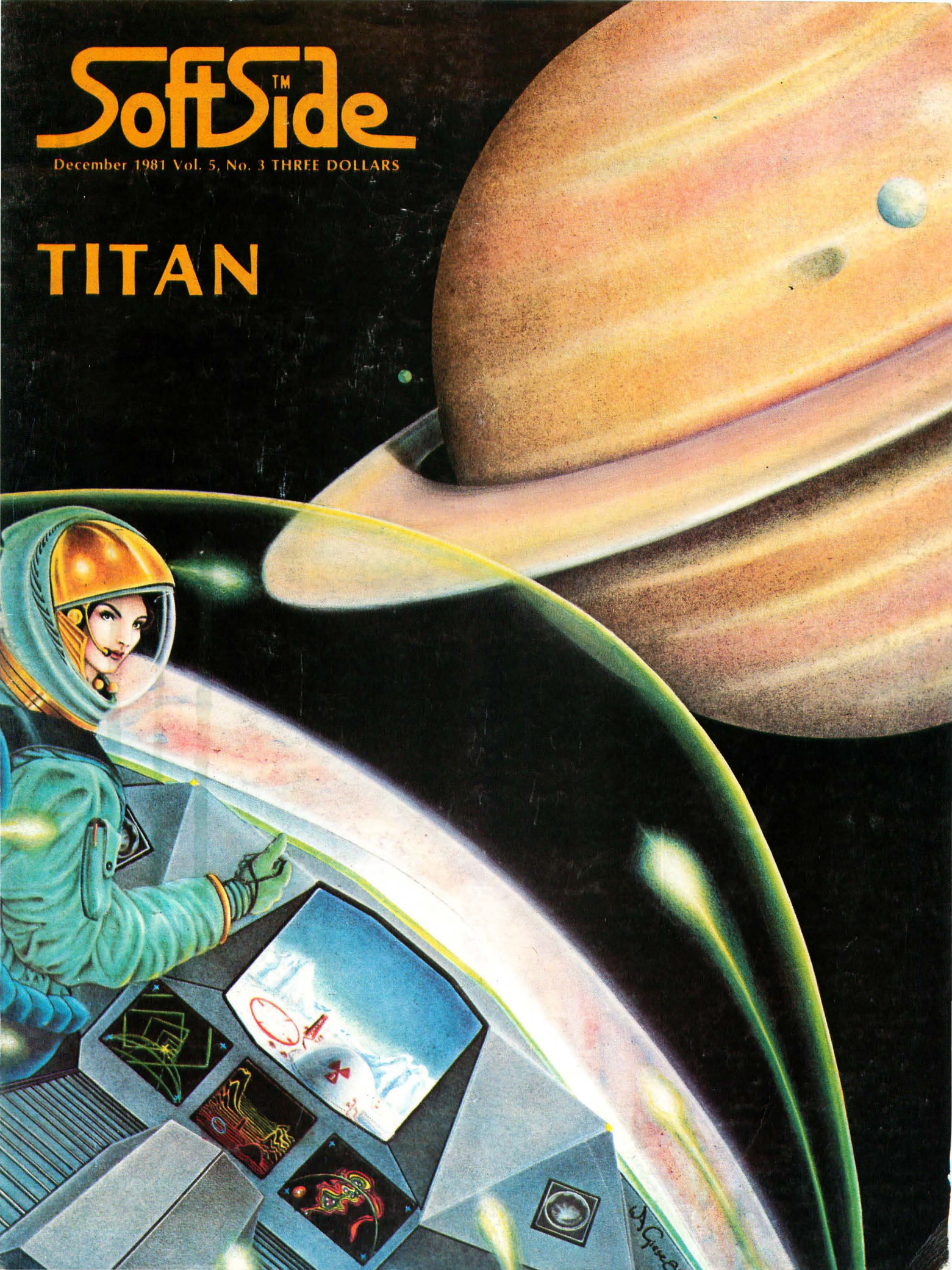


SoftSide™

December 1981 Vol. 5, No. 3 THREE DOLLARS

TITAN





diversionsTM

thru Envyrn

Module One
PARSECTM

You hover on the fringe, five parsecs from initialization of your mission. Your culture is tired and in need of new vitality. The Chamber can duplicate the essence of life from any species and you're about to start your "collection." What fate awaits you on this unique mission? Is it true that cultures opposed to your sampling have been preparing their defenses and stand ready to resist your intrusions?

The first module is prepared for take-off — **diversions thru EnvyrnTM** presents **ParsecTM**, a graphics space adventure unlike any you've ever played. **ParsecTM**, created with the **EnvyrnTM** graphics editor, simulates deep space in your TRS-80[®] computer, creating an action-filled field of play larger than any other microcomputer game. **ParsecTM** takes you beyond the screen and into worlds not yet seen by you or your computer. Join the adventure — subscribe to **diversions thru EnvyrnTM** now or send \$20 for **ParsecTM**, the first of six modules to be released in 1982.

diversionsTM

thru Envyrn

6 South Street Milford, NH 03055

- One-year subscription (6 modules) — \$60.00
 Sample **EnvyrnmentTM** ParsecTM only — \$20.00

Name _____

Address _____

City/State _____ Zip _____

Payment enclosed

MasterCard VISA Exp. Date _____

Name of Cardholder _____

MC# and Interbank#/VISA# _____

Signature _____

I own a 48K TRS-80[®] Model I Model III

TRS-80 is a registered trademark of Tandy Corporation

Envyrn™

Putting
micro-
graphics
into
action



Micrographics are now practical. With Envyrn™, maps, charts, graphs, floor plans, or any physical situation may be created or duplicated — in three levels of resolution. After you've created your graphics, save them, redesign them, create hard copy, and put them into action. With the artificial intelligence features of Envyrn™, you can make your graphics "smart," with prescribed reactions and information hidden under the display. You can truly create your own world.

In the Envyrn™ Participation Program, you will be a part of the documented experience in this new software innovation. During your full year of participation, you will receive the initial Envyrn™ editor (a much expanded version of the one published in SoftSide), extensive supporting documentation, program notes, suggested applications and at least one completely updated editor later in the year — all for only \$200. Envyrn™ is destined to become one of the standard utilities in your program library. Take part in this unique experience. The Envyrn™ Participation Program — Putting micrographics into action.

Envyrn™

Participation Program

6 South Street, Milford, NH 03055

One-year subscription to the Envyrn™ Participation Program — \$200.00

Name _____

Address _____

City/State _____ Zip _____

Payment enclosed

MasterCard VISA Exp. Date _____

Name of Cardholder _____

MC# and Interbank#/VISA# _____

Signature _____

I own a 48K TRS-80® Model I Model III

TRS-80 is a registered trademark of Tandy Corporation



Season's Greetings

from all of us
at

SoffSide™



SoftSide™

Volume V — Number Three

PUBLISHER
Roger W. Robitaille, Sr.

ASSOCIATE PUBLISHER
MANAGING EDITOR
Randal L. Kottwitz

PROGRAMMING EDITOR
Jon Voskuil

EDITORIAL DEPARTMENT
Scott Adams
Rich Bouchard
Dean F. H. Macy
Lance Micklus
Mark Pelczarski
Joan Truckenbrod
Ed Umlor
Alan J. Zett

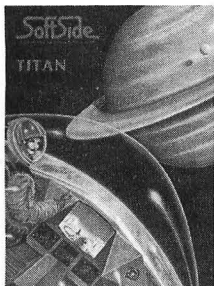
PRODUCTION MANAGER
ADVERTISING
Nancy Lapointe

PRODUCTION DEPARTMENT
Lynda Fedas
Tom Stanton
Lynn Wood

CUSTOMER SERVICE
Nancy Macy

DEALER SALES
Kathie Maloof

STAFF
Kathleen Boucher
Pam Demmons
Stephen Justus
Doris Miller
Cindy Schalk
Christine Spade
Anmar William



Titan is our featured program this month. On the cover an astronaut is shown flying to Titan, one of Saturn's moons. Will she succeed in her mining operation there? Illustration by Bill Giese.

Photographs done by Dean F. H. Macy

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. Entire contents copyright 1981. *SoftSide* Publications. All rights reserved.

POSTMASTER - send address changes to:

SoftSide Publications
515 Abbot Drive
Broomall, PA 19008

If you have not received your November issue of *SoftSide* by Dec. 6, contact *SoftSide* Publications, 515 Abbot Drive, Broomall, PA, 19008 or call 1-800-345-8112 (In PA call 1-800-662-2444).

ARTICLES

- 10 VARPTR Edited**
Editing packed strings John T. Phillip, M.D.
- 70 POKE Your Atari — Part 2**
CLOADing troubles solved Alan J. Zett
- 91 User-Defined Functions**
TRS-80® DEFFN feature explored David A. Gash

REVIEWS

- 84 Voyage of the Valkyrie** Alan J. Zett
- 85 Race for Midnight** Ulf Lindmark
- 86 Missile Command** Dean F. H. Macy

APPLE™, ATARI™, TRS-80® PROGRAMS

- 21 Microtext 1.0**
A simple word processor Jon Voskuil
- 29 Titan**
Mining on a Saturn moon William Morris & John Cope

APPLE™ PROGRAM

- 65 Aircraft Commander**
Instrument flight simulator William J. Edmunds

ATARI™ PROGRAMS

- 71 Developing Database**
The new, improved version Mark Pelczarski & Paul Marentette
- 77 Design Master**
Experiment with geometric patterns Richard Lamb

TRS-80® PROGRAM

- 59 Electronics Assistant**
A circuit-designing utility John D. Adamson

SOFTSIDE DV

- 50 ATARI™ — Survive** Randy Massey
- 50 APPLE™ — Bobsledding** Maxwell Su
- 51 TRS-80® — Konane** Norman Whaland

DEPARTMENTS

- 5 Editorial** Jon Voskuil
- 6 Input** From our readers
- 7 Hints and Enhancements** From our readers
- 8 About This Issue** Randal L. Kottwitz
- 11 Outgoing Mail** Randal L. Kottwitz
- 14 The Sensuous Programmer** "J"
- 18 Entertainment Tomorrow** Allen L. Wold & Fred D'Ignazio
- 28 Calendar** Kathleen Boucher
- 52 K-Byters** From our readers
- 58 Envyrnese™** Roger W. Robitaille, Sr.
- 83 Bugs, Worms, and Other Undesirables** Editors
- 87 My Side of the Page** Lance Micklus
- 94 Hardware Corner** Edward E. Umlor
- 96 Machine Head** Spyder Webb

TRS-80®, Apple, and Atari are registered trademarks of The Tandy Corporation, The Apple Computer Company, and Warner Communications, respectively. Envyrn, Envyrnment, Envyrnese, and diversions thru Envyrn are registered trademarks of Roger W. Robitaille, Sr.

SoftSide DV, the magazine of the future, is here!

If your computer could pick a magazine, wouldn't it prefer one in its own language? Now there's one available.

SoftSide DV is an enhancement of the **SoftSide** you have in your hands.

SoftSide DV contains not only the complete programs listed in every month's issue of **SoftSide**, but additional programs of every conceivable type, as well — multiple and Machine Language programs, modified languages, ongoing modular programs and software so extensive, it would take an entire issue of **SoftSide** just to print the code. Only the documentation for these programs will appear in **SoftSide Magazine**, **NOT** the code.

Feel as though you're missing something? You are! But, you needn't miss out on another issue. **SoftSide DV** is now available for Apple™, Atari™ and the TRS-80™. The cost to you — \$125 for 12 magazines and 12 disks, packed with some of the best software available, all delivered to your home in the next year. For orders outside the USA, please add \$36. For your convenience, we offer an installment payment plan for VISA and MasterCard holders: You pay only \$32.50 per quarter (a total of \$130, which includes a \$5 billing charge). Please use the bind-in card in this issue to order.

Computerists are offered the rare opportunity of marching into a new frontier. Advance to the front of the parade by subscribing to **SoftSide DV**, the magazine of the future, available today!





by Jon Voskuil

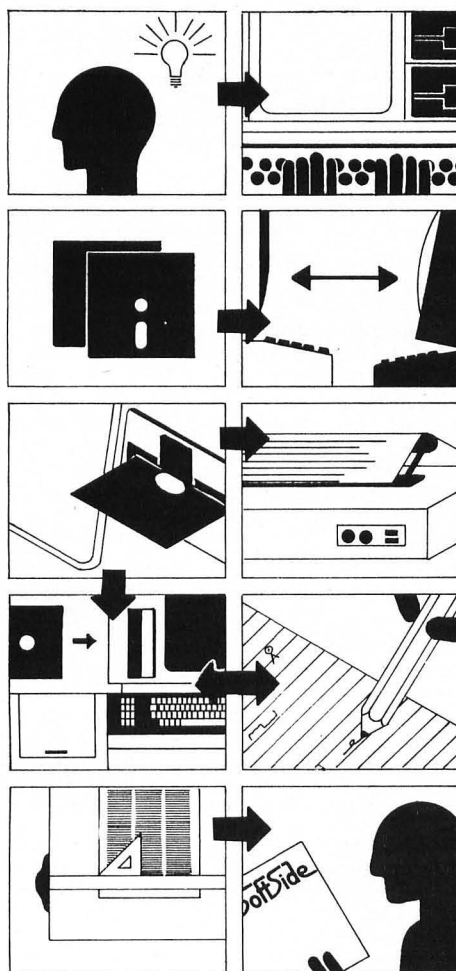
I am sitting at my desk, typing on my Apple's keyboard. The letters I type are being displayed, by the text processor program I'm using, on the video monitor in front of me. Since I'm writing in my head at the same time that I'm typing, I'm doing a lot of backspacing over poorly worded phrases, going back and deleting, inserting, and changing assorted words and sentences. Every few minutes I type a few control characters and the text I've entered is saved onto a magnetic disk, as a defense against the gremlins that lurk in the shadows waiting to cause a system crash. (I am typing this first paragraph over a second time because of just such a gremlin attack in the middle of the previous sentence.)

After completing the first draft, I'll go back over what I've written and do a little editing on it, smoothing rough edges and correcting spelling and punctuation. That will all be done from the Apple keyboard, of course, as I scroll through the text. At the same time, I'll make sure that the various special symbols which the typesetter will need are all in the right places, and that the overall format conforms to *SoftSide's* standard style.

With final draft on disk, and backup disk safely tucked away, I'll then connect a long cable between my Apple and the Model I TRS-80® sitting on Alan Zett's desk nearby. His computer and mine have been on friendly terms lately, and have done a lot of talking to each other. We like to encourage that, so we give them things to talk about. Like editorials. His computer will open its buffer, mine will open its communications port, and the text will go from Apple disk to Apple memory to TRS-80® memory to TRS-80® disk.

Then I'll take that Model I disk (after immediately making a backup) over to the Model III, where I'll convert it to yet another disk storage format (and make yet another backup). After a final reading to make sure that everything came out in the wash (and that I really did say what I thought I said), I'll dump the file to a printer. That's just so another editor can quickly read what I wrote, to see if it will pass standards of common decency and reasonable good sense.

If it does, then one of the backup



copies of the finished disk gets walked into the next room and inserted in another Model III. With appropriate magic movements and incantations, the contents of the disk will be converted into yet another file form, and then transmitted through yet another cable to yet another machine: the typesetter. And after the talented people in yonder room get through punching keys on the typesetter, and developing the film that said wondrous machine spews forth, and pasting such developed film onto pieces of cardboard — voila! a page of the December issue of *SoftSide*, ready to be sent to the printer.

So what's the point?

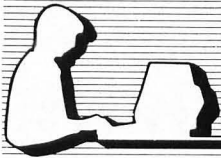
Well, part of the point is that ten (five?) years ago very few of us would have guessed that we'd have technology like this so readily available. Sorry, but I just can't get over those occasional flashes of "gee

whiz!" feelings about all this. We're living in a kind of magical time when computer technology is coming into its own, visibly developing and becoming more sophisticated month by month. Even those of us who weren't involved in the REALLY formative stages of the microcomputer revolution (let's see, how many months ago was that?) have a feeling that we're in on the ground level of something momentous.

Another part of the point is the frustration involved in the incompatibility of all these dumb machines. There are means available to make them talk to one another, but one has the feeling that the necessary patches, fixes, and jerry-rigged connections almost cancel out the potential advantages of machine interfacing. Not quite, but almost. Will there ever be some kind of magical device that will intelligently and automatically interface anything to anything? Fat chance. Then again, part of the problem is me, since I insist on using the Apple instead of walking eight feet over to the Model III, to do the typing there in the first place. (Often I do so, but it's in use right now. Besides, my desk chair is more comfortable, and that helps me meditate better — I need all the help I can get.)

Finally, a third point, which will directly affect those of you who submit articles and programs to us to be considered for publication. We've decided to computerize more of what we do (a radical idea, we freely admit), and will henceforth be storing all program documentation on disk rather than in file folders. We now require that all program documentation be submitted in machine-readable form, and will be increasing our payments for programs based on the quality of that documentation. In this issue you will find a simple tape- and disk-compatible text editor for all three of our computer systems, for your use in submitting text to us. This will help us with the compatibility problem as well as the paper overflow problem, so that anyone can both write and read media-based documentation with an absolute minimum of hardware and software.

Computers will never take over the world, but I sure won't mind having them take over all that repetitive retyping. ☺



MICROPHILIA

Dear *SoftSide*,

I have just finished reading Jon Voskuil's editorial on "Microphilia." I believe it was very well-written and right on the mark. I was a victim of this very serious problem and it was nobody's fault but my own. I think that all too many people abuse their computers by handing over all their free time and attention at the expense of their personal and family lives.

Computers are wonderful and amazing and all kinds of wonderful things, but we, as people, could be even better if we could teach ourselves to handle both talking to computers and talking to people. Communication skills are low enough without finding ways to completely eliminate all personal contact with each other.

We are the master and the computer is our tool. We still live in a world of people, not machines, and as such we should keep in touch with our human friends as well as our electronic ones.

Thank you for your observations. I hope they reach the people who need it the most.

Paul J. Cohen
Beverly, MA

HANDLE WITH CARE, THE SOFTWARE

Dear *SoftSide*,

Having visited several friends who also own computers, I realize that very few people really know how to take care of their investments. Having worked as a professional tape librarian for almost two years, I picked up quite a few "tricks of the trade" which I thought your readers might appreciate knowing.

1. Keep your disk drive doors and cassette lids closed when not in use. The magnetic heads attract dust and lead to faulty program loads and ruined data files.

2. Keep cassettes in their boxes and diskettes in their sleeves when not in use. This cannot be stressed enough. Unless your area is hermetically sealed, there is dust everywhere. It usually (according to Murphy's Law) ends up on the 48K program of which you have no backup copy or that irrecoverable data file.

3. Clean your disk drives and your cassette player regularly, with a commercially-prepared solution. This solution or cleaning disk or tape is relatively cheap compared to the cost of replacing the drive or the read/write heads.

3a. When cleaning, use only good quality swabs or lint-free cloth (not paper towels). Keep them in a zip-lock bag when not in use to keep dust away and never reuse them! Don't touch the cloth to the cleaning solution's bottle or dip the swabs into it. This contaminates it and further compounds your problems.

3b. When using a cleaning cassette or disk, replace it according to manufacturer's directions. They become imbedded with dirt, and could make your drive worse instead of better.

4. Keep your cassettes and diskettes in boxes designed for that use; not lying loose in a drawer or lying exposed on a table.

5. Dust your area regularly *after* you have finished for the night and *after* you have put everything away. Dusting doesn't pick up 100% of the dust around. It actually kicks some into the air which will later fall. This is the reason you only do it when done for the night.

6. Don't use your computer or store your tapes

or diskettes in very hot or very cold places, such as an attic or a basement. Optimum temperature is 70 degrees.

7. Use only good quality tapes, not cheapos. These will destroy your read/write heads faster than normal and your recording quality will diminish in the process. The short-term savings will not be realized when you have to buy a new recorder every six months.

Craig Tiano
LaSalle College

ROSES AND THORNS

Dear *SoftSide*,

I'd like to say that I think your magazine is one of the best of the computer magazines I've seen (if not THE BEST). Most magazines I've seen have pages and pages (about 300), which from a distance makes the \$3 cover price look small compared to *SoftSide's* \$3, but if examined, have about 9/10 of the magazine filled with advertisements. The programs and articles published are not worth but about five of the three hundred cents. Again, I'd like to say THANK YOU and keep up the very good work.

Kevin C. McHugh
Hinesville, GA

Editor's Reply: Many thanks for your compliments. You've hit on an important point — we consider *SoftSide* to be a product in and of itself rather than an advertising medium justified by editorial copy. The trade-off we make to maintain that attitude is asking our readers to pay a greater percentage of our production costs — thus our higher cover price.

Dear *SoftSide*,

Well, I guess that all good things must come to an end sometime! The notice has arrived that tells us our gift subscription to *SoftSide* is just about to come to its completion. All of us here at Folsom want to thank you for the year's worth of enjoyment and knowledge gained from reading and working with *SoftSide* magazine and its accompanying tape. It is a great combination and we will surely miss it. If we had the funds in our computer group, it would be the first publication that we would renew, but alas, our cupboard is bare! Anyway, we all wanted to thank you at *SoftSide* for your consideration and generosity.

During the current school semester, our Education Department is conducting three classes in programming theory and design, and has a capacity enrollment of 30 students. Each of the three classes has a one-hour lab period each day, so you can see that our limited selection of computing units is kept very busy. We are hoping to increase the 30 students to 60 by the beginning of the next school semester as there are presently over 100 men on the waiting list to take the programming courses. We recently had a disk drive (5 1/4" floppy) donated to us for use with our TRS-80® Model I so now we have a 48K disk system to work with and are enjoying working with its new capabilities. With the tape and disk system and the 48K of RAM, many programming possibilities are open to us, and we are working hard to absorb and utilize our new-found computing power.

Gottfried R. von Kronenberger
Folsom State Prison
Repressa, CA

Editor's Reply: It may be true that most good things come to an end, but not your subscription to *SoftSide*! We have extended your cassette subscription for another year to encourage your superb program. Happy computing!

Dear *SoftSide*,

Congratulations on your consistently enjoyable and informative magazine. My Atari and I look forward to each issue. Unfortunately, my checkbook rises in revolt at the mention of buying your cassette edition, so I do all the typing myself. Problems have appeared in the August issue for the games *Battlefield* and *Quest 1*. *Battlefield* was corrected in your October issue, but *Quest 1* was not. I have checked my typing and corrected those errors, but my debugging skill is not good enough to check the program itself except to correct Line 1031 to read MS=M1(RM)/10 so that the monsters attack the player. (The tipoff was that the original line was not accepted.) Other problems with *Quest 1* are:

1. The variables are not well-defined; page 26 does not begin to list all the contortions the data undergoes, and makes debugging impossible for amateurs.

2. In the wound routine, I allowed the character to be wounded. The wound value went by fives from 100 to minus 40 until I pressed a movement key, which stopped the subtraction.

3. In the room routine, the room progression has not gone past two, after which it goes to zero and stays there. (There is no exit from room zero.)

4. Passages (in the passage/intersection routine) are occasionally drawn *away* from the character, resulting in a monster and/or treasure being separated from a character by a wall. I have found that magic arrows can breach a wall, but I don't think it was supposed to work that way.

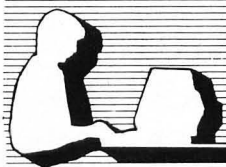
5. There is no room memory. On exiting from a room, you cannot go back and expect the room you just left to be the same.

6. Defining "&" — the red block alluded to in the explanation never occurred. On my machine, the Quest is in blue and white. I did not type in lines 812-884 and 20018-20060 to save time and because these lines only had to do with "saved" programs. Did I chop off something essential?

All in all, I think *Quest 1* has the potential to be a great program. Alan J. Zett, normally one of your best people, must have had an off day. Again, thanks for putting out a magazine that saves us hundreds of dollars over buying software in retail outlets and giving us information on how to make better use of our machines.

Mac Salfen
Arlington, TX

Author's Reply: In most cases, the problem is in a DATA line — in particular, line 112. Many people have mistaken the 9.9 (nine-point-nine) at the end of the line for a 9,9 (nine-comma-nine). To fix the program, retype the lines that you have changed and if you make sure there is a 9.9 at the end of line 112, *Quest 1* should run properly. (The lines that you omitted have nothing to do with the problems you've been having.) If you still have problems after making the necessary corrections, I would appreciate it if you would let me know. See also "Bugs, Worms..." in this issue. (A.J.Z.)



Hints & Enhancements

TRS-80® PROGRAMMING HINTS

To print a flashing "Press ENTER" centered horizontally on the current line, use the following (spaces inserted for legibility):

```
2000 CU = PEEK(16416 + PEEK(16417) * 256 + 32:
PRINT TAB(25) "<PRESS ENTER>";
2001 IF INKEY$ = CHR$(13) RETURN ELSE FOR X
= 1 TO 70: NEXT: IF PEEK(CU) = 32 PRINT CU
AND 1023, "ENTER";: GOTO 2001 ELSE PRINT CU
AND 1023, " ";: GOTO 2001
```

Locations 16416 and 16417 contain the current cursor address.

The expression CU AND 1023 in the above example is used to convert from video memory addresses to PRINT @ addresses. (You can also go in the opposite direction: to convert a PRINT @ address to a video memory address, use CU = PA + 15360, where PA is the PRINT @ location, and CU is the video memory address.)

(Editor's Comment: These seem technically correct, but advocate coding practices which make programs hard to understand. However they do save bytes when memory is crucial.)

You don't need a semicolon after the TAB(25) in line 2000, and you don't need any THENs in the IF statements. Which leads me up to the next programming hint: When do you need a semicolon in a PRINT, or a THEN in an IF? You only need THENs when there would be ambiguity without them; and you only need semicolons when necessary to eliminate ambiguity, in a PRINT USING, and at the end of a PRINT list to inhibit a carriage return. For example:

```
PRINT "Some text"; A$; B; "Some more text"
```

should be written as:

```
PRINT "Some text" A$ B "Some more text"
(Note: Spaces added for readability.)
```

Similarly, you *never* need a THEN after a right parenthesis or a type-delimiter (\$, #, !, or %). A THEN is only required if the controlling expression in an IF ends in a quote, or if there would be ambiguity as to where the conditional expression ended and the first controlled statement began; however, it is almost always possible to rearrange the expression to make it end in a character which never needs a THEN. For example:

```
IF ABS (A) = B THEN C = D + 1
INPUT "Play again (Y/N)"; A$ = "Y" THEN
RUN ELSE END
```

can be rewritten as:

```
IF B = ABS (A) C = D + 1
INPUT ... : IF "Y" = A$ RUN ELSE END
```

By reversing the order of the terms in the Boolean expression, we eliminated the trailing quote, thus eliminating the need for the THEN — thereby saving one byte of memory, not four. It is stored internally as CHR\$(202). In fact, you don't really need the THEN in an implied GOTO (i.e. IF A\$ = B\$ 1000 is just as valid as IF A\$ = B\$ THEN 1000).

If the controlling expression ends in a number, a THEN is needed *only* if the first controlled statement is an implied GOTO, or if it is an assignment statement beginning with a D or an E. For instance:

```
IF X > 0 THEN A = B ELSE IF SIN(X) < 0
THEN 1000
```

would work just as well as:

```
IF X > 0 A = B ELSE IF 0 > SIN(X) 1000
```

There are actually many other cases where a THEN can be eliminated. If you aren't sure, try leaving it out, and see if the result is two variable names being run together. If not, you can probably do without the THEN!

Sam Hills
New Orleans, LA

APPLE ENHANCEMENT

As a subscriber to *SoftSide* magazine, I think it is tops. In reference to the *Collision* game by Mark Pelczarski (November, 1980), I think the game is really enjoyable. The addition which Lynn Leopard of Chillicothe, MO added to the game (August, 1981) to keep a short sequential text file called "high" that holds the scores and names of the top ten scorers makes this game even more interesting. Lines 200 and 205 which were added are correct but after playing the game several times, I found it did not record the scores every time. I found it needed three more lines changed. The next three lines should be changed to read as follows:

```
450 IF XC=XY AND YC=YY THEN 900
650 IF XD=XY AND YD=YY THEN 900
652 IF XD=XY AND YD=YY THEN 900
```

After you make these changes, I find it records the top ten scores every time. I would appreciate it if you would pass this information along. Thank you.

Nathan Calloway
Kingsport, TN

TRS-80® TAPE PROBLEMS

The TRS-80® tape loading problems mentioned in R. E. Noble's letter in the October issue of *SoftSide* seem to be operator error. Level II BASIC will search the tape for a one-letter file or program. Thus a search for "program 1" is not valid. The computer will very likely try to find program "p". If there is no such file or program, nothing will load. Disk BASIC does not have search and will only CLOAD the next program.

TRS-80® does have a loading problem which was very hard to find. It is casually mentioned in "Special Notes" on page iii of my Level II manual. To load a program, the program *must* be positioned on a *blank* spot ahead of the program leader. Any blip, click or noise ahead of the program leader will cause the TRS-80® to "lock

up" without indicating when or why. The volume "may" be adjusted low enough that the computer will not see the blip and still read data. This is not easy.

I found the "blips" by monitoring the cassette output with a scope while I was trying to find a "magic" volume setting. The computer does not lock up when it sees the blip but it will lock up when it gets to the leader. The volume setting is not critical if there is no blip.

The tape *must* be positioned at a *blank* spot ahead of the program leader. If the computer sees any blip ahead of the leader, it *will* lock up. The cassette keeps running so it is hard to know when (or if) the TRS-80® has locked up.

To load problem tapes:

1. Listen for leader. (I use Radio Shack Mini Amp plugged into the phone jack with a Y-plug or unplug the black plug from the phone jack.) The leader is the "pure" tone before the hash of data.

2. Stop tape and note index # (in case you miss in step 3).

3. Rewind slightly — until just ahead of program leader.

4. Reset computer, replace black plug if pulled.

5. CLOAD

I can hear the cassette output with my setup so I know if there was a blip ahead of the leader when I attempted to load the program.

I plug the scope into the Mini Amp when I want to compare outputs. This way I can either hear or see the signal.

I hope that this will help solve some of the TRS-80® tape problems. I seldom have to touch the volume now that I know the problem.

Walter H. Peaslee
Elizabeth, IL

HI-RES GRAPHICS

I read with great interest the article "Mixed Text and Graphics in HGR2" by Jon Voskuil (September, 1981). I ran into the same sort of problem early on in my use of the Hi-Res graphics on my Apple, namely a War (card game) program that I wrote. Jon obviously put a lot of work into his solution and is to be commended.

My problem with the War program, was the same as his with *Hangman*. There are, however, two better ways to solve this problem. In both our efforts, we blew our program space with all those "scores of HPLOTS".

The best way to solve the problem would have been to eliminate all the HPLLOT statements and create a series of SHAPE tables for the various parts of the graphics display. These could have been placed at the top of his 32K and the program would have functioned on page 1, without the need for any special routines.

Shape tables look scary at first, but after you've done a few, they're a walk in the park.

