routine (spaced every twenty lines).

```
1000 X1=5*(X MOD 10)-3:Y1=X/10*5
     -3: FOR X2=1 TO 7:Y1=Y1-5: IF
     Y1<0 THEN RETURN : IF SCRN(
     X1,Y1)=YOUR AND X2=1 OR SCRN(
     X1,Y1)=6 THEN RETURN
1010 IF SCRN(X1,Y1)=OPPONENT THEN
     NEXT X2: IF X2=8 THEN RETURN
     : GOSUB 1160:P1=X2: FOR X3=
     1 TO X2:X=X-10: GOSUB 1160:
      NEXT X3: RETURN
1020 X1=5*(X MOD 10)-3:Y1=X/10*5
     -3: FOR X2=1 TO 7:Y1=Y1+5: IF
     Y1>39 THEN RETURN : IF SCRN(
     X1,Y1)=YOUR AND X2=1 OR SCRN(
     X1,Y1)=6 THEN RETURN
1030 IF SCRN(X1,Y1)=OPPONENT THEN
     NEXT X2: IF X2=8 THEN RETURN
     : GOSUB 1160:P1=X2: FOR X3=
     1 TO X2:X=X+10: GOSUB 1160:
      NEXT X3: RETURN
1040 X1=5*(X MOD 10)-3:Y1=X/10*5
     -3: FOR X2=1 TO 7:X1=X1-5: IF
     X1<0 THEN RETURN : IF SCRN(
     X1,Y1)=YOUR AND X2=1 OR SCRN(
     X1,Y1)=6 THEN RETURN
1050 IF SCRN(X1,Y1)=OPPONENT THEN
     NEXT X2: IF X2=8 THEN RETURN
     : GOSUB 1160:P1=X2: FOR X3=
     1 TO X2:X=X-1: GOSUB 1160: NEXT
     X3: RETURN
1060 X1=5*(X MOD 10)-3:Y1=X/10*5
     -3: FOR X2=1 TO 7:X1=X1+5: IF
     X1>39 THEN RETURN : IF SCRN(
     X1,Y1)=YOUR AND X2=1 OR SCRN(
     X1,Y1)=6 THEN RETURN
1070 IF SCRN(X1,Y1)=OPPONENT THEN
     NEXT X2: IF X2=8 THEN RETURN
     : GOSUB 1160:P1=X2: FOR X3=
     1 TO X2:X=X+1: GOSUB 1160: NEXT
     X3: RETURN
1080 X1=5*(X MOD 10)-3:Y1=X/10*5
     -3: FOR X2=1 TO 7:X1=X1-5:Y1=
     Y1-5
1085 IF X1<0 OR Y1<0 THEN RETURN
     : IF SCRN(X1,Y1)=YOUR AND X2=
     1 OR SCRN(X1,Y1)=6 THEN RETURN
1090 IF SCRN(X1,Y1)=OPPONENT THEN
     NEXT X2: IF X2=8 THEN RETURN
     : GOSUB 1160:P1=X2: FOR X3=
     1 TO X2:X=X-11: GOSUB 1160:
      NEXT X3: RETURN
1100 X1=5*(X MOD 10)-3:Y1=X/10*5
     -3: FOR X2=1 TO 7:X1=X1+5:Y1=
```

continued on next page

continued from previous page

```
     Y1-5
1105 IF X1>39 OR Y1<0 THEN RETURN
     : IF SCRN(X1,Y1)=YOUR AND X2=
     1 OR SCRN(X1,Y1)=6 THEN RETURN

1110 IF SCRN(X1,Y1)=OPPONENT THEN
     NEXT X2: IF X2=8 THEN RETURN
     : GOSUB 1160:P1=X2: FOR X3=
     1 TO X2:X=X-9: GOSUB 1160: NEXT
     X3: RETURN
1120 X1=5*(X MOD 10)-3:Y1=X/10*5
     -3: FOR X2=1 TO 7:X1=X1+5:Y1=
     Y1+5
1125 IF X1>39 OR Y1>39 THEN RETURN
     : IF SCRN(X1,Y1)=YOUR AND X2=
     1 OR SCRN(X1,Y1)=6 THEN RETURN

1130 IF SCRN(X1,Y1)=OPPONENT THEN
     NEXT X2: IF X2=8 THEN RETURN
     : GOSUB 1160:P1=X2: FOR X3=
     1 TO X2:X=X+11: GOSUB 1160:
      NEXT X3: RETURN
1140 X1=5*(X MOD 10)-3:Y1=X/10*5
     -3: FOR X2=1 TO 7:X1=X1-5:Y1=
     Y1+5
1145 IF X1<0 OR Y1>39 THEN RETURN
     : IF SCRN(X1,Y1)=YOUR AND X2=
     1 OR SCRN(X1,Y1)=6 THEN RETURN

1150 IF SCRN(X1,Y1)=OPPONENT THEN
     NEXT X2: IF X2=8 THEN RETURN
     : GOSUB 1160:P1=X2: FOR X3=
     1 TO X2:X=X+9: GOSUB 1160: NEXT
     X3: RETURN
```

Draw chip at selected location.

```
1160 X1=5*(X MOD 10)-5:Y1=X/10*5
     -5: VLIN Y1+1,Y1+2 AT X1: VLIN
     Y1,Y1+3 AT X1+1: VLIN Y1,Y1+
     3 AT X1+2: VLIN Y1+1,Y1+2 AT
     X1+3
1165 POKE 768+X,CHIP: FOR Z=1 TO
      3: POKE -16336, PEEK (-16336
     ): NEXT Z: RETURN
```

Step through the best moves for the computer. If none are found then pass or concede game.

```
2000 IF P2-SY=0 OR SY=0 THEN 2010
     : IF FL=0 THEN IP=0: IF FL=
     1 THEN YP=0
2001 FOR B=1 TO 60:X=A(B):X1=5*(
     X MOD 10)-3:Y1=X/10*5-3: POKE
     -16336, PEEK (-16336): IF SCRN(
     X1,Y1)=6 THEN GOSUB 3000: NEXT
     B
2005 IF P2-SY<>0 AND SY<>0 AND FL=
     0 AND YP=0 THEN 2006: IF P2-
     SY<>0 AND SY<>0 AND FL=1 AND
     IP=0 THEN 2007: GOTO 2010
2006 IP=1: VTAB 24: TAB 15: PRINT
     "I MUST PASS";: FOR X=1 TO
     1200: NEXT X: VTAB 24: TAB
     1: CALL -958: GOTO 630
2007 YP=1: RETURN
2010 PRINT : PRINT "NO POSSIBLE MOVES
      LEFT!";: FOR X=1 TO 1500: NEXT
     X: PRINT : IF P2-SY>SY THEN
     2030: IF P2-SY=SY THEN 2040
2020 PRINT "YOU WON THIS ROUND BY "
     ;2*SY-P2;" CHIPS.";: GOTO 2050
2030 PRINT "I ONCE AGAIN PROVE MY SUP
     ERIORITY";: FOR X=1 TO 1500
     : NEXT X: PRINT : PRINT "I WON B
     Y ";P2-2*SY;" CHIPS.";: GOTO
     2050
2040 PRINT "IT'S A DRAW, I'LL BEAT YO
     U NEXT TIME";: GOTO 2050
2050 FOR X=1 TO 10: POKE -16336,
      PEEK (-16336): NEXT X: FOR
     X=1 TO 1500: NEXT X: POKE 34
     ,20: CALL HOME: PRINT "WANT TO P
     LAY AGAIN?"
2060 IF PEEK (-16384)<128 THEN 2060
     : IF PEEK (-16384)=217 THEN
     120: POKE -16368,0: TEXT : CALL
     HOME: END
```

The computer has found an empty square; now check to see if it is a legal move.

```
3000 FOR C=1 TO 8:X1=5*(X MOD 10
     )-4:Y1=X/10*5-3: POKE -16336
      , PEEK (-16336): GOSUB 2990
      +20*C: NEXT C: RETURN
```

Check legality of computer's moves (up-left, up-right, etc.). If it is a legal move then jump out of loop and do capture.

```
3010 FOR D=1 TO 7:Y1=Y1+5: IF Y1>
     39 THEN RETURN : IF ( SCRN(
     X1,Y1)=OPPONENT AND D=1) OR
     SCRN(X1,Y1)=6 THEN RETURN :
      IF SCRN(X1,Y1)=YOUR THEN NEXT
     D
3020 IF D=8 THEN RETURN : POP : POP
     : GOTO 5150
3030 FOR D=1 TO 7:Y1=Y1-5: IF Y1<
     0 THEN RETURN : IF ( SCRN(X1,
     Y1)=OPPONENT AND D=1) OR SCRN(
     X1,Y1)=6 THEN RETURN : IF SCRN(
     X1,Y1)=YOUR THEN NEXT D
3040 IF D=8 THEN RETURN : POP : POP
     : GOTO 5150
3050 FOR D=1 TO 7:X1=X1-5: IF X1<
     0 THEN RETURN : IF ( SCRN(X1,
     Y1)=OPPONENT AND D=1) OR SCRN(
     X1,Y1)=6 THEN RETURN : IF SCRN(
     X1,Y1)=YOUR THEN NEXT D
3060 IF D=8 THEN RETURN : POP : POP
     : GOTO 5150
3070 FOR D=1 TO 7:X1=X1+5: IF X1>
     39 THEN RETURN : IF ( SCRN(
     X1,Y1)=OPPONENT AND D=1) OR
     SCRN(X1,Y1)=6 THEN RETURN :
      IF SCRN(X1,Y1)=YOUR THEN NEXT
     D
3080 IF D=8 THEN RETURN : POP : POP
     : GOTO 5150
3090 FOR D=1 TO 7:X1=X1+5:Y1=Y1+
     5: IF X1>39 OR Y1>39 THEN RETURN
     : IF SCRN(X1,Y1)=6 OR SCRN(
     X1,Y1)=OPPONENT AND D=1 THEN
     RETURN
3100 IF SCRN(X1,Y1)=YOUR THEN NEXT
     D: IF D=8 THEN RETURN : POP
     : POP : GOTO 5150
3110 FOR D=1 TO 7:X1=X1-5:Y1=Y1+
     5: IF X1<0 OR Y1>39 THEN RETURN
     : IF SCRN(X1,Y1)=6 OR SCRN(
     X1,Y1)=OPPONENT AND D=1 THEN
     RETURN
3120 IF SCRN(X1,Y1)=YOUR THEN NEXT
     D: IF D=8 THEN RETURN : POP
     : POP : GOTO 5150
3130 FOR D=1 TO 7:X1=X1-5:Y1=Y1-
     5: IF X1<0 OR Y1<0 THEN RETURN
     : IF SCRN(X1,Y1)=6 OR SCRN(
     X1,Y1)=OPPONENT AND D=1 THEN
     RETURN
3140 IF SCRN(X1,Y1)=YOUR THEN NEXT
     D: IF D=8 THEN RETURN : POP
     : POP : GOTO 5150
3150 FOR D=1 TO 7:X1=X1+5:Y1=Y1-
     5: IF X1>39 OR Y1<0 THEN RETURN
     : IF SCRN(X1,Y1)=6 OR SCRN(
     X1,Y1)=OPPONENT AND D=1 THEN
     RETURN
3160 IF SCRN(X1,Y1)=YOUR THEN NEXT
     D: IF D=8 THEN RETURN : POP
     : POP : GOTO 5150
```

Set up an old game.

```
4000 CALL HOME: PRINT "HOW DO YOU WAN
     T TO SET IT UP?": PRINT "1-REGUL
     AR 2-MANUAL 3-TAPE/DISK":P2=
     0:SY=0
4010 IF PEEK (-16384)<128 THEN 4010
     :COMMND= PEEK (-16384)-176:
      POKE -16368,0: IF COMMND<1
      OR COMMND>3 THEN 4010
```

```
4020 IF COMMND=1 THEN 4250: IF COMMND
     =2 THEN 4050: CALL HOME: PRINT
     "LOAD DATA INTO MEMORY $300.364"
     : PRINT "THEN DO A 'GOTO RESTART
     '.";: END

4030 CALL HOME: FOR Y=1 TO 8: FOR
     A=1 TO 8: COLOR=6: IF PEEK
     (768+10*Y+A)=32 THEN COLOR=
     15: IF PEEK (768+10*Y+A)=48
      THEN COLOR=0:X=10*Y+A: GOSUB
     1160:Y1=Y1+1
4040 IF SCRN(X1,Y1)=15 OR SCRN(X1,
     Y1)=0 THEN P2=P2+1: IF SCRN(
     X1,Y1)=YOUR THEN SY=SY+1: NEXT
     A,Y:COLR= SCRN(XX,YY-1): GOTO
     610
4050 CALL HOME: PRINT "PRESS (ESC) WH
     EN FINISHED": PRINT "USE BUTTON
     TO SELECT SQUARE"
4060 A=63* PDL (0)/255: COLOR=COLR:
      PLOT XX,YY: COLOR=9:XX=5*(
     A MOD 8)+2:YY=A/8*5+2:COLR=
      SCRN(XX,YY-1): PLOT XX,YY
4070 IF PEEK (-16384)<128 THEN 4080
     :COMMND= PEEK (-16384): POKE
     -16368,0: IF COMMND=155 THEN
     4160
4080 IF PEEK (-16287)<128 THEN 4060
     : CALL HOME
4090 CALL HOME: INPUT "WHICH COLOR? (
     W,B)",A$: IF A$="" THEN 4090
     : IF A$(1,1)#"W" AND A$(1,1
     )#"B" THEN 4090:X=A MOD 8+1
     +10*(A/8+1)
4100 COLOR=0: IF A$(1,1)="W" THEN
     COLOR=15: GOSUB 4240:Y1=Y1+
     1:B=0: IF X1>4 THEN B=B+( SCRN(
     X1-5,Y1)<>6)
4110 IF X1<35 THEN B=B+( SCRN(X1+
     5,Y1)<>6): IF Y1>4 THEN B=B+
     ( SCRN(X1,Y1-5)<>6): IF Y1<
     35 THEN B=B+( SCRN(X1,Y1+5)
     <>6)
4120 IF X1>4 AND Y1>4 THEN B=B+(
      SCRN(X1-5,Y1-5)<>6): IF X1>
     4 AND Y1<35 THEN B=B+( SCRN(
     X1-5,Y1+5)<>6)
4130 IF X1<35 AND Y1>4 THEN B=B+
     ( SCRN(X1+5,Y1-5)<>6): IF X1<
     35 AND Y1<35 THEN B=B+( SCRN(
     X1+5,Y1+5)<>6): IF B=0 THEN
     4150
4140 GOSUB 1160:P2=P2+1: POKE 768
     +X,32: IF A$(1,1)="B" THEN
     POKE 768+X,48:COLR= SCRN(XX,
     YY-1): GOTO 4050
4150 IF P2=0 THEN 4140: CALL HOME:
      GOSUB 4220: PRINT "YOU'RE MAKIN
     G UP THIS GAME!": PRINT "A PIECE
      COULDN'T GET THERE": FOR X=
     1 TO 800: NEXT X: GOTO 4050
4160 X=44: GOSUB 4240:Y1=Y1+1: IF
     SCRN(X1,Y1)=6 THEN 4230:X=45
     : GOSUB 4240:Y1=Y1+1: IF SCRN(
     X1,Y1)=6 THEN 4230
4170 X=54: GOSUB 4240:Y1=Y1+1: IF
     SCRN(X1,Y1)=6 THEN 4230:X=55
     : GOSUB 4240:Y1=Y1+1: IF SCRN(
     X1,Y1)=6 THEN 4230:P2=0: FOR
     Y=1 TO 8: FOR A=1 TO 8
4180 X=10*Y+A: GOSUB 4240:Y1=Y1+
     1:X1=X1+1: IF SCRN(X1,Y1)=6
      THEN 4190:P2=P2+1: IF SCRN(
     X1,Y1)=YOUR THEN SY=SY+1
4190 NEXT A,Y: IF P2-SY=0 OR SY=
     0 THEN 4200: IF P2=64 THEN
     4210: COLOR= SCRN(XX,YY-1):
      GOTO 610
4200 GOSUB 4220: CALL HOME: PRINT
     "YOU HAVE TO GIVE ME A CHANCE!"
     :P2=0:SY=0: FOR X=1 TO 800:
      NEXT X: GOTO 4050
4210 GOSUB 4220: CALL HOME: PRINT
     "YOU DIDN'T LEAVE ANY ROOM!"
     : FOR X=1 TO 800: NEXT X: GOTO
     4050
4220 FOR X=1 TO 100: POKE -16336
     , PEEK (-16336): NEXT X: RETURN
4230 CALL HOME: GOSUB 4220: PRINT
     "THE CENTRAL SQUARES MUST BE FIL
     LED": FOR X=1 TO 1500: NEXT
     X: GOTO 4050
4240 X1=5*(X MOD 10)-5:Y1=X/10*5
     -5: RETURN
4250 P2=4:SY=2: GOTO 600
```

Print instructions for game.

```
5000 TEXT : CALL -936: VTAB 12: INPUT
     "NEED INSTRUCTIONS?",A$: IF
     A$#"Y" THEN RETURN
5010 CALL -936: VTAB 2: TAB 16: PRINT
     "OTHELLO": PRINT : PRINT "THE OB
     JECT OF THIS GAME IS TO COMPLETE
     LY"
5020 PRINT "FILL THE BOARD WITH AS MA
     NY PIECES OF": PRINT : PRINT
     "YOUR COLOR AS YOU CAN.  TO DO T
     HIS YOU": PRINT
5030 PRINT "OUTFLANK YOUR OPPONENT'S
     PIECES AND FLIP": PRINT "THEM TO
      YOUR COLOR.  OUTFLANKING CAN"
     : PRINT
5040 PRINT "OCCUR HORIZONTALLY, VERTI
     CALLY, OR DIAG-": PRINT "ONALLY.
      THE GAME ENDS WHEN THE BOARD IS
     ": PRINT
5050 PRINT "FULL OR A PLAYER CANNOT M
     AKE A MOVE.": PRINT : PRINT
     "WHOEVER HAS THE MOST PIECES WIN
     S."
5060 VTAB 24: PRINT "    (PRESS SPACE
      BAR TO CONTINUE)";
5065 IF PEEK (-16384)<128 THEN 5065

5070 POKE -16368,0: CALL -936: VTAB
     2: PRINT "YOU MAKE YOUR MOVE BY
     TURNING PADDLE 0": PRINT
5080 PRINT "UNTIL THE CURSOR IS IN TH
     E SQUARE YOU": PRINT : PRINT
     "WANT, AND THEN PRESSING THE BUT
     TON.": PRINT
5090 PRINT : PRINT "DURING PLAY YOU M
     AY ALSO CHOOSE ONE OF": PRINT
     : PRINT "THE FOLLOWING OPTIONS B
     Y PRESSING THE": PRINT : PRINT
     "KEY INDICATED:"
5100 PRINT : PRINT "   Q - QUIT THE G
     AME": PRINT : PRINT "   N - NOT
     ABLE TO MAKE A MOVE": PRINT
5110 PRINT "   B - ASK THE COMPUTER T
     O RECOMMEND": PRINT "       YOUR
      BEST MOVE.": PRINT
5120 VTAB 24: PRINT "    (PRESS SPACE
      BAR TO BEGIN)";
5130 IF PEEK (-16384)<128 THEN 5130
5140 POKE -16368,0
5150 X=A(B): RETURN
```

Data to set up new game in memory.

```
6000 A$="5I5I5I5I5I5I5I5I5I5I5I5I5I1000000
     0000000005I5I0000000000000000005I
     5I0000000000000000005I5I0000000000
     0000005I": RETURN
6010 A$="5I0000000000000000005I5I000000
     0000000000005I5I0000000000000000005I
     5I0000000000000000005I5I5I5I5I5I5I
     5I5I5I5I": RETURN
```

Data to set up square-choice priority for computer.

```
6020 A$="11811888316113831686386833633
     3666415114841585485843533464356 5
     4656425224742575475732622373267 6
     37672171": RETURN
6030 A$="12821787287822722777": RETURN
```

Chip-count update.

```
7000 POKE 34,22: CALL HOME: PRINT
     "YOUR CHIPS =";SY;: TAB (20
     ): PRINT "APPLE'S CHIPS =";
     P2-SY: POKE 34,23: RETURN
```

continued on next page

continued from previous page

Atari Version

```
10 REM ########################
20 REM #       FLIP-IT        #
30 REM # By Michael Prescott #
40 REM #                      #
50 REM #    ATARI Version     #
60 REM #   By Alan J. Zett    #
70 REM ########################
80 REM
```

Initialization.

```
100 GRAPHICS 0:DIM A(60),A$(120):Z=1:G
OSUB 6020:FOR X=1 TO LEN(A$) STEP 2:PD
L=0:B=0:FL=0
110 A(Z)=ASC(A$(X,X))-48+(ASC(A$(X+1,X
+1))-48)*10:Z=Z+1:NEXT X:GOSUB 5000:OP
EN #1,4,0,"K":PDL=0:B=0:F=0
120 POKE 764,255:B=C=D:P2=4:SY=2
```

Graphic introduction.

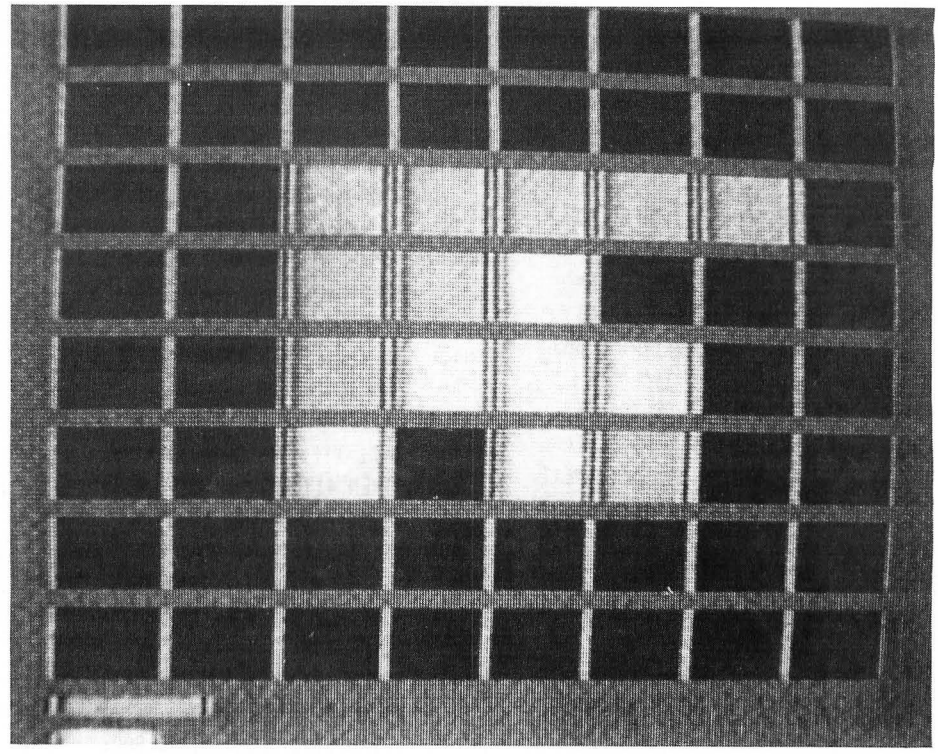
```
150 GRAPHICS 5:SETCOLOR 2,0,0:SETCOLOR
 4,0,0
160 FOR Y=0 TO 14 STEP 3:FOR D=0 TO 33
 STEP 0.2:SOUND 0,D,12,4:NEXT D:SOUND
0,0,0,4:IF (Y/2)=INT(Y/2) THEN 180
170 COLOR 2:SETCOLOR 1,Y,4:FOR X=12 TO
 39:PLOT X,0:DRAWTO X,38:PLOT 78-X,0:D
RAWTO 78-X,38:NEXT X:GOTO 190
180 COLOR 1:SETCOLOR 0,Y,4:FOR X=0 TO
19:PLOT 12,X:DRAWTO 66,X:PLOT 12,38-X:
DRAWTO 66,38-X:NEXT X
190 NEXT Y:SOUND 0,0,0,0:COLOR 0:FOR X
=1 TO 7:PLOT 7*X+11,0:DRAWTO 7*X+11,39
:PLOT 12,5*X-1:DRAWTO 67,5*X-1:NEXT X
```

Get choice of color, and clear portion of memory which keeps track of game progress.

```
200 SETCOLOR 1,0,10:PRINT CHR$(125);"W
HICH COLOR (RED/ BLUE) ";:TRAP 200:INP
UT A$:IF A$(1,1)<>"R" THEN 210
205 YOUR=2:OPPONENT=3:GOTO 230
210 YOUR=3:OPPONENT=2
230 XX=14:YY=2:COLR=1
```

Prompt for type of game setup and choose who goes first.

```
240 PRINT CHR$(125);"WANT TO GO FIRST"
;:TRAP 240:INPUT A$
245 IF A$(1,1)="Y" THEN F=1
250 PRINT CHR$(125);"WANT TO SET UP YO
UR OWN GAME";:TRAP 250:INPUT A$
255 IF A$(1,1)="Y" THEN SETCOLOR 4,0,4
:SETCOLOR 0,1,2:GOTO 4000
260 TRAP 33333:GOSUB 6000:GOTO 600
300 ST=STICK(0)
310 IF ST=15 THEN ST=0
320 IF ST=14 AND PDL>7 THEN PDL=PDL-8
```



```
330 IF ST=13 AND PDL<56 THEN PDL=PDL+8
340 IF ST=11 AND PDL>0 THEN PDL=PDL-1
350 IF ST=10 AND PDL>8 THEN PDL=PDL-9
360 IF ST=9 AND PDL<56 THEN PDL=PDL+7
370 IF ST=7 AND PDL<63 THEN PDL=PDL+1
380 IF ST=6 AND PDL>7 THEN PDL=PDL-7
390 IF ST=5 AND PDL<55 THEN PDL=PDL+9
400 RETURN
```

Set up new game.

```
600 COLOR 3:X=45:GOSUB 1160:X=54:GOSUB
 1160:COLOR 2:X=44:GOSUB 1160:X=55:GOS
UB 1160
610 POKE 764,255
620 IF F=0 THEN 840
```

If board is filled then end; otherwise update the chip-selecter cursor.

```
630 IF P2=64 THEN 2010
632 GOSUB 7000
```

Get command, or update chip selector cursor.

```
640 IF PEEK(764)=255 THEN 800
645 CO=PEEK(764):POKE 764,255:IF CO<>4
7 AND CO<>35 AND CO<>21 THEN 800
650 IF CO=47 THEN 2012
652 IF CO=35 THEN 680
654 GOSUB 700:GOTO 800
```

No-possible-move command.

```
680 GOSUB 700:IF YP=1 THEN 840
682 SOUND 0,255,12,15:FOR X=1 TO 150:N
EXT X:SOUND 0,0,0,0:GOTO 800
```

Find best possible move.

```
700 FL=1:Q=YOUR:YOUR=OPPONENT:OPPONENT
=Q:GOSUB 2000:IF YP=1 THEN 710
702 FOR XXX=1 TO 20:IF (XXX-INT(XXX/2)
*2)=0 THEN COLOR 0
704 IF (XXX-INT(XXX/2)*2)<>0 THEN COLO
R 1
706 IF (XXX-INT(XXX/5)*5)<>0 THEN GOSU
B 1160
708 NEXT XXX
710 Q=YOUR:YOUR=OPPONENT:OPPONENT=Q:FL
=0:RETURN
```

Position and display player's move.

```
800 GOSUB 300:IF A=PDL THEN 810
802 A=PDL:COLOR COLR:PLOT XX,YY:COLOR
0:XX=7*(A-INT(A/8)*8)+14:YY=INT(A/8)*5
+2
804 LOCATE XX,YY-1,COLR:PLOT XX,YY
810 IF STRIG(0)=1 THEN 640
812 IF COLR<>1 THEN 640
820 P3=SY:COLOR YOUR:P4=0:D=0:FOR XXX=
1 TO 8:X=(A-INT(A/8)*8)+1+10*(INT(A/8)
+1):P1=0:GOSUB 980+XXX*20
822 P4=P4+P1:IF P1<>0 THEN D=D+1
830 NEXT XXX:IF D=0 THEN 640
832 SY=SY+P4-D+1:P2=P2+1
```

Do computer's move.

```
840 GOSUB 7000:GOSUB 2000:FOR XXX=1 TO
 42:IF (XXX-INT(XXX/2)*2)=0 THEN COLOR
 0
842 IF (XXX-INT(XXX/2)*2)<>0 THEN COLO
R 1
844 IF (XXX-INT(XXX/3)*3)=0 THEN GOSUB
 1160
846 NEXT XXX:COLOR OPPONENT
```

```
900 P2=P2+1:Q=YOUR:YOUR=OPPONENT:OPPON
ENT=Q:P4=0:D=0:FOR XXX=1 TO 8:X=A(B):P
1=0:GOSUB 980+20*XXX
905 IF P1<>0 THEN D=D+1:P4=P4+P1
910 NEXT XXX:SY=SY-P4+D
920 Q=YOUR:YOUR=OPPONENT:OPPONENT=Q:LO
CATE XX,YY-1,COLR:GOTO 630
```

Capture routine (spaced every twenty lines).

```
1000 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:Y1=Y1-5:IF Y1<0
THEN RETURN
1005 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1010 IF Z=OPPONENT THEN NEXT X2
1012 IF X2=8 THEN RETURN
1014 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X-10:GOSUB 1160:NEXT X3:RETURN
1020 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:Y1=Y1+5:IF Y1>38
 THEN RETURN
1025 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1030 IF Z=OPPONENT THEN NEXT X2
1032 IF X2=8 THEN RETURN
1034 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X+10:GOSUB 1160:NEXT X3:RETURN
1040 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:X1=X1-7:IF X1<12
 THEN RETURN
1045 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1050 IF Z=OPPONENT THEN NEXT X2
1052 IF X2=8 THEN RETURN
1054 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X-1:GOSUB 1160:NEXT X3:RETURN
1060 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:X1=X1+7:IF X1>66
 THEN RETURN
1065 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1070 IF Z=OPPONENT THEN NEXT X2
1072 IF X2=8 THEN RETURN
1074 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X+1:GOSUB 1160:NEXT X3:RETURN
1080 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:X1=X1-7:Y1=Y1-5:
IF X1<12 OR Y1<0 THEN RETURN
1085 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1090 IF Z=OPPONENT THEN NEXT X2
1092 IF X2=8 THEN RETURN
1094 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X-11:GOSUB 1160:NEXT X3:RETURN
1100 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:X1=X1+7:Y1=Y1-5:
IF X1>66 OR Y1<0 THEN RETURN
1105 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1110 IF Z=OPPONENT THEN NEXT X2
1112 IF X2=8 THEN RETURN
1114 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X-9:GOSUB 1160:NEXT X3:RETURN
1120 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:X1=X1+7:Y1=Y1+5:
IF X1>66 OR Y1>38 THEN RETURN
1125 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1130 IF Z=OPPONENT THEN NEXT X2
1132 IF X2=8 THEN RETURN
1134 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X+11:GOSUB 1160:NEXT X3:RETURN
1140 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:FOR X2=1 TO 7:X1=X1-7:Y1=Y1+5:
IF X1<12 OR Y1>38 THEN RETURN
1145 LOCATE X1,Y1,Z:IF Z=YOUR AND X2=1
 OR Z=1 THEN RETURN
1150 IF Z=OPPONENT THEN NEXT X2
1152 IF X2=8 THEN RETURN
1154 GOSUB 1160:P1=X2:FOR X3=1 TO X2:X
=X+9:GOSUB 1160:NEXT X3:RETURN
```

Draw chip at selected location.

```
1160 X1=7*(X-INT(X/10)*10)+5:Y1=INT(X/
10)*5-5:FOR X1=X1 TO X1+5:PLOT X1,Y1:D
RAWTO X1,Y1+3:NEXT X1
1165 FOR Z=1 TO 6:POKE 53279,0:NEXT Z:
RETURN
```

Step through the best moves for the computer. If none are found then pass or concede game.

```
2000 POKE 77,0:IF P2-SY=0 OR SY=0 THEN
 2010
2001 IF FL=0 THEN IP=0
2002 IF FL=1 THEN YP=0
2003 FOR B=1 TO 60:X=A(B):X1=7*(X-INT(
X/10)*10)+7:Y1=INT(X/10)*5-3:POKE 5327
9,0:POKE 53279,0:LOCATE X1,Y1,Z
2004 IF Z=1 THEN GOSUB 3000
2005 NEXT B:IF P2-SY<>0 AND SY<>0 AND
FL=0 AND YP=0 THEN 2008
2006 IF P2-SY<>0 AND SY<>0 AND FL=1 AN
D IP=0 THEN YP=1:RETURN
2007 GOTO 2010
2008 IP=1:FOR ST=170 TO 220:SOUND 0,ST
,10,10:NEXT ST:SOUND 0,0,0,0:GOTO 630
2010 FOR X=0 TO 200:SOUND 0,X,12,6:NEX
T X:SOUND 0,0,0,0:PRINT "NO POSSIBLE M
OVES LEFT!";:FOR X=1 TO 300:NEXT X
2012 POKE 752,1:IF P2-SY>SY THEN 2030
2014 IF P2-SY=SY THEN 2040
2020 PRINT :PRINT "YOU WON THIS ROUND
BY ";2*SY-P2;" CHIPS.";:GOTO 2050
2030 ? :? "I ONCE AGAIN PROVE MY SUPER
IORITY!";:FOR X=1 TO 300:NEXT X:? :? "
I WON BY ";P2-2*SY;" CHIPS.";
2032 GOTO 2050
2040 ? :? "IT'S A DRAW! I'LL BEAT YOU
NEXT TIME!!";
2050 FOR X=1 TO 20:POKE 53279,0:NEXT X
:FOR X=1 TO 300:NEXT X:PRINT :PRINT "W
ANT TO PLAY AGAIN?";:POKE 764,255
2060 CO=PEEK(764):IF CO=255 THEN 2060
2062 IF CO=43 THEN POKE 764,255:RUN
2064 POKE 764,255:? :? "OK. SEE YOU LA
TER!":PRINT :END
```

The computer has found an empty square; now check to see if it is a legal move.

```
3000 FOR C=1 TO 8:X1=7*(X-INT(X/10)*10
)+6:Y1=INT(X/10)*5-3:POKE 53279,0:POKE
 53279,0:GOSUB 2990+20*C:NEXT C:RETURN
```

Check legality of computer's moves (up-left, up-right, etc.). If it is a legal move then jump out of loop and do capture.

```
3010 FOR D=1 TO 7:Y1=Y1+5:IF Y1>38 THE
N RETURN
3012 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3014 IF Z=YOUR THEN NEXT D
3020 IF D=8 THEN RETURN
3022 POP :POP :POP :POP :POP :GOTO 515
0
3030 FOR D=1 TO 7:Y1=Y1-5:IF Y1<0 THEN
 RETURN
3032 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3034 IF Z=YOUR THEN NEXT D
3040 IF D=8 THEN RETURN
3042 POP :POP :POP :POP :POP :GOTO 515
0
3050 FOR D=1 TO 7:X1=X1-7:IF X1<12 THE
N RETURN
3052 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3054 IF Z=YOUR THEN NEXT D
3060 IF D=8 THEN RETURN
3062 POP :POP :POP :POP :POP :GOTO 515
0
3070 FOR D=1 TO 7:X1=X1+7:IF X1>66 THE
N RETURN
3072 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3074 IF Z=YOUR THEN NEXT D
3080 IF D=8 THEN RETURN
3082 POP :POP :POP :POP :POP :GOTO 515
0
3090 FOR D=1 TO 7:X1=X1+7:Y1=Y1+5:IF X
1>66 OR Y1>38 THEN RETURN
3092 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3094 IF Z=YOUR THEN NEXT D
3100 IF D=8 THEN RETURN
```

continued on next page

continued from previous page

```
3102 POP :POP :POP :POP :POP :GOTO 515
0
3110 FOR D=1 TO 7:X1=X1-7:Y1=Y1+5:IF X
1<12 OR Y1>38 THEN RETURN
3112 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3114 IF Z=YOUR THEN NEXT D
3120 IF D=8 THEN RETURN
3122 POP :POP :POP :POP :POP :GOTO 515
0
3130 FOR D=1 TO 7:X1=X1-7:Y1=Y1-5:IF X
1<12 OR Y1<0 THEN RETURN
3132 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3134 IF Z=YOUR THEN NEXT D
3140 IF D=8 THEN RETURN
3142 POP :POP :POP :POP :POP :GOTO 515
0
3150 FOR D=1 TO 7:X1=X1+7:Y1=Y1-5:IF X
1>66 OR Y1<0 THEN RETURN
3152 LOCATE X1,Y1,Z:IF Z=1 OR Z=OPPONE
NT AND D=1 THEN RETURN
3154 IF Z=YOUR THEN NEXT D
3160 IF D=8 THEN RETURN
3162 POP :POP :POP :POP :POP :GOTO 515
0
```

Set up an old game.

```
4000 TRAP 33333:? CHR$(125);"HOW DO YO
U WANT TO SET IT UP?":? "1 - REGULAR,
2 - MANUAL":POKE 752,1:P2=0:SY=0
4010 GET #1,CO:IF CO<49 OR CO>50 THEN
4010
4020 POKE 764,255:IF CO=49 THEN 4250
4050 PRINT CHR$(125);"PRESS <ESC> WHEN
 FINISHED":PRINT "USE BUTTON TO SELECT
 SQUARE":POKE 752,1
4060 GOSUB 300:A=PDL:COLOR COLR:PLOT X
X,YY:COLOR 0:XX=7*(A-INT(A/8)*8)+15:YY
=INT(A/8)*5+2
4062 LOCATE XX,YY-1,COLR:PLOT XX,YY
4070 IF PEEK(764)=255 THEN 4080
4072 CO=PEEK(764):POKE 764,255:IF CO=2
8 THEN 4160
4080 IF STRIG(0)=1 THEN 4060
4082 PRINT CHR$(125);
4090 PRINT "WHICH COLOR (R/B) ";:GET #
1,ST:IF ST<>82 AND ST<>66 THEN 4082
4092 X=((A-INT(A/8)*8)+1)+(10*(INT(A/8
)+1))
4100 COLOR 3:IF ST=82 THEN COLOR 2
4102 GOSUB 4240:Y1=Y1+1:B=0:IF X1>18 T
HEN LOCATE X1-7,Y1,Z:B=B+(Z<>1)
4110 IF X1<60 THEN LOCATE X1+7,Y1,Z:B=
B+(Z<>1)
4112 IF Y1>4 THEN LOCATE X1,Y1-5,Z:B=B
+(Z<>1)
4114 IF Y1<34 THEN LOCATE X1,Y1+5,Z:B=
B+(Z<>1)
4120 IF X1>18 AND Y1>4 THEN LOCATE X1-
7,Y1-5,Z:B=B+(Z<>1)
4122 IF X1>18 AND Y1<34 THEN LOCATE X1
-7,Y1+5,Z:B=B+(Z<>1)
4130 IF X1<60 AND Y1>4 THEN LOCATE X1+
7,Y1-5,Z:B=B+(Z<>1)
4132 IF X1<60 AND Y1<34 THEN LOCATE X1
+7,Y1+5,Z:B=B+(Z<>1)
4134 IF B=0 THEN 4150
4140 GOSUB 1160:P2=P2+1:LOCATE XX,YY-1
,COLR:GOTO 4050
4150 IF P2=0 THEN 4140
4152 GOSUB 4220:PRINT "YOU'RE MAKING U
P THIS GAME! A PIECE  COULDN'T BE THE
RE!":FOR X=1 TO 300:NEXT X:GOTO 4050
4160 X=44:GOSUB 4240:Y1=Y1+1:LOCATE X1
,Y1,Z:IF Z=1 THEN 4230
4162 X=45:GOSUB 4240:Y1=Y1+1:LOCATE X1
,Y1,Z:IF Z=1 THEN 4230
4170 X=54:GOSUB 4240:Y1=Y1+1:LOCATE X1
,Y1,Z:IF Z=1 THEN 4230
4172 X=55:GOSUB 4240:Y1=Y1+1:LOCATE X1
,Y1,Z:IF Z=1 THEN 4230
4174 P2=0:FOR Y=1 TO 8:FOR A=1 TO 8
4180 X=10*Y+A:GOSUB 4240:Y1=Y1+1:X1=X1
+1:LOCATE X1,Y1,Z:IF Z=1 THEN 4190
4182 P2=P2+1:IF Z=YOUR THEN SY=SY+1
4190 NEXT A:NEXT Y:IF P2-SY=0 OR SY=0
THEN 4200
4192 IF P2=64 THEN 4210
4194 LOCATE XX,YY-1,Z:COLOR Z:GOSUB 60
00:GOTO 610
4200 GOSUB 4220:? CHR$(125);"YOU HAVE
TO GIVE ME A CHANCE!":P2=0:SY=0:FOR X=
1 TO 300:NEXT X:GOTO 4050
4210 GOSUB 4220:? CHR$(125);"YOU DIDN'
T LEAVE ANY ROOM!":FOR X=1 TO 300:NEXT
 X:GOTO 4050
4220 PRINT CHR$(125);:FOR X=1 TO 200:P
OKE 53279,0:NEXT X:RETURN
4230 GOSUB 4220:? CHR$(125);"THE CENTR
AL SQUARES MUST BE FILLED!":FOR X=1 TO
 300:NEXT X:GOTO 4050
4240 X1=7*(X-INT(X/10)*10)+7:Y1=INT(X/
10)*5-3:RETURN
4250 GOSUB 6000:P2=4:SY=2:GOTO 600
```

Print instructions for game.

```
5000 POSITION 2,12:PRINT "NEED INSTRUC
TIONS";:TRAP 5000:INPUT A$:IF A$(1,1)<
>"Y" THEN RETURN
5010 GRAPHICS 0:POKE 752,1:POSITION 14
,2:PRINT " FLIP-IT ":PRINT :PRINT "
 THE OBJECT OF THIS GAME IS TO"
5020 PRINT "COMPLETELY FILL THE BOARD
WITH AS     MANY PIECES OF YOUR COLOR
AS YOU CAN."
5030 PRINT "TO DO THIS YOU MUST OUTFLA
NK YOUR     OPPONENT'S PIECES AND FLIP
 THEM TO"
5040 PRINT "YOUR COLOR. OUTFLANKING CA
N OCCUR     HORIZONTALLY, VERTICALLY,
OR DIAG-"
5050 PRINT "ONOLLY. THE GAME ENDS WHEN
 THE BOARD  IS FULL OR WHEN BOTH PLAYE
RS CAN'T    MOVE. ":PRINT
5060 PRINT "WHOEVER HAS THE MOST PIECE
S WINS.":POSITION 5,23:PRINT "(PRESS A
NY KEY TO CONTINUE)";
5065 IF PEEK(764)=255 THEN 5065
5070 POKE 764,255:GRAPHICS 0:POKE 752,
1:POSITION 2,2:PRINT "YOU MAKE MOVES B
Y MOVING THE CURSOR"
5080 PRINT "WITH THE JOYSTICK TO THE S
QUARE YOU   DESIRE AND PRESS FIRE":PRI
NT
5090 PRINT "DURING THE GAME YOUR CAN A
LSO CHOOSE  ONE OF THE OPTIONS BY PRES
SING THE    KEY INDICATED:":PRINT
5100 PRINT "Q - TO QUIT THE GAME":PRIN
T :PRINT "N - NOT ABLE TO MOVE":PRINT
:PRINT "B - ASK FOR THE BEST MOVE"
5110 POSITION 7,23:PRINT "(PRESS ANY K
EY TO BEGIN)";
5130 IF PEEK(764)=255 THEN 5130
5140 POKE 764,255
5150 X=A(B):RETURN
```

Color setup routine.

```
6000 TRAP 33333:GRAPHICS 5+48:SETCOLOR
 1,5,4:SETCOLOR 2,7,4:SETCOLOR 4,12,4:
SETCOLOR 0,0,0:RETURN
```

Data to set up square-choice priority for computer.

```
6020 A$="1181188831611383168638683363З
6664151148415854858435333464356546564 25
2247425754757326223732676376 72171"
6030 A$(LEN(A$)+1)="128217872878227227
77":RETURN
```

Chip-count update.

```
7000 COLOR 0:PLOT 0,40:DRAWTO 79,40:PL
OT 0,42:DRAWTO 79,42
7010 IF SY>0 THEN COLOR YOUR:PLOT 12,4
0:DRAWTO SY+11,40
7020 IF (P2-SY)>0 THEN COLOR OPPONENT:
PLOT 12,42:DRAWTO (P2-SY)+11,42
7030 RETURN
```

S-80 Version

```
10 REM ↕↕↕↕↕↕↕↕↕↕↕↕↕
20 REM ↕       FLIP-IT       ↕
30 REM ↕ By Michael Prescott ↕
40 REM ↕                     ↕
50 REM ↕    S-80  Version    ↕
60 REM ↕   By Alan J. Zett   ↕
70 REM ↕↕↕↕↕↕↕↕↕↕↕↕↕
80 REM
```

Initialization.

```
90 CLEAR200:DEFINTA-Z:DIMA(60),B(99):A$="11811888316113831686386
8336336664151148415854858435334643565465642522474257547573262237
3267637672171128217872878227227777":Z=1
100 FORX=1TO120STEP2:A(Z)=ASC(MID$(A$,X,1))-48+(ASC(MID$(A$,X+1,
1))-48)*10:Z=Z+1:NEXT:FORX=0TO99:B(X)=0:NEXT:B(45)=-1:B(54)=-1:B
(44)=1:B(55)=1:GOSUB620
```

Initialize game board and graphic strings.

```
110 PDL=0:B=0:FL=0:P2=0:RF=0:BL=-1:WH=1:OP=0:YO=0:FORX=0TO9:B(X)
=9:B(X+90)=9:NEXT:FORX=9TO89STEP10:B(X)=9:B(X+1)=9:NEXT
120 B$=CHR$(191)+STRING$(3,131)+CHR$(191)+STRING$(5,24)+CHR$(26)
+STRING$(5,131):W$=STRING$(5,191)+STRING$(5,24)+CHR$(26)+STRING$
(5,131)
```

Get choice of color, who goes first, and whether or not a normal board is to be used. Also set up players' pieces.

```
130 CLS:F=0:A$="":PRINT@24,"F L I P - I T":PRINT:INPUT"Which col
or do you want (W or B) ";A$:IFLEFT$(A$,1)<>"W"THEN150
140 YO$=W$:OP$=B$:YO=WH:OP=BL:GOTO160
150 YO$=B$:OP$=W$:YO=BL:OP=WH
160 A$="":INPUT"Do you want to go first ";A$:IFLEFT$(A$,1)="Y"TH
EN F=1
170 A$="":INPUT"Want to set up your own game ";A$:CLS:GOSUB180:I
FLEFT$(A$,1)="Y"THENB(44)=0:B(45)=0:B(54)=0:B(55)=0:GOTO280ELSEI
FF=1THEN340ELSEGOSUB190:GOTO410
```

Draw game board.

```
180 FORX=0TO7:FORY=0TO7:PRINT@X*128+Y*7,STR$(X+1);STR$(Y+1);:NEX
TY:NEXTX:RETURN
```

Plot pieces and update game display data.

```
190 P2=0:WP=0:BP=0:FORX=0TO99:IFB(X)<>0ANDB(X)<>99THENY=INT((X-1
1)/10)*128+((X-11)-INT((X-11)/10)*10)*7
200 IFB(X)=1THENPRINT@Y,W$;:WP=WP+1:P2=P2+1
210 IFB(X)=-1THENPRINT@Y,B$;:BP=BP+1:P2=P2+1
220 NEXT:FORY=0TO47:SET(110,Y):NEXT:PRINT@56,"FLIP-IT";:PRINT@18
5,"W =";WP;:PRINT@249,"B =";BP;:RETURN
```

No-possible-move command.

```
230 PRINT@376,"CHECKING";:FORZ=1TO60:PDL=A(Z):P=PDL:C=YO:SC=OP:F
L=0:GOSUB500:IFLG=1THEN240ELSENEXT:IFIP=1THEN430ELSEYP=1:GOTO410
240 PRINT@376,"?WHAT   ";:PRINT@440,"ABOUT  ";:A$=LEFT$(STR$(PDL
),2)+" "+RIGHT$(STR$(PDL),1):FORX=1TO15:PRINT@504,A$;:FORY=0TO99
:NEXT:PRINT@504,"       ";:FORY=0TO99:NEXT:NEXT:GOTO350
```

Find player's best move.

```
250 PRINT@376,"LOOKING ";:FORZ=1TO60:PDL=A(Z):P=PDL:C=YO:SC=OP:F
L=0:GOSUB500:IFLG=1THEN260ELSENEXT:IFIP=1THEN430ELSEYP=1:GOTO410
260 PRINT@440,"LOOKS   ";:PRINT@504,"GOOD!";:A$=LEFT$(STR$(PDL),
2)+" "+RIGHT$(STR$(PDL),1):FORX=1TO15:PRINT@376,A$;:FORY=0TO99:N
EXT:PRINT@376,"       ";:FORY=0TO99:NEXT:NEXT:PRINT@504,"     ";
:GOTO350
```

Quit-game routine.

```
270 A$="":CLS:PRINT@24,"F L I P - I T":PRINT:INPUT"Do you really
 want to quit ";A$:IFLEFT$(A$,1)<>"Y"THENCLS:GOSUB180:GOTO340ELS
E440
```

Set up your own board.

```
280 RF=1:PRINT@504,"(D=DONE)";:GOSUB350:PRINT@376,"COLOR?  ";
290 A$=INKEY$:IFA$=""THEN290ELSEIFA$="D"THENPRINT@504,"        "
;:GOSUB180:IFF=1THEN340ELSEGOSUB190:GOTO410
300 PRINT@INT((PDL-11)/10)*128+((PDL-11)-INT((PDL-11)/10)*10)*7,
;
310 IFA$="W"THENPRINTW$;:B(PDL)=WH
320 IFA$="B"THENPRINTB$;:B(PDL)=BL
330 GOTO280
```

Get player's move.

```
340 IFP2=64THEN430ELSEGOSUB190:IFP2=64THEN430
350 PDL=0:YP=0:PRINT@440,"       ";:PRINT@376,;:IFRF=1THENPRINT"
SQUARE? ";ELSEPRINT"MOVE?   ";
360 FORX=0TO1
370 A$=INKEY$:IFA$=""THEN370ELSEIFA$="R"THENCLS:GOSUB180:GOSUB19
0:GOTO350ELSEIFA$="Q"THEN270ELSEIFA$="N"THEN230ELSEIFA$="B"THEN2
50ELSEIFA$="D"ANDRF=1THENRF=0:RETURN
380 IFA$<"1"ORA$>"8"THENPRINT@440,"INVALID";:FORY=1TO555:NEXT:GO
TO350
390 PRINT@441+X*2,A$;:PDL=PDL+VAL(A$)*10[(1-X):NEXT:IFRF=1THENRF
=0:RETURN
400 C=YO:SC=OP:FL=1:GOSUB500:IFLG=0THENA$="0":GOTO380ELSEGOSUB19
0
```

Choose computer's move.

```
410 IP=0:PRINT@376,"MAYBE   ";:FORZ=1TO60:PDL=A(Z):PRINT@440,LEF
T$(STR$(PDL),2)" "RIGHT$(STR$(PDL),1);:P=PDL:IF B(P)<>0THENNEXTZ
ELSEC=OP:SC=YO:FL=1:GOSUB500:IFLG=0THENNEXTZELSE340
```