There is, however, a second way to solve the "Mixed Text and Graphics (problem) in HGR2", in the event that a program is really too large. It's simple, and once done can be forgotten, as it will always be there, ready to use.

I don't claim credit for the idea, as it came directly from Apple, and was published by their engineering department on May 15, 1979. What

continued on page 9

About This Issue

by Randal L. Kottwitz

'Tis the season to be jolly! Sit down at your computer and take a break from the hectic holidays with the unique programs in this issue of *SoftSide*. We've combined the serious and the whimsical for your edification and enjoyment.

□ William Morris and John Cope, authors of many past *SoftSide* features, have done it again and the results will wow you! *Titan* is our cover feature this month and the illustration only hints at the excitement awaiting you on the surface of Titan, the frozen moon of Saturn. At least you'll be guaranteed a white Christmas. Just hope you can avoid freezing!

□ The typewriter is becoming an obsolete piece of equipment as word processors infiltrate the home and office environment. Jon Voskuil initiates a series on word processor development with *MicroText 1.0*. This "developing word processor" will not only help you understand how word processors are constructed but aid all of you prospective authors in the preparation of documentation and articles as well. Jon promises printer routines in next month's installment.

□ At long last, we are proud to present the complete *Developing Database* for the Atari. Few programs have been as popular as this series. New developments have taken place in the use of the Atari since the series was

first published and this version has been updated (and debugged) to fully utilize the computer's potential.

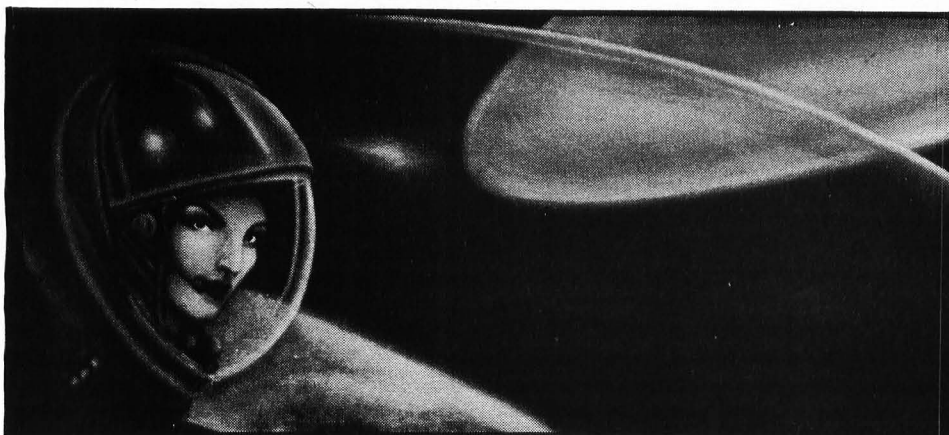
□ Have you ever wondered what it would be like to fly a jet in the dark, depending only on your instrumentation? Wonder no longer as you may now experience flight (or the next best thing to it) with *Aircraft Commander* on your Apple.

□ Take a close look at *Y-Wing* in our K-Byters for this month. This is an example of "total documentation" as we hope to include in some of the future *SoftSide* programs. You'll be able to watch *Y-Wing* grow over the next few months as more and more features will be published.

□ Plus — *Design Master* for the Atari, *Electronics Assistant* for the TRS-80®, *The Sensuous Programmer*, *My Side of the Page*, reviews, hints, and informative articles.

Happy Holidays to all of our readers. 1981 has been a year of change and growth for *SoftSide* and we'd like to thank all of you for your help. May peace, happiness and prosperity be yours in 1982!

□ **Special Note:** Starting with the November issue, all TRS-80® DV programs now use DOSPLUS. Some of the commands are different with this DOS, i.e., to return to DOS from BASIC, use CMD instead of CMD'S.



REWARD! TRANSLATION APPEAL

SoftSide will give a \$100 software certificate to the author of the best translation of a past *SoftSide* feature program. Each month we will publish at least one of these translations. Your portfolio will be enhanced to say nothing of your software library!

We will accept entries for all past *SoftSide* programs at any time. However, we suggest you submit translations of recent programs within three months of their original publication date for maximum consideration. Entries must be submitted on tape or disk, accompanied by complete documentation. Please enclose a self-addressed stamped envelope if you would like your entry returned.

The quality of each translation will be judged by the *SoftSide* editorial staff and prizes will be awarded at the time of publication.

5

ONE LINERS

Scattered throughout the pages of this issue, you will find some very short programs called "One Liners". You can contribute to this department by following three rules and sending your contribution to:

TRS-80®
Atari (pick one)
Apple

ONE LINERS
c/o *SoftSide* Magazine
6 South Street
Milford, New Hampshire 03055

RULES:

1) The programs must be written as a single line of BASIC.

2) The program must be self-contained. Make no assumptions about DIMENSIONS, available string space, current graphics mode, etc.

3) The program can be of any type: graphics, sound, text, utility, etc.

5

Hints & Enhancements

continued from page 7

it involves is a simple pin modification to one of the IC chips. Anyone can do it, and it only takes a minute or two. Once completed, you have your choice of: 1. HGR (with all normal functions), 2. HGR2 (with normal full screen graphics), or 3. HGR2:POKE-16301,0 (with the exact functions as HGR).

Here's what to do:

1. Turn off the Apple power supply. I always pull the plug to be sure.

2. Carefully remove IC number C12. It's in the same row as the lowermost row of RAM chips, the second from the right. It's a little longer than the other three in that row and bears the number 74LS154.

3. Holding the chip so that the notch is away from you, bend pins 2 and 5 straight out from the chip so that they won't go back in the socket. Pin 2 is the second pin from the notched end, on the left side, and pin 5 is the fifth pin from the notch, on the left side.

4. Add a small gauge wire from pin 2 to pin 16. Pin 16 is the first pin from the notch on the right side.

5. Add a small gauge wire from pin 5 to pin 8. Pin 8 is the last pin from the notch on the left side.

Be sure that the wires touch only the referenced pins, and replace the chip in your Apple. Be sure pins 2 and 5 don't touch anything else when reinstalled.

There you have it! Now HGR2:POKE-16301,0 is the exact equivalent of HGR, except that now you've got another 8K of memory to play around with before you need to worry about interfacing with Hi-Res graphics.

One of your readers seemed to feel there were lots of problems with your *Galaxia* program. Frankly, I entered it just as published, and it ran perfectly. Apparently the errors were in his listing, not yours.

Robert R. Devine
Adona, AR

BATS AND "YOU"

My son and I really enjoy playing the *Bats* game you published in *SoftSide* (June, 1981). We were rather annoyed with the method of input to move the "YOU". I therefore took the liberty to slightly change a few lines of your program to those listed below.

```
2000 DELETE
2002 CHANGE 100 TO 50
2008 CHANGE "K" TO "Z"
Add the following lines:
```

```
2031 IF Q$="Z" THEN Q=L:GOTO 2032 ELSE Q=Q
2032 ADD L=Q: TO BEGINNING OF LINE AS PRINTED
```

These slight changes have the effect of keeping "YOU" moving in the direction of the last key pushed until another key is chosen or "K" is pushed to halt movement.

I thought you might be interested in trying these changes and seeing whether you find the game somewhat faster and involving as we did.

Thank you for sharing your game with us. I trust you will keep up the good work.

Note: As we own a Model III, we also changed the "YOU" to CHR\$(253) which is the "little man" character.

David Humphrey
Cranbrook, BC, Canada

APPLE PRINTER MODIFICATION

Let me congratulate *SoftSide* and Mr. D. W. Durkee on his fabulous program, *Word-Search Puzzle Generator* (June, 1981). Boy, they're so good that I have to cheat by looking at the answer sheet to finish any of the puzzles I try! My cheers to Mr. Durkee. However, when the time comes for the printer to print the word list, it prints only one word per line, and if you have 60 or so words, a lot of paper is wasted. Here is a routine which will get the printer to print out five words per line (Apple only).

```
740 FOR I =1 TO Z -1 STEP 5: PRINT
W$(I),W$(I+1), W$(I+2), W$(I+3),
W$(I+4): NEXT I
```

Mr. C. G. Burrow
Niue Island

ATARI ENHANCEMENT

When I bought my Atari six months ago, I had previously had experience only with TRS-80® BASIC and Z80 machine code. (I've owned an TRS-80® for three years now.) So at the time I wrote *Character Generator* and the *Quest 1* translation I was only familiar with Atari BASIC.

Since then I've taken the time to learn 6502 machine code, in the interest of improving the overall quality of my Atari and Apple programs. Presented below are two program modifications for speeding up the down-loading of the character set. The *Character Generator* improvement is the same one included in the cassette and disk versions for October. It now writes the faster loading routine with the character set. The *Quest 1* mod is being published for the first time.

In the lines below, the variable X\$ or XFR\$ consists of a string of character codes ranging from 0 to 255. In order to get these codes, use the following chart. In this chart, a character that is underlined means an inverse letter (use the Atari logo key), and a character that is overlined means a control character (use the CTRL key).

```
XFR$ = "h,EKEM).EN%jXiAEL
1MQKPyflN%NidPm."
```

Quest 1 changes:

```
30020 DIM XFR$(38):XFR$="
":Z=USR(ADR(X
FR$)):RESTORE 30050
```

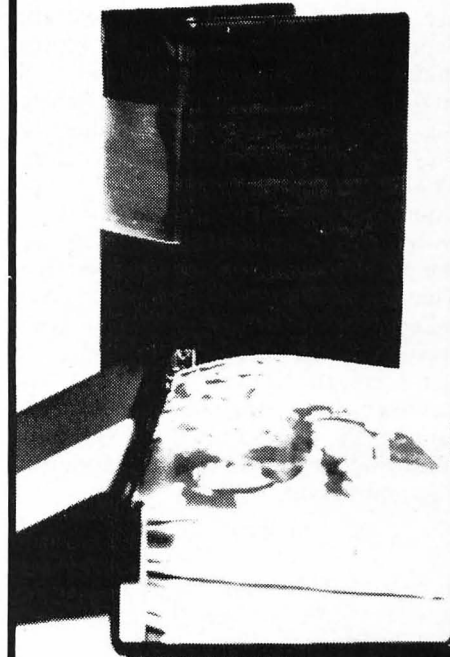
Character Generator changes:

```
650 PRINT #3;"32010 DIM XFR$(38):XFR$=
";CHR$(34);"
";CHR$(34)
660 PRINT #3;"32020 Z=USR(ADR(XFR$)):R
ESTORE 32100"
10030 ? "INITIALIZING . . ."
10040 DIM XFR$(38)
10050 XFR$="
"
10060 Z=USR(ADR(XFR$)):FOR Z=0 TO 127
```

Alan J. Zett
SoftSide ☺

SoftSide

Protect Your Investment!



With *SoftSide*™ Vinyl Binders

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 supply lasts) with the purchase of Volume I or II (12 issue collection of *SoftSide*).

SMALL \$3.95
8½ x 11 \$7.95

SoftSide™
6 South Street Milford NH 03055



VARPTR EDITed

by John T. Phillipp, M.D.

Although I said in my articles on the use of VARPTR (July, August and September *SoftSide* issues) that strings that have been packed cannot be EDITed, since they turn into garbage, it is becoming common knowledge that they CAN be EDITed after packing. The "secret" is to replace the first "quote" — CHR\$(34) — of the string with an asterisk — CHR\$(42). This has the effect of "de-tokenizing" the line. That is, the line now has all the command words in their fully-written-out forms, not as the "token" codes. In that form, the line can be EDITed. The de-tokenizing is done (for line #20 containing A\$, for example) by typing the following in the Command Mode (no line numbers):

```
I = VARPTR (A$): J = PEEK
(I+1) + 256 * PEEK (I+2)
[ENTER]
POKE J-1, 42 [ENTER]
```

The first line uses VARPTR to find the memory location of the first character of the text of the string. The second line changes the memory location before that (the first "quote") to an asterisk. Write down the value of J-1 (PRINT J-1) somewhere. EDITing a line sets all variables to "0" and you'll need J-1 later. The line containing A\$ may now be EDITed. However, the whole command word representing any "token" number you want to add or change must be INSERTed or DELETED, not the number itself. For example, suppose you had packed A\$ with three CHR\$(175)'s. Token "175" represents the command word "LPRINT", so when you LISTed the string, you would see:

```
20 A$ = "LPRINTLPRINT
LPRINT"
```

This is the same as A\$ = CHR\$(175) + CHR\$(175) + CHR\$(175). The altered string (after the asterisk was POKEd) would look like:

```
20 A$ = *LPRINTLPRINTLPRINT"
```

If you want to change the last CHR\$(175) to a CHR\$(191), you would use a table of "tokens," and find that 191 expands to "USING". You would DELETE the word "LPRINT" from the string in the EDIT mode, and INSERT the word "USING". Your string would then look like this:

```
20 A$ = *LPRINTLPRINTUSING"
```

This is the same as A\$ = CHR\$(175) + CHR\$(175) + CHR\$(191). The final step, after EDITing, is to replace the asterisk with the original "quote." This will "re-tokenize" the line (convert all the command words back to their "tokens"). In the Command Mode:

```
POKE the value you wrote down for
J-1, 34 [ENTER]
```

Congratulations, you've just EDITed a packed string . . . and the string has survived.

Of course, it's a pain in the neck looking for a table of which "token" stands for which command word, so I wrote the following "one liner". When RUN, it prints a list of all the graphic character values — 128 to 191 — and the command words they stand for. Just the thing to refer to when EDITing packed strings . . . but you'll have to hit BREAK to get out of it!!

```
1 CLS:L = 128:FORA = 5712TO
5982:IFG = 0THENG = 1:PRINT
L;"=" ";CHR$(PEEK(A)-128);:
NEXTA:ELSEIFPEEK(A)>90
THENL = L + 1:PRINT" ",L;
"=" ";CHR$(PEEK(A)-128);:
NEXTA:ELSEPRINTCHR$
(PEEK(A));:NEXTA:FORS =
0TO1STEP0:NEXTS
```

ATTENTION AUTHORS

SoftSide Publications is actively seeking programs, article and review submissions for the TRS-80®, Apple and Atari home computers. This is a chance for programmers as well as users to make some money to help pay for the "computer addiction" and get their efforts out where they can be appreciated.

Programs — *SoftSide* has always been the leader in the field of BASIC software and BASIC remains our specialty. However, with the advent of Disk Version (DV), we can now also offer an outlet for Machine Language and multiple language programs which do not lend themselves to printed versions. Games, utilities and educational software, as well as any other applications for the home computer user are preferred, although we will consider virtually any type of program. Hybrid mixes of articles and programs are also welcomed.

When submitting a program, please be sure to include full documentation of subroutines and a list of variables, as well as a brief article describing the program.

Reviews — Well written, informed reviews of all software for the systems we cover are a regular feature of *SoftSide*. Reviewers should take into consideration all aspects of a particular software package, from speed of execution to programming creativity to the estimated length of time that the product will hold the customer's interest.

Articles — We welcome article submissions of all types, but prefer those specifically geared to the home computer market. We give our readers information as a first priority, but vary our content to include some humor and commentary.

All text, including documentation and descriptive articles for programs should be typewritten and double-spaced. Extra monetary consideration will be given to articles and reviews submitted on machine-readable media (Scripsit, Super-Text II, etc.). Programs should be submitted on a good cassette or disk. TRS-80® BASIC programs should function under both Level II and Disk BASIC.

Send to:

SoftSide Publications
SUBMISSIONS DEPARTMENT
6 South Street
Milford, NH 03055

We regret that due to the volume we receive, we are unable to return submissions.

Be sure to send for our **FREE Author's Guide**. It further outlines the specifics of our submission procedure.

TRS-80 is a registered trademark of Tandy corporation.



Outgoing Mail

by Randal L. Kottwitz

Happy Holidays! They truly are happy ones here at *SoftSide* as some of the problems I spoke of last month are being resolved. We are now in full operation with our fulfillment service and the difficulties some of you have been experiencing with your subscriptions are coming to an end. We've made some improvements in our duplication services (effecting both efficiency and quality) and those of you receiving tapes and disks should be much happier with our services. Thanks for your patience.

Envyrn™ — Some Comments

Our October issue has caused much comment. First, an omission: The **Envyrn** program requires a full 48K TRS-80® with disk. There are modifications for a 32K version in this issue. That is as small as we will be able to pare down the program. We will not be able to make it tape-compatible as the MIDS command is handled in a different manner. There's no doubt that **Envyrn** is "RAM hungry" and in order to fully utilize its capabilities you need 48K.

There is an awkward element in the tile-editing function of the current **Envyrn** (something akin to chiseling). The commercial TRS-80® **Envyrn** is scheduled to be shipped in January. We now have a tile-editing function which operates more like a paintbrush. What was tedious concentration in the old is creative enjoyment in the new.

Those of you operating on the Atari and Apple have reason to be encouraged. We have no definite release date for the Apple and Atari **Envyrn**, but the project is proceeding at a rapid pace. We are especially impressed by the Atari's ability to handle the meat of **Envyrn** — graphics. The result will most definitely be worth your wait.

Users' Groups

You may have already noticed the bind-in card in this issue asking for information concerning computer clubs and users' groups. We are in the process of compiling such listings for year-ly publication of a directory. We are considering combining the directory with a special issue devoted to software, programming hints and articles

cooperatively produced by clubs and users' groups. I'm interested in your input on this matter. One of the main determinants in the production of such an issue will be the availability of the "raw material." If you or members of your organization are producing software or articles on a cooperative basis, please get the details in the mail to me soon so a decision may be reached on further project development.

Multi-System Format

In October, *SoftSide* ceased publishing letters promoting name-calling from one system to another. However, the letters concerning our multi-system format continue to pour in and some important points are being made. When I purchased my first computer, I was told I'd be a fool if I were to think this was my last computer and not to be surprised if I owned more than one at a time. I'm now convinced the comment was correct. As our world becomes more and more "terminal oriented," familiarity with more than one computer will become an important skill. *SoftSide* has a goal of encouraging cooperation and exchange of information among computerists and expanding the computer literacy of our readers. We received one letter which best exemplifies the attitudes we take toward different machines. The writer, Dr. Krell, owns a TRS-80® and an Atari 800, however, his comments could as easily apply to the Apple.

Dear *SoftSide*,

In comparison, I like both of my machines for different functions. The speed and instruction set of the Z80 CPU in the TRS-80® make it ideal for performing scientific and database computations and for uploading and downloading programs and data to mainframe computers. However, programming graphics on the Atari is quite addictive. As an owner of both machines, I resent the blatant ignorance with which people put down machines which they do not own. The home computer market is quite expansive and has room for several types of computers. So let's quit the name-calling and start to be open minded about the differing good and bad qualities of each machine as a function of its intended use.

Bruce E. Krell, Ph.D.

Thanks, Bruce. Merry Christmas and Happy Hacking!

K-Byters

ANOTHER PROGRAMMING CHALLENGE

Some time ago *SoftSide* began inviting its readers to submit "One Liners" — self-contained single-line programs for the TRS-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, TRS-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 a \$10 software certificate for their programming excellence!

Send submissions to:

K-Byters, c/o *SoftSide*
6 South Street
Milford, NH 03055

Your Adventures



June Adventure of the Month
Arabian Adventure

As Sinbad, the mightiest sailor in ancient Arabia, your mission is to rescue Princess Jasmine from the clutches of the Wizard of Darkness. You will cross the Seven Seas to the deadly Cyclops Mountain, and do battle with skeletons, a one-eyed beast, a hairy tarantula and more monsters who try to thwart your noble pursuit.



July Adventure of the Month
Alien Adventure

You are the sole survivor of a crew on a mission to deliver a cargo of oil to Earth. A crash landing has left you stranded on a small planet, harshly alien but rich in lead, gold and platinum. You must find provisions and a means of leaving the planet. But beware of the **THING** that massacred your crew!



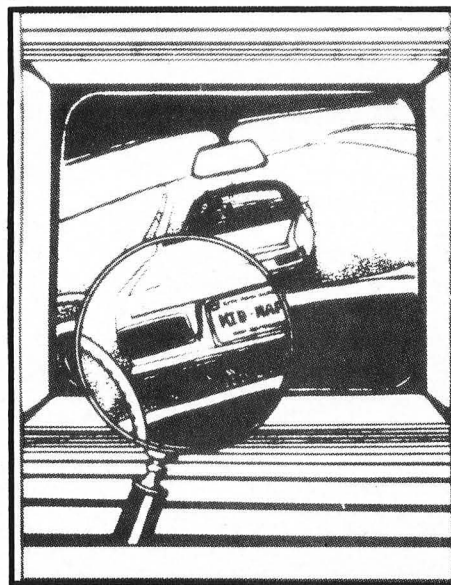
August Adventure of the Month
Treasure Island Adventure

You are a hardy adventurer in search of fame, fortune, and whatever else you can get. You find yourself on an island where there is rumor of pirate's treasure. But watch out for the evil magician and the underground torture chamber! You may end up in a spot where all roads coming into it are paved with good intentions. . .



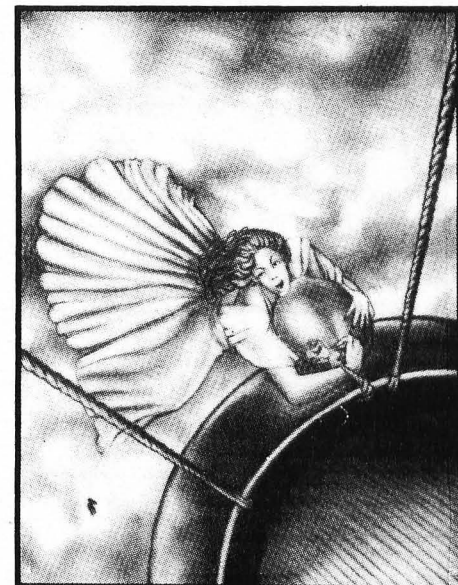
September Adventure of the Month
Jack The Ripper Adventure

Jack the Ripper is running rampant in London and you must stop him! Scotland Yard demands that 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!



October Adventure of the Month
Crime Adventure

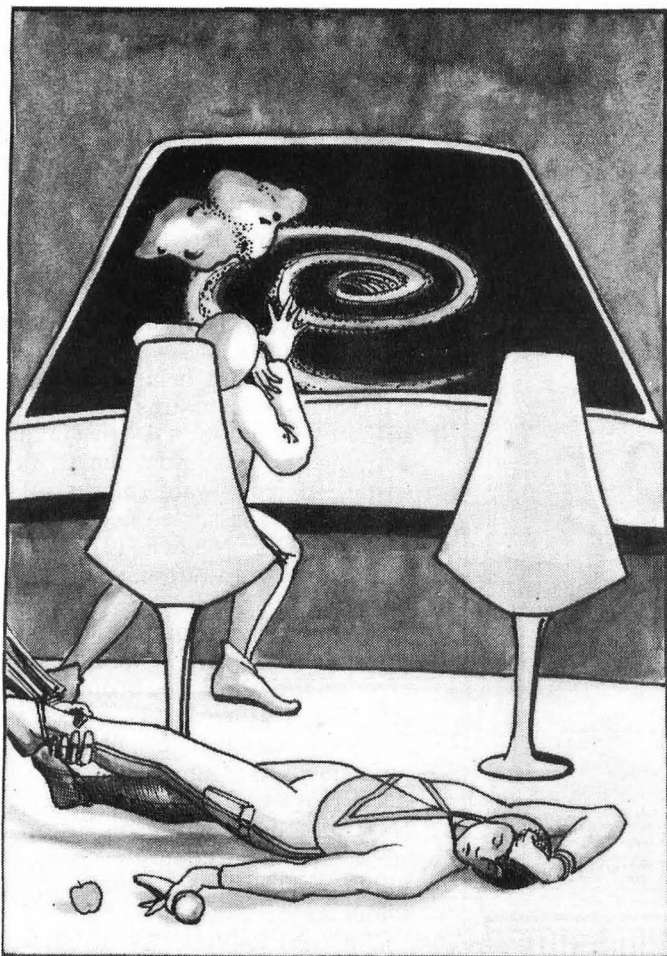
Test your skills as a detective by sifting through hundreds of clues. You may have to become the new Sherlock Holmes to solve this one! Look for the strange, but don't overlook the obvious, as you try to find Mrs. Fenwick and return her to where she belongs.



November Adventure of the Month
Around the World in Eighty Days Adventure

Try to repeat the feat of the classic novel, complete with a balloon and other exciting features of the original adventure. Are you ready to take the challenge? Bon voyage!

will start here —



December Adventure of the Month Black Hole Adventure

The crew of an interstellar craft discovers the long-lost Deep-Space Probe One, the Cygnus, at the edge of the vortex surrounding an immense black hole. See if you can foil the plans of Dr. Hans Reinhardt.

Subscribe to Adventure of the Month

How would you like to go back in time to 19th century London to match wits with Jack the Ripper? Out into space to brave the swirling vortex of a black hole? Into the depths of the ocean, or on a quest to rescue a beautiful princess from the clutches of evil monsters?

You never know where **SoftSide's Adventure of the Month** might take you. But you can be sure that each month you will experience new delights and new challenges as you receive an original adventure on tape or disk, ready to load into your computer.

The cost? A six-month membership is just \$27 for the tape (\$4.50 per adventure) or \$45 for the disk (\$7.50 per adventure). If you're not sure that you can take six full months of excitement, you can order a single tape for \$6 or a disk for \$9. Or, if you're especially adventuresome, we're offering two disks, each packed with three great adventures, for only \$24 per disk.

Please use the coupon below (or the bind-in card in this issue) to order.

Adventure of the Month 6 South Street, Milford NH 03055

Yes, I'm ready to start! Send me Adventures

■ **Six month subscription:**

Cassette (\$27)

Disk (\$45)

■ **Individual adventures (please specify)**

Cassette (\$6)

Disk (\$9)

■ **Three-adventures on one super disk (\$24 each):**

Arabian, Alien, & Treasure Island Adventure

Jack the Ripper, Crime & Around the World Adventure

Please specify which computer:

Apple® (req. 24K for tape, 32K for disk)

Atari® (req. 32K for tape, 40K for disk)

TRS-80™ (req. 16K for tape, 32K for disk)

Name _____

Address _____

City/State _____ Zip _____

Payment enclosed

MasterCard VISA Name of Cardholder _____

MC# and Interbank#/VIS# _____

Exp. Date _____ Signature _____

Prices subject to change without notice. Apple, Atari and TRS-80 are registered trademarks of The Apple Computer Company, Warner Communications and The Tandy Corporation respectively.

The Sensuous Programmer

by "J"

MEET ME AT EIGHT

In the past few weeks I have actually received three letters. This surprises me somewhat, since deep down inside I strongly suspect that no one is out there. Oh, they tell me that some 20,000 copies of this magazine get distributed all over creation, but I usually just assume that they're pulling my leg. I'm not even sure that the letters aren't simply part of a conspiracy to convince me that they're not lying. Despite these solipsistic tendencies, though, I shall take a few lines to respond to those who have written (whether they be real or chimerical).

Two of the letters kindly called my attention to the fact that the Apple stores the line number in which an error occurs in memory locations 218-219. (In the September column I had confessed my ignorance of these addresses.) The line number is given by the quantity, $PEEK(218) + PEEK(219)*256$. Although one of my informants claims that this information is in plain black-and-white on page 135 of the *Applesoft Reference Manual*, I was unable to verify that assertion. It seems that the bottom quarter of that page, including the bold heading concerning ERROR MESSAGES, has been completely obliterated from my copy . . .

The third letter was more emotional. You may recall from the September column (vain hope!) "J's Theorem," "All computer users are idiots;" and "J's Corollary," "All computer programmers who don't believe that all computer users are idiots, are greater idiots." These shocking assertions introduced a discussion on idiot-proofing programs: writing programs so that user errors would not cause the program to crash in an ungracious way. They also provoked a response from one reader which deserves reprinting in part:

"May I offer you Hall's Law of Computer Journalism: The more small systems an author pretends to be knowledgeable about, the more likely is his advice to be wholly without redeeming practical merit.

"J's Corollary: What does it take to know an idiot?"

"... Atari owners are given nonsen-



sical and wasteful advice on protecting programs from unwanted INPUT responses. The Atari is made to look like an incompetently designed system with the suggested error-checking code . . .

"Input integrity can be accomplished in several simple ways on the Atari. Where the user response will be one digit or letter, a GET command is error-free, and the response can easily be checked against all valid entries. Where the numerical input may be multi-digit or negative, a TRAP command prior to the INPUT statement

which leads to a PRINT statement telling the user his input was not a valid response followed by a loop back to the TRAP will avoid any problems where non-numeric characters are entered to an arithmetic variable. Need I add that TRAPs so set can and should be cancelled once the response passes the test?

"As for strings being too long, the Atari allows for limiting the length of a string by its unique DIM technique of defining the maximum length of the string. If you want names of ten characters or less, you need only program `DIM NAMES$(10)`. Extra characters will automatically be ignored.

"While I am being critical, the expression `ASC(A$,1,1)` would cause an error on the Atari. The proper syntax is `ASC(A$(1,1))`.

"As a designer of programs for amateur users, I am well aware of the need for user-friendly and error-trapped code. The reason is not, as J says, because computer users are idiots, but because computers are idiots . . .

"... As an Atari owner and programmer, I am tired of seeing implicit criticisms of the Atari appearing in articles authored by people who are hung up on their love for the TRS-80® or Apple."

Mr. Hall makes several valid points. The most obvious one is that it really is the computer and not the user who wins the contest for greater idiot. (On the other hand, who designed the infernal machine??) The next most obvious one is that `ASC(A$(1,1))` is indeed the correct syntax. (Sorry about that.) And the most helpful one to me was the point about DIMensioning a string to the maximum desired length in order to truncate unwanted characters. I had suggested using a statement such as `IF LEN(NAMES$) > 10 THEN NAMES$ = NAMES$(1,10)` following the input of the name. Mr. Hall's suggestion is certainly simpler and more elegant, and deserves passing along as a programming hint.

On the other hand, the matter of trapping bad input is still open to some debate in my mind. Which is simpler

and more elegant: to test input with a statement such as the one I suggested (IF A > 57 OR (A < 48 AND A <> 46 AND A <> 45 AND A <> 43) THEN 100), or to use a TRAP statement with accompanying error-handling routine, associated PRINT statements, a loop back to the INPUT statement, and cancellation of the TRAP? Both are valid programming methods, and I wouldn't want to be so dogmatic about which is preferable.

Finally, on the matter of implicit criticisms of the Atari, let me make one thing explicit. While Atari BASIC has many superior features, especially its sound and graphics enhancements, it is weaker and less versatile in other areas. The way in which it handles strings, in particular, is usually more of a frustration than an asset. Not always, but usually. A program originally written in another dialect of BASIC almost invariably requires more coding when converting it for the Atari — often a great deal more, when it makes heavy use of string arrays. Personal preferences aside, this is a reality, and it makes me very eager for the release of Microsoft's BASIC for the Atari (which ought to be available on disk by the time you read this).

I work every day with all three computer systems. While I did make a choice of one of them for my home computer, I've enjoyed learning more about the strengths of each of them. I don't think that any of the three is inherently superior to the others; and all three have weaknesses and shortcomings that leave me pounding my head against my desk some days. I don't pretend to be an expert in any of them — just a learner, like most of you, eruditely expounding what I'm learning.

Well, that took up enough space that I don't have to come up with as much original material this month. Last month I did promise a further discussion of arrays, and since that's not a really big topic it will be a good one to tackle with my remaining resources. Not only that, but it relates well to the above comments about the Atari's string-handling methods.

An array is simply a numbered list of items. The items might be numbers or strings, and the list might have one or more dimensions. The following is a grocery list which can be considered an example of a one-dimensional string array:

```
Groceries
(0) milk
(1) bread
(2) cheese
(3) eggs
```

```
(4) frozen pizza
(5) extra pepperoni
(6)
(7)
(8)
```

The numbers in the parentheses are, of course, the item numbers (or index numbers), and the groceries are the actual items (or elements) in the array. Notice that the item numbers start with 0 rather than the more socially acceptable number 1. Computers normally start numbering everything with zeroes (a result of their binary upbringing), so you have to get used to having one more array element than the highest item number might lead you to expect. Notice also that while the item numbers go up to 8, reserving nine places in the list/array, not all of them are filled with actual items. This one-dimensional array, then, has a dimension of 8 (actual capacity of 9), with the last three places occupied by the "null string."

In the case of the actual grocery list which you would write out on a piece of paper, you would often put a title on it so that you would know at a glance what sort of list it was. Computers appreciate titles on their lists/arrays as well; in fact, they insist on them. When giving a title to an array for the benefit of the computer, there are several things to keep in mind. First, the title must be a legal variable name; second, the title must indicate the type of array you want to set up (real numbers, integer numbers, or strings); and third, there must be a statement telling the computer how many spaces to reserve for items in your array. The following lines of code will create the above list as an array in the computer's memory and fill in the items listed:

```
100 DIM GROCERIES$(8)
110 GROCERIES$(0) = "MILK"
120 GROCERIES$(1) = "BREAD"
130 GROCERIES$(2) = "CHEESE"
140 GROCERIES$(3) = "EGGS"
150 GROCERIES$(4) = "FROZEN
PIZZA"
160 GROCERIES$(5) = "EXTRA
PEPPERONI"
```

The dollar sign, as usual, indicates a string variable, and the DIM statement sets the maximum size of the array.

Such string arrays are not allowed by Atari BASIC or Apple Integer BASIC (although numerical arrays are allowed). In these BASIC dialects, the DIM statement is used quite differently: It defines the maximum number of characters that a particular string variable may contain, as discussed in

the October column. Lacking the capability of defining string arrays, it is nevertheless possible to simulate them using a bit of additional coding. This involves creating one very large string composed of lots of little substrings, and then somehow keeping track of where each of those substrings (each simulated array element) is located. The length of this string is limited only by available memory on the Atari, but is confined to 255 characters by Apple Integer BASIC.

One method of keeping track of these substrings is to store them at regular intervals within the larger string. This requires that you know how long the longest "array element" (substring) is going to be, in order to allow enough of a gap between them. If you know that the longest element will be ten characters, for instance, then you can store the first element in positions 1-10, the second in positions 11-20, and so on. To make room for six such elements, you would need a string DIMensioned to 60 characters. Then, to access the nth element of the array GROCERY\$, you would look at characters (n*10-9) through (n*10). Here's an example (I've shortened the string name 'cause I'm getting tired of typing it):

```
200 DIM G$(60)
210 G$(1,10) = "MILK"
220 G$(11,20) = "BREAD"
230 G$(21,30) = "CHEESE"
240 G$(31,40) = "EGGS"
250 G$(41,50) = "PIZZA"
260 G$(51,60) = "PEPPERONI"
270 PRINT "CHOOSE ONE (1-6): ";
280 INPUT NUM
290 PRINT G$(N*10-9,N*10)
300 GOTO 270
```

Notice that the elements are filled up to ten characters, using blank spaces, when they are plugged into the string. Without this or a similar precaution, the remaining characters might be filled with miscellaneous garbage. An alternative method is to fill the entire array with spaces before it is used at all, with a loop such as this:

```
205 FOR I = 1 TO 60: G$(I) = " ";
NEXT I
```

This has the added advantage of filling out the whole array, so that elements of it which are not yet plugged in will have their spaces reserved for them. If this is not done, an attempt to access an "un-entered entry" might result in a program crash.

An alternative method for keeping track of the elements in a simulated string array is to create a numerical ar-



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 7 PM WEEKDAYS
11 AM TO 6 PM 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 7 PM WEEKDAYS
11 AM TO 6 PM WEEKENDS

THE NORTHEAST COMPUTER SHOW

BOSTON
Hynes Auditorium
PRUDENTIAL CENTER

THURS-SUN
OCT 15-18, 1981

11 AM TO 7 PM WEEKDAYS
11 AM TO 6 PM 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 7 PM WEEKDAYS
11 AM TO 6 PM WEEKENDS

THE SOUTHERN CALIFORNIA COMPUTER SHOW

LOS ANGELES
LA Convention Center
1201 SOUTH FIGUEROA

THURS-SUN
MAY 6-9, 1982

11 AM TO 7 PM WEEKDAYS
11 AM TO 6 PM WEEKENDS

ray to hold index numbers. In this method, each element of the numerical array is a pointer to the location of the corresponding element of the string array. Let's say that the simulated string array is to be stored in G\$, and we create a numerical array P(n) to point to the elements of the array. We can then set P(1) equal to the location of the last character of the first string element, P(2) equal to the location of the last character of the second element, and so on. Element N of the string array can then be accessed by looking at characters P(N-1)+1 through P(N) of G\$ — that is, from the character following the last character of the previous element, through the last character of the present element. This would be expressed as G\$(P(N-1)-1,P(N)).

In this approach, the P(n) array needs to be dimensioned to the maximum number of elements that will be stored in the simulated string array, and P(0) needs to be defined as 0 in order to properly access the first element of the array. (Editor's note: This is the method used in the Atari version of *Microtext 1.0* elsewhere in this issue, to keep track of the location of each line of text in T\$.) The advantage of this method over the first is that it doesn't waste string space by reserving, say, 20 characters for each substring when many of them might be only a fraction of that length. The disadvantage is that an element cannot be changed very easily, since it is fitted into its own precisely-sized niche. To replace it with a longer element involves reorganizing the whole simulated array, along with the numerical pointers. The situation is exactly the same as the choice between random and sequential disk files, and only the requirements of the particular situation can determine which approach would be best.

So far I've been talking about one-dimensional arrays. They are the most common type, but there are times when two or three or even more dimensions come in very handily. Atari BASIC allows two-dimensional numerical arrays, and Applesoft and TRS-80® BASIC allow n-dimensional numerical and string arrays (i.e., any number of dimensions). If a one-dimensional array can be visualized as a list, then a two-dimensional array is like a chart with numbered rows and columns: Any particular entry can be accessed using two index numbers, specifying the row and the column in which it is located. A two-dimensional array named CHART with ten columns and four rows would be set up by the statement DIM CHART(9,3).

Likewise, a three-dimensional array can be visualized as a rectangular solid built of cubical cubbyholes — each cubbyhole capable of storing one element of the array, and accessible by specifying its unique row, column, and "depth" into the array. Arrays having more than three dimensions are quite rare, and more difficult to visualize, but you may occasionally find an application for one.

When setting up multidimensional arrays like this, it becomes even more important to remember that there is a zero index in each of the dimensions. If CHART were DIMENSIONED (10,4) instead of (9,3), room would be reserved in memory for 55 entries instead of 40. While this isn't a large waste, it can become more significant when using more dimensions or larger arrays. A numerical array DIMENSIONED (500,2) would waste 504 storage cells if only a 500-by-2 array were actually needed. And those storage cells aren't just one byte of memory each; for a real-number array, each element occupies four to eight bytes of memory (depending upon the computer and whether a double-precision mode is specified).

In addition to making use of the zeroth array index, defining an array as an integer array (in Applesoft or TRS-80® BASIC) can save vast quantities of

memory. This limits you to storing whole numbers between -32767 and +32767, but that's often no problem at all; and each element takes only two bytes of memory rather than four or more. Defining an integer array is as simple as adding "%" to the variable name in the DIMENSION statement and all subsequent uses of it, or using the TRS-80®'s DEFINT statement to avoid having to type the "%" each time.

The uses of arrays are legion. Perhaps the most obvious application is to store a quantity of numbers or strings that need to be accessible to your program, and which may also need to be saved onto a tape or disk for later use. A less obvious application crops up in all kinds of programs (not just ones that process a lot of data): avoiding repetitive code. If you find yourself writing sections of program code that are virtually identical except for a variable or two, there's almost certainly a way to conserve time and space by using an array or two in conjunction with a loop. With proper use of arrays, loops, and (those other codesavers) subroutines, there should be almost no occasion for duplicating a single line of code in any program. a single line of code in any program. a single line of code in any program. 5

The powerful package:

Super-Text II™

Allows you to learn the basics of text editing quickly. Advanced features will meet your expanding word processing requirements far into the future. \$150.00

plus Form Letter™

Provides automatic repetitive printing of letters. Allows insertion anywhere in a letter, also direct entry, optional prompting, special commands. \$100.00

plus Address Book™

Stores names, addresses, and telephone numbers and prints mailing labels. Has user-definable category system. \$49.95

**From the leader in word processing
for the Apple II or II Plus**

MUSE SOFTWARE™

Apple II is a trademark of Apple Computer Corp.

330 N. CHARLES STREET
BALTIMORE, MD 21201
(301) 659-7212

Call or write for information and
the name of your nearest MUSE dealer



by Allen L. Wold and Fred D'Ignazio

SoftSide welcomes Fred D'Ignazio and Allen L. Wold whose *Entertainment Tomorrow* column will appear each month.

Fred D'Ignazio is a computer enthusiast and author of several books on computers for young people, including: *The Creative Kid's Guide to Home Computers*; *Katie and the Computer*; and *Small Computers: Exploring Their Technology and Future*. He is presently writing a series of books on how to create graphics-and-sound adventure games on the Apple, Atari, Commodore VIC, and TRS-80® Color Computer.

Allen L. Wold is a science-fiction writer with two published novels, *The Planet Masters* and *Star God*, and a dozen or so others in various stages of conception or revision. Wold is a long-time devotee of role-playing games and other forms of gaming.

D'Ignazio and Wold do their writing on word processors. Both see the computer not just as a tool or toy, but as a force that is profoundly shaping our lives. Both are optimistic about future computer applications and believe in the value of bold speculation about computers and their impact on our lives.

Computerized games and recreation are becoming more common, more complex, and more sophisticated all the time. For an admittedly biased example, we can take a look at the November/December (1981) issue of *Games* magazine in which the editors list their 100 favorite games. Twenty-six of these are electronic, including one construction set, one robot tank, five hand-held action games, four home video systems, and six board games. Among the "top" games were computerized chess and backgammon, computerized bridge, computerized *Dungeons and Dragons* (Mattel), a fascinating computer-moderated board game called *Dark Tower* (Milton Bradley), and a version of the old *Jungle Game* (once called *Stratego*) but now called *The Generals* (Ideal).

In last year's list of favorites, there

were 23 electronic games. The games, though almost as numerous, were less complex and less flexible than the current crop (most of them were designed to play only one limited game). This year there are many "multi-game" games, and those which do only one thing have become more sophisticated. For example, a grand prix auto race game features impressive 3-D graphics on a liquid crystal display (LCD). Another game challenges you to escape from 1000 mazes and displays each maze in a 3-D prisoner's-eye view from *inside the maze*.

Games of this sort are only a small part of the far larger field of electronic and computerized recreation, but they indicate a massive trend. Let's take an extreme example, to see where this trend might lead us.

Some time in the not too distant future, when you telecommute home from a hard day's work and want a little recreation, it might go like this:

You leave your work terminal in your study and wander into your family "media" room and flop down in a comfortable chair. You put on something that looks like a cross between a football helmet and a welder's mask. It is very lightweight and consists of a wrap-around video screen, earphones, microphone, and a tiny, extremely short-range radio. That radio links the microchip computer implanted in your skull to the super-fast computer that dwells in distributed nodes inside the walls of your home.

(The information wiring diagram of your home looks like a sketch of the neural nets and clusters of ganglia that make up the human brain. In a sense, your house is a brain, and you are a mobile information node that migrates around the brain's interior.)

With the helmet in place, you are now in a direct link with your computer system, which in turn is linked to every other computer in the world via laser-microwave satellites. The chip in your head is wired to portions of your

brain. You have learned, by biofeedback methods, to use the chip like an extra hand, except the "hand" has a thousand fingers and works with a mere thought. It is the controller of a complex computer terminal device.

As soon as the helmet is in place, the LCD screen forms an image, letting you choose from a number of network circuits monitored by your house computer. You can pick news, movies, communication with other people, education, or games.

You decide on games, and the image on the screen clears. As your eyes adjust to the close focus, it no longer looks like a screen. It's just like normal vision, except that the graphics quality is definitely (though subtly) superior. That quality is there to remind you that what you see is an electronic fantasy, not the real thing.

You can scan different radio frequencies, which you perceive analogously as different footpaths through a forest, as dirt roads, country roads, and superhighways.

Whooop! With a feeling of roller-coaster acceleration and a slight touch of vertigo, you land on a high-speed game channel. You're automatically up and running on any of a hundred old games you've recently abandoned or you can start a thousand other games from scratch. You float over a game menu that scrolls swiftly beneath you, indicating in sights and sounds all the fabulous forms of entertainment that can be instantly yours. You make your choice (you don't have to speak or push buttons, you just think): **ADVENTURE!**

You see new images, new sets of choices, and a picture that indicates that there are others on the laser-and-fiber-optic, global network who, like you, want partners to play with or against. *Satellite Spy* sounds good tonight, and there are two others who want to play too, so with another thought you switch on the game.

The computerized game moderator

(an Artificial Intelligence) relates the game rules to you by shuffling through a fast sequence of visuals and by talking like a tobacco auctioneer over your audio channel. You have heard its spiel before, so you shut your eyelids tight, mentally call up an override, and leap directly into the game. All three of you — the other two are probably thousands of miles away — are experienced gamers, so tonight there will be secret rules which you'll have to discover to make your best score.

Suddenly you see that you are in the lounge of a military experimental satellite, with ten other people. Two of them are your opponents, the others are "simulacra" created by a program running on a large computer multi-network that probably straddles a good part of the planet. Part of your mission in the game is to figure out who your opponents are, and which of the simulacra will help you steal the particle-beam jamming device. But you don't think in those terms, since all you know is that you are a spy, your nation's security depends on your obtaining the device, and you have only two hours (in real time) in which to do it.

Whenever you need information in order to play the game, an electronic voice tells you what you need to know. Otherwise, you are on your own. You explore the satellite, get to know the other people — it is almost impossible to tell the real ones from the false — and plan your strategy. One person tries to seduce you, but you are suspicious. Another seems angry with you, but you figure out that here is your ally, pretending to hate you so others won't suspect.

You find out that the device is not on board, but the device specifications are recorded in the satellite's computers, and you have to break computer security to get the information. It doesn't matter that you know nothing in real life about computer cryptography and parallel operating systems, since the simulation provides you with a means, if you can figure it out.

You know you are on the right track when you discover that one of the secret rules concerns telepathy. You quickly master the skill (you've played a lot of magic-oriented games) and learn that only the commanding officer knows which data bank contains the secret. But you also learn that someone else has read his mind before you. One of your opponents is ahead of you.

You suspect one of the characters is an opponent, so you pick a fight with her. She is much stronger than you, but

you fight dirty, and you win.

Was your guess correct? It better have been because you've now precipitated the end-game.

You have to find the particle-beam data and get out at once. But you discover that the data bank has already been raided. The information has been randomly recoded. The jamming-device specifications are lost. Your mission is aborted. You have only one source of action remaining: Escape.

You escape from the satellite aboard a small, one-person garbage tug that has put in at the satellite for fuel and repairs. You smile grimly as you crouch, not moving a muscle, concealed beneath a huge pile of foul-smelling refuse.

When the game is finally over, the pictures before your eyes become more stylized. You see representatives of two figures, your opponents in the game. The person you neutralized was one of them, but the other was the Security Officer, whom you assumed to be a simulacrum, but who was in fact the other spy, playing a double game, as her skill required.

You made points for locating the jamming-device specifications, for figuring out and using telepathy, for neutralizing an opponent, and for identifying one of the simulacra as a helper. The person you neutralized is the real loser of the game, having only located the data without retrieving it. But even he made points for mastering the social situation, to which you paid no attention at all.

Your other opponent, the Security Officer, is the winner. She prevented you from stealing the jamming device, and then stole it herself. She escaped successfully from the satellite, and even discovered a plot to conquer the world. In winning the game, she amassed a huge number of points that she can carry with her to the next game like medals or battle trophies.

After the game is over your scores are displayed, your rankings with others who have played this game are compared, and you return to float above the rolling menu display.

You are physically and emotionally exhausted. There will be no more games tonight. Maybe some music. Then sleep.

The above scenario is a simple role-playing game that makes use of some advanced technology. We rushed through the game, but it serves to make a point: electronic entertainment in the future will be absorbing, stimulating,

and mind-boggling. And this is just a taste of what is yet to come.

Satellite Spy represents only a midway point in the development of computerized recreation. It is nowhere near the ultimate. And there are other forms of recreation which are quite different from role-playing games. Computers can take you to new worlds of fantasy and the imagination. Yet as computers continue to shrink, like Lewis Carroll's Alice, they too can enter new worlds. Our games, our game clothes, our sports equipment, even our bodies can become hosts for tiny, implanted computers that moderate a game or act as our teammates or opponents. In the near future, our leisure time will be filled with computerized footballs, hockey pucks, ski bindings, go-carts, and motorboats. Like the sports-equipment company of today in its well-known TV ad, the *sportscomputer* companies of the future will boast that "We make weekends!"

In the coming months, we will use this column to focus on computer-assisted role-playing games — and on many of these other rapidly evolving areas of computerized, electronic entertainment. The point of view we'll adopt is that computerized entertainment (or entertainment of any kind) is not a trivial thing. There is big money in recreation as the arcade video game companies have demonstrated, riding the backs of computer chips out of non-existence in the early 1970s to a Midas-like income of \$2.9 billion dollars in 1980.

Recreation is also the stuff of culture, providing us with music, plays, and literature, as well as games. You can farm with a stick, or farm with a ten-ton tractor, but unless you have a way to play, you have no culture.

In the months to come we hope to explore many surprising facets of electronic entertainment. We will look at some of the possibilities in the near future, some in the distant future, and some that might not really be possible at all. There is a whole area of board games, as hinted at by *Dark Tower* and computerized bridge, to be explored.

There are game applications for a multitude of computer-related technologies, including intelligent video disks, lasers, fiber optics, and other optical (light-based) forms of information processing and storage. There is artificial intelligence, super high-resolution graphics and animation, music and speech synthesis, and computerized vision and speech *understanding*.



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



And there is what science-fiction writer Roger Zelazny calls the "telefactoring" device (what AI expert Marvin Minsky terms a "telepresence"): a super-controlled robot loaded with miniature computers and sensors of every sort. A human operating a telefactoring device from some remote location would electronically see what the machine sees, hear what the machine hears, and feel what the machine feels. He or she wouldn't be present with the machine (inside its metallic skin) but would be telepresent.

Telepresences will at first be used mainly by NASA, the military, oceanographic agencies, and large mining and manufacturing operations. Later on, they might become a new generation of cybernetic gladiators, remote-controlled warrior slaves battling each other in fantastic shapes and disguises while satellite networks cover the action for millions of people cheering in the stands of an electronic global coliseum.

This is only a hint at the types of electronic entertainment that will soon be invented. Humans love games. Games mean big money. Electronics and computers are inexpensive, powerful vehicles for delivering games and making huge sums of money. The rapid evolution of computerized entertainment and the fierce competition among its providers and distributors will accelerate at a dizzying pace throughout the foreseeable future.

We will make frequent use of scenarios in this column in an attempt to give you a feeling for the types of games and recreation the various technologies will actually produce. The scenarios may sound like science fiction, but they will all be based on technologies already in existence or soon to emerge from the laboratories and drawing boards of scientists and engineers. Each month we will select one area of technology that is likely to have a major impact on gaming and recreation. We will speculate on entertainment applications for that technology and extrapolate into the future from a solid grounding in present-day facts.

The trends and technologies are all around us like fast-moving mechanical walkways. We invite you to climb aboard a new walkway each month and join us on a brief but fascinating ride into the future of computerized equipment.

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





by Jon Voskuil

PROGRAM DOCUMENTATION TEXTWRITER, PART 1

Microtext 1.0 is a word processor program for a 16K Apple (with Applesoft), Atari, or TRS-80® (Models I or III).

Although it will have a variety of uses, this program was designed with one specific application in mind. *SoftSide* now requires that all programs which are submitted to us for possible publication, ALONG WITH THEIR DOCUMENTATION, be submitted on tape or disk. This introduction is a guide to the use of *Microtext 1.0*, and to the form of documentation which we need for submitted programs. We are always seeking original, high-quality software from our readers for publication in *SoftSide* or *SoftSide DV* (Disk Version), and for marketing through *SoftSide* LIBRARY. Notice that because of DV and LIBRARY, we are now accepting submissions which are written in Machine Language or which may require the use of a disk drive, in addition to those which are in BASIC and compatible with both disk and tape systems.

SoftSide Documentation Guidelines

When you send a program to us for consideration, here is what your submission disk (preferably) or tape (OK) should contain:

1. A thoroughly tested, debugged version of your program. If you observe the following suggestions, your program will have a better chance of being accepted, and of receiving a higher payment if accepted:

(a) Design your program to be user-oriented. Include an option to print an instruction summary, make every effort to trap unexpected inputs, and prompt the user to do what is expected of him.

(b) Keep line numbers spaced apart, so that the program has a neater appearance, and modifications can be made more easily.

(c) If you document your program using REM statements, please do not reference a REM line with a GOSUB or GOTO; you should be able to delete all REMs and leave the program's operation unaffected. (Try it!) We suggest that the line numbers of all REMs end in a "9."

(d) If your submission is on tape, save it at least twice. (Not a bad idea for disks, either.) We don't advise using bargain-basement tapes, since they are subject not only to data losses but to jamming (especially after a trip through the mail). If it won't load, we can't review it.

2. A ONE-PAGE typed or printed summary — on paper — of how to use the program, such that a computer-knowledgeable person can run through the program with no other information in hand. This is the only thing that we need (or want) on paper. Be sure to include the program's name and your name, address, and phone number on this sheet.

3. One or more text files, created with *Microtext 1.0*, containing your name, address, and phone number, the program's name, and the following three types of documentation:

(a) An introduction to the program. This should contain any necessary background information, hype, etc., plus complete instructions for use.

(b) A programmer's explanation of the coding, describing the various program sections and providing comments on programming techniques which would benefit others. If you use PEEKs, POKEs, CALLs, and other such statements, include notes describing what they do.

(c) A listing of all variables used in the program. These should be in one list (not separate lists for strings, arrays, etc.), in alphabetical order. Briefly describe how each variable is used in the program.

About *Microtext*

Microtext 1.0 is a simple word processor which allows you to write text

onto disk or tape. It was designed with *SoftSide*'s LIBRARY in mind, so that files created by it can be easily displayed and automatically formatted on the video screen using a simple utility program (which will be published next month). In its present form it is used more like an electric typewriter than a full-featured text editor, requiring that you have your text in reasonably polished form before entering it. Unlike a typewriter, it does allow you to backspace over mistakes on the current line, and it automatically breaks a line between words. The only other editing option available in this version is to delete from any line to the end of the text. Therefore, it's advisable to check each line for errors as you are typing it in, or at least before you have entered very many lines in succession.

If you have no word processor program for your computer system, you'll find *Microtext* to be quite an improvement! On the other hand, if you do have one of the many excellent commercially available, full-featured text editors, you'll find *Microtext* lacking some of your favorite features. (So what do you expect from 4K of BASIC code??) The rationale for requiring its use for submitted documentation is basically consistency. Files created by the many word processors on the market are not all compatible with one another, and may not be easily readable at all without the program that created them. In our LIBRARY we need media-based documentation for all three computer systems that can be easily created and easily accessed by anyone with the appropriate computer and no additional software.

When you run *Microtext 1.0* you will be confronted by a mostly blank screen. The program is self-prompting, and the instructions are simple: Start typing. Because of the TRS-80®'s slower clock speed, that computer's version will seem a bit sluggish for fast typists. But then, more careful typing will generally pay off in greater accuracy. While typing, the keyboard may occasionally seem to "lock up." This happens when the computer

pauses to clear unused string space, and is perfectly normal (though somewhat annoying). This will happen more frequently as your text file grows larger. Incidentally, if you have lower case available on your computer, feel free to use it, keeping in mind that the command codes must be typed as capital letters.

The three command codes allow you to review the text you've entered, save it to disk or tape, and load from disk or tape a previously saved file. In the Apple and Atari versions, these operations are accessed by pressing ctrl-R, ctrl-S or ctrl-L. In the TRS-80® version, press CLEAR and then R, S or L. You can also quit the program using ctrl-Q or CLEAR Q. If you should type in a save or load command inadvertently, you can return to the typing mode by pressing CLEAR (TRS-80®) or ESC (Apple and Atari).

In the review mode the computer will scroll through the text in memory from the beginning, allowing you to pause at any time by pressing any key. Then RETURN/ENTER will continue the review, the space bar will display one additional line and pause again, and X or ctrl-X will allow you to delete all text following the last line displayed. If you choose the last option, you'll be asked to confirm that you do indeed want to delete to the end of the text; if not, you'll be returned to the review mode.

The save option should be exercised occasionally while typing in your text, to minimize the effect of system (or program?) crashes. Loading of a previously saved file can be done at any time; be aware that this will destroy any other text you have in memory at the time.

The DIMENSION statement in line 120, and the CLEAR in line 110 of the TRS-80® version, reserve memory for the strings which hold the text. The numbers listed work fine for a 48K system (40K Atari) with DOS booted, with some memory left over. If you have less than the full complement of memory in your system, you'll need to adjust these downward appropriately; if you're really tight on memory, you can delete all REMARK lines. Or, if you have full memory and no DOS hogging part of it, you can increase the amounts. The Apple and TRS-80® versions use the L\$ array to store the text line by line; the Atari version uses T\$ to store it character by character.

If you're using *Microtext* for program documentation, all the documentation need not be in a single text file. If it can't be fit into one file in memory, feel free to store it in as many files as necessary (but not too many

MORE than necessary!) Currently there is no check to prevent you from trying to write more text than the array dimensions or the computer's memory will permit. (Definitely a matter for later enhancement!)

If you have questions about our requirements for program documentation, please don't hesitate to call (603-673-0585) or write. Since this is a new system, we anticipate that there will be problems and ambiguities to be ironed out. We welcome your suggestions for improvements to the procedures, and for enhancements to *Microtext*. We will be publishing additional modules for the program which will enable you to use it for producing formatted hard-copy printouts, and to add further editing capabilities.