Computer must pass.

```
420 IF P2<>64 AND YP=0 THEN PRINT@569,"PASS!";:IP=1:FORX=0TO555:
NEXTX:PRINT@569,"     ";:GOTO340
```

End-game routine.

```
430 PRINT@568,"NO MOVES";:PRINT@634,"LEFT!";:PRINT@760,"* GAME *
";:PRINT@824,"* OVER *";:FORX=0TO5000:NEXTX
440 CLS:PRINT@24,"F L I P - I T":PRINT:PRINT"FINAL SCORE:":PRINT
:PRINT"    WHITE ="WP:PRINT"    BLACK ="BP:PRINT
450 IFWP=BPTHENPRINT"It's a DRAW! I'll beat you next time!!":PRI
NT:GOTO480ELSEIF(WP>BPANDOP=WH)OR(BP>WPANDOP=BL)THENPRINT"I once
 again prove my superiority! I won by";:GOTO470
460 PRINT"You won this round by";
470 PRINTABS(WP-BP)"chips!":PRINT:PRINT
```

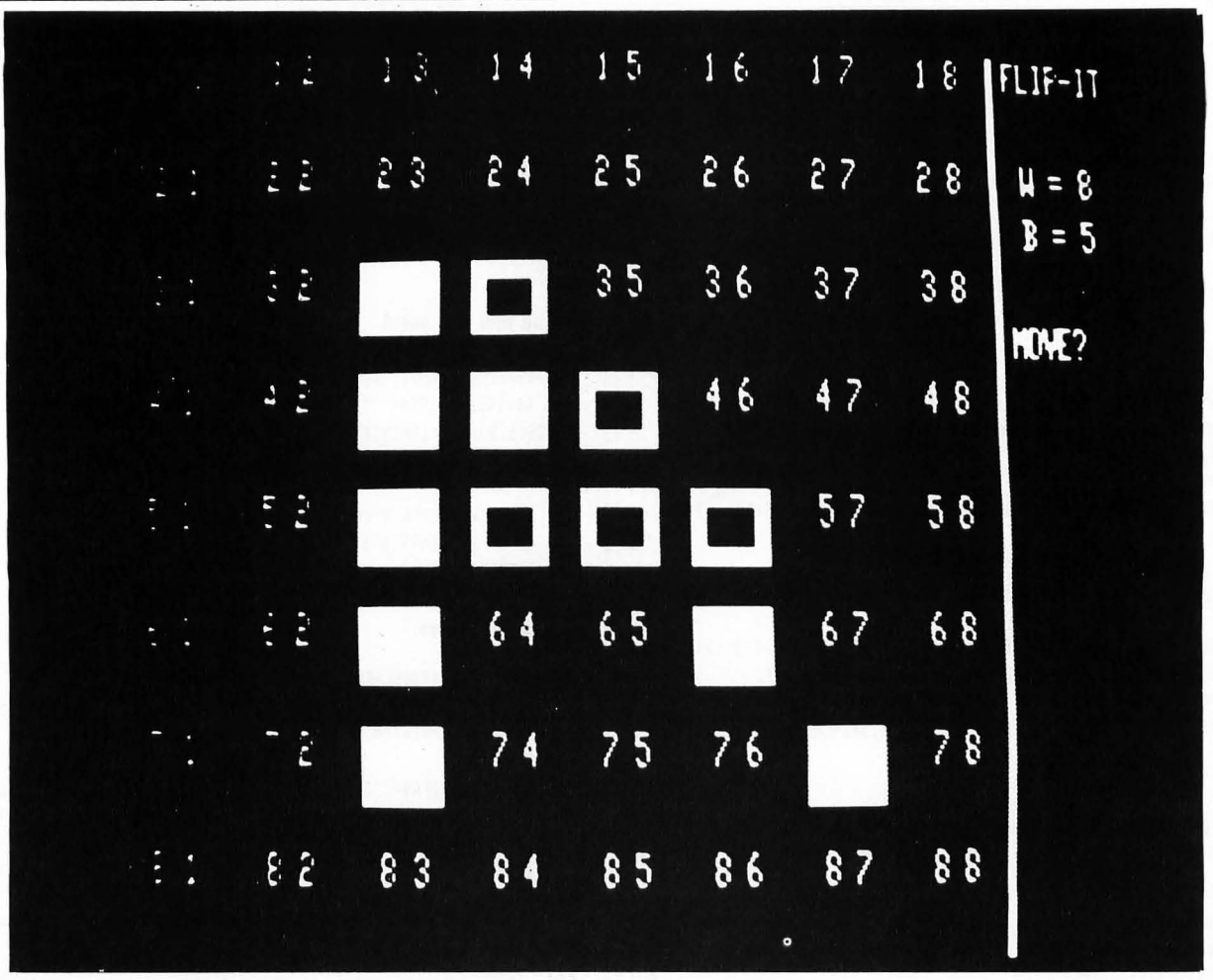
Check for new game.

```
480 A$="":INPUT"Want to play again ";A$:IFLEFT$(A$,1)="Y"RUN
490 CLEAR50:PRINT:PRINT"OK. See you later!":END
```

Legal check: Combines a search for legal moves with a routine to flip pieces.

continued on next page

continued from previous page




```
500 LG=0::IFB(PDL)<>0THEN610
510 FORY=1TO8:F=0:ONYGOTO520,530,540,550,560,570,580,590
520 P=PDL:FORX=1TO8:P=P+10:IFP>99THEN600ELSEIFB(P)=0OR(B(P)=CAND
X=1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO60
0ELSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOT
0520
530 P=PDL:FORX=1TO8:P=P-10:IFP<0THEN600ELSEIFB(P)=0OR(B(P)=CANDX
=1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO600
ELSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOTO
530
540 P=PDL:FORX=1TO8:P=P+1:IFP>99THEN600ELSEIFB(P)=0OR(B(P)=CANDX
=1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO600
ELSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOTO
540
550 P=PDL:FORX=1TO8:P=P-1:IFP<0THEN600ELSEIFB(P)=0OR(B(P)=CANDX=
1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO600E
LSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOTO5
50
560 P=PDL::FORX=1TO8:P=P+9:IFP>99THEN600ELSEIFB(P)=0OR(B(P)=CAND
X=1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO60
0ELSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOT
0560
570 P=PDL:FORX=1TO8:P=P-9:IFP<0THEN600ELSEIFB(P)=0OR(B(P)=CANDX=
1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO600E
LSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOTO5
70
580 P=PDL:FORX=1TO8:P=P+11:IFP>99THEN600ELSEIFB(P)=0OR(B(P)=CAND
X=1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO60
0ELSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOT
0580
590 P=PDL:FORX=1TO8:P=P-11:IFP<0THEN600ELSEIFB(P)=0OR(B(P)=CANDX
=1)ORB(P)=9THEN600ELSEIFB(P)=SCTHENIFF=1THENB(P)=C:NEXTX:GOTO600
ELSENEXTX:GOTO600ELSEF=1:LG=1:P=PDL:IFFL=0THEN610ELSEB(P)=C:GOTO
590
600 FORX=0TO1:NEXTX:NEXTY
610 FORX=0TO1:NEXTX:FORY=0TO1:NEXTY:RETURN
```

Instructions.

```
620 CLS:A$="":PRINT@512,;:INPUT"Need instructions ";A$:IFLEFT$(A
$,1)<>"Y"THEN110
630 CLS:PRINT@24,"F L I P - I T":PRINT:PRINT"The object of this
game is to completely fill the board with as many pieces of your
 color as you can. To do this you must out- flank your opponent
's pieces and flip them to your color."
640 PRINT:PRINT"Outflanking can occur horizontally, vertically,
or diagonally.  The game ends when the board is full, or when bo
th players can'tmove. At that time, whoever has the most pieces
wins."
650 PRINT@977,"(Press <ENTER> to continue)";
660 IFINKEY$<>CHR$(13)THEN660
670 CLS:PRINT@24,"F L I P - I T":PRINT:PRINT"You make moves by e
ntering the row number combined with the     column number. For
example: The upper right-hand corner is the  position 18 - that
is 1 for the first row and 8 for the eighth"
680 PRINT"column. If you make a mistake entering a number, type
a nine forany of the numbers and it will mark the entry as inval
id.":PRINT"During the game you also have the following options:"
:PRINT
690 PRINT"Q - to Quit the game":PRINT:PRINT"N - Not able to move
":PRINT:PRINT"B - ask for the Best move":PRINT@977,"(Press <ENTE
R> to start game)";
700 IFINKEY$<>CHR$(13)THEN700ELSE110
```

Translation Appeal

Reward! Reward! Reward!

We will give away a $100 software certificate each month for the best translation of a feature program in **SoftSide** magazine. Furthermore, we will publish the translation in the magazine. Your portfolio will be enhanced and you will garner fame and fortune for your efforts!

We will allow three months after initial publication of a program for the translation to be sent. After that time we will not accept entries. The quality of the translation will be judged by the **SoftSide** editorial staff and the winning entry will be published the following month, i.e., four months after publication of the original program.

Entries must be submitted on cassette or disk, accompanied by documentation. Please enclose a self-addressed stamped envelope if you would like your entry returned to you.

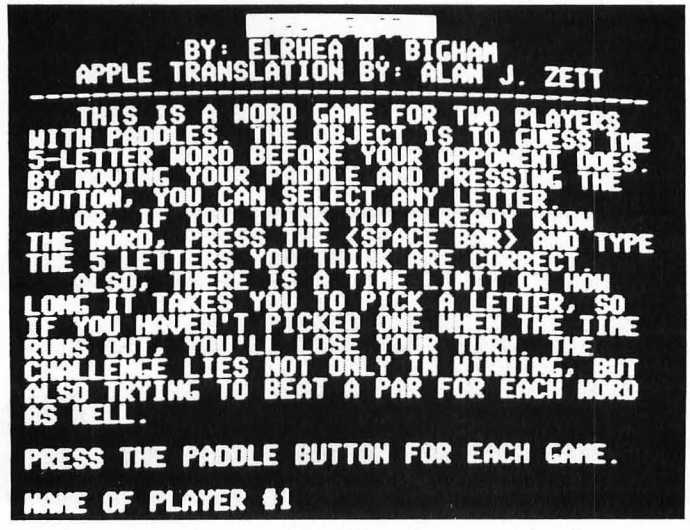
Word Challenge

by Elrhea Bigham

Apple and S-80 translations by Alan J. Zett

"Word Challenge" is a word game for a 32K Atari (modifiable for smaller systems) or a 16K Apple or S-80.

Hang onto your hats, word buffs. Here is a program that will challenge even the best of you. This two-player game not only pits one player against another in competition for the best score, but adds the additional challenge of beating the "par" for each word to be guessed.



For each game the computer selects a five-letter word at random from its programmed list. The object is to guess the word before your opponent does, by choosing letters that you think might be in the word. The computer keeps track of how many games each player has won, so there won't be any disputes about their respective linguistic abilities.

On the screen, you will see a display of all 26 letters of the alphabet, with a "window" under each letter. Also displayed is a clue or definition for the word chosen by the computer, and the par for that word. The initials of the two players are shown, and above them the total number of guesses and the number of correct guesses each has made. The numerals 1-5 are also displayed, marking the locations of the five letters of the word to be guessed.

To guess if a letter is contained in the 5-letter word, a player positions the "cursor" over the chosen letter. This is done with the joystick on the Atari, with the paddle on the Apple, and with the arrow keys on the S-80. Pressing the fire button, paddle button, or "ENTER" key, respectively, tells the computer that you have made your selection. If that letter is contained in the word, the player gets another guess; if not, he loses his turn and the other player selects a letter.

At any time during a player's turn, he may type in the whole word if he thinks he knows what it is. Of course, he may also simply keep selecting letters with his joystick, paddle, or arrow keys until the word is complete. If he does guess and the guess is wrong, he continues with his turn without penalty.

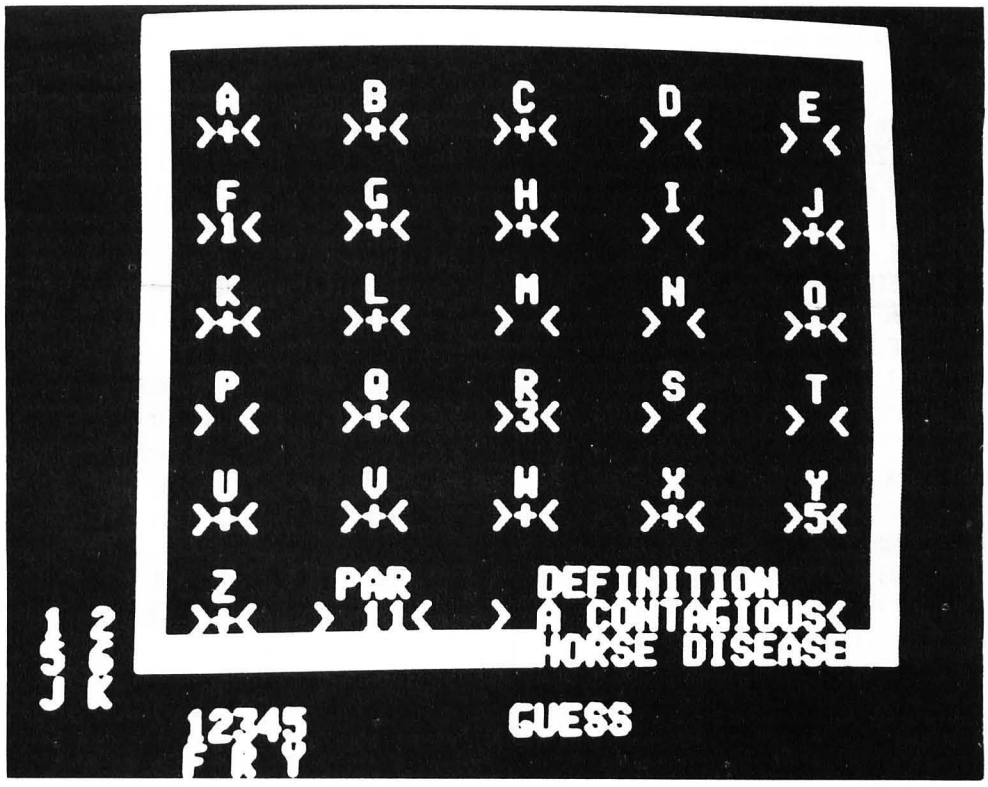
To keep the game going, there is a time limit for a player to complete his turn. After about one minute, if the current player has not made a guess he will lose his turn.

All the words and definitions come from a standard dictionary. (Editor's note: I ain't never heard of NONE of them thar words, so I ain't takin' no responserbility for 'em.) None of the words contain double letters. Some are common words but with an uncommon definition, while others are rarely used. There is no penalty for exceeding the par for a word.

There are 130 different words included in the program. With so many words, even when you type them all in, you'll be surprised at how much of a challenge it will be to guess them during play after a lapse of a few hours or days. If the memory size of your computer is limited, by the way, you can easily shorten the program by omitting some of the word definition lines and changing the random selection routine accordingly. Likewise, you can easily add your own to further enhance the challenge of the game.

VARIABLES

A$: Cursor character.
D: The number of the letter where the cursor is located.
D$: Definition of the word.
GS: Flag for a typed-in guess.
GS$: Input for a typed-in guess.
GUS(*): Total number of guesses for each player.
I,J,K: Loop variables.
N$: The word to be guessed.
P$,P1$: Input for the names of the players.
PL: Player number (0 or 1).
PUZ: Puzzle number (0 to 129).
PZ(*): Puzzle letter number (0 to 5).
SC(*): Number of correctly-guessed letters for each player.
ST: Joystick/paddle value.
T: Time-limit counter.
W(*): Total wins for each player.
X,Y: Horizontal and vertical position of the cursor.
ZP: Number of letters correctly guessed.



Apple Version

Setup for start of game.

```
1  TEXT : HOME : HTAB 15: INVERSE
     : PRINT " CHALLENGE! ": NORMAL
     : HTAB 11: PRINT "BY: ELRHEA
     M. BIGHAM": PRINT "   APPLE
     TRANSLATION BY: ALAN J. ZET
     T"
2  PRINT "----------------------
     ----------------";
3  GOSUB 500
5  PRINT : INPUT "NAME OF PLAYER
     #1 ";P$: INPUT "NAME OF PLAY
     ER #2 ";P1$
6 PUZ =  INT ( RND (1) * 130) + 1

7  DIM PZ(30),W(2),SC(2),GUS(2):A
     $ = "+":X = 8:Y = 2:D = 1:PL
      = 0:ZP = 0:W(0) = 0:W(1) =
     0
8 SC(0) = 0:SC(1) = 0:GUS(0) = 0:
     GUS(1) = 0
9  FOR I = 1 TO 30:PZ(I) = 0: NEXT
     I: GOSUB 2000: FOR I = 1 TO
     5:PZ( ASC ( MID$ (N$,I,1)) -
     64) = I: NEXT I
```

Print Screen display.

```
10  HOME : HTAB 8: VTAB 3: PRINT
     "A     B     C     D     E":
      HTAB 7: PRINT ")+(    )+(
     )+(    )+(    )+("
20  HTAB 8: VTAB 6: PRINT "F
     G     H     I     J": HTAB 7
     : PRINT ")+(    )+(    )+(    )
     +(    )+("
30  HTAB 8: VTAB 9: PRINT "K
     L     M     N     O": HTAB 7
     : PRINT ")+(    )+(    )+(    )
     +(    )+("
40  HTAB 8: VTAB 12: PRINT "P
      Q     R     S     T": HTAB
     7: PRINT ")+(    )+(    )+(
     )+(    )+("
50  HTAB 8: VTAB 15: PRINT "U
      V     W     X     Y": HTAB
     7: PRINT ")+(    )+(    )+(
     )+(    )+("
60  HTAB 8: VTAB 18: PRINT "Z
     PAR     DEFINITION": HTAB 7:
      PRINT ")+(  )+++(  )+++++++
     ++++++("
64  INVERSE : VTAB 1: FOR I = 6 TO
     34: HTAB I: PRINT " ";: NEXT
     I
65  FOR I = 1 TO 20: HTAB 35: VTAB
     I: PRINT " ";: NEXT I
66  VTAB 20: FOR I = 34 TO 6 STEP
      - 1: HTAB I: PRINT " ";: NEXT
     I: FOR J = 20 TO 1 STEP  - 1
     : HTAB 5: VTAB J: PRINT " ";
     : NEXT J: NORMAL
67  IF GS = 1 THEN  RETURN
69  HTAB 7: VTAB 22: PRINT "12345
     ";: HTAB 1: VTAB 20: PRINT "
     0 0";: HTAB 1: VTAB 21: PRINT
      LEFT$ (P$,1);" "; LEFT$ (P1
     $,1);: GOSUB 125
70  HTAB X: VTAB Y: PRINT A$;
```

Gets player's move or guess.

```
78  IF T >  = 215 THEN  FOR J = 1
      TO 19: PRINT  CHR$ (7);: NEXT
     J:T = 0: IF PL > 1 THEN PL =
     0
79 T = T + 1: IF T >  = 215 THEN
     PL = PL + 1: HTAB X: VTAB Y:
      PRINT " ";:X = 8:Y = 2:D =
     1: GOTO 78
80 ST =  PDL (PL): IF ST > 235 THEN
     ST = 235
81 ST =  INT (ST / 9.4): HTAB X: VTAB
     Y: PRINT " ";:Y =  INT (ST /
     5) * 3 + 2:X = (ST -  INT (S
     T / 5) * 5) * 6 + 8:D = ST +
     1
85  IF  PEEK ( - 16384) > 127 AND
     ST1 <  > 1 THEN  HTAB 20: VTAB
     22: INPUT "GUESS ";GS$:GS =
     1:ST1 = 1
```

continued on next page

continued from previous page

```
86  IF GS = 1 AND GS$ <  > N$ THEN
     HTAB 20: VTAB 22: PRINT  SPC(
    12);:GS = 0: GOTO 80
87  IF GS = 1 AND GS$ = N$ THEN S
    T1 = 0: GOTO 210
88  IF  PEEK ( - 16384) > 127 AND
    ST1 = 1 THEN  POKE  - 16368,
    0
99  IF  PEEK ( - 16287 + PL) > 12
    7 THEN 120
```

Check for a legal move.

```
100  POKE  - 16336, PEEK ( - 1633
     6): IF D < 27 AND D > 0 AND
     X < 33 AND X > 7 THEN  HTAB
     X: VTAB Y: PRINT A$;: GOTO 7
     9
105  IF D < 1 OR D > 26 THEN X =
     8:Y = 2:D = 1: GOTO 79
110  HTAB X: VTAB Y: PRINT A$;: GOTO
     79
120  IF D <  > 28 THEN 141
```

Print definition and par for word.

```
125 LN =  LEN (D$): HTAB 20: VTAB
    19: FOR I = 1 TO LN: IF I <
     = 13 THEN  PRINT  MID$ (D$,
    I,1);: NEXT I
130  IF I = 14 THEN  HTAB 21: VTAB
    20
135  IF I >  = 14 AND LN > 13 THEN
     PRINT  MID$ (D$,I,1);: NEXT
    I
140  FOR I = 1 TO 3: PRINT  CHR$
    (7);: NEXT I: HTAB 13: VTAB
    19: PRINT  LEFT$ (" " +  STR$
    (PZ(27)) + " ",3);: RETURN
```

Check if a letter was already used.

```
141  IF PZ(D) < 0 THEN  FOR I = 1
     TO 3: FOR J = 1 TO 20: POKE
     - 16336, PEEK ( - 16336): NEXT
    J: FOR J = 1 TO 15: NEXT J: NEXT
    I: GOTO 79
```

Erase letter from board and display its position in the word.

```
149  IF PZ(D) <  > 0 THEN  HTAB X
    : VTAB Y + 2: PRINT  STR$ (P
    Z(D));: IF D <  > 27 THEN  HTAB
    PZ(D) + 6: VTAB 23:GUS(PL) =
    GUS(PL) + 1
150  IF PZ(D) = 0 THEN  HTAB X: VTAB
    Y + 2: PRINT " ";: IF D <  >
    27 THEN  HTAB PZ(D) + 6: VTAB
    23:GUS(PL) = GUS(PL) + 1
151  IF PZ(D) = 0 THEN  FOR I = 1
     TO 60: POKE  - 16336, PEEK
    ( - 16336): NEXT I
152  IF D = 27 THEN 150
153  IF PZ(D) <  > 0 THEN  FOR J =
    1 TO PZ(D): PRINT  CHR$ (7);
    : FOR I = 1 TO 18: POKE  - 1
    6336, PEEK ( - 16336): NEXT
    I: NEXT J: PRINT  MID$ (N$,P
    Z(D),1);:ZP = ZP + 1:SC(PL) =
    SC(PL) + 1
154  HTAB PL * 2 + 1: VTAB 19: PRINT
    SC(PL);
155  IF PZ(D) = 0 THEN PL = PL +
    1:ST1 = 0: IF PL > 1 THEN PL
     = 0
160 PZ(D) =  - 1: HTAB 1: VTAB 20
    : PRINT GUS(0);" ";GUS(1);
200  IF ZP < 5 AND GS = 0 THEN T =
    0: GOTO 79
```

Routine for a word guess correctly.

```
210  HTAB 1: VTAB 22: PRINT  SPC(
    111);:W(PL) = W(PL) + 1: HTAB
    11: VTAB 22: PRINT P$;"=";W(
    0); SPC( 6);P1$;"=";W(1);
211  FOR K = 0 TO 10 STEP 3: HTAB
    20: VTAB 19: PRINT  SPC( 13)
    ;: HTAB 20: VTAB 20: PRINT  SPC(
    14);
212  VTAB 1: FOR I = 6 TO 34: HTAB
    I: PRINT " ";: NEXT I
213  FOR I = 1 TO 20: HTAB 35: VTAB
    I: PRINT " ";: NEXT I
214  VTAB 20: FOR I = 34 TO 6 STEP
     - 1: HTAB I: PRINT " ";: NEXT
    I: FOR J = 20 TO 1 STEP  - 1
    : HTAB 5: VTAB J: PRINT " ";
    : NEXT J
215 GS = 1: GOSUB 64: GOSUB 125: NEXT
    K
```

Setup for next game.

```
220 PUZ = PUZ + 1:X = 8:Y = 2:D =
    1:PL = 0:ZP = 0:GS = 0:T = 0
    :ST1 = 0
230  IF  PEEK ( - 16286) > 127 OR
     PEEK ( - 16287) > 127 THEN
    8
232  IF PUZ > 130 THEN PUZ = 1
235  GOTO 230
```

Print instructions.

```
500  PRINT "  THIS IS A WORD GAM
    E FOR TWO PLAYERS  WITH PADD
    LES. THE OBJECT IS TO GUESS
    THE5-LETTER WORD BEFORE YOUR
     OPPONENT DOES.BY MOVING YOU
    R PADDLE AND PRESSING THE  B
    UTTON, YOU CAN SELECT ANY LE
    TTER."
510  PRINT "  OR, IF YOU THINK Y
    OU ALREADY KNOW  THE WORD,
     PRESS THE <SPACE BAR> AND T
    YPETHE 5 LETTERS YOU THINK A
    RE CORRECT."
520  PRINT "  ALSO, THERE IS A T
    IME LIMIT ON HOW  LONG IT T
    AKES YOU TO PICK A LETTER, S
    O  IF YOU HAVEN'T PICKED ONE
     WHEN THE TIME RUNS OUT, YOU
    'LL LOSE YOUR TURN. THE  C
    HALLENGE LIES NOT ONLY IN WI
    NNING, BUT"
530  PRINT "ALSO TRYING TO BEAT A
     PAR FOR EACH WORD AS WELL."
    : PRINT : PRINT "PRESS THE P
    ADDLE BUTTON FOR EACH GAME."

540  RETURN
999  STOP
```

Definitions and word data.

```
1000 D$ = " HERB-CARROT FAMILY":P
    Z(27) = 9:N$ = "ANISE": RETURN

1001 D$ = " VIGOR,PLUMP CHEERFUL"
    :PZ(27) = 11:N$ = "BUXOM": RETURN

1002 D$ = " A SUPPORTING TIMBER":
    PZ(27) = 11:N$ = "JOIST": RETURN

1003 D$ = " GATHER FIELD LEAVINGS
    ":PZ(27) = 9:N$ = "GLEAN": RETURN

1004 D$ = " FASHION, IN  STYLE":P
    Z(27) = 10:N$ = "VOGUE": RETURN

1005 D$ = " TO IMPLY OR  HINT":PZ
    (27) = 10:N$ = "INFER": RETURN

1006 D$ = " A TREE":PZ(27) = 10:N
    $ = "LARCH": RETURN
1007 D$ = "ECCLESIASTI- CAL MEETI
    NG":PZ(27) = 11:N$ = "SYNOD"
    : RETURN
1008 D$ = "OPENINGS INTOPASSAGES"
    :PZ(27) = 11:N$ = "NARES": RETURN

1009 D$ = " WORN, TIRED, VEXED":P
    Z(27) = 9:N$ = "WEARY": RETURN
```

```
1010 D$ = "DETEST,LOATHE":PZ(27) =
     12:N$ = "ABHOR": RETURN
1011 D$ = " LIST, POINT,HAND OF C
     LOCK":PZ(27) = 12:N$ = "INDE
     X": RETURN
1012 D$ = " A MOUNTAIN  RANGE":PZ
     (27) = 9:N$ = "OZARK": RETURN

1013 D$ = " A CONTAGIOUSHORSE DIS
     EASE":PZ(27) = 11:N$ = "FARC
     Y": RETURN
1014 D$ = " A MASS OR A GRINDING
     TOOTH":PZ(27) = 9:N$ = "MOLA
     R": RETURN
1015 D$ = " PEACEFUL RE-LATIONS":
     PZ(27) = 9:N$ = "AMITY": RETURN

1016 D$ = " PIN FITTING A HOLE":P
     Z(27) = 10:N$ = "DOWEL": RETURN

1017 D$ = " ACCENT ON A WORD":PZ(
     27) = 14:N$ = "ICTUS": RETURN

1018 D$ = " PLATE ARMOR FOR THE T
     HIGH":PZ(27) = 15:N$ = "QUIS
     H": RETURN
1019 D$ = " CEMETARY,SE-CRETING C
     AVITY":PZ(27) = 10:N$ = "CRY
     PT": RETURN
1020 D$ = " SHADOWY,HAZY":PZ(27) =
     9:N$ = "VAGUE": RETURN
1021 D$ = " CONICAL HEADDRESS,CAP
     ":PZ(27) = 11:N$ = "TOQUE": RETURN
1022 D$ = " CORRESPOND  IN SOUND"
     :PZ(27) = 9:N$ = "RHYME": RETURN

1023 D$ = " TO APPLY    COSMETICS
     ":PZ(27) = 10:N$ = "PAINT": RETURN
1024 D$ = " A NATURAL   RUBBER":P
     Z(27) = 14:N$ = "PULAY": RETURN

1025 D$ = " DARKENED,   GLOOMY":P
     Z(27) = 10:N$ = "MURKY": RETURN

1026 D$ = " BROOM,LITTLEBUNCH":PZ
     (27) = 11:N$ = "WHISK": RETURN

1027 D$ = " NOTHING":PZ(27) = 10:
     N$ = "ZILCH": RETURN
1028 D$ = " SIMPLETONS":PZ(27) =
     16:N$ = "LAMBS": RETURN
1029 D$ = " INDIAN HEMP NARCOTIC"
     :PZ(27) = 14:N$ = "BHANG": RETURN

1030 D$ = " BEING SUR-  ROUNDED":
     PZ(27) = 12:N$ = "MIDST": RETURN

1031 D$ = " TO BE,OCCUR":PZ(27) =
     9:N$ = "EXIST": RETURN

1032 D$ = " ROUNDED     BUNCHES":
     PZ(27) = 11:N$ = "KNOBS": RETURN

1033 D$ = " ANY PLANT   STEM":PZ(
     27) = 12:N$ = "HAULM": RETURN

1034 D$ = " LOOSELY,A   GHOST":PZ
     (27) = 10:N$ = "ZOMBI": RETURN

1035 D$ = " MALE,FEMALE FASTENER"
     :PZ(27) = 9:N$ = "SCREW": RETURN

1036 D$ = " SHAPES HOT  GLASS":PZ
     (27) = 13:N$ = "PUNTY": RETURN

1037 D$ = " DUSKY HUE":PZ(27) = 9
     :N$ = "UMBER": RETURN
1038 D$ = " A WINDLASS, A CRANK":
     PZ(27) = 9:N$ = "WINCH": RETURN

1039 D$ = " A VARICOSE  VEIN":PZ(
     27) = 11:N$ = "VARIX": RETURN

1040 D$ = " OF EXTREME  DRYNESS,A
     RID":PZ(27) = 18:N$ = "XERIC
     ": RETURN
1041 D$ = " ANY APE OR  MONKEY":P
     Z(27) = 13:N$ = "LACKO": RETURN

1042 D$ = " JERSEY WORN BY SAILOR
     S":PZ(27) = 9:N$ = "FROCK": RETURN
1043 D$ = " THE LARGEST FLOWER":P
     Z(27) = 14:N$ = "KRUBI": RETURN

1044 D$ = " SHELTERS":PZ(27) = 11
     :N$ = "ABRIS": RETURN
1045 D$ = " A BOVINE    RUMINANT"
     :PZ(27) = 14:N$ = "BISON": RETURN

1046 D$ = " GOLF TERM":PZ(27) = 1
     0:N$ = "DORMY": RETURN
1047 D$ = "  EMANCIATED MEAGER,TH
     IN":PZ(27) = 9:N$ = "GAUNT":
      RETURN
1048 D$ = " AN OPEN SHEDOR HUT":P
     Z(27) = 9:N$ = "HOVEL": RETURN

1049 D$ = " LAYMAN":PZ(27) = 9:N$
      = "LAITY": RETURN
1050 D$ = "SEMI-PRECIOUSYELLOW ST
     ONE":PZ(27) = 9:N$ = "TOPAZ"
     : RETURN
1051 D$ = " GRIND,STRIKETOGETHER"
     :PZ(27) = 13:N$ = "GNASH": RETURN

1052 D$ = " A RAVINE":PZ(27) = 12
     :N$ = "GULCH": RETURN
1053 D$ = " A RECESS IN A WALL":P
     Z(27) = 11:N$ = "NICHE": RETURN

1054 D$ = "ONE WHO JOKES":PZ(27) =
     16:N$ = "JOKER": RETURN
1055 D$ = " POWER":PZ(27) = 10:N$
     = "MIGHT": RETURN
1056 D$ = " TO BURN OR  SCORCH":P
     Z(27) = 9:N$ = "PARCH": RETURN

1057 D$ = " TO ENDURE ORTO WAIT F
     OR":PZ(27) = 10:N$ = "ABIDE"
     : RETURN
1058 D$ = " MIMICRY":PZ(27) = 17:
     N$ = "APERY": RETURN
1059 D$ = "A MINUTE ORI-FICE":PZ(
     27) = 15:N$ = "STOMA": RETURN

1060 D$ = " A RADICAL":PZ(27) = 1
     7:N$ = "ULTRA": RETURN
1061 D$ = " FOLD OF SKINBETWEEN F
     INGERS":PZ(27) = 18:N$ = "PL
     ICA": RETURN
1062 D$ = " EASTER":PZ(27) = 14:N
     $ = "PASCH": RETURN
1063 D$ = " CRUDE CREAM OF TARTAR
     ":PZ(27) = 12:N$ = "ARGOL": RETURN
1064 D$ = " A GASEOUS   ELEMENT":
     PZ(27) = 11:N$ = "ARGON": RETURN

1065 D$ = " STORED FOR  FUTURE US
     E":PZ(27) = 12:N$ = "HIVED":
      RETURN
1066 D$ = "POTTERS' CLAY":PZ(27) =
     10:N$ = "ARGIL": RETURN
1067 D$ = " YELLOW COM- PACT LIME
     STONE":PZ(27) = 12:N$ = "CHA
     LK": RETURN
1068 D$ = " TO EXPRESS  ANGER":PZ
     (27) = 9:N$ = "FUMED": RETURN
1069 D$ = " ANY EDIBLE  PURPLE SE
     AWEED":PZ(27) = 13:N$ = "LAV
     ER": RETURN
1070 D$ = " DESERVE":PZ(27) = 10:
     N$ = "MERIT": RETURN
1071 D$ = " A TRICKSTER;RASCAL":P
     Z(27) = 11:N$ = "ROGUE": RETURN

1072 D$ = " TEMPESTUOUS;STORMY":P
     Z(27) = 12:N$ = "WINDY": RETURN

1073 D$ = "ASIAN FOXLIKECARNIVORE
     ":PZ(27) = 18:N$ = "ZIBET": RETURN
1074 D$ = "SECRETED AND USED AS P
     ERFUME":PZ(27) = 18:N$ = "CI
     VET": RETURN
1075 D$ = " CIRCULAR    MOTION":P
     Z(27) = 11:N$ = "GYRAL": RETURN

1076 D$ = " BOUND WITH  FETTERS":
     PZ(27) = 16:N$ = "GYVED": RETURN
```

continued on next page

continued from previous page

```
1077 D$ = " AN ORIENTAL OUTSIDE G
     ARMENT":PZ(27) = 14:N$ = "HA
     ICK": RETURN
1078 D$ = " CHIPMUNK":PZ(27) = 12
     :N$ = "HACKY": RETURN
1079 D$ = " DRINK MADE  WITH HERB
     S":PZ(27) = 14:N$ = "JULEP":
      RETURN
1080 D$ = " PRIMAL STATEOF THE UN
     IVERSE":PZ(27) = 17:N$ = "CH
     AOS": RETURN
1081 D$ = " A FACTION;  A CABAL":
     PZ(27) = 11:N$ = "JUNTO": RETURN

1082 D$ = " A KNOT OR   LUMP":PZ(
     27) = 15:N$ = "KNURL": RETURN

1083 D$ = " EMITS A NOX-IOUS ODOR
     ":PZ(27) = 18:N$ = "ZORIL": RETURN
1084 D$ = "HEMICELLULOSEIN PLANT
     WASTES":PZ(27) = 19:N$ = "XY
     LEM": RETURN
1085 D$ = " ADOLENSCENCE":PZ(27) =
     11:N$ = "YOUTH": RETURN
1086 D$ = " USED TO MAKEPORCELAN"
     :PZ(27) = 14:N$ = "PASTE": RETURN

1087 D$ = " AFRICAN ODD TOED MAMM
     AL":PZ(27) = 16:N$ = "RHINO"
     : RETURN
1088 D$ = " WRINKLES":PZ(27) = 11
     :N$ = "RUGAE": RETURN
1089 D$ = " A SHARP     VIBRANT S
     OUND":PZ(27) = 12:N$ = "TWAN
     G": RETURN
1090 D$ = " A POINTED   ARCH":PZ(
     27) = 15:N$ = "OGIVE": RETURN

1091 D$ = " AN AFRICAN  RUMINANT"
     :PZ(27) = 19:N$ = "OKAPI": RETURN

1092 D$ = " A FIBER USEDFOR CARPE
     T":PZ(27) = 14:N$ = "ISTLE":
      RETURN
1093 D$ = " TO MAKE A   GOD OF":P
     Z(27) = 14:N$ = "DEIFY": RETURN

1094 D$ = " THIN LAYERS OF ROCKS"
     :PZ(27) = 13:N$ = "FOLIA": RETURN

1095 D$ = " CRYSTALS OF ICE":PZ(2
     7) = 10:N$ = "FROST": RETURN

1096 D$ = " THE NUMBER  SIX":PZ(2
     7) = 11:N$ = "HEXAD": RETURN

1097 D$ = " AN ANCIENT  DRINKING
     CUP":PZ(27) = 19:N$ = "KYLIX
     ": RETURN
1098 D$ = " A YOUNG     PIGEON":P
     Z(27) = 16:N$ = "SQUAB": RETURN

1099 D$ = " FLAT; DULL; INSIPID":
     PZ(27) = 12:N$ = "VAPID": RETURN

1100 D$ = " A GRAVE-ROB-BER":PZ(2
     7) = 10:N$ = "GHOUL": RETURN

1101 D$ = "OBTAINED FROMAPPLES":P
     Z(27) = 11:N$ = "MALIC": RETURN

1102 D$ = " ANY SPIRAL":PZ(27) =
     10:N$ = "HELIX": RETURN
1103 D$ = " A PETTICOAT":PZ(27) =
     14:N$ = "JUPON": RETURN
1104 D$ = " SLANG FOR ANOPIUM ADD
     ICT":PZ(27) = 16:N$ = "GHOST
     ": RETURN
1105 D$ = "GROWING UNDERTHE SNOW"
     :PZ(27) = 11:N$ = "NIVAL": RETURN

1106 D$ = " A CUP":PZ(27) = 17:N$
      = "CALIX": RETURN
1107 D$ = "VIRUS USED INVACCINATI
     ON":PZ(27) = 19:N$ = "LYMPH"
     : RETURN
1108 D$ = "AN ANTHROPOIDAPE":PZ(2
     7) = 11:N$ = "ORANG": RETURN

1109 D$ = " TO FINE OR  TO CHEAT"
     :PZ(27) = 17:N$ = "MULCT": RETURN

1110 D$ = " A MOUNTAIN  PASS":PZ(
     27) = 12:N$ = "GHANT": RETURN

1111 D$ = " A HANGMAN":PZ(27) = 1
     3:N$ = "KETCH": RETURN
1112 D$ = " TO STIMULATEOR AROUSE
     ":PZ(27) = 16:N$ = "PIQUE": RETURN
1113 D$ = " MOUNTAIN    SPINACH":
     PZ(27) = 12:N$ = "ORACH": RETURN

1114 D$ = " ANY SHIELD":PZ(27) =
     19:N$ = "AEGIS": RETURN
1115 D$ = " USED AS A   FUEL OIL"
     :PZ(27) = 14:N$ = "MAZUT": RETURN

1116 D$ = " FERMENTED   MILK":PZ(
     27) = 13:N$ = "KEFIR": RETURN

1117 D$ = " AN EARRED   SEAL":PZ(
     27) = 9:N$ = "OTARY": RETURN

1118 D$ = " AN ANCIENT  JEWEL CAS
     E":PZ(27) = 20:N$ = "PYXIS":
      RETURN
1119 D$ = " A CHARLATAN":PZ(27) =
     12:N$ = "QUACK": RETURN
1120 D$ = " A BIN FOR   SATLING F
     ISH":PZ(27) = 11:N$ = "KENCH
     ": RETURN
1121 D$ = "OLD-FASHIONED":PZ(27) =
     13:N$ = "FUSTY": RETURN
1122 D$ = " ANY WAR FOR A FAITH":
     PZ(27) = 15:N$ = "JIHAD": RETURN

1123 D$ = " ANY FORM OF DENTINE":
     PZ(27) = 10:N$ = "IVORY": RETURN

1124 D$ = " A MILKY     EXUDATION
     ":PZ(27) = 16:N$ = "OPIUM": RETURN
1125 D$ = " RELATING TO BLOOD":PZ
     (27) = 9:N$ = "HEMIC": RETURN

1126 D$ = " OCCURRING BYLUCKY CHA
     NCE":PZ(27) = 11:N$ = "FLUKY
     ": RETURN
1127 D$ = " A RIDING    WHIP":PZ(
     27) = 14:N$ = "QUIRT": RETURN

1128 D$ = " MONEY":PZ(27) = 9:N$ =
     "LUCRE": RETURN
1129 D$ = " A HORN":PZ(27) = 14:N
     $ = "CORNU": RETURN
```

Calculated-GOSUB routine.

```
2000  IF PUZ < 43 THEN  ON PUZ GOTO
      1000,1001,1002,1003,1004,100
      5,1006,1007,1008,1009,1010,1
      011,1012,1013,1014,1015,1016
      ,1017,1018,1019,1020,1021,10
      22,1023,1024,1025,1026,1027,
      1028,1029,1030,1031,1032,103
      3,1034,1035,1036,1037,1038,1
      039,1040,1041
2010  IF PUZ < 85 THEN  ON PUZ -
      42 GOTO 1042,1043,1044,1045,
      1046,1047,1048,1049,1050,105
      1,1052,1053,1054,1055,1056,1
      057,1058,1059,1060,1061,1062
      ,1063,1064,1065,1066,1067,10
      68,1069,1070,1071,1072,1073,
      1074,1075,1076,1077,1078,107
      9,1080,1081,1082,1083
2020  IF PUZ < 127 THEN  ON PUZ -
      84 GOTO 1084,1085,1086,1087,
      1088,1089,1090,1091,1092,109
      3,1094,1095,1096,1097,1098,1
      099,1100,1101,1102,1103,1104
      ,1105,1106,1107,1108,1109,11
      10,1111,1112,1113,1114,1115,
      1116,1117,1118,1119,1120,112
      1,1122,1123,1124,1125
2030  ON PUZ - 126 GOTO 1126,1127
      ,1128,1129
```

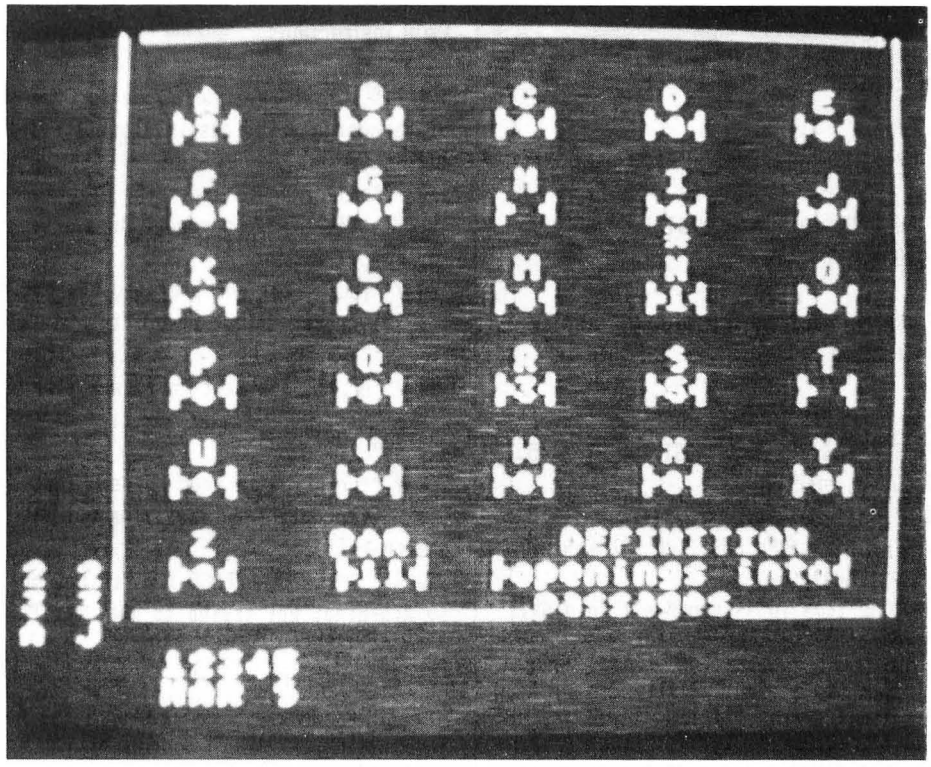
Atari Version

Setup for start of game.

```
0 REM CHALLENGE by Elrhea M. Bigham 2,
1981
1 GRAPHICS 0:SETCOLOR 2,12,1:? "
     CHALLENGE!":? "
                 ":REM 36 CTRL-M'S
2 POKE 752,1
3 GOSUB 500
5 ? "NAME OF PLAYER #1";:DIM P$(10),P1
$(10):INPUT P$:? :? "NAME OF PLAYER #2
";:INPUT P1$
6 PUZ=INT(RND(0)*130)
7 DIM PZ(30),W(2),A$(2),D$(30),GS$(20)
,N$(5),SC(2),GUS(2):A$="*":X=7:Y=1:D=1
:PL=0:ZP=0:W(0)=0:W(1)=0
8 SC(0)=0:SC(1)=0:GUS(0)=0:GUS(1)=0
9 FOR I=1 TO 30:PZ(I)=0:NEXT I:GOSUB P
UZ+1000:FOR I=1 TO 5:PZ(ASC(N$(I,I))-6
4)=I:NEXT I
```

Print Screen display.

```
10 REM 2ND STRING IN LNS. 15-50 IS:
    1 SPACE + 5 GROUPS OF (3 SPACES
    + CTRL-A + CTRL-T + CTRL-D)
15 GRAPHICS 0:SETCOLOR 2,8,1:? :? :? "
     A     B     C     D     E":? "
                              "
20 ? :? "     F     G     H     I
J":? "                              "
30 ? :? "     K     L     M     N
O":? "                              "
40 ? :? "     P     Q     R     S
T":? "                              "
50 ? :? "     U     V     W     X
Y":? "                              "
58 REM 2ND STRING IN LN. 60 IS:
    4 SPACES +CTRL-A +CTRL-T +CTRL-D
    +3 SPACES +CTRL-A + 2 CTRL-T'S
59 REM +CTRL-D +2 SPACES +CTRL-A
    +13 CTRL-T'S +CTRL-D
60 ? :? "     Z    PAR.      DEFINITION
":? "                               "
64 POKE 752,1:POSITION 5,0:? "
                    ":REM 30 CTRL-M'
S
65 FOR I=0 TO 19:POSITION 34,I:? " ":N
EXT I:REM SHIFT-=
66 FOR I=33 TO 5 STEP -1:POSITION I,19
:? " ":NEXT I:FOR J=19 TO 0 STEP -1:PO
```




```
SITION 4,J:? " ":NEXT J:REM SHIFT--; S
HIFT-=
67 IF GS=1 THEN RETURN

69 POSITION 6,21:? "12345":POSITION 1,
19:? "0 0":POSITION 1,20:? P$(1,1);" "
;P1$(1,1):GOSUB 125
70 POSITION X,Y:? A$
```

Gets player's move or guess.

```
78 IF T>=215 THEN FOR J=1 TO 5:SOUND 0
,75,12,8:FOR I=1 TO 75:NEXT I:SOUND 0,
0,0,0:NEXT J:T=0:IF PL>1 THEN PL=0
79 T=T+1:IF T>=215 THEN PL=PL+1:POSITI
ON X,Y:? " ":X=7:Y=1:D=1:GOTO 78
80 ST=STICK(PL):SOUND 0,X+D,10,8:IF ST
=7 THEN POSITION X,Y:? " ":X=X+6:D=D+1
:GOTO 100
81 IF ST=14 THEN POSITION X,Y:? " ":Y=
Y-3:D=D-5:GOTO 100
82 IF ST=13 THEN POSITION X,Y:? " ":Y=
Y+3:D=D+5:GOTO 100
83 IF ST=11 THEN POSITION X,Y:? " ":X=
X-6:D=D-1:GOTO 100
84 IF ST=15 THEN SOUND 0,0,0,0
85 IF PEEK(764)<>255 AND ST1<>1 THEN P
OSITION 17,20:? "GUESS";:INPUT GS$:GS=
1:ST1=1
86 IF GS=1 AND GS$<>N$ THEN POSITION 1
7,20:? "            ":GS=0:GOTO 80
87 IF GS=1 AND GS$=N$ THEN ST1=0:GOTO
210
88 IF PEEK(764)<>255 AND ST1=1 THEN PO
KE 764,255
99 IF STRIG(PL)=0 THEN SOUND 0,0,0,0:G
OTO 120
```

Check for a legal move.

```
100 FOR J=1 TO 50:NEXT J:IF D<=26 AND
D>=1 AND X<=31 AND Y<>16 AND X>=7 THEN
 POSITION X,Y:? A$:POKE 77,0:GOTO 79
101 IF X>7 AND Y=16 THEN X=7:D=26:POSI
TION X,Y:? A$:GOTO 79
105 IF D<1 OR D>26 THEN X=7:Y=1:D=1:GO
TO 79
106 IF X<7 THEN X=7:D=D+1:POSITION X,Y
:? A$:GOTO 79
110 IF X>=32 THEN X=7:Y=Y+3:POSITION X
,Y:? A$:GOTO 79
111 POSITION X,Y:? A$:GOTO 79
120 IF D<>28 THEN 141
```

Print definition and par for word.

```
125 LN=LEN(D$):POSITION 19,18:FOR I=1
TO LN:IF I<=13 THEN ? D$(I,I);:SOUND 0
,I+10,10,8:NEXT I
130 IF I=14 THEN POSITION 20,19
135 IF I>=14 AND LN>13 THEN ? D$(I,I);
:NEXT I
```

continued on next page

continued from previous page

```
140 POSITION 13,18:SOUND 0,0,0,0:? PZ(
27):RETURN
```

Check if a letter was already used.

```
141 IF PZ(D)<0 THEN FOR I=75 TO 55 STE
P -1:SOUND 0,I,10,8:NEXT I:SOUND 0,0,0
,0:GOTO 79
```

Erase letter from board and display its position in the word.

```
149 IF PZ(D)<>0 THEN POSITION X,Y+2:?
PZ(D):IF D<>27 THEN POSITION PZ(D)+5,2
2:GUS(PL)=GUS(PL)+1
150 IF PZ(D)=0 THEN POSITION X,Y+2:? "
 ":IF D<>27 THEN POSITION PZ(D)+5,22:G
US(PL)=GUS(PL)+1
151 FOR I=25 TO 35:SOUND 0,I,10,8:NEXT
 I:SOUND 0,0,0,0
152 IF D=27 THEN 160
153 IF PZ(D)<>0 THEN FOR I=25 TO 35:SO
UND 0,I,10,8:NEXT I:? N$(PZ(D),PZ(D)):
ZP=ZP+1:SOUND 0,0,0,0:SC(PL)=SC(PL)+1
154 POSITION PL+1,18:? SC(PL):IF PL=1
THEN POSITION PL+1,18:? " ";SC(PL)
155 IF PZ(D)=0 THEN PL=PL+1:ST1=0:IF P
L>1 THEN PL=0
160 PZ(D)=-1:POSITION 1,19:? GUS(0);"
";GUS(1)
200 IF ZP<5 AND GS=0 THEN T=0:GOTO 79
205 REM
```