Variables

B\$: Backspace character.
 BKSP: ASCII of backspace character.
 C: ASCII of C\$.
 C\$: Input character.
 CHAR: Current character position in text line.
 CLR: ASCII of CLEAR.
 CR\$: Character to indicate a carriage return.
 D\$: CHR\$(4), for Apple disk operations.
 ESC: ASCII of ESCAPE.
 F\$: File name.
 FT\$: Temporary file name string (Atari).
 H, HH: Horizontal cursor position (Atari, Apple).
 I, J: Loop variables.
 L: Temporary variable.
 L\$: Text in current line (Atari); also used as temporary variable.
 L\$(n): Main string array to hold lines of text (Apple, TRS-80®).
 LIN\$: A line of hyphens (Atari).
 LN: Number of lines in text.
 LNXT\$: Text in next line (temporary).
 LP(n): Pointer to the last character of line n in string T\$ (Atari).
 LWID: Maximum number of characters per line.
 P, PP: Cursor location (TRS-80®).
 RTN: ASCII of RETURN/ENTER.
 SS: A line of spaces (Atari).
 SL, SLOC: Location of last blank space in current line.
 SPC: ASCII of a blank space.
 SS: Number of characters to erase.
 STP: Flag used in review mode to indicate whether to step through line by line.
 T\$: Main string to hold text (Atari).
 V, VV: Vertical cursor position (Atari, Apple).
 X, X\$: General-purpose variables.
 Z: Timing loop variable.

```

1 REM #####
2 REM $
3 REM $ MICROTEXT 1.0 $
4 REM $ APPLESOFT VERSION $
5 REM $ $
6 REM $ BY JON R. VOSKUIL $
7 REM $ 10/81 $
8 REM $ $
9 REM #####
10 TEXT : HOME
20 VTAB 8: PRINT TAB( 8)"M I C
   R O T E X T 1 . 0"
30 VTAB 12: PRINT TAB( 12)"BY J
   ON R. VOSKUIL"
40 VTAB 14: PRINT " COPYRIGHT 19
   81 SOFTSIDE PUBLICATIONS"
50 FOR I = 1 TO 3000: NEXT I

Initialization.

99 REM INITIALIZATION
100 HOME
110 POKE 35,22
120 DIM L$(1000)
130 ESC = 27:BKSP = 8:RTN = 13:SP
   C = 32:B$ = CHR$( 8):CR$ =
   CHR$( 93)
140 CHAR = 1
150 LN = 1
160 D$ = CHR$( 4)
170 PRINT D$;"MON C,I,O"
180 LWID = 38
190 V = 1:H = 1
200 VTAB 23: FOR I = 1 TO 39: PRINT
   "-";: NEXT : PRINT
210 VTAB 24: HTAB 1: INVERSE : PRINT
   "SAVE:CTRL-S REVIEW:CTRL-R
   LOAD:CTRL-L";: NORMAL
220 VTAB V: HTAB H

Main input loop.

499 REM INPUT LOOP
500 GET C$:C = ASC (C$)
510 IF C = ESC THEN C$ = "0":C =
   64
520 IF C = RTN THEN C$ = CR$
639 REM BACKSPACE
640 IF C < > BKSP THEN 720
650 IF CHAR < 2 THEN 500
660 PRINT B$;" ";B$;
670 CHAR = CHAR - 1
675 IF CHAR < 2 THEN L$(LN) = ""
   : GOTO 500
680 L$(LN) = LEFT$( L$(LN), LEN
   (L$(LN)) - 1)
700 GOTO 500
719 REM CTRL CHARACTER
720 IF C < ESC AND C < > RTN THEN
   2000
739 REM END OF LINE

```



```

740 CHAR = CHAR + 1: IF CHAR < LW
  ID OR C = SPC OR C = RTN THEN
  760
750 GOSUB 1000
759 REM ADD TO STRING
760 L$(LN) = L$(LN) + C$
779 REM RETURN
780 IF C < > RTN THEN 880
820 PRINT C$
840 LN = LN + 1:L$(LN) = "":SLOC =
  0
850 CHAR = 1
860 GOTO 500
879 REM PRINT ON SCREEN
880 PRINT C$;
890 IF CHAR = LWID - 3 THEN PRINT
  CHR$(7);
899 REM UPDATE SPC POINTER, CHK
  FOR LINE END
900 IF C = 32 THEN SLOC = CHAR: IF
  CHAR = LWID THEN LN = LN + 1
  :L$(LN) = "":CHAR = 1:SLOC =
  0: PRINT
920 GOTO 500
Subroutine to break line at a space
and to initialize the next line.
999 REM JUSTIFY & INCR LINE
1000 IF SLOC = 0 THEN PRINT : GOTO
  1100
1020 HTAB SLOC
1040 PRINT SPC(LWID - SLOC): PRINT
1050 IF SLOC = LWID - 1 THEN 110
  0
1060 L$(LN + 1) = RIGHT$(L$(LN)
  ,LWID - 1 - SLOC)
1080 L$(LN) = LEFT$(L$(LN),SLOC
  - 1)
1100 LN = LN + 1
1120 PRINT L$(LN);
1140 CHAR = LEN(L$(LN)) + 2
1150 SLOC = 0
1160 RETURN
Subroutine to process command codes.
1999 REM PROCESS CTRL CHARS
2000 V = PEEK(37) + 1:H = PEEK
  (36) + 1
2100 IF C = 18 THEN GOSUB 3000:
  REM CTRL-R
2200 IF C = 19 THEN GOSUB 4000:
  REM CTRL-S
2300 IF C = 12 THEN GOSUB 5000:
  REM CTRL-L
2400 IF C = 17 THEN TEXT : STOP
  : REM CTRL-Q
2900 GOTO 210
Subroutine to review entered text.
2999 REM REVIEW TEXT
3000 VTAB 24: HTAB 1: PRINT SPC(

```

```

39)
3010 VTAB 24: HTAB 1: PRINT "PRE
  SS SPACE BAR TO PAUSE";
3020 HOME : SPEED= 220
3030 FOR Z = 1 TO 500: NEXT Z
3040 IF LN = 1 THEN 3210
3050 FOR I = 1 TO LN - 1
3060 PRINT L$(I)
3070 IF PEEK(-16384) < 127 AND
  NOT STP THEN 3200
3080 STP = 0: SPEED= 255: POKE -
  16368,0
3090 VV = PEEK(37) + 1
3100 VTAB 24: HTAB 1: POKE 35,24
  : POKE 34,23: PRINT
3110 PRINT : PRINT "RTN:CONT SP
  C:STEP CTRL-X:DEL TO END";
3120 GET X$:X = ASC(X$)
3130 IF X < > 24 THEN 3160
3140 PRINT : PRINT "DELETE FROM
  HERE TO END OF TEXT? (Y/N)";
  : GET X$: IF X$ < > "Y" THEN
  3110
3150 LN = I + 1:L$(LN) = "":CHAR =
  1:SLOC = 0: POKE 34,0: POKE
  35,22: GOTO 3000
3160 IF X = 13 THEN 3190
3170 IF X < > 32 THEN 3110
3180 STP = 1
3190 SPEED= 220: POKE 34,0: POKE
  35,22: VTAB VV: HTAB 1
3200 NEXT I
3210 PRINT L$(LN);
3220 V = PEEK(37) + 1:H = PEEK
  (36) + 1
3230 SPEED= 255: RETURN
Subroutine to save text to disk or
tape.
3999 REM SAVE TO DISK/TAPE
4000 VTAB 24: HTAB 1: POKE 35,24
  : POKE 34,23: PRINT
4020 VTAB 24: HTAB 1: PRINT "SAV
  E TO TAPE OR DISK? (T/D/ESC)
  ";
4030 GET X$: PRINT
4040 IF X$ = CHR$(ESC) THEN 42
  20
4050 L$(0) = STR$(LN * 10000 +
  SLOC * 100 + CHAR)
4060 IF X$ = "T" THEN 4190
4070 IF X$ < > "D" THEN 4020
4080 INPUT "FILE NAME: ";F$
4090 INPUT "INSERT DISK AND PRES
  S RETURN ";X$
4100 PRINT D$;"OPEN ";F$
4110 PRINT D$;"DELETE ";F$
4120 PRINT D$;"OPEN ";F$
4130 PRINT D$;"WRITE ";F$
4140 FOR I = 0 TO LN
4150 PRINT L$(I)

```

```

4160 NEXT I
4170 PRINT D$;"CLOSE ";F$
4180 GOTO 4220
4190 INPUT "START RECORDER AND P
  RESS RETURN ";X$
4200 FLASH : PRINT " SAVING ";: NORMAI
4210 STORE L$
4220 PRINT
4230 POKE 34,0: POKE 35,22
4240 RETURN
Subroutine to load text from disk or
tape.
4999 REM LOAD FROM DISK/TAPE
5000 VTAB 24: HTAB 1: POKE 35,24
  : POKE 34,23: PRINT
5020 VTAB 24: HTAB 1: PRINT "LOA
  D FROM TAPE OR DISK? (T/D/ES
  C) ";
5030 GET X$: PRINT
5040 IF X$ = CHR$(ESC) THEN 52
  50
5050 IF X$ = "T" THEN 5210
5060 IF X$ < > "D" THEN 5020
5070 INPUT "FILE NAME: ";F$
5080 INPUT "INSERT DISK AND PRES
  S RETURN ";X$
5090 PRINT D$;"OPEN ";F$
5100 PRINT D$;"READ ";F$
5110 INPUT L$(0)
5120 GOSUB 5290
5130 FOR I = 1 TO LN
5140 L$(I) = ""
5150 GET C$: IF C$ = CHR$(13) OR
  C$ = CHR$(14) THEN 5170
5160 L$(I) = L$(I) + C$: GOTO 515
  0
5170 NEXT I
5180 PRINT : PRINT D$;"CLOSE ";F
  $
5190 FOR Z = 1 TO 500: NEXT Z
5200 GOTO 5250
5210 INPUT "START RECORDER AND P
  RESS RETURN ";X$
5220 FLASH : PRINT " LOADING ";:
  NORMAL
5230 RECALL L$
5240 GOSUB 5290
5250 PRINT
5260 POKE 34,0: POKE 35,22
5270 GOSUB 3000
5280 RETURN
5290 L$ = L$(0):L = LEN(L$)
5300 CHAR = VAL(RIGHT$(L$,2))
  :L$ = LEFT$(L$,L - 2)
5310 SLOC = VAL(RIGHT$(L$,2))
  :L$ = LEFT$(L$,L - 4)
5320 LN = VAL(L$)
5330 RETURN

```

```

1 REM XXXXXXXXXXXXXXXXXXXXXXXX
2 REM %
3 REM % MICROTEXT 1.0 %
4 REM % ATARI VERSION %
5 REM %
6 REM % BY JON R. VOSKUIL %
7 REM % 10/81 %
8 REM %
9 REM XXXXXXXXXXXXXXXXXXXXXXXX
10 DIM CL$(1):CL%=CHR$(125):PRINT CL$
15 POKE 752,1
20 POSITION 8,8:PRINT "M I C R O T E X
T 1.0"
30 POSITION 12,12:PRINT "BY JON R. VOS
KUIL"
40 PRINT :PRINT "COPYRIGHT 1981 SOFTS
IDE PUBLICATIONS"
50 FOR Z=1 TO 1500:NEXT Z

Initialization.

99 REM INITIALIZATION
100 PRINT CL$
120 DIM L$(40),T$(15000),B$(1),C$(1),C
R$(1),LN$(37),LNXT$(40),F$(14),S$(37)
,FT$(14),LP(500),TT$(40),X$(5)
130 BKSP=126:RTN=155:SPC=32:B%=CHR$(12
6):CR%=CHR$(20)
135 T$="":L$="":LP(0)=0
140 CHAR=1
150 LN=1
160 LN$="-----
-----":S$="
"

170 OPEN #1,A,0,"K:"
180 LWID=36
185 POKE 752,0
190 V=2:H=2
200 POSITION 2,0:PRINT "SAVE:ctrl-S R
EVW:ctrl-R LOAD:ctrl-L";
210 POSITION 2,1:PRINT LN$;
220 POSITION H,V:PRINT " ";B$;

Main input loop.

499 REM INPUT LOOP
500 GET #1,C:C%=CHR$(C)
520 IF C=RTN THEN C%=CR$
639 REM BACKSPACE
640 IF C<>BKSP THEN 720

```

```

650 IF CHAR<2 THEN L$="":GOTO 500
660 PRINT B$;" ";B$;
670 CHAR=CHAR-1
675 IF CHAR<2 THEN 500
680 L%=L$(1,CHAR-1)
700 GOTO 500
719 REM CHK FOR CTRL CHAR
720 IF C<SPC THEN 2000
739 REM CHK FOR END OF LN
740 CHAR=CHAR+1:IF CHAR<LWID OR C=SPC
OR C=RTN THEN 760
750 GOSUB 1000
759 REM ADD TO STRING
760 L$(CHAR-1)=C$
779 REM RETURN
780 IF C<>RTN THEN 880
820 PRINT C$
840 GOSUB 6000:SLOC=0
850 CHAR=1
860 GOTO 500
879 REM PRINT ON SCRIN
880 PRINT C$;
899 REM UPDATE SPC PNTER, CHK FOR END
OF LINE
900 IF C=32 THEN SLOC=CHAR-1:IF CHAR=L
WID THEN GOSUB 6000:CHAR=1:SLOC=0:PRIN
T
920 GOTO 500

Subroutine to break line at a space
and to initialize the next line.

1000 IF SLOC=0 THEN PRINT :GOTO 1100
1020 SS=LWID-SLOC-1:FOR J=1 TO SS:PRIN
T B$;:NEXT J
1040 FOR J=1 TO SS:PRINT " ";:NEXT J:P
RINT
1050 IF SLOC=LEN(L$) THEN LNXT$="":GOT
O 1100
1060 LNXT%=L$(SLOC+1)
1080 L%=L$(1,SLOC)
1100 GOSUB 6000
1110 L%=LNXT$
1115 LNXT$=""
1120 PRINT L$;
1140 CHAR=LEN(L$)+2
1150 SLOC=0
1160 RETURN

```

```

Subroutine to process command
codes.

1999 REM PROCESS CTRL CHAR
2000 V=PEEK(84):H=PEEK(85)
2050 TL=LEN(T$)
2060 T$(TL+1)=L$
2070 POSITION 2,0:PRINT S$:PRINT LN$;
2100 IF C=18 THEN GOSUB 3000:REM CTL-R
2200 IF C=19 THEN GOSUB 4000:REM CTL-S
2300 IF C=12 THEN GOSUB 5000:REM CTL-L
2400 IF C=17 THEN STOP :REM CTL-Q
2900 IF TL>0 THEN T%=T$(1,TL)
2950 GOTO 200

Subroutine to review entered text.

2999 REM REVIEW TEXT
3000 PRINT CL$;"Press any key to pause
":PRINT LN$
3040 IF LN=1 THEN 3210
3050 FOR I=1 TO LN-1
3055 FOR Z=1 TO 20:NEXT Z
3060 PRINT T$(LP(I-1)+1,LP(I))
3070 IF PEEK(764)=255 AND NOT STP THE
N 3200
3080 STP=0:POKE 764,255
3090 VV=PEEK(84)
3100 POSITION 2,1:PRINT LN$;:POSITION
2,0:PRINT "RTN:Cont SPC:Stp X:Delet
e to end ";
3120 X=PEEK(764):POKE 764,255
3130 IF X<>22 THEN 3160
3140 POSITION 2,0:PRINT "Delete from h
ere to the end of text?";:GET #1,X:IF
X<>89 THEN 3100
3150 LN=LN+1:L$="":CHAR=1:SLOC=0:T%=T$(
1,LP(LN-1)):TL=LEN(T$):GOTO 3000
3160 IF X=12 THEN 3190
3170 IF X<>33 THEN 3120
3180 STP=1
3190 POSITION 2,VV:POKE 764,255
3200 NEXT I
3210 X=LP(LN-1)+1:IF X<=LEN(T$) THEN P
RINT T$(X);
3220 H=PEEK(85):V=PEEK(84)
3230 RETURN

Subroutine to save text to disk or
tape.

3999 REM SAVE TO DISK/TAPE
4000 POSITION 2,0:PRINT S$;:POSITION 2
,0:PRINT "Save to Tape or Disk? (T/D/E
SC) ";
4020 GET #1,X
4030 IF X=27 THEN 4400
4060 IF X=84 THEN 4200
4070 IF X<>68 THEN 4000
4080 POSITION 2,0:PRINT S$;:POSITION 2
,0:PRINT "File Name: ";:INPUT F$
4085 IF F$(1,2)<>"D:" THEN FT$="D":FT
$(3)=F$:F$=FT$

```

Continental Adventures

4975 Brookdale Dept. 03
Bloomfield Hills, Mich. 48013
(313) 645-2140

Continental Adventures presents three adventures and one graphics game for the Atari 400 and 800 computer owner

The Ghost Tower — Combat with diabolical demons, 16K \$16.95

Town of Derango — Avenging the death of a father, 8K \$16.95

Talisman of Power — A search for the four keys of Gremlock, 16K.... \$18.95

Super Shape Builder — A graphics game for creating your own pictures. Joysticks reqd. 8K \$14.95

```

4090 POSITION 2,0:PRINT "Insert disk a
nd press RETURN";:GET #1,X
4100 OPEN #2,8,0,F#:GOTO 4210
4200 POSITION 2,0:PRINT "Start tape re
corder and press RETURN";:GET #1,X
4205 OPEN #2,8,0,"C:"
4210 PRINT #2;LN:PRINT #2;SLOC:PRINT #
2;CHAR
4220 FOR I=1 TO LN-1
4230 PRINT #2;T$(LP(I-1)+1,LP(I))
4240 NEXT I
4250 IF CHAR>1 THEN PRINT #2;T$(LP(LN-
1)+1)
4300 CLOSE #2
4400 RETURN

```

Subroutine to load text from disk or tape.

```

4999 REM LOAD FROM DISK/TAPE
5000 POSITION 2,0:PRINT S$;:POSITION 2
,0:PRINT "Load from Tape or Disk? (T/D
/ESC) ";
5020 GET #1,X
5030 IF X=27 THEN 5400
5060 IF X=84 THEN 5200
5070 IF X<>68 THEN 5000
5080 POSITION 2,0:PRINT S$;:POSITION 2
,0:PRINT "File Name: ";:INPUT F$
5085 IF F$(1,2)<>"D:" THEN FT$="D":FT
$(3)=F$:F$=FT$
5090 POSITION 2,0:PRINT "Insert disk a
nd press RETURN";:GET #1,X
5100 OPEN #2,4,0,F#:GOTO 5210
5200 POSITION 2,0:PRINT "Start tape re
corder and press RETURN";:GET #1,X
5205 OPEN #2,4,0,"C:"
5210 INPUT #2;LN:INPUT #2;SLOC:INPUT #
2;CHAR
5215 T$=""
5220 FOR I=1 TO LN-1
5230 INPUT #2,TT$:T$(LEN(T$)+1)=TT$
5240 LP(I)=LEN(T$)
5250 NEXT I
5255 TL=LEN(T$):L$=""
5260 IF CHAR>1 THEN INPUT #2,TT$:T$(LE
N(T$)+1)=TT$:LP(LN)=LEN(T$):L$=TT$
5300 CLOSE #2
5350 GOSUB 3000
5400 RETURN

```

Subroutine to add finished line to the text string.

```

5999 REM ADD LN TO TEXT STRING
6000 T$(LEN(T$)+1)=L$
6080 LP(LN)=LEN(T$)
6100 LN=LN+1:LP(LN)=LEN(T$)
6150 L$=""
6200 RETURN

```



GALACTIC CHASE™

The aliens have swept undefeated across the galaxy. You are an enterprising star ship captain—the final defender of space.

As the aliens attack, you launch a deadly barrage of missiles. Flankers swoop down on your position. Maneuvering to avoid the counterattack, you disintegrate their ships with your magnetic repellers. As your skill improves, the attackers increase their speed. And as a last resort, the aliens use their invisible ray to slow the speed of your missile launcher.

GALACTIC CHASE provides Atari owners with the most challenging one or two person game in the galaxy.



Atari 400/800 16k. Written in machine language. Requires joysticks. Payment: Personal Checks—allow three weeks to clear.

American Express, Visa, & Master Charge—include all numbers on card. Please include phone number with all orders. 24.95 for cassette or 29.95 for disk plus 2.00 shipping. Michigan residents add 4%.

Check the dealer in your local galaxy. Dealer inquiries encouraged. Galactic Chase © 1981 Stedek Software.

SPECTRUM
COMPUTERS

Dept S.
26618 Southfield
Lathrup Village, MI. 48076
(313) 559-5252

```

1 ' XXXXXXXXXXXXXXXXXXXXXXXX
2 ' % %
3 ' % MICROTEXT 1.0 %
4 ' % S-80 VERSION %
5 ' % %
6 ' % BY JON R. VOSKUIL %
7 ' % 10/81 %
8 ' % %
9 ' XXXXXXXXXXXXXXXXXXXXXXXX
10 CLS: PRINT CHR$(23)
20 PRINT @198, "MICROTEXT 1.0"
30 PRINT @462, "BY JON R. VOSKUIL"
40 PRINT @640, "(C) 1981 SOFTSIDE PUBLICATIONS"
50 FOR Z=1 TO 1000: NEXT Z

Initialization.

99 ' INITIALIZATION
100 CLS
110 CLEAR 30000
115 DEFINT A-Z
120 DIM L$(500)
130 BKSP=B: RTN=13: SPC=32: CLR=31: B%=CHR$(8): CR%=CHR$(140)
140 CHAR=1
145 CU%=CHR$(95)
150 LN=1
180 LWID=62
190 P=128
200 PRINT @0, "SAVE: CLEAR-S REVIEW: CLEAR-R
LOAD: CLEAR-L"
210 PRINT@ 64, STRING$(63, "-");
220 PRINT @P, "";

Main input loop.

499 ' INPUT LOOP
500 PRINT CU%;
505 C%=INKEY%; IF C%="" THEN 505 ELSE C=ASC(C%)
520 IF C=RTN THEN C%=CR%
639 ' BACKSPACE
640 IF C=BKSP THEN IF CHAR<2 THEN 505 ELSE PRINT B%;B%; CHAR=CH
AR-1: L$(LN)=LEFT$(L$(LN), LEN(L$(LN))-1): GOTO 500
719 ' CTRL CHARACTER
720 IF C=CLR THEN 2000
739 ' END OF LINE
740 CHAR=CHAR+1: IF CHAR=LWID AND C<>SPC AND C<>RTN THEN GOSUB 1
000
759 ' ADD TO STRING
760 L$(LN)=L$(LN)+C%
779 ' RETURN
780 IF C=RTN THEN PRINT B%;C%; LN=LN+1: L$(LN)="" : SL=0: CHAR=1:
GOTO 500
879 ' PRINT ON SCREEN
880 PRINT B%;C%;
899 ' UPDATE SPC POINTER, CHK END OF LINE
900 IF C<>32 THEN 500 ELSE SL=CHAR: IF CHAR=LWID THEN LN=LN+1: L
$(LN)="" : CHAR=1: SL=0: PRINT
920 GOTO 500

Subroutine to break line at a space and to initialize
the next line.

999 ' JUSTIFY AND INCR LINE
1000 IF SL=0 THEN PRINT: GOTO 1100
1020 SS=LWID-SL: FOR J=1 TO SS: PRINT B%; NEXT
1040 PRINT STRING$(SS,32);:PRINT
1050 IF SL=LWID-1 THEN 1100
1060 L$(LN+1)= RIGHT$(L$(LN), LWID-1-SL)

```

```

1080 L$(LN)= LEFT$(L$(LN), SL-1)
1100 LN=LN+1
1120 PRINT L$(LN);CU%;
1140 CHAR=LEN(L$(LN))+2
1150 SL=0
1160 RETURN

```

Subroutine to process command codes.

```

1999 ' PROCESS CTRL CHARACTERS
2000 P=PEEK(16416) + PEEK(16417)*256 - 15360: 'CURSOR PSN
2020 C%=INKEY%; IF C%="" THEN 2020
2030 C=ASC(C%)
2100 IF C=B2 THEN GOSUB 3000: GOTO 200: 'REVW
2200 IF C=B3 THEN GOSUB 4000: GOTO 200: 'SAVE
2300 IF C=76 THEN GOSUB 5000: GOTO 200: 'LOAD
2400 IF C=B1 THEN STOP: 'QUIT
2800 P=P-1: GOTO 200

```

Subroutine to review entered text.

```

2999 ' REVIEW TEXT
3000 CLS: PRINT@ 21, "PRESS ANY KEY TO PAUSE": PRINT STRING$(63,
"-")
3040 IF LN=1 THEN 3210
3050 FOR I=1 TO LN-1
3055 FOR Z=1 TO 50: NEXT Z
3060 PRINT L$(I)
3070 IF INKEY%="" AND NOT STP THEN 3200
3080 STP=0
3090 PP=PEEK(16416) + PEEK(16417)*256 - 15360
3110 PRINT@ 0, "ENTER: CONTINUE SPACEBAR: STEP 1 LINE X:
DELETE TO END "; STRING$(63, "-");
3120 X%=INKEY%; IF X%="" THEN 3120
3130 X=ASC(X%): IF X>88 THEN 3160
3140 PRINT@ 0, "DO YOU WANT TO DELETE FROM HERE TO THE END OF TH
E TEXT? (Y/N) ";
3145 X%=INKEY%; IF X%="" THEN 3145
3148 IF X%<"Y" THEN 3110
3150 LN=LN+1: L$(LN)="" : CHAR=1: SL=0: GOTO 3000
3160 IF X=13 THEN 3190
3170 IF X<>32 THEN 3110
3180 STP=-1
3190 PRINT @PP, "";
3200 NEXT I
3210 PRINT L$(LN);
3220 P=PEEK(16416) + PEEK(16417)*256 - 15360
3230 RETURN

```

Subroutine to save text to disk or tape.

```

3999 ' SAVE TO DISK/TAPE
4000 PRINT@64, STRING$(63, "-");: PRINT@ 0, STRING$(63,32);: PRINT@
0, "SAVE TO TAPE OR DISK? (T/D/CLEAR) ";
4020 X%=INKEY%; IF X%="" THEN 4020
4040 IF X%=CHR$(CLR) THEN 4220
4050 L$(0)=STR$(LN) + STR$(10000+SL*100+CHAR)
4055 PRINT@ 0, STRING$(35,32);
4060 IF X%="T" THEN 4190
4070 IF X%<"D" THEN 4000
4080 PRINT@ 0, "": LINEINPUT "FILE NAME: ";F%
4090 PRINT@ 0, "INSERT DISK AND PRESS ENTER.": PRINT STRING$(63,
"-");
4095 IF INKEY%<>CHR$(13) THEN 4095
4100 OPEN "O", F%
4140 FOR I=0 TO LN

```

```

4150 PRINT#1, L$(I)
4160 NEXT I
4170 CLOSE
4180 GOTO 4220
4190 PRINT# 0, "START RECORDER AND PRESS ENTER.";
4195 IF INKEY$(<)CHR$(13) THEN 4195
4200 FOR I=0 TO LN
4205 PRINT#-1, CHR$(34);L$(I);CHR$(34)
4210 NEXT I
4220 P=P-1: RETURN

```

Subroutine to load text from disk or tape.

```

4999 ' LOAD FROM DISK
5000 PRINT#64, STRING$(63,"-"); PRINT# 0, STRING$(63,32); PRIN
T# 0, "LOAD FROM TAPE OR DISK? (T/D/CLEAR) ";
5020 X$=INKEY$; IF X$="" THEN 5020
5040 IF X$=CHR$(CLR) THEN 5250
5045 PRINT# 0, STRING$(40,32);
5050 IF X$="T" THEN 5210
5060 IF X$(">"D" THEN 5000
5070 PRINT# 0, " "; LINEINPUT "FILE NAME: ";F$
5080 PRINT# 0, "INSERT DISK AND PRESS ENTER.": PRINT STRING$(63,"
-");
5085 IF INKEY$(<)CHR$(13) THEN 5085
5090 OPEN "I",1,F$
5110 INPUT#1, L$(0)
5120 GOSUB 5290
5130 FOR I=1 TO LN
5140 LINEINPUT#1, L$(I)
5170 NEXT I
5180 CLOSE
5200 GOTO 5250
5210 PRINT# 0, "START RECORDER AND PRESS ENTER.";
5215 IF INKEY$(<)CHR$(13) THEN 5215
5220 INPUT#-1, L$(0): GOSUB 5290
5230 FOR I=1 TO LN
5235 INPUT#-1, L$(I)
5240 NEXT I
5250 GOSUB 3000: RETURN
5290 L$=L$(0): L=LEN(L$)
5300 CHAR= VAL(RIGHT$(L$,2)): L$=LEFT$(L$,L-2)
5310 SL= VAL(RIGHT$(L$,2)): L$=LEFT$(L$,L-5)
5320 LN=VAL(L$)
5330 RETURN

```

57

TRS-80® One Liner

```

1 CLS: I=987: F=10: D#=CHR$(191): B=32: FORS=1707E9: PRINT#1, D#; PRINT
#1+I, D#; PRINT#57, S; I=I+RND(3)-2: C=B+15360: IF PEEK(C)=32 AND PEEK(C
+1)=32C=PEEK(14400): PRINT#B, CHR$(153)CHR$(166); IF C=32B=B-1: NEXT
ELSE IF C=64B=B+1: NEXT ELSE F=F-.01: NEXT ELSE PRINT#B, "CRASH!!!";

```

MIKE KENEKES
 BOX 139
 HARRISVILLE, PA, 16038

TRS-80® One Liner

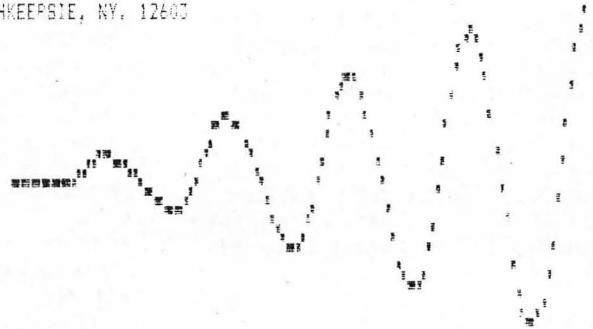
This program allows the user to create sine wave patterns on the screen. Press the → key to decrease the frequency of the wave. Press ← to increase the frequency. Press ↑ to increase the amplitude of the wave, and press ↓ to decrease the amplitude. The CLEAR key will clear the display and continue the pattern on the left side of the screen. All keys will repeat.

```

0 FORX=0TO127: IF A<0, A=0: NEXT: GOTO ELSE IF A>1, A=1: NEXT: GOTO ELSE Y=25
+A*INT(SIN(F*X)*23): SET(X, Y): W=PEEK(14400): IF W=64, F=F-.01: NEXT: G
OTO ELSE IF W=32, F=F+.01: NEXT: GOTO ELSE IF W=8, A=A+.01: NEXT: GOTO ELSE IF
W=16, A=A-.01: NEXT: GOTO ELSE IF W=2, CLS: GOTO ELSE NEXT: GOTO

```

GARY WEXLER
 83 MARTIN DRIVE
 Poughkeepsie, NY, 12600



YOU ARE A FOOL, AMERIKANER, IF YOU THINK YOU CAN ESCAPE FROM CASTLE WOLFENSTEIN! REST WELL TONIGHT! FOR TOMORROW OUR TANKS ROLL FORWARD TO CRUSH THE ALLIES!

CASTLE WOLFENSTEIN™

by Silas Warner

For Apple II or Apple II Plus with 48K,
 \$29.95

MUSE SOFTWARE™

Apple II is a trademark of Apple Computer Corp.

330 N. CHARLES STREET
 BALTIMORE, MD 21201
 (301) 659-7212

Call or write for information and the name of your nearest MUSE dealer

CALENDAR

December 1-3

Legal Info

Shoreham Hotel, Washington, DC

This conference and exposition will cover the automation of legal information systems. Attorneys who use computers, or are interested in doing so, are invited to attend this computer expo geared to their needs.

Contact: Legal Info, 1730 N. Lynn Street, Suite 400, Arlington, VA 22209, (703)521-6209

December 1-4

Computer Network Design and Protocols Washington, DC

This course, presented by ICS (Integrated Computer Systems), will focus on the fundamentals in computer communication network concepts, implementation, and technology. The practical aspects of network design, interfacing, protocols, and packet switching will be emphasized. Contact: Ruth Dordick, ICS, 3304 Pico Blvd., P.O. Box 5339, Santa Monica, CA 90405, (800)421-8166, California residents (800)352-8251

December 2-4

Speech Coding

Jack Tar Hotel, San Francisco, CA

This seminar on various synthesized voice techniques will focus on the basic elements of digital speech processing. Topics will include analog speech signals and their processing; Fourier Transform and the speech time series; analog/digital and digital/analog conversion of speech signals; filtering applied to digital processing of speech signals; speech forms on Fourier Transform, correlation, and autocorrelation; arithmetic requirements for speech processing applications; and the future of voice applications. Registration fee for this seminar, led by William F. Adiletta, is \$650.

Contact: Maria McCabe, McGraw-Hill Conference Center, 1221 Avenue of the Americas, Room 3677, New York, NY 10020, (212)997-4853 or (212)997-4930

December 2-4

Data Processing and Paperwork Reduction

Locations below

Three seminars are planned to be held this month on implementation of Public Law 96-511, the Paperwork Reduction Act of 1980. Each is scheduled to be a one-day meeting to provide a thorough update on the impact of this new piece of legislation for data processing managers and computer specialists. This seminar will give you an opportunity to find out how the new law will affect field organization and computer operations. Dates and cities are:

December 2, Seattle, WA
December 3, San Francisco, CA
December 4, San Diego, CA
Contact: Seminar Coordinator, U.S. Professional Development Institute, Inc., 12611 Davan Drive, Silver Spring, MD 20904, (301)622-0066

December 7-9

National Conference on Computer Graphics: Tools for Productivity Washington, DC

The theme of this conference is centered around exploring the application and implementation of computer graphics within the overall framework of visual information. Three days will be devoted to the latest technological developments, trends and methods of utilizing computer graphics in business, government, engineering, finance, education and other fields. A computer graphics equipment and services demonstration, plus case histories in implementing computer graphics in corporate and government organizations, will conclude the conference.

Contact: Conference Manager, U.S. Professional Development Institute, Inc., 12611 Davan Drive, Silver Spring, MD 20904, (301)622-0066

December 9-11

1981 Winter Simulation Conference (WSC '81)

Peachtree Plaza, Atlanta, GA

This conference is centered around papers, panel discussions, tutorials on discrete and combined simulation and modeling. WSC '81 will be organized into tutorial, methodology, and application sessions.