Routine for a word guess correctly.

```
210 W(PL)=W(PL)+1:POSITION 15,21:? P$,
P1$:POSITION 15,22:? W(0),W(1)
211 FOR K=75 TO 85 STEP +3:SOUND 0,K,1
0,8:POSITION 19,18:? "              ":S
OUND 0,0,0,0
212 POKE 752,1:POSITION 5,0:? "
                    ":FOR I=0 TO 19
:POSITION 34,I:? " ":NEXT I
213 FOR I=33 TO 5 STEP -1:POSITION I,1
9:? " ":NEXT I:FOR J=19 TO 0 STEP -1:P
OSITION 4,J:? " ":NEXT J
215 SETCOLOR 2,K-74,K-74:GS=1:GOSUB 65
:GOSUB 125:NEXT K:SETCOLOR 2,0,0
```

Setup for next game.

```
220 PUZ=PUZ+1:X=7:Y=1:D=1:PL=0:ZP=0:GS
=0:T=0:ST1=0
230 IF STRIG(1)=0 OR STRIG(0)=0 THEN P
OKE 77,0:GOTO 8
232 IF PUZ>129 THEN PUZ=0
235 GOTO 230
```

Print instructions.

```
500 ? "   This is a word game for two
playersusing joysticks in slots 1 and
2.  The";
510 ? "object is to guess the 5-letter
 word  before your opponent does.  By
moving"
520 ? "your stick and pressing the but
ton,   you can select any of the 26 le
tters."
530 ? "Or, if you want to take a stab
at the word, you may at any time durin
g your"
540 ? "turn type in your guess.  There
 is a  1-minute time limit for guessin
g each letter."
550 ? "   The CHALLENGE lies in not on
ly win-ning, but trying to beat the PA
R for"
560 ? "each word as well."
570 ? "   You may play as many games a
s you  want, and the computer will kee
p track";
580 ? "of the score.  Just press the F
IRE    button after each game to conti
nue."
590 ? :FOR I=1 TO 500:NEXT I:RETURN
```

Definitions and word data.

```
1000 D$=" herb-carrot family":PZ(27)=9
:N$="ANISE":RETURN
1001 D$=" vigor,plump cheerful":PZ(27)
=11:N$="BUXOM":RETURN
1002 D$=" a supporting timber":PZ(27)=
11:N$="JOIST":RETURN
1003 D$=" gather field leavings":PZ(27
)=9:N$="GLEAN":RETURN
1004 D$=" fashion, in  style":PZ(27)=1
0:N$="VOGUE":RETURN
1005 D$=" to imply or  hint":PZ(27)=10
:N$="INFER":RETURN
1006 D$=" a tree":PZ(27)=10:N$="LARCH"
:RETURN
1007 D$="ecclesiasti- cal meeting":PZ(
27)=11:N$="SYNOD":RETURN
1008 D$="openings intopassages":PZ(27)
=11:N$="NARES":RETURN
1009 D$=" worn, tired, vexed":PZ(27)=9
:N$="WEARY":RETURN
1010 D$="detest,loathe":PZ(27)=12:N$="
ABHOR":RETURN
1011 D$=" list, point,hand of clock":P
Z(27)=12:N$="INDEX":RETURN
1012 D$=" a mountain  range":PZ(27)=9:
N$="OZARK":RETURN
1013 D$=" a contagioushorse disease":P
Z(27)=11:N$="FARCY":RETURN
1014 D$=" a mass or a grinding tooth":
PZ(27)=9:N$="MOLAR":RETURN
1015 D$=" peaceful re-lations":PZ(27)=
9:N$="AMITY":RETURN
1016 D$=" pin fitting a hole":PZ(27)=1
0:N$="DOWEL":RETURN
1017 D$=" accent on a word":PZ(27)=14:
N$="ICTUS":RETURN
1018 D$=" plate armor for the thigh":P
Z(27)=15:N$="QUISH":RETURN
1019 D$=" cemetery,se-creting cavity":
PZ(27)=10:N$="CRYPT":RETURN
1020 D$=" shadowy,hazy":PZ(27)=9:N$="V
AGUE":RETURN
1021 D$=" conical headdress,cap":PZ(27
)=11:N$="TOQUE":RETURN
1022 D$=" correspond  in sound ":PZ(27
)=9:N$="RHYME":RETURN
1023 D$=" to apply    cosmetics":PZ(27
)=10:N$="PAINT":RETURN
1024 D$=" a natural   rubber":PZ(27)=1
4:N$="PULAY":RETURN
1025 D$=" darkened,   gloomy":PZ(27)=1
0:N$="MURKY":RETURN
1026 D$=" broom,littlebunch":PZ(27)=11
:N$="WHISK":RETURN
1027 D$=" nothing":PZ(27)=10:N$="ZILCH
":RETURN
1028 D$=" simpletons":PZ(27)=16:N$="LA
MBS":RETURN
1029 D$=" Indian hemp narcotic":PZ(27)
=14:N$="BHANG":RETURN
1030 D$=" being sur-  rounded":PZ(27)=
12:N$="MIDST":RETURN
1031 D$=" to be,occur":PZ(27)=9:N$="EX
IST":RETURN
1032 D$=" rounded     bunches":PZ(27)=
11:N$="KNOBS":RETURN
1033 D$=" any plant   stem":PZ(27)=12:
N$="HAULM":RETURN
1034 D$=" loosely, a  ghost":PZ(27)=10
:N$="ZOMBI":RETURN
1035 D$=" male        fastener":PZ(27)
=9:N$="SCREW":RETURN
1036 D$=" shapes hot  glass":PZ(27)=13
:N$="PUNTY":RETURN
1037 D$=" dusky hue":PZ(27)=9:N$="UMBE
R":RETURN
1038 D$=" a windlass, a crank":PZ(27)=
9:N$="WINCH":RETURN
1039 D$=" a varicose  vein":PZ(27)=11:
N$="VARIX":RETURN
1040 D$=" of extreme  dryness,arid":PZ
(27)=18:N$="XERIC":RETURN
1041 D$=" any ape or  monkey":PZ(27)=1
3:N$="JACKO":RETURN
1042 D$=" jersey worn by sailors":PZ(2
7)=9:N$="FROCK":RETURN
```

```
1043 D$=" the largest flower":PZ(27)=1
4:N$="KRUBI":RETURN
1044 D$=" shelters":PZ(27)=11:N$="ABRI
S":RETURN
1045 D$=" a bovine   ruminant":PZ(27)
=14:N$="BISON":RETURN
1046 D$=" golf term":PZ(27)=10:N$="DOR
MY":RETURN
1047 D$="  emaciated, meager,thin":PZ(
27)=9:N$="GAUNT":RETURN
1048 D$=" an open shedor hut":PZ(27)=9
:N$="HOVEL":RETURN
1049 D$=" layman":PZ(27)=9:N$="LAITY":
RETURN
1050 D$="semi-preciousyellow stone":PZ
(27)=9:N$="TOPAZ":RETURN
1051 D$=" grind,striketogether":PZ(27)
=13:N$="GNASH":RETURN
1052 D$=" a ravine":PZ(27)=12:N$="GULC
H":RETURN
1053 D$=" a recess in a wall":PZ(27)=1
1:N$="NICHE":RETURN
1054 D$="one who jokes":PZ(27)=16:N$="
JOKER":RETURN
1055 D$=" power":PZ(27)=10:N$="MIGHT":
RETURN
1056 D$=" to burn or  scorch":PZ(27)=9
:N$="PARCH":RETURN
1057 D$=" to endure orto wait for":PZ(
27)=10:N$="ABIDE":RETURN
1058 D$=" mimicry":PZ(27)=15:N$="APERY
":RETURN
1059 D$="a minute ori-fice; pore":PZ(2
7)=15:N$="STOMA":RETURN
1060 D$=" a radical":PZ(27)=17:N$="ULT
RA":RETURN
1061 D$=" fold of skinbetween fingers"
:PZ(27)=18:N$="PLICA":RETURN
1062 D$=" Easter":PZ(27)=14:N$="PASCH"
:RETURN
1063 D$=" crude cream of tartar":PZ(27
)=12:N$="ARGOL":RETURN
1064 D$=" a gaseous  element":PZ(27)=
11:N$="ARGON":RETURN
1065 D$=" stored for  future use":PZ(2
7)=12:N$="HIVED":RETURN
1066 D$="potters' clay":PZ(27)=10:N$="
ARGIL":RETURN
1067 D$=" yellow com- pact limestone":
PZ(27)=12:N$="CHALK":RETURN
1068 D$=" to express  anger":PZ(27)=9:
N$="FUMED":RETURN
1069 D$=" any edible  purple seaweed":
PZ(27)=13:N$="LAVER":RETURN
1070 D$=" deserve":PZ(27)=10:N$="MERIT
":RETURN
1071 D$=" a trickster;rascal":PZ(27)=1
1:N$="ROGUE":RETURN
1072 D$=" tempestuous;stormy":PZ(27)=1
2:N$="WINDY":RETURN
1073 D$="Asian foxlikecarnivore":PZ(27
)=18:N$="ZIBET":RETURN
1074 D$="secreted and used as perfume"
:PZ(27)=18:N$="CIVET":RETURN
1075 D$=" circular   motion":PZ(27)=1
1:N$="GYRAL":RETURN
1076 D$="  bound with fetters":PZ(27)=
16:N$="GYVED":RETURN
1077 D$=" an Oriental outside garment"
:PZ(27)=14:N$="HAICK":RETURN
1078 D$=" chipmunk":PZ(27)=12:N$="HACK
Y":RETURN
1079 D$=" drink made  with herbs":PZ(2
7)=14:N$="JULEP":RETURN
1080 D$=" primal stateof the universe"
:PZ(27)=17:N$="CHAOS":RETURN
1081 D$=" a faction;  a cabal":PZ(27)=
11:N$="JUNTO":RETURN
1082 D$=" a knot or   lump":PZ(27)=15:
N$="KNURL":RETURN
1083 D$=" emits a nox-ious odor":PZ(27
)=18:N$="ZORIL":RETURN
1084 D$="hemicellulosein plant wastes"
:PZ(27)=19:N$="XYLEM":RETURN
1085 D$=" adolescence":PZ(27)=11:N$="Y
OUTH":RETURN
1086 D$=" used to makeporcelan":PZ(27)
=14:N$="PASTE":RETURN
1087 D$=" African odd toed mammal":PZ(
27)=16:N$="RHINO":RETURN
1088 D$=" wrinkles":PZ(27)=11:N$="RUGA
E":RETURN
1089 D$=" a sharp    vibrant sound":P
Z(27)=12:N$="TWANG":RETURN
1090 D$=" a pointed   arch":PZ(27)=15:
N$="OGIVE":RETURN
1091 D$=" an African  ruminant":PZ(27)
=19:N$="OKAPI":RETURN
1092 D$=" a fiber usedfor carpet":PZ(2
7)=14:N$="ISTLE":RETURN
1093 D$=" to make a   god of":PZ(27)=1
4:N$="DIEFY":RETURN
1094 D$=" thin layers of rocks":PZ(27)
=13:N$="FOLIA":RETURN
1095 D$=" crystals of ice":PZ(27)=10:N
$="FROST":RETURN
1096 D$=" the number  six":PZ(27)=11:N
$="HEXAD":RETURN
1097 D$=" an ancient  drinking cup":PZ
(27)=19:N$="KYLIX":RETURN
1098 D$=" a young     pigeon":PZ(27)=1
6:N$="SQUAB":RETURN
1099 D$=" flat; dull; insipid":PZ(27)=
12:N$="VAPID":RETURN
1100 D$=" a grave-rob-ber":PZ(27)=10:N
$="GHOUL":RETURN
1101 D$="obtained fromapples":PZ(27)=1
1:N$="MALIC":RETURN
1102 D$=" any spiral":PZ(27)=10:N$="HE
LIX":RETURN
1103 D$=" a petticoat":PZ(27)=14:N$="J
UPON":RETURN
1104 D$=" slang for anopium addict":PZ
(27)=16:N$="GHOST":RETURN
1105 D$="growing underthe snow":PZ(27)
=11:N$="NIVAL":RETURN
1106 D$=" a cup":PZ(27)=17:N$="CALIX":
RETURN
1107 D$="virus used invaccination":PZ(
27)=19:N$="LYMPH":RETURN
1108 D$="an anthropoidape":PZ(27)=11:N
$="ORANG":RETURN
1109 D$=" to fine or  to cheat":PZ(27)
=17:N$="MULCT":RETURN
1110 D$=" a mountain  pass":PZ(27)=12:
N$="GHANT":RETURN
1111 D$=" a hangman":PZ(27)=13:N$="KET
CH":RETURN
1112 D$=" to stimulateor arouse":PZ(27
)=16:N$="PIQUE":RETURN
1113 D$=" mountain    spinach":PZ(27)=
12:N$="ORACH":RETURN
1114 D$=" any shield":PZ(27)=19:N$="AE
GIS":RETURN
1115 D$=" used as a   fuel oil":PZ(27)
=14:N$="MAZUT":RETURN
1116 D$=" fermented   milk":PZ(27)=13:
N$="KEFIR":RETURN
1117 D$=" an earred   seal":PZ(27)=9:N
$="OTARY":RETURN
1118 D$=" an ancient  jewel case":PZ(2
7)=20:N$="PYXIS":RETURN
1119 D$=" a charlatan":PZ(27)=12:N$="Q
UACK":RETURN
1120 D$=" a bin for   salting fish":PZ
(27)=11:N$="KENCH":RETURN
1121 D$="old-fashioned":PZ(27)=13:N$="
FUSTY":RETURN
1122 D$=" any war for a faith":PZ(27)=
15:N$="JIHAD":RETURN
1123 D$=" any form of dentine":PZ(27)=
10:N$="IVORY":RETURN
1124 D$=" a milky     exudation":PZ(27
)=16:N$="OPIUM":RETURN
1125 D$=" relating to blood":PZ(27)=9:
N$="HEMIC":RETURN
1126 D$=" occurring bylucky chance":PZ
(27)=11:N$="FLUKY":RETURN
1127 D$=" a riding    whip":PZ(27)=14:
N$="QUIRT":RETURN
1128 D$=" money":PZ(27)=9:N$="LUCRE":R
ETURN
1129 D$=" a horn":PZ(27)=14:N$="CORNU"
:RETURN
```

continued on next page

continued from previous page

S-80 Version

Setup for start of game.

```
1 CLS:PRINT@26,"CHALLENGE!":PRINTTAB(21);"BY: ELRHEA M. BIGHAM":
PRINTTAB(15);"S-80 TRANSLATION BY: ALAN J ZETT"
2 PRINT"------------------------------------------------------------
--------";
3 CLEAR500:GOSUB 500
5 PRINT@256,CHR$(31):INPUT"ENTER NAME OF PLAYER #1";P$:PRINT:INP
UT"ENTER NAME OF PLAYER #2";P1$
6 PUZ=RND(130)
7 DIM PZ(30),W(2),SC(2),GUS(2):A$=CHR$(92):X=4:Y=1:D=1:PL=0:ZP=0
:W(0)=0:W(1)=0
8 SC(0)=0:SC(1)=0:GUS(0)=0:GUS(1)=0
9 FORI=1TO30:PZ(I)=0:NEXTI:GOSUB2000:FORI=1TO5:PZ(ASC(MID$(N$,I,
1))-64)=I:NEXTI
```

Print Screen display.

```
10 CLS:PRINT@2%64+4,"A     B     C     D     E     F     G     H
     I     J":PRINTTAB(3);")+(   )+(   )+(   )+(   )+(   )+(   )
+(   )+(   )+(   )+("
20 PRINT@5%64+4,"K     L     M     N     O     P     Q     R
 S     T":PRINTTAB(3);")+(   )+(   )+(   )+(   )+(   )+(   )+(
 )+(   )+(   )+("
30 PRINT@8%64+4,"U     V     W     X     Y     Z    PAR     DEFI
NITION":PRINTTAB(3);")+(   )+(   )+(   )+(   )+(   )+(  )+++( )
++++++++++++++("
60 FORI=0TO63:PRINT@I,CHR$(191);:NEXTI:FORI=61TO701STEP64:PRINT@
I,STRING$(3,191);:NEXTI:FORI=767TO704STEP-1:PRINT@I,CHR$(191);:N
EXTI:FORI=701TO61STEP-64:PRINT@I,STRING$(6,191);:NEXTI
65 IFGS=1THENRETURNELSEPRINT@832," 0   0";TAB(13);"12345":PRINT"
 ";LEFT$(P$,1);"   ";LEFT$(P1$,1);:GOSUB125
70 PRINT@X+Y%64,A$;
```

Gets player's move or guess.

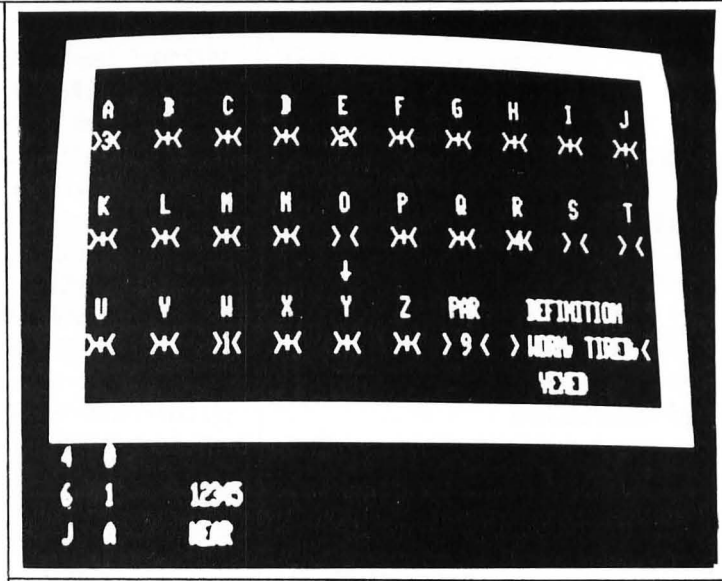
```
78 IFT>=215THENT=0:PRINT@960,"YOU TOOK TOO LONG! YOU LOSE YOUR T
URN!";:FORI=1TO1500:NEXT:PRINT@960,STRING$(55,32);:IFPL>1THENPL=
0
79 T=T+1:IFT>=215THENPL=PL+1:PRINT@X+Y%64," ";:X=4:Y=1:D=1:GOTO7
8
80 ST=PEEK(14400):IFST=64THENPRINT@X+Y%64," ";:X=X+6:D=D+1:GOTO1
00
81 IFST=8THENPRINT@X+Y%64," ";:Y=Y-3:D=D-10:GOTO100
82 IFST=16THENPRINT@X+Y%64," ";:Y=Y+3:D=D+10:GOTO100
83 IFST=32THENPRINT@X+Y%64," ";:X=X-6:D=D-1:GOTO100
85 IFINKEY$=" "ANDST(1)<>1THENPRINT@934,"GUESS ";:INPUTGS$:GS=1:
ST(1)=1
86 IFGS=1ANDGS$<>N$THENPRINT@934,STRING$(20,32);:GS=0:GOTO80
87 IFGS=1ANDGS$=N$THENST(1)=0:GOTO210
88 IFINKEY$<>""THEN88
99 IFPEEK(14400)=1THEN120
```

Check for a legal move.

```
100 FORJ=1TO10:NEXTJ:IFD<=26ANDD>=1ANDX<=58ANDX>=4ANDY<>7THENPRI
NT@X+Y%64,A$;:GOTO79
101 IFX>34ANDY=7THENX=34:D=26:PRINT@X+Y%64,A$;:GOTO79
105 IFD<1ORD>26THENX=4:Y=1:D=1:GOTO79
106 IFX<4THENX=58:Y=Y-3:PRINT@X+Y%64,A$;:GOTO79
110 IFX>=59THENX=4:Y=Y+3:PRINT@X+Y%64,A$;:GOTO79
111 PRINT@X+Y%64,A$;:GOTO79
120 IFD<>28THEN141
```

Print definition and par for word.

```
125 LN=LEN(D$):PRINT@622,;:FORI=1TOLN:IFI<=13THENPRINTMID$(D$,I,
1);:NEXTI
130 IFI=14THENPRINT@687,;
135 IFI>=14ANDLN>13THENPRINTMID$(D$,I,1);:NEXTI
140 PRINT@615,LEFT$(STR$(PZ(27))+" ",3);:RETURN
```



Check if a letter was already used.

```
141 IFPZ(D)<0THEN79
```

Erase letter from board and display its position in the word.

```
149 IFPZ(D)<>0THENPRINT@X+(Y+2)%64,RIGHT$(STR$(PZ(D)),1);:IFD<>2
7THENPRINT@(PZ(D)+12)+896,;:GUS(PL)=GUS(PL)+1
150 IFPZ(D)=0THENPRINT@X+(Y+2)%64," ";:IFD<>27THENPRINT@(PZ(D)+1
2)+896,;:GUS(PL)=GUS(PL)+1
152 IFD=27THEN160
153 IFPZ(D)<>0THENPRINTMID$(N$,PZ(D),1);:ZP=ZP+1:SC(PL)=SC(PL)+1
154 IFPL=0THENPRINT@768,SC(PL);ELSEIFPL=1THENPRINT@772,SC(PL);
155 IFPZ(D)=0THENPL=PL+1:ST(1)=0:IFPL>1THENPL=0
160 PZ(D)=-1:PRINT@832,GUS(0);" ";GUS(1);
200 IFZP<5ANDGS=0THENT=0:GOTO79
```

Routine for a word guess correctly.

```
210 W(PL)=W(PL)+1:PRINT@768,CHR$(31);:PRINT@916,P$,P1$;:PRINT@98
0,W(0),W(1);
211 FORK=0TO10STEP3:PRINT@622,STRING$(13,32);:PRINT@686,STRING$(
14,32);
212 FORI=0TO63:PRINT@I,CHR$(32);:NEXTI:FORI=61TO701STEP64:PRINT@
I,STRING$(3,32);:NEXTI:FORI=767TO704STEP-1:PRINT@I,CHR$(32);:NEX
TI:FORI=701TO61STEP-64:PRINT@I,STRING$(6,32);:NEXTI
215 GS=1:GOSUB60:GOSUB125:NEXTK
```

Setup for next game.

```
220 PUZ=PUZ+1:X=4:Y=1:D=1:PL=0:ZP=0:GS=0:T=0:ST(1)=0
230 IFPEEK(14400)=1THEN8
232 IFPUZ>130THENPUZ=1
235 GOTO230
```

Print instructions.

```
500 PRINT"   THIS IS A WORD GAME FOR TWO PLAYERS. THE OBJECT OF
THE GAME  IS TO GUESS THE 5-LETTER WORD BEFORE YOUR OPPONENT DOE
S. BY     MOVING WITH THE ARROW KEYS YOU CAN POSITION THE 'CURSO
R' OVER"
```

```
510 PRINT"THE LETTER YOU THINK BELONGS IN THE WORD, AND PRESS <E
NTER>.":PRINT"   OR, IF YOU THINK YOU ALREADY KNOW THE WORD, PRE
SS THE <SPACE BAR> AND TYPE THE 5 LETTERS YOU THINK ARE CORRECT.
"
520 PRINT"   ALSO, THERE IS A TIME LIMIT ON HOW LONG IT TAKES YO
U TO PICK A LETTER. IF YOU HAVEN'T PICKED ONE WHEN THE TIME RUNS
 OUT YOU  LOSE YOUR TURN. ";
530 PRINT"THE CHALLENGE LIES NOT ONLY IN WINNING, BUT ALSOIN TRY
NG TO BEAT THE PAR FOR EACH WORD AS WELL.":PRINT:PRINT"PRESS <EN
TER> TO START EACH NEW GAME.";
540 IFINKEY$<>CHR$(13)THEN540ELSERETURN
999 STOP
```

Definitions and word data.

```
1000 D$=" HERB-CARROT FAMILY":PZ(27)=9:N$="ANISE":RETURN
1001 D$=" VIGOR,PLUMP CHEERFUL":PZ(27)=11:N$="BUXOM":RETURN
1002 D$=" A SUPPORTING TIMBER":PZ(27)=11:N$="JOIST":RETURN
1003 D$=" GATHER FIELD LEAVINGS":PZ(27)=9:N$="GLEAN":RETURN
1004 D$=" FASHION, IN  STYLE":PZ(27)=10:N$="VOGUE":RETURN
1005 D$=" TO IMPLY OR  HINT":PZ(27)=10:N$="INFER":RETURN
1006 D$=" A TREE":PZ(27)=10:N$="LARCH":RETURN
1007 D$="ECCLESIASTI- CAL MEETING":PZ(27)=11:N$="SYNOD":RETURN
1008 D$="OPENINGS INTOPASSAGES":PZ(27)=11:N$="NARES":RETURN
1009 D$=" WORN, TIRED, VEXED":PZ(27)=9:N$="WEARY":RETURN
1010 D$=" DETEST,LOATHE":PZ(27)=12:N$="ABHOR":RETURN
1011 D$=" LIST, POINT,HAND OF CLOCK":PZ(27)=12:N$="INDEX":RETURN
1012 D$=" A MOUNTAIN  RANGE":PZ(27)=9:N$="OZARK":RETURN
1013 D$=" A CONTAGIOUSHORSE DISEASE":PZ(27)=11:N$="FARCY":RETURN
1014 D$=" A MASS OR A GRINDING TOOTH":PZ(27)=9:N$="MOLAR":RETURN
1015 D$=" PEACEFUL RE-LATIONS":PZ(27)=9:N$="AMITY":RETURN
1016 D$=" PIN FITTING A HOLE":PZ(27)=10:N$="DOWEL":RETURN
1017 D$=" ACCENT ON A WORD":PZ(27)=14:N$="ICTUS":RETURN
1018 D$=" PLATE ARMOR FOR THE THIGH":PZ(27)=15:N$="QUISH":RETURN
1019 D$=" CEMETARY,SE-CRETING CAVITY":PZ(27)=10:N$="CRYPT":RETUR
N
1020 D$=" SHADOWY,HAZY":PZ(27)=9:N$="VAGUE":RETURN
1021 D$=" CONICAL HEADDRESS,CAP":PZ(27)=11:N$="TOQUE":RETURN
1022 D$=" CORRESPOND  IN SOUND":PZ(27)=9:N$="RHYME":RETURN
1023 D$=" TO APPLY    COSMETICS":PZ(27)=11:N$="PAINT":RETURN
1024 D$=" A NATURAL   RUBBER":PZ(27)=14:N$="PULAY":RETURN
1025 D$=" DARKENED,   GLOOMY":PZ(27)=10:N$="MURKY":RETURN
1026 D$=" BROOM,LITTLEBUNCH":PZ(27)=11:N$="WHISK":RETURN
1027 D$=" NOTHING":PZ(27)=10:N$="ZILCH":RETURN
1028 D$=" SIMPLETONS":PZ(27)=16:N$="LAMBS":RETURN
1029 D$=" INDIAN HEMP NARCOTIC":PZ(27)=14:N$="BHANG":RETURN
1030 D$=" BEING SUR-  ROUNDED":PZ(27)=12:N$="MIDST":RETURN
1031 D$=" TO BE,OCCUR":PZ(27)=9:N$="EXIST":RETURN
1032 D$=" ROUNDED     BUNCHES":PZ(27)=11:N$="KNOBS":RETURN
1033 D$=" ANY PLANT   STEM":PZ(27)=12:N$="HAULM":RETURN
1034 D$=" LOOSELY, A  GHOST":PZ(27)=10:N$="ZOMBI":RETURN
1035 D$=" MALE,FEMALE FASTENER":PZ(27)=9:N$="SCREW":RETURN
1036 D$=" SHAPES HOT  GLASS":PZ(27)=13:N$="PUNTY":RETURN
1037 D$=" DUSKY HUE":PZ(27)=9:N$="UMBER":RETURN
1038 D$=" A WINDLASS, A CRANK":PZ(27)=9:N$="WINCH":RETURN
1039 D$=" A VARICOSE  VEIN":PZ(27)=11:N$="VARIX":RETURN
1040 D$=" OF EXTREME  DRYNESS,ARID":PZ(27)=18:N$="XERIC":RETURN
1041 D$=" ANY APE OR  MONKEY":PZ(27)=13:N$="JACKO":RETURN
1042 D$=" JERESY WORN BY SAILORS":PZ(27)=9:N$="FROCK":RETURN
1043 D$=" THE LARGEST FLOWER":PZ(27)=14:N$="KRUBI":RETURN
1044 D$=" SHELTERS":PZ(27)=11:N$="ABRIS":RETURN
1045 D$=" A BOVINE   RUMINANT":PZ(27)=14:N$="BISON":RETURN
1046 D$=" GOLF TERM":PZ(27)=10:N$="DORMY":RETURN
1047 D$=" EMACIATED, MEAGER,THIN":PZ(27)=9:N$="GAUNT":RETURN
1048 D$=" AN OPEN SHEDOR HUT":PZ(27)=9:N$="HOVEL":RETURN
1049 D$=" LAYMAN":PZ(27)=9:N$="LAITY":RETURN
1050 D$="SEMI-PRECIOUSYELLOW STONE":PZ(27)=9:N$="TOPAZ":RETURN
1051 D$=" GRIND,STRIKETOGETHER":PZ(27)=13:N$="GNASH":RETURN
1052 D$=" A RAVINE":PZ(27)=12:N$="GULCH":RETURN
1053 D$=" A RECESS IN A WALL":PZ(27)=11:N$="NICHE":RETURN
1054 D$="ONE WHO JOKES":PZ(27)=16:N$="JOKER":RETURN
1055 D$=" POWER":PZ(27)=10:N$="STOMA":RETURN
1056 D$=" TO BURN OR  SCORCH":PZ(27)=9:N$="PARCH":RETURN
1057 D$=" TO ENDURE ORTO WAIT FOR":PZ(27)=10:N$="ABIDE":RETURN
1058 D$=" MIMICRY":PZ(27)=15:N$="APERY":RETURN
1059 D$="A MINUTE ORI-FICE; PORE":PZ(27)=15:N$="STOMA":RETURN
1060 D$=" A RADICAL":PZ(27)=17:N$="ULTRA":RETURN
1061 D$=" FOLD OF SKINBETWEEN FINGERS":PZ(27)=17:N$="ULTRA":RETU
RN
1062 D$=" EASTER":PZ(27)=14:N$="PASCH":RETURN
1063 D$=" CRUDE CREAM OF TARTAR":PZ(27)=12:N$="ARGOL":RETURN
1064 D$=" A GASEOUS   ELEMENT":PZ(27)=11:N$="ARGON":RETURN
1065 D$=" STORED FOR  FUTURE USE":PZ(27)=12:N$="HIVED":RETURN
1066 D$="POTTERS' CLAY":PZ(27)=10:N$="ARGIL":RETURN
1067 D$=" YELLOW  COM-PACT LIMESTONE":PZ(27)=12:N$="CHALK":RETUR
N
1068 D$=" TO EXPRESS  ANGER":PZ(27)=9:N$="FUMED":RETURN
1069 D$=" ANY EDIBLE  PURPLE SEAWEED":PZ(27)=13:N$="LAVER":RETUR
N
1070 D$=" DESERVE":PZ(27)=10:N$="MERIT":RETURN
1071 D$=" A TRICKSTER;RASCAL":PZ(27)=11:N$="ROGUE":RETURN
1072 D$=" TEMPESTUOUS;STORMY":PZ(27)=12:N$="WINDY":RETURN
1073 D$="ASIAN FOXLIKECARNIVORE":PZ(27)=18:N$="ZIBET":RETURN
1074 D$="SECRETED AND USED AS PERFUME":PZ(27)=18:N$="CIVET":RETU
RN
1075 D$=" CIRCULAR    MOTION":PZ(27)=11:N$="GYRAL":RETURN
1076 D$="  BOUND WITH FETTERS":PZ(27)=16:N$="GYVED":RETURN
1077 D$=" AN ORIENTAL OUTSIDE GARMENT":PZ(27)=14:N$="HAICK":RETU
RN
1078 D$=" CHIPMUNK":PZ(27)=12:N$="HACKY":RETURN
1079 D$=" DRINK MADE  WITH HERBS":PZ(27)=14:N$="JULEP":RETURN
1080 D$=" PRIMAL STATEOF UNIVERSE":PZ(27)=17:N$="CHAOS":RETURN
1081 D$=" A FACTION;  A CABAL":PZ(27)=11:N$="JUNTO":RETURN
1082 D$=" A KNOT OR   LUMP":PZ(27)=15:N$="KNURL":RETURN
1083 D$=" EMITS A NOX-IOUS ODOR":PZ(27)=18:N$="ZORIL":RETURN
1084 D$="HEMICELLULOSEIN PLANT WASTES":PZ(27)=19:N$="XYLEM":RETU
RN
1085 D$=" ADOLESCENCE":PZ(27)=11:N$="YOUTH":RETURN
1086 D$=" USED TO MAKEPOCELAN":PZ(27)=14:N$="PASTE":RETURN
1087 D$=" AFRICAN ODD TOED MAMMAL":PZ(27)=16:N$="RHINO":RETURN
1088 D$=" WRINKLES":PZ(27)=11:N$="RUGAE":RETURN
1089 D$=" A SHARP     VIBRANT SOUND":PZ(27)=12:N$="TWANG":RETURN
1090 D$=" A POINTED   ARCH":PZ(27)=15:N$="OGIVE":RETURN
1091 D$=" AN AFRICAN  RUMINANT":PZ(27)=19:N$="OKAPI":RETURN
1092 D$=" A FIBER USEDFOR CARPET":PZ(27)=14:N$="ISTLE":RETURN
1093 D$=" TO MAKE A   GOD OF":PZ(27)=14:N$="DEIFY":RETURN
1094 D$=" THIN LAYERS OF ROCKS":PZ(27)=13:N$="FOLIA":RETURN
1095 D$=" CRYSTALS OF ICE":PZ(27)=10:N$="FROST":RETURN
1096 D$=" THE NUMBER  SIX":PZ(27)=10:N$="HEXAD":RETURN
1097 D$=" AN ANCIENT  DRINKING CUP":PZ(27)=19:N$="KYLIX":RETURN
1098 D$=" A YOUNG     PIGEON":PZ(27)=16:N$="SQUAB":RETURN
1099 D$=" FLAT; DULL; INSIPPID":PZ(27)=12:N$="VAPID":RETURN
1100 D$=" A GRAVE-ROB-BER":PZ(27)=10:N$="GHOUL":RETURN
1101 D$="OBTAINED FROMAPPLES":PZ(27)=11:N$="MALIC":RETURN
1102 D$=" ANY SPIRAL":PZ(27)=10:N$="HELIX":RETURN
1103 D$=" A PETTICOAT":PZ(27)=14:N$="JUPON":RETURN
1104 D$=" SLANG FOR ANOPIUM ADDICT":PZ(27)=16:N$="GHOST":RETURN
1105 D$="GROWING UNDERTHE SNOW":PZ(27)=11:N$="NIVAL":RETURN
1106 D$=" A CUP":PZ(27)=17:N$="CALIX":RETURN
1107 D$="VIRUS  USED INVACCINATION":PZ(27)=19:N$="LYMPH":RETURN
1108 D$="AN ANTHROPOIDAPE":PZ(27)=11:N$="ORANG":RETURN
1109 D$=" TO FINE OR  TO CHEAT":PZ(27)=11:N$="ORANG":RETURN
1110 D$=" A MOUNTAIN  PASS":PZ(27)=12:N$="GHANT":RETURN
1111 D$=" A HANGMAN":PZ(27)=13:N$="KETCH":RETURN
1112 D$=" TO STIMULATEOR AROUSE":PZ(27)=16:N$="PIQUE":RETURN
```

continued on next page

continued from previous page

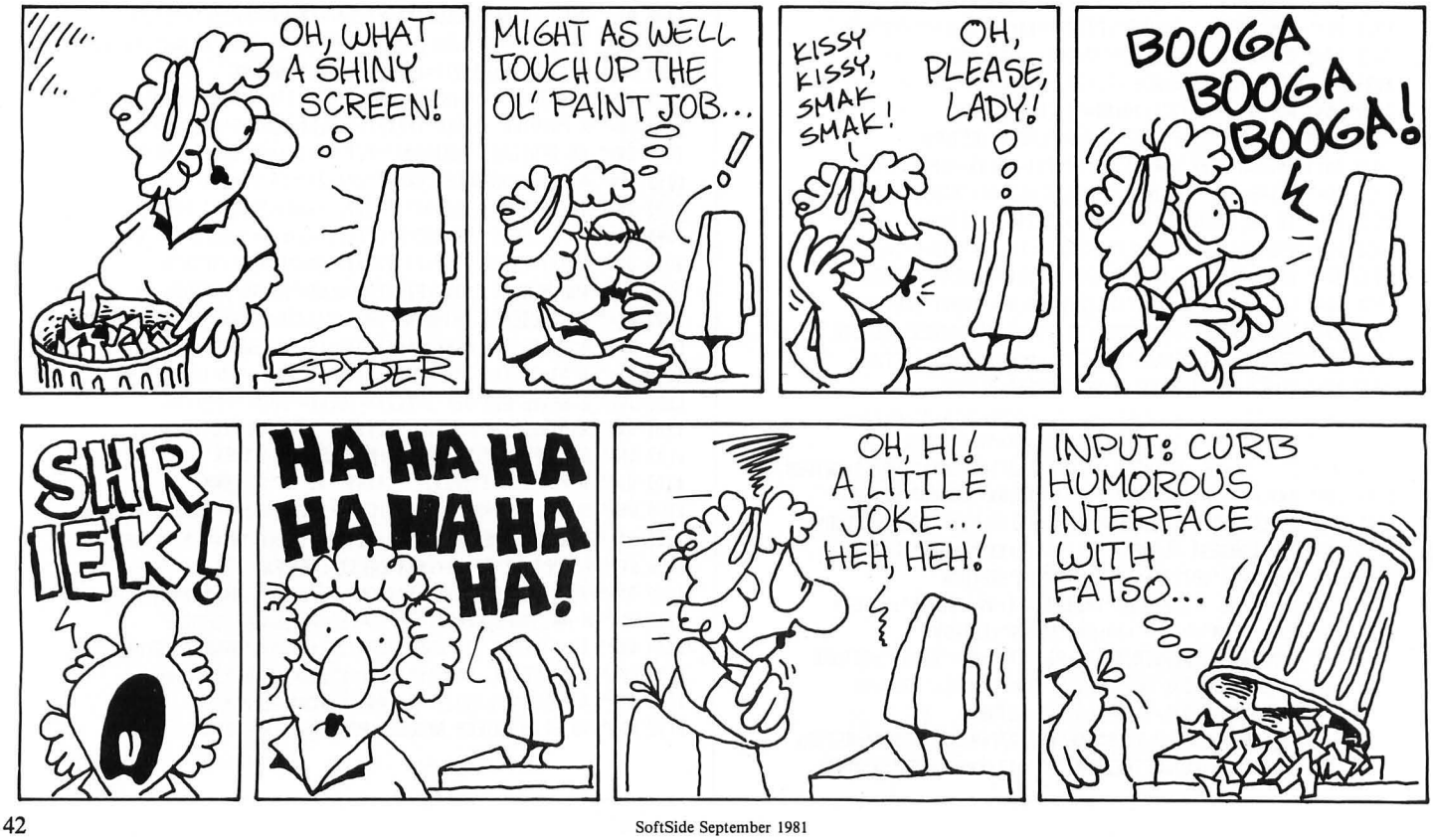
```
1113 D$=" MOUNTAIN    SPINACH":PZ(27)=12:N$="ORACH":RETURN
1114 D$=" ANY SHIELD":PZ(27)=19:N$="AEGIS":RETURN
1115 D$=" USED AS A   FUEL OIL":PZ(27)=14:N$="MAZUT":RETURN
1116 D$=" FERMENTED   MILK":PZ(27)=13:N$="KEFIR":RETURN
1117 D$=" AN EARRED   SEAL":PZ(27)=9:N$="OTARY":RETURN
1118 D$=" AN ANCIENT  JEWEL CASE":PZ(27)=20:N$="PYXIS":RETURN
1119 D$=" A CHARLATAN":PZ(27)=12:N$="QUACK":RETURN
1120 D$=" A BIN FOR   SALTING FISH":PZ(27)=11:N$="KENCH":RETURN
1121 D$="OLD-FASHIONED":PZ(27)=13:N$="FUSTY":RETURN
1122 D$=" ANY WAR FOR A FAITH":PZ(27)=15:N$="JIHAD":RETURN
1123 D$=" ANY FORM OF DENTINE":PZ(27)=10:N$="IVORY":RETURN
1124 D$=" A MILKY      EXUDATION":PZ(27)=16:N$="OPIUM":RETURN
1125 D$=" RELATING TO BLOOD":PZ(27)=9:N$="HEMIC":RETURN
1126 D$=" OCCURING BY LUCKY CHANCE":PZ(27)=11:N$="FLUKY":RETURN
1127 D$=" A RIDING     WHIP":PZ(27)=14:N$="QUIRT":RETURN
1128 D$=" MONEY":PZ(27)=9:N$="LUCRE":RETURN
1129 D$=" A HORN":PZ(27)=14:N$="CORNU":RETURN
```

Calculated-GOSUB routine.

```
2000 IFPUZ<44THENONPUZGOTO1000,1001,1002,1003,1004,1005,1006,100
7,1008,1009,1010,1011,1012,1013,1014,1015,1016,1017,1018,1019,10
20,1021,1022,1023,1024,1025,1026,1027,1028,1029,1030,1031,1032,1
033,1034,1035,1036,1037,1038,1039,1040,1041,1042
2010 IFPUZ<87THENONPUZ-43GOTO1043,1044,1045,1046,1047,1048,1049,
1050,1051,1052,1053,1054,1055,1056,1057,1058,1059,1060,1061,1062
,1063,1064,1065,1066,1067,1068,1069,1070,1071,1072,1073,1074,107
5,1076,1077,1078,1079,1080,1081,1082,1083,1084,1085
2020 ONPUZ-86GOTO1086,1087,1088,1089,1090,1091,1092,1093,1094,10
95,1096,1097,1098,1099,1100,1101,1102,1103,1104,1105,1106,1107,1
108,1109,1110,1111,1112,1113,1114,1115,1116,1117,1118,1119,1120,
1121,1122,1123,1124,1125,1126,1127,1128,1129
```

MACHINE HEAD

BY SPYDER



NIKROM TECHNICAL PRODUCTS PRESENTS A DIAGNOSTIC PACKAGE FOR THE APPLE II AND APPLE II + COMPUTER.

"THE BRAIN SURGEON"

Apple Computer Co. has provided you with the best equipment available to date. The Diagnostic's Package was designed to check every major area of your computer, detect errors, and report any malfunctions. ***The Brain Surgeon*** will put your system through exhaustive, thorough procedures, testing and reporting all findings.

The Tests Include:

- MOTHERBOARD ROM TEST
- APPLESOFT ROM CARD TEST
- INTEGER ROM CARD TEST
- MOTHERBOARD RAM TESTS
- DISK DRIVE SPEED CALIBRATION
- DISK DRIVE MAINTENANCE
- DC HAYES MICROMODEM II TEST (HARDWARE & EPROM)
- MONITOR & MODULATOR ROUTINES
- MONITOR SKEWING TESTS
- MONITOR TEST PATTERN
- MONITOR TEXT PAGE TEST
- MONITOR & TV YOKE ALIGNMENT
- LO-RES COLOR TESTS
- HI-RES COLOR TESTS
- RANDOM HI-RES GENERATOR
- SPEAKER FUNCTION TESTS
- SQUARE WAVE MODULATION
- PADDLE & SPEAKER TEST
- PADDLE & BUTTON TEST
- PADDLE STABILITY
- INTERNAL MAINTENANCE
- GENERAL MAINTENANCE
- ON BOARD "HELP"

NEW!

The Brain Surgeon allows you to be confident of your system. This is as critical as the operating system itself. You *must* depend on your computer 100% of it's running time. ***The Brain Surgeon*** will monitor and help maintain absolute peak performance.

Supplied on diskette with complete documentation and maintenance guide.
(#47-254001D)
PRICE: $49.95
REQUIRES: 48K, FP in ROM
1 Disk Drive, DOS 3.2 or 3.3

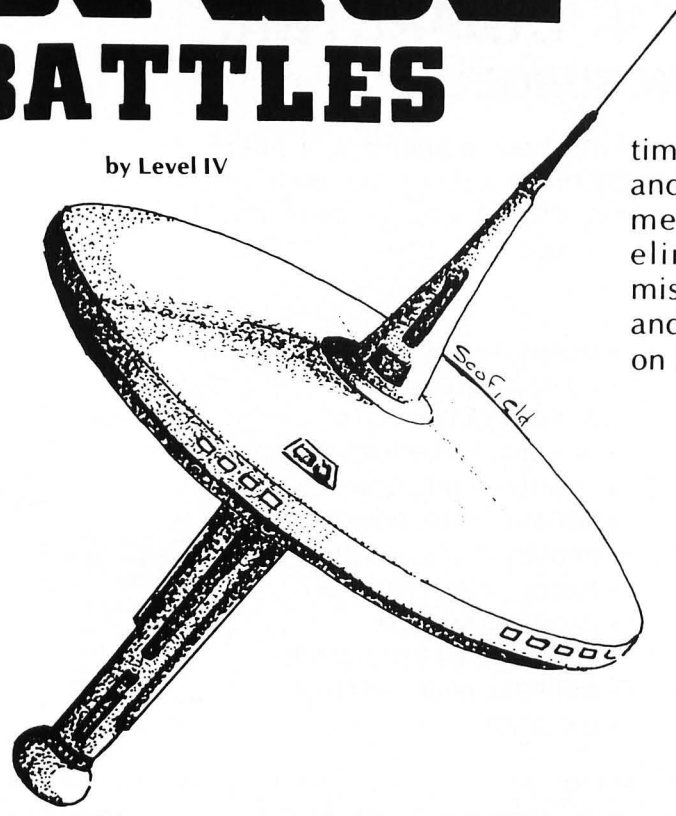
MasterCard VISA

TSE·HARDSIDE
6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

APPLE is Registered Trademark of Apple Computer Co.
MICROMODEM II is Registered Trademark of D.C. HAYES

SPACE BATTLES

by Level IV



Space Battles is one of the best space games we've seen in a long time. Features three levels of play, fast, machine language graphics, real-time input, and "smart" enemy ships that move and shoot! You'll find that playing the part of a mercenary isn't simple. It's not enough to eliminate the aliens; you must turn a profit, missiles are expensive, the rewards are small, and watch out for the radiation belts!!! Available on Level II, 16K Tape or 32K Disk.

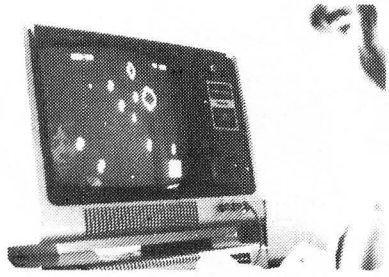
Tape — $14.95
Disk — $19.95

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585

*Games from **BIG FIVE** will turn your computer into a*

TRS-80™ HOME ARCADE

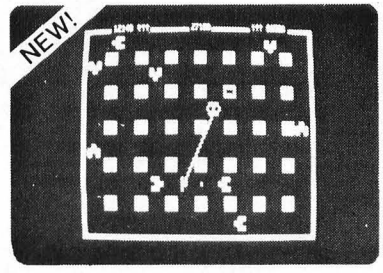
SUPER NOVA©



"Huge ASTEROIDS have invaded the galaxy! Your mission is to destroy them and the alien saucers before they destroy you!" *Our #1 top selling game!*

$15.95 16K Mod I or III

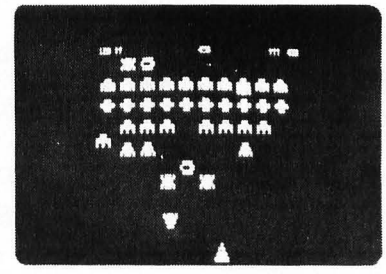
ATTACK FORCE©



"Eight alien ramships are warping down toward your destroyer ship. You must shoot them down quickly before they crush you!" *With sound!*

$15.95 16K Mod I or III

GALAXY INVASION©



"The newest and most exciting Invaders-type game yet! Smooth sound effects, sharp graphics, and the 'Flagship' alien from **Super Nova** combine to make this our finest TRS-80 game!"

$15.95 16K Mod I or III

TSE:HARDSIDE

6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

NEWBASIC

Editor's Note: With NEWBASIC SoftSide initiates a policy of publishing documentation for DV (Disk Version). The program is of such a nature that it would be impractical to publish a printed line listing in the magazine.

This is the documentation for "NEWBASIC", a program which is only available on the S-80 Disk Version of SoftSide.

"NEWBASIC" is a set of enhancements for Disk BASIC with added utilities for the S-80 Model I with 32K or more RAM with one or more disk drives running under TRSDOS 2.3, NEWDOS 2.1, or NEWDOS 80.

NEWBASIC CREATOR

This program produces a NEWBASIC program on disk, composed of user-selected commands. Once a NEWBASIC program has been created, it can be executed as described in EXECUTING NEWBASIC. You need run CREATOR only when you desire a different version of NEWBASIC.

When you execute CREATOR, you will be asked what commands you wish to include in the NEWBASIC program you are creating. You will be shown every keyword in alphabetical order. A few keywords, such as DO (UNTIL) and PON (POFF), are shown in pairs (the second keyword will be displayed in parentheses). After deciding what keywords to include, the NEWBASIC program you have just created will automatically be DUMPed to disk. You are now ready to execute NEWBASIC.

To execute the NEWBASIC CREATOR, type "CREATOR" when you see "DOS READY". (CREATOR occupies memory from 7900H to BFFFH, and therefore will overwrite any custom driver routines, etc., that occupy this region.) The screen will clear, and the following will appear:

MODULAR SOFTWARE ASSOCIATES - NEWBASIC CREATOR - VERSION 1.0

KEYWORD	DESCRIPTION	SIZE	USE?
BLINK	SWITCH BLINKING CURSOR ON/OFF	161	?

Notice the information that is displayed for BLINK — this is the same type of data that will appear for every NEWBASIC keyword. First, there is the keyword BLINK, followed by a short description of what BLINK does. Next is the number of bytes (in decimal) that would be added to NEWBASIC if you decide to include BLINK. The size for a particular keyword can vary greatly from one use of CREATOR to another, depending on what other keywords have (or have not) been included at the point when the size is displayed. As an example, the size of EXIT is 6 bytes if DRUN is included, but 103 if it is not included. The size is displayed to give an approximate idea of the size of code associated with a keyword. It is not intended to be an aid in achieving optimal memory usage, since the figure shown can be misleading.

You must now decide whether or not to include BLINK in the NEWBASIC program you are creating. If you wish to include it, you must press the "Y" key; to exclude it press the "N" key. The same type of information for the next keyword will now be shown, and you must decide whether or not to use it. When you have answered "Y" or "N" for every NEWBASIC keyword, the following will appear:

TOTAL SIZE OF COMMANDS INCLUDED: XXXX

FILESPEC TO SAVE TO (PRESS ENTER FOR "NEWBASIC/CMD" DEFAULT)?

The total size given is the total byte count which will be taken up by the NEWBASIC program being created ONCE IT IS EXECUTED. All NEWBASIC programs take up the same amount of space on disk, regardless of their size when they are run. Hence, a NEWBASIC program that has no NEWBASIC commands included will be the same size on disk as one in which every NEWBASIC command has been included.

The NEWBASIC program just created will now be saved to disk. If you press ENTER in response to the "FILESPEC ..." question, it will be saved under the name NEWBASIC/CMD. If you desire to have it saved under a different name, you may type in a standard DOS filespec and press ENTER. (The filespec may not include a password, however.) An extension of /CIM will automatically be appended if your filespec does not include an extension. A delay will occur in which the relocatable NEWBASIC program is created. An asterisk in the upper right-hand corner will blink during the delay to assure you that CREATOR is still functioning properly. After the delay, the disk drive should turn on and the NEWBASIC program will be DUMPed to disk.

IMPORTANT NOTE: The NEWBASIC program to be DUMPed will take up 8 grans on the disk. If you are using TRSDOS or NEWDOS 2.1, the drive motor will remain on if there is not enough room on the disk for the file to be DUMPed. If the DUMP takes longer than 30 seconds, then this has occurred. You must reset the computer and run CREATOR again, after insuring enough file space exists.

EXECUTING NEWBASIC

Once you have run the NEWBASIC CREATOR program and created a NEWBASIC program on disk, it is very simple to execute NEWBASIC and enter your NEW, enhanced, BASIC. The normal "BASIC/CMD" file must exist for the created NEWBASIC program to execute. You execute NEWBASIC just as you would execute whatever BASIC you are using, except that you substitute "NEWBASIC" (if the created NEWBASIC program is called "NEWBASIC/CMD") for "BASIC". If you are using NEWDOS BASIC, you may type the number of files desired, memory size, etc., all on the same line as you normally would to enter BASIC. If you are using TRSDOS, then you should only type "NEWBASIC" and press ENTER. Answer the "HOW MANY FILES?" and "MEMORY SIZE?" questions as usual. For both TRSDOS and NEWDOS there will be a short delay to allow you to read the copyright notices of the respective BASICs. The screen will then clear, MODULAR SOFTWARE ASSOCIATES' copyright notice will appear, and below that a number will be displayed ("USER MEMORY START ADDRESS = XXXXX"). This number gives the start address in decimal of your BASIC program text area. You are now ready to use your NEW enhanced BASIC.

IMPORTANT NOTE: DEBUG must be off before executing NEWBASIC, or the computer will "hang". Once you have entered NEWBASIC, you may reenable DEBUG by CMD"D" if desired.

EXAMPLES:

NEWBASIC will execute NEWBASIC under all compatible DOS's.

NEWBASIC 15,50000 will execute NEWBASIC and enter BASIC with 15 files and memory size set to 50000. (NEWDOS only!)

MISCELLANEOUS NOTES

NEWBASIC occupies RAM below normal BASIC. This moves BASIC programs higher in RAM than usual. Some programs which use PEEK and POKE do not allow for this fact, and may not run properly under NEWBASIC.

All NEWBASIC commands must be at the start of a BASIC statement. This means that NEWBASIC commands included in IF/THEN/ELSE statements must be preceeded by a colon to indicate the start of a new statement.

In the syntax descriptions that follow, optional arguments are surrounded by parentheses "( )".

BLINK

The BLINK command toggles the blinking block cursor on/off. The first time the command is executed, the blinking cursor will appear. The next time the command is executed, the blinking cursor will be replaced with the standard underline character. If BLINK is used again, after the underline cursor has been

continued on next page

continued from previous page

restored, the blinking block cursor will reappear (and so on...).

EXAMPLES:

10 BLINK
if this is the first time that BLINK is used, the blinking block cursor will be turned on.

```
10 BLINK
20 INPUT"NAME",A$
30 BLINK
40 PRINT "HELLO, ";A$
```

if the underline cursor is being used prior to running this program, the following will occur: The question "NAME?" will be displayed with a blinking block cursor after it. If the reply to the input statement was "JOHN", the line "HELLO, JOHN" will appear. Note the blinking cursor will have been replaced with the standard cursor.

CALL hex-constant
CALL int-expression
CALL lsb,msb expression
CALL pos-expression

This command allows you to easily call a machine language subroutine. No DEFUSR or USR(X) statement is needed. The subroutine must be in memory at the time of the CALL. You can not directly pass an argument between the BASIC program and the machine language subroutine being called. If you wish to pass one or more arguments, this can be done by POKEing and PEEKing into specified memory locations. The subroutine being called should return to BASIC by the Z-80 RET statement. The CALL statement saves all registers used by BASIC.

EXAMPLES:

CALL &HE000
will call a subroutine located at hex E000.

20 CALL +50000
will call a subroutine located at decimal 50000.

30 CALL X + PEEK(35000)
will call a subroutine located at the sum of the current value of X and the contents of memory location 35000.

CALL 0,200
will call a subroutine at 0,200 in lsb,msb (decimal) notation.

200 CALL -100
will call a subroutine located at -100 (65436 decimal).

10 X=-100 : CALL X
will call a subroutine located at -100 (65436 decimal).

1000 X=65436 : CALL X
The intent of this program line is the same as in the preceeding examples, but will result in an OVERFLOW error. This is because "X" will be interpreted as an "int-expression". To force "pos-expression" interpretation, the correct usage is CALL +X.

CONV hex-constant
CONV int-expression
CONV lsb,msb expression
CONV pos-expression
CONV "A"
CONV "AA"

This command is used to display different representations of the same number. CONV "A" means convert the single alphanumeric character A. CONV "AA" means convert the two-character string AA, where AA can be any two alphanumeric characters (not necessarily the same).
The following illustrates the display format of CONV:

```
CONV +50000                              (user entry)
&HC350   "P"   -15536   +50000   80,195   (computer reply)
```

The first number displayed is the hexadecimal (base 16) representation. Next follows the two-character alphanumeric or graphic representation. The first character displayed corresponds to the most significant byte (msb) of the hexadecimal representation, while the second character corresponds to the least significant byte (lsb). The byte will be displayed as a blank if it is less than 0021H (33 decimal), or greater than 00BFH (191 decimal). An exception is that a zero byte will be considered as an ASCII NUL (""), and will not be displayed. The third representation given is the decimal integer form of the number. This is the number you would use to PEEK or POKE an address above 32767. The fourth number displayed is the positive decimal representation. The fifth (last) representation is the number in decimal lsb,msb format. This number is very useful when PEEKing or POKEing addresses in decimal.

EXAMPLES:

CONV "bD"
will display the representations of the letters "bD".

CONV &HBD
will display the representations of 00BDH.

10 X=100 : Y=0 : CONV X,Y
will display the representations of the lsb,msb number 100,0.

DESPOOL "filespec" (,S)

The DESPOOL command without the trailing ",S" will despool the specified file to a parallel printer. The command with the trailing ",S" will despool the specified file to the RS-232-C serial port. The spooler is interrupt driven; normal processing can continue while a file is being printed. An occasional short pause will occur as the disk file is being accessed for more information. This command allows the operator to compose, edit, or run BASIC programs which may include disk I/O, while a previously spooled file is being printed. The despooling is automatically terminated upon reaching the end-of-file. To abort the despooling operation, hold down the shift key, the down arrow key, and the "C" key simultaneously (this outputs a 'control C').
Only ASCII encoded files may be despooled. BASIC programs saved without the ",A" option may not be despooled. An attempt to despool two files at the same time will result in an ILLEGAL FUNCTION CALL error. CMD"T", which halts interrupts, will cause despooling to pause. CMD"R", which enables interrupts, will cause the despooling to continue.

EXAMPLES:

10 DESPOOL"TEXT/FIL"
will start the despooling of the file "TEXT/FIL" to the parallel port printer.

20 DESPOOL"CODEFILE/TXT.SECRET:1",S
will, upon execution, start despooling the file "CODEFILE/TXT" on drive 1 with the password "SECRET" to the serial port.

DIR (:) (drive no.)

The DIR command allows the display of a diskette's directory. As in DOS, if no drive number is specified, the directory of drive 0 will be displayed. If a drive number is specified, it can be immediately after the command (DIR1) or seperated from the DIR by a space (DIR 2). The "normal" colon may also be used (DIR :1). The drive number must be used in a BASIC program.
The display will be formatted with diskette data on the top line of the display, and file data on the lines below. If there are more files than may be displayed at one time, as many as possible will be displayed and pressing any key will display the remaining files. The diskette data will contain the drive number being read, the date the diskette was formatted, and the number of free sectors (grans * 5) remaining on the diskette. The data displayed for a file includes the filename, including extension if any, followed by the size of the file in sectors. System files will not be displayed.
The DIR command will not execute in NEWDOS/80 (nor is it needed). If the command is used, a MISSING NEWBASIC COMMAND error will result.

EXAMPLES:

DIR
when this command is given, the directory of drive 0 will be displayed.

DIR 1
will result in the display of drive 1's directory.

10 DIR :2 : PRINT"DONE"
will, when run, display the directory of drive 2, if it exists, and then display the line "DONE". If a DIR of a drive number that does not exist is executed the drive motors will remain on and the system will "hang".

DLOAD "filespec"

The DLOAD command is used to load object-code files in RAM and returns control to BASIC. Command (/CMD) files may be loaded and, providing the entry point is known, executed from within BASIC.

EXAMPLES:

10 DLOAD"PRINTER/CMD"
will load the command file "PRINTER/CMD" into RAM.

```
10 DLOAD "SORT/CIM"
20 CALL &HF000
30 PRINT"SORT DONE"
```
will load the object-code file "SORT/CIM" and execute it (if the start address is hex F000). The line "SORT DONE" will then be displayed.

DO (statement) (statement) ... (statement)
UNTIL (true/false expression)

The DO and UNTIL commands form an iterative (repetitive) loop. The DO command denotes the start of the loop, while the UNTIL command terminates the loop. The body of the loop consists of 0 or more statements between DO and UNTIL. These statements will be executed repeatedly until the (true/false expression) argument following UNTIL is true. (The parentheses surrounding the "true/false expression" after UNTIL are mandatory.) Nesting of DO/UNTIL loops past a level of 10 will cause an OVERFLOW error.

If an UNTIL is executed without previously executing a matching DO, an UNTIL WITHOUT DO error will occur. Mismatched DOs and UNTILs can result from transferring (GOTO) out of the DO/UNTIL loop body. To immediately terminate a DO/UNTIL loop, use the following procedure — NEVER transfer (GOTO) out of the loop.

```
10 I=0
20 DO
30   GOSUB 1000
.
.
100  IF T=2 THEN I=10 : GOTO 200
.
.
190  I=I+1
200 UNTIL (I=10)
```

In line 100, the loop terminating condition (I=10) is set to true. Control is then transferred to the line containing UNTIL, where the loop will be properly terminated.

A DO/UNTIL loop should not be imbedded within IF/THEN/ELSE statements, unless the entire loop is contained within the same program line as the IF. The following two examples illustrate this restriction. The first example shows an incorrect use of a DO/UNTIL within an IF/THEN/ELSE. The second example shows how the first example can be written correctly.

```
10 IF I=1 THEN : DO
20   PRINT"I=1"
30   INPUT I
40 UNTIL (I     1)
```

The fault in the above program is that line 20 will be executed even if I does NOT equal 1. This is due to the fact that the "scope" of the IF statement only applies to the line in which it appears.

```
10 IF I=1 THEN : DO : PRINT "I=1" : INPUT I : UNTIL (I<>1)
```

This program line will execute the DO/UNTIL loop only if I=1. There are, of course, several other methods of coping with the IF/THEN/ELSE restriction, two of which are shown below.

```
10 IF I=1 THEN : NGOSUB #"DO I LOOP"
.
.
999 END
1000 #"DO I LOOP"
1010 DO
1020   PRINT "I=1"
1030   INPUT I
1040 UNTIL (I<>1)
1050 RETURN
```

```
10 IF I<>1 THEN : NGOTO #"DONE LOOP"
20 DO
30   PRINT "I=1"
40   INPUT I
50 UNTIL (I< >1)
60 #"DONE LOOP" : REM REST OF PROGRAM FOLLOWS
```

The DO/UNTIL loop construct executes slower than a FOR/NEXT loop, since it is implemented as an add-on to an existing interpreter (LEVEL II BASIC). However, it can provide structure and clarity to a BASIC program, which are *often more* desirable than speed of execution.

EXAMPLES:

```
10 A$="" : DO : INPUT"WHAT IS YOUR NAME";A$ : UNTIL (A$<>"")
```
will continue to INPUT A$ until something other than ENTER is INPUT.

```
10 TRUE=0
20 DO : CLS
30   GOSUB 200
.
.
100 UNTIL (TRUE OR A=B)
```
will execute the DO/UNTIL loop until the variable TR equals -1 or until A=B.

```
10 DO
20   NGOSUB #"INIT GAME"
30   DO : NGOSUB #"PLAY ROUND" : UNTIL (GAMEOVER)
40   NGOSUB #"WINNER"
50 UNTIL (0)
```
is an example of nested DO/UNTIL loops. The DO/UNTIL loop in line 30 is nested within the loop of lines 10-50. This is a nesting level of 2. (Notice that if the subroutine #"PLAY ROUND" has a DO/UNTIL loop, it too would be nested within the loop of lines 10-50.) The DO/UNTIL loop in line 30 will terminate when the variable GA equals -1. The loop of lines 10-50 will "never" terminate, since 0 can never be true (that is, can never equal -1).

DRUN "filespec"

The DRUN command exits NEWBASIC and executes the requested object file. This is a final exit. A command file may not be run with a return to NEWBASIC. If an error occurs during the execution of DRUN a return to DOS READY will result. The complete file name must be used including extension as no defaults are implemented.

EXAMPLES:

```
DRUN"DIRCHECK/CMD"
```
will exit NEWBASIC, run the program "DIRCHECK/CMD" and exit to DOS.

```
10 DRUN "BASIC/CMD"
```
will exit NEWBASIC and load and execute "BASIC/CMD". This will allow the user to enter "normal" BASIC without exiting to DOS.

EXIT

This command exits NEWBASIC and returns to DOS READY. The EXIT and DRUN commands are the only safe way to exit NEWBASIC other than a "reset".

IMPORTANT: Once NEWBASIC has been exited, there is no way to re-enter NEWBASIC to recover a program. Neither "BASIC *" nor "NEWBASIC *" will succeed in doing so.

EXAMPLES:

```
EXIT
```
will exit NEWBASIC and return to DOS.

```
500 IF A$="STOP" THEN : EXIT
```
will exit NEWBASIC if the variable A$ equals "STOP".

FILL X1,Y1,X2,Y2 (,R)

The FILL command without the trailing ",R" turns on all the graphics blocks of the rectangle determined by the diagonal line drawn from the point X1,Y1 to the point X2,Y2. The FILL command with the trailing ",R" resets (i.e. erases) the graphics blocks of the determined rectangle. The X-coordinates are numbered from left to right, 0 to 127. The Y-coordinates are numbered from the top to the bottom, 0 to 47.

The arguments X1,Y1,X2,Y2 may be numeric constants, variables, or expressions. FILL uses the integer portion, hence, the arguments need not be integers. The command is valid for Xs between 0 and 127 inclusive and Ys between 0 and 47 inclusive. Arguments outside this range will produce an "ILLEGAL FUNCTION CALL" error message.

NOTE: Parentheses before and after the arguments field (unlike SET and RESET) are not needed and if present will cause an error.

EXAMPLES:

```
10 FILL 0,0,127,47
```
will turn all graphics blocks on (i.e. white the screen).

```
20 FILL 127,47,0,0,R
```
will "clear" the screen as all graphics blocks will be reset.

```
10 FILL X+RND(5),IY+2*Y1,4,Y2*Y2
```
will draw and fill the rectangle determined by the evaluated arguments as long as the argument values obtained are within the limits specified in the text.

```
20 FILL X*(3+Z),Y1+.5,X2,47-Y2,R
```

continued on next page

continued from previous page

will reset the rectangle determined by the evaluated arguments with the same cautions as in the preceding example.

HIMEM hex-constant
HIMEM int-expression
HIMEM lsb,msb expression
HIMEM pos-expression

HIMEM allows you to protect high memory so that it will not be used by BASIC. The argument specifies the address of the last byte usable by BASIC. An ILLEGAL FUNCTION CALL error will result if this address is not at least 200 bytes above the start of user memory. HIMEM resets the values of all variables and does a CLEAR 50; caution should therefore be employed when using HIMEM within a BASIC program.

EXAMPLES:

HIMEM &HFFFF
will allow BASIC to use all of memory (48K machine). This will result in an ILLEGAL FUNCTION CALL error if used in a 32K machine.

10 HIMEM +40000
will protect memory above 40000 (decimal).

10 INPUT X : HIMEM +X
will protect memory above the input value of X. Note the use of the positive (''+'') sign before X. This is to allow for values greater than 32767. A different way to limit the values of the argument is:

10 INPUT X% : HIMEM X%
will allow X to range from -32768 to +32767.

#''label''

The pound sign (''#'') preceding a quoted-string serves to identify the string as a label. If a line is to be labeled, the pound sign must be the first non-blank character in the line after the line number. The pound sign must be followed by a double quote, a string of 0 to 251 alpha-numeric characters (except a double quote), and the terminating double quote. If the line is not null, there must be a colon seperating the label and the BASIC statement which follows. Labels should be unique, although no check is performed to insure this. If two lines are labeled with the identical label, a reference to the label will always refer to the first (lowest numbered) line. A line should only be labeled once — any subsequent labels will be ignored.

EXAMPLES:

100 #''DELAY'' : DO : T$=INKEY$: UNTIL (T$<>'''') : RETRUN
labels line 100 as #''DELAY''. It may now be referenced by an NGOSUB #''DELAY'' statement.

10 #''START''
.
.
5000 NGOTO #''START''
labels line 10 #''START''. A sample reference is shown in line 5000.

LCASE

The LCASE command toggles the lower-case display enable. The LCASE command supports both the Radio Shack and Electric Pencil type lower-case hardware modifications. The first time the command is used, lower-case letters may be entered and displayed by using the shift key and the appropriate letter. The next time the LCASE command is used, lower-case entries are inhibited.

Lower-case characters can be used anywhere in composing a BASIC program but can only be displayed by PRINT statements or in REM lines.

EXAMPLES:

LCASE
10 PRINT''This is a test.''
20 LCASE
if LCASE has not been used prior to the LCASE command executed in command mode, lower case will be enabled. Line 10 may be input by using the shift-key entry for all lower-case characters. Line 20 will inhibit any further lower-case entry or display. When the program is RUN, the line, ''This is a test.'', will be displayed.

10 LCASE
20 STOP
will result, if lower case is not enabled before the program is run, in the display of the line, ''Break in 20''. Note that the message is displayed in both upper- and lower-case letters.

LINE X1,Y1,X2,Y2 (,R)

The LINE command without the trailing '',R'' turns on the graphics blocks of the line determined and terminated by the points X1,Y1 and X2,Y2. The LINE command with the trailing '',R'' resets (i.e. erases) the graphics blocks of the determined line. This command has the same limits and parameters as the FILL command.

NOTE: Parenthesis before and after the arguments field (unlike SET and RESET) will cause an error.

EXAMPLES:

10 LINE 0,0,127,47
will draw a line from the top left of the video display to the lower right.

20 LINE 127,47,0,0,R
will reset (i.e. erase) the line set in the preceding example.

10 LINE 128-1,0,2*0,(4*10)+7
will draw a line from the top right of the video screen to the lower left.

100 LINE (B*B)-(A*A),C*SIN(Y),X2-X1,Y2,R
will reset the line determined by the evaluation of the command's arguments.

LOC. (''string'')
LOC. (string)

This command allows you to locate instances of a string within the BASIC program currently in memory. The LOC. command with an argument locates the first occurrence of the string within the program. LOC. without an argument locates the next occurrence of the last string searched for. The maximum argument string length is 20 characters.

The first argument form of LOC. (''string'') should be used if the string to be searched for occurs in a quoted string (PRINT ''string'' or INPUT ''string'', for example) or a REMark. Double quotes may NOT be imbedded within the argument string since they serve as delimiters of the string. The second argument form of LOC. (string) should be used to locate a string that is not within a quoted string or REMark. The two forms of LOC. allow you to distinguish between a BASIC keyword and the corresponding character string, as illustrated in the examples below.

The LOC. command maintains its own current line pointer, but will adjust BASIC's LIST and EDIT line pointer as described below. To start a search for a string, use LOC. with an argument. LOC. will start its search from the first program line. If the string is located, the line containing the string will be LISTed on the screen. BASIC's LIST and EDIT pointer will now be changed to point to the LISTed line. LOC.'s own pointer will also have been changed so that the next LOC. (without an argument) will search beginning with the next line after the line just LISTed. (This means that two occurrences of the same string within the same line will be located only once.) Once the line has been LISTed, you have several choices:

1). Press BREAK to end LOC. and return to Command mode or the executing BASIC program (should LOC. have been executed within a program);

2). Press the ''E'' key to enter the EDIT mode in order to EDIT the line just LISTed. After ending EDIT, you will return to the Command mode as usual;

3). Press any other key to do an automatic LOC. (LOC. without an argument). This will find the next occurrence of the string just located.

If a search for a string fails, no line will be LISTed to the screen, and BASIC's LIST and EDIT pointer will not be altered.

EXAMPLES:

The examples below assume the following program is in memory:

```
10 NGOSUB #''NAME?''
20 PRINT : PRINT ''THANK YOU'';
30 PRINT ''VERY MUCH'';NM$
40 GOSUB 100
50 END
60 #''NAME?'' : INPUT''WHAT IS YOUR NAME'';NM$ : RETURN
```

```
LOC.? - (Locate the first occurrence of the BASIC token PRINT)
Line 20 is LISTed to the screen.
ENTER - (Locate the next occurrence of PRINT)
Line 30 is LISTed to the screen.
BREAK - (Exit LOC.)
Command mode entered.
LOC.''?'' - (Locate the first use of a question mark)
Line 10 is LISTed.
SPACE BAR - (Locate the next occurrence of ''?'')
Line 60 is LISTed.
E key pressed - (EDIT line 60)
EDIT mode is entered.
 .    .
 .    .
EDIT mode is exited
```

LOC. - (Locate the next occurrence of "?")
No line is LISTed and Command mode is entered
LOC.GOSUB - (Locate the first use of the GOSUB token)
Line 10 is LISTed
BREAK - (Exit LOC.)
Command mode is entered
LOC.NM$ - (Locate the first use of the variable NM$)
Line 30 is LISTed
BREAK - (Exit LOC.)
Command mode is entered

NGOSUB line number
NGOSUB #"label"

This command allows for greater flexibility in calling a BASIC subroutine. The argument "line number" may be a constant, variable, or expression. It denotes the BASIC program line number to GOSUB. If the second argument form of NGOSUB is used '#"label" ' indicates that the line number to GOSUB is labeled with #"label". Other than providing additional methods of referring to the subroutine being called, NGOSUB performs identically to GOSUB. However, NGOSUB may not be substituted for GOSUB in an ON n GOSUB command.

EXAMPLES:

10 NGOSUB 40000 : NGOSUB 100
20 NGOSUB SN*100
will cause calls to subroutines located at line numbers 40000, 100, and the product of the variable SN and the number 100.

10 NGOSUB #"INIT"
20 NGOSUB #"PROCESS"
.
.
1121 #"INIT"
.
.
1300 RETURN
.
.
1380 #"PROCESS"
.
.
1420 RETURN
will cause the subroutines located at lines 1121 & 1380 to be called.

NGOTO line number
NGOTO #"label"

This command provides additional methods of specifying the line to GOTO. The argument "line number" may be a constant, variable, or expression. It denotes the BASIC program line number to GOTO. The argument '#"label" ' refers to the line which begins with the matching #"label". Other than allowing greater flexibility in referring to the line number to GOTO, NGOTO performs identically to GOTO. However, NGOTO may not be used in place of GOTO within an ON n GOTO or ON ERROR GOTO command.

EXAMPLES:

10 NGOTO 10
will cause an "infinite" loop upon execution.

10 #"HERE" : NGOTO #"HERE"
will cause an "infinite" loop similar to the previous example.

10 NGOTO X
will cause execution to transfer to the line specified by the current value of the variable X, assuming such a line exists.

NTROFF
NTRON (expression) (,expression) ... (,expression)
NTRON line range (,expression) (,expression) ... (,expression)

The NTRON command enables a trace facility for debugging BASIC programs. It allows you to specify what lines of your BASIC program are to be traced, as well as providing a unique expression-trace capability. A BASIC program must be in memory at the time the NTRON command is executed, or an ILLEGAL FUNCTION CALL error will occur. The NTROFF command disables this trace facility.

The "line range" argument determines what lines of the BASIC program currently in memory will be traced. (See below for a detailed description of "line range".) If missing, all lines of the BASIC program will be traced. It is important to remember that the line range entered remains fixed, even if lines are added to the BASIC program (until you execute another NTRON). For example, suppose the BASIC program in memory, at the time an "NTRON" (no line range) is executed, has lines ranging from 10 to 50. If you later add lines between lines 10-50, these WILL be traced without doing another NTRON. However, if you add lines outside this range (before line 10, or after line 50), these lines will NOT be traced unless you execute another NTRON. The above example would still apply even if the command entered had been "NTRON (0:65529)".

The remaining arguments to NTRON constitute the expressions (if any) whose values are to be traced. There may be 0 or more of these expressions. Too many expressions (the limit depends on the sum of the length of the expressions) will result in an OUT OF STRING SPACE error. (You canNOT increase this limit by CLEARing more string space.) If two or more expressions are given, they must be seperated by commas. (Also note that if the first argument (line range) is given, a comma must seperate it from an expression.) The expressions used may be ANY valid BASIC expression which could properly be put in a single PRINT statement. However, NTRON does virtually no syntax checking on these expression arguments. If you enter an invalid expression, NTRON will more than likely accept it. An error will not occur until the program is run and the first line is about to be traced! At this point you will receive a syntax (or another) error, in what might appear to be a correct line. If you execute an NTROFF and the program line subsequently runs without an error, the cause of the error was probably an improper NTRON expression argument.

NTRON remains active with the arguments given (if any) until an NTROFF is executed. BEFORE a BASIC program line being traced is about to be executed, NTRON's output will be displayed on the video screen. This consists of zero or more lines (depending on the number of arguments given with the NTRON command) of "expression=" expression value, followed by a LIST of the line to be executed. It is important to remember that the values of the expressions (if any) are those that exist BEFORE the line is executed. (This is particularly important when the expression contains a division operation. If the divisor has not yet been assigned a value, a DIVISION BY ZERO error will occur.)

NTRON can be of great value in determining BASIC program bugs. Used within a BASIC program, it can provide for the tracing of several disjointed line ranges. This can be accomplished by simply having numerous NTRON commands with different line ranges throughout the program.

Although most output formatting commands (TAB, ;, USING, etc.) can be included within "expression", this is not recommended. In particular, it is very difficult to use USING properly. Any "expression" which appears after USING will be formatted with USING (or cause an error if this cannot be done). (Remember that the expression arguments must be expressions which may be entered into a SINGLE PRINT statement.) Also, the USING variable must be defined before the first line to be traced is executed, or an error will result. It is best to use expressions which do not alter the output format of NTRON; however, if formatting commands ARE used, be prepared for anomalous results.

EXAMPLES:

NTRON
will cause a trace of all BASIC program lines currently in memory. No expressions will be traced.

NTRON(500:1000) ,X,Y
will cause a trace of all BASIC lines between 500 and 1000 (inclusive). The current values of the variables X and Y will be displayed preceeding the line trace (LIST).

NTRON INT(X)+A*B,A$
will cause a trace of all BASIC program lines. Before each line is traced, the values of the expression INT(X)+A*B and of the variable A$ will be displayed.

10 NTRON(:50),RB,TE$,LEN(A$),MID$(A$,LEN(A$)-1,2)
.
.
50 REM
.
.
1000 NTRON(1000:)
will cause lines 10 through 50 to be traced after line 10 is executed. Additionally, the values of the expressions shown will be displayed prior to the line trace display. After line 1000 is executed, only line 1000 and any lines that follow will be traced.

Line Range

Line range specifies a line or a range of lines of a BASIC program. Whenever used it must be enclosed by parentheses. A line range must be followed by a comma or a colon (whichever is appropriate) if it does not terminate a line. A line range can be given in several forms, as detailed below. Several examples illustrate the usage of each form. The examples use constants for "line number", but it should be understood that variables or numeric expressions may also be used. All examples assume the following program is in memory:

10 REM
20 PRINT "THIS IS A SHORT PROGRAM"
30 PRINT "WHOSE ONLY PURPOSE IS TO"
40 PRINT "OCCUPY LINES 10 - 50"
50 REM

continued on next page

continued from previous page

1). (line number 1 : line number 2)
specifies the range of lines beginning from "line number 1" (inclusive) through "line number 2" (inclusive). If "line number 1" does not exist, the first line number that DOES exist which is greater than "line number 1" will be used as the lower limit of the range. If "line number 2" does not exist, the first line number that DOES exist which is less than "line number 2" will be used as the upper limit of the range.

EXAMPLES:

(20 : 40)
includes lines 20, 30 and 40.

(5:25)
includes lines 10 and 20.

(18:100)
includes lines 20, 30, 40 and 50.

2). (: line number)
specifies the range of lines beginning from the first program line through "line number" (inclusive). If "line number" does not exist, the first line number that DOES exist which is less than "line number" will be used as the upper limit of the range.

EXAMPLES:

(:100)
includes lines 10, 20, 30, 40 and 50.

(: 28)
includes lines 10 and 20.

(:5)
does not include any line and will result in an error.

3). (line number :)
specifies the range of lines beginning from "line number" (inclusive) through the last program line. If "line number" does not exist, the first line that DOES exist which is greater than "line number" will be used as the lower limit of the line range.

EXAMPLES:

(0 :)
includes lines 10, 20, 30, 40 and 50.

(45 :)
includes line 50.

4). (line number)
specifies a particular line which MUST exist. If "line number" does not exist, an error will result.

EXAMPLES:

(10)
specifies line 10 as the "line range".

(18)
will result in an error since line 18 does not exist.

ON ERROR GOTO line number

The ON ERROR GOTO command has been modified in several ways, but will still function properly with existing programs. The changes made include:

1). The "line number" argument may be a variable or expression, as well as the usual numeric constant. This change does NOT apply for ON ERROR GOTO used in Command mode.

2). An ON ERROR GOTO "line number constant" issued in Command mode will be ignored. (An ON ERROR GOTO with a variable or expression given as the "line number" will cause an error).

3). If an error occurs in Command mode, any ON ERROR GOTO in effect will be ignored.

Note: You canNOT trap NEWBASIC's UNTIL WITHOUT DO or MISSING NEWBASIC COMMAND errors. These errors will always cause execution to terminate.

EXAMPLES:

10 ON ERROR GOTO X+100
will cause execution to transfer to the line number determined by the sum of the current value of X and 100, upon an error during program execution.

ON ERROR GOTO 10000
will be ignored since it was issued in Command mode.

POFF
PON

The PON command, when executed, will cause video output to be echoed to the printer. The printer must be on and ready or the system will "hang". The POFF command is used to disable the video to printer echo. The command may be used with custom printer driver routines (serial,TRS232,etc.) providing the routine has been loaded before issuing a PON. See SPOOLON for the use of it and PON simultaneously.

EXAMPLES:

10 PON
20 PRINT"THIS IS A TEST"
30 POFF
will display the line "THIS IS A TEST" on the video display and also output the line to the printer.

PON : LIST
this command will cause a program in memory to be listed on the video display and to also be output to the printer.

QUICKEY

This command toggles the quick key entry method on and off. The first time the command is given, quick key entry is enabled; the next time it is given, quick key entry will be disabled.

Quick key entry allows you to enter entire BASIC keywords with two keystrokes. The first key alerts the computer that you desire to make a quick key entry. This key is either the down-arrow on the left hand side of the keyboard, or CLEAR on the right hand side. If a second key is not entered immediately, a blinking asterisk will appear at the current cursor position. This is a reminder that you are in quick key entry mode. If the key you press next is NOT a defined quick key, then it will be displayed (or the action such as ENTER or CLEAR will occur) as though you had never initiated a quick key entry. If the key you press IS defined, then the associated keyword will immediately be displayed on the video screen. When quick key entry is enabled, the CLEAR and down-arrow keys do not function as usual. You can issue a down-arrow (that is, a line feed) or clear simply by pressing the desired key TWICE rather than once. You may also press (SHIFT) CLEAR to issue a clear.

The table below lists BASIC keywords and associated keys, organized alphabetically by the BASIC keyword. An attempt has been made to make quick key entries easily recallable. Many of the keywords are mnemonic (that is, the first letter of the keyword matches the associated key). Exceptions are the BASIC string keywords (that is, keywords with "$" in them) which have been assigned to numeric keys. Other keywords also use mnemonics different from the BASIC keyword (Xfer (X) for GOTO or Bring (B) for LOAD, for example). Some keywords use keyboard position to aid in recalling their assigned quick keys (RETURN (H), for example, is next to GOSUB (G)). Of course, some keywords had to be assigned at random (you can't win them all!).

AUTO	- A	KILL"	- K	READ	- Y
CHR$(	- 4	LEFT$(	- 1	RESET(	- Q
CLEAR	- Z	LIST	- L	RETURN	- H
CLOSE	- J	LOAD"	- B	RIGHT$(	- 3
CMD"	- C	MEM	- M	RUN <ENTER>	- R
DATA	- D	MID$(	- 2	SAVE"	- S
DIM	- U	MKD$	- 8	SET(	- W
ELSE	- E	MKI$	- 9	STR$(	- 6
FOR	- F	MKS$	- 0	STRING$(	- 7
GOSUB	- G	NEXT	- N	THEN	- T
GOTO	- X	OPEN"	- O	VARPTR(	- V
INKEY$	- 5	PEEK(	- P	blank:blank	- :
INPUT	- I	POKE	- @		

The above table may be photocopied and placed in the lower right corner of the video display for easy reference.

RECT X1,X2,Y1,Y2 (,R)

The RECT command without the trailing ",R" turns on the graphics blocks of the perimeter of the rectangle determined by the imaginary diagonal line drawn from the point X1,Y1 to the point X2,Y2. The RECT command with the ",R" resets (i.e. erases) the determined rectangle perimeter. This command has the same limits and parameters as the FILL command.

NOTE: Parentheses before and after the arguments field (unlike SET and RESET) are not needed and if present will cause an error.

EXAMPLES:

10 RECT 0,0,127,47
will draw a rectangle perimeter one graphic block wide around the outside of the video display.

20 RECT 127,0,0,47,R
will reset the rectangle displayed in the preceding example.

300 RECT X+15,C*SIN(D),X2*X2,43
will draw the determined rectangle perimeter provided the evaluated arguments are in range.

40 RECT 127-I,0+I,I-127,0-I,R
will reset the determined rectangle perimeter with the same precaution as in the previous example.

REPEAT

The REPEAT command implements the keyboard auto repeat and "beep" toggle. The first time this command is executed, the routine takes effect. A key depressed for about half a second will repeat. If a speaker/amplifier is hooked to the cassette-out line of the S-80 (the line that goes to the AUX on the cassette recorder), a beep will be heard upon depressing a key. The second time this command is executed, the auto repeat and "beep" are turned off. The REPEAT command should not be used with Radio Shack's lower-case software as lower-case characters cannot be input with REPEAT in use.

NOTE: The motor-on relay in the S-80 keyboard is not used. It is recommended that the audio and control cable be attached to the S-80 CPU/keyboard rather then the expansion interface. If this is not the case, the "beep" will not be heard unless a motor-on command has been issued at least once to the expansion interface. The seperate relay in the E/I must be turned on for audio output through its rear jacks.

EXAMPLES:

REPEAT
will enable key repeat and "beep" if this is the first time the command is issued.

20 REPEAT
30 INPUT"NAME?",A$
40 REPEAT
if a REPEAT command has already been issued, as in the preceding example, line 20 will turn off the auto repeat. Line 30 will input A$ without allowing auto-repeat. Line 40 will again enable repeat/"beep".

REST line number
REST #"label"

This command allows for selected reading of DATA statements. The argument specifies a BASIC program line number containing a DATA statement. After the REST command is executed, the next READ will start with the first item in the DATA statement within the line just RESTored.

The "line number" argument form may be given as a constant, variable, or expression. The second REST argument form (**#**"label") refers to a line which has been so labeled. With either argument form, if the line referenced does not contain a DATA statement, an ILLEGAL FUNCTION CALL error will result.

EXAMPLES:

10 REST 40000
will reset the DATA pointer so that the next READ will start with the first item in the DATA statement in line 40000.

10 REST A(X)*100
will cause the next READ to start with the first item in the DATA statement in the line determined by the product of A(X) and 100.

RS232

This command initiates dialog that will result in the initialization of the RS-232-C interface. Current status of the UART sense switches will be displayed within vertical bars. If the baud setting is not within the range possible to set with RS232, the word "OTHER" will be displayed. To use the parameters that were switch set, hit "ENTER" in reply to each question. To change settings, enter the appropriate number. Recommended settings are denoted by an asterisk.

EXAMPLE:

RS232
will result in the following dialog:

```
                    RS-232-C INITIALIZATION
BAUD RATE (0=110, 1=134.5, 2=300*) |2|? 0
PARITY (0=ENABLE*, 1=DISABLE) |0|? 0
STOP BITS (1*, 2) |1|? 1
WORD LENGTH (5, 6, 7*, 8) |7|? 7
PARITY (0=ODD, 1=EVEN*) |1|?
```

the numbers after the "?" are user entered. The numbers inside the vertical bars are the current UART switch settings. This dialog has resulted in a baud rate of 110, parity enabled, 1 stop bit, 7 bit word length excluding parity bits, and even parity.

RSIN (,R)

The RSIN command without the trailing ",R" will enable the input of characters from either the keyboard or the RS-232-C interface. Providing processing is not too lengthy between input of characters, rates up to and including 300 baud can be supported. The RSIN command with the trailing ",R" (i.e. RSIN,R) will inhibit input from the serial port. Further input will only be from the keyboard.

This command used in conjunction with the RSOUT command makes possible the implementation of terminal programs, automatic logon to time share networks, and remote terminal input and output in BASIC!

NOTE: Do not implement the RSIN command if a RS-232-C board or similar serial I/O device is not installed. An infinite loop will occur and a system reset will be necessary.

EXAMPLES:

10 RSIN : RSOUT
20 LPRINT : LPRINT
30 LINEINPUT A$: IF A$<>"USER ID?" THEN 30
40 LPRINT"123,45,6789" : REM USER LOGON ID
this example shows how simple an automatic logon procedure for a time share system is to implement. In line 10, the RSIN command enables input from the serial port. "RSOUT" routes printer output to the serial port. Line 20 outputs two carriage returns. This is usually necessary to establish baud rate synchronization with a time share network. In line 30, a character stream, up to a carriage return, is stored in the string variable A$. If A$ is not equal to "USER ID?" the program loops at line 30. Once the IF/THEN clause is true, the string "123,45,6789" is output to the serial port in line 40. Further processing could include an interactive terminal program, etc.

100 RSIN,R
will inhibit any further input from the serial port.

RSOUT (,R)

The RSOUT command without the trailing ",R" will route printer output to the RS-232-C interface. The serial port should be initialized before implementing this command. While any baud rate possible with RS-232-C is supported, no provision has been made for outputting nulls after a carriage return. Nulls are required by some serial printers and other devices. The RSOUT command with the trailing ",R" (i.e. RSOUT,R) will reset printer output to the condition existing before the RSOUT command. Refer to the RSIN command.

EXAMPLES:

10 RSOUT
20 FOR I=1 TO 10 : LPRINT"HELLO" : NEXT I
30 RSOUT,R
this example uses the RSOUT command in line 10. The 10 "HELLO"'s output to the printer in line 20 will go to the serial output port. Line 30 resets to conditions existing before line 10 was executed.

10 RSOUT : PON
20 PRINT"THIS IS A TEST"
30 POFF : RSOUT,R
line 10 first routes printer output to the RS-232-C interface and then echoes all video output to the printer (i.e. the serial port). Line 20 outputs the line "THIS IS A TEST" to the video display and the serial port. Line 30 turns off the video to printer echo and reroutes further printer output as per the conditions existing before line 10 was executed.

SOUND frequency, duration

The SOUND command outputs, to the cassette port, a tone of requested frequency and duration. The arguments of this command may be constants, variables, or expressions. SOUND uses the integer portion, hence, the arguments themselves need not be integers. The command is valid for frequency and duration arguments between 0 and 255 inclusive. Arguments outside this range will produce an "ILLEGAL FUNCTION CALL" error.

The motor-on relay in the S-80 CPU/keyboard is not used. It is recommended that the audio and control cable be attached to the CPU/keyboard rather than the expansion interface. If this is not the case, the sound will not be heard unless a motor-on command has been issued to the expansion interface at least once. The

continued on next page

continued from previous page

audio out cable (AUX IN on the cassette recorder) should be connected to a speaker/amplfier or to the AUX IN of an audio system.

A table of approximate frequencies and durations for SOUND follows:

FREQUENCY ARGUMENTS

argument	tone in Hertz	argument	tone in Hertz
0	14250	17	1040
1	8100	18	1000
2	5500	19	950
3	4500	20	910
4	3500	25	750
5	3000	30	630
6	2600	35	540
7	2300	40	475
8	2100	50	380
9	1850	60	310
10	1700	80	240
11	1500	100	190
12	1425	125	150
13	1320	150	140
14	1250	200	90
15	1170	255	65
16	1100		

DURATION ARGUMENTS

argument	time in seconds	argument	time in seconds
0	.01	50	.36
5	.04	100	.72
10	.08	150	1.08
15	.12	200	1.44
25	.19	255	1.84

EXAMPLES:

```
10 SOUND 19,150
```
will produce a tone of 1000 hertz for about 1 second.

```
100 SOUND X+3*(A+B),255-X
```
will produce a tone of the determined frequency for the determined duration.

SPOOLOFF
SPOOLON "filespec"

The SPOOLON command directs printer output to the specified disk file. The command is disabled and the disk file closed by the SPOOLOFF command. Spooling, used in conjunction with the DESPOOL command, allows for more efficient hardcopy output. Printouts that normally would require a large amount of time to output may be quickly spooled. Later the created file may be despooled while other computer operations are being performed.

SPOOLON may be used before a PON command. This allows all output to the video display to also be output to a disk file. This capability is useful for keeping track of programs run, or for debugging purposes. A POFF command should be issued before SPOOLOFF if PON is previously used and a printer is not "ready". If this is not done, and the printer is not powered up and ready, the system may "hang".

Only one file at a time may be spooled to. If there is a problem initializing the requested file, or SPOOLON has already been issued, an ILLEGAL FUNCTION CALL will occur. If an error such as DISK SPACE FULL occurs during spooling, an error message will be displayed and the spooled-to file will be closed. If PON was used, a POFF command will be issued.

NOTE: SPOOLOFF must be used to halt spooling or the contents of the spooled-to file will be lost. The simultaneous use of SPOOLON, PON, and DIR under NEWDOS 2.1 or NEWDOS/80 may, under some circumstances, cause anomalous results.

EXAMPLES:

```
10 SPOOLON"HOLD/TXT"
```
this creates a file "HOLD/TXT" and future output directed to the printer will be placed in the file.

```
100 SPOOLON"JUNK/PRT.PASSWORD:1"
```
will create a file "JUNK/PRT" on drive 1 with the password, "PASSWORD". Printer output will be directed to this file.

```
10 SPOOLON"TEST/HLD"
20 PON
30 PRINT"TESTING ... 1,2,3"
40 POFF : SPOOLOFF
```
this program will create the file, "TEST/HLD", output the line "TESTING ... 1,2,3" to both the video display and the file, stop video-to-printer echo, and close the file.

WTS

The WTS command is used to "white" the video screen. (That is, turn all graphics blocks "on".) The current cursor position is NOT changed.

EXAMPLES:

```
WTS
```
will white the screen "instantly".

```
10 FOR I=1 TO 10
20  WTS : PRINT@148, CHR$(23);"RED ALERT!";
30  FOR J=1 TO 100 : NEXT
40  CLS : PRINT@148, CHR$(23);"RED ALERT!";
50  FOR J=1 TO 100 : NEXT
60 NEXT
```
will display "RED ALERT!" while the video screen alternates between white and black.

GLOSSARY FOR NEWBASIC

ASCII - American Standard Code for Information Interchange. This method of coding is used for textual data.

baud - Signalling speed in bits per second. The term is usually used in conjunction with serial input and output. See RS-232-C.

command file - A DOS file with the extension /CMD. The file consists of Z-80 object code (machine language).

DCB (data/device control block) - An area in RAM associated with an Input/Output device.

expression - A meaningful sequence of one or more constants or variables possibly used with operators and functions.

filespec - A sequence of characters which specifies a particular disk file. It consists of a mandatory filename, of up to eight characters, followed by an optional extension, password, and drivespec.

hex-constant - An integer number expressed in hexadecimal notation (base 16), preceded by "&H". Leading zeros are ignored. The largest unsigned integer which can be expressed as a hex-constant is 65535 (&HFFFF).

int-expression - An expression that when evaluated lies within the BASIC integer range (-32768 to +32767). If the expression value is not an integer, it will be rounded to the closest integer.

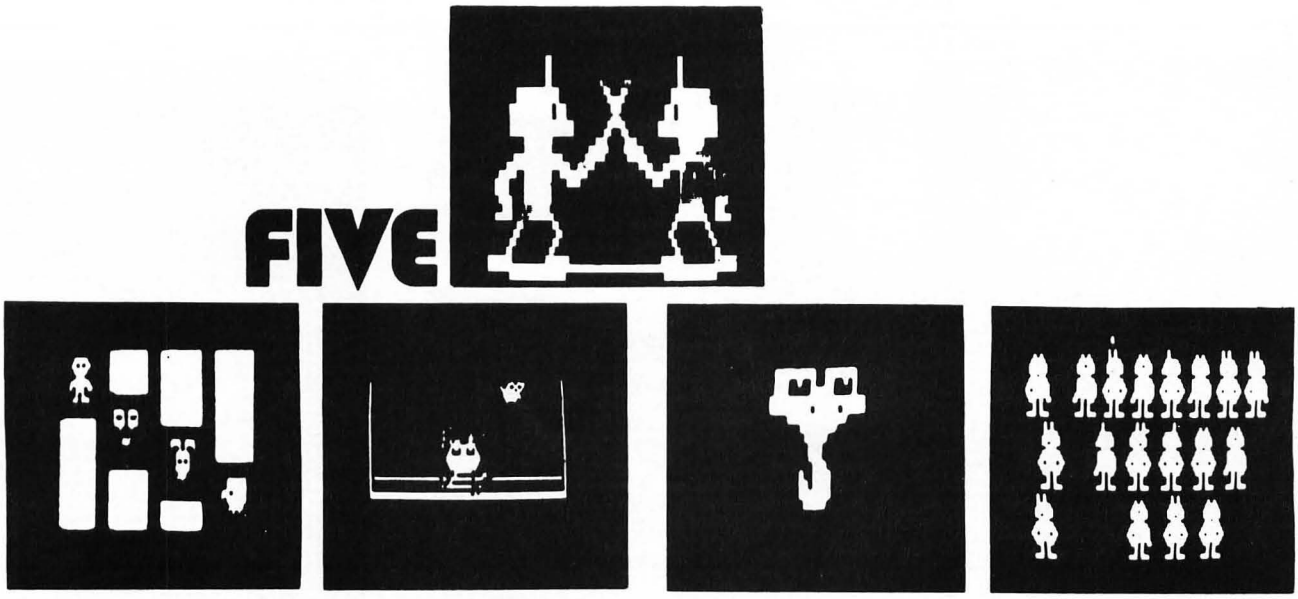
#"label" - A string of up to 251 alphanumeric characters preceded by a pound sign and enclosed in double quotes (# "ALABEL"). Any characters other than a double quote may appear within the label defined by the begin and end quotes.

line number - A constant, variable, or numeric expression which specifies a BASIC program line number. Usually this line number must exist within the BASIC program currently in memory, or an error will result. When used within a "line range", however, "line number" need not exist.

line range - A line or a range of lines of a BASIC program. Whenever used it must be enclosed by parentheses. See NTRON.

pos-expression - An expression preceded by a plus sign ("+") indicating that the value of the expression should be evaluated as an integer between 0 and 65535 inclusive.

RS-232-C - Refers to a specific EIA (Electronics Industries Association) standard which defines a widely accepted method for interfacing data communications equipment. The S-80 holds its RS-232-C interface in the expansion interface.



FROM CHRISTOPHERSON

LIFE TWO

by Leo Christopherson (Ramware)

Two in one: ''Game of Life,'' at an astounding 100 generations a minute, plus ''Battle of Life'' with animated creatures and sound.

16K, S-80 Cassette Level II ..... ~~$14.95~~ $13.50

SNAKE EGGS

by Leo Christopherson (Ramware)

Here is a computerized reptillian version of 21 complete with arrogant snakes and appropriate sound.

16K, S-80 Cassette Level II ..... ~~$14.95~~ $13.50

BEE WARY

by Leo Christopherson (Ramware)

This fast-paced real time action game is a contest between a bee operated by the player and a spider operated by the computer. Machine language subroutines, but loads as Level II for easy operation.

16K, S-80 Cassette..... ~~$14.95~~ $13.50

ANDROID NIM

by Leo Christopherson (Ramware)

Three rows of continuously animated androids and three executioners provide a new dimension as you try to beat the computer at the age old game of ''Nim.''

16K, S-80 Cassette,

Machine Language ..... ~~$14.95~~ $13.50

DUEL-N-DROIDS

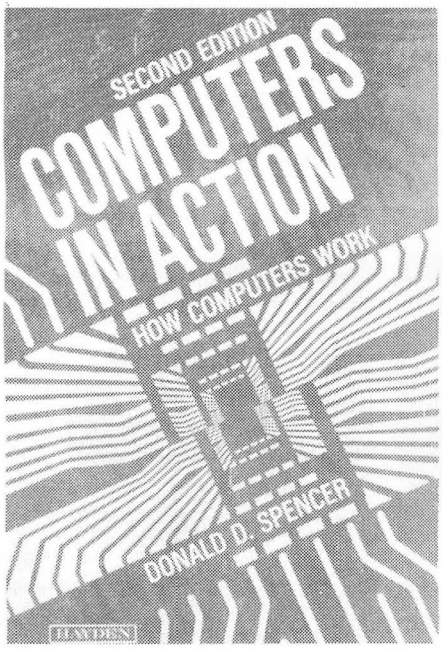
by Leo Christopherson (Ramware)

You are the fencing master and all you need to do is train your android by making him duel the machine's androids. After he is trained, enter him in the tournament and sit back and enjoy the fun. Features included in this game are: sound effects, extensive graphic displays & multiple playing levels.

16K, S-80 Cassette..... ~~$14.95~~ $13.50
16K, S-80 Disk..... ~~$14.95~~ $19.00

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585

New Books from TSE HARDSIDE



Computers in Action

by DONALD D. SPENCER

What do banks, airlines, hospitals, the police and utility companies have in common? They all have made computers a vital part of their daily routine. And now you can understand the vital part that computers play in our society with this second edition of a popular introduction to computers. The book answers the questions: "What are computers?" "How do computers work?" and "What can a computer do?" New features of this second edition include: microcomputers and microprocessors; mass memory devices such as floppy disks; new input/output units; and a section on structured programming, the APL language, and the RPG language. Numerous photographs, diagrams and cartoons illustrate specific concepts and equipment. Technical terms are kept to a minimum and are appended with easy-to-understand definations. HAYDEN BOOK COMPANY, Inc.