Contact: Claude M. Delfosse, CACI Inc., 1815 N. Fort Myer Drive, Arlington, VA 22209, (703)841-7800

December 14-15

Software Quality Assurance and Configuration Management

Copley Plaza Hotel, Boston, MA

This new seminar is designed for both the vendor and the users. Topics include defining the roles of Software Quality Assurance (SQA), Software Quality (SQ) within the business structure, SQA and SCM management tools, U.S. Department of Defense SQA requirements, and improving your SQA and SCM performance. Registration fee is \$500. Contact: Maria McCabe, McGraw-Hill Conference Center, 1221 Avenue of the Americas, Room 3677, New York, NY 10020, (212)997-4853 or (212)997-4930

December 16-18

The 20th IEEE Conference on Decision and Control (CDC)

Vacation Village Hotel, San Diego, CA

This annual meeting of the Institute of Electrical and Electronics Engineers (IEEE) Control Systems Society is the basis of CDC. It will include sessions plus tutorials and presentations on all aspects of theory and applications which involve

decision, control, and adaptation. Topics will involve linear and nonlinear system theory, large scale theory system, stability theory, decentralized control, estimation, identification, signal processing, and control systems.

Contact: The Institute of Electrical and Electronics Engineers, Inc., 445 Hoes Lane, Piscataway, NJ 08854

December 25

Merry Christmas

Sponsored by the Editorial Munchkins at *SoftSide* magazine, this meeting is scheduled for the annual arrival of Saint Nicholas, alias Santa Claus. The one-day session is especially welcomed by computer orphans and dreaded by computer widows. Featured will be toys, games (adventure, space and war), what-nots for everyone, and more computer equipment for the person that used to be part of the family and is now part of the computer peripherals. Special requests will be received at the address below, but should not arrive any later than December 20th as the elves must generate the mailing labels on their ELF-80 system, powered by hamsters on a treadmill.

(Unfortunately, the hamsters have been out on strike due to the union's contract negotiations, and updating the mail list is a slow and tedious process.)

Contact: Santa Claus, Christmas Tree Lane, North Pole, Planet Earth, Solar System, Milky Way Galaxy

December 28-30

Computer Modeling of Linguistic Theory Grand Hyatt Hotel, New York, NY

Sponsored by the Association for Computational Linguistics (ACL), three sessions will encompass computer modeling of linguistic theory combined with the annual meeting of the Linguistic Society of America. Readings of contributed papers will cover part of the sessions with the main focus placed on new models for grammars and new strategies for phrasing.

Contact: Stan Petrick, IBM Research Center, P.O. Box 218, Yorktown Heights, NY 10598

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:

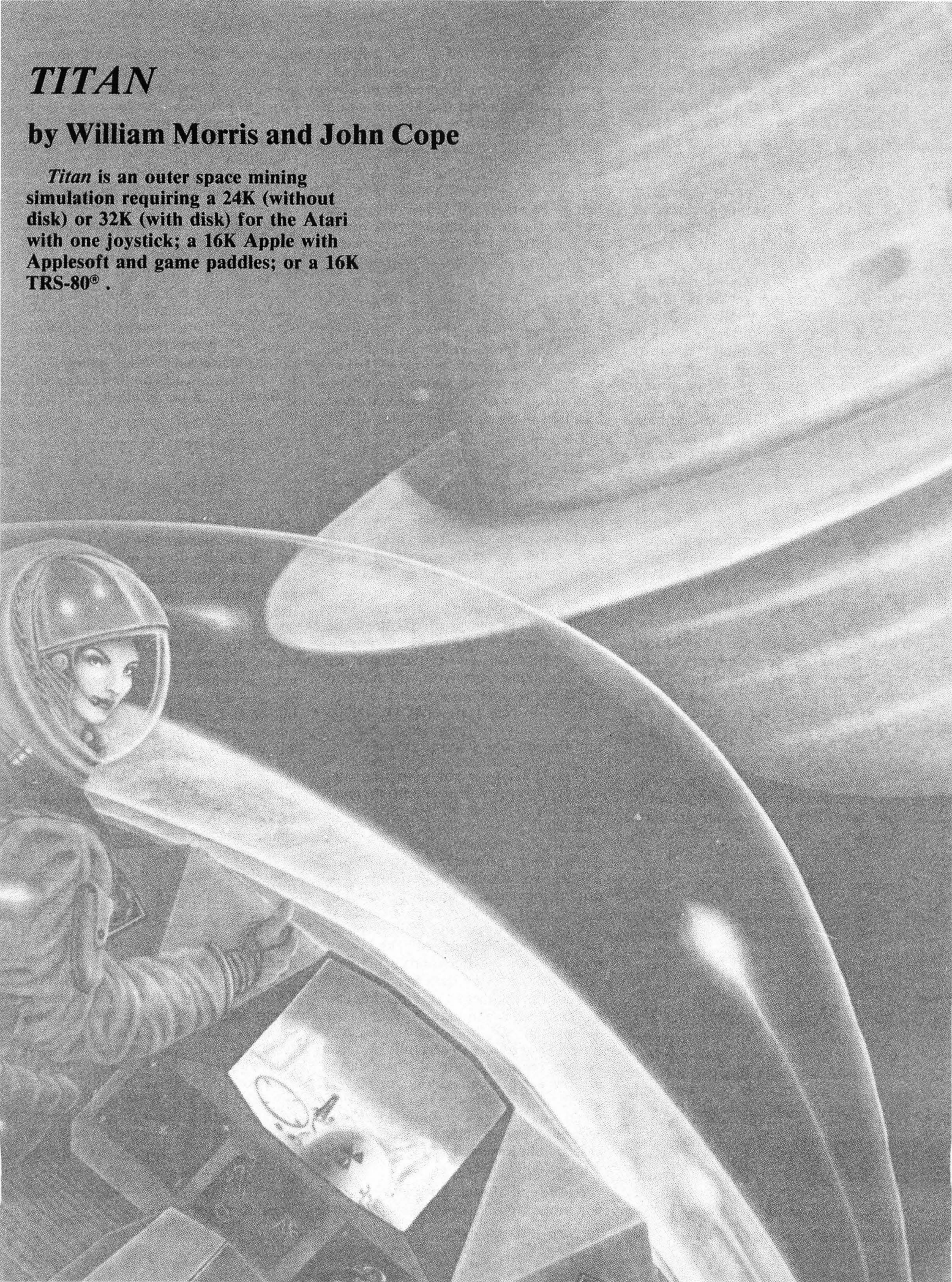
SoftSide Publications
Calendar Editor
6 South Street
Milford, NH 03055

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

TITAN

by William Morris and John Cope

Titan is an outer space mining simulation requiring a 24K (without disk) or 32K (with disk) for the Atari with one joystick; a 16K Apple with Applesoft and game paddles; or a 16K TRS-80®.



Note: The Atari version listed here is divided into two parts, enabling it to run in 24K tape or 32K disk. The first part merges the second with itself. This requires that the second part be LISTed to tape or disk. If using tape, record the second part immediately after the first.

Those who subscribe to Atari DV will also receive an all-in-one version requiring 40K. Apple DV subscribers will receive an additional Hi-Res version requiring 48K.

The year is 2050. The Solar Mining Authority, in its ever-expanding quest for raw material, has finally decided to entertain bids from different corporations for the exclusive mining rights to *Titan*, one of the moons of Saturn.

In an attempt to ensure maximum efficiency from the operators of the Titan concession, the Solar Mining Authority has decided to permit up to four companies to operate on Titan using Probationary License Permits, for a period of one Earth year.

Sometime during the first quarter of the second year, an on-site inspection of the competing companies will result in one of them being awarded exclusive authority to mine the valuable Dilithium 3 crystals peculiar to Titan. The stakes are high; the risk and expense factors, even by 21st century standards, are enormous. The reward, however, makes the gamble more than worthwhile: for your company, fantastic profits; for you, possible promotion to the parent corporation's Board of Directors, as well as immense financial gain! Failure, of course, carries its own reward.

Having decided to accept the post of Superintendent of your corporation's Titan operation, you will be assigned one of four possible base sites: Actaeon, Bellona, Chimera, or Daedalus. As the highest ranking on-site representative of management, you must make the initial decisions concerning budgetary allotment. The only mandatory purchase is one power plant; all of the remaining choices are subject to your final authority. What about drilling rigs and robot miners? Drill rigs are required for vertical mining, while robominers carry out horizontal digging. Will you invest in an on-site Research and Development Station? Investing in this category of equipment can increase the likelihood of finding the Dilithium 3 crystals. How many Meteor Deflection Shields will be installed to protect your company's investment from the ever-present danger of stray asteroids from the rings? Refineries are needed, of course, to process the Dilithium 3 once

you have discovered a vein of this elusive mineral. Without refineries, your profits will suffer greatly. Finally, energy consumption is a key factor on your Titan operation. Having an extra energizer unit or two could be very helpful! Remember: All of these factors are interrelated, with careful timing and meticulous attention to detail being necessary if the subsequent stages of your operation are to prosper. Costs fluctuate with increases or decreases of the availability of equipment.

Once you have made your initial budgetary decisions, you enter the management phase of your undertaking. Your decision concerning the number of surplus laborers (Labor Pool), the length of their work shifts, the nature of the Recreational Facilities, as well as the nature of the Bonus Schedule and Safety Program, must tread the path best suited to achieve your goals. For example, cave-ins are a frequent problem on Titan because of gravity fluctuations caused by Saturn. Investing in safety measures could pay dividends. Your decision to pay your workers well will have positive results that are quickly evident. On the other hand, too much generosity in the area of labor relations could hamper the profit picture. The ultimate goal is to increase your efficiency; however, be advised that there are occasions when you will have to sacrifice efficiency for greater output. This knowledge can only be gained from "hands-on" experience. No one said the job was going to be easy!

Having set your policy for the current work cycle, your attention must now shift to the actual task of locating and mining the precious Dilithium 3 crystals. Once located at their designated mining strikes, your workers will await your decisions as to drilling locations. A unique screen controller will allow you to carefully monitor and direct the drilling phase and subsequent use of the robot miners. Through the careful interpretation of the Assay data beamed back to you, the number of drilling sites can be kept to a minimum. Again, the deployment of the robot miners will reflect your ability to carefully monitor on-site reports. Both of these latter utilities are prodigious users of power; but then, you will have thought of that when you made your initial decisions regarding power plant purchases or investment in Research and Development units. Right?

At the end of each work-cycle period you must decide, once again, on the

budgetary decisions for the following month. Your experience during the previous cycle(s) should enable more efficient judgments as you progress. It is for this reason that you may view a summary of your decisions and accomplishments in the Monthly Report. Your statement will also reflect your financial capabilities for the next turn.

One minor last detail: As you will remember from an earlier part of this briefing, there is a periodic danger of damage to your operation from meteoroids. It will be your task to use the Meteor Deflection Shields to prevent any on-site damage to your corporation's property. Of course, these units require substantial amounts of power; but then, you have carefully planned for this contingency. Haven't you?

Good luck and good mining!

On Playing Titan

One of the primary goals in designing Titan was to make it as independent of the keyboard as possible. For the Apple, this meant the use of the game paddles, and for the Atari, the use of the joystick. As the TRS-80® does not allow for either capability, a joystick-emulator was set up which utilizes the arrow keys for the equivalent left-right, up-down joystick actions, and the ENTER key for the button function. Use of this technique allows both the Atari and TRS-80® versions to play identically in handling input from the player. The Apple version differs in that the use of game paddles required some modification of the input routines and one change to the graphics display. To compensate for this difference, prompting has been included which is activated on the first player's initial turn. The instructions for this have been included in the body of the program.

Some effort has been made to allow a comparison of the different versions of BASIC being used. For example, the TRS-80® version has been set up so the string and array handling is almost identical to that of Atari BASIC. The Apple version has been designed to reflect the more conventional Microsoft procedures in string and array handling. This feature might prove of interest to those who are interested in translations.

Joystick Use — (Please use the appropriate arrow key and the ENTER key for the TRS-80® version.) The joystick routine utilizes the up-down,

left-right registers to permit viewing of possible choices and the button to register a final input. It is used in all of the key sections of the program.

Game Initialization — When indicating the number of players and the level of difficulty, simply use the left-right joystick capability to move the appropriate number and press the button to finalize your input.

Equipment Management and Labor Relations Phases — Toggling the joystick to the left or right indicates your decision to increase or decrease within a category, with the button registering this decision. Upward or downward action on the joystick moves you through the different categories.

Mining Phase — Left or right action on the joystick will allow you to move between the drill rig or robominer options, while upward toggling will permit you to consider ending this phase of the simulation. In each case pressing the joystick button registers your decision. Be aware, however, that the robominers cannot be activated unless you have at least one drill rig on site.

Once you have decided upon a drill rig or a robominer, you may move it to the drill site by using the joystick. Press the button when you have decided that you have the equipment on site. Downward toggling of the joystick permits drilling or the movement of the robominer down the shaft to occur. In the latter case you also have an upward movement capability within the drill shaft and lateral mining ability once the button has been pressed. Be aware that pressing the button during lateral

mining terminates the robominer currently in use.

The TRS-80® version differs only slightly during the mining phase, in that pressing the appropriate arrow key starts movement in that direction. This movement can be suspended by pressing any key, and resumed in the desired direction by pressing the correct key.

Monthly Report — Pressing the joystick button will terminate this section as you move on to decisions for the next month's mining operations.

Variables

A\$: Name of player's base on Titan.
A: Drill Rig array.
AC: Action flag indicating decision to buy or sell.
C: Cost of items purchased during Equipment phase, or the price of worker-related decisions.
C2: Choice flag which is set during the Labor Management phase.
CH: Choice variable.
DA: Month of the year.
EF: Efficiency rating.
EN: Energy consumption.
EQ\$: Equipment.
EX\$: Extras.
FF: Efficiency counter.
HO: Drill hole(s).
K: Player's variable table.
LV: Level of play.
P: Current player.
PL: Total number of players in simulation.

R, RA, RB: Random numbers.
UL, UN, UY, UZ: Sound variables for pitch, duration, etc.
V1: Horizontal location of dilithium.
V2: Vertical location of dilithium.
VE: Number of veins located.
WC\$: Worker related items.
YE: Current year.

All other variables not specifically identified in the range of X-Z are counters or conditional flags.

Variables unique to the Apple version

S\$: Title display for each phase.
SC: Screen color used in Machine Language routine to fill the graphics screen with a specified color.
WB: Color of drill rig or robominer shape.

Variables unique to the Atari version

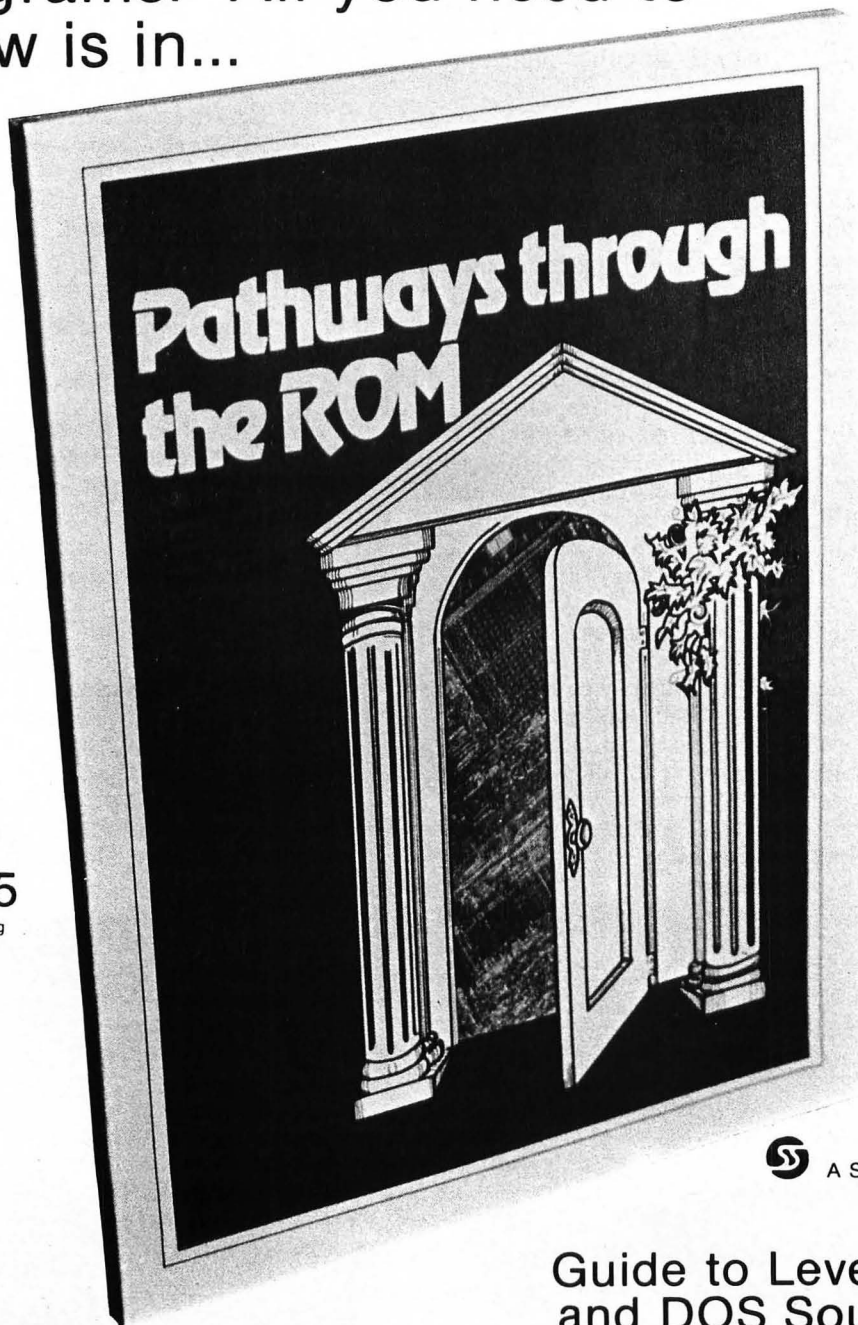
J0-J7: Line numbers.

Variables unique to the TRS-80® version

I\$: Keyboard scan.
JB\$: Packed graphic string of blanks to erase robominer and drill rig.
JC\$: String variable to equate with either JD\$ or JR\$ in graphic display.
JD\$: Packed graphic string of drill rig prior to being rooted on site.
JR\$: Packed graphic string of robominer.
JF\$: Reverse screen Machine Language routine.
JM\$: Sound routine string.



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.



Apple Version

Initialization and title display.

```

10 DIM C(12),K(4,16),EQ$(7),MC$(
   8),EX$(4),A(40,2):Z1 = 18: TEXT
   : GOSUB 30000
20 HOME : INVERSE : FOR X = 1 TO
   5: READ Z$: FOR Y = 22 TO 4 STEP
   - 1: VTAB Y: HTAB Z1: PRINT
   Z$: FOR Z = 1 TO 10: NEXT :
   VTAB Y: HTAB Z1: PRINT CHR$(
   32): NEXT : VTAB Y: HTAB Z1
   : PRINT Z$: FOR X1 = 15 TO 0
   STEP - 0.5: POKE 768,X1: POKE
   769,3: CALL 770: NEXT
30 Z1 = Z1 + 1: NEXT
40 : VTAB 23: HTAB 7: PRINT "(C)
   WM. MORRIS & J. COPE 1981":
   NORMAL : UN = 00: UL = 29: GOSUB
   30080: GOSUB 40010
50 FOR Z = 1 TO 4: READ Z$: A$(Z)
   = Z$: NEXT : FOR Z = 1 TO 7
   : READ Z$: EQ$(Z) = Z$: NEXT
   : FOR Z = 1 TO 7: READ Z$: MC
   $(Z) = Z$: NEXT : FOR Z = 1 TO
   4: READ Z$: EX$(Z) = Z$: NEXT
   : Z$ = "": FOR Z = 1 TO 39: Z$
   = Z$ + CHR$(32): NEXT : GOSUB
   9000

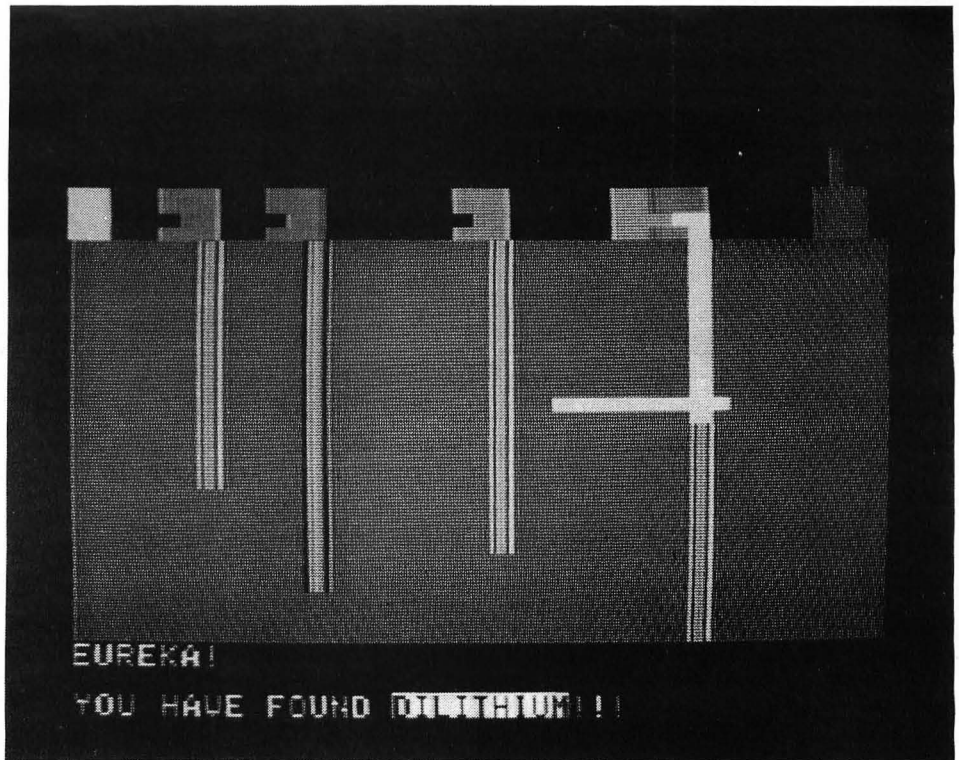
```

Playing parameters.

```

100 HOME : VTAB 2: HTAB 12: FLASH
   : PRINT "WELCOME TO TITAN!":
   NORMAL : GOSUB 30010: GOSUB
   40020: GOSUB 30010: GOSUB 40
   020
110 VTAB 7: HTAB 10: PRINT "USE
   GAME PADDLE ": INVERSE : PRINT
   "#1": NORMAL : PRINT " TO":
   PRINT : HTAB 11: PRINT "SEL
   ECT THE NUMBER OF": PRINT : HTAB
   16: PRINT "PLAYERS": GOSUB 4
   0010: GOSUB 200
120 : PL = X2: VTAB 7: HTAB 10: CALL
   - 958: FOR Z = 1 TO 4: PRINT
   CHR$(7): NEXT : GOSUB 4001
   0: HTAB 10: PRINT "LEVEL OF
   ": INVERSE : PRINT "DIFFICU
   LTY": NORMAL : PRINT "?": GOSUB
   200: LV = X2 + 4
130 HOME : FOR Z = 1 TO PL: VTAB
   Z : 4: INVERSE : PRINT "PLAY

```



```

ER $": NORMAL : PRINT Z$: PRINT
" ": INVERSE : PRINT "COMMA
NDS BASE": NORMAL : PRINT "
": A$(Z): GOSUB 30020: NEXT
140 VTAB 21: HTAB 9: PRINT "PRES
S ": FLASH : PRINT "BUTTON"
: NORMAL : PRINT " TO BEGIN
"
150 WAIT - 16286,128
160 FOR Z = 1 TO PL: K(Z,6) = 1: K
(Z,13) = 500: K(Z,14) = 100: K
(Z,15) = 5: NEXT : GOTO 300
200 X2 = 1: VTAB 20: FOR X1 = 5 TO
35 STEP 10: HTAB X1: GOSUB 3
0020: INVERSE : PRINT X2: GOSUB
40010: GOSUB 220: NORMAL : HTAB
X1: PRINT X2: X2 = X2 + 1: NEXT
210 GOTO 200
220 X = PEEK (- 16286): IF X <
128 THEN RETURN
230 NORMAL : POP : RETURN
300 GOSUB 7000: GOSUB 8000: FOR
P = 1 TO PL: GOSUB 1000: NEXT
: GOSUB 6000: GOTO 300

```

Equipment Management phase.

```

1000 SC = 7: GR : GOSUB 20000: HOME
: VTAB 22: HTAB 9: S$ = "EQUI
PMENT MANAGEMENT": PRINT S$:

```

```

FOR Z = 1 TO 3: GOSUB 30010
: NEXT : TEXT : HOME
1005 IF DA = 1 AND P = 1 AND YE =
2050 THEN GOSUB 1015
1010 GOTO 1030
1015 TEXT : HOME : FOR X1 = 22 TO
1 STEP - 1: VTAB X1: HTAB 9
: PRINT S$: VTAB X1: PRINT Z
$: NEXT : VTAB 1: HTAB 9: PRINT
S$
1020 VTAB 5: PRINT "USE GAME PAD
DLE 1 BUTTON TO ENTER": PRINT
: PRINT "AND LEAVE A SPECIFI
C CATEGORY.": PRINT : PRINT
"GAME PADDLE 1 DIAL REGISTER
S YOUR": PRINT : PRINT "DECI
SION TO INCREASE OR DECREASE
WITHIN": PRINT : PRINT "THA
T CATEGORY."
1025 PRINT : PRINT "GAME PADDLE
0 BUTTON IS USED TO": PRINT
: PRINT "OFFICIALLY RECORD Y
OUR TRANSACTION.": VTAB 22: HTAB
4: INVERSE : PRINT "PRESS BU
TTON 1 TO CONTINUE": NORMAL
: WAIT - 16286,128: PRINT CHR$(
7): RETURN
1030 HOME : FOR X = 1 TO 39: VTAB
1: HTAB X: PRINT "$": VTAB
22: HTAB X: PRINT "$": NEXT
: CH = 1: AC = 0: C2 = 0: ZC = 1
1035 VTAB 2: HTAB 15: INVERSE : PRINT
"BASE": NORMAL : PRINT " "
: A$(P): PRINT DA: "/1/": YE: HTAB

```

```

31: INVERSE : PRINT "STATUS:
"; NORMAL : PRINT " " ; K(P,0
); GOSUB 30030
1040 FOR X = 5 TO 7 STEP 2: INVERSE
: VTAB X: HTAB 1: PRINT Z#: NEXT
: NORMAL : VTAB 6: HTAB 1: PRINT
"EQUIPMENT          OW
MED COST"
1045 Z1 = 9: VTAB 9: HTAB 1: FOR
Z = 1 TO 7: PRINT EQ*(Z): PRINT
: NEXT : GOSUB 1065: VTAB 9:
HTAB 1: FOR Z = 1 TO 7: INVERSE
: PRINT EQ*(Z): GOSUB 1100: IF
CH = 7 THEN Z = CH: GOTO 105
5
1050 GOSUB 40010: GOSUB 1170: GOSUB
1065: NORMAL : VTAB Z1: HTAB
1: PRINT EQ*(Z): PRINT : Z1 =
Z1 + 2
1055 NEXT : IF CH = 7 THEN CH =
1: GOTO 1390
1060 GOTO 1045
1065 VTAB 9: FOR Z2 = 1 TO 6: HTAB
29: PRINT K(P,Z2): HTAB 36:
PRINT C(Z2): PRINT : NEXT :
VTAB 4: HTAB 15: PRINT "CRE
DIT: "; K(P,13): " " : RETURN

```

Subroutine to register paddle decision to increase or decrease as part of buying and selling subroutine which follows.

```

1100 IF CH > 7 THEN CH = CH - 7
1110 NORMAL : VTAB 23: HTAB 1: GOSUB
1130: PRINT "INCREASE": NORMAL
: GOSUB 1150: HTAB 31: PRINT
"DECREASE": NORMAL
1120 RETURN
1130 IF PDL (1) < 125 THEN FLASH
: UN = 60: AC = 0: IF ZC THEN
GOSUB 30050: RETURN
1140 RETURN
1150 IF PDL (1) > 124 THEN FLASH
: AC = 1: UN = 243: IF ZC THEN
GOSUB 30050: RETURN
1160 RETURN
1170 X = PEEK ( - 16286): IF X >
127 THEN CH = Z: PRINT CHR#
(7): GOSUB 40010: GOTO 1190
1180 RETURN
1190 IF CH = 7 THEN RETURN

```

Buy and Sell routine. Accessed by Equipment Management and Labor Relations phases.

```

1200 ZC = 0: GOSUB 1100: GOSUB 40
020

```

```

1210 Y = PEEK ( - 16287): IF Y >
128 THEN GOSUB 40020: GOTO
1240
1220 GOSUB 40010: X = PEEK ( - 1
6286): IF X > 127 THEN ZC =
1: RETURN
1230 GOTO 1200
1240 IF C2 = 1 THEN CH = CH + 6
1250 IF AC = 1 AND CH = 6 AND K(
P,6) = 1 THEN 1350
1260 IF AC = 0 AND CH = 8 AND K(
P,8) > 17 THEN 1350
1270 IF AC = 1 AND K(P,CH) < 1 THEN
1350
1280 IF AC = 0 AND CH = 9 AND K(
P,9) = 0 THEN 1310
1290 IF AC = 0 AND K(P,13) < C(C
H) THEN 1350
1300 IF AC = 1 THEN K(P,CH) = K(
P,CH) - 1: K(P,13) = K(P,13) +
C(CH): IF CH = 7 THEN K(P,14
) = K(P,14) - 1
1310 IF AC = 0 THEN K(P,CH) = K(
P,CH) + 1: K(P,13) = K(P,13) -
C(CH): IF CH = 7 THEN K(P,14
) = K(P,14) + 1
1320 IF C2 = 0 THEN GOSUB 30040
: GOSUB 1065: GOTO 1370
1330 UN = 100: GOSUB 30050: GOSUB
1510
1340 GOTO 1370
1350 IF C2 = 0 THEN GOSUB 30060
: GOSUB 1065: GOTO 1370
1360 GOSUB 30050: GOSUB 1510
1370 IF C2 THEN CH = CH - 6
1380 GOTO 1200

```

Labor Relations phase.

```

1390 S# = "LABOR RELATIONS": SC =
4: BR : GOSUB 20000: HOME : VTAB
21: HTAB 9: PRINT S#: FOR Z =
1 TO 4: GOSUB 30030: NEXT : C
2 = 1: IF DA = 1 AND P = 1 AND
YE = 2050 THEN GOSUB 1015
1400 TEXT : HOME : C2 = 1
1410 INVERSE : HTAB 15: PRINT "B
ASE": NORMAL : PRINT " " ; A#
(P): VTAB 3: INVERSE : PRINT
"MANPOWER": NORMAL : PRINT
" " ; K(P,14): INVERSE : HTAB
29: PRINT "CREDIT": NORMAL
: PRINT K(P,13): GOSUB 30030
1420 K(P,7) = K(P,14) - (K(P,1) &
LV) - (K(P,2) & LV & 2) - (K
(P,3) & LV & 3) - (K(P,4) &
LV & 4) - (K(P,5) & LV & 5) -
(K(P,6) & LV & 10)

```

```

1430 C(9) = K(P,14): C(10) = K(P,1
4) & 2: C(11) = K(P,14) & 3: C
(12) = K(P,14) & 4: EN = K(P,
6) & 500 & (11 - LV)
1440 GOSUB 5000: INVERSE : VTAB
5: HTAB 9: PRINT "EFFICIENCY
RATE": NORMAL : PRINT K(P
,15): INVERSE : PRINT "%"
1450 VTAB 7: PRINT "CONDITIONS
SET COST "
: NORMAL
1460 Z1 = 9: VTAB 9: FOR Z = 1 TO
7: PRINT WC*(Z): PRINT : NEXT
: GOSUB 1510: VTAB 22: INVERSE
: PRINT Z#: VTAB 9: FOR Z =
1 TO 7: INVERSE : PRINT WC*(
Z): PRINT
1470 GOSUB 1100: NORMAL : GOSUB
40010: GOSUB 1170: IF CH = 7
THEN Z = 7: GOTO 1490
1480 GOSUB 40020: VTAB Z1: HTAB
1: PRINT WC*(Z): PRINT : Z1 =
Z1 + 2
1490 NEXT : IF CH = 7 THEN HOME
: GOTO 2000
1500 GOTO 1460
1510 VTAB 9: FOR Z2 = 1 TO 6: HTAB
28: PRINT K(P,Z2 + 6): " " : HTAB
35: PRINT C(Z2 + 6): PRINT :
NEXT : VTAB 3: HTAB 36: PRINT
K(P,13): " " : GOSUB 5000: VTAB
5: HTAB 25: PRINT EF: INVERSE
: PRINT "%": RETURN

```

Mining phase and graphics display:

```

2000 IF K(P,1) = 0 THEN 3000
2005 BR : SC = 12: GOSUB 20000: S#
= "MINING OPERATIONS": HOME
: VTAB 22: HTAB 11: PRINT S#
: FOR Z = 1 TO 5: PRINT CHR#
(7): NEXT : GOSUB 40020: TEXT
: HOME : IF DA > 1 OR P > 1 OR
YE > 2050 THEN 2030
2010 FOR X1 = 22 TO 1 STEP - 1:
VTAB X1: HTAB 11: PRINT S#:
VTAB X1: PRINT Z#: NEXT : VTAB
1: HTAB 11: PRINT S#: PRINT
: PRINT "USE PADDLE 1 TO FOC
US TRACTOR BEAM ON": PRINT "
DRILL RIG OR ROBO MINER. BUTT
ON 1 LOCKS": PRINT "BEAM AND
CONTROLS DRILL RIG MOVEMENT
."
2015 PRINT "ROBO MINERS WILL NOT
ACTIVATE WITHOUT A": PRINT "
DRILL RIG ON SITE.": PRINT :
PRINT "BUTTON 0 LOCKS DRILL

```

```
RIG ON SITE WITH"; PRINT "B
UTTON 1 ACTIVATING THE DRILL
."; PRINT "HORIZONTAL MINING
NEEDS A ROBOMINER."
```

```
2020 PRINT ; PRINT "MORE THAN ON
E DRILL RIG ON SITE "; PRINT
"REQUIRES A CHOICE AS TO ROB
OMINER "; PRINT "LOCATION BY
USING BUTTON 1."; PRINT ; PRINT
"PADDLE 1 DIAL CONTROLS DIRE
CTION FOR"; PRINT "ALL HORIZ
ONTAL MOVEMENT. BUTTON 0 END
S"; PRINT "EACH STAGE OF WOR
K."
```

```
2025 PRINT "TO END ALL MINING US
E BUTTON 1 BEFORE"; PRINT "C
HOOSING EQUIPMENT."; PRINT ;
INVERSE ; PRINT "PRESS BUTT
ON 1 TO CONTINUE"; NORMAL ; WAIT
- 16286,128; PRINT CHR# (7
)
```

Set the graphic display screen and establish the location of the Dilithium 3 crystals.

```
2030 IF K(P,1) > 0 THEN GR : SC =
12; GOSUB 20000; HOME ; COLOR=
0; FOR Z = 0 TO 8; HLINE 0,39
AT Z; NEXT ; V1 = INT ( RND
(1) * 37 + 2); V2 = INT ( RND
(1) * 30 + 9); RA = 9 - LV; RB
= INT ( RND (1) * 2); HO =
0; GOTO 2035
```

```
2035 HOME ; IF K(P,1) = 0 AND (H
O = 0 OR K(P,2) = 0) THEN 30
00
```

```
2040 W = 36; WB = 9; GOSUB 2900; GOSUB
```

```
2045; W = 0; WB = 3; GOSUB 205
0; GOSUB 2900; GOTO 2085
```

Subroutine to display drill rig.

```
2045 COLOR= A(W - 1,1); VLINE 6,8
AT W - 1; COLOR= WB; FOR Z =
W TO W + 3; VLINE 5,8 AT Z; NEXT
; VLINE 2,5 AT Z - 3; COLOR=
A(W + 3,1); VLINE 5,8 AT W +
3; IF W = 34 THEN COLOR= 9;
VLINE 2,5 AT Z - 1
```

Subroutine to display robominer.

```
2050 COLOR= WB; FOR Z = W TO W +
1; VLINE 5,8 AT Z; NEXT ; PLOT
W + 2,7
```

```
2055 RETURN
```

```
2060 GOSUB 2065; VTAB 21; HTAB 3
; PRINT "ROBOMINERS"; NORMAL
; GOSUB 2075; HTAB 28; PRINT
"DRILL RIGS"; NORMAL ; HTAB
7; PRINT K(P,2); HTAB 33; PRINT
K(P,1); GOTO 2085
```

```
2065 IF PDL (1) < 127 THEN FLASH
; W = 1; RETURN
```

```
2070 NORMAL ; RETURN
```

```
2075 IF PDL (1) > 126 THEN FLASH
; W = 37; RETURN
```

```
2080 NORMAL ; RETURN
```

```
2085 ZN = SCRN( W,7); COLOR= 15;
PLOT W,7; GOSUB 40020; COLOR=
ZN; PLOT W,7; GOSUB 40010; XX
= PEEK ( - 16287); IF XX >
128 THEN 3000
```

```
2090 X = PEEK ( - 16286); IF X <
```

```
128 THEN 2060
2095 VTAB 24; HTAB 1; PRINT Z#;
IF ZN = 3 THEN 2160
```

Drill rig movement sequence.

```
2100 W = 36; K(P,1) = K(P,1) - 1; PRINT
CHR# (7); IF K(P,1) < 0 THEN
K(P,1) = K(P,1) + 1; GOSUB 3
0060; GOTO 2085
```

```
2105 XX = 0; HOME
```

```
2110 WB = 6; GOSUB 2045; C2 = 0; GOSUB
2910; IF XX THEN 2125
```

```
2115 WB = 0; GOSUB 2045; W = W + Y
Y; IF W > 36 OR W < 2 THEN W
= W - YY
```

```
2120 WB = 6; GOSUB 2045; GOTO 210
5
```

Routine to fix drill rig on site.

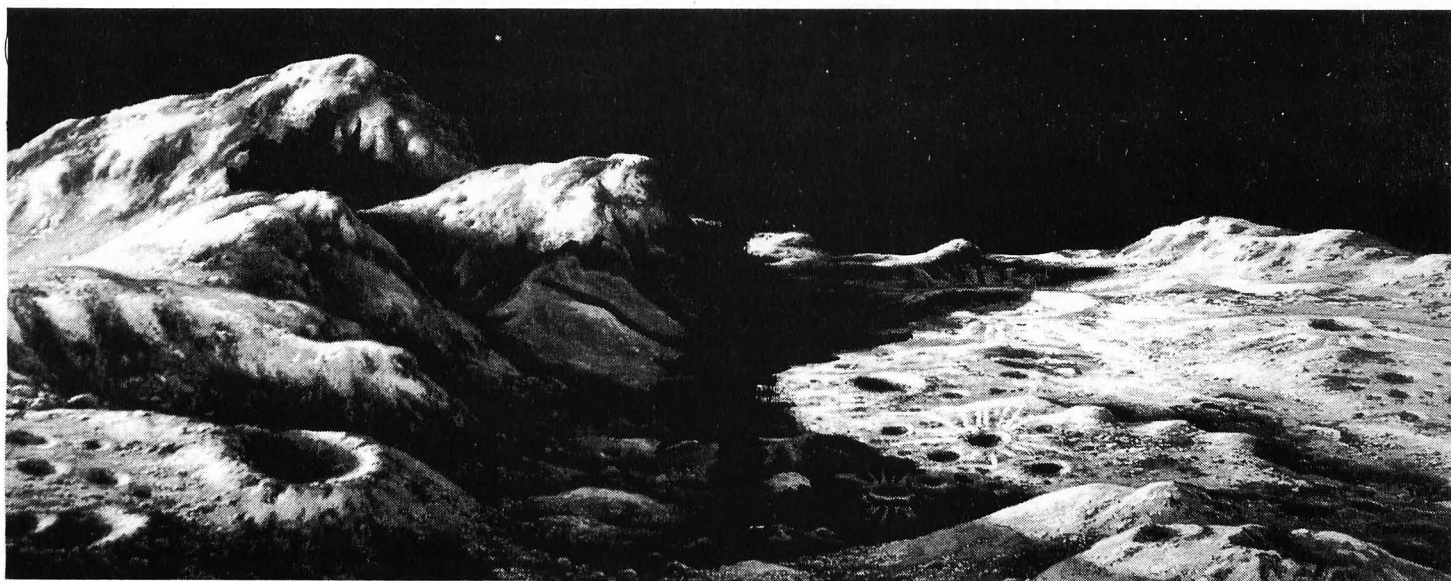
```
2125 WB = 0; GOSUB 2045; HO = HO +
1; Y(HO) = 8; COLOR= 10; FOR
Z = W TO W + 2; GOSUB 30060;
VLINE 5,8 AT Z; NEXT ; B(HO) =
W; GOSUB 2900; W = W + 2
```

Vertical drill routine.

```
2130 VE = 1; GOSUB 2500; VTAB 22;
HTAB 14; PRINT "ENERGY; "; E
N; "; IF EN < 1 OR VE = 1 THEN
2600
```

```
2135 XX = PEEK ( - 16287); IF XX
> 128 THEN 2035
```

```
2140 X = PEEK ( - 16286); IF X <
128 THEN 2130
```



```

2145 COLOR= 2:Y(HO) = Y(HO) + 1;
      IF Y(HO) > 39 THEN Y(HO) =
      39
2150 Y = Y(HO):VE = 4: GOSUB 2500
      :VTAB 22: HTAB 14: PRINT "E
      NERGY: ";EN;" "; IF EN < 1 OR
      VE = 1 THEN 2600
2155 PLOT W,Y: CALL B14: GOTO 21
      35
2160 IF HO = 0 THEN GOSUB 30060
      : GOTO 2060
2165 K(P,2) = K(P,2) - 1: HOME : PRINT
      CHR# (7): IF K(P,2) < 0 THEN
      K(P,2) = K(P,2) + 1: GOSUB 3
      0060: GOTO 2035

```

Place robominer on site if there is only one drill rig.

```

2170 GOSUB 2900: IF HO = 1 THEN
      W = B(HO): COLOR= 0: PLOT W,
      7: GOSUB 30050: GOSUB 40020:
      W = B(HO) - 2:WB = 13: GOSUB
      2050: GOTO 2225

```

Sort routine to organize tab positions of multiple drill rigs situation.

```

2175 FOR Z = 1 TO HO - 1:ZA = 0:
      FOR ZZ = 1 TO HO - Z: IF B(
      ZZ) < B(ZZ + 1) THEN 2185

```

```

2180 Z1 = B(ZZ):B(ZZ) = B(ZZ + 1)
      :B(ZZ + 1) = Z1:ZC = Y(ZZ):Y
      (ZZ) = Y(ZZ + 1):Y(ZZ + 1) =
      ZC:ZA = 1

```

```
2185 NEXT
```

```
2190 IF ZA = 0 THEN Z = HO - 1
```

```

2195 NEXT : FOR ZQ = 1 TO HO: COLOR=
      0:W = B(ZQ): PLOT W,7: GOSUB
      30050: GOSUB 40020: NEXT

```

Move robominer to different drill rigs until choice is made.

```

2200 FOR ZQ = 1 TO HO:WB = 13:W =
      B(ZQ) - 2: GOSUB 2050

```

```

2205 GOSUB 40010:X = PEEK ( - 1
      6286): IF X > 127 THEN Y(HO)
      = Y(ZQ):ZQ = HO:ZY = 1

```

```

2210 IF ZY = 0 THEN WB = 0: GOSUB
      40010: GOSUB 2050

```

```

2215 NEXT : IF ZY THEN ZY = 0: GOTO
      2225

```

```
2220 GOTO 2200
```

```

2225 GOSUB 2900: IF K(P,2) > =
      0 THEN Y = 9:W = W + 4: GOTO
      2235

```

```
2230 GOTO 2170
```

Robominer vertical motion.

```

2235 COLOR= 15: PLOT W - 1,7: CALL
      B14: GOSUB 40020: FOR ZZ = 7
      TO Y: PLOT W,ZZ: CALL B14: GOSUB
      40020: NEXT

```

```

2240 IF PDL (1) > 127 THEN FLASH
      :YY = 1

```

```

2245 VTAB 21: HTAB 1: PRINT "INC
      REASE DESCENT": NORMAL :XX =
      PEEK ( - 16287): IF XX > 12
      7 THEN 2290

```

```

2250 IF PDL (1) < 128 THEN FLASH
      :YY = - 1

```

```

2255 VTAB 21: HTAB 23: PRINT "DE
      CREASE DESCENT": NORMAL

```

```

2260 X = PEEK ( - 16286): IF X <
      128 THEN VE = 1: GOSUB 2500:
      VTAB 22: HTAB 14: PRINT "EN
      ERGY: ";EN;" "; IF EN < 1 OR
      VE = 1 THEN 2600

```

```
2265 IF X < 128 THEN 2240
```

```

2270 COLOR= 15: PLOT W,Y:Y = Y +
      YY: IF Y > Y(HO) OR Y < 9 THEN
      Y = Y - YY

```

```

2275 COLOR= 0: PLOT W,Y: CALL B1
      4

```

```

2280 VE = 4: GOSUB 2500: VTAB 22:
      HTAB 14: PRINT "ENERGY: ";E
      N;" "; IF EN < 1 OR VE = 1 THEN
      2600

```

```
2285 GOTO 2240
```

```

2290 HOME : FOR Z = 1 TO 5: PRINT
      CHR# (7): NEXT

```

Robominer horizontal motion.

```

2295 C2 = 1: GOSUB 2910: COLOR= 1
      5: PLOT W,Y:VE = 4: GOSUB 25
      00: VTAB 22: HTAB 14: PRINT
      "ENERGY: ";EN;" "; IF EN <
      1 OR VE = 1 THEN 2600

```

```

2300 W = W + YY: IF W > 39 OR W <
      1 THEN W = W - YY

```

```

2305 COLOR= 0: PLOT W,Y: CALL B1
      4

```

```

2310 X = PEEK ( - 16287): IF X <
      128 THEN 2295

```

Shut down all active drill rig sites.

```

2315 FOR Z = 1 TO HO:W = B(Z): COLOR=
      0: FOR ZZ = W - 3 TO W - 1: VLIN
      5,B AT ZZ: GOSUB 40020: CALL
      B14: NEXT : COLOR= 6: FOR ZZ
      = W TO W + 2: VLIN 5,B AT Z
      Z: GOSUB 40020: CALL B14: NEXT
      : NEXT : GOTO 2035

```

Checks to compare distance from drill rig bit or robominer to Dilithium.

```

2500 EN = INT (EN - (100 / K(P,1
      5)) * VE)

```

```

2510 VE = 0: INVERSE : VTAB 23: HTAB
      13: IF Y < 9 THEN PRINT "SU
      RFACE      ": NORMAL : RETURN

```

```

2520 IF (W < V1 - RA OR W > V1 +
      RA + 2) OR (Y < V2 - RB * 3 OR
      Y > V2 + K(P,4) + RB * 4) THEN
      PRINT "ICE AND ROCK": NORMAL
      : RETURN

```

```

2530 IF (W < V1 - RA OR W > V1 +
      RA + 2) OR (Y < V2 - RB * 2 OR
      Y > V2 + K(P,4) + RB * 3) THEN
      PRINT "THRIDIUM      ": NORMAL
      : RETURN

```

```

2540 IF (W < V1 - RA OR W > V1 +
      RA + 2) OR (Y < V2 OR Y > V2
      + K(P,4) + RB) THEN PRINT
      "SECONIUM      ": NORMAL : RETURN

```

```

2550 IF (W < V1 OR W > V1 + 2) OR
      (Y < V2 OR Y > V2 + K(P,4) +
      RB) THEN PRINT "FIRIDIUM
      ": NORMAL : RETURN

```

```
2560 VE = 1: RETURN
```

Discovery of Dilithium.

```
2600 IF EN < 1 THEN 2800
```

```

2610 HOME : FLASH : PRINT "EUREK
      A!": NORMAL : PRINT : PRINT
      "YOU HAVE FOUND ": INVERSE
      : PRINT "DILITHIUM": NORMAL
      : PRINT "!!!": FOR Z = 1 TO
      5: PRINT CHR# (7): NEXT : GOSUB
      40000

```

```
2620 HOME
```

```

2630 VTAB 22: NORMAL : GOSUB 265
      0: PRINT "STATUS": NORMAL :
      GOSUB 2670: HTAB 27: PRINT
      "500 CREDITS": NORMAL : X =
      PEEK ( - 16286): IF X < 128 THEN
      2630

```

```

2640 K(P,X1) = K(P,X1) + Y:K(P,16
      ) = K(P,16) + 1: GOTO 2700

```

```

2650 IF PDL (1) < 127 THEN FLASH
      :X1 = 0:Y = 1: GOSUB 30050: RETURN

```

```
2660 NORMAL : RETURN
```

```

2670 IF PDL (1) > 128 THEN FLASH
      :X1 = 13:Y = 500: GOSUB 3005
      0: RETURN

```

```
2680 NORMAL : RETURN
```

End of drilling operations due to lack of equipment or energy.

```

2700 IF EN < 1 THEN 2800
2710 IF K(P,1) > 0 THEN 2030
2730 HOME : PRINT "YOU DO NOT HA
VE ENOUGH EQUIPMENT LEFT"; PRINT
" TO CONTINUE DRILLING.":
GOSUB 30070: GOTO 3000
2800 HOME : PRINT "YOU DO NOT HA
VE ENOUGH ENERGY LEFT TO": PRINT
" CONTINUE DRILLING.": GOSUB
30070: GOTO 3000

Read screen color of lines 6 and 7
into an array to be used by drill rig
graphic display.
2900 FOR ZZ = 1 TO 38:ZA = 1: FOR
ZD = 6 TO 7:A(ZZ,ZA) = SCRNI
(ZZ,ZD):ZA = ZA + 1: NEXT : NEXT
: RETURN
2905 IF C2 THEN VE = 2: GOSUB 25
00: VTAB 22: HTAB 14: PRINT
"ENERGY: ";EN: IF EN < 1 OR
VE = 1 THEN POP :ZM = 0: GOTO
2600

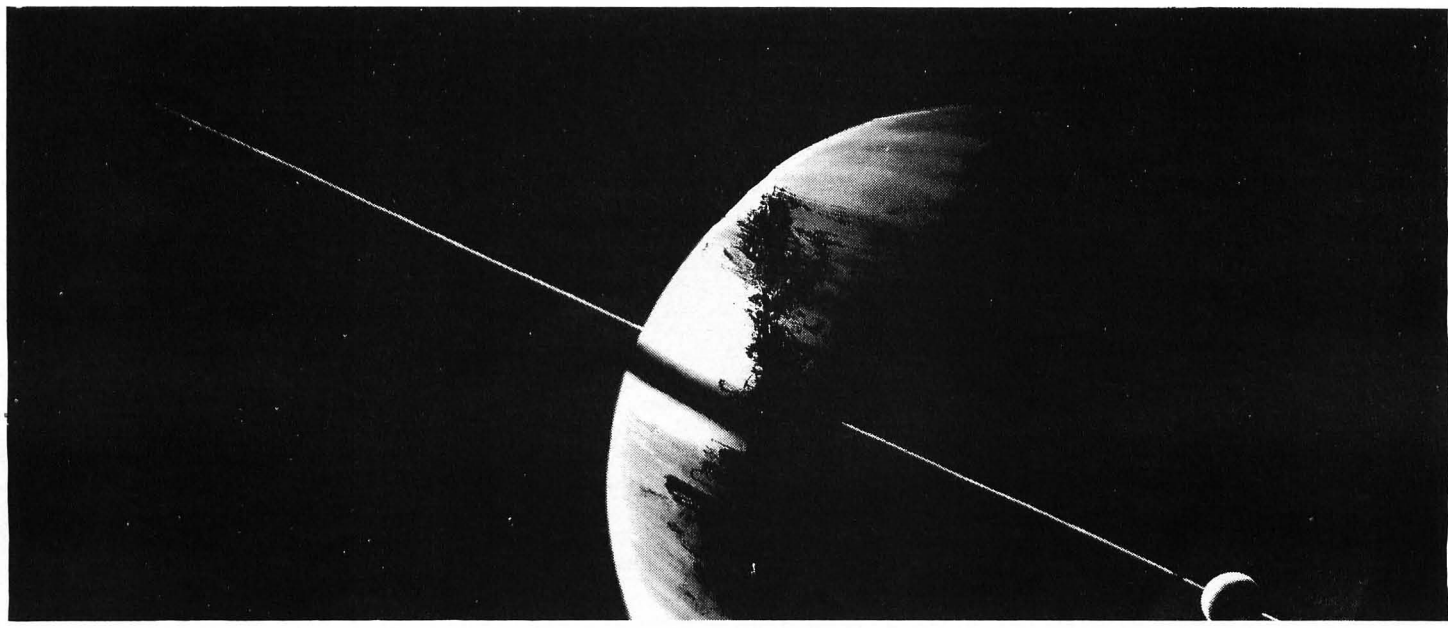
Subroutine for horizontal motion
direction display.
2910 YY = 0: IF PDL (1) < 128 THEN
FLASH :YY = - 1
2920 VTAB 21: HTAB 1: PRINT "<<<<
<<<<<<<<"; NORMAL : HTAB 16:
PRINT "DIRECTION";
2930 IF PDL (1) > 127 THEN FLASH
:YY = 1
2940 HTAB 31: PRINT ">>>>>>>>>"
: NORMAL
2945 X = PEEK ( - 16287): IF X >
127 THEN XX = 1: RETURN
2950 GOSUB 40020:X = PEEK ( - 1
6286): IF X < 128 THEN 2905
2960 RETURN

Cave-in sequence.
3000 R = RND (1) * 10 - K(P,11):
IF R < 7 OR K(P,16) < 1 THEN
3500
3010 GR :SC = 2: GOSUB 20000: HOME
: FLASH : VTAB 21: HTAB 15: PRINT
"RED ALERT!": NORMAL : FOR Y
= 30 TO 0 STEP - 2: POKE 7
68,Y: POKE 769,20: CALL 770:
NEXT : GOSUB 40010
3020 HOME : PRINT : CALL 791: INVERSE
: VTAB 21: HTAB 9: PRINT "CA
VE IN: "; NORMAL :R = INT (
RND (1) * 11)
3030 PRINT R;" WORKERS LOST!";K(
P,14) = K(P,14) - R
3040 IF R > 6 THEN FLASH : PRINT
: HTAB 12: PRINT "SHAFT NOW
CLOSED!";K(P,16) = K(P,16) -
1: NORMAL
3050 HTAB 6: PRINT "PRESS BUTTO
N #"; INVERSE : PRINT "1";:
NORMAL : PRINT " TO CONTINU
E"; WAIT - 16286,128

Meteor storm sequence.
3500 R = RND (1) * 10 - K(P,3): IF
R < 8 THEN 4000
3510 GR :SC = 9: GOSUB 20000: VTAB
21: HTAB 15: PRINT "RED ALER
T!": NORMAL : FOR Y = 200 TO
0 STEP 20: POKE 768,Y: POKE
769,20: CALL 770: NEXT
3520 HOME : PRINT : CALL 791: INVERSE
: VTAB 21: HTAB 4: PRINT "ME
TEOR STORM!";: NORMAL :R =
INT ( RND (1) * 10 + 1)
3530 PRINT R;" WORKERS LOST!";K(
P,14) = K(P,14) - R: IF R <
7 THEN 3550
3540 R = INT ( RND (1) * 5 + 1):
IF K(P,R) < 1 THEN 3550
3550 HTAB 5: PRINT "PRESS BUTTO
N #"; INVERSE : PRINT "1";:
NORMAL : PRINT " TO CONTINU
E"; WAIT - 16286,128: FOR
Z = 1 TO 3: PRINT CHR$( 7):
NEXT

Monthly report.
4000 Z = K(P,5): IF K(P,16) < Z THEN
Z = K(P,16)
4010 K(P,13) = INT (K(P,13) + K(
P,13) * 0.1 + 500 + (50 * K(
P,0)) + (Z * 25 * K(P,8)))
4020 TEXT : HOME
4100 PRINT " ";: VTAB 1: HTAB 17:
FLASH : PRINT "TITAN": NORMAL
4110 PRINT : PRINT "MONTHLY REPO
RT";: HTAB 29: PRINT DA;"/1/
";YE
4120 GOSUB 30080: GOSUB 40010
4130 INVERSE : FOR Z = 4 TO 6 STEP
2: VTAB Z: HTAB 1: PRINT LEFT$(
Z$,39): NEXT : NORMAL : VTAB
5: HTAB 1: PRINT " ITEM
ACTA BELL CHIM DAED
"
4140 VTAB 8: PRINT "STATUS": FOR
Z = 1 TO 6: PRINT EQ$(Z): NEXT
: FOR Z = 1 TO 4: PRINT EX$(

```



```

Z): NEXT
4150 ZA = 0
4160 FOR Z = 15 TO 36 STEP 7: VTAB
8:ZA = ZA + 1: FOR X = 0 TO
6: HTAB Z: PRINT K(ZA,X): NEXT
: FOR X = 13 TO 16: HTAB Z: PRINT
K(ZA,X): NEXT : NEXT
4170 VTAB 23: HTAB 8: PRINT "PRE
SS "; INVERSE : PRINT "BUTT
ON"; NORMAL : PRINT " TO CO
NTINUE"
4180 WAIT - 16286,128: GOSUB 30
060: RETURN

```

Efficiency rating algorithms.

```

5000 FF = 0: FOR Z3 = 1 TO 5: FF =
FF + K(P,Z3) * Z3: NEXT : IF
FF < 1 THEN FF = 1
5010 EF = K(P,6) * 10 / FF: IF EF
> 1 THEN EF = FF / (K(P,6) *
10)
5020 EF = EF * 100: FF = K(P,7) *
2.5: IF FF < 0 THEN FF = FF *
- 2
5030 EF = EF - FF: FF = (K(P,8) -
6) * 3: IF FF < 0 THEN FF =
FF * - 2
5040 EF = EF - FF + (K(P,9) * 5) +
(K(P,10) * 5) + (K(P,11) * 7
.5) + (K(P,12) * 10): EF = INT
(EF): IF EF < 5 THEN EF = 5
5050 IF EF > 100 THEN EF = 100
5060 IF K(P,9) = 0 THEN EF = 1
5070 K(P,15) = EF: RETURN

```

New cost algorithms.

```

6000 FOR Z = 1 TO 8: X = 0: FOR Y

```

```

= 1 TO 4: X = X + K(Y,Z): NEXT
: X = X / PL
6010 IF Z > 5 THEN C(Z) = INT (
C(Z) + (C(Z) * X / (55 - LV)
)): GOTO 6050
6020 C(Z) = INT (C(Z) + (C(Z) *
X / (25 - LV)) - (C(Z) * 1 /
(25 - LV)))
6030 IF C(Z) < 100 * Z THEN C(Z)
= 100 * Z
6040 IF C(Z) > 200 * Z THEN C(Z)
= 200 * Z
6050 IF Z > 6 AND C(Z) < 1 THEN
C(Z) = 1
6060 IF Z > 6 AND C(Z) > 50 THEN
C(Z) = 50
6070 IF Z = 6 AND C(6) > 4000 THEN
C(Z) = 4000
6080 NEXT : FOR Z = 1 TO 4: FOR
Y = 8 TO 12: K(Z,Y) = 0: NEXT
: NEXT : RETURN

```

Date.

```

7000 DA = DA + 1: IF DA = 13 THEN
DA = 1: YE = YE + 1
7010 RETURN

```

End of game sequence.

```

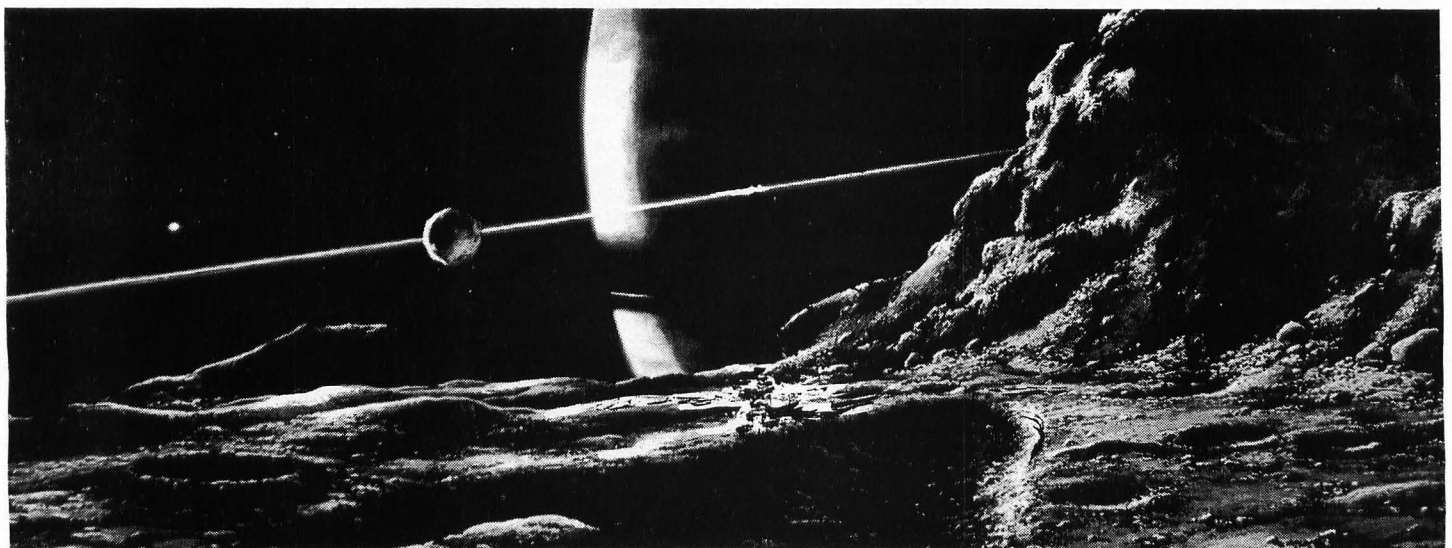
8000 IF YE = 2051 AND DA > 3 THEN
8500
8010 R = INT ( RND (1) * 2): IF
R = 0 OR YE = 2050 THEN RETURN
8020 R = INT ( RND (1) * 2): IF
R = 0 THEN RETURN
8500 FOR X = 1 TO 5: PRINT CHR#