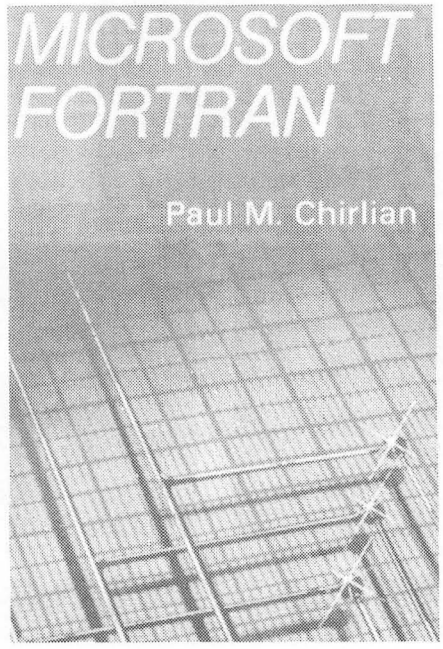
Order #65-210007B..... $7.25

Apple Basic for Business

by ALAN PARKER & JOHN STEWART

Here is a totally practical guide to learning BASIC on the APPLE-II series of computers. "Apple BASIC for Business" can help you build the programming skills you need to succeed in business data processing. You'll develop speed and technical accuracy for the scores of business applications encountered in a typical company. A problem-solving approach leads you through the BASIC language and its use with the Apple-II. Step-by-step you'll master the processing techniques with sample programs, flowcharts and problems all designed to give you actual practice in finding programming solutions for every business data processing challenge. RESTON PUBLISHING COMPANY, Inc.

Order #65-211002B..... $14.95



Microsoft Fortran

by PAUL M. CHIRLIAN

This is an introductory book on FORTRAN. It is intended for first time users that have no experience with FORTRAN or any other programming language. It is a general introduction to the language as well as a specific introduction to MICROSOFT FORTRAN. This is the version of FORTRAN used on a TRS-80 or an APPLE. This book presents procedures for using FORTRAN in a variety of computers. The various topics are presented in a straightforward and simple manner. You will learn such things as structured programming and top down programming also. Best of all, you will be able to write your program when you get to page 7. In edition to showing you how to program in FORTRAN, this book will also show how to use your computer's compiler, editor and linker. DILITHIUM PRESS.

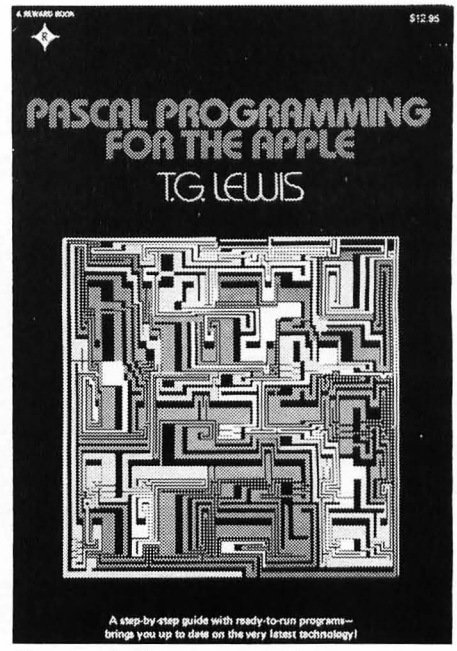
Order #65-226010B..... $14.95

Pascal Primer

by DAVID FOX & MITCHELL WAITE

If you are learning programming or have dabbled in the popular language BASIC and wish to learn the capabilities of PASCAL, this book is definitely written for you. PASCAL is a computer language with features and capabilites found only among the most exotic and expensive languages. The purpose of this book is to teach you how to use PASCAL to write powerful programs. The most widely used version of PASCAL is the UCSD version, and this is the version that was used as the guide for writing this book. Written and illustrated with a touch of humor, the informative text describes PASCAL program structure, PASCAL variables, PASCAL procedures, and many other features. There are chapters on decision-making statements, numeric functions, string functions, arrays and sets and much more. HOWARD W. SAMS & CO., Inc.

Order #65-251001B..... $16.95



Pascal Programming for the Apple

by T.G. LEWIS

Now you can upgrade your computer expertise to include PASCAL, the greatest advance in microcomputer software since BASIC. Discussing features unique to UCSD PASCAL on the apple-II, this easy-to-use guide puts you in touch with the latest technology. The step-by-step instructions and clear-cut examples will benefit both the hobbyist and the professional. Opening chapters cover the fundamentals of the PASCAL System and provide a review of the language. You will then sharpen your programming skills by learning new techniques, and you will learn all about graphics modules in turtlegraphics... musical tone generators... and file structure operations. Plus, there are scores of valuable programs that are ready to run, including financial applications for home mortgage payments, stock market charting, and cash flow analysis. Special sections on text processing, combining the Apple computer's graphics and sound reproduction hardware with UCSD software, and implementing large programs on small computers, brings you up to date on the very latest developments in programming.

RESTON PUBLISHING COMPANY,Inc.

Order #65-211003B..... $12.95

OOPS. . . New items we almost forgot.

CONFLICT
Conflict is a strategic simulation for 2 players and is loosely based on the famous board game of world conquest. The game takes place in outer space with 2 factions, you and your opponent, both fighting for control of the newly discovered planet in the Andromeda system. Control of this planet would give the Earth the badly needed natural resources and minerals it used up back in the late part of the twentieth century. Operates on the TRS-80 Color Computer, requires: 16K RAM with or without Extended BASIC.
Order Number #29-242010T..... $ 14.95

ANIMATED HANGMAN
Animated Hangman is a delightful computerized adaption of standard Hangman. He winks... he blinks... he almost lives! An outstanding game for the whole family! Operates on the TRS-80 Color Computer, requires: 16K RAM with or without Extended BASIC.
Order Number #29-242011T..... $ 14.95

DISASSEMBLER 6809
BASIC language Disassembler for the TRS-80 Color Computer. Requires 16K RAM with or without Extended BASIC.
Order Number #29-242012T..... $ 14.95

NEWSCRIPT 6.0
This is a self-contained, comprehensive Word Processing system. It combines the excellent functions of IBM's "mainframe" computers with the ease-of-use of the TRS-80, and supports the advanced features of several modern printers. Works on TRS-80 Model-I or Model-III, requires: at least 1 disk drive and 48K RAM.
Order Number #25-259001D (Model-I)..... $ 99.95
Order Number #27-259001D (Model-III)..... $ 99.95

INFERNO A Fantasy Adventure
Enter the INFERNO and visit: The Underground Forest, The Mines of Moria, The Cave of the Winds, The Lava Bridge over the Molten River. Oh, by the way leaving is quite another matter! Operates on the APPLE-II, requires: Applesoft in ROM, 48K RAM and DOS 3.3.
Order Number #36-260001D.. $ 29.95

6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

Exterminate

by George Geczy

A buggy S-80 graphics game requiring 16K of memory.

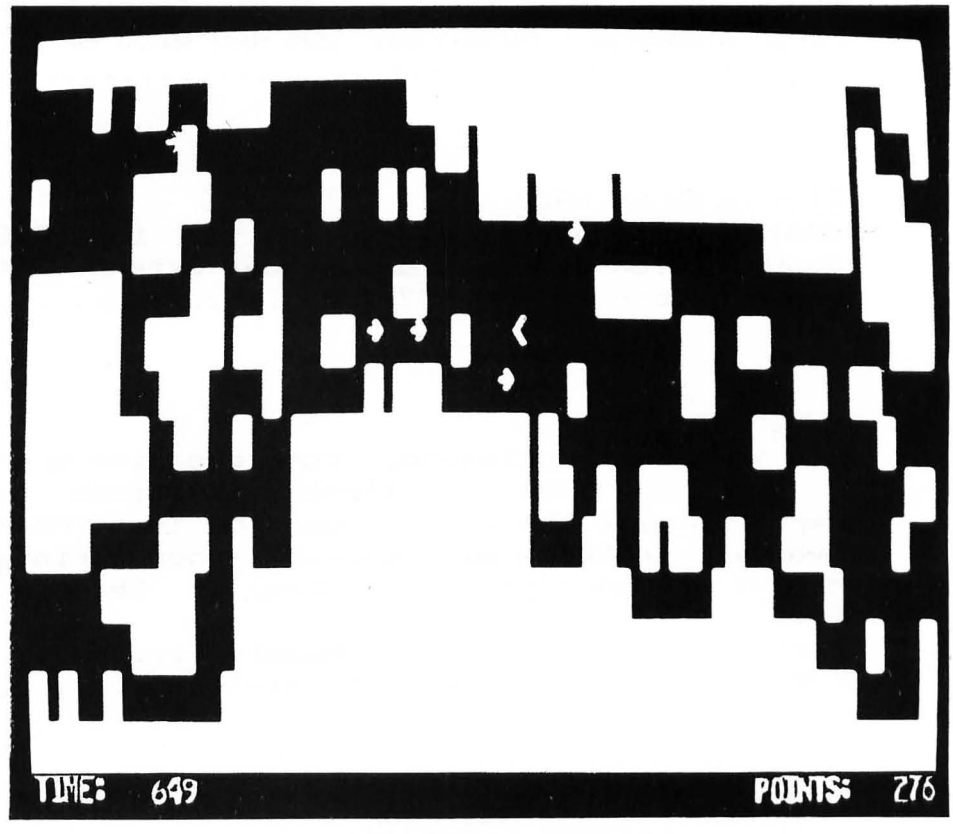
For once, we freely admit that this program has bugs in it. Lots of them. In fact, you're likely to be quite overrun with bugs when you try playing it. But then, that's the whole point.

It seems that you have just inherited a nice, old-fashioned cave from one of your favorite uncles. It is, unfortunately, infested with bugs; and since you're anxious to avoid getting stung by a high-priced exterminator, you decide to do the job yourself.

Whether that is a wise decision remains to be seen. This program will try your mettle to see what kind of bug basher you really are. The object is to kill as many bugs as you can before they overrun you, or before your preselected number of bugs or turns is reached. Five different skill levels allow you to advance from timid amateur to vicious professional as you litter the cave floor with squashed corpses. Complete instructions for this sadistic flight of fancy are included in the program.

VARIABLES

A: Used in PEEKing keyboard memory.
AD: The advantage of the player over the computer. This decreases as the game progresses.
CL: Screen location of the player.
D(x): Contains the ASCII codes for the directional arrows. (If you have a Model III you may want to insert other characters, in line 100.)
DD: Number of bugs killed so far.
E1(n): Starting coordinate of bug number n.
E2(n): Direction of the last move of bug number n.
EL: Screen location of the bug currently being moved.
EX: Number of bugs on the screen at one time, minus five.
EX(n),EY(n): The x,y coordinates of bug number n.
IN(x),IK(x): Used to convert the variables A and ZA$ into a direction number.
PO: Total accumulated points.
PT: Point total specified by user.
SAD: Used to calculate variable AD.
SL: Skill level specified by user.
TL: Turn limit specified by user.
TR: Turn number.
TT: ASCII value of a space (=32).
YC: ASCII value of your character (=60).
XX,YY: Player's x,y coordinates on the screen.
ZA$: INKEY$ variable.




```
0 'EXTERMINATE          COPYRIGHT (C) 1980 BY GEORGE GECZY
```

Draw the title page.

```
5 CLS:PRINTCHR$(23);STRING$(32,153):FORX=15486TO16320STEP64:POKE
X,191:POKEX+2,191:NEXT:PRINT@960,STRING$(31,155);:POKE16382,155:
POKE15424,191:PRINT@266,"E X T E R M I N A T E";:PRINT@516,"COPY
RIGHT (C) 1980";:PRINT@720,"BY GEORGE GECZY";
7 PRINT@900,"WOULD YOU LIKE INSTRUCTIONS?";
10 FORX=16260TO16316STEP2:SC=PEEK(X):POKEX,191:FORY=1TO15:NEXT:P
OKEX,SC:NEXT:A$=INKEY$:IFA$=""THEN10ELSEIFA$="N"THEN100ELSEIFA$<
>"Y"THEN10
```

Print instructions with a special printing routine using DATA statements..

```
15 CLS:PRINTSTRING$(24,145);"*-INSTRUCTIONS-*";STRING$(24,162);:
FORX=15424TO16320STEP64:POKEX,157:NEXT:FORX=15487TO16383STEP64:P
OKEX,174:NEXT:A$=INKEY$
20 FORY=1TO7:FORA=1TO10:READA$:SC=134+64*A+15360:FORX=1TO50:SC=S
C+1:POKESC,191:POKESC,ASC(MID$(A$,X,1)):NEXT:NEXT:PRINT@915,"PRE
SS ANY KEY TO CONTINUE";:GOSUB95::NEXTY
```

```
21 GOTO100
```

DATA statements containing the instruction lines.

```
22 DATA"CONGRATULATIONS!!!!  YOU HAVE JUST INHERITED  A ","CAV
E FROM YOUR LONG LOST UNCLE GOOBERVINCLE!  A  ","CAVE, YOU ASK?
  THAT'S WHAT I SAID!  A REAL OLD- ","FASHIONED UNDERGROUND-TYP
E CAVE.  NOW  THE  BAD  "
23 DATA"NEWS...  IT SEEMS THAT THE CAVE IS INFESTED WITH  ","SOM
E RATHER LARGE BUGS...  I KNOW THAT YOU'RE  ","EAGER TO MOVE
RIGHT IN, BUT THEY HAVE TO BE MOVED ","OUT FIRST. SINCE AN EXTER
MINATOR COSTS $54 AN HOUR"
24 DATA"PLUS $10 PER BUG, I'M SURE THAT YOU'D RATHER DO  ","THE
 WORK YOURSELF.  IT CAN'T BE THAT HARD, CAN IT?"
25 DATA"HOW DO YOU EXTERMINATE BUGS, YOU ASK?  SIMPLE!  ","YOU
 JUST STEP ON THEM!  (WHAT CAN BE EASIER?)  ","BUT THERE ARE
SOME POINTS TO BE AWARE OF - FIRST, ","THESE BUGS HAVE SOME BIG
FRIENDS IN THE GOVERN-  "
26 DATA"MENT, AND IF THEY GET OUT OF THE CAVE THEY CAN GO ","AND
 GET YOU ARRESTED.  ALSO, THEY TEND TO BREED  ","FASTER THAN AN
Y OTHER LIVING THING, SO YOU MUST  ","KILL QUITE A FEW IN AS SH
ORT A TIME AS POSSIBLE.  "
27 DATA"BUT SITTING IN THE FRONT OF THE CAVE AND KILLING  ","THE
M AS THEY COME WILL NOT WORK.  ","YOU MUST GET T
HEM AS CLOSE TO THE BACK OF THE CAVE","(WHICH IS THE LEFT SIDE O
F THE SCREEN) AS YOU CAN."
28 DATA"IF YOU DON'T KILL THEM FAST ENOUGH, OR NOT CLOSE  ","END
UGH TO THE BACK, THEY CAN SOON OVERCOME YOU.  ","
                              ","BEFORE YOU BEGIN PLAYING,
 YOU WILL BE ASKED THREE "
29 DATA"QUESTIONS. FIRST, 'SKILL LEVEL (1-5) ?' WILL LET  ","YOU
 CONTROL THE AMOUNT OF BUGS YOU ENCOUNTER AND  ","OTHER ADVANTAG
ES.  AN ANSWER OF '1' WILL PROVIDE  ","AN EASY, BEGINNERS GAME,
WHILE '5' IS VERY HARD.  "
31 DATA"BEFORE I GET ON WITH THE OTHER TWO QUESTIONS, I'D ","BET
TER ANSWER THOSE OF YOU WHO ARE ASKING HOW MANY","BUGS YOU ACTUA
LLY HAVE TO KILL.  WELL, I SAID THAT","THEY BREED QUICKLY, AND Y
OU'LL FIND THAT AS SOON  "
32 DATA"AS YOU KILL ONE, ANOTHER WILL COME IN ITS PLACE.  ","WEL
L, TOUGH LUCK. YOU CAN AT LEAST DIE TRYING!  ","YOU WILL PROBA
BLY BE ABLE TO GO ON FOREVER IN  ","SKILL LEVEL 1 AFTER THE F
IRST FEW GAMES, BUT 5 IS "
33 DATA"NOT SO EASY.  VERY FEW PEOPLE HAVE EVER MADE IT  ","PAS
T 2000 POINTS. BUT FOR YOU PEOPLE WHO DON'T  ","WANT TO STEP I
N THE CAVE (THERE WAS A PUN IN  ","THERE) FOR TOO LONG, YOU
CAN INPUT A NUMBER WHEN  "
34 DATA"THE COMPUTER ASKS YOU FOR A 'POINT TOTAL'. AS SOON","AS
YOU GET THAT MANY POINTS, THE GAME WILL END.  ","BUT SEEING AS
MOST OF YOU WILL NOT WANT TO DO THIS","BECAUSE YOU WILL NOT LAST
 LONG ANYWAY, YOU JUST  "
36 DATA"HAVE TO PRESS ENTER ONLY.  NOW, IF YOU DO PRESS  ","ENT
ER ONLY, YOU WILL BE ASKED FOR A 'TURN LIMIT'. ","IF YOU INPUT A
 NUMBER HERE, THE GAME WILL END WHEN","YOU REACH THAT NUMBER OF
TURNS.  USE THIS ONLY IF "
37 DATA"YOU WANT TO TRY FOR AS MANY POINTS AS YOU CAN GET ","IN
A SPECIFIC NUMBER OF TURNS, OTHERWISE JUST HIT ","RETURN.  FINA
L POINTS: YOU ARE THE '<' IN THE  ","UPPER RIGHT CORNER, THE B
UGS WILL BE THE ARROWS.  "
39 DATA"THE COMPUTER WILL DRAW THE CAVE FIRST, AND THEN  ","THE
 BUGS WILL START COMING WHEN YOU PRESS ANY KEY.","YOU USE THE FO
UR ARROW KEYS TO MOVE YOURSELF.  ","HOLD THE KEYS DOWN TO MOV
E LONG DISTANCES, OR JUST"
40 DATA"TAP THE KEY TO MOVE OVER ONE.  YOU STEP ON A BUG  ","BY
RUNNING ONTO IT, AND IF IT'S FAR ENOUGH TO THE ","LEFT YOU GET P
OINTS.  BUT HURRY - YOU DON'T HAVE  ","TO EVEN BE NEAR BUGS, BUT
 YOU CAN BE OVERRUN IF  "
41 DATA"YOU HAVEN'T KILLED ENOUGH.  LAST BUT NOT LEAST,  ","YOU
 MIGHT HAVE TO ADJUST YOUR CONTRAST/BRIGHTNESS "
42 DATA"AFTER THE CAVE IS DRAWN.  ","
                                  ","
      GOOD LUCK!!  ","
                  "
43 DATA"  ","
                                  ","
                                ","
                  "
```

Subroutine to wait for a keypress.

```
95 IFINKEY$=""THEN95ELSEPRINT@915,"
";:RETURN
```

Beginning of main program. Initialize variables.

```
100 CLEAR500:RANDOM:DEFINT A-Z:YY=0:XX=63:YC=60:TT=32:D(1)=94:D(
2)=91:D(3)=92:D(4)=93:DL=300:D1=250:CLS
105 IN(1)=9:IN(2)=8:IN(3)=91:IN(4)=10:IK(1)=64:IK(2)=32:IK(3)=8:
IK(4)=16
```

Input the skill level and other information. Calculate the user's advantage (AD) over the computer.

```
110 PRINT:INPUT"SKILL LEVEL (1-5)";SL
115 IFSL<1ORSL>5THEN110
120 INPUT "POINT TOTAL";PT
130 IFPT=0THENINPUT"TURN LIMIT";TL
140 EX=SL:SAD=(6-SL)+5:AD=SAD-1:PO=150
```

Draw the cave.

```
200 CLS:FORSC=1TO4:PRINTSTRING$(240,191);:NEXT
220 FOR SC=15423TO16383STEP64:POKESC,32:NEXT:FORSC=16320TO16383:
POKESC,32:NEXT
225 YY=0:XX=63:POKEYY$64+XX+15360,60
230 PRINT@960,"DRAWING THE CAVE...  PLEASE WAIT.";
250 FORMA=1TO5
300 B=RND(13):E1(MA)=B:E1(MA+EX)=B:EY(MA)=B:EY(MA+EX)=B:D=0:POKE
64*B+15360,32
350 C=RND(9)
400 IFC<6THEND=D+1:POKE64*B+D+15360,32:IFD=62THENNEXTMA ELSE350
450 IFMA>5THEN900
500 IFC<8ANDB>1THENB=B-1:POKE64*B+D+15360,32:GOTO350
550 IFC<10 ANDB<13 THENB=B+1:POKE64*B+D+15360,32:GOTO350
650 GOTO350
```

Wait for a keypress to start the game, and clear the last lines of the screen.

```
900 PRINT@ 960,"PRESS ANY KEY TO BEGIN  ";
920 G$=INKEY$:IFG$=""THEN920
930 FORT=16320TO16383:POKET,32:NEXT
```

Beginning of repeating program loop.

```
950 PRINT@960,"TIME:";:PRINT@1010,"POINTS:";
```

Check to see if turn limit has expired. Also set up a FOR-NEXT loop to move the player a maximum number of times as determined by his advantage.

```
1000 FORTU=1TOAD:TR=TR+1:IFTR=TLTHEN20100
```

Get a keypress, using either INKEY$ or a PEEK to keyboard memory. The PEEK is used to give a repeating-key function, and the INKEY$ senses if a key was pressed and released before the PEEK.

```
1025 ZA$=INKEY$:IFZA$><""THEN X=ASC(ZA$):FORSC=1TO4:IFX=IN(SC) T
HEN A=IK(SC):SC=4:NEXT:GOTO1075:ELSE NEXT
1050 A=PEEK(14444):IFA=0THENNEXT:GOTO2000
```

continued on next page

continued from previous page

Determine the current screen position of the player (CL), and then convert the keypress information from above to a directional value (1-4).

```
1075 CL=YY*64+XX+15360:FORSC=1TO4:IFA=IK(SC) THEN X=SC:SC=4:NEXT
:ELSE NEXT
```

Branch to appropriate section of program according to direction of movement.

```
1080 ON X GOTO 1400,1100,1200,1300
```

Forward movement.

```
1100 G=PEEK(CL-1):IFXX>0ANDG=TT THEN POKECL,TT:XX=XX-1:POKECL-1,
YC:NEXT:GOTO2000:ELSEIFXX<=0THEN NEXT:GOTO2000
1150 IFG<>191 THENPOKECL,TT:XX=XX-1:GOTO10000
1175 NEXT:GOTO2000
```

Upward movement.

```
1200 G=PEEK(CL-64):IFYY>0 AND G=TT THEN POKECL,TT:YY=YY-1:POKECL
-64,YC:NEXT:GOTO2000:ELSE IFYY<=0 THEN NEXT:GOTO2000
1250 IFG<>191 THENPOKECL,TT:YY=YY-1:GOTO10000
1275 NEXT:GOTO2000
```

Downward movement.

```
1300 G=PEEK(CL+64):IFG=TT AND YY<14 THEN POKECL,TT:YY=YY+1:POKEC
L+64,YC:NEXT:GOTO2000:ELSE IF YY>=14 THENNEXT:GOTO2000
1350 IFG><191 THENTU=6:POKECL,TT:YY=YY+1:GOTO10000
1375 NEXT:GOTO2000
```

Reverse movement.

```
1400 G=PEEK(CL+1):IFG=TT AND XX<63 THEN POKECL,TT:XX=XX+1:POKECL
+1,YC:NEXT:GOTO2000:ELSE IF XX>=63 THEN NEXT:GOTO2000
1450 IFG><191 THENTU=6:POKECL,TT:XX=XX+1:GOTO10000
1500 NEXT TU
```

Print the number of turns elapsed, and calculate whether player has been killing bugs fast enough. If not, then branch to the end routine; otherwise, set up loop to move the bugs on the screen.

```
2000 PRINT@967,TR;:IFPO/TR<SL/2.5THEN20000:ELSEFOREN=1TO5+EX
```

Determine the screen location of the current bug, and store in variable EL.

```
2050 EL=EY(EN)*64+EX(EN)+15360
```

Test to see if bug can move forward and, if so, test to see that bug has not reached right side of cave.

```
2100 IFPEEK(EL+1)=TT THENPOKEEL,TT:EX(EN)=EX(EN)+1:E2(EN)=1:POKE
EL+1,D(E2(EN)):IFEX(EN)=63 THEN20000 ELSE NEXT:GOTO1000
```

If bug cannot go forward, then move it in same direction as last turn. If bug move forward last turn, then randomly choose between up and down.

```
2125 ONE2(EN) GOTO2150,2200,2300
2150 Q=RND(2):ON Q  GOTO2200,2300
```

Move the bug up, if possible; if not, change direction to down for next turn.

```
2200 IFPEEK(EL-64)=TT THEN POKEEL,TT:EY(EN)=EY(EN)-1:E2(EN)=2:PO
KEEL-64,D(E2(EN)):NEXT:GOTO1000
2250 E2(EN)=3:NEXT:GOTO1000
```

Move bug down, if possible; if not, change direction to up for next turn.

```
2300 IFPEEK(EL+64)=TT THENPOKEEL,TT:EY(EN)=EY(EN)+1:E2(EN)=3:POK
EEL+64,D(E2(EN)):NEXT:GOTO1000
2350 E2(EN)=2:NEXT:GOTO1000
```

Routine to "kill" a bug.

```
10000 H=YY*64+XX+15360:FORE=32TO60:POKEH,E:NEXT:DD=DD+1
```

Compare screen locations to find which bug was killed.

```
10050 FORE=1TO5+EX:IFEY(E)*64+EX(E)+15360=H THEN10150
10100 NEXT E
```

Add points, if any were earned.

```
10150 IFEX(E)>40THEN10200
10175 PO=PO+(40-EX(E)):PRINT@1017,USING "######";PO;
10177 IFPT<>0 AND PO>PTTHEN20100
10180 IF(6-AD)*200<PO THEN AD=AD-1:IFAD=1THEN AD=2
```

Reset all pointers for the bug, positioning it on the left side.

```
10200 EY(E)=E1(E):EX(E)=0:E2(E)=1
```

Return to the beginning of the main loop.

```
10300 GOTO2000
```

Inform the player that the game has ended, and tell him how well he did.

```
20000 PRINT@960, "Y O U   H A V E   B E E N   O V E R R U N ! !"
;
20050 SC=YY*64+XX+15360:POKESC,91:POKESC-2,92:POKESC+64,93:POKES
C-64,94:POKESC+1,91:POKESC-66,92
20100 FORE=1TO1400:NEXT:PRINT@960,"KILLED";:PRINTUSING "####";DD
;:PRINT" IN";:PRINTUSING "#####";TR;:PRINT" TURNS.";:PRINTUSING
"#####";PO;:PRINT" POINTS. WANT TO PLAY AGAIN?     ";
```

End game if player does not want to play again.

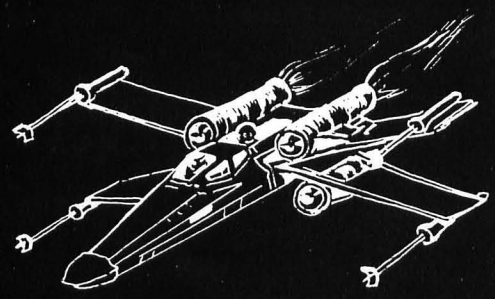
```
20150 G$=INKEY$:IFG$<>"N" AND G$<>"Y"THEN20150
20200 IFLEFT$(G$,1)="N"THEN CLS:PRINT "HAVE A GOOD DAY!":END
```

Re-initialize a new game.

```
20300 PRINT@960,"WOULD YOU LIKE TO USE THE SAME BOARD?
          ";
20350 G$=INKEY$:IFG$<>"Y" AND G$<>"N" THEN20350
20400 IFG$="N"THENCLEAR:CLS:GOTO100
20425 POKESC,32:POKESC-2,32:POKESC+64,32:POKESC-64,32:POKESC+1,3
2:POKESC-66,32
20450 FORE=15360+960TO15360+960+63:POKEE,32:NEXT
20475 POKEYY*64+XX+15360,32:YY=0:XX=63:POKEYY*64+XX+15360,60:AD=
SAD-1:PO=0:TR=0:DD=0
20500 FOREN=1TO5+EX:POKEEY(EN)*64+EX(EN)+15360,32:EY(EN)=E1(EN):
EX(EN)=0:E2(EN)=1:NEXT
20550 PRINT@960,"SKILL LEVEL (1-5)?                              ";
20600 G$=INKEY$:IFVAL(G$)<1 OR VAL(G$)>5THEN20600
20700 PRINT@960,"PRESS ANY KEY TO BEGIN";
20750 IFINKEY$=""THEN20750
20850 SL=VAL(G$):EX=SL:SAD=(6-SL)+5:AD=SAD-1:PO=150:PRINT@960,"T
IME:                    ";:GOTO950
```

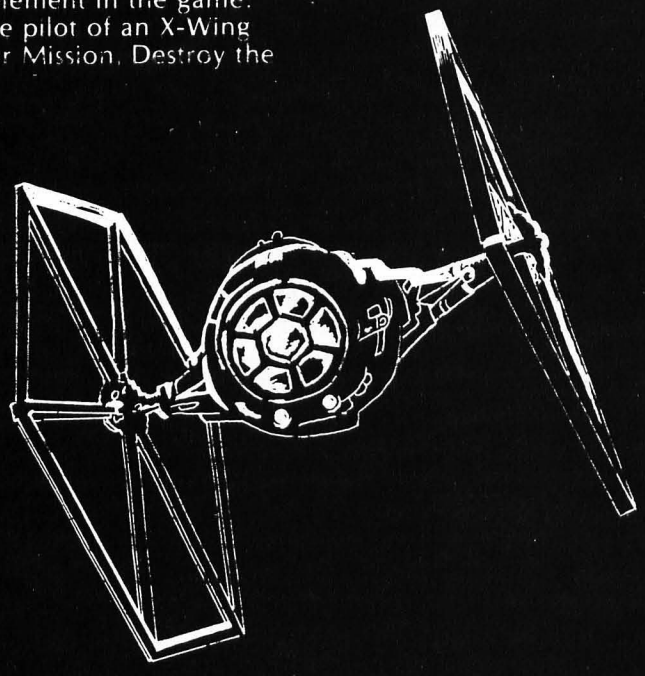
X-Wing II

by Chris Freund



For the thousands who have enjoyed X-Wing Fighter, X-Wing II presents a totally new element in the game:

You are the pilot of an X-Wing fighter...Your Mission, Destroy the Death Star!



Where X-Wing I left Death Star looming on the screen, X-Wing II lets you guide your fighter into the trench, find the exhaust port, aim and fire—all the while avoiding enemy fighters. Excellent graphics, 12 levels of play, and extensive INKEY$ commands make this one of our most exciting "real-time" games.

16K, S-80, Level II, Cassette $9.95

SoftSide Selections

6 South Street Milford NH 03055

For Orders Only 603-673-0585

Nine Games for Preschool Children

by George Blank

Even pre-schoolers deserve a shot at the wonders of microcomputing. With these nine games, they not only will have a chance to tickle the keyboard, but learn letters and numbers to boot. And if that isn't enough, they'll have a good time doing so. What more could a parent ask for? Here are education and entertainment for the very young in a single package!

S-80, Level II, 16K, Cassette $9.95

SoftSide Selections

6 South Street Milford NH 03055

For Orders Only 603-673-0585

Mixed Text and Graphics in HGR2

And You Thought It Couldn't Be Done

by Jon Voskuil

Verbose Applesoft programmers of the world, take heart! Now you can write a full 14K of code and still use mixed text and Hi-Res graphics.

About six months before I had enough money saved to buy my Apple II, I went to the dealer and bought all the Apple manuals. Abandoning my wife and children, I began to teach myself BASIC. The first real program I wrote, still some time before I actually got the machine, was an elaborate Hi-Res hangman game. To try to get my wife interested in my new hobby, I cajoled her into drawing a detailed, gruesome figure on graph paper, and I then painstakingly translated that into scores of HPLOTs.

HOW THIS PROGRAM HAPPENED

When the day finally arrived, and I plunked down my VISA (I still didn't have enough money) and brought home my Apple; one of the first things I did was to type in that hangman program. And upon running it, what should I see on the Hi-Res screen but hundreds of little flickering specks which meant (I soon learned) that my program was spilling over into the area of memory which holds the Hi-Res graphics display. There just wasn't enough room to store my program between address 2052 where program memory begins, and address 8192 where the first page of Hi-Res graphics begins. And after much time spent at the keyboard with the Applesoft Reference Manual opened to Appendix D ("Space Savers"), the seriousness of my plight began to dawn on me: There was no way to shorten it enough to fit into the available space.

Ah, but what if I could just put the program above the Hi-Res memory area rather than below it? With 32K of RAM, there was almost three times as much memory up there as down here. In my naivete I wondered whether those LOMEM and HIMEM commands that I remembered reading about could do the trick. If so, I couldn't find the magic formula. Was there any way, I asked the salesman at the computer store the next time I dropped in, to get that program relocated above the Hi-Res memory area? Sorry — no way that he knew of. Another dead end.

If only there were a way to use the SECOND Hi-Res graphics screen — it's located 8K higher in memory, and would free up plenty of space for my program. Plotting the picture would be no problem, of course; all the HPLOTs would work in HGR2 just as they would in HGR. The problem was the text: The HGR2 command gives you a full screen of graphics rather than HGR's mixed graphics-plus-four-lines-of-text format. Further browsing in the Reference Manual turned up the fact that it IS possible to have mixed text and graphics in HGR2, using a POKE to flip an internal software switch from full-screen to mixed-screen mode: POKE -16301,0.

The answer to all my problems? Hardly. I discovered that the four lines of text which are displayed in the mixed-screen HGR2 mode are from "text page 2" rather than the normal page of text. Indeed, when I looked at the mixed screen in HGR2, the stuff on the bottom four lines was all garbage, and there was no way I could PRINT to it. The only information I could find in the manual about this mysterious page 2 of text was the foreboding statement that it "is not easily accessible to the user." And after some further teeth-gnashing and fist-pounding, I finally cast off the Hi-Res hangman albatross forever. (I ended up writing a Lo-Res version.)

Since then it has happened several other times, that I wished I had more program memory available below HGR, or that I could successfully mix text and graphics in HGR2. The other day, while reading through the Reference Manual, I came across that foreboding statement again, and took it as a challenge to find out just HOW "not easily accessible" page 2 of text really is. Some investigation revealed that, in fact, page 2 is where the first part of an Applesoft program is stored — which seems to make it inaccessible indeed for use in displaying text! A program that tried to write anything into that part of memory would be committing suicide, altering its own code in unpredictable ways and crashing into oblivion.

Unless, that is, one knew exactly what memory locations would be needed to display text on those four bottom lines of the screen, and could write the program in such a way that it didn't matter what was in those locations. After some experimentation I located the memory addresses of those four lines at 2640-2679, 2768-2807, 2896-2935, and 3024-3063. And reasoning that REM statements would be a logical choice to occupy the areas of memory that would be in flux, some further work yielded the magic formula for success.

USING THE PROGRAM

The resulting program, when placed at the beginning of your own program, allows you to display four lines of text at the bottom of the screen while using HGR2. Admittedly, it's not quite as convenient as PRINTing, but neither is it difficult to use. Lines 0 through 7 must begin your program, the variables in line 9 must be initialized as shown, and the subroutines beginning at lines 20 and 40 must be included somewhere. I recommend keeping them where they are, at the beginning of the program, for the sake of execution speed.

Assuming that you simply transplant lines 0-44 as listed, your program will then begin at line 100, in place of the demonstration routine that I stuck in. When the time comes to make use of the mixed text and graphics mode, use HGR2 followed by a POKE -16301,0. The first time this is done, the whole screen will be blank, since the GOSUB 40 in line 10 cleared the four text lines to spaces. (This subroutine can be called whenever needed throughout your program, to quickly clear all four lines.) Then, whenever you want to display text on the screen, follow this procedure:

1. Set QQ$ equal to the text you want to print — by using an assignment statement such as QQ$ = "HELLO", or by READing in a string as in the demonstration program, or by any other method.

2. Set VTAAB and HTAAB equal to the line number (1-4) and column number (1-40) where you want to start printing.
3. GOSUB 20.

(Step 2 doesn't have to be repeated each time — only when you want to change the values.)

SOME PROGRAMMING NOTES

Lines 0-7 of the listed program occupy all of text page 2, and they MUST be the first eight lines of the program. They are simply space fillers which exactly fill the required amount of space. The 160 memory cells for the four text lines are occupied by the last 40 periods in lines 4-7. After these locations have been used by your program to display text, LISTing them will show that they have been altered: The periods have been replaced with garbage, generally consisting of BASIC keywords. Don't let the apparent increase in line length worry you: The lines stored in memory are still exactly the same length; they just LIST that way because Applesoft interprets the numbers it finds as tokenized keywords. To avoid saving that garbage with the program (not that it will do any harm if you do), you must SAVE it before you RUN it.

Note that when you type in lines 0-7, you must begin typing the periods immediately after the word "REM", with no intervening space. (Applesoft adds that space when it LISTs the program.) Each of the REMs must have exactly 170 periods or other characters. Using spaces as characters is fine, as long as you keep count of them along with all the others; avoid tacking extra, unseen ones onto the end of the line before hitting RETURN.

The printing subroutine at line 20 works by taking apart the text string and POKEing it, one character at a time, into the proper memory locations. Line 20 checks for proper values of VTAAB and HTAAB, and line 21 verifies that the printing will not overflow past the end of text page 2. If there is any problem, nothing is printed and control is returned to the calling line. These error checks are very important, since an error of one byte may destroy your program beyond recovery. Line 24 then POKEs in each character, adding 128 to the ASCII value to make it display as a normal character. After each character is POKEd, line 26 increments the memory location and, if the end of the line has been reached, jumps to the beginning address of the next one.

The subroutine at line 40 merely pokes a value of 160 (the ASCII value of a space, plus 128) into each of the 160 memory locations used for text display. This clears the lines several times faster than the alternative of using subroutine 20 to print a string of 160 spaces. If you want to clear just a portion of the four text lines, you can either print spaces using subroutine 20, revise subroutine 40 so that it will blank only the lines you specify, or insert other subroutines for blanking specific lines.

Now then, where DID I put that old Hi-Res hangman program...?

```
0  REM ..MIXED GRAPHICS/TEXT.....
     .....ROUTINE FOR HGR2.......
     ............................
     ..BY JON R. VOSKUIL...6/81..
     .............
1  REM ..THESE REM LINES 0-7.....
     ....MUST NOT BE ALTERED.....
     .OR THE PROGRAM WILL QUITE..
     .LITERALLY SELF-DESTRUCT!!!.
     .............
2  REM .....LINES 4-7 WILL.......
     ....ALTER THEMSELVES IN.....
     ....STRANGE WAYS DURING.....
     .....PROGRAM EXECUTION;.....
     .............
3  REM .......THEREFORE..........
     ....THIS PROGRAM SHOULD.....
     .........BE SAVED...........
     ......BEFORE RUNNING........
     .............
4  REM ..........................
     ............................
     ............................
     ............................
     .............
5  REM ..........................
     ............................
     ............................
     ............................
     .............
6  REM ..........................
     ............................
     ............................
     ............................
     .............
7  REM ..........................
     ............................
     ............................
     ............................
     .............
9 WUN = 1:H28 = 128:M1 = 2680:M2 =
     2808:M3 = 2936:H60 = 160
10  GOSUB 40: GOTO 100
19  REM

POKE TEXT ONTO SCREEN

20 VTAAB =  INT (VTAAB):HTAAB =  INT
     (HTAAB): IF VTAAB < 1 OR VTA
     AB > 4 OR HTAAB < 1 OR HTAAB
      > 40 THEN 30
22 MQ = 2511 + VTAAB * 128 + HTAA
     B:LQ =  LEN (QQ$): IF MQ + L
     Q + ((LQ > 40) + (LQ > 80) +
     (LQ > 120) + (LQ > 160)) * 8
     8 > 3064 THEN 30
24  FOR QQ = 1 TO LQ:CQ =  ASC ( MID$
     (QQ$,QQ,WUN)) + H28: POKE MQ
     ,CQ
26 MQ = MQ + WUN: IF MQ = M1 OR M
     Q = M2 OR MQ = M3 THEN MQ =
     MQ + 88
28  NEXT
30  RETURN
39  REM

CLEAR TEXT LINES

40 MQ = 2640: FOR Q = 1 TO 4: FOR
     QQ = 0 TO 39
42  POKE MQ + QQ,H60
44  NEXT :MQ = MQ + H28: NEXT : RETURN

99  REM

BEGIN MAIN PROGRAM HERE

100  HGR2 : POKE  - 16301,0
120  FOR I = 1 TO 4: FOR J = 1 TO
     20
130 HC =  RND (1) * 8:X =  INT ( RND
     (1) * 279):Y =  INT ( RND (1
     ) * 159)
140  HCOLOR= HC: HPLOT X,Y TO 278
      - X,Y TO 278 - X,158 - Y TO
     X,158 - Y TO X,Y
150  NEXT J
160  READ QQ$
170 VTAAB = I
180 HTAAB = 20 -  LEN (QQ$) / 2: IF
     HTAAB < 1 THEN HTAAB = 1
210  GOSUB 20
220  NEXT I
300  FOR Z = 1 TO 1000: NEXT Z
310  GOSUB 40
320  FOR Z = 1 TO 1000: NEXT Z
330  TEXT : HOME : END
1000  DATA   THIS IS A SIMPLE DEM
     ONSTRATION,OF MIXED GRAPHICS
      AND TEXT,ON HIGH-RES SCREEN
      NUMBER TWO,(BET YOU THOUGHT
      IT COULDN'T BE DONE!!)
```

THE COMPLETE GRAPHICS SYSTEM

by Mark Pelczarski

A user-oriented graphics system for the Apple II

- A Drawing module allows you to draw on the Hi-Res screen using paddles or a joystick. Options include an automatic filling routine with over 100 blended colors, a "paintbrush" mode that turns your cursor into one of nine different size and shape brushes (you can also define your own), and a shape mode that lets you rotate, scale, and plot shapes from a shape table. You can also easily draw lines, circles, and ellipses. Drawings may be saved for use by other programs.

- A Text module lets you put text at any x,y location on the Hi-Res screen (not just at specified rows and columns). A standard upper/lower case character set is provided, along with a large set twice as tall and wide. The large font may be used with most of the 100 fill colors. Typing may be set to overlay the background, reverse the background, or erase the background. A character editor allows you to create and edit your own large and small fonts. Creating your own graphic character sets gives you a fast, colorful alternative to using shape tables for animation.

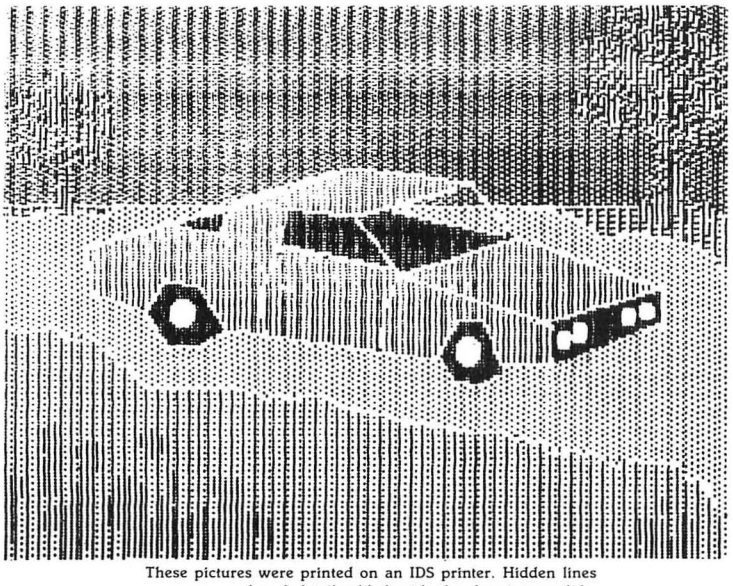
- The 3-D Graphics module lets you create, view, and manipulate 3-D objects. Several color figures may be in memory and viewed simultaneously, yet each figure is capable of being manipulated individually. An added module lets you draw 2-dimensional "panels" on the screen (for example, the surfaces of a house). The 3-D module will then allow you to assemble these panels into a 3-D object--without having to use a single X-Y-Z coordinate! Any figure may be rotated, scaled, moved, and distorted (scaled in one direction) with machine language speed. Several editing functions allow you to move points and lines, change colors, and enter, change, and delete coordinates. Any particular view of the figures may be saved as a picture to be used with the other modules or your programs.

- The Shape Table module allows you to create and edit Apple shape tables for use in the drawing module or your programs. Shapes may be created and edited either by keystrokes or by drawing with the paddles.

- A Shrink utility allows you to shrink your pictures to ¼th size, so that four may be displayed at a time. By re-using "Shrink", you may also reduce pictures further.

- Programmers' instructions are included for using pictures, shape tables, 100 color fill routine, and text writer in other programs, along with the file structure information for 3-D files.

Price: $59.95, requires 48K and Applesoft firmware, or the language system.
Machine language, DOS 3.3.**(#47-256002)**



These pictures were printed on an IDS printer. Hidden lines were removed and detail added with the drawing module.

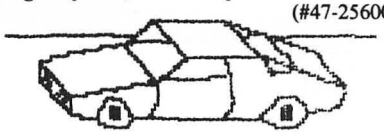
MAGIC PAINTBRUSH 4.0

Graphics and Games for the Apple II

Magic Paintbrush 4.0 contains a simplified version of the Drawing module of the **Complete Graphics System**, along with the Shape module, and several games that use graphics created with these two modules. The drawing program lets you draw on the screen using the paddles, and also uses the "paintbrush" mode from the larger package. The games included are **Color Invaders, Slot Machine, Collision, Sailboat Race,** and **Dogfight.** This is a good graphics package for beginners. It requires 32K and Applesoft firmware or the language system, and is priced at $29.95. **(#47-256001)**

THE 100-COLOR DRAWING SYSTEM

The **100-color Drawing System** is ideal for generating any type of screen image and for creating figures for use in animation. It contains the Drawing, Text, Shape, and Shrink modules from the **Complete Graphics System,** and by far offers the most capability of any graphics drawing software available. It requires 48K and Applesoft firmware, or the language system, and is priced at $32.95. **(#47-256003)**

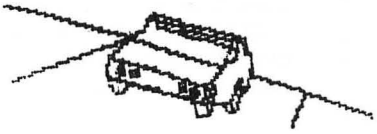


THE 3-D DRAWING SYSTEM

The **3-D Drawing System** contains the 3-D modules from the **Complete Graphics System.** It is the only 3-D software available that emphasizes the design aspect of 3-D graphics. You may enter figures by using their coordinates, or you may simply draw their surfaces on the screen. Once a figure is created, numerous options let you alter, manipulate, and combine figures. The **100-Color Drawing System** and the **3-D Drawing System** together give the equivalent of the **Complete Graphics System.** 48K and Applesoft firmware, or the language system, are required, and the price is $32.95. **(#47-256004)**

"The three-dimensional utilities verge on the phenomenal... the entire 3-D set is graced with easy input routines."

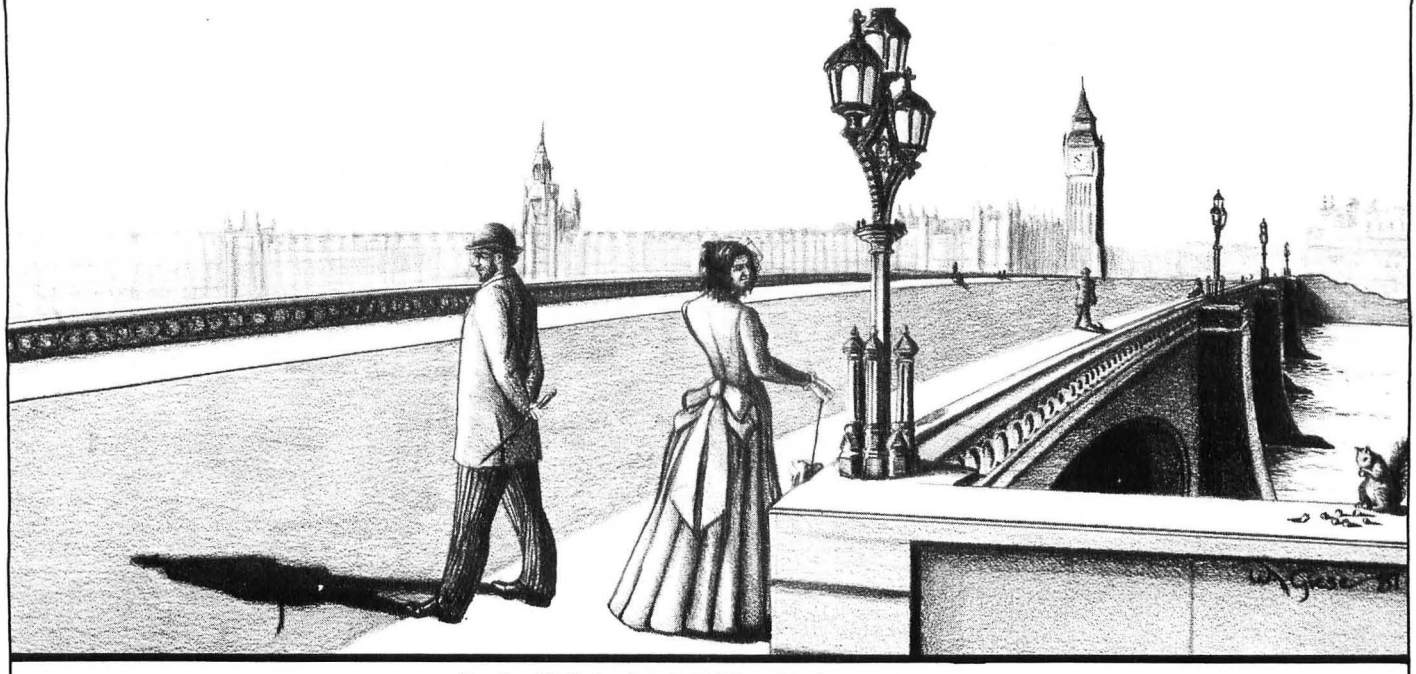
--David Lubar in **Creative Computing**



TERMS: Prices and specifications are subject to change. **TSE HARDSIDE** accepts VISA & MASTERCARD, Certified checks and Money Orders. Personal checks accepted (takes 3 weeks to clear). **TSE HARDSIDE pays all shipping charges (within the 48 states) on all PREPAID orders over $100.00.** On all orders under $100 a $2.50 handling charge must be added. COD orders accepted (orders over $250 require 25%deposit), there is a $5.00 handling charge. UPS Blue Label and Air Freight available at extra cost.

TSE·HARDSIDE
6 South St., Milford, NH 03055 (603) 673-5144
TOLL FREE OUT-OF-STATE 1-800-258-1790

ADVENTURE OF THE MONTH CLUB



JACK THE RIPPER

SEPTEMBER ADVENTURE OF THE MONTH

Jack the Ripper is running rampant in London and you must stop him! Scotland Yard demands you take action and the only answer is to set yourself up as a decoy. Be careful how you plan your costume, or dear Jack will laugh hysterically and leave you in the dust!

What is Adventure of the Month?

Everybody likes Adventures — they're challenging and entertaining every time you play. But too often, preprogrammed cassettes and disks cost upwards of $35, a price the manufacturer must charge to defray promotional and packaging costs.