```

```

(7): NEXT : PRINT "THE ";
INVERSE : PRINT "INSPECTORS
"; NORMAL : PRINT " HAVE AR
RIVED!": GOSUB 30010: GOSUB
40000
8510 NORMAL : VTAB 23: PRINT "YO
UR PERFORMANCE IS BEING ASSE
SSED.": GOSUB 30010: GOSUB 4
0000: IF PL = 1 THEN 8600
8520 Z = - 1: Y = 0: FOR X = 1 TO
PL: IF K(X,0) > = Z THEN Z =
K(X,0): Y = Y + 1
8530 NEXT : IF Y = 1 THEN 8590
8540 FOR X = 1 TO Y - 1: IF K(X,
0) = Z THEN ZZ = 1: GOTO 856
0
8550 GOTO 8580
8560 FOR X = 1 TO Y: IF K(X,0) <
0 THEN K(X,0) = 0: NEXT : GOTO
8520
8570 K(X,0) = K(X,16) * 100 + K(X
,15) + K(X,14) + K(X,13) + RND
(1)
8580 NEXT : IF ZZ THEN ZZ = 0: GOTO
8520
8590 VTAB 22: CALL - 958: HTAB
4: PRINT "THE SUPERVISOR OF
"; A$(Y); " HAS WON.": GOTO 86
30
8600 IF K(1,0) > 19 THEN 8620
8610 : VTAB 22: CALL - 958: PRINT
"I AM SORRY TO REPORT THAT Y
OU DID NOT": PRINT "ACHIEVE
ENOUGH STATUS TO GAIN ALL TI
TAN": GOSUB 30070: GOTO 8700
8620 VTAB 22: CALL - 958: PRINT
"BECAUSE OF YOUR STATUS YOU

```




```

HAVE WON"
8630 PRINT " THE RIGHT TO MINE
ALL OF "; INVERSE : PRINT
" TITAN"; NORMAL : PRINT "!"
;: B0SUB 30080
8700 END

Initial values.

9000 DA = 0:YE = 2050: FOR Z = 1 TO
4: FOR Y = 0 TO 16:K(Z,Y) =
0: NEXT : NEXT
9020 C(1) = 100:C(2) = 200:C(3) =
300:C(4) = 400:C(5) = 500:C(
6) = 2000:C(7) = 5:C(8) = 10
:C(9) = 100:C(10) = 200:C(11
) = 300:C(12) = 400
9090 RETURN

Data lines.

10000 DATA 173,48,192,136,208,5,
206,1,3,240,9,202,208,245,17
4,0,3,76,2,3,96
10010 DATA 162,64,169,10,133,0,1
73,48,192,188,0,186,136,208,
253,202,208,244,198,0,208,24
0,96
10020 DATA 169,1,133,0,169,160,1
33,1,164,0,173,48,192,166,1,
202,208,253,136,208,245,198,
1,208,239,96,216,160,0,76,44
,254
10030 DATA T,I,T,A,N
10040 DATA ACTAION,BELLONA,CHIM
ERA,DAEDALUS,DRILL RIG,ROBOM
INERS,DEFLECTOR,R&D UNIT,REF
INERY,ENERGIZER,CONTINUE THE
GAME
10050 DATA LABOR POOL,SHIFT TIME
,WAGE SCALE,RECREATION,SAFET
Y,BONUSES,CONTINUE THE GAME
10060 DATA CREDIT,MANPOWER,EFFIC
IENCY,*VEINS*

Machine Language routine to color
lo-res screen.

20000 POKE 1024,SC + SC * 16: POKE
60,0: POKE 61,4: POKE 62,254
: POKE 63,7: POKE 66,1: POKE
67,4: CALL 840: RETURN

Sound routines.

30000 FOR X = 770 TO 845: READ Z
: POKE X,Z: NEXT : RETURN
30010 : FOR UZ = 15 TO 0 STEP 0.5
: FOR UY = 3 TO 0 STEP - 1:
POKE 768,15 - UY: POKE 769,
UZ: CALL 770: NEXT : NEXT : RETURN
30020 FOR UY = 0 TO 5: POKE 768,
215 - UY: POKE 769,16: CALL
770: NEXT : RETURN
30030 FOR UY = 5 TO 10: POKE 768
,UY: POKE 769,10: CALL 770: NEXT
: RETURN
30040 FOR UY = 50 TO 100 STEP 10
: POKE 768,150 - UY: POKE 76
9,15: CALL 770: NEXT : RETURN
30050 POKE 768,UN: POKE 769,10: CALL
770: RETURN
30060 POKE 768,255: POKE 769,60:
CALL 770: RETURN
30070 FOR UY = 150 TO 200 STEP 1
0: POKE 768,UY: POKE 769,155
: CALL 770: NEXT : RETURN
30080 POKE 801,UN: POKE 802,UL: CALL
791: POKE 801,00: POKE 802,1
86: RETURN

Delay loops.

40000 FOR UZ = 1 TO 3000: NEXT
40010 FOR UZ = 1 TO 500: NEXT
40020 FOR UZ = 1 TO 100: NEXT : RETURN

```

Bound to Please: SoftSide's Deluxe Edition

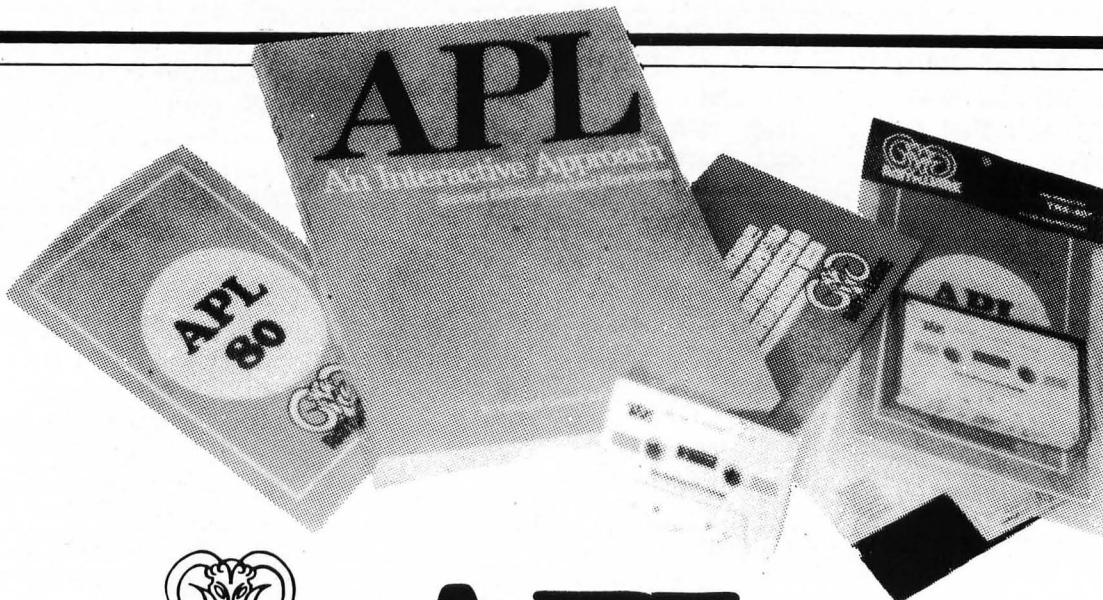
Twelve Issues of the New SoftSide October 1980 to September 1981

SoftSide is pleased to announce this new limited edition: twelve issues sewn into blue buckram and lettered on the spine. This sturdy library bound collection will be available in limited quantities, so order yours now.

The SoftSide Deluxe Edition — \$75.00

Send a check, or use your MasterCharge/VISA credit card. To order write:
SoftSide Publications 6 South Street Milford, NH 03055





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 matrix 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 803-673-0585

Atari Version

Part 1

Initialization and title display.

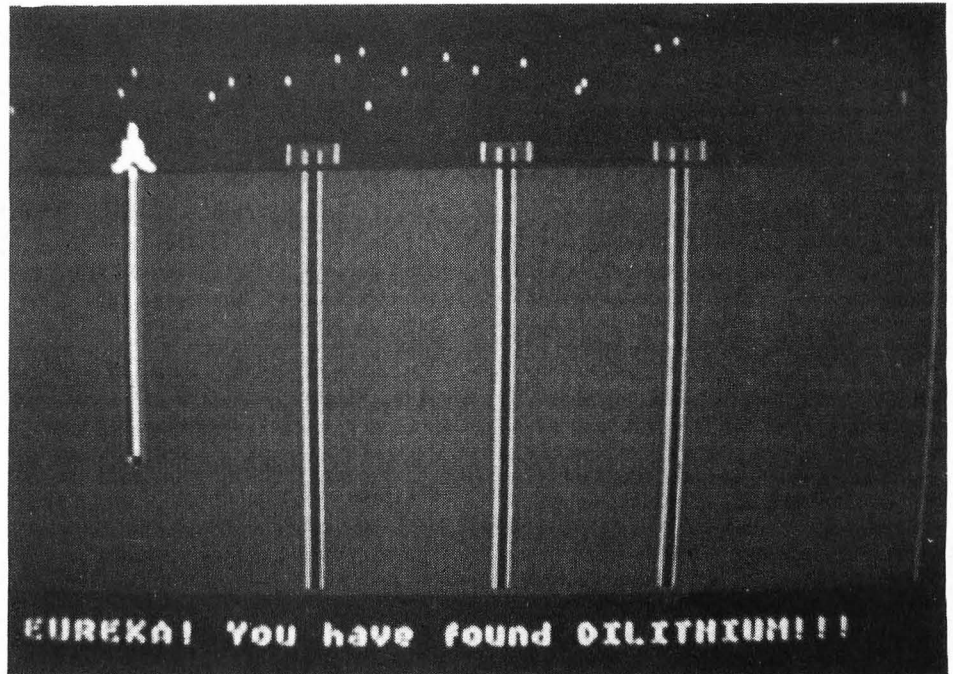
```
1100 DIM C(12),K(4,16),Z$(1),A$(40),EQ
$(70),WC$(70),EX$(40),DB$(72):J0=20000
:J1=20010:J2=20020:J3=20030:J4=20040
1110 J5=20050:J6=20060:J7=20070
1120 GRAPHICS 2:SETCOLOR 0,0,8:SETCOLO
R 1,0,8:SETCOLOR 2,8,0:SETCOLOR 4,8,0
1130 POSITION 7,4:FOR Z=0 TO 3:READ Z$
,Y,X: ? #6;Z$:SOUND Z,Y,10,4:FOR Y=1 T
O X:NEXT Y:NEXT Z: ? #6;"N"
1140 SOUND 3,68,10,4:SOUND 0,136,10,4:
POKE 752,1: ? " (c) Wm. Morris & J. C
ope 1981":FOR Z=1 TO 999:NEXT Z
```

Playing parameters.

```
1150 SOUND 3,0,0,0:FOR Z=0 TO 3:SOUND
0,162,10,8:SETCOLOR 2,3,15:SETCOLOR 4,
3,15:FOR Y=0 TO 200:NEXT Y
1160 SOUND 0,217,10,8:SETCOLOR 2,3,0:S
ETCOLOR 4,3,0:FOR Y=0 TO 200:NEXT Y:NE
XT Z
1170 FOR Z=0 TO 999:NEXT Z:FOR Z=0 TO
3:SOUND Z,0,0,0:NEXT Z:GOSUB 3000
1200 GRAPHICS 17:SETCOLOR 2,4,15:SETCO
LOR 4,12,0:POSITION 2,1: ? #6;"WELCOME
TO TITAN!":POSITION 1,7:GOSUB J0
1210 ? #6;"USE THE JOYSTICK TO": ? #6;"
SELECT THE NUMBER OF":POSITION 6,11: ?
#6;"PLAYERS.":GOSUB 2100
1220 PL=X:GRAPHICS 17:SETCOLOR 2,4,15:
SETCOLOR 4,2,0:POSITION 0,7: ? #6;"LEVE
L OF DIFFICULTY?":GOSUB 2100
1230 LV=X+4:GRAPHICS 17:SETCOLOR 4,15,
0
1240 FOR Z=1 TO PL:POSITION 1,4*Z-3: ?
#6;"PLAYER # ";Z:" COMMANDS BASE";A
$(Z*10-9,Z*10):GOSUB J1:NEXT Z
1250 POSITION 2,19: ? #6;"PRESS TRIGGER
TO":POSITION 8,21: ? #6;"BEGIN":GOSUB
J0
1260 IF STRIG(0) THEN 1260
1270 FOR Z=1 TO PL:K(Z,6)=1:K(Z,13)=50
0:K(Z,14)=100:K(Z,15)=5:NEXT Z: ? "I'M
NOW LOADING PART2."
1280 ? : ? "WHEN I SAY 'READY'," : ? "TYP
E 'GOTO 300' TO CONTINUE . . .":ENTER
"D:TITAN24K.PT2"
1290 REM ENTER"C:" for cassette.
```

Joystick selection routine.

```
2100 POSITION 6,17:Y=2110
2110 ? #6;"1 2 3 4":GOTO 2150
2120 ? #6;"1 2 3 4":GOTO 2150
```



```
2130 ? #6;"1 2 3 4":GOTO 2150
2140 ? #6;"1 2 3 4"
2150 GOSUB J1
2160 POSITION 6,17:FOR Z=0 TO 14 STEP
2:SETCOLOR 2,4,Z:NEXT Z:IF STRIG(0)=0
THEN X=(Y-2100)/10:GOSUB J0:RETURN
2170 IF STICK(0)=15 THEN 2160
2175 IF STICK(0)=11 THEN Y=Y-20:IF Y<2
100 THEN Y=2100
2180 Y=Y+10:IF Y>2140 THEN Y=2110
2190 GOTO Y
```

Set initial numeric variables and string arrays.

```
3000 DA=0:YE=2050:FOR Z=1 TO 4:FOR Y=0
TO 16:K(Z,Y)=0:NEXT Y:NEXT Z
3010 C(1)=100:C(2)=200:C(3)=300:C(4)=4
00:C(5)=500:C(6)=2000:C(7)=5:C(8)=10:C
(9)=100:C(10)=200:C(11)=300:C(12)=400
3020 EQ$="DRILL RIG ROBOMINER DEFLECTO
R R&D UNIT REFINERY ENERGIZER CONTIN
UE "
3030 A$=" ACTAEON BELLONA CHIMERA
DAEDALUS "
3040 WC$="LABOR POOLSHIFT TIMEWAGE SCA
LERECREATIONSAFETY BONUSES CONTIN
UE "
3050 EX$="CREDIT MANPOWER EFFICIEN
CY# VEINS # "
3060 RETURN
```

Sound routines. Common to parts one and two.

```
10000 DATA T,162,400,I,108,400,T,81,60
0,A,64,100
```

```
10010 DATA 8,8,8,8,8,20,28,20,28,20,62
,54,99
10020 DATA 240,208,240,97,255,241,144,
216
20000 FOR UZ=15 TO 0 STEP -0.5:FOR UY=
3 TO 0 STEP -1:SOUND 0,15-UY,10,UZ:NEX
T UY:NEXT UZ:RETURN
20010 FOR UZ=16 TO 0 STEP -2:FOR UY=0
TO 5:SOUND 0,15-UY,10,UZ:NEXT UY:NEXT
UZ:RETURN
20020 FOR UZ=10 TO 0 STEP -2:FOR UY=5
TO 10:SOUND 0,15-UY,10,UZ:NEXT UY:NEXT
UZ:RETURN
20030 FOR UZ=2 TO 0 STEP -2:FOR UY=50
TO 100 STEP 10:SOUND 0,150-UY,10,UZ:NE
XT UY:NEXT UZ:RETURN
20040 SOUND 0,UN,10,2:FOR UZ=1 TO 2:NE
XT UZ:SOUND 0,0,0,0:RETURN
20050 SOUND 0,255,10,4:FOR UZ=1 TO 9:N
EXT UZ:SOUND 0,0,0,0:RETURN
20060 FOR UZ=16 TO 0 STEP -1:FOR UY=15
0 TO 200 STEP 10:SOUND 0,UY,10,UZ:NEXT
UY:NEXT UZ:RETURN
20070 FOR UZ=8 TO 0 STEP -1:FOR UY=55
TO 105 STEP 10:SOUND 0,UY,10,UZ:SOUND
1,UY-50,90,UZ:NEXT UY:NEXT UZ:RETURN
```

Part 2

```
300 GOSUB 7000:GOSUB 8000:FOR P=1 TO P
L:POKE 77,0:GOSUB 1000:NEXT P:GOSUB 60
00:GOTO 300
```

Equipment Management phase.

```
1000 GRAPHICS 1:SETCOLOR 2,8,4:SETCOLD
```

```
R 4,P+4,2:CH=1:AC=0:C2=0
1010 POSITION 3,0:? #6;"BASE:";A*(P*10
-9,P*10):POSITION 1,2:? #6;DA;"/1/";YE
;" STATUS ";K(P,0):GOSUB J2
1020 POSITION 0,4:? #6;"EQUIPMENT OWNE
D COST":POSITION 0,6:FOR Z=1 TO 6:? #6
;EQ*(Z*10-9,Z*10):NEXT Z
1030 FOR Z=1 TO 6:POSITION 12,5+Z:? #6
;K(P,Z);" ":POSITION 16,5+Z:? #6;C(Z):
NEXT Z
1040 POSITION 7,15:? #6;"credit":POSIT
ION 8,17:? #6;K(P,13);" ":GOSUB 1100:
IF CH=7 THEN CH=1:AC=0:GOTO 1300
1050 GOTO 1020
```

Buy and Sell routine. Accessed by Equipment Management and Labor Relations phases.

```
1100 POKE 752,1:POKE 77,0:? "):";IF CH
>7 THEN CH=CH-7
1110 IF AC=0 THEN ? ," INCREASE":UN=
60:GOSUB J4
1120 IF AC THEN ? ," DECREASE":UN=24
3:GOSUB J4
1125 IF C2=0 THEN ? :? ," ";EQ*(CH*1
0-9,CH*10):GOTO 1135
1130 ? :? ," ";WC*(CH*10-9,CH*10)
1135 IF CH=7 THEN ? :UN=121:GOSUB J4
1140 IF STICK(0)=15 AND STRIG(0)<>0 TH
EN 1140
1150 IF STRIG(0)=0 THEN 1200
1160 IF STICK(0)=11 THEN AC=0
1170 IF STICK(0)=7 THEN AC=1
1180 IF STICK(0)=13 THEN CH=CH+1
1190 IF STICK(0)=14 THEN CH=CH-1:IF CH
<1 THEN CH=7
1195 GOTO 1100
1200 IF CH=7 THEN RETURN
1205 IF C2 THEN CH=CH+6
1210 IF (AC=1 AND CH=6 AND K(P,6)=1) D
R (AC=0 AND CH=8 AND K(P,8)>17) OR (AC
=1 AND K(P,CH)<1) THEN 1280
1220 IF AC=0 AND CH=9 AND K(P,9)=0 THE
N 1250
1230 IF AC=0 AND K(P,13)<C(CH) THEN 12
80
1240 IF AC THEN K(P,CH)=K(P,CH)-1:K(P,
13)=K(P,13)+C(CH):IF CH=7 THEN K(P,14)
=K(P,14)-1
1250 IF AC=0 THEN K(P,CH)=K(P,CH)+1:K(P,
13)=K(P,13)-C(CH):IF CH=7 THEN K(P,1
4)=K(P,14)+1
1260 SETCOLOR 2,11,0:IF C2 THEN CH=CH+
1:IF CH=8 THEN SETCOLOR 2,8,4:RETURN
1270 GOSUB J3:SETCOLOR 2,8,4:RETURN
1280 SETCOLOR 2,3,0:GOSUB J5:SETCOLOR
2,8,4:IF C2 THEN CH=CH+1
1290 RETURN
```

Labor Relations phase.

```
1300 GRAPHICS 1:SETCOLOR 2,8,4:SETCOLD
R 4,P+3,2:C2=1
1310 POSITION 3,0:? #6;"BASE:";A*(P*10
-9,P*10):POSITION 4,2:? #6;"MANPOWER "
;K(P,14):POSITION 0,4:GOSUB J2
1320 K(P,7)=K(P,14)-(K(P,1)*LV)-(K(P,2
)*LV*2)-(K(P,3)*LV*3)-(K(P,4)*LV*4)-(K
(P,5)*LV*5)-(K(P,6)*LV*10)
1330 C(9)=K(P,14):C(10)=K(P,14)*2:C(11
)=K(P,14)*3:C(12)=K(P,14)*4:EN=K(P,6)*
500*(11-LV)
1340 GOSUB 5000:? #6;"EFFICIENCY RATE
";K(P,15);"Z"
1350 POSITION 0,6:? #6;"conditions se
t cost":POSITION 0,8:FOR Z=1 TO 6:? #6
;WC*(Z*10-9,Z*10):NEXT Z
1360 FOR Z=1 TO 6:POSITION 12,7+Z:? #6
;K(P,Z+6);" ":POSITION 16,7+Z:? #6;C(Z
+6):NEXT Z
1370 POSITION 7,15:? #6;"credit":POSIT
ION 8,17:? #6;K(P,13);" ":GOSUB 1100
:IF CH=7 THEN 2000
1380 GOSUB 5000:POSITION 16,4:? #6;EF;
"% ";:GOTO 1360
```

Mining phase and graphics display.

```
2000 IF K(P,1)<1 THEN 3000
2010 GRAPHICS 7:SETCOLOR 0,1,6:SETCOLO
R 2,0,0:SETCOLOR 4,0,0:COLOR 3:PLOT 0,
19:DRAWTO 159,19
2020 COLOR 1:FOR Z=20 TO 79:SOUND 0,20
0-Z*2,10,2:PLOT 0,Z:DRAWTO 159,Z:NEXT
Z:SETCOLOR 2,11,2:GOSUB J0
2030 COLOR 1:FOR Z=1 TO 10:FOR Y=1 TO
2:R=INT(RND(0)*159):PLOT R,Z:NEXT Y:NE
XT Z
```

Sets the location of the Dilithium 3 crystals.

```
2040 V1=INT(RND(0)*129+15):V2=INT(RND(
0)*59+20):RA=10-LV:RB=INT(RND(0)*3)
```

This line is central to setting the missile graphics routines for Titan: (1) sets playfield for single-line resolution; (2) reserves memory for missile graphics; (3) sets the high byte of PMBASE; (4) enables missile graphics.

```
2100 POKE 559,62:Z=PEEK(106)-8:POKE 54
279,Z:POKE 53277,3:X=Z*256+1280:RESTOR
E :FOR Z=1 TO 12:READ Z$:NEXT Z:HO=0
```

Reads the data for the robominer and drill rig into the missile graphics screen display locations.

```
2110 FOR Z=57 TO 69:READ Y:POKE X+Z,Y:
NEXT Z:X=X+256:FOR Z=62 TO 69:READ Y:P
OKE X+Z,Y:NEXT Z
```

Sets playfield so that players have priority.

```
2120 SETCOLOR 2,11,2:POKE 623,1:POKE 7
52,1:IF K(P,1)=0 AND (HO=0 OR K(P,2)=0
) THEN 3000
2130 ? CHR*(125):? ,,"DRILL RIG(S) ";:
X=1:W=200:GOTO 2150
2140 ? CHR*(125):? "ROBOMINER(S) ";:X=
2:W=50
2150 ? K(P,X):POKE 53248+X,W:Z=704+X:P
OKE Z,78:GOSUB J0+X*10
2160 FOR Y=78 TO 64 STEP -1:POKE Z,Y:1
F STRIG(0)=0 THEN Y=64:NEXT Y:POKE Z,7
8:GOTO 2200
2165 IF STICK(0)=7 THEN Y=0:NEXT Y:POK
E 53250,0:GOTO 2130
2170 IF STICK(0)=11 THEN Y=0:NEXT Y:PO
KE 53249,0:GOTO 2140
2175 IF STICK(0)=14 THEN 2185
2180 NEXT Y:GOTO 2160
2185 ? CHR*(125):? ," CONTINUE":POK
E 53249,0:POKE 53250,0:GOSUB J3:IF STR
IG(0)=0 THEN 3000
2190 FOR Y=78 TO 64 STEP -1:IF STICK(0
)=15 THEN 2185
2195 GOTO 2165
2200 IF K(P,X)<1 OR (X=2 AND HO=0) THE
N SETCOLOR 2,3,2:GOSUB J5:SETCOLOR 2,1
1,2:GOTO 2160
2210 K(P,X)=K(P,X)-1:FOR Z=20 TO 0 STE
P -1:SOUND 0,Z,10,2:NEXT Z
```

Horizontal movement phase.

```
2220 SOUND 0,W,10,2:IF STRIG(0)=0 THEN
2300
2230 IF STICK(0)=11 THEN W=W-1:IF W<50
AND X=1 THEN W=50
2235 IF W<40 THEN W=40
2240 IF STICK(0)=7 THEN W=W+1:IF W>200
THEN W=200
2250 POKE 53248+X,W:GOTO 2220
2300 IF X=1 AND (W=50 OR W=200) THEN 2
220
2310 IF X=2 THEN 2400
2320 W=W-44:Y=19
```

Vertical drilling sequence.

```
2330 COLOR 2:FOR Z=10 TO 14:SETCOLOR 1
,0,Z:PLOT W,19:VE=1
2333 IF STICK(0)=13 THEN Y=Y+1:SOUND 0
,200,2,8:VE=2:IF Y>78 THEN Y=78
2335 GOSUB 2500:? ,"ENERGY ";EN:IF EN<
1 OR VE=1 THEN Z=14:NEXT Z:GOTO 2600
2340 SOUND 0,150,12,2:DRAWTO W,Y:NEXT
Z:IF STRIG(0)<>0 THEN 2330
2350 COLOR 1:FOR Z=16 TO 18:PLOT W-4,Z
:DRAWTO W+4,Z:NEXT Z:COLOR 0:PLOT W,Y:
```

```
DRAWTO W,17:POKE 53249,0:GOSUB J1
2360 HD=HD+1:GOTO 2120
```

Robominer vertical drilling sequence. Allows up and down movement.

```
2400 LOCATE W-41,19,Z:IF Z<>0 THEN 222
0
2410 W=W-41:Y=17
2420 COLOR 2:FOR Z=10 TO 14:SETCOLOR 1
,0,Z:PLOT W,17:IF STICK(0)=13 THEN Y=Y
+1:LOCATE W,Y,X:IF X<>0 THEN Y=Y-1
2430 IF STICK(0)=14 THEN COLOR 0:PLOT
W,Y:Y=Y-1:IF Y<20 THEN Y=20
2435 VE=1:GOSUB 2500:SOUND 0,150,12,2:
DRAWTO W,Y:?"ENERGY ";EN:IF EN<1 THE
N Z=14:NEXT Z:GOTO 2600
2440 NEXT Z:IF (STICK(0)<>7 AND STICK(
0)<>11) OR Y<20 THEN 2420
2445 COLOR 0:PLOT W,17:DRAWTO W,Y
```

Robominer horizontal drilling sequence.

```
2450 VE=1:COLOR 0:SETCOLOR 1,0,15:PLOT
W,Y:IF STICK(0)=7 THEN W=W+1:SOUND 0,
200,2,8:VE=2:IF W>150 THEN W=W-1
2460 IF STICK(0)=11 THEN W=W-1:SOUND 0
,200,2,8:VE=2:IF W<10 THEN W=W+1
2465 GOSUB 2500:?"ENERGY ";EN:IF EN<
1 OR VE=1 THEN 2600
2470 SOUND 0,150,12,2:COLOR 2:PLOT W,Y
:IF STRIG(0)<>0 THEN 2450
2480 POKE 53250,0:GOSUB J1:GOTO 2120
```

Compare distance of drill rig or robominer to Dilithium.

```
2500 POKE 77,0:EN=INT(EN-(100/K(P,15))
#VE)
2510 VE=0:?"CHR*(125):IF Y<20 THEN SET
```

```
COLOR 2,11,2:?"SURFACE "":RETURN
2520 IF (W<V1-RA*3 OR W>V1+RA*4) OR (Y
<V2-RB*3 OR Y>V2+K(P,4)+RB*4) THEN SET
COLOR 2,1,2:?"ICE AND ROCK"::RETURN
2530 IF (W<V1-RA*2 OR W>V1+RA*3) OR (Y
<V2-RB*2 OR Y>V2+K(P,4)+RB*3) THEN SET
COLOR 2,7,2:?"THIRIDIUM "":RETURN
2540 IF (W<V1-RA OR W>V1+RA*2) OR (Y<V
2-RB OR Y>V2+K(P,4)+RB*2) THEN SETCOLO
R 2,5,2:?"SECONIUM "":RETURN
2550 IF (W<V1 OR W>V1+RA) OR (Y<V2 OR
Y>V2+K(P,4)+RB) THEN SETCOLOR 2,3,2:?"
FIRIDIUM "":RETURN
2560 VE=1:RETURN
```

Discovery of Dilithium.

```
2600 IF EN<1 THEN 2800
2605 COLOR 3:PLOT W,Y:?"CHR*(125):?"E
UREKA! You have found DILITHIUM!!":FO
R Y=1 TO 10:FOR Z=15 TO 0 STEP -1
2610 SETCOLOR 2,3,Z:SOUND 0,160-(Z*10)
,10,Y:NEXT Z:NEXT Y:SOUND 0,0,0,0:GOSU
B J0:SETCOLOR 2,11,2
2620 ?"CHR*(125):? ,,, "STATUS":GOSUB J
1:X=0:Y=1:GOTO 2640
2630 ?"CHR*(125):?"500 CREDITS":GOSUB
J2:X=13:Y=500
2640 IF STRIG(0)=0 THEN K(P,X)=K(P,X)+
Y:K(P,16)=K(P,16)+1:GOTO 2700
2650 IF STICK(0)=7 THEN 2620
2660 IF STICK(0)=11 THEN 2630
2670 GOTO 2640
```