On the other hand, you can enter Adventures yourself, but when you do, you type away all the surprises. As a result, the game loses some of its challenge.

At **SoftSide**, we've found a way to beat the high cost of Adventuring without having to miss out on any of the fun. We're offering no-frill Adventures — high quality Adventures — on cassette or disk at an almost unheard-of price: **$5 for cassette, $8 on disk.**

We save you money by only advertising this offer to **SoftSide** readers (you won't see us anywhere else) and by foregoing fancy packaging and documentation — you'll get the software and only the software, but we believe it's as good as the $50 packages.

You'll save even more by joining the Adventure of the Month Club.

Here's how it works: **SoftSide**'s editorial department will select an original BASIC language Adventure each month and make it available to you on a subscription basis:

6 months on cassette: just $27
6 months on disk: just $45

Every month we'll tell you about the Adventure you'll be getting in **SoftSide** Magazine. To order, use the convenient order form in this issue — fill it out and send, with payment to:

Minimum System Requirements:

Apple — 24K Tape — 32K Disk
Atari — 32K Tape — 40K Disk
S-80 — 16K Tape — 32K Disk

Adventure of the Month Club
Department 681
6 South Street
Milford, NH 03055

SoftSide Selections Order Form

S-80 PROGRAMS

- ☐ Android Nim, 16K Tape, Machine Language ..... ~~$14.95~~ $13.50 *
- ☐ APL 80, 16K Tape (Mod. I only) $14.95
- ☐ APL 80, 32K Disk (Mod. I only) $39.95
- ☐ Bee Wary, 16K Tape ..... ~~$14.95~~ $13.50 *
- ☐ Duel-N-Droids, 16K Tape ..... ~~$14.95~~ $13.50 *
- ☐ Duel-N-Droids, 32K Disk ..... ~~$20.95~~ $19.00 *
- ☐ Dynamic Database, 32K Disk (Mod. I only) $39.95
- ☐ End Zone II, 16K Tape $14.95
- ☐ End Zone II, 32K Disk $15.95
- ☐ File Manager 80, 32K Disk (Mod. I only) ..... ~~$49.95~~ $39.95 *
- ☐ Floppy Disk Diagnostic, 16K to 48K Disk (Mod. I only) ..... ~~$24.95~~ $22.50 *
- ☐ Galactic Empire, 16K Tape ..... ~~$14.95~~ $13.50 *
- ☐ Galactic Trader, 16K Tape ..... ~~$14.95~~ $13.50 *
- ☐ Galactic Revolution, 16K Tape ..... ~~$14.95~~ $13.50 *
- ☐ Galactic Trilogy (Incl. All Three from above) 32K Disk ..... ~~$39.95~~ $35.95 *
- ☐ Kriegspiel II, 16K Tape (Mod. I only) $14.95
- ☐ Level I in Level II, 16K Tape $15.00
- ☐ Masters Golf, 16K Tape $9.95
- ☐ Nine Games for Preschool Children, 16K Tape .. $9.95
- ☐ Pascal 80, 32K Disk (Mod. I only) $99.95
- ☐ Snake Eggs, 16K Tape ..... ~~$14.95~~ $13.50 *
- ☐ Space Battles, 16K Tape $14.95
- ☐ Space Battles, 32K Disk $19.95
- ☐ Star Trek III.5, 16K Tape $14.95
- ☐ Star Trek III.5, 32K Disk $14.95
- ☐ Stat Pal, 32K & 48K Versions on one Disk $29.95
- ☐ Tiny Comp, 16K Tape (Mod. I only) $19.95
- ☐ Tiny Comp, 32K Disk (Mod. I only) $24.95
- ☐ Typing Tutor, 16K Tape ..... ~~$19.95~~ $15.95 *
- ☐ Up Periscope, 16K Tape (Mod. I only) $14.95
- ☐ Warpath, 16K Tape (Mod. I only) $14.95
- ☐ World Series Baseball, 16K Tape $9.95
- ☐ X-Wing II, 16K Tape $9.95

APPLE PROGRAMS

- ☐ Galactic Empire, 48K Disk ..... ~~$24.95~~ $22.50 *
- ☐ Galactic Trader, 48K Disk ..... ~~$24.95~~ $22.50 *
- ☐ Galactic Revolution, 48K Disk ..... ~~$24.95~~ $22.50 *
- ☐ Hayden Applesoft Compiler, 48K Disk ..... ~~$200.00~~ $175.00 *
- ☐ RobotWar, 48K Disk ..... ~~$39.95~~ $34.95 *
- ☐ Time Lord, 48K Disk $29.95
- ☐ World Series Baseball, 16K Tape $9.95
- ☐ World Series Baseball, 32K Disk $14.95

ATARI PROGRAMS

- ☐ Masters Golf, 8K Tape $9.95
- ☐ World Series Baseball, 16K Tape $9.95

OTHER

- ☐ The Basic Handbook ..... ~~$19.95~~ $17.95 *
- ☐ Pathways Through the ROM (Mod. I only) $19.95
- ☐ Vinyl Binder (small) $3.95
- ☐ Vinyl Binder (large) $7.95

ADVENTURE OF THE MONTH

- ☐ Jack the Ripper Adventure Tape $5.00
  for the _____ computer
- ☐ Jack the Ripper Adventure Disk $8.00
  for the _____ computer

ADVENTURE OF THE MONTH CLUB

- ☐ 6 month Cassette subscription $27.00
  for the _____ computer
- ☐ 6 month Disk subscription $45.00
  for the _____ computer

SOFTSIDE SUBSCRIPTIONS

- ☐ Magazine only (12 issues) $24.00

With the SoftSide disk or cassette subscription you get not only the magazine, but all the programs in it delivered on your choice of media.

- ☐ Magazine and cassette (12 issues) $75.00
  for the _____ computer
- ☐ Magazine and disk (12 issues) $125.00
  for the _____ computer
- ☐ Magazine and disk (one trial issue — September issue will be sent) ..... ~~$19.95~~ $12.95 *
  for the _____ computer

*offer expires October 15.

Please use facing bind-in card to order. If card is missing, be sure to add $1.50 handling charge to your order total.

Canada/Mexico Orders
No C.O.D. to Canada or Mexico. The prefered method of payment is by Master Card or Visa. NO PERSONAL OR COMPANY CHECKS. A bank check is acceptable if it has been preprinted for payment in U.S. dollars. The handling charge on all Canadian or Mexican orders is $5.00 PLUS actual shipping charges.

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 Master Card. All shipping and duty charges are the customer's responsibility. All overseas orders are subject to a $10.00 handling charge PLUS actual shipping charges.

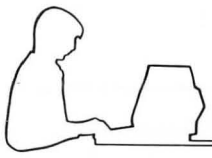
Guarantee
All software is guaranteed to load and run. If you experience difficulties with the product within 30 days, the tape or disk may be returned. Call (603) 673-0585 or 673-0586 for a Return Authorization Number. Any returns without a Return Authorization Number clearly marked on the outside WILL BE REFUSED. Send your properly protected disk or tape to the attention of Customer Service Representative with a note including your name and address.

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 of operation of such software.

Prices:

Prices are subject to change without notice. We are not responsible for typographical errors.

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585



Orienteering at Jacque's Coulee

Original S-80 version by Michael A. O'Keefe.

Apple translation by Robert White (translation contest winner).

"Orienteering at Jacque's Coulee" is an Apple simulation written in Applesoft and requiring 24K of memory.

"Orienteering" first appeared in **SoftSide** in its S-80 version in May, 1981. This Apple translation retains all the features of the original, makes several changes in the formatting of output to fit the Apple's screen display, and adds one enhancement: a "competition orienteering" option. The following articles and illustrations will give you all the information you need to step out with your Apple into the world of computer orienteering.

PART I: AN INTRODUCTION TO ORIENTEERING

"Orienteering" is the sport of using a map and compass to find your way through unfamiliar terrain. The skills developed in orienteering can be directly applied to backpacking, hiking mountaineering, birdwatching, and other outdoor activities.

BASIC SKILLS

1. Using a Compass

North, south, east, and west are the four cardinal points of the compass. When you orienteer, however, you will often want to go in some direction other than just these four. This is why a modern compass is divided into 360 different directions, one direction for each degree of a circle. Each of these directions is called a bearing, and is identified by the angle formed clockwise from north to the direction of travel. The bearing of east is 90 degrees; south is 180 degrees; west is 270 degrees; and north has a bearing of 0 (or 360) degrees.

On the map a normal protractor can be used to measure the bearing of any direction of travel. Once this has been determined, a magnetic compass is used to determine this bearing in real life. A special device called a protractor compass, which does both of these things, has been developed for orienteering. You will find a version of such a compass, for use with the computer simulation, on page 70.

2. Reading a Map

If you remove all the spaghetti-like contour lines which tend to clutter up all topographic maps, what is left is generally easy to understand. Such a map is like a photograph taken from high up. If you can read a highway map, you can read these portions of a topographic map.

Contour lines are more difficult to get used to. They do not represent anything you can see, yet they are probably the most important part of the map. Contour lines show the shape of hills, the steepness of slopes, and the presence of all sorts of holes, bumps, gullies, and imperfections in the surface of the ground.

A contour line is defined as a set of all points which are at a given altitude — say 800 feet above sea level. Topographic maps contain many contour lines, each at an altitude a certain interval from the text. A map could have contour lines for 800, 810, 820, 830 and so on; such a map would have a contour interval of 10 feet. The following rules should help in understanding contour lines.

1. A hill summit is circled by contour lines.

2. If you travel across one contour line to another, you are on a slope going either uphill or downhill.

3. The closer together contour lines are, the steeper is the slope they represent.

4. If a contour line has a "v" in it and the "v" points uphill, the "v" represents a depression or gully on the side of the hill.

5. If a contour line has a "u" in it and the "u" points downhill, the "u" represents a bump or ridge on the side of the hill.

Any map is a miniature representation of an area. Just HOW miniature is indicated by the map's scale. A scale of 1:15000 means that one unit of measure on the map is the same as 15000 of the same unit of measure in real life. Since converting inches to yards or feet is difficult, orienteers measure in metric. With a scale of 1:15000, then, one centimeter on the map equals 150 meters in real life.

ORIENTEERING TECHNIQUES

In orienteering one tries to choose the fastest route between two points. Often the shortest route is not the fastest. The ability to select which is the fastest is the essence of orienteering. Common sense and experience are essential in this route selection, but there is a basic core of techniques and rules which can be of great help. A summary of these follows.

1. Avoid depending on following a precise compass bearing. Even if you go slowly and carefully, you'll almost always wander a few degrees off the course you want to travel; and if you try to hurry at all, you could end up 20 to 30 degrees off your course. If you need to follow a precise bearing, try to do so for less than a couple of hundred meters; a few degrees of error is acceptable over such short distances.

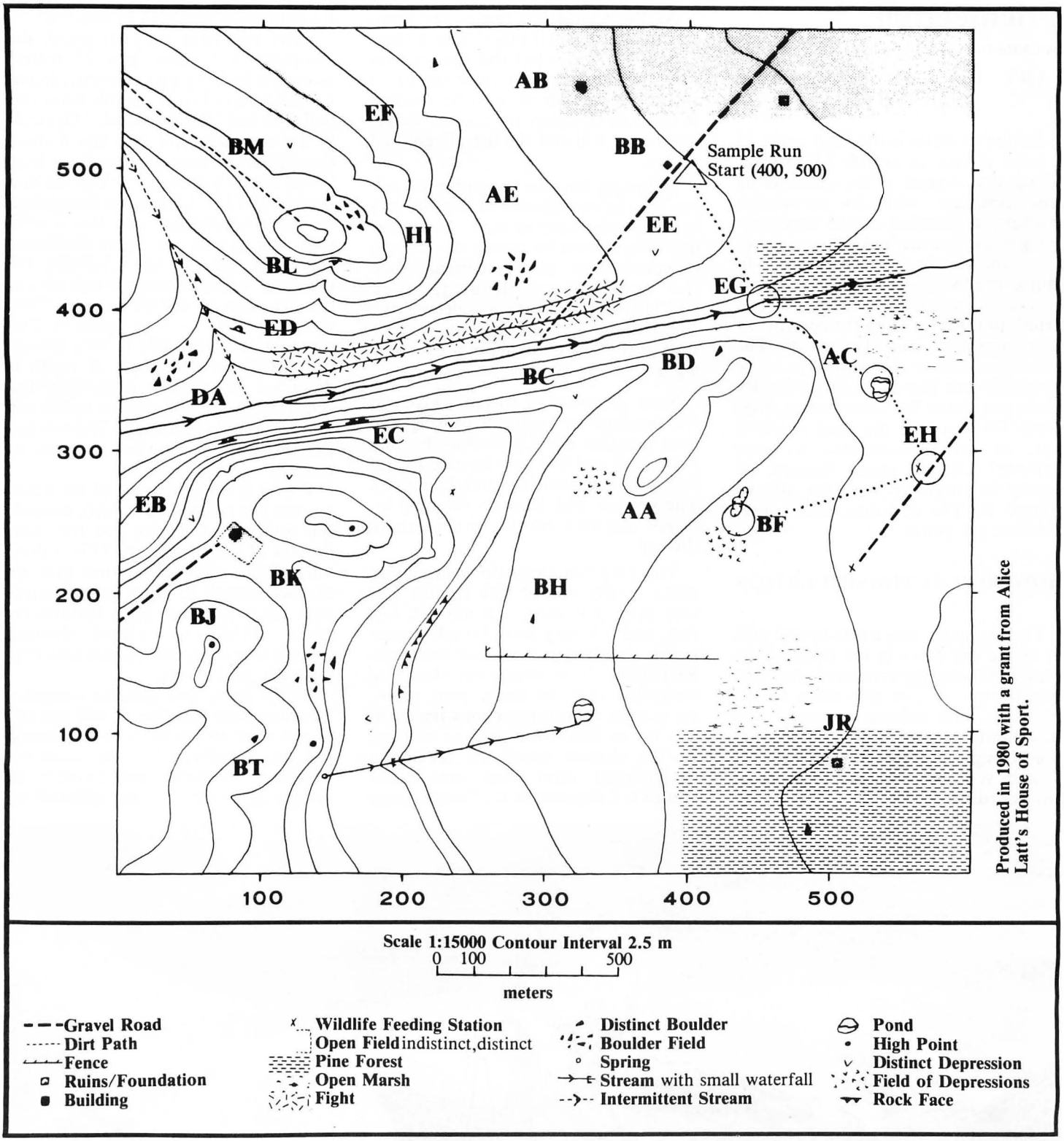
2. Look for an easy-to-find feature near your goal. If there's a pond, crossroad, or some other easily found feature within a couple of hundred meters of your goal, go fast to the easy feature and then slowly and carefully to your goal. This is called using an attack point and is a very useful technique.

If your goal is on a path or road, you can take advantage of this by travelling fast to a portion of the path to one side of your goal, then following the path to the goal. It is important, of course, that you travel so that when you reach the path you're sure which way to follow it to get where you're going. This technique is called aiming off and is also very useful.

3. Recognize when you've made an error. Since you WILL make mistakes, it's valuable to try to detect them as soon as possible. Look for features that will tell you when you've gone too far, or wandered to the right, or gone in the wrong direction.

4. Be aware of how far you've gone. Keep a mental note of how far you've travelled from your last known location. You can simply guess at this by intuition, but it is better to keep a count of your paces. This information will tell you when you've gone past your goal, where you are on a path, or about where you are when you get disoriented. (Orienteers never get "lost"!)

5. Follow along linear features when possible. If a path, roadway, ridge top, stream, or other linear feature roughly



follows the direction you're traveling, follow along with it. Such features are easy to locate and can be followed as fast as you care to travel with no danger of wandering off course. The problem with using such a feature is that it's hard to determine where to leave it. Pace-counting or an identifying feature (trail intersection, building, etc.) is necessary to establish when it's time to change direction.

6. Go as far as you think you should, then go a little further. Unless you have a very good reason to think otherwise, you generally don't reach a goal as fast as you think you should. So if you're travelling along and haven't been pace-counting, and there are no distinctive features in sight, you generally won't reach your goal until after you expect to.

7. Believe your compass. If you think that your compass is wrong, check to be sure that there aren't any large pieces of metal immediately around it. If there aren't, assume that the compass is correct — even if you're sure that it's wrong. Your internal sense of direction is much easier to confuse than is the compass.

8. If you get disoriented, look for a long, distinctive feature and head for it. Typically, such a feature is a road. When you reach it, follow it until you reach a distinctive feature (e.g. a stream crossing) and use that to establish your location.

continued on next page

Orienteering

continued from previous page

PART II: THE COMPUTER SIMULATION

Jacque's Coulee is the local name of a small stream in central Minnesota. Along that stream is an orienteering area complete with 24 permanent markers, as detailed on the accompanying map. You are invited to orienteer there; and you don't even have to go to Minnesota to do so. With the Jacque's Coulee orienteering simulation you can "run" in this area from the comfort of your armchair and keyboard. The simulation allows the computer to be your eyes and legs at Jacque's Coulee while you make the orienteering decisions. Do you take the road to travel fast, or go cross-country to save distance? Use an attack feature, or precise bearing? Up over the hill, or around it? The decisions, and consequences, are yours.

HOW TO PLAY THE SIMULATION

The computer uses a 600-by-600 grid to locate any place in the orienteering area. Whenever the computer gives you coordinates, or you give them to the computer, the column is always the first number and the row is the second. See-through graph paper can be used to help you determine the location of any coordinates.

As the simulation begins the computer will ask you if this is to be a competition. If you select this option, you will then be asked to enter up to 12 control points which must be reached in order to complete the course. Entering a zero will end the list of required points.

Following this, the computer will ask you for the coordinates of the location at which you want to start. You enter this information by typing the column (X-coordinate), then a comma, then the row (Y-coordinate), then RETURN. If you have selected the competition option, you can elect to start at the first control point by entering a zero for the X-coordinate.

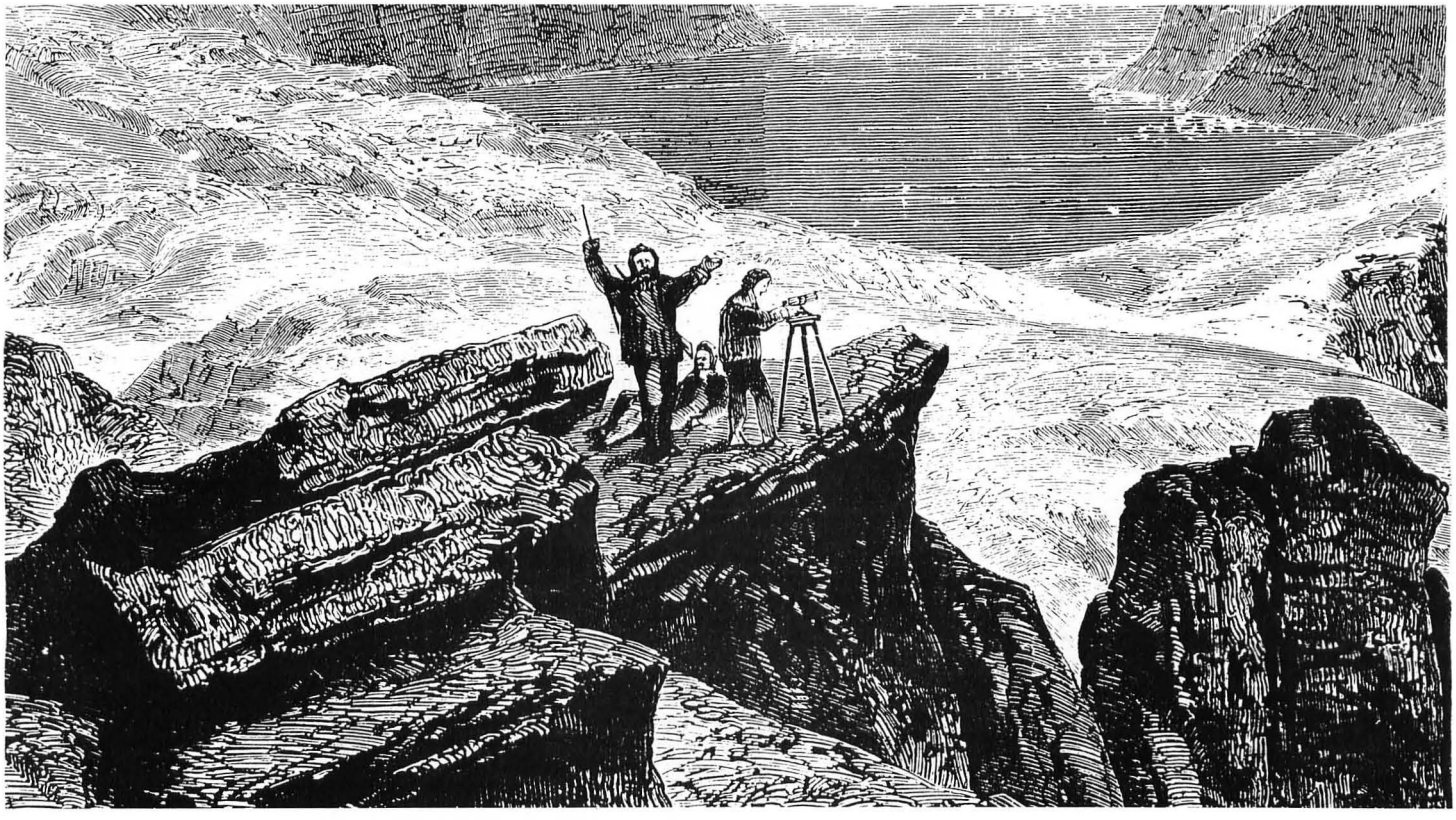
Your direction of travel is given to the machine by typing a bearing and then pressing RETURN. Any bearing between 0 and 360 may be used (0 and 360 both representing straight north). The vertical grid lines on the map indicate magnetic north, with north to the top.

You may run along the bearing you select at any one of five speeds: 1 is very slow, 2 is slow, 3 is normal, 4 is fast, and 5 is very fast. To select your speed, simply type a number and press RETURN. It is important that you remember that the faster your speed, the greater the error you may realize as you try to follow along your bearing. At the slowest speed the maximum navigational error that you could realize is 2 degrees; at the fastest speed it could be as much as 35 degrees.

After you have set your speed, the computer will move you 25 meters along the bearing you selected, altered by navigational error. At this point you will stop and "look around." Generally the computer will give you a short description of your surroundings (e.g. "You are by a stream," "You are in a boulder field"). Unless the description mentions vegetation, you can assume that you are in an open deciduous forest. If you are on a hillside, the computer will give you a rough idea of the direction of the slope (e.g. "You are on a slope — N is uphill"). This direction is meant only to be a general description of the slope. If north is described as uphill, it is possible that northeast is actually straight uphill and north is uphill at an angle. The description is never wrong, but may not be precise.

If you are within sight of an attack feature when you stop to look around, the computer will give you the coordinates of your location. This is done only within sight of features that are distinct enough to allow you to orient your exact location. Such features include buildings, isolated distinct boulders, high points, stream crossings on trails, and others.

At the same time that the computer describes your location, it will ask you if you want to continue on the course just set. Pressing "Y" will cause the computer to move you another 25 meters on your bearing (altered by



navigational error) at the speed you last selected. Pressing "N" will allow you to change direction and speed.

It is possible to exercise almost all orienteering techniques while "running" at Jacque's Coulee. Attack points, collecting features, roads, corridor features, rough compass, and aiming off all can be used in the same way as in real orienteering events. Following a linear feature (such as a road or stream) to avoid any directional error is possible by simply reaching such a feature and then entering a direction of travel equal to the direction in which the feature runs. It is important that this be accurate within a degree or two, or else the computer will assume that you want to angle away rather than follow along it. In this case you would get a navigational error as usual. The simulation does not allow any way to use contouring as a navigational technique.

When you come within sight of a marker, the computer will tell you that you can see it. It will also ask if you want to go to the marker. If you do, then press "Y". This will move you to the marker and allow you to see its control code. Be sure to check it against the code of the marker you are looking for, since there are 24 markers in Jacque's Coulee and some are near each other. If you want to punch in, type "Y" in response to the computer's question, and it will register the marker on your control card. If you don't punch in, no record is kept of your having reached the marker.

If you have selected the competition option, the required control points are constantly displayed at the bottom of the screen. As you punch in at each of these, they are reprinted in inverse video to keep a tally of your progress.

The computer constantly updates and displays the time it has taken to travel along your route. The time is determined both by your speed and by the terrain; you can go one-third faster on a gravel road than through flat woods. The time that it takes you to consider your route and respond to the computer's prompts is not accumulated — only the simulated run time is counted.

As you run at Jacque's Coulee you gradually get more and more tired, just as you would in a real competition. The more tired you get, the harder it is for you to concentrate, and therefore the more navigation errors you make. For every "fatigue error point" you accumulate, the maximum possible error you could have at any given speed increases by one degree. Running at "very slow" speed adds fatigue points slowly — only one point in nearly two hours. By contrast, running only five minutes at "very fast" speed will add the same fatigue unit. The computer displays your current fatigue error points throughout the simulation.

If you become hopelessly disoriented, you can either abandon the course by entering a bearing of 999, or pray. For the latter option, type "PRAY" in response to the computer's question about whether you want to continue as you have been.

USES OF THE SIMULATION

1. As a Teaching Aid. Any subject is learned more thoroughly when students discover techniques on their own rather than simply being taught abstract rules. In addition, the simulation can be used to illustrate both how and why various orienteering techniques are used.

2. Point-to-Point Competition. Any number of point-to-point courses may be set using the 24 markers in the area.

3. Score Competition. You can set a start location and then assign appropriate score points to all or some of the available markers. Competitors are then allowed a given accumulated time to put as many markers on their control card as they can. Penalty points are given to those who use more than the allotted time. It is suggested that the finish point be a specified marker so that it can be reached precisely at the end.

4. Lost. An interesting game for two or more competitors can be played using the simulation. One player secretly enters a start location into the computer. Another player then takes over and tries to figure out where he is. He does this by deducing his general

continued on next page

Orienteering

continued from previous page

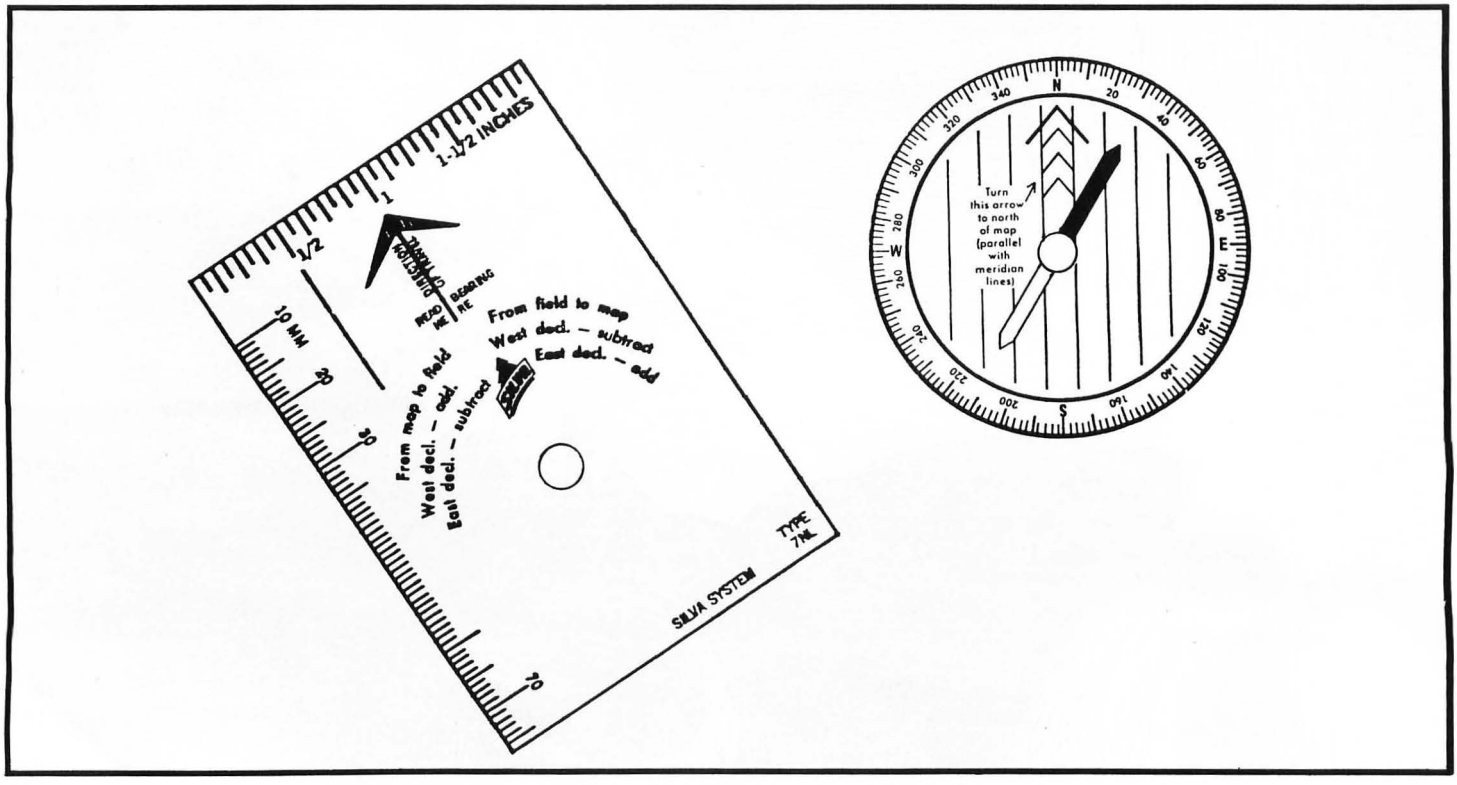
location from the description of his surroundings given by the computer, and then navigating to a feature distinctive enough to allow him to orient his exact location. The person who is able to orient himself in the shortest time wins.

VARIABLES

A: Slope of travel bearing.
AD %: Distance traveled by runner since last change in speed or direction.
B: The point on the Y axis intersected by the travel bearing.
BG: Runner's intended bearing.
BY$: Used in describing runner's location.
CARD$: String of control codes of markers at which runner has punched in.
CC$: Control code of a marker.
CP$: Control point for competition.
D: Distance traveled by runner since last time/fatigue update (usually 25 m.)
DB: Difference between travel bearing and straight uphill.
DER: Actual directional error.
DIST: Total distance travelled.
DR$: Cardinal direction closest to straight uphill.
F: Percent of top speed which defines the other four speeds.
FER: Maximum fatigue-caused directional error.
INC: X-distance between points which are 25 m. apart on the runner's travel bearing.
MER: Maximum possible speed-caused directional error.
MSP: The time (in seconds) needed to cover one meter of a given terrain at maximum speed.
N$, 0$: Used in describing runner's location.
SP: Speed code of runner.
SR: Flag indicating whether the runner is searching for a marker. (Used to avoid "seeing" a marker that the runner is leaving.)
TB: Runner's travel bearing.
TEST$: Competition control points.
TM, TS: Runner's accumulated time in minutes and seconds.
X, Y: Runner's horizontal and vertical coordinates.
XT, YT: Horizontal and vertical coordinates of point which is 1000 m. further along runner's bearing.
ZONE: Flag indicating whether runner's current location will allow exact orientation.

GLOSSARY

AIMING OFF — An orienteering technique in which the bearing followed is to one side of a direct bearing. This is done so that the direction in which to look for the goal is known.
ATTACK POINT — A distinctive, easy-to-find feature near a goal which is used as the start of a final approach to the goal.
BEARING — Also called azimuth, a direction of travel defined by the angle formed clockwise from magnetic north (eg. the bearing of east is 90°).
CATCH FEATURE — A linear feature (eg. stream, path) that is perpendicular to your direction of travel and which can be used to indicate your general location, and especially to indicate when you've gone beyond your goal.
CLUE SHEET — A list of short descriptions of the locations of orienteering markers and the identifying codes written on those markers.
CONTROL CARD — The card which is punched with identifying codes at each orienteering marker to prove which markers were found.
CONTROL CODE — A unique identifying code, either letters or a number, that is written on every orienteering marker and is used to confirm that the marker found is the marker sought.
FIGHT — Thick brush of any sort that a person has to "fight" to get through.
HAND RAIL — A linear feature (eg. stream, path) that parallels your direction of travel and which can be used to guide you to your goal.
RE-ENTRANT — A small valley on a hillside, usually formed by an intermittent stream.
ROCK FACE — A small exposed rock bluff.
SADDLE — The low point on a ridge, normally a low point between the summits of a hill with more than one summit.
SPUR — A small ridge on a hillside, usually formed by two intermittent streams, one on each side of the spur.



Initialize program and variables.

```
1  HOME
2  VTAB 23: FOR K = 1 TO 39: PRINT
     "*";: NEXT K: PRINT : VTAB 1
     : FOR K = 1 TO 39: PRINT "*"
     ;: NEXT K: PRINT
3  FOR K = 1 TO 22: PRINT "*";: HTAB
     39: PRINT "*": NEXT K
4  VTAB 4: HTAB 14: INVERSE : PRINT
     "ORIENTEERING"
5  VTAB 6: HTAB 9: PRINT "AT   JA
     CQUE'S   COULEE": NORMAL
6  VTAB 8: HTAB 9: PRINT "(A COMP
     UTER SIMULATION)"
7  VTAB 10: HTAB 13: PRINT "COPYR
     IGHT 1980"
8  VTAB 11: HTAB 11: PRINT "MICHA
     EL A. O'KEEFE"
9  VTAB 13: HTAB 11: PRINT "APPLE
      VERISION BY"
10  VTAB 14: HTAB 14: PRINT "ROBE
     RT WHITE"
11  VTAB 21: HTAB 14: INVERSE : PRINT
     "PRESS ANY KEY": NORMAL
12  POKE  - 16368,0: WAIT  - 1638
     4,128: HOME : POKE 35,20
13  DIM AA$(24),XS(24),YS(24)
15  DATA AA,AB,AC,AE,BB,BC,BD,BF,
     BH,BJ,BK,BL,BM,BT,DA,EB,EC,E
     D,EE,EF,EG,EH,HI,JR
16  FOR FF = 0 TO 23: READ AA$(FF
     ): NEXT FF
18  DATA 369,325,530,236,384,294,
     422,432,290,65,99,152,124,95
     ,91,50,165,82,356,187,450,56
     5,205,505
20  FOR FF = 0 TO 23: READ XS(FF)
     : NEXT FF
22  DATA 281,560,352,487,506,371,
     368,255,182,162,233,433,514,
     95,332,251,324,384,455,511,4
     05,287,475,80
24  FOR FF = 0 TO 23: READ YS(FF)
     : NEXT FF
90 O$ = "YOU ARE ON A":N$ = "YOU
     ARE IN A":BY$ = "YOU ARE BY
     A":CARD$ = "":FER = 0:TM = 0
     :TS = 0:ZNE = 1:DIST = 0:TES
     T$ = "":TQ = 0
91  HOME : PRINT "IS THIS A COMPE
     TITION? (Y/N) ";: GET A$: PRINT
     A$
92  IF A$ = "N" THEN 110
93  IF A$ <  > "Y" THEN 91
94 TQ = 1: PRINT : PRINT "ENTER C
     ONTROL POINTS (ENTER '0' TO
     END)": PRINT
95  FOR CP = 0 TO 12
96  INPUT CP$: IF CP$ = "0" THEN
      100
97  FOR CK = 0 TO 23: IF CP$ <  >
     AA$(CK) THEN  NEXT CK: CALL
      - 1052: PRINT "INVALID C.P.
     !!": GOTO 96
98  IF CP = 0 THEN SP = CK
99 TEST$ = TEST$ + CP$: NEXT CP
100  PRINT : PRINT "ENTER START P
     OINT - X=0 WILL USE THE    F
     IRST CONTROL POINT ";: INPUT
     X,Y
101  VTAB 24: HTAB 1
102  FOR PT = 1 TO  LEN (TEST$) STEP
     2
103  PRINT  MID$ (TEST$,PT,2);" "
     ;
104  NEXT PT
105  IF X = 0 THEN  INVERSE : VTAB
     24: HTAB 1: PRINT  LEFT$ (TE
     ST$,2);: NORMAL :X = XS(SP):
     Y = YS(SP):CARD$ = CARD$ +  LEFT$
     (TEST$,2)
106  VTAB 1: HTAB 1
110  HOME
112  IF TQ = 1 THEN  PRINT "START
      COORDINATES ARE: ";X;", ";Y
     : GOTO 190
```

Input the starting coordinates, speed, and direction of travel.

```
120  VTAB 3: PRINT "WHAT ARE THE
     COORDINATES"
130  PRINT "OF THE START?"
150  VTAB 4: HTAB 20
160  INPUT "";X,Y
170  IF X < 0 OR Y < 0 OR X > 600
      OR Y > 600 THEN  CALL  - 10
     52: GOTO 150
190  VTAB 6: PRINT "WHAT IS THE B
     EARING YOU": PRINT "INTEND T
     O TRAVEL?"
220  VTAB 7: HTAB 25:AD% = 0
230  INPUT "";TB
235  IF TB = 999 THEN 8000
240  IF TB < 0 OR TB > 360 THEN  CALL
      - 1052: GOTO 220
250  VTAB 9: PRINT "YOU MAY RUN T
     HIS BEARING": PRINT "AT ANY
     ONE OF THESE SPEEDS:"
280  PRINT  TAB( 5);"1.  VERY SLO
     W"
285  PRINT  TAB( 5);"2.  SLOW"
290  PRINT  TAB( 5);"3.  NORMAL"
300  PRINT  TAB( 5);"4.  FAST"
305  PRINT  TAB( 5);"5.  VERY FAS
     T": PRINT "ENTER THE NUMBER"

310  VTAB 16: HTAB 20
315  INPUT "";SP
320  IF  INT (SP) <  > SP OR SP <
     1 OR SP > 5 THEN  CALL  - 10
     52: GOTO 310
330  VTAB 20: HTAB 10: INVERSE : PRINT
     "TAKING THE BEARING": NORMAL
```

Determine if location and direction of travel protect runner from receiving a navigational error.

```
350  IF Y >  - 2.1 * X + 508 AND
     Y <  - 2 * X + 530 AND Y > 3
     36 AND ((TB > 150 AND TB < 1
     58) OR (TB > 330 AND TB < 33
     8)) THEN 7000
360  IF Y >  - .78 * X + 554 AND
     Y <  - .77 * X + 569 AND Y >
     460 AND ((TB > 123 AND TB <
     132) OR (TB > 302 AND TB < 3
     10)) THEN 7000
370  IF Y > .84 * X + 346 AND Y <
     .99 * X + 339 AND Y <  - 1.2
      * X + 807 AND Y > 480 AND (
     (TB > 44 AND TB < 52) OR (TB
      > 220 AND TB < 230)) THEN 7
     000
380  IF Y < .8 * X + 348 AND Y >
     .81 * X + 328 AND Y >  - X +
     630 AND Y <  - X + 770 AND (
     (TB > 45 AND TB < 53) OR (TB
      > 225 AND TB < 233)) THEN 7
     000
390  IF Y > .44 * X + 377 AND Y <
     .4 * X + 400 AND Y >  - 1.67
      * X + 763 AND Y <  - 2 * X +
     970 AND ((TB > 62 OR TB < 72
     ) OR (TB > 242 AND TB < 252)
     ) THEN 7000
400  IF Y < 1.21 * X + 35 AND Y >
     1.25 * X - 5 AND Y > .167 +
     323 AND ((TB > 34 AND TB < 4
     4) OR (TB > 216 AND TB < 224
     )) THEN 7000
410  IF Y < .2 * X + 320 AND Y >
     .2 * X + 304 AND ((TB > 75 AND
     TB < 83) OR (TB > 255 AND TB
      < 263)) THEN 7000
420  IF Y < 3.5 * X - 65 AND Y >
     4 * X - 178 AND Y > 372 AND
     Y < 432 AND ((TB > 12 AND TB
      < 20) OR (TB > 192 AND TB <
     200)) THEN 7000
```

continued on next page

continued from previous page

```
440  IF Y > .44 * X + 182 AND Y <
     .45 * X + 195 AND X < 370 AND
     X > 210 AND ((TB > 61 AND TB
      < 69) OR (TB > 241 AND TB <
     249)) THEN 7000
450  IF Y > 1.33 * X - 483 AND Y <
     1.3 * X - 442 AND Y > - .8 *
     X + 620 AND ((TB > 34 AND TB
      < 42) OR (TB > 214 AND TB <
     222)) THEN 7000
460  IF Y > - .02 * X + 160 AND
     Y < - .02 * X + 170 AND X >
     260 AND X < 422 AND ((TB > 8
     6 AND TB < 94) OR (TB > 266 AND
     TB < 274)) THEN 7000
470  IF Y < .2 * X + 48 AND Y > .
     23 * X + 28 AND X > 46 AND X
      < 320 AND ((TB > 75 AND TB <
     83) OR (TB > 255 AND TB < 26
     3)) THEN 7000
480  IF Y > .74 * X + 172 AND Y <
     .72 * X + 190 AND X < 70 AND
     ((TB > 49 AND TB < 57) OR (T
     B > 229 AND TB < 237)) THEN
     7000
485  IF Y > 335 AND Y < 354 AND X
      > 528 AND X < 544 THEN 7000
```

Assign navigational error.

```
500  ON SP GOTO 505,510,515,520,5
     25
505 MER = 3: GOTO 550
510 MER = 6: GOTO 550
515 MER = 15: GOTO 550
520 MER = 25: GOTO 550
525 MER = 35
550 MER = MER + FER
560 MER = MER * .8:DER =  INT (ME
     R *  RND (MER)) -  INT (MER *
      RND (MER)) +  INT (MER *  RND
     (MER)) -  INT (MER *  RND (M
     ER)):TB = TB + DER: IF TB <
     0 THEN TB = TB + 360
561  IF TB > 360 THEN TB = TB - 3
     60
650  IF TB < 4 OR TB > 356 THEN V
     ERT = 1: GOTO 655
651  IF TB > 176 AND TB < 184 THEN
     VERT =  - 1: GOTO 655
653 VERT = 0: GOTO 670
655 Y = Y + VERT * 6.562: GOTO 85
     0
660 VERT = 0
```

Move runner to new coordinates.

```
670 QD =  INT (TB / 90) + 1:K1 =
     .01745329:K2 = 262.48: ON QD
      GOTO 680,690,700,710
680 TX = X +  SIN (TB * K1) * K2:
     TY = Y +  COS (TB * K1) * K2
     : GOTO 720
690 TX = X +  COS ((TB - 90) * K1
     ) * K2:TY = Y -  SIN ((TB -
     90) * K1) * K2: GOTO 720
700 TX = X -  SIN ((TB - 180) * K
     1) * K2:TY = Y -  COS ((TB -
     180) * K1) * K2: GOTO 720
710 TX = X -  COS ((TB - 270) * K
     1) * K2:TY = Y +  SIN ((TB -
     270) * K1) * K2
720 A = (Y - TY) / (X - TX):B = Y
      - A * X:INC = (TX - X) / 40

800 X = X + INC:Y = A * X + B
```

Check to see if the runner is still on the map; if not, end the run.

```
850 D = 25:AD% = AD% + D:DIST = D
    IST + D
860  IF X > 0 AND X < 600 AND Y >
     0 AND Y < 600 THEN 1140
960  HOME : VTAB 5: PRINT "YOU HA
     VE RUN OFF THE MAP": PRINT "
     AND BEEN EATEN BY MONSTERS!"
     : VTAB 23: HTAB 9: INVERSE :
      PRINT "PRESS ANY KEY": NORMAL

961  POKE  - 16368,0: WAIT  - 163
     84,128
970  GOTO 8000
```

Determine which square inch the runner is in.

```
1140  HOME
1141 X% = X / 100:X% = X% + 1:Y% =
     Y / 100:Y% = Y% + 1
1142 K =  PEEK ( - 16384): IF K <
     127 THEN 1150
1143  IF K <  > 210 THEN 1150
1145  PRINT X;" ";Y;"  ";TB;"  ";
     MSP
1150  ON X% GOTO 1160,1170,1180,1
     190,1200,1210
1160  ON Y% GOTO 2000,2100,2200,2
     300,2400,2500
1170  ON Y% GOTO 2600,2700,2800,2
     900,3000,3100
1180  ON Y% GOTO 3200,3300,3400,3
     500,3600,3700
1190  ON Y% GOTO 3800,3900,4000,4
     100,4200,4300
1200  ON Y% GOTO 4400,4500,4600,4
     700,4800,4900
1210  ON Y% GOTO 5000,5100,5200,5
     300,5400,5500
```

Determine the zone that the runner is in, set the defining characteristics, and go to the appropriate program line.

```
2000 DR$ = "W": IF Y < 72 AND Y >
     30 AND X > 13 AND X < 26 THEN
      GOSUB 10500: GOTO 6200
2010  IF Y > 90 AND X > 90 THEN M
     X = 95:MY = 95:CC$ = "BT": GOSUB
     10500: GOTO 6000
2020  IF Y <  - .5 * X + 105 AND
     Y >  - .5 * X + 90 AND Y < 2
     .75 * X - 128 THEN  GOSUB 11
     100: GOTO 6200
2030  GOSUB 11300: GOTO 6200
2100  IF X > 55 AND X < 70 AND Y >
     135 AND Y < 170 THEN MX = 65
     :MY = 162:CC$ = "BJ": GOSUB
     10500: GOTO 6000
2110  IF Y > .74 * X + 172 AND Y <
     .72 * X + 190 THEN  GOSUB 10
     700: GOTO 6200
2120  IF X > 80 AND Y < 108 THEN
      GOSUB 11450: GOTO 6200
2130  IF Y <  - 2 * X + 190 AND Y
      > 133 THEN DR$ = "SE": GOSUB
     11100: GOTO 6200
2135  IF X > 50 AND X < 61 AND Y >
     115 AND Y < 136 THEN  GOSUB
     11250: GOTO 6200
2140  IF X > 80 AND Y < 140 AND Y
      > 130 THEN DR$ = "W": GOSUB
     11100: GOTO 6200
2150  IF Y < 3.3 * X - 43 THEN DR
     $ = "W": GOSUB 11300: GOTO 6
     200
2160 DR$ = "SE": GOSUB 11300: GOTO
     6200
2200  IF Y < 1.2 * X + 158 AND Y >
     1.7 * X + 60 AND Y >  - .7 *
     X + 281 AND Y <  - .7 * X +
     312 THEN MX = 99:MY = 233:CC
     $ = "BK":MSP = .4: GOSUB 105
     00: GOTO 2202
2201  GOTO 2210
2202  IF X > 90 THEN 6000
```

```
2203  GOTO 6200
2210  IF X > 70 AND X < 82 AND Y <
     292 AND Y > 278 THEN  GOSUB
     10500: GOTO 6200
2215  IF X > 45 AND X < 55 AND Y <
     259 AND Y > 248 THEN MX = 50
     :MY = 251:CC$ = "EB": GOSUB
     10500: GOTO 6000
2220  IF Y > .74 * X + 172 AND Y <
     .72 * X + 190 AND X < 75 THEN
      GOSUB 10700: GOTO 6200
2230  IF Y < 1.7 * X + 90 THEN  GOSUB
     11250: GOTO 6200
2240 DR$ = "SE": GOSUB 11300: GOTO
     6200
2300  IF Y < 390 AND Y > 372 AND
     X < 90 AND X > 75 THEN  GOSUB
     10500:MX = 82:MY = 384:CC$ =
     "ED": GOTO 6000
2310  IF X > 85 AND Y > 326 AND Y
      < 339 THEN MX = 91:MY = 332
     :CC$ = "DA": GOSUB 10500: GOTO
     6000
2320  IF X > 20 AND X < 55 AND Y <
     .5 * X + 349 AND Y > .7 * X +
     321 THEN  GOSUB 10600: GOTO
     6200
2330  IF X > 69 AND X < 83 AND Y <
     315 AND Y > 302 THEN  GOSUB
     11200: GOTO 6200
2340  IF Y < .2 * X + 320 AND Y >
     .2 * X + 304 THEN  GOSUB 115
     00: GOTO 6200
2350  IF Y > - 2.1 * X + 508 AND
     Y < - 2 * X + 518 THEN  GOSUB
     10650: GOTO 6200
2360  IF Y > = .2 * X + 320 THEN
     DR$ = "N": GOSUB 11300: GOTO
     6200
2370 DR$ = "S": GOSUB 11400: GOTO
     6200
2400  IF X > 46 AND X < 54 AND Y <
     429 AND Y > 415 THEN  GOSUB
     11200: GOTO 6200
2410  IF Y > - 2.1 * X + 508 AND
     Y < - 2 * X + 530 THEN  GOSUB
     10650: GOTO 6200
2420  IF Y < = - 2.1 * X + 508 THEN
     DR$ = "W": GOSUB 11300: GOTO
     6200
2430  IF Y > - .78 * X + 554 AND
     Y < - .77 * X + 569 THEN  GOSUB
     11650: GOTO 6200
2440  IF Y > = - .77 * X + 569 THEN
     DR$ = "SW": GOSUB 11300: GOTO
     6200
2450 DR$ = "NE": GOSUB 11300: GOTO
     6200
2500  IF Y > - .78 * X + 554 AND
     Y < - .77 * X +  + 569 THEN
      GOSUB 11650: GOTO 6200
2510  IF Y < = - .78 * X + 554 THEN
     DR$ = "NE": GOSUB 11300: GOTO
     6200
2520 DR$ = "SW": GOSUB 11300: GOTO
     6200
2600  IF X > 130 AND X < 140 AND
     Y > 88 THEN  GOSUB 10500: GOTO
     6200
2610  IF X > 139 AND X < 148 AND
     Y > 65 AND Y < 75 THEN  GOSUB
     10500: GOTO 6200
2620  IF Y < .2 * X + 48 AND Y >
     .23 * X + 28 AND X > 146 THEN
     2625
2622  GOTO 2630
2625  IF X > 187 THEN  GOSUB 1050
     0: GOTO 6200
2627  GOSUB 11500: GOTO 6200
2630  IF Y > - 8 * X + 1410 AND
     Y < - 3 * X + 640 THEN  GOSUB
     10800: GOTO 6200
2640 DR$ = "W": GOSUB 11300: GOTO
     6200
2700 DR$ = "W": IF X > 170 AND Y <
     180 AND Y > 105 AND Y < 115 THEN
      GOSUB 10500: GOTO 6200
2710  IF X > 121 AND X < 151 AND
     Y > 130 AND Y < 163 THEN  GOSUB
     10600: GOTO 6200
2730  IF (Y > 125 AND Y < 135 AND
     X > 195) OR (Y > 145 AND Y <
     2.3 * X - 307) THEN  GOSUB 1
     1200: GOTO 6200
2740  IF (X > 163 AND X < 190 AND
     Y < 160) OR (Y > 160 AND Y <
     1.1 * X - 17) THEN  GOSUB 10
     800: GOTO 6200
2750  IF Y > 125 AND Y < 145 AND
     X < 128 THEN  GOSUB 11100: GOTO
     6200
2760  GOSUB 11300: GOTO 6200
2800  IF X > 152 AND X < 180 AND
     Y > 238 AND Y < 260 THEN  GOSUB
     10500: GOTO 6200
2810  IF X < 122 AND X > 114 AND
     Y < 285 AND Y > 273 THEN  GOSUB
     10500: GOTO 6200
2820  IF Y > - 1.6 * X + 506 AND
     Y < - 1.3 * X + 474 AND Y <
     238 THEN DR$ = "NW": GOSUB 1
     1100: GOTO 6200
2830  IF Y < - .8 * X + 343 AND
     X < 150 THEN  GOSUB 11250: GOTO
     6200
2840  IF Y > 260 AND Y > 1.3 * X +
     55 THEN DR$ = "S": GOSUB 113
     00: GOTO 6200
2850  IF Y > - 1.7 * X + 520 THEN
     DR$ = "W": GOSUB 11300: GOTO
     6200
2860  IF Y > 240 THEN DR$ = "E": GOSUB
     11300: GOTO 6200
2870 DR$ = "N": GOSUB 11300: GOTO
     6200
2900  IF Y < .17 * X + 352 AND Y >
     .17 * X + 332 AND X > 110 THEN
      GOSUB 11600: GOTO 6200
2910  IF Y < 3.5 * X - 65 AND Y >
     4 * X - 178 AND Y > 372 THEN
     DR$ = "N": GOSUB 11100: GOTO
     6200
2920  IF Y < .1 * X + 312 AND Y >
     .14 * X + 296 AND ((X > 135 AND
     X < 150) OR (X > 156 AND X <
     175)) THEN MX = 165:MY = 324
     :CC$ = "EC":ZNE = 2: GOSUB 1
     1200: GOTO 2925
2922  GOTO 2930
2925  IF X > 155 THEN 6000
2926  GOTO 6200
2930  IF Y < .15 * X + 303 AND Y >
     .19 * X + 284 THEN DR$ = "S"
     : GOSUB 11400: GOTO 6200
2940  IF Y > = .2 * X + 320 THEN
     DR$ = "N": GOSUB 11300: GOTO
     6200
2950  IF Y < = .2 * X + 304 THEN
     DR$ = "S": GOSUB 11300: GOTO
     6200
2960  GOSUB 11500: GOTO 6200
3000  IF X > 130 AND X < 140 AND
     Y > 450 AND Y < 459 THEN  GOSUB
     10500: GOTO 6200
3005  IF Y > - .78 * X + 554 AND
     Y < - .77 * X + 569 AND X <
     130 THEN  GOSUB 11650: GOTO
     6200
3010  IF Y < 437 AND Y > 425 AND
     X < 155 AND X > 140 THEN MX =
     152:MY = 433:CC$ = "BL": GOSUB
     10500: GOTO 6000
3020  IF Y > .84 * X + 346 AND Y <
     .99 * X + 339 AND Y > 480 THEN
     DR$ = "SW": GOSUB 11100: GOTO
     6200
3030  IF Y < .8 * X + 348 AND Y >
     .81 * X + 328 AND Y > - X +
     630 THEN  GOSUB 11450: GOTO
     6200
3040  IF Y > .44 * X + 377 AND Y <
     .4 * X + 400 AND Y > - 1.67
      * X + 763 THEN DR$ = "SW": GOSUB
     11100: GOTO 6200
```

continued on next page

continued from previous page

```
3050  IF Y < 3.5 * X - 65 AND Y >
      4 * X - 178 AND Y < 432 THEN
      DR$ = "N": GOSUB 11100: GOTO
      6200
3060  IF Y <  - X + 643 AND Y > -
      X + 610 AND Y < 481 AND Y >
      456 THEN  GOSUB 11300: GOTO
      6200
3080 DR$ = "N": GOSUB 11300: GOTO
      6200
3100 DR$ = "N": IF Y < .6 * X + 6
      44 AND Y > .67 * X + 443 THEN
       GOSUB 10900: GOTO 6200
3110  IF Y > 510 AND Y < 520 AND
      X > 115 AND X < 124 THEN MX =
      124:MY = 514:CC$ = "BM": GOSUB
      10500: GOTO 6000
3120  IF Y > .84 * X + 346 AND Y <
      .99 * X + 339 THEN MX = 187:
      MY = 511:CC$ = "EF":ZNE = 2:
       GOSUB 11100: GOTO 3135
3130  GOSUB 11300: GOTO 6200
3135  IF X > 180 AND X < 190 THEN
      6000
3136  GOTO 6200
3200  IF Y < .2 * X + 48 AND Y >
      .23 * X + 28 THEN  GOSUB 115
      00: GOTO 6200
3210  IF Y <  - 2 * X + 510 AND Y
       > - 2 * X + 450 THEN DR$ =
      "W": GOSUB 11300: GOTO 6200
3220  GOSUB 10800: GOTO 6200
3300  IF Y >  - .02 * X + 160 AND
      Y <  - .02 * X + 170 AND X >
      260 THEN  GOSUB 10500: GOTO
      6200
3310  IF X > 285 AND Y > 175 AND
      Y < 189 THEN MX = 290:MY = 1
      82:CC$ = "BH": GOSUB 10500: GOTO
      6000
3320  IF Y < .2 * X + 48 THEN  GOSUB
      11500: GOTO 6200
3330  IF (Y < 1.5 * X - 150 AND Y
       > 1.75 * X - 222) OR (Y < 1
      35 AND Y > 123 AND X < 205) THEN
       GOSUB 11200: GOTO 6200
3340  GOSUB 10800: GOTO 6200
3400  IF X > 210 AND X < 226 AND
      Y > 220 AND Y < 232 THEN  GOSUB
      10500: GOTO 6200
3410  IF Y > .44 * X + 182 AND Y <
      .45 * X + 195 AND X > 210 THEN
       GOSUB 11150: GOTO 6200
3420  IF X > 230 AND X < 240 AND
      Y > 266 AND Y < 279 THEN  GOSUB
      10500: GOTO 6200
3430  IF X > 235 AND X < 260 AND
      Y > 262 THEN  GOSUB 10800: GOTO
      6200
3440 DR$ = "W": GOSUB 11300: GOTO
      6200
3500  IF Y < .4 * X + 296 AND Y >
      .15 * X + 338 AND X > 210 THEN
       GOSUB 11600: GOTO 6200
3510  IF X > 286 AND Y < 389 AND
      Y > 361 THEN MX = 294:MY = 3
      71:CC$ = "BC": GOSUB 10500: GOTO
      6000
3520  IF X > 229 AND X < 246 AND
      Y < 322 AND Y > 311 THEN  GOSUB
      10500: GOTO 6200
3530  IF Y < 1.21 * X + 35 AND Y >
      1.25 * X - 5 AND Y > 370 THEN
       GOSUB 11650: GOTO 6200
3540  IF Y > .44 * X + 182 AND Y <
      .45 * X + 195 THEN  GOSUB 11
      150: GOTO 6200
3550  IF Y >  = .2 * X + 320 OR Y
      <  = .44 * X + 182 THEN DR$
       = "N": GOSUB 11300: GOTO 62
      00
3560  IF Y <  = .2 * X + 304 THEN
      DR$ = "S": GOSUB 11300: GOTO
      6200
3570  GOSUB 11500: GOTO 6200
3600  IF Y > .81 * X + 328 THEN  GOSUB
      11450: GOTO 6200
3610  IF X > 260 AND X < 293 AND
      Y > 413 AND Y < 441 THEN  GOSUB
      10600: GOTO 6200
3620 DR$ = "SW": IF Y > .44 * X +
      377 AND Y < .4 * X + 400 AND
      Y <  - 2 * X + 970 THEN 3645

3630  IF X > 260 THEN  GOSUB 1080
      0: GOTO 6200
3640 DR$ = "W": GOSUB 11300: GOTO
      6200
3645  IF X > 230 THEN MX = 236:MY
       = 487:CC$ = "AE":ZNE = 2: GOSUB
      11100: GOTO 6000
3646  IF X < 211 THEN MX = 205:MY
       = 475:CC$ = "HI":ZNE = 2: GOSUB
      11100: GOTO 6000
3647  GOSUB 11100: GOTO 6200
3700 DR$ = "SW": IF X > 215 AND X
      < 225 AND Y > 569 AND Y < 5
      80 THEN  GOSUB 10500: GOTO 6
      200
3710  IF Y > .84 * X + 346 AND Y <
      .99 * X + 339 AND Y <  - 1.2
      * X + 807 THEN  GOSUB 11100
      : GOTO 6200
3720  IF Y < .8 * X + 348 AND Y >
      .81 * X + 328 AND Y <  - X +
      770 THEN  GOSUB 11450: GOTO
      6200
3730  IF Y <  - 1.8 * X + 948 THEN
       GOSUB 11300: GOTO 6200
3740  GOSUB 10800: GOTO 6200
3800  IF Y > .23 * X + 28 THEN  GOSUB
      11500: GOTO 6200
3810  GOSUB 10800: GOTO 6200
3900  IF Y >  - .02 * X + 160 AND
      Y <  - .02 * X + 170 THEN  GOSUB
      11050: GOTO 6200
3910  IF X > 321 AND X < 331 AND
      Y < 121 AND Y > 111 THEN  GOSUB
      11000: GOTO 6200
3920  IF X > 310 AND X < 340 AND
      Y > 101 AND Y < 131 THEN  GOSUB
      10500: GOTO 6200
3930  IF X < 320 AND Y < .2 * X +
      48 AND Y > .23 * X + 28 THEN
       GOSUB 11500: GOTO 6200
3940  GOSUB 10800: GOTO 6200
4000  IF Y < 290 AND Y > 270 AND
      X > 320 AND X < 350 THEN  GOSUB
      11800: GOTO 6200
4010  IF X < 380 AND X > 365 AND
      Y > 278 AND Y < 290 THEN MX =
      369:MY = 281:CC$ = "AA": GOSUB
      10500: GOTO 6000
4020  IF Y < 3.5 * X - 945 AND Y >
      .7 * X AND Y >  - .9 * X + 5
      97 THEN  GOSUB 11350: GOTO 6
      200
4030  IF Y > 1.5 * X - 170 THEN D
      R$ = "NW": GOSUB 11300: GOTO
      6200
4040  GOSUB 10800: GOTO 6200
4100  IF X > 315 AND X < 325 AND
      Y < 341 AND Y > 330 THEN  GOSUB
      10500: GOTO 6200
4110  IF Y < 1.21 * X + 35 AND Y >
      1.25 * X THEN  GOSUB 11750: GOTO
      6200
4120  IF Y > .44 * X + 182 AND Y <
      .45 * X + 195 AND X < 370 THEN
       GOSUB 11150: GOTO 6200
4130  IF Y > .3 * X + 285 THEN  GOSUB
      11600: GOTO 6200
4140  IF Y >  = .2 * X + 320 THEN
      DR$ = "N": GOSUB 11300: GOTO
      6200
4150  IF Y > .2 * X + 304 THEN  GOSUB
      11500: GOTO 6200
4160  IF Y >  = .45 * X + 196 THEN
      DR$ = "S": GOSUB 11300: GOTO
      6200
4170  IF Y > X - 20 THEN DR$ = "N
      ": GOSUB 11300: GOTO 6200
4180  IF Y < 1.2 * X - 125 THEN  GOSUB
```

```
     11350: GOTO 6200
4190  GOSUB 10800: GOTO 6200
4200  IF X > 370 AND X < 383 AND
     Y > 432 AND Y < 448 THEN  GOSUB
     10500: GOTO 6200
4210  IF Y > 490 AND X > 380 THEN
      GOSUB 10500: GOTO 6200
4220  IF Y < 1.2 * X + 35 AND Y >
     1.25 * X - 5 THEN 4245
4230  IF X < 351 AND Y < .18 * X +
     358 THEN  GOSUB 11600: GOTO
     6200
4240  GOSUB 10800: GOTO 6200
4245  IF Y > 460 THEN  GOSUB 1070
     0: GOTO 6200
4246  IF Y > 450 THEN MX = 356:MY
      = 455:CC$ = "EE":ZNE = 2: GOSUB
     11650: GOTO 6000
4247  IF Y > 416 THEN  GOSUB 1165
     0: GOTO 6200
4248  GOSUB 11750: GOTO 6200
4300  IF Y < 510 AND X > 379 THEN
     CC$ = "BB":MX = 384:MY = 506
     : GOSUB 10500: GOTO 4355
4310  IF Y < 1.2 * X + 35 AND Y >
     1.25 * X - 5 THEN  GOSUB 107
     00: GOTO 6200
4320  IF X > 322 AND X < 328 AND
     Y > 876 - X AND Y < 559 THEN
      GOSUB 11000: GOTO 6200
4330  IF X > 310 AND X < 340 AND
     Y > 540 AND Y < 570 THEN MX =
     325:MY = 560:CC$ = "AB": GOSUB
     10500: GOTO 6000
4340  IF Y > 538 THEN  GOSUB 1090
     0: GOTO 6200
4350  GOSUB 10800: GOTO 6200
4355  IF Y > 502 THEN 6000
4356  GOTO 6200
4400  IF X > 480 AND X < 490 AND
     Y > 25 AND Y < 339 THEN  GOSUB
     10500: GOTO 6200
4410  GOSUB 10950: GOTO 6200
4500  IF Y > 145 AND Y < 160 AND
     X > 405 AND X < 425 THEN  GOSUB
     10500: GOTO 6200
4510  IF Y >  - .02 * X + 160 AND
     Y <  - .02 * X + 170 AND X <
     420 THEN  GOSUB 11050: GOTO
     6200
4520  IF Y < X - 100 AND Y < 170 AND
     X > 420 THEN  GOSUB 10850: GOTO
     6200
4530  GOSUB 10800: GOTO 6200
4600  IF Y < 2.2 * X - 680 AND Y >
     2.5 * X - 845 AND Y > 252 AND
     Y < 277 THEN  GOSUB 10500:MX
      = 432:MY = 255:CC$ = "BF": GOTO
     4625
4610  IF Y > 223 AND Y < 248 AND
     X > 410 AND X < 441 THEN  GOSUB
     11800: GOTO 6200
4620  GOSUB 10800: GOTO 6200
4625  IF Y < 260 THEN 6000
4626  GOTO 6200
4700  IF X < 427 AND X > 418 AND
     Y > 365 AND Y < 379 THEN MX =
     422:MY = 368:CC$ = "BD": GOSUB
     10500: GOTO 6000
4710  IF Y < .2 * X + 320 AND Y >
     .2 * X + 306 THEN  GOSUB 115
     00: GOTO 6200
4720  IF X > 412 AND X < 423 AND
     Y > 330 AND Y < 342 THEN DR$
      = "NW": GOSUB 11100: GOTO 6
     200
4725  IF Y > 1.4 * X - 270 AND Y <
     .78 * X + 30 AND Y <  - X +
     822 THEN  GOSUB 11350: GOTO
     6200
4730  IF Y > .25 * X + 265 THEN D
     R$ = "S": GOSUB 11300: GOTO
     6200
4740  IF X > 480 AND Y > 320 THEN
     DR$ = "W": GOSUB 11300: GOTO
     6200
4750  GOSUB 10800: GOTO 6200
4800  IF Y < .2 * X + 320 AND Y >
     .2 * X + 304 THEN 4825
4810  IF X > 450 AND Y < 444 THEN
      GOSUB 10950: GOTO 6200
4820  GOSUB 10800: GOTO 6200
4825  IF X > 450 THEN  GOSUB 1155
     0:MX = 450:MY = 405:CC$ = "E
     G":ZNE = 2: GOTO 4827
4826  GOTO 4810
4827  IF X < 460 GOTO 6000
4828  GOTO 6200
4900  IF Y < 1.21 * X + 35 AND Y >
     1.25 * X - 5 THEN 4935
4910  IF Y > 534 AND Y < 565 AND
     X > 450 AND X < 485 THEN  GOSUB
     10500: GOTO 6200
4920  IF Y > 534 THEN  GOSUB 1090
     0: GOTO 6200
4930  GOSUB 10800: GOTO 6200
4935  IF Y > 534 THEN  GOSUB 1075
     0: GOTO 6200
4936  GOSUB 10700: GOTO 6200
5000  IF X > 498 AND X < 512 AND
     Y > 72 AND Y < 83 THEN MX =
     505:MY = 80:CC$ = "JR": GOSUB
     10500: GOTO 6000
5010  GOSUB 10950: GOTO 6200
5100  GOSUB 10800: GOTO 6200
5200  IF X > 510 AND X < 522 AND
     Y > 211 AND Y < 222 THEN  GOSUB
     10500: GOTO 6200
5220  IF X > 559 AND X < 575 AND
     Y > 280 AND Y < 295 THEN MX =
      = 565:MY = 287:CC$ = "EH": GOSUB
     10500: GOTO 6000
5225  IF Y > 1.33 * X - 480 AND Y
      < 1.3 * X - 441 AND Y >  -
     .8 * X + 635 THEN  GOSUB 107
     00: GOTO 6200
5230  GOSUB 10800: GOTO 6200
5300  IF Y > 1.33 * X - 483 AND Y
      < 1.3 * X - 442 THEN  GOSUB
     10700: GOTO 6200
5320  IF X > 531 AND X < 540 AND
     Y > 338 AND Y < 351 THEN  GOSUB
     11000: GOTO 6200
5330  IF X > 523 AND X < 550 AND
     Y > 330 AND Y < 358 THEN MX =
     530:MY = 352:CC$ = "AC": GOSUB
     10500: GOTO 5345
5335  IF Y >  - 3.2 * X + 2046 AND
     Y > .3 * X + 185 THEN  GOSUB
     10850: GOTO 6200
5340  GOSUB 10800: GOTO 6200
5345  IF Y > 345 THEN 6000
5346  GOTO 6200
5400  IF Y < .2 * X + 320 AND Y >
     .2 * X + 304 THEN 5425
5410  IF Y < 450 AND X < 550 THEN
      GOSUB 10950: GOTO 6200
5420  GOSUB 10800: GOTO 6200
5425  IF X < 551 THEN  GOSUB 1155
     0: GOTO 6200
5426  GOSUB 11500: GOTO 6200
5500  IF Y > 536 THEN  GOSUB 1090
     0: GOTO 6200
5510  GOSUB 10800: GOTO 6200
```

Input runner's reactions to new location and respond appropriately.

```
6000  IF SR = 0 THEN SR =  - 1: GOTO
     6200
6001 SR =  - 1: PRINT "YOU HAVE S
     IGHTED A MARKER!!": PRINT "G
     O TO IT OR NOT? (G/N) ";: GET
     A$: PRINT A$
6030  IF A$ = "N" THEN 6200
6031  IF A$ <  > "G" THEN  CALL  -
     1052: GOTO 6001
6039  HOME
6040 D =  SQR ((X - MX) ^ 2 + (Y -
     MY) ^ 2):X = MX:Y = MY
6041  PRINT "THE CONTROL CODE IS
     ";CC$: PRINT "DO YOU WANT TO
      PUNCH IN?": PRINT "(Y/N)";:
      GET A$: PRINT A$: PRINT
```

continued on next page

continued from previous page

```
6090  IF A$ = "N" THEN 6100
6092  IF A$ <  > "Y" THEN 6041
6094 TS = TS + 3:CARD$ = CARD$ +
     CC$
6095  FOR CM = 1 TO  LEN (CARD$) STEP
     2:CP$ =  MID$ (CARD$,CM,2):C
     T =  - 2:CP = 1
6096 CT = CT + 3
6097  IF CP$ <  > MID$ (TEST$,CP
     ,2) THEN CP = CP + 2: GOTO 6
     100
6098  VTAB 24: HTAB CT: INVERSE :
      PRINT  MID$ (TEST$,CP,2);: NORMAL
6099  NEXT CM: GOTO 6104
6100  IF CP >  LEN (TEST$) THEN 6
     099
6101  GOTO 6096
6104  HOME : IF ZNE = 1 THEN  GOSUB
     10500: GOTO 6130
6105  GOSUB 10000
6130  IF  LEN (CARD$) > 7 THEN  PRINT
     "ENTER A TRAVEL BEARING OF 9
     99": PRINT "IF THIS WAS YOUR
      LAST MARKER": PRINT
6140 SR = 0: GOTO 6510
6200 SR = SR + 1: ON SP GOTO 6220
     ,6230,6240,6250,6260
6220 A$ = " VERY SLOW": GOTO 6265

6230 A$ = " SLOW": GOTO 6265
6240 A$ = " NORMALLY": GOTO 6265
6250 A$ = " FAST": GOTO 6265
6260 A$ = " VERY FAST"
6265 BG = TB - DER: IF BG < 0 THEN
     BG = 360 + BG
6270  PRINT "YOU'VE BEEN RUNNING"
     ;A$;" ON A": PRINT "BEARING
     OF ";BG: PRINT "AND HAVE GON
     E ";AD%;" METERS": PRINT "CO
     NTINUE LIKE THIS (Y/N) ";: GET
     A$: PRINT A$
6271  PRINT
6300  IF A$ = "N" THEN 6500
6302  IF A$ = "P" THEN 8500
6303  IF A$ <  > "Y" THEN 6270
6330  IF VERT <  > 0 THEN Y = Y +
     VERT * 6.562: GOTO 850
6340  GOTO 800
6500  HOME
6510  PRINT "ENTER A NEW BEARING
     ": GOTO 220
```

No directional error assigned because runner is following a linear feature.

```
7000 DER = 0: GOTO 650
```

Print control cards and end the run.

```
8000  HOME
8001 PY = 0
8005  PRINT : PRINT
8007  PRINT  TAB( 14);"CONTROL CA
     RD"
8010  PRINT
8020  FOR CM = 1 TO  LEN (CARD$) STEP
     2
8022 CM$ =  MID$ (CARD$,CM,2): IF
     CM$ = "**" THEN PY = PY + 1
8024 CT = 1
8026  IF CM$ <  > MID$ (TEST$,CT
     ,2) THEN CT = CT + 2: GOTO 8
     100
8030  INVERSE
8050  PRINT CM$;"  ";
8060  NORMAL : NEXT CM: GOTO 8120

8100  IF CT >  LEN (TEST$) THEN 8
     050
8105  GOTO 8026
8120  PRINT : PRINT
8130 PY = PY * 10: PRINT "ADD ";P
     Y;" MINUTES FOR DIVINE": PRINT
     "INTERVENTION!"
8140  POKE  - 16368,0: WAIT  - 16
     384,128
8150  TEXT : HOME
8155  POKE  - 16368,0
8160  PRINT : PRINT : PRINT
8170  PRINT "ANOTHER TRY? (Y/N) "
     ;: GET A$: PRINT A$
8175  IF A$ = "Y" THEN 90
8199  PRINT : PRINT : PRINT : PRINT

8200  VTAB 18: HTAB 15: INVERSE :
      PRINT "BYE - BYE": NORMAL :
      END
```

Divine guidance from Silvus.

```
8500  REM
8549  SPEED= 127
8550  HOME : FOR I = 1 TO 400: NEXT
     I: PRINT "THERE IS A SUDDEN
     STILLNESS IN THE WORLDAROUND
      YOU.....": PRINT : FOR I =
     1 TO 650: NEXT I: HOME
8560  PRINT "YOU HEAR A VOICE"
8561  PRINT "IT COMES FROM NOWHER
     E, BUT IS EVERYWHERE": PRINT

8562  INVERSE : PRINT "I AM SILVU
     S - GOD OF ORIENTEERING": NORMAL
     : FOR I = 1 TO 750: NEXT I
8563  PRINT : PRINT "YOU ARE AT "
     ; INT (X + .5);","; INT (Y +
     .5): PRINT
8564  FOR I = 1 TO 2000: NEXT I:C
     ARD$ = CARD$ + "**"
8565  HOME : SPEED= 255: GOTO 190
```

Subroutine to update accumulated time and fatigue error.

```
10000 CV =  PEEK (37)
10010 F = 1.8: GOTO 10050
10015 F = 1.5: GOTO 10050
10020 F = 1.29: GOTO 10050
10025 F = 1.13: GOTO 10050
10030 F = 1
10050 T% = F * MSP * D:F = SP * S
     P * T% / 7000:TS = TS + T%:M
      =  INT (TS / 60):TM = TM +
     M:TS = TS - 60 * M:FER = FER
      + F:F% = FER
10052  VTAB 21: HTAB 1: PRINT "AC
     CUMULATED TIME"; TAB( 20);"D
     IST"; TAB( 26);"FATIGUE ERRO
     R";
10053  VTAB 22: HTAB 1: PRINT TM;
     " MIN ";TS;" SEC"; TAB( 20);
     DIST;" M"; TAB( 28);F%;
10099  VTAB CV + 2: HTAB 1
10130 ZNE = 1: RETURN
```

Series of one-line subroutines to print description of runner's location.

```
10500  PRINT "YOU HAVE ORIENTED Y
     OURSELF": PRINT "AT COORDINA
     TES "; INT (X + .5);","; INT
     (Y + .5): PRINT : GOSUB 1000
     0: RETURN
10600  PRINT N$;" BOULDER FIELD":
      PRINT :MSP = .55: GOTO 1000
     0
10650  PRINT BY$;" DRY STREAM": PRINT
     :MSP = .5: GOTO 10000
10700  PRINT O$;" GRAVEL ROAD": PRINT
     :MSP = .38: GOTO 10000
10750  PRINT O$;" GRAVEL ROAD IN
     A MEADOW": PRINT :MSP = .38:
      GOTO 10000
10800  PRINT O$;" LEVEL AREA": PRINT
     :MSP = .45: GOTO 10000
10850  PRINT N$;" MARSH": PRINT :
     MSP = 1: GOTO 10000
10900  PRINT N$;" MEADOW": PRINT
     :MSP = .4: GOTO 10000
10950  PRINT N$;" PINE WOODS": PRINT
     :MSP = .5: GOTO 10000
```

```
11000  GOSUB 10500: PRINT "(I SUR
    E HOPE YOU CAN SWIN)": PRINT
    :MSP = 4: RETURN
11050  PRINT BY$;" RAIL FENCE": PRINT
    :MSP = .45: GOTO 10000
11100  PRINT N$;" REENTRANT - ";D
    R$;" IS UPHILL": PRINT :MSP =
    .5: GOTO 10000
11150  PRINT O$;" RIDGE": PRINT :
    MSP = .45: GOTO 10000
11200  PRINT BY$;" ROCK FACE": PRINT
    :MSP = .8: GOTO 10000
11250  PRINT N$;" SADDLE": PRINT
    :MSP = .55: GOTO 10000
11300  PRINT O$;" SLOPE - ";DR$;"
     IS UPHILL"
11301  PRINT
11310  IF DR$ = "N" THEN UB = 0
11311  IF DR$ = "NE" THEN UB = 45

11312  IF DR$ = "E" THEN UB = 90
11313  IF DR$ = "SE" THEN UB = 12
    5
11314  IF DR$ = "S" THEN UB = 180

11315  IF DR$ = "SW" THEN UB = 22
    5
11316  IF DR$ = "W" THEN UB = 270

11317  IF DR$ = "NW" THEN UB = 31
    5
11320 DB =  ABS (TB - UB): IF DB >
    180 THEN DB = 360 - DB
11330  IF DB < 90 THEN MSP = .55 -
    DB / 45 * .05: GOTO 10000
11331 MSP = .45: GOTO 10000
11350  PRINT O$;" SMALL HILL": PRINT
    :MSP = .55: GOTO 10000
11400  PRINT O$;" STEEP SLOPE - "
    ;DR$;" IS UPHILL": PRINT :MS
    P = .8: GOTO 10000
11450  PRINT O$;" SPUR": PRINT :M
    SP = .45: GOTO 10000
11500  PRINT BY$;" STREAM": PRINT
    :MSP = .5: GOTO 10000
11550  PRINT BY$;" STREAM IN A PI
    NE WOODS": PRINT :MSP = .5: GOTO
    10000
11600  PRINT N$;" THICK BRUSH ARE
    A": PRINT :MSP = 2.1: GOTO 1
    0000
11650  PRINT O$;" TRAIL": PRINT :
    MSP = .4: GOTO 10000
11750  PRINT O$;" TRAIL IN THICK
    BRUSH": PRINT :MSP = .4: GOTO
    10000
11800  PRINT N$;" FIELD OF SMALL
    DEPRESSIONS": PRINT :MSP = .
    5: GOTO 10000
```

The One Liner Challenge

by Bob Howell

Editor's Note: The first one-liner was submitted to SoftSide by Arne Rohde of Denmark in July, 1980. Since then, they have become a standing tradition in the magazine.

Almost anyone can write big, multi-line BASIC programs with fancy graphics, colors and sounds. Thousands of programmers are doing it every day. You have probably written a few yourself. But, in case you missed it, the real programming challenge lies in writing what is called a "one-liner." To write a one-liner that does something meaningful or displays something creative can be a very interesting and rewarding task. I am going to explain a little bit about the process of writing a one-liner. As a result, I hope you will take this information, add a little of your own creativity and produce some interesting one-liners for all **SoftSide** readers to appreciate. (Note: the examples I use are from Atari BASIC but the concepts can be applied to any computer.)

Just what is a one-liner? A one-liner is defined as a complete, self-contained computer program, written in BASIC, that consists of one logical line with one and only one statement number (for instance, to provide a continuously changing graphics display). This is a concept that was introduced in the pages of **SoftSide** in the summer of 1980. Most versions of BASIC allow several statements to appear on one line and allow one logical line to consist of several physical lines on the screen. This means that a one-liner can consist of a large number of BASIC statements and can be quite complex. However, it must be completely self-contained on one BASIC line and cannot assume anything. No assumptions can be made, for example, on what graphics mode the computer is in, whether or not the screen is clear, that the cursor is in a certain position, or that particular variables have been initialized to certain values. All of these things must be taken care of within the program itself.

How big can a one-liner be? That depends on the computer. On the Atari, for example, a physical line (ie., one line on the screen) can contain a maximum of 40 characters. But a logical line (one line in BASIC) may consist of up to three physical lines. Therefore, a one-liner in Atari BASIC may consist of everything you can pack into a 40 X 3 or 120 characters. On the S-80 and Apple computers, a one-liner may contain up to 255 characters.

How do you go about writing a one-liner? First, you must get an idea. Go over in your mind the different graphics modes available on your computer, its color and sound options, its graphic character set, etc., and develop an idea for a program that you think might qualify as a one-liner. At this point, all you need is the idea. Begin to program this idea, one statement per line, just as if you were writing a regular multi-line BASIC program.

For example, suppose you decide to place characters from your computer's ASCII character set randomly on the screen. This may not seem very creative to you, but for the purpose of this article, I am going to use it as my example as I don't want to give away any creative one-liners (that's the ingredient you are required to add to the task.) Here is an Atari program that actually runs and will place characters (from the full 255 ASCII character set) randomly on the screen:

```
10   GRAPHICS 0
20   POKE 82,0
30   FOR I=1 TO999999999
40   SETCOLOR2,
     INT(RND(0)*16),0
50   FOR J=1 TO 50
60   X=INT(RND(0)*40)
70   Y=INT(RND(0)*24)
80   C=INT(RND(0)*256)
90   POSITION X,Y
100  PRINT CHR$(C);
110  SOUND O,C,10,8
120  NEXT J
130  NEXT I
```

No you can see why I suggest that you first write the program, one statement per line. Your first attempt at writing the one-liner usually will not fit onto one line and it's much easier to debug the program in this form.

If you count the number of characters occupied by this program (including spaces), you will find it is 233 characters long. Since the Atari allows a maximum of 120 characters for a one-liner, you may be tempted at this point to give up. Don't! I will show you how to turn this program into a legitimate Atari one-liner occupying 120 characters or less. But first, I'll explain each program line so that any non-Atari readers can understand what the program is doing:

Line 10 — Puts the computer into full screen text mode and clears the screen. (Note: Although this is the default mode, you cannot assume the computer will be in this mode.)

Line 20 — Extends the left margin on each line from the default character position of 2 to character position 0 in order to allow use of a full 40 characters per line.

Line 30 — Puts the program into an essentially infinite loop.

Line 40 — Sets the background color randomly to one of Atari's 16 colors.

Line 50 — Inner loop to generate 50 random characters for each background color change.

Lines 60 to 70 — Generate X and Y coordinates of a random point on the screen (0 to 39 character positions per line, 0 to 23 line positions per screen.)

Line 80 — Generates a random ASCII character code (0 to 255.)

Line 90 — Positions the cursor to the random point on the screen.

Line 100 — Prints the random character and leaves the cursor positioned there.

Line 110 — Turns on a different sound for each character.

Lines 120 to 130 — End of FOR loops.

As you can see, this is a fairly simple BASIC program.

Once the program is written, one statement per line, and is debugged and running properly, it is time to condense it to one line. Note that in order to save space, only one character variable name is used. Also, all blanks are eliminated (except in strings) as they serve no purpose and take up valuable space. Most BASICs will allow blanks to be eliminated and statements to be squeezed together. However, there is an exception in Atari BASIC. The statement:

```
FORI=1TO2*NSTEP2
```

will not be correctly interpreted because NSTEP2 is considered to be a variable. Therefore, a space must be left between N and STEP or the statement must be rewritten as follows:

```
FORI=1TON*2STEP2
```

Atari BASIC will handle this statement because the last character before the

keyword STEP is numeric.

Once blanks are eliminated from our random character program, its length is reduced from 223 to 196 characters. A few more improvements can easily be made. For example, the INT functions may be eliminated since Atari BASIC will round up to an integer value where an integer is expected. Therefore, the statement:

```
60 X=INT(RND(0)*40)
```

which yields values of 0 <=X <=39 may be shortened to

```
60 X=RND(0)*39
```

to yield values of 0.0 <= X< = 38.9999999. In the POSITION statement (Line 90), X will be rounded to 0 <= X <= 39 giving the required result. Also, statements 60 and 70 may be eliminated and the RND calculations placed right on statement 90. Thus we have made the following changes in statements 40 through 90:

```
40 SETCOLOR 2,
RND(0)*15,0
50 FOR J=1 TO 50
80 C=RND(0)*255
90 POSITION
RND(0)*39,RND(0)*23
```

and have further reduced the program's size from 196 to 166 characters.

Once we have condensed the program as much as possible, the next step is to abbreviate. Most BASICs allow an abbreviation to be used instead of spelling out the full BASIC keyword. By using abbreviations, several more statements may be placed on a logical line. Following are the most common Atari BASIC keywords that may be abbreviated, followed by their abbreviations:

Keyword	-	Abbr.
CLOSE	-	CL.
COLOR	-	C.
DATA	-	D.
DRAWTO	-	DR.
FOR	-	F.
GOSUB	-	GOS.
GOTO	-	G.
GRAPHICS	-	GR.
INPUT	-	I.
LOCATE	-	LOC.
NEXT	-	N.
OPEN	-	O.
PLOT	-	PL.
POSITION	-	POS.
PRINT	-	? or PR.
RESTORE	-	RES.
SETCOLOR	-	SE.
SOUND	-	SO.

Here is what our random character program looks like with the changes we made, all blanks eliminated, keywords abbreviated and placed on one line, 40 characters per line:

```
1REMGR.0:POKE82,0:F.I=1TO
999999999:SET.2,RND(0)*15,0:F.J=
1TO50:C=RND(0)*255:POS.
RND(0)*39,RND(0)*23:?CHR$(C);:S
O.0,C,10,8:N.J:N.I
```

The program is now down to 126 characters (excluding the REM) and almost fits into one line. We are very close to having a new one-liner! You may have noticed that the one-liner has been included behind a REMark keyword. There is a very good reason for doing this. Suppose we had entered it into Line 1 without the REM, then found out when we executed it that some changes were required. When we LIST Line 1 on the screen, we would see that Atari BASIC had eliminated our abbreviations and expanded the keywords back out to their original length. Furthermore, we would find the spaces we removed had been reinserted into the line. This is due to the way Atari BASIC stores statements in memory in a tokenized form and for ease of program readability. Therefore, to make a simple modification to the one-liner, we must first go back through, reabbreviate everything and eliminate the spaces again. Then we must type back onto the third physical line the last part of the one-liner which overflowed onto the fourth (and possibly the fifth) physical line when the statement was expanded for LISTing on the screen. In order to avoid this statement rework each time a minor change is made, the one-liner is included as a REMark statement. When stored in memory this way, since it is a REMark, BASIC does not tokenize it and it remains unmodified when LISTed on the screen. In order to make a change and retest the one-liner, we simply do the following:

1. LIST the one-line program.
2. Make the change and reenter the one-liner into memory, keeping it as a REMark.
3. Using the screen editor, delete the first four characters from the statement on the screen (ie., delete 1REM).
4. Press the RETURN key with the statement number and REM missing in order to execute the one-liner. (Note: By removing the statement number, the statement is executed in direct mode right from the screen without having to type RUN.)

By repeating steps 1 through 4, the one-liner may easily be modified and debugged. When debugging is complete and the one-liner working properly, just eliminate the REM and enter the statement into memory on Line 1 as a standard BASIC statement.

Now back to our example. If we eliminate the REM, we see it is still six characters too long. We can solve this problem by looking at the first FOR statement. There are three forms of the FOR statement which will put the one-liner into an infinite loop. They are:

1. FOR I = 1 TO 999999999
2. FOR I = 1 TO 2 STEP 0
3. FOR I = 1 TO 1E9

Obviously, the third form is the shortest in length. If we substitute 1E9 for the 999999999, our one-liner is reduced to 120 characters and now truly fits into one line.

Here are two more hints which may be useful in writing one-liners:

1. To alternate the value of a variable between 0 and 1 use:

```
A=ABS(A-1)
```

This statement can be used in conjunction with the statement:

```
ON A GOTO 1
```

to alternate between continuing forward (A=0) or going back to the beginning of the one-liner (A=1).

2. Some useful POKEs are:

a. POKE 77,0 — Prevents the computer from going into attract mode (random color switching.) This POKE must be executed at least once every nine minutes.

b. POKE 82,N — Sets left margin for text to character position N (default is N=2).

c. POKE 83,N — Sets right margin for text to character position N (default is N=39).

d. POKE 559,0 — Turns off screen and speeds up CPU (screen goes black.)

POKE 559,34 — Turns screen back on.

e. POKE 752,0 — Turns cursor on.

POKE 752,1 — Turns cursor off.

f. POKE 755,4 — Prints text upside down on the screen.

POKE 755,2 — Prints text in normal form on the screen.

g. POKE 766,0 — Executes control characters when printed to the screen (ie., perform their function).

POKE 766,1 — Prints control characters on the screen (instead of performing their function.)

As you see, writing one-liners can be a lot of fun and a real challenge. We started with a 223 character, 13 statement program and were able to reduce it to a 120 character one-liner. Here is what the final 120 character one-liner looks like:

```
1gr.0:POKE82,0:F.I=1TO1E9:S
ET.2,RND(0)*15,0:F.J=1TO50:
C=RND(0)*255:POS.RND(0)*39
,RND(0)*23:?CHR$(C);:SO,0,C,
10,8:N.J:N.I
```

continued on page 81

SPORTS FANS!

MASTERS' GOLF

written by David Bohlke

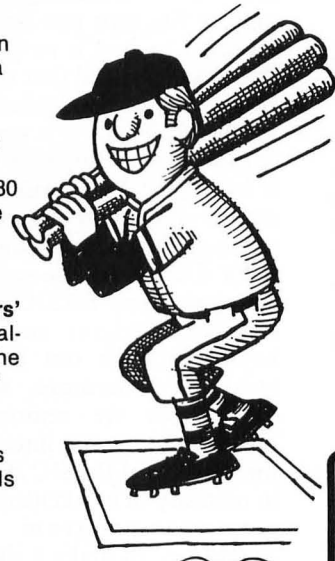
It's the middle of January, and there's at least two feet of snow on the ground. Lee Trevino is out on a golf course, but you're not. You could follow the sun in pursuit of your favorite pastime, but that can be quite expensive...

Don't despair! Power up your S-80 or Atari and prepare to tee off. The multi-talented Dave Bohlke has created these golfing programs to help you get over those missing links blues. For 1-4 players, **Masters' Golf** (S-80 version published originally as "Protour Golf") provides all the excitement of a nine hole round of golf, including full club selection, sand traps, and contoured greens. The money you save on caddy fees is enough to pay for endless rounds of golf played right in your own home.

S-80 16K Tape $9.95
Atari 8K Tape..... $9.95

WORLD SERIES

written by David Bohlke



Ah yes, it's springtime and a young man's thoughts turn to...baseball? How would you like a baseball season with no threats of player strikes or free agent negotiations that leave the shattered remnants of once-mightly teams strewn about the playing fields? We offer you your own league...Apple, Atari or S-80. Batter up!

S-80 16K Tape $9.95
Apple 16K Tape..... $9.95
Apple 32K Disk..... $14.95
Atari 16K Tape $9.95

RAMWARE

SoftSide Selections
6 South Street Milford NH 03055

END ZONE II

by Roger W. Robitaille, Sr.

Rough and tumble gridiron action, from the toss of the coin to the 2-minute warning... Four 15-minute quarters, provisions for interceptions, touchbacks, timeouts, fumbles, penalties. Everything except the cheerleaders.

S-80 16K Tape
..... $14.95

S-80 32K Disk
..... $15.95

SoftSide Selections
6 South Street Milford NH 03055

One Liner Challenge

continued from page 79

A true one-liner, self-contained, complete with color, sound and action on the screen! To summarize, here are the steps to use in writing one-liners:

1. Study your computer's graphics, color, sound and character set capabilities and develop an idea.
2. Implement this idea in a computer program, one statement per line.
3. Debug the program and get it running properly.
4. Reduce its length as much as possible (repeating step 3 as often as necessary.)
5. Squeeze out all spaces, abbreviate BASIC keywords and place the program on line 1 as a REMark. (Atari owners first POKE 82,0 to allow use of the full 40 characters per line.)
6. Do final debugging, modifying and program shrinking by keeping the program stored as a REMark. Execute it in direct mode after removing the REM and statement number. (Note: This step may push you to the limits of your programming ability and beyond!)
7. When the program fits into one line and is working properly, remove the REM (but not the statement number) and store it in memory in final form.
8. Make final test runs.
9. Send the one-liner on tape or disk to:

SOFTSIDE PUBLICATIONS
One-Liners
PO Box 68
Milford, N.H. 03055

so we may all enjoy your creation.

At the end of this article, you will find six additional one-liners for an Atari computer that I wrote using this method. Run them, study the code, learn how they work and you will see that quite a complex program can be written on one line. You now have a method to write one-liners, you have examples, you know it is possible and you certainly have the ability to write them! What are you waiting for?

Color Wheel

```
1 GRAPHICS 11:SETCOLOR 2,0,0:DEG :FOR
I=879 TO 2663:X=29*COS(I)+39:Y=95*SIN(
I)+95:COLOR I/7:PLOT X,Y:DRAWTO 79-X,1
91-Y:PLOT X,191-Y:DRAWTO 79-X,Y:NEXT I
```

Geometric Polygons

```
1 GRAPHICS 24:COLOR 1:DEG :A=RND(0)*10
:PLOT 159,0:FOR I=270 TO 990/A*25 STEP
 360/A+A:DRAWTO 95*COS(I)+159,95*SIN(I
)+95:NEXT I:FOR I=1 TO 5000:NEXT I:GOT
```

Sunburst

```
1 GRAPHICS 11:SETCOLOR 2,0,0:SETCOLOR
4,0,0:Y=191:Z=79:FOR I=0 TO Y:COLOR RN
D(0)*Z:J=I/Y*Z:PLOT J,0:DRAWTO Z-J,Y:P
LOT Z,I:DRAWTO 0,Y-I:NEXT I:FOR I=0 TO
 10000:NEXT I:GOTO 1
```

Space Chase

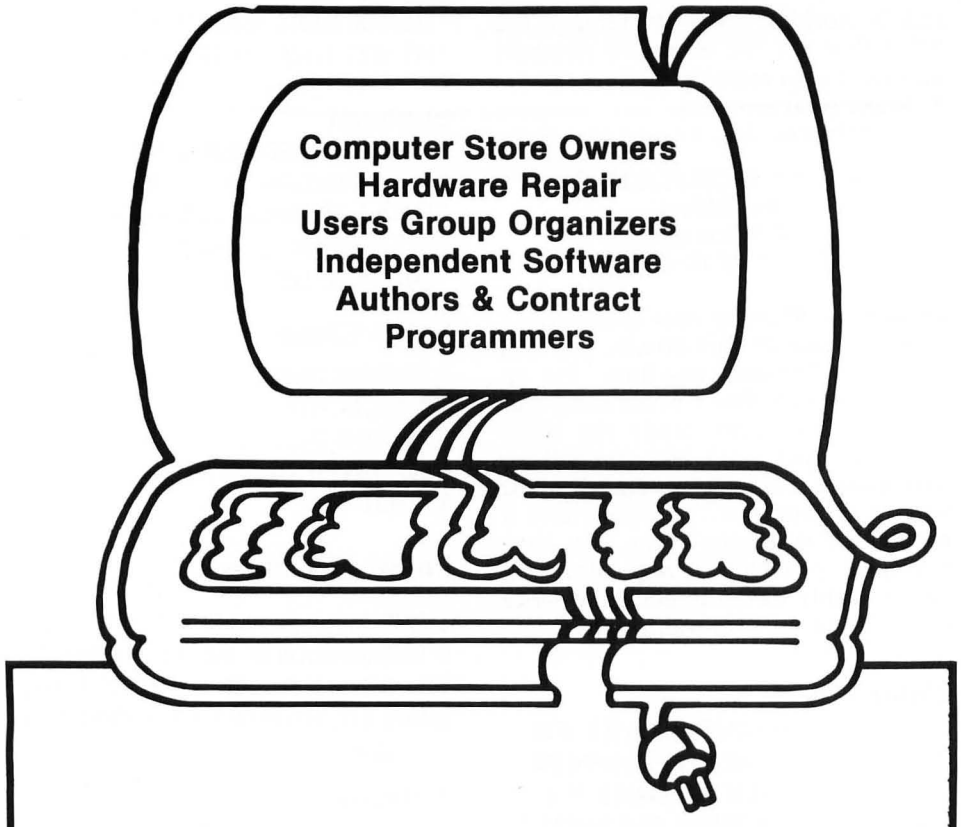
```
1 GRAPHICS 24:FOR I=0 TO 1000000000:SO
UND 0,H,10,4:COLOR 0:PLOT A,B:DRAWTO C
,D:A=C:B=D:C=E:D=F:E=G:F=H:G=R*319:R=R
ND(0):H=R*191:COLOR 1:PLOT E,F:DRAWTO
G,H:NEXT I
```

Forest of Colors

```
1 GRAPHICS 11:SETCOLOR 2,0,0:FOR I=1 T
O 1000000000:COLOR RND(0)*16:X=RND(0)*
78:Y=RND(0)*184:PLOT X,Y:DRAWTO X,Y+7:
DRAWTO X+1,Y+7:DRAWTO X+1,Y:SOUND 0,X+
Y,10,8:NEXT I
```

Polygons

```
1 GRAPHICS 8:COLOR 1:DEG :A=INT(RND(0)
*24)+2:? ,A;" POINTS":PLOT 159,0:FOR I
=270 TO 990/A*25 STEP 360/A:DRAWTO 79*
COS(I)+159,79*SIN(I)+79:NEXT I:GOTO 1
```



Do you offer products or services that should be of interest to the computer hobbyists in your area? OR, do you possess the skills to repair computers, offer consultation, organize users groups, but don't really know how to get started? Well, the first step is locating potential customers, efficiently and inexpensively. We at **SoftSide** can help.

Tap a responsive market with **SoftSide**'s mailing lists. We've compiled a list of close to 80,000 names of individuals who have bought computers, peripherals and software or who have inquired about these items through ads in national magazines. Now you can access selections from this list to advertise the services you now provide or are considering.

Most sources of lists require a minimum order of 5000 names. **SoftSide**'s list is available to you in groups of 1000, arranged by zip code, for $100.

We'll send you the list of people in your area on 4-up cheshire labels or on pressure-sensitive labels (for an additional $1 per 100), ready for you to mail with your promotional flyer to potential customers. Just let us know which area you wish to cover with your mailing.

Our lists are a good start toward advertising locally — you'll reach the people who will be responsive to your offers. It's more efficient than the Yellow Pages, less expensive than newspapers.

We'll put you in touch with the right people. Between our connections and your talents, we can make a great combination!

For a distribution of available groupings of names in your area, send a self-addressed stamped envelope to:

SoftSide Mailing Lists
6 South Street
Milford, NH 03055

Labels are authorized for one-time use only.

K-Byters

ANOTHER PROGRAMMING CHALLENGE

Last summer **SoftSide** began inviting its readers to submit "One Liners" — self-contained, single-line programs for the S-80, Apple, or Atari which would provide a continuously changing graphics display. The response has been excellent, and we're still looking for more submissions.

Now we have a new challenge for you as well: "K-Byters". A K-Byter is a BASIC program which fits into 1K (1024) bytes of program memory. There aren't any restrictions on the nature of the program, other than its size. It can be a graphics display, a game, a mini-adventure, or anything your imagination and programming skills can create.

Note that the program does not have to RUN in 1K of memory; it can use as much RAM for arrays, strings, graphics mapping, etc., as you need. We'd prefer that it be able to run in a 16K system, but this is not an absolute limit.

Here, then, are the official rules:

1. The program must be written for the Apple, S-80, or Atari, entirely in BASIC (although it may create and call Machine Language routines).

2. The program must occupy no more than 1024 bytes of memory before running.

3. The program must be submitted on tape or disk, accompanied by your name, address, phone number, and a brief written description of its operation.

4. The tape or disk will be returned only if accompanied by a self-addressed envelope with adequate postage AFFIXED (do not send money).

5. Winners will have their programs published in **SoftSide** and will receive certificates extolling their virtues as programming wizards, for all the world to see!

Send submissions to:

K-Byters, c/o **SoftSide**
6 South Street
Milford, NH 03055

Programming Hint

by Edward E. Umlor

Yes, there is a way to set MEMORY SIZE from BASIC, and here's how. You can even put it in your program.

There are two ways of finding out what values you will need for setting the memory size from within the program. First we will cheat and do it the easy way. Power up your system normally and default right down into BASIC (do not use DISK BASIC). PRINT PEEK(16561), PEEK(16562) and press ENTER to get the values of your full memory:

16561(LSB) 16562(MSB)
16K 255 127 LSB = least significant byte
32K 255 191 MSB = most significant byte
48K 255 255

You now have the address (in decimal) of the last byte of RAM. Now type "SYSTEM" ENTER and when the ?* appears type "/0" ENTER. This time, answer the MEMORY SIZE with the correct value. Peek at the two locations again and copy down their values. Now you can add a line at the very front of the program like this: 5 POKE 16561,(NEW VALUE):POKE 16562,(NEW VALUE). This will reset the two locations to reserve the upper memory as soon as you run the program.

Let's say you know how much space to reserve, but you don't know the memory size of the computer. This is a good method for doing it with a program you wish to sell. We need to reserve 549 bytes of upper memory.
1) Divide 256 into 549 = 2 with 37 remainder.
2) The results of 1 give you the values for the formula MSB-2,LSB-37. You can see that number in the MSB = 256 bytes and each number in the LSB = 1 byte.
3) Add this line at the very beginning of your program:
5A = PEEK(16561):B = PEEK(16562):
POKE16561,A-37:POKE16562,B-2:
A = 0:B = 0
4) Be sure to save your program and not loose all the good stuff you just put in.

I want to stress that the numbers in the two locations are decimal in nature. Each number of location 16561 is a byte of memory, and each number of location 16562 is 256 bytes of memory. Just remember, a known amount of memory can be converted to high address number and low address number by the formula: 256 will go into the required bytes MSB number with LSB number remaining.

VARPTR Again

by John T. Phillip, M.D.

The last of a three-part series on the S-80 VARPTR function.

Machine Language routines execute much faster than BASIC, and use less memory, so there are good reasons for mixing Machine Language with our BASIC. But why would we want to "pack" our Machine Language into strings?

BASIC uses the highest addresses of RAM for storage of strings and the stack. If we locate our Machine Language routine there, it will be written over by BASIC. To avoid that, we have to set MEMORY SIZE? to a value which BASIC will use as the upper limit of RAM, and put our routine in memory above that. The "protected memory" — from the address set in MEMORY SIZE up to the highest memory address actually in the computer — will not be used by BASIC. But we have to set the MEMORY SIZE every time we power up (in Level II), and every time we go from DOS to BASIC (disk system), or BASIC will write over our Machine Language routine!!

Inside a string is a protected area in the BASIC program itself where we can put our Machine Language routines. If we define a string such as A$="//////////", there are ten bytes (filled with slashes at present) which are part of the program text, and protected from BASIC. We can POKE anything we want into those ten bytes and it will be safe.

Not all Machine Language subroutines lend themselves to string packing. For an example, see Figure 1 for a subroutine to "white-out" the screen which was specifically written to be packed into a string.

A (brief) word of explanation for non-Machine Language programmers. Assembly Language (also known as the "source code") is the listing found in columns 4, 5, and 6, with comments in column 7. The instructions are in a form that can easily be read by humans: LD A,H (load the accumulator with whatever is in the H register), and so on. This is the form in which the routine is WRITTEN. Machine Language (also known as the "object code") is the listing of hexadecimal numbers in column 2. These are the actual instructions which the computer will EXECUTE. The machine instruction C9 hex may not say much to you (you'd rather see the Assembly Language instruction RET), but to the computer it says "Return" loud and clear. Assembly Language (for the humans to use) is translated into Machine Language (for the computer to use) by a program called an "assembler", and the process of translation is called "assembly" or "assembling the code".

Let's look at the conditions that must be met by a Machine Language routine before it can be packed into a string.

First, the routine must be less than 245 bytes long. When we packed strings with graphics, if there were more than 245 characters, we just POKEd the extra ones into a second string, and then PRINTed both strings. Machine Language programs, though, should be in consecutive bytes — and that means in one string — unless you have the expertise to tell the computer where to jump to find the start of the second string. In practice, this 245-byte limit isn't really much of a handicap. You can put a lot of Machine Language into 245 bytes (the average Machine Language instruction is only two or three bytes long), and most subroutines are much shorter.

Second, values of "0" and "34" decimal CANNOT be POKEd into a string. Well, they can be POKEd. . . any decimal value of 0 to 255 can be legally POKEd. . . but they wreak havoc when the string is read by BASIC. BASIC uses the number "0" as an "end of line" marker, and "34" (remember, CHR$(34) is the CHR$ code for a quotation mark) as the "end of string" marker. When BASIC comes across one of those values, that point is the end of the string as far as BASIC is concerned. All the instructions POKEd into the string after that point will be lost.

The solution seems simple — don't use instructions with the value of 0 or 34 (22 hex). "0" is the NOP (no operation) instruction, so that should be easy to avoid. "34" (22 hex) is the LD (NN),HL (load memory address NN with the contents of the HL register pair) instruction, which looks like one we could program around if necessary. But those aren't the only things we can't put into our routine. Since ANY "0" or "34" will stop our string in its tracks, we also can't load a register with data or an address that contains a "0" or "34". Look closely at the Machine Language listing (column 2 in Figure 1) of any routine you're considering packing. That column is the actual Machine Language in hexadecimal, and those are the values that will be POKEd into the string. If you see any "00" or "22" (34 in hex) there, you'll have to rewrite the routine or forget it!

A third consideration is that the

1	2	3	4	5	6	7
7F00		010		ORG	7F00H	; Start it at 20364 dec.
7F00	D9	020		EXX		; Save S-80 registers
7F01	21FF3F	030		LD	HL,3FFFH	; Last screen address
7F04	063C	040		LD	B,3CH	; Value for "end" test
7F06	36BF	050	LOOP	LD	(HL),OBFH	; BF hex = CHR$(191)
7F08	2B	060		DEC	HL	; Next left screen position
7F09	7C	070		LD	A,H	; High byte of screen positions
7F0A	B8	080		CP	B	; Screen full yet?
7F0B	30F9	090		JR	NC,LOOP	; LOOP if not yet
7F0D	D9	100		EXX		; Done, get back registers
7F0E	C9	110		RET		; Back to BASIC
0000		120		END		

Figure 1

routine to be "packed" must be completely "relocatable", which means that it will run, regardless of the address at which it starts. It's "relocatable" because it can be moved to a new place in memory and still work. Note the key words there — AND STILL WORK.

All Machine Language programs are not relocatable, since they may contain references to specific memory locations, such as absolute jumps (JP) and CALLs. Absolute jumps tell the computer to jump (like a GOTO in BASIC) to a specific address in memory. When the computer gets there, it expects to find the next instruction of the program. CALLs tell the computer to jump to an address, execute the subroutine found there, then return when it encounters a RET (like a BASIC GOSUB). But if the whole program has been moved in memory, the address jumped to may not contain the next instruction or the subroutine CALLed at all.

Look carefully at the jumps and CALLS of the Assembly Language listing (source code) of the routine you want to pack. If the majority of them are relative jump (JR) instructions, and the CALLs are only to addresses in the ROM (which are fixed between 0000 and 3000 hex and won't move if the program is moved) there's a good chance that the routine is relocatable.

Once you've decided that the Machine Language routine will meet the requirements for "packing"; length less than 245 bytes, no values of "0" or "34", relocatable code, and no subroutine CALLs except to the ROM — you STILL have to get it into the string, and then you have to be able to use it.

Getting the routine packed into a string, uses the same process that we used to pack strings with graphics. We define a "dummy" string with the same number of characters as our routine, put the values we want POKEd into a DATA statement, use VARPTR to find the beginning of the "dummy" string in memory, and then use a FOR/NEXT loop to READ the values and POKE them into the string.

If you look at the assembled listing of the "white-out" routine, you find the Machine Language in column 2. A DATA statement made of those values would be:

```
30 DATA D9,21,FF,3F,06,3C,36,BF,
2B,7C,B8,30,F9,D9,C9
```

Unfortunately, we can't POKE these values into a string, since POKE will only accept decimal values between 0 and 255. The hexadecimal numbers must be converted into POKE-able decimal numbers. It's the old business of the columns and powers of 16 coming back to haunt us one more time.

column $1x16^1(16)$ = "sixteens"
+ column $0x16^0(1)$ = "units" or "ones"

For example, if we want to convert D9 hex to decimal, we substitute the hex digits for the columns (remembering that A = 10, B = 11, C = 12, D = 13, E = 14, and F = 15):

D (13) x 16^1 (16) = 208
+ 9 (13) x 16^0 (1) = 9

which adds up to a total of 217 decimal. If we repeat that process for each one of the hexadecimal digits, we get these 15 decimal values to put into our DATA statement:

```
30 DATA 217,33,255,63,6,60,54,191,
43,124,184,48,249,217,201
```

Now let's define a "dummy" string and use VARPTR. The string has to be 15 characters long, because we need room to put the 15 Machine Language instructions that make up the routine.

```
10 A$ = "///////////////"
```

NOW we call on VARPTR. Exactly as for graphics, we can't POKE values into A$ until we know where it is in memory:

```
20 I = VARPTR(A$): J = PEEK(I + 1)
+ 256*PEEK(I + 2)
```

"J" contains the starting address of A$ in decimal, and the 15 slashes in A$ are stored in memory locations "J" to "J + 14". Now we READ the machine instructions from the DATA statement, and POKE them into the string:

```
25 FORK = JTOJ + 14: READX:
POKEK,X: NEXTK
```

RUN the program. When done, the string is packed with the Machine Language routine. Let's see if we can make the packing process more convenient. Converting the hexadecimal Machine Language into decimal for POKEing is cumbersome, but POKE requires decimal values, and most Machine Language listings are in hexadecimal. Why not let the computer read the DATA in hexadecimal, and then convert it to decimal before it's POKEd:

```
25 FORK = JTOJ + 14: READHX$:
HB$ = LEFT$(HX$,1):
LB$ = RIGHT$(HX$,1)
26 IFASC(HB$) > 64THENHB =
ASC(HB$)-55ELSEHB = VAL (HB$)
27 IFASC(LB$) > 64THENLB =
ASC(LB$)-55ELSELB = VAL (LB$)
28 POKEK,(HB*16 + LB): NEXTK
```

Watch out, though. To avoid conversion errors, all hexadecimal numbers MUST be written as two digits in the DATA statement: "0" must be 00, "A" must be 0A, and so on. Once the string is packed we can DELETE the lines used to "pack" it, lines 20, 25, 26 to 28 (if they were used), and 30. The routine is in A$, and they are no longer needed.

But here is a big difference between packing a string with graphics, and packing a Machine Language routine. After we packed graphics, we didn't have to know where the string was in memory to use it. All we had to do was PRINT A$, and there was the picture. To use a Machine Language routine, though, we must tell the computer where the routine starts in memory, so it can jump to that address and begin execution. The S-80 moves strings in memory, so we have to use VARPTR twice with Machine Language routines: once with the DATA statements to POKE the routine into the string, and once just before execution of the routine so the computer can be told the starting address for the jump.

The VARPTR routine which finds the start of the routine is used differently depending on whether you are running Level II BASIC or Disk BASIC. Level II BASIC has only one USR call available. USR (User SubRoutine) allows BASIC to jump to a Machine Language subroutine from a BASIC program. If the subroutine ends with a Machine Language RET (Return) instruction, control will return to the BASIC Program at the point where it left.

When the USR command is encountered in Level II BASIC, the computer looks at the contents of memory locations 16526 (408E hex) and 16527 (408F hex) to find the starting address of the Machine Language routine being called. If you want the computer to execute your routine, you have to POKE its starting address into those memory locations, with the low order byte in 16526 and the high order byte in 16527. PRINT PEEK(16526),PEEK (16527), before you've POKEd anything in there. . . the values 74 and 30 are returned. 74 equals 4A hex, the low order byte, and 30 equals 1E hex,

continued on next page

continued from previous page

the high order byte. At power-up, those locations contain the address 1E4A, to which the computer will jump if you make a USR call without POKEing your starting address into those locations first. What is 1E4A? It's the address of BASIC's ILLEGAL FUNCTION CALL error routine. Very clever.

Let's avoid making an ILLEGAL FUNCTION CALL, by having the starting address of our routine POKEd into all the right places before we use USR. If our routine is in A$, then we know that PEEK(VARPTR(A$) +2) gives us the high byte. How convenient. That's JUST what we need for POKEing:

```
20 I=VARPTR(A$):
POKE16526,PEEK(I+1):
POKE16527,PEEK(I+2)
```

USR calls can pass arguments to the subroutine, but if the routine doesn't need any arguments, then the number between the parentheses is a "dummy value" (we'll use "0", but it could be anything). We call our Machine Language routine by:

```
30 X=USR(0)
```

This will cause the screen to be "whited-out" with Machine Language speed. Be sure to put those POKEs just before the USR command is encountered by the program. You can never be sure just WHERE the computer has stashed your string, and you want to make sure that it jumps to the right address!

In Disk BASIC, we have up to ten USR calls, USRØ to USR9, so we can call up to ten Machine Language routines. Instead of POKEing the starting addresses of the routines into memory as we did in Level II, we define the start of each by using the ten DEFUSR commands, DEFUSRØ to DEFUSR9, followed by the starting address in decimal or hexadecimal. Since our routine is in a string, and VARPTR returns the starting address in decimal, it's convenient to stay with that:

```
20 I=VARPTR(A$): DEFUSRØ=
PEEK(I+1)+256*PEEK(I+2)
30 X=USRØ(0)
```

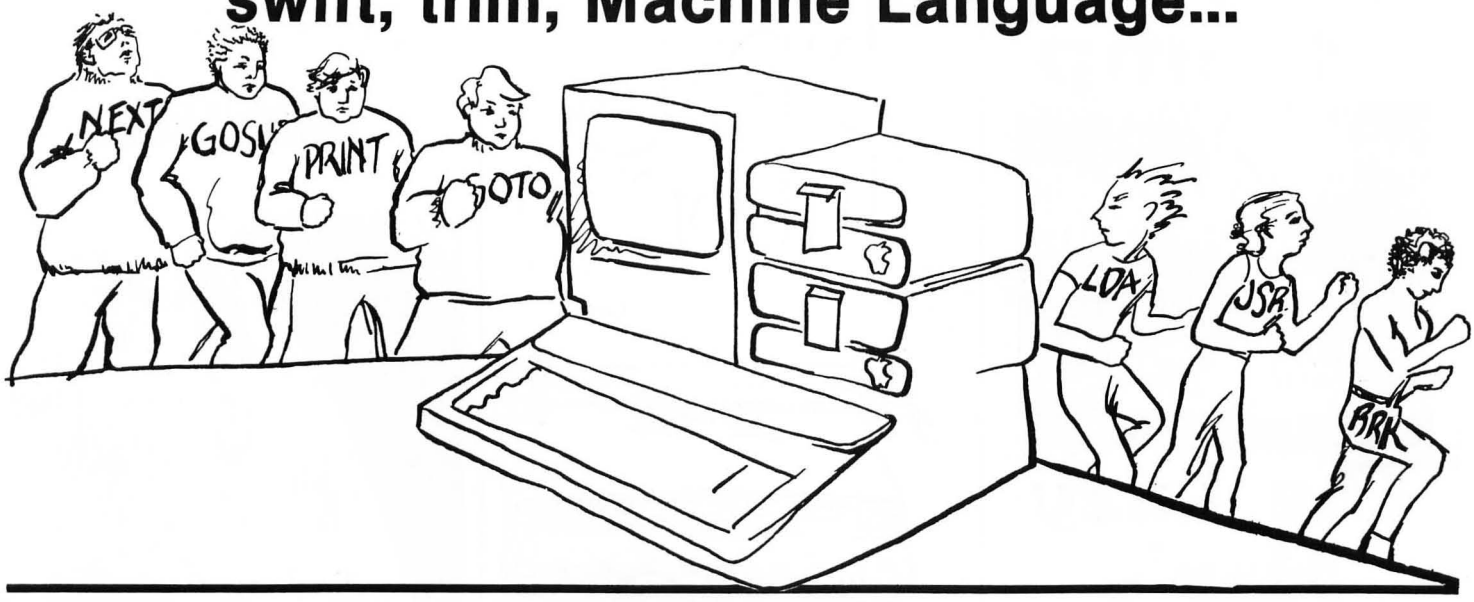
Don't forget. Just as we did in Level II, we have to make sure that we define the starting address of our routine using DEFUSR just before we call the routine, in case BASIC has moved the string.

If you want to pack a routine into a string, and then use it in a BASIC program that will be RUN under Level II and Disk BASIC, you must include both the POKEs and DEFUSR. Calling DEFUSR from Level II will give an L3 ERROR, and Disk BASIC doesn't look at addresses 16526 and 16527 when USR is called. Have your BASIC program check to see IF a disk is present, and THEN execute the correct statement. Memory location 16396 (400C hex) contains a Machine Language RET instruction (decimal 201) if no disk is connected, and a jump to 4BA2 (RST 28, DOS request processing) if disk BASIC is RUNning:

```
20 I=VARPTR(A$)
25 IFPEEK(16396)<>201THEN
CMD"T":DEFUSR=PEEK
(I+1)+256*PEEK(I+2)ELSE
POKE16526,PEEK(I+1):
POKE16527,PEEK(I+2)
30 X=USR(0)
```

Packing any Machine Language routines into strings makes a program much easier to use, since the MEMORY SIZE never needs to be set, and many routines can be used by the same program without worrying about them conflicting in memory.

Take the fat off BASIC and turn it into swift, trim, Machine Language...



...without any sweat from you.

INTRODUCING THE HAYDEN APPLESOFT COMPILER
For only $175.00

You can cut the fat from BASIC with the **HAYDEN APPLESOFT COMPILER**, the first true compiler available for the Apple II computer. This multi-pass compiler allows your BASIC programs to run up to ten times faster in Machine Language — faster than your Apple II can normally interpret Applesoft BASIC.

The **HAYDEN COMPILER FEATURES:**

* Runs standard Applesoft BASIC programs up to ten times faster.
* Seventeen-pass compiler produces true machine code.
* Compiles code at user specified address.
* Requires only Applesoft in **ROM** and one disk drive.
* Handles Hi-Res graphics and shape tables.
* Arithmetic completed for faster operation.
* Allows multiple programs to reside in memory at the same time.
* Works in a single or multiple disk environment or with hard disk.
* At the end of compilation, the compiler provides a detailed analysis of the structure.

System Requirements

* A 48K Apple II Plus or Apple II with Applesoft in ROM.
* The Autostart ROM chip and at least one disk drive.
* Systems equipped with language system or Microsoft RAMCard.

* Introductory price for our readers
Save $25 Limited time only*
$175.00

The HAYDEN APPLESOFT COMPILER is a product of the Hayden Book Company, Inc.

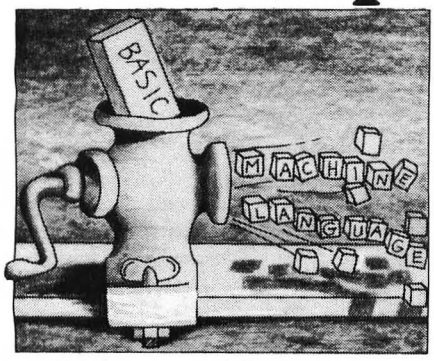
*offer expires October 15, 1981

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585

MasterCard VISA

Tiny Comp

RAMWARE



The Lazy Man's Shortcut to Machine Language!

A BASIC **Compiler** in BASIC! Run your source program in BASIC, compile it into FAST Z-80 Code and execute the compiled version — all without reloading. 26 integer variables, GOTO, GOSUB, END, REM, RND, LET, +, *, /, IF, THEN, =, INKEY$, CLS, PRINT@, CHR$, PEEK, POKE. Compiled programs may by saved via TAPEDISK.

Supplied with game program, "3-D TIC TAC TOE", which uses all of the TINY COMP statement set and is ready to compile.

Manual includes several sample programs as well as thorough documentation of the Compiler for those who like to know "how things work" and for those who might even wish to EXPAND on TINY COMP's capabilities.

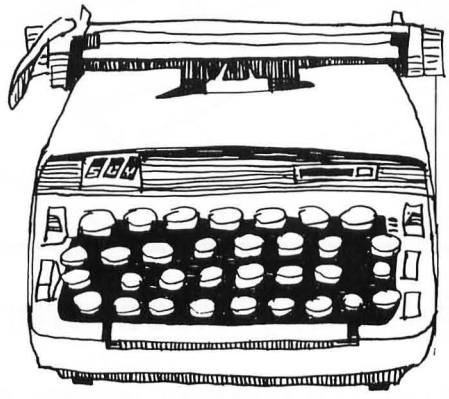
16K, S-80 Cassette $19.95
32K, S-80, Disk $24.95

VISA MasterCard

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585

Typing Tutor

by Roy Groth



Wish you were a better typist, but don't want to take (or pay for) a class? Teach yourself to type with the aid of your micro-computer. With TYPING TUTOR you will be quizzed and graded, but you set the pace at which you learn. TYPING TUTOR is a set of programs that lets you become as good a typist as you wish, allowing you to advance from one level to the next when you feel comfortable with your skills.

Let "hunt and peck" slip into the past, teach yourself speed and accuracy on the keyboard with TYPING TUTOR.

S-80/16K/Cassette ~~$19.95~~ $15.95

RAMWARE

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585

MasterCard VISA

RAMWARE

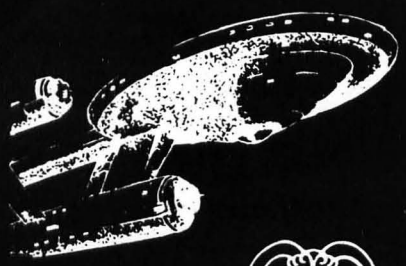
STATPAL

by Bruce Chalmers

Stumped by statistics? Here's the program for you!
Written by a statistician but designed for use in the real world. Helps you create files; examine and edit data; and understand descriptive statistics. Sophisticated enough for the working statistician. This powerful interactive statistical package features complete error diagnostic, missing value specification, and sophisticated graphics.

S-80 Disk 32K and 48K versions on one disk $29.95

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585



RAINYWARE

STAR TREK III.5

ALL NEW VERSION!

by Lance Micklus

Now with Sound Capability and Increased Speed of Execution.

You are in command of the starship Enterprise and her complement of 371 officers and crew. You must enter and explore the Omega VI region of the galaxy with its 192 quadrants containing star systems and planets (a few of which are habitable). Astronomical hazards such as pulsars, Class 0 stars, and black holes are known to be present in the region. Klingon battle cruisers are also present, so the utmost care is needed.

Star Trek III.5 includes: playboard 8 by 3 by 3 quadrants; weapons system of Phasers and Photon Torpedos; Warp and Impulse power systems; Science and Ship's computers; Long and Short Range sensors; Damage Control and Status reports; and 20 Klingon battle cruisers, and 100 stars, planets, black holes, and pulsars.

S-80 16K **Cassette** $14.95
S-80 32K **Disk** $14.95

MasterCard VISA

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585

RAINYWARE

Wargamer's delight

Three from Potkin

Warpath

The Indians are on the warpath! The Chief, along with 24 braves, is out to take the garrison at the fort, or at least to stop reinforcements from entering the stockade. The General, with his 14 troopers, is trying to relieve the garrison before the flag is captured. The player determines the scenario through placement of boulders that provide both shelter and obstacles. Favorite scenarios may be replayed.

S-80 Level II, 16K cassette $14.95

Up Periscope

The author of the popular Kriegspiel II has done it again. This time the action takes place at sea with one player controlling the submarines while the other attempts to sail around RADSHA Island, with at least three of his fleet surviving the attempt. This realistic wargame includes sonar, depth charges, and torpedos.

S-80 Level II, 16K cassette $14.95

Kriegspiel II

A much improved two-player version of the original. Kriegspiel II is a wargamer's delight. Choose the number of mountains (up to 200) and pick a scenario from the 9,999 possible, and then watch the computer set up the pieces, towns, mountains and a river. To win, you must enter the capital city of your opponent or reduce his fighting strength to below half of your own

S-80 Level II, 16K cassette $14.95

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585

Hardware Corner

by Edward E. Umlor

Once again the Granite block gets banged by the hammer and stammers out a few more lines of Widgets news. The mail that I am receiving (boy am I glad to get it) is aimed primarily at disk drives and the problems associated with them. Most of my direct experience is with S-80 systems and a very wide variety of compatible drives. I have had the fun of playing with disk systems for well over two years now, so I forget that there are those who have just received their first drive. The S-80 has several built-in problems (Tandy did not take the advice of the manufacturer of the 1771-B01 control chip) from the way they designed the Expansion Interface (EI). What this is all leading up to is answering some problems being faced by a novice disk user. I have also received a letter asking for an article or series (it will have to be the later) on disk drives: state of the art, 5 1/4", 8", and hard disk. This is an area of the microcomputing industry that is moving so fast right now, as to make it almost impossible to keep up. By the time an article hits the stands, the information contained is obsolete (as far as the manufacturers are concerned). I will try to have an article on just 5 1/4", on just 8", and probably more than one on hard disk drives.

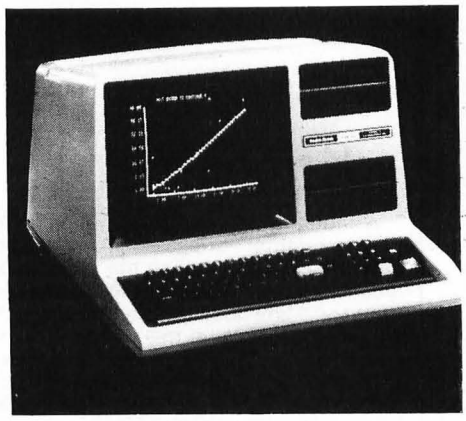
You have just turned on your system. You insert a DOS disk and press RESET. **PARITY ERROR DURING READ**. So you press RESET again and **GAT ERROR** (Granule Allocation Table). So you press RESET again and **HIT ERROR** (Hash Index Table). So you press RESET again and **CRC ERROR**. By this time you are ready to throw the whole system out the window, and as the last strands of your hair drift softly to the floor, an hysterical scream of "WHAT HAVE I GOTTEN INTO??" can be heard from several hundred kilometers away. The catacombs of the Tandy designer is what you find yourself buried in. BUT DON'T FRET!!!! ALL IS NOT LOST. HELP LOOMS ON THE HORIZON.

The S-80 has more non-manufacturer designed fixes and enhancements, than just about any other system in the world. The main reason for this, in my humble opinion, is the Tandy design group and their desire to cut cost at all costs. The 1771 disk controller chip was designed by Western Digital Corporation (WDC). At WDC the decision was made to design a limited data separator into the circuit of the chip itself. The following note appears as part of the 1771 specification sheet:

NOTE

Internal data separation may work for some applications.

However,for applications requiring high data recovery reliability, WDC recommends external data separation be used.



The internal data separator times out on the clock pulse supplied by the EI. If the data is there early (disk above 300 RPM), or late (disk below 300 RPM), the data gate is not open and booiiinnnnggggg. The circuit that WDC recommends for the external data separator times itself to the data and not to the EI clock. The installation of a data separator does wonders for the S-80 disk user. There are two readily available ways to obtain and install this type of data separator: 1) the PERCOM DATA SEPARATOR (can be ordered from TSE/HARDSIDE), which is only a data separator circuit and is very easily installed; 2) the PERCOM DOUBLER (also available from TSE/HARDSIDE), which allows you to run double/single density (1.8/1 data ratio) and includes a high performance data separator. For low volume of data storage, I would recommend the separator. If you are saving large quantities of data, then the Doubler is what you want. I have been using the Doubler for almost a year now and have NO, and I mean NO, complaints at all. This should solve most of your error and track lockouts.

The MEDIA is the next area that can give you grief. There are many brands of diskettes on the market and some are better than others. I have used BASF, VERBATIM, SCOTCH (3M), MEMOREX, DYSAN, and SYNCOM. The ones that have given me the most trouble are MEMOREX and SYNCOM. I was running single-density rated diskettes in double-density mode. This gives them a real workout and any flaws show up fast. In defense of all diskette manufacturers I will state that mishandling of the diskette is the biggest error-causing problem in the field. You should always protect the area of the diskette exposed by the long oval opening in the jacket (both sides). A slight rub or contact with an abrasive surface can make a diskette useless, and NEVER BUT NEVER touch that surface. The oils and acids from your hand are pure murder to a diskette. So remember to always treat your diskettes with respect, and always make sure you have a backup of anything important stashed away in a safe place. I know that a lot of software suppliers are making their disks uncopiable. Their paranoia is understandable to a certain extent, but they should supply a minimum of two copies with the original package and free (postage only) remake service. I cannot agree that a person who pays $100.00 to $200.00 for a program, should have to pay again because of the DREADED TANDY ZAP.

The DREADED TANDY ZAP can sneak up and GOTCHA without you knowing what happened. There are several causes and some cures that can be recommended.

1. Cause: Turning off your system with a disk in the drive and the door closed **GOTCHA**.

Solution: NEVER LEAVE A DISK IN THE DRIVE!!! Always remove all disks (or at least open the door which is interlocked) before shutting your system down. This is a **GOTCHA** that sneaks up on a lot of us old timmers too.

2. Cause: Power line spikes reset your system in the middle of a program **GOTCHA**. Your friendly neighbor

decides to weld his car back together, and up to a 1,000 volt spike zaps your system when rod touches metal.

Solution: Get a good line filter and spike suppressor with enough outlets on it for your system. The drives and lineprinters should be isolated from the system itself. The ISO-2 is the one I use and am very pleased with it.

3. Cause: You walk up to your system and **ZAP**. You just shot your system with up to 1,500 volts of static electricity. This is very prevalent north of the Mason-Dixon line in the winter (or any very dry climate). When there is a lot of humidity (50-55% and up to about 80%), the charge your body builds up is dissipated rapidly to the moisture in the air.

Solution: Under favorable conditions for static, don't shuffle across the rug, and do ground yourself by touching something metal (the wood stove or a lamp).

4. Cause: Lightening strikes nearby and **GOTCHA**.

Solution: Make it a practice to shut off and unplug the system and go pester your wife or nag your husband (whichever the case may be), or just sit by a window and enjoy the free fireworks.

5. Cause: The screen goes almost totally dark and **GOTCHA**. The power company decided you didn't need all that voltage, or someone shortened a power pole with their car, or junior just completed his science project and plugged it in.

Solution: PRAY A LOT!!!!! There are several uninterruptable (so-called) power systems on the market. TSE/HARDSIDE carries the MAYDAY line. It really depends on the power in your area as to your need for something of this expense.

In many cases of being zapped, you can recover your program. When going back into BASIC, type BASIC* and press ENTER. The * keeps from zeroing out the RAM as BASIC is loaded, and DO NOT PUT A SPACE BETWEEN BASIC AND THE *.

One last area before finishing up. Using both sides of a diskette. Some say NEVER DO THAT because you will erase side A while working on side B. They will not buy a flippy drive (a drive that allows you to flip the diskette over and use side B) for that very reason. HOGWASH!! BAHH!! HUMBUG!!! and all those other neat things. BOTH SIDES OF THE DISKETTE CAN BE USED WITH A HIGH DEGREE OF RELIABILITY, and that is the name of that tune. I consult for several companies that are using flippy drives or making the diskette itself flippy with excellent results. So go right ahead and buy that flippy drive or two-headed drive, and enjoy the luxury of buying half as many diskettes as your friends. You can laugh all the way to the movie (or bank) with the diskette money you did not have to spend.

For you people that have single-sided drives (like myself) who want to use both sides ot the disk, there is a way to do so. If you have crunched the center of a diskette, or scratched it, or it is just plain unuseable, remove the magnetic disk from the jacket by pulling it out through the middle hole. You have just manufactured an accurate template of the write protect notch and index hole locations. You didn't know you were a machinist, did you? You will need a marking pen with a fine point and a paper punch — you know one of those things conductors on trains use (whoops I just dated myself, didn't I?). Set the template over a diskette so that the write protect notch of the template is on the opposite side of the one on the diskette. Using the pen, mark the notch and the index hole. Place the template on the opposite side of the diskette and repeat. You now have the location for the new notch and the index hole on each side of the diskette. Use extreme care not to damage the magnetic surface as mentioned earlier. Using the paper punch, make the new write protect notch slightly deeper than the mark to compensate for the difference between the mark and the normal depth. Slide a piece of plastic (a baggie or something) between the magnetic disk and the cover. Center the hammer of the punch (the rod part that makes the hole) over the index hole mark and punch a hole in the cover only. DO NOT PUT ANY MORE HOLES IN THE MAGNETIC DISK!!!!!!. Repeat the procedure for the index hole on the other side. You now have a diskette that can be flipped over in your single-sided drive and you can use side B. Be sure your new index holes in the cover are as free of lint as you can get them. I have one company that is using this method with great success. They have had to buy about 100 diskettes so far, and at approximately $30.00 per box of 10 (average cost), you can see that they have saved enough money to almost pay for another drive in less than one year.

Well that's about all for this trip out of the mountain. I will try to gather material on disk drives present and future. I will enjoy to the fullest any free samples that I can obtain. This is old GRANITE KNOGGIN saying TA TA 'til next time.

ROBOTWAR

from Muse Software

"Robotwar" is for an Apple with Applesoft in ROM and one disk drive, DOS 3.2 or 3.3. $39.95 retail.

What do you call a program which is a combination of a strategy game, a high-resolution graphics shoot-em-up, and a learning tool which gets the players involved in programming the behavior of their very own robots? Muse Software calls it "Robotwar," and it makes for uniquely creative, challenging, and satisfying entertainment.

The fictional backdrop for "Robotwar" is a future world in which wars are not fought by humans but by robots. Your task is to create better robots than the other side, so that (naturally) your side can win the war. You do this by programming their semiconductor brains with various kinds of strategies, to make full use of their fighting abilities.

All robots have the same physical capabilities:

a. They can move in any direction and at variable speed.

b. They can keep track of their location on the battlefield using an x,y coordinate system.

c. They have radar devices which can give the range of an object in any direction. (The radar distinguishes between another robot and a battlefield boundary wall.)

d. They have the ability to aim a gun in any direction and fire a shell which will explode at a specified range.

e. They can assess the amount of damage they have suffered from the attacks and collisions.

Using these capabilities, your task is to program your robot with a winning battle strategy. This isn't something that you can do by sitting down for a minute or two and choosing various options from a menu. Rather, you actually write a program in a special programming language which will control your robot's decisons and actions in every circumstance of battle. This programming language is easy enough to learn in an hour or two, and versatile enough to develop rather elaborate battle strategies.

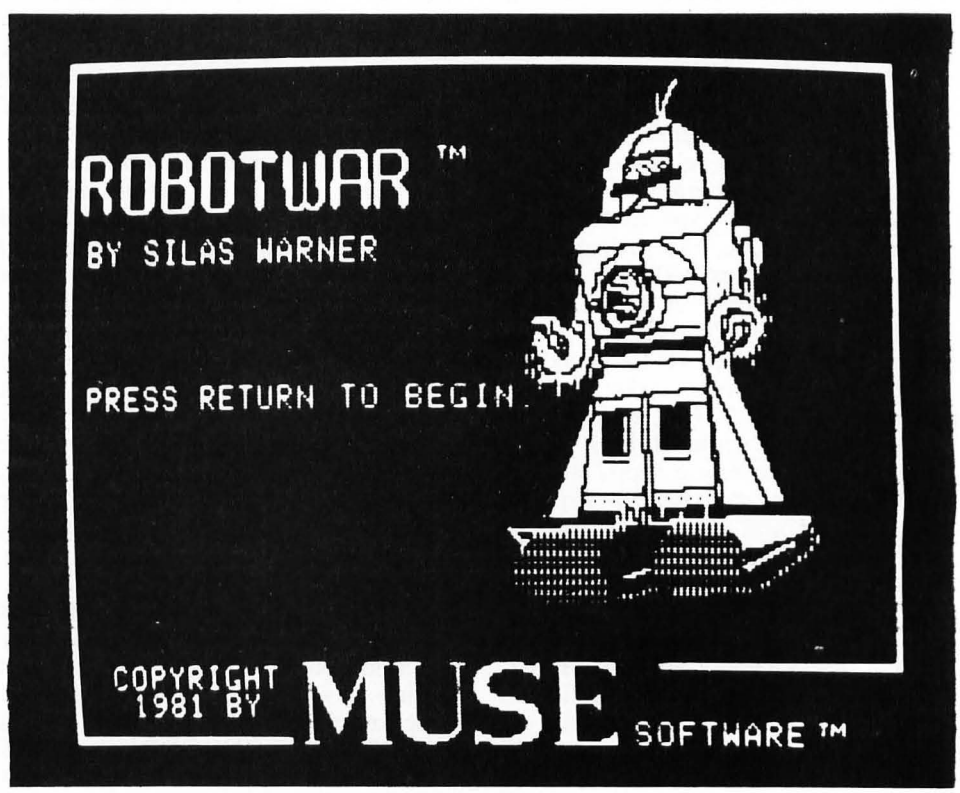
While the program for your robot can be developed on paper, it must eventually be typed into the computer. Muse includes an adaptation of their (excellent) "Super-Text" word processor as part of the total program package, which allows you to enter your program and edit it very easily. If you are familiar with "Super-Text," this part will come naturally to you; if not, it will take a little practice before you feel thoroughly familiar with the editing features.

As you enter and edit your program, you can save it to disk at any time, under your robot's name. And here's where the lasting challenge and entertainment value of "Robotwar" becomes evident. You can keep programming and reprogramming any number of robots, keeping them all saved on disk to be used in battle any time you choose. You're not limited by the available space on the program disk. By the way, you can initialize auxillary disks to store all that your programming skills can create. The obvious result is that competition will develop among "Robotwar" aficionados to create the most invincible robot. (It seems likely that some kind of mini-cult will develop, and a lot of robot data disks will be sent through the mail and exchanged at computer club meetings.)

Here is an example of a simple program to control a robot:

```
          SAMPLE ROBOT
SCAN
 AIM + 5 TO AIM; MOVE GUN
 AIM TO RADAR; SEND RADAR
PULSE
LOOP
 IF RADAR < 0 GOSUB FIRE;
TEST RADAR
 GOTO SCAN
FIRE
 0 - RADAR TO SHOT; FIRE THE
GUN
 ENDSUB
```

Each section of the program has a label: SCAN, LOOP, and FIRE, in this case. These labels can be almost anything you want, and allow you to control the flow of the program through GOTOs and GOSUBs. Anything following a semicolon (;) is simply a comment, and doesn't affect the program's function. The words AIM, RADAR, and SHOT are the names of the memory registers the robot uses in aiming its gun, positioning and reading its radar, and shooting and monitoring its gun. There are seven other special registers named X, Y, SPEEDX, SPEEDY, DAMAGE,



RANDOM, and INDEX; and 24 additional memory registers which your program can use for storing values.

A listing such as this one is called the "source code" for your robot's brain. Before your program can actually be used by the robot, it must be "assembled" into an "object code" form. The built-in assembler, which you can call in at any time to do its work, breaks down the source code into more elementary instructions which the robot can then execute. You can have this object code listed to a printer if you choose, and the assembler automatically saves it onto disk when assembly is finished. When battle time comes, it's this object code which is then loaded into the computer's memory for each robot that will be fighting. Here's what the assembled object code for the above sample program looks like:

```
SCAN
0       ,       AIM
1       +       5
2       TO      AIM
3       ,       AIM
4       TO      RADAR
LOOP
5       IF      RADAR
6       <       0
7       GOSUB   FIRE
8       GOTO    SCAN
FIRE
9       ,       0
10      -       RADAR
11      TO      SHOT
12      ENDSUB
```

You don't, however, have to plunge your robot straight from the programming stage into mortal combat. There's a clever "test bench" feature which allows you to do dry runs of your program without fatal consequences. The test bench will walk your robot through his paces, displaying each object-code instruction and the contents of various registers as the program is executed. You can specify continuous execution, at ten different speeds; or stepwise execution, where only one instruction at a time is performed. You can test your battle routines by simulating radar detection of another robot and by simulating damage to your own robot.

All these preparations finally culminate in a battle, or a match involving a series of battles, between two or more robots on the battlefield. This is a high-resolution display, graphically portraying the robots moving, sending radar pulses in various directions, and firing exploding shells at one another. The sounds of the battle (which are quite good) can be switched on or off at any time. The display shows each robot's current score and the extent of the damage it has sustained. When that figure reaches 100%, the robot in question is destroyed and removed from battle, adding one point to the scores of each of the remaining robots. The battle is over when only one robot remains, or when you (as the referee) call "time."

At that point, the scores of all the participating robots are automatically updated on the disk. All robots begin their existence with a score of zero, and accumulate points only as they survive longer in combat than others. If your robot's programming is good, it will obviously accumulate points faster than its competitors. Be aware, however, that even the slightest change in that program will reset the robot's score to zero.

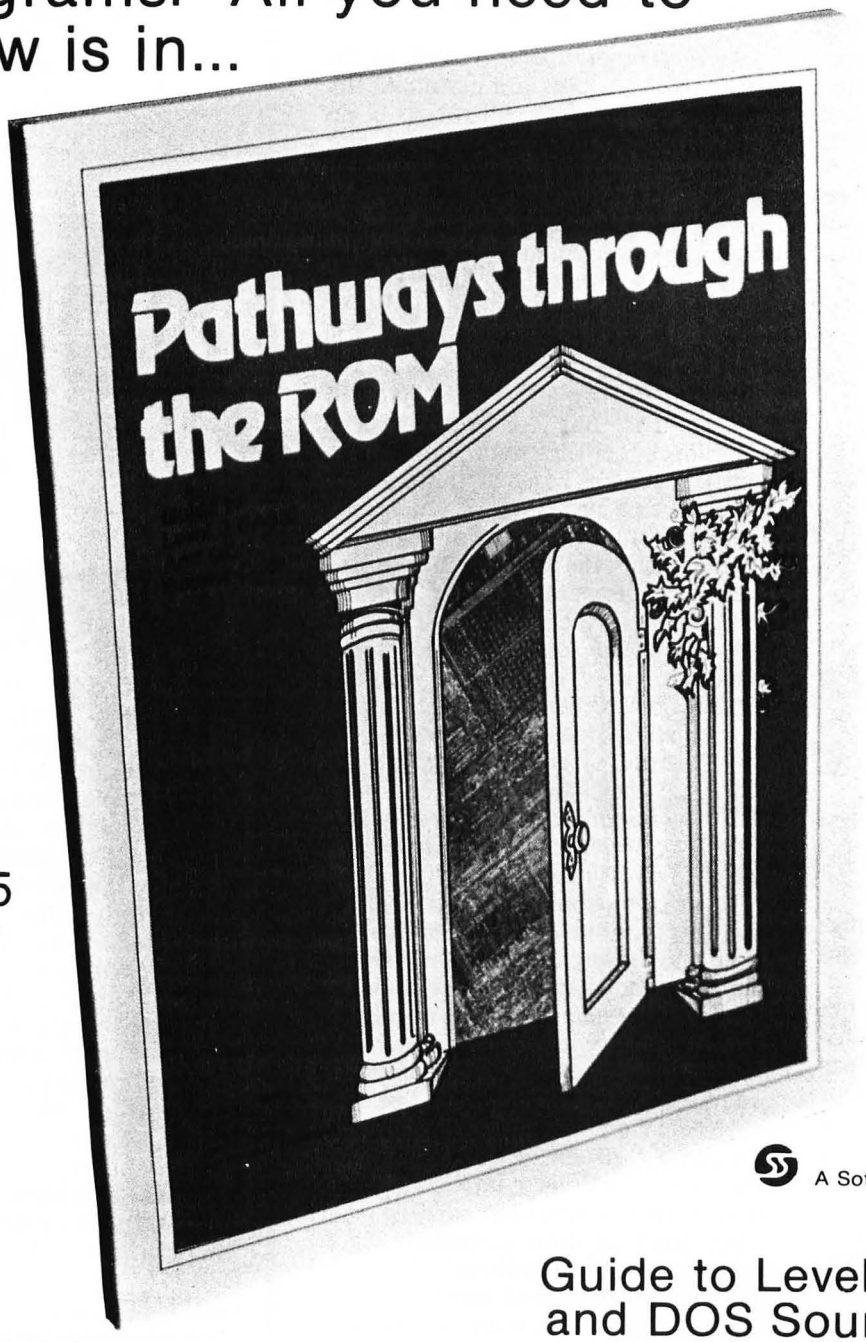
If all of this sounds rather involved — well, it is. But is it ever fun! The program is self-prompting once you have the general hang of things, with options being presented on the screen in menu form. If you do nothing when presented with the main menu, the program will automatically begin running battles among the five sample robots included on the disk. The documentation, in the form of a 76-page booklet, is excellent. (I noticed only two small typographical errors: A "+" instead of an "*" on page 61, and a "2" instead of a "TO" on page 62.) The instructions are clearly and logically presented, and tell you everything you need to know.

An obvious application of "Robotwar," in addition to entertainment at home and among computer hobbyists, is in the school classroom. For students in the upper elementary grades and above, this would be a painless and engrossing way to learn what computer programming is all about. If you are involved in computer literacy or programming instruction with an Apple, you should definitely introduce your students to this program.

But don't let those comments scare you away from "Robotwar" if you're just looking for a good game program for your Apple. This is an exceptionally fine program and deserves recognition for its innovation and overall entertainment value. I recommend it wholeheartedly.

Jon Voskuil

Unlock the hidden power of your computer for fast and easy programming! Use ROM routines in your BASIC and Assembly Language programs! All you need to know is in...



INCLUDES:

SUPERMAP
From Fuller Software ($18.95)

TRS-80 DISASSEMBLED HANDBOOK
by Robert Richardson ($10.00)

HEX MEM
by John Phillipp
Monitor written in BASIC

Z-80 DISASSEMBLER
by George Blank

ALL ONLY $19.95
plus $1 shipping

A SoftSide Publication

Guide to Level II BASIC and DOS Source Code

Description of the contents of the Level II BASIC ROM by memory locations, by function, and in lesson format. Includes several BASIC and Assembly Language programs in listing format to examine and use ROM routines.

MasterCard VISA

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585

Let our fingers do the typing!

With a **SoftSide** Disk or Cassette subscription you can get each month's TRS-80, Apple, or ATARI programs delivered on disk or cassette with every issue of your **SoftSide**. No more hours of typing. No hunting for typing mistakes. The programs for your computer are tested and ready to go the day you receive them.

If you already receive **SoftSide** magazine, you will receive credit for the remainder of your subscription toward your new cassette or disk subscription.

Magazine $24/year
Magazine and Cassette $75/year
Magazine and Disk $125/year

SoftSide™
cassette & diskette

Make your **SoftSide** library complete with back issues of **SoftSide Apple Edition, SoftSide: S-80 Edition,** and **PROG/80.** Programs, games, complete documentation and lots more! If you have missed out on any past issues, now is the time to order.

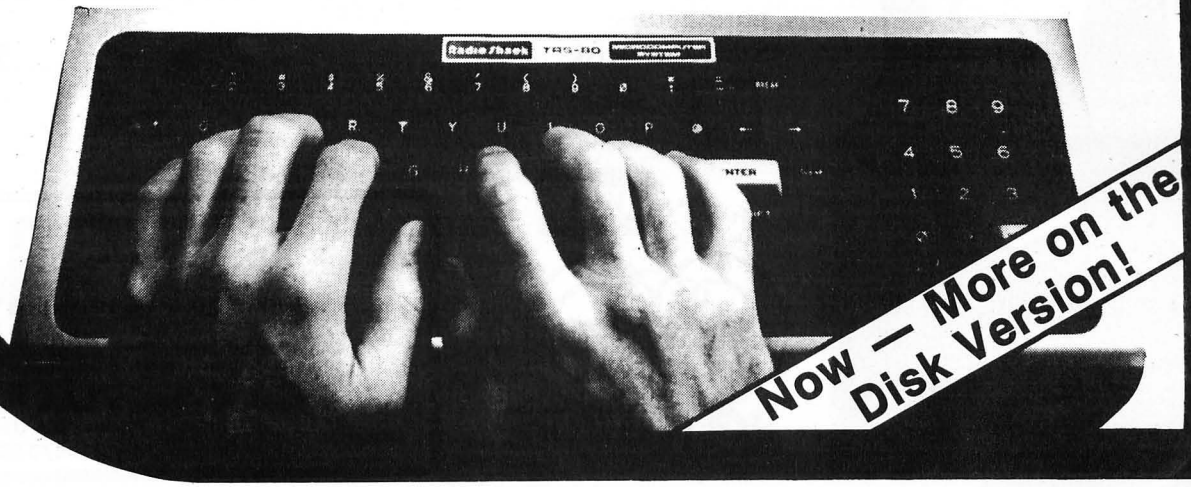
SoftSide: S-80, and **SoftSide: Apple** back issues (magazine only) $2.50
PROG/80 back issues $4.00
Magazine with Cassette programs $9.95
Magazine with Disk programs $14.95
New Super **SoftSide** back issues
(Magazine Only)
August or September, 1980 $3.00
October, 1980 to present $3.50

For descriptions of the contents of our back issues, please see the May, 1981, issue of **SoftSide.**

Collectors! Protect your **SoftSide** back issues, Volumes I and II, or any publication of your choice, with these durable wood grain vinyl binders with inside pocket and clear spine sleeve for easy identification. Holds and protects 12 back issues. A regular $4.95 value, SALE priced at $3.95. Free (while the supply lasts) with the purchase of volume I or II (12 issue collection of **SoftSide**).

Small	$3.95
8½ X 11	$7.95

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585



Let PASCAL-80 talk some sense into your computer

Phelps Gates, the author of "APL-80", brings you "Pascal-80" for your S-80. Now you can add another dimension to your programming skills by using this fast version of the compiled language Pascal.

"Pascal-80" is a powerful, structured and well-defined language for the S-80 microcomputer. This easy-to-use language makes writing well-structured, and therefore easily understandable programs simple. "Pascal-80" supports most of the features of UCSD Pascal, including RECORD, SET (to 256 members), FILE (text and record oriented), n-dimensional ARRAY (and ARRAY of ARRAY, etc.), global GOTO ELSE in CASE statements, and BCD arithmetic accurate to a full 14 places (including log and trig functions), 6-digit optional. "Pascal-80"features a 23600 byte workspace in 48K, a 1000 line per minute compiler, an easy-to-use text editor, and plain English error messages, all the features you would expect in a Pascal costing hundreds more.

Variable Types: . . . . Boolean, integer, char, real, real6, and text.
Constants: . . . . . Maxint, minint, true, false, and pi.
Files: . . . . . Input, output, and lp.
Procedures:. . . . . Read, readin, write, writein, reset, rewrite, close, seek, cls, and poke.
Functions: . . . . . Abs, arctan, call, chr, cos, eof, eoin, exp, inkey, in, mem, odd, ord, peek, pred, round, sin, signif, sqr, sqrt, succ, and trunc.

"Pascal-80" does not implement variant records, pointer and window variables, or functions and procedures used as parameters.

S-80 32K Disk.....$99.95

MasterCard

VISA

RAMWARE™

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585

TSE-HARDSIDE

1982 Micro-Computer Buyer's Guide

Are you weary of anxiously opening your latest Micro-Computer catalogs only to find that the product information is sadly lacking? Wouldn't it be nice if just once you could find all the comparative charts, graphs and background information on a product you're considering in a single book? You needn't look any further—The TSE-HARDSIDE 1982 MICRO-COMPUTER BUYER'S GUIDE is available to you now! We've included such valuable information as print samples from each of the printers we carry,feature-by-feature comparisions of Micro-Computer systems in an easy-to-read table format, an informative article on Micros to lessen the confusion for novices and pages and pages of complete product descriptions.

Such a guide should sell for $10 or more, but our experience has taught that the best customer is a well-informed consumer. So, we're making this valuable reference available to you for only $2.95. Of course, your $2.95 is refundable on your next purchase from TSE-HARDSIDE. Do yourself a favor and don't make another "guess purchase." Fill out the coupon below and drop it in the mail today. Charge customers are welcome to call our toll-free number:1-800-258-1790 (in NH 673-5144). THE TSE-HARDSIDE 1982 MICRO-COMPUTER BUYER'S GUIDE will soon arrive at your address via first class mail.

YES! Send me the **TSE-HARDSIDE Micro-Computer Buyers's Guide**

☐ I've enclosed $2.95
☐ Charge to my credit card
☐ MasterCard ☐ Visa
☐ Please send only your **FREE** Price List.

Card Number _____
Interbank Number _____ Exp. Date _____
Signature _____

Name _____
Address _____
City _____
State _____ Zip _____

Send to:

TSE-HARDSIDE
Dept. C
14 South Street
Milford, NH 03055

TOLL FREE OUT-OF-STATE
1-800-258-1790

Hot pursuit through space and the vortices of time!

RAMWARE PRESENTS. . .

Time Lord

The fallen Time Lord, who presumptuously calls himself The Master, is at large. The elders of Waldrom have supplied you with the hyperspace-worthy vessel TARDIS, and commissioned you to eliminate the evil "Master". Your resources include clones who will fight for you, the formidable CRASER weapons of the TARDIS, and magic weapons such as Fusion Grenades and Borelian Matrix Crystals.

Travelling through hyperspace in search of the evil one, you will encounter Time Eaters, Neutron Storms, and other alien creatures and phenomena. Entering real space to search planets, you will encounter still other dangers. You will enter native settlements to buy food and supplies — or to fight for survival.

And once you find The Master can you destroy him?

Based on Dr. Who of PBS fame.
Apple Integer Basic,
Disk, 48K . . . $29.95

SoftSide Selections
6 South Street Milford NH 03055
For Orders Only 603-673-0585