Test for lack of energy or equipment for drilling.

```
2700 IF EN<1 THEN 2800
2710 POKE 53250,0:POKE 53249,0:IF K(P,
1)>0 THEN 2000
```

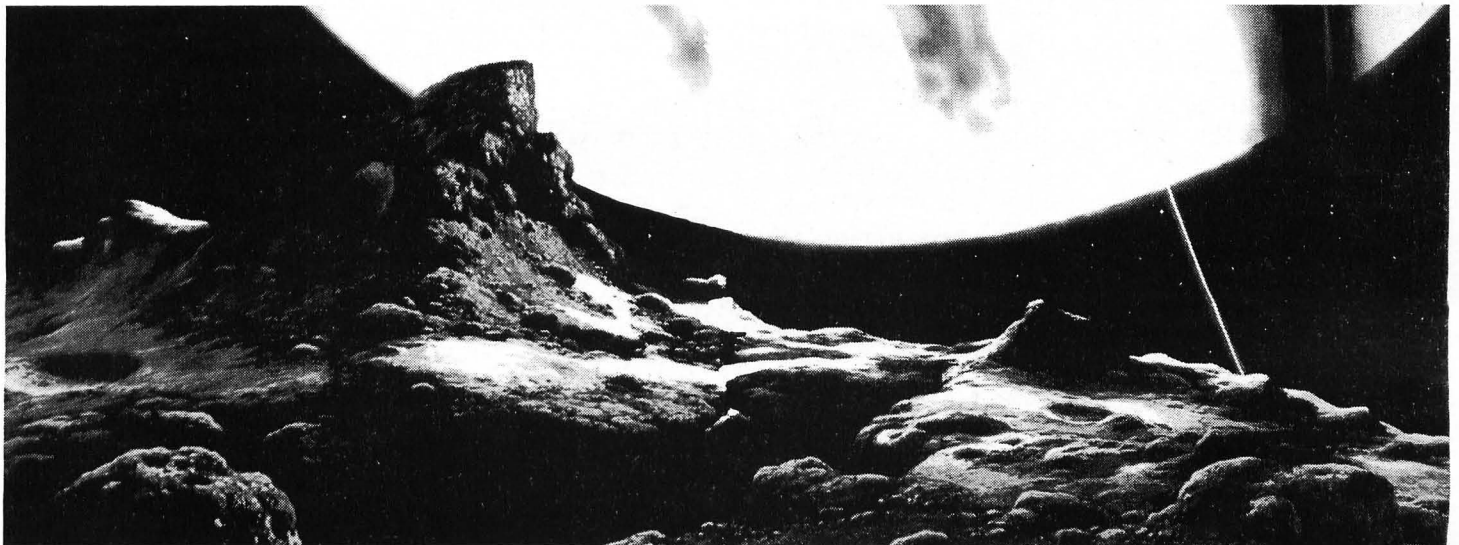
```
2720 ?"CHR*(125):?"YOU DO NOT HAVE EN
OUGH EQUIPMENT LEFT":?"TO CONTINUE DR
ILLING.":GOSUB J6:GOTO 3000
2800 POKE 53250,0:POKE 53249,0
2810 ?"CHR*(125):?"YOU DO NOT HAVE EN
OUGH ENERGY LEFT TO":?" CONTINUE DR
ILLING.":GOSUB J6:GOTO 3000
```

Cave-in sequence.

```
3000 R=RND(0)*10-K(P,11):IF R<7 OR K(P
,16)<1 THEN 3500
3010 GRAPHICS 18:SETCOLOR 4,3,0:POSITI
ON 5,0:?"#6;"RED ALERT!":Z=0
3020 FOR Y=30 TO 0 STEP -1:SOUND 0,150
-Y,10,6:SETCOLOR 0,14,Y:NEXT Y
3025 Z=Z+1:IF Z<5 THEN 3020
3030 SETCOLOR 0,14,10:SOUND 0,0,0,0:PO
SITION 1,3:?"#6;"CAVE IN":R=INT(RND(0)
)*11)
3040 POSITION 3,5:?"#6;R;" WORKERS LOS
T!":K(P,14)=K(P,14)-R
3050 IF R>6 THEN POSITION 3,6:?"#6;"SH
AFT NOW CLOSED!":K(P,16)=K(P,16)-1
3060 IF STRIG(0) THEN 3060
3070 GOTO 4000
```

Meteor storm sequence.

```
3500 R=RND(0)*10-K(P,3):IF R<8 THEN 40
00
3510 GRAPHICS 18:SETCOLOR 4,3,0:POSITI
ON 5,0:?"#6;"RED ALERT!":Z=0
3520 FOR Y=30 TO 0 STEP -1:SOUND 0,150
-Y,10,6:SETCOLOR 0,14,Y:NEXT Y:Z=Z+1:
IF Z<5 THEN 3520
3530 SETCOLOR 0,14,10:SOUND 0,0,0,0:PO
SITION 1,3:?"#6;"METEOR STORM!":R=INT
(RND(0)*10+1)
```



```

3540 POSITION 3,5: ? #6;R;" WORKERS LOS
T!":K(P,14)=K(P,14)-R:IF R<7 THEN 3590
3550 POSITION 0,6:R=INT(RND(0)*5+1):IF
K(P,R)<1 THEN 3590
3560 K(P,R)=K(P,R)-1: ? #6;EQ$(10*R-9,1
0*R);"DESTROYED!"
3590 IF STRIG(0) THEN 3590

```

Monthly report.

```

4000 Z=K(P,5):IF K(P,16)<Z THEN Z=K(P,
16)
4010 K(P,13)=INT(K(P,13)+K(P,13)*0.1+5
00+(50*K(P,0))+(Z*25*K(P,8)))
4020 GRAPHICS 0:SETCOLOR 2,11,0:SETCOL
OR 4,11,0:POKE 752,1:GOTO 4100
4030 FOR Z=1 TO 36: ? Z#;:NEXT Z:RETURN
4099 REM IN LINES 4100 TO 4200, LOWER
CASE IN A PRINT STATEMENT REPRESENTS
THE CORRESPONDING CONTROL CHARACTER.
4100 ? "b";:Z#="m":GOSUB 4030: ? "v";
4105 ? "b";:POSITION 19,1: ? "TITAN":PO
SITION 39,1: ? "vb";:POSITION 39,2: ? "v
";
4110 ? "b";: ? " MONTHLY REPORT":POSITI
ON 29,3: ? DA;"/1/";YE:POSITION 39,3: ?
"vb";:POSITION 39,4: ? "v";
4120 ? "b";:Z#="m":GOSUB 4030: ? "v";:G
OSUB J7
4130 ? "b ITEM      VACTA VBELL VCHIM
VDAED v";
4140 ? "b";:Z#="m":GOSUB 4030: ? "v";
4145 ? "b
      v      v      v
      v      v";
4150 Y=8:FOR Z=0 TO 6:Y=Y+1:POSITION 1
,Y:W=Z*10:IF W=0 THEN ? " b STATUS":GO
TO 4170
4160 ? " b ";EQ$(W-9,W)
4170 FOR X=1 TO 4:POSITION 6*X+9,Y: ? "
v";K(X,Z):POSITION 39,Y: ? "v";:NEXT X:
NEXT Z
4180 FOR Z=1 TO 4:Y=Y+1:POSITION 1,Y: ?
" b ";EX$(Z*10-9,Z*10)
4190 FOR X=1 TO 4:POSITION 6*X+9,Y: ? "
v";K(X,Z+12):POSITION 39,Y: ? "v";:NEXT
X:NEXT Z
4195 ? "b
      v      v      v
      v      v";
4200 ? " ";:FOR Z=1 TO 36: ? "m";:NEXT
Z
4210 POSITION 8,23: ? "Press TRIGGER to
continue.":GOSUB J6
4220 IF STRIG(0)=0 THEN GOSUB J5:RETUR
N
4230 GOTO 4220

```

Efficiency rating algorithms.

```

5000 FF=0:FOR Z=1 TO 5:FF=FF+K(P,Z)*Z:
NEXT Z:IF FF<1 THEN FF=1
5010 EF=K(P,6)*10/FF:IF EF>1 THEN EF=F
F/(K(P,6)*10)
5020 EF=EF*100:FF=K(P,7)*2.5:IF FF<0 T
HEN FF=FF*-2
5030 EF=EF-FF:FF=(K(P,8)-6)*3:IF FF<0
THEN FF=FF*-2
5040 EF=EF-FF+(K(P,9)*5)+(K(P,10)*5)+(
K(P,11)*7.5)+(K(P,12)*10):EF=INT(EF):I
F EF<5 THEN EF=5
5050 IF EF>100 THEN EF=100
5060 IF K(P,9)=0 THEN EF=1
5070 K(P,15)=EF:RETURN

```

New cost algorithms.

```

6000 FOR Z=1 TO 8:X=0:FOR Y=1 TO 4:X=X
+K(Y,Z):NEXT Y:X=X/PL
6010 IF Z>5 THEN C(Z)=INT(C(Z)+(C(Z)*X
/(55-LV))):GOTO 6050
6020 C(Z)=INT(C(Z)+(C(Z)*X/(25-LV)))-(C
(Z)*1/(25-LV))
6030 IF C(Z)<100*Z THEN C(Z)=100*Z
6040 IF C(Z)>200*Z THEN C(Z)=200*Z
6050 IF Z>6 AND C(Z)<1 THEN C(Z)=1
6060 IF Z>6 AND C(Z)>50 THEN C(Z)=50
6070 IF Z=6 AND C(6)>4000 THEN C(Z)=40
00
6080 NEXT Z:FOR Z=1 TO 4:FOR Y=8 TO 12
:K(Z,Y)=0:NEXT Y:NEXT Z:RETURN

```

Adjust date.

```

7000 DA=DA+1:IF DA=13 THEN DA=1:YE=YE+
1
7010 RETURN

```

End of game phase.

```

8000 IF YE=2051 AND DA>3 THEN 8500
8010 R=INT(RND(0)*2):IF R=0 OR YE=2050
THEN RETURN
8500 POSITION 6,23: ? "The INSPECTORS h
ave arrived!":FOR Y=1 TO 10:FOR Z=15
TO 0 STEP -1
8510 SETCOLOR 2,11,Z:SOUND 0,160-(Z*10
),10,Y:NEXT Z:NEXT Y:SOUND 0,0,0,0:GOS
UB J0
8520 POSITION 3,23: ? "Your performance
is being assessed.":FOR Z=1 TO 200:N
EXT Z:IF PL=1 THEN 8600
8530 Z=-1:Y=0:FOR X=1 TO PL:IF K(X,0)
=Z THEN Z=K(X,0):Y=Z
8540 NEXT X:IF Y=1 THEN 8590

```

```

8550 FOR X=1 TO Y-1:IF K(X,0)=Z THEN B
570
8560 NEXT X:GOTO 8590
8570 FOR X=1 TO Y:IF K(X,0)<0 THEN K(X
,0)=0:NEXT X:GOTO 8530
8580 K(X,0)=K(X,16)*100+K(X,15)+K(X,14
)+K(X,13)+RND(1):NEXT X:GOTO 8530
8590 SETCOLOR 4,5,0:SETCOLOR 2,5,0:POS
ITION 4,22: ? "The supervisor of";A$(Y*
10,Y*10+10);"has won":GOTO 8640
8600 IF K(1,0)>19 THEN 8630
8610 SETCOLOR 4,3,0:SETCOLOR 2,3,0:POS
ITION 2,22: ? "I am sorry to report tha
t you did not";
8620 POSITION 0,23: ? "achieve enough s
tatus to gain all TITAN":GOSUB 20060:
GOTO 8700
8630 SETCOLOR 4,8,0:SETCOLOR 2,8,0:POS
ITION 4,22: ? "Because of your status y
ou have won"
8640 ? " the right to mine all of TI
TAN!! ";:GOSUB J7
8700 GOTO 8700

```

Sound routines. Common to parts one and two.

```

10000 DATA T,162,400,1,108,400,T,81,60
0,A,64,100
10010 DATA B,B,B,B,B,20,28,20,28,20,62
,54,99
10020 DATA 240,208,240,97,255,241,144,
216
20000 FOR UZ=15 TO 0 STEP -0.5:FOR UY=
3 TO 0 STEP -1:SOUND 0,15-UY,10,UZ:NEX
T UY:NEXT UZ:RETURN
20010 FOR UZ=16 TO 0 STEP -2:FOR UY=0
TO 5:SOUND 0,15-UY,10,UZ:NEXT UY:NEXT
UZ:RETURN
20020 FOR UZ=10 TO 0 STEP -2:FOR UY=5
TO 10:SOUND 0,15-UY,10,UZ:NEXT UY:NEXT
UZ:RETURN
20030 FOR UZ=2 TO 0 STEP -2:FOR UY=50
TO 100 STEP 10:SOUND 0,150-UY,10,UZ:NE
XT UY:NEXT UZ:RETURN
20040 SOUND 0,UN,10,2:FOR UZ=1 TO 2:NE
XT UZ:SOUND 0,0,0,0:RETURN
20050 SOUND 0,255,10,4:FOR UZ=1 TO 9:N
EXT UZ:SOUND 0,0,0,0:RETURN
20060 FOR UZ=16 TO 0 STEP -1:FOR UY=15
0 TO 200 STEP 10:SOUND 0,UY,10,UZ:NEXT
UY:NEXT UZ:RETURN
20070 FOR UZ=8 TO 0 STEP -1:FOR UY=55
TO 105 STEP 10:SOUND 0,UY,10,UZ:SOUND
1,UY-50,90,UZ:NEXT UY:NEXT UZ:RETURN

```

TRS-80® Version

Initialization

```
10 CLS: CLEAR 300: DEFINT W-Z: DIM C(12), K(4,16): GOSUB 20000: G
OSUB 30000: GOSUB 50000
```

Title display.

```
20 CLS: GOSUB 20010: J=STRING$(10,32)+STRING$(43,128)+STRING$(11,32
): PRINT@192, J; J; J; J; J; J; J; FOR Z=1 TO 5: FOR Y=1 TO 18: READ X: PRINT@X+268,
CHR$(191); : NEXT: READ UN, UL: GOSUB 20100: NEXT
30 PRINT@657, "(c) Wm. Morris & J. Cope 1981"
40 FOR Z=1 TO 4: GOSUB 30010: US=USR(0): UN=162: UL=20: GOSUB 20010: GOSUB 2
0100: GOSUB 30010: US=USR(0): UN=217: UL=18: GOSUB 20010: GOSUB 20100: NEX
T: FOR Z=1 TO 1000: NEXT: GOSUB 9000
```

Playing Parameters.

```
100 GOSUB 20110: CLS: PRINT CHR$(23); : PRINT@144, "Welcome to TITAN!"
110 PRINT@388, "Use the ARROWS to select the          number of pla
yers.": GOSUB 200
120 PL=X: CLS: GOSUB 20110: PRINT CHR$(23); : PRINT@396, "LEVEL of diffic
ulty?": GOSUB 200
130 LV=X+4: CLS: GOSUB 20110: PRINT CHR$(23)
140 FOR Z=1 TO PL: PRINT@-128+192*Z, " Player # "; Z; " commands b
ase          "; MID$(A$, Z*10-9, 10): NEXT
150 PRINT@906, "Press ENTER to begin."
160 GOSUB 40000: IF I(>13) THEN 160
170 FOR Z=1 TO PL: K(Z,6)=1: K(Z,13)=500: K(Z,14)=100: K(Z,15)=5: NEX
T Z: GOTO 300
200 PRINT@848, " "; Y=1
210 PRINT "1#  2  3  4  ": GOTO 250
220 PRINT " 1  #2# 3  4  ": GOTO 250
230 PRINT "  1  2  #3# 4  ": GOTO 250
240 PRINT "   1  2  3  #4#": GOTO 250
```

```
250 PRINT@848, " "; GOSUB 40000: IF I=13 THEN X=Y: RETURN
260 IF I (<) 8 AND I (<) 9 THEN 250
270 IF I=8 THEN Y=Y-2: IF Y<0 THEN Y=0
280 Y=Y+1: IF Y>4 THEN Y=1
290 ON Y GOTO 210, 220, 230, 240
```

Game play sequence.

```
300 GOSUB 7000: GOSUB 8000: FOR P=1 TO PL: GOSUB 1000: NEXT P: GOSUB
6000: GOTO 300
```

Equipment Management phase.

```
1000 CLS: PRINT CHR$(23); : CH=1: AC=0: C2=0
1010 PRINT@16, "BASE: "; MID$(A$, P*10-9, 10): PRINT@128, DA; "/1/"; YE, "
STATUS "; K(P,0)
1020 PRINT@256, "EQUIPMENT      OWNED      COST": PRINT: FOR Z=1 TO
6: PRINT MID$(EQ$, Z*10-9, 10): NEXT Z
1030 FOR Z=1 TO 6: PRINT@348+Z*64, K(P,Z); " "; : PRINT@366+Z*64, C(Z)
; : NEXT Z
1040 PRINT@848, "CREDIT  "; K(P,13); " "; GOSUB 1100: IF CH=7 THEN C
```



```
H=1:AC=0:GOTO 1300
1050 GOTO 1020
```

Buy & Sell routine. Accessed by Equipment Management and Labor Relations phases.

```
1100 IF CH>7 THEN CH=CH-7
1110 IF AC=0 THEN PRINT@960,"INCREASE";:PRINT@1004,"";
1120 IF AC=1 THEN PRINT@1004,"DECREASE";:PRINT@960,"";
1125 IF C2=0 THEN PRINT@982,MID$(EQ$,CH*10-9,10);:GOTO 1140
1130 PRINT@982,MID$(WC$,CH*10-9,10);
1140 GOSUB40000
1150 IF I=13 THEN 1200
1160 IF I=8 THEN AC=0
1170 IF I=9 THEN AC=1
1180 IF I=10 THEN CH=CH+1
1190 IF I=91 THEN CH=CH-1:IF CH<1 THEN CH=7
1195 GOTO 1100
1200 IF CH=7 THEN RETURN
1205 IF C2 THEN CH=CH+6
1210 IF (AC=1 AND CH=6 AND K(P,6)=1) OR (AC=0 AND CH=8 AND K(P,8)
) >17) OR (AC=1 AND K(P,CH)<1) THEN 1280
1220 IF AC=0 AND CH=9 AND K(P,9)=0 THEN 1250
1230 IF AC=0 AND K(P,13)<C(CH) THEN 1280
1240 IF AC THEN K(P,CH)=K(P,CH)-1:K(P,13)=K(P,13)+C(CH):IF CH=7
THEN K(P,14)=K(P,14)-1
1250 IF AC=0 THEN K(P,CH)=K(P,CH)+1:K(P,13)=K(P,13)-C(CH):IF CH=
7 THEN K(P,14)=K(P,14)+1
1260 IF C2 THEN CH=CH+1:IF CH=8 THEN RETURN
1270 GOSUB20120:RETURN
1280 IF C2 THEN CH=CH+1
1290 UN=255:UL=5:GOSUB20100:PRINTCHR$(23);:RETURN
```

Labor Relations phase.

```
1300 CLS:PRINTCHR$(23);:C2=1
1310 PRINT@16,"BASE";:MID$(A$,P*10-9,10):PRINT@128,"MANPOWER";K(
P,14);
1320 K(P,7)=K(P,14)-(K(P,1)*LV)-(K(P,2)*LV*2)-(K(P,3)*LV*3)-(K(P
,4)*LV*4)-(K(P,5)*LV*5)-(K(P,6)*LV*10)
1330 C(9)=K(P,14):C(10)=K(P,14)*2:C(11)=K(P,14)*3:C(12)=K(P,14)*
4:EN=K(P,6)*500*(11-LV)
1340 GOSUB 5000:PRINT@158,"EFFICIENCY";K(P,15);"%";
1350 PRINT@256,"CONDITIONS SET COST":PRINT:FOR Z=1 TO
6:PRINT MID$(WC$,Z*10-9,10):NEXT Z
1360 FOR Z=1 TO 6:PRINT@348+Z*64,K(P,Z+6);" ";:PRINT@366+Z*64,C(
Z+6);:NEXT Z
1370 PRINT@848,"CREDIT ";K(P,13);" ";:GOSUB 1100:IF CH=7 THEN 2
000
1380 GOTO1340
```

Mining phase & graphic display.

```
2000 IF K(P,1)<1 THEN 3000
2010 CLS:PRINT@192,STRING$(64,131):PRINT@832,STRING$(64,176)
2030 FORY=0TO2:FORZ=0TO4:R=INT(RND(127));SET(R,Y):NEXTZ:NEXTY
```

Set the location of Dilithium 3 crystals.

```
2040 V1=INT(RND(109)+9):V2=INT(RND(131)+9):RA=9-LV:RB=INT(RND(2)-
1)
2100 HO=0
2120 IF K(P,1)=0 AND (HO=0 OR K(P,2)=0) THEN 2720
2130 PRINT@960,"";:PRINT@1008,"DRILL RIG(S)";:X=
1:W=124:UN=81:UL=10:JC=JD:GOTO 2150
```

```
2140 PRINT@1008,"";:PRINT@960,"ROBOMINER(S)";:X=2
:W=65:UN=108:JC=JR
2150 PRINT K(P,X);:PRINT@W,JC;:GOSUB20100
2160 GOSUB40000:IFI=13THEN2200
```

Inkey input triggers the appropriate screen display.

```
2165 IF I=9 THEN PRINT@W,JB;:GOTO2130
2170 IF I=8 THEN PRINT@W,JB;:GOTO2140
2175 IF I=91 THEN 2185
2180 GOTO2160
2185 PRINT@960,STRING$(63,32);:PRINT@988,"CONTINUE";:PRINT@W,JB;
:UN=40:GOSUB20100
2190 GOSUB40000:IFI=13THEN3000 ELSE IF I<80R1>9 THEN2185
2195 PRINT@988,"";:GOTO 2165
2200 IF K(P,X)<1 OR (X=2 AND HO=0) THEN GOSUB20140:GOTO2160
2210 K(P,X)=K(P,X)-1:UN=40:GOSUB20100
2220 I$=INKEY$:IFI$=""THEN2230 ELSEI=ASC(I$):IFI=13THEN 2300 ELS
E IFI>7ANDI<10THENPRINT@W,JB;
```

Horizontal movement phase.

```
2230 PRINT@W,JB;:IFI=8THENW=W-1:IFW<66ANDX=1THENW=66
2235 IF W<65 THEN W=65
2240 IFI=9THENW=W+1:IFW>124THENW=124
2250 PRINT@W,JC;:GOTO2220
2300 IF X=1 AND (W<66 OR W>124) THEN 2220
2310 IF X=2 THEN 2400
2320 W=(W-63)*2+1:Y=9
```

Vertical drilling sequence.

```
2330 SET(W,Y):VE=2
2333 IF I=10 THEN Y=Y+1:OUT255,0:OUT255,1:VE=4:IF Y>40THENY=40
2335 GOSUB 2500:PRINT@1008,"ENERGY";EN;:IF EN<1 OR VE=1 THEN 260
0
2340 OUT255,0:OUT255,6:I$=INKEY$:IFI$=""THEN2330ELSEI=ASC(I$):IF
I<>13THEN2330
2360 PRINT@((W-1)/2+63,JB);:HO=HO+1:GOTO 2120
```

Robominer vertical drilling sequence. Allows up & down movement.

```
2400 W1=W:W=(W-63)*2+3:Y=10:IFPOINT(W,Y)=0THENW=(W-3)/2+63:GOTO2
220
2410 Y=B
2420 SET(W,Y):I$=INKEY$:IFI$=""THEN2425 ELSE I=ASC(I$)
2425 IFI=10THENY=Y+1:IFPOINT(W,Y)=0THENY=Y-1
2430 IF I=91 THEN SET(W,Y):Y=Y-1:IF Y<8 THEN Y=8
2435 VE=2:GOSUB 2500:OUT255,0:OUT255,1:PRINT@1008,"ENERGY";EN;:I
F EN<1 THEN 2600
2440 IF(I<>8ANDI<>9) OR Y<8THEN 2420
2445 SET(W,Y):I$=INKEY$:IFI$=""THEN2450 ELSE I=ASC(I$)
2450 VE=2:SET(W,Y):IFI=9 THEN W=W+1:OUT255,0:OUT255,1:VE=4:IF W>
126 THEN W=W-1
2460 IFI=8 THEN W=W-1:OUT255,0:OUT255,1:VE=4:IF W<1 THEN W=W+1
2465 GOSUB 2500:PRINT@1008,"ENERGY";EN;:IF EN<1 OR VE=1 THEN 260
0
2470 SET(W,Y):IFI<>13THEN2445
2480 PRINT@W1,JB;:GOTO2120
```

Compare distance of drill rig or robominer to Dilithium.


```

2500 RESET(W,Y):EN=INT(EN-(100/K(P,15))*VE)
2510 VE=0:PRINT@960,STRING$(63,32);:PRINT@960,"";:SET(W,Y):IF Y<
10 THEN PRINT "SURFACE ";:RETURN
2520 IF (W<V1-RA#3 OR W>V1+RA#4) OR (Y<V2-RB#3 OR Y>V2+K(P,4)+RB
#4) THENPRINT " ICE AND ROCK";:RETURN
2530 IF (W<V1-RA#2 OR W>V1+RA#3) OR (Y<V2-RB#2 OR Y>V2+K(P,4)+RB
#3) THENPRINT "THIRIDIUM ";:RETURN
2540 IF (W<V1-RA OR W>V1+RA#2) OR (Y<V2-RB OR Y>V2+K(P,4)+RB#2)
THEN PRINT "SECONIUM ";:RETURN
2550 IF (W<V1 OR W>V1+RA) OR (Y<V2 OR Y>V2+K(P,4)+RB) THEN PRINT
"FIRIDIUM ";:RETURN
2560 VE=1:RETURN

```

Discovery of Dilithium.

```

2600 IF EN<1 THEN 2800
2605 PRINT@960,STRING$(63,32);:PRINT@975,"EUREKA! You have found
DILITHIUM!!!";
2610 FORZ=1TO4:GOSUB30010:US=USR(0):GOSUB20010:GOSUB20110:NEXT
2620 PRINT@960,STRING$(63,32);:PRINT@1008,"STATUS";:X=0:Y=1:GOTO
2640
2630 PRINT@960,STRING$(63,32);:PRINT@965,"500 CREDITS";:X=13:Y=5
00:GOTO 2640
2640 GOSUB40000:IFI=13 THEN K(P,X)=K(P,X)+Y:K(P,16)=K(P,16)+1:GO
TO 2700
2650 IFI=9THEN 2620
2660 IFI=8THEN 2630
2670 GOTO 2640

```

Test for lack of energy or equipment for drilling.

```

2700 IF EN<1 THEN 2800
2710 IF K(P,1)>0 THEN 2000
2720 PRINT@960,"You do not have enough equipment left to continu
e drilling. ";:FORZ=1TO200:OUT255,0:OUT255,1:NEXT:GOTO3000
2800 PRINT@960,"You do not have enough energy left to continue d
rilling. ";:FORZ=1TO200:OUT255,0:OUT255,1:NEXT

```

Cave-In sequence.

```

3000 R=RND(10)-K(P,11):IF R<7 OR K(P,16)<1 THEN 3500
3010 CLS:GOSUB20130:PRINTCHR$(23);:PRINT@22,"RED ALERT!"
3030 PRINT@394,"CAVE IN:":R=INT(RND(10))
3040 PRINT@528,R;"WORKERS LOST!":K(P,14)=K(P,14)-R
3050 IF R>6 THENPRINT@658,"SHAFT NOW CLOSED!":K(P,16)=K(P,16)-1
3060 GOSUB40000:IF I<>13 THEN 3060
3070 GOTO 4000

```

Meteor Storm sequence.

```

3500 R=RND(10)-K(P,3):IF R<8 THEN 4000
3510 CLS:GOSUB20130:PRINTCHR$(23);:PRINT@22,"RED ALERT!"
3530 PRINT@394,"METEOR STORM!":R=INT(RND(10))
3540 PRINT@528,R;"WORKERS LOST!":K(P,14)=K(P,14)-R:IFR<7 THEN 35
90

```

```

3550 R=INT(RND(5)):IF K(P,R)<1 THEN 3590
3560 K(P,R)=K(P,R)-1:PRINT@658,MID$(EQ$,10#R-9,10);"DESTROYED!"
3590 GOSUB40000:IF I<>13 THEN 3590

```

Monthly report.

```

4000 Z=K(P,5):IF K(P,16)<Z THEN Z=K(P,16)
4010 K(P,13)=INT(K(P,13)+K(P,13)*0.1+500+(50#K(P,0))+(Z#25#K(P,8
)))
4100 CLS:GOSUB20110:PRINT@0,"MONTHLY REPORT":PRINT@29,"TITAN":PR
INT@50,DA;"/1/";YE
4130 PRINT:PRINT" ITEM ACTAEON BELLONA CHIM
ERA DAEDALUS":PRINT
4140 FORZ=0TO6:W=Z#10:IFW=0 THEN PRINT " STATUS ";:GOTO417
0
4160 PRINT " ";MID$(EQ$,W-9,10);
4170 FOR X=1 TO 4:PRINTTAB(7+X#11);K(X,Z);:NEXT:PRINT:NEXT
4180 FOR Z=1 TO 4:PRINT " ";MID$(EQ$,Z#10-9,10);
4190 FOR X=1 TO 4:PRINTTAB(7+X#11);K(X,Z+12);:NEXT:PRINT:NEXT
4210 PRINT@979,"Press ENTER to continue";:GOSUB20130
4220 GOSUB40000:IF I<>13 THEN 4220
4230 RETURN

```

Efficiency Rating algorithms.

```

5000 FF=0:FOR Z=1 TO 5:FF=FF+K(P,Z)#2:NEXT Z:IF FF<1 THEN FF=1
5010 EF=K(P,6)*10/FF:IF EF>1 THEN EF=FF/(K(P,6)*10)
5020 EF=EF#100:FF=K(P,7)*2.5:IF FF<0 THEN FF=FF#-2
5030 EF=EF-FF:FF=(K(P,8)-6)*3:IF FF<0 THEN FF=FF#-2
5040 EF=EF-FF+(K(P,9)*5)+(K(P,10)*5)+(K(P,11)*7.5)+(K(P,12)*10):
EF=INT(EF):IF EF<5 THEN EF=5
5050 IF EF>100 THEN EF=100
5060 IF K(P,9)=0 THEN EF=1
5070 K(P,15)=EF:RETURN

```

New Cost algorithms.

```

6000 FOR Z=1 TO 8:X=0:FOR Y=1 TO 4:X=X+K(Y,Z):NEXT Y:X=X/PL
6010 IF Z>5 THEN C(Z)=INT(C(Z)+(C(Z)*X/(55-LV))):GOTO 6050
6020 C(Z)=INT(C(Z)+(C(Z)*X/(25-LV))-(C(Z)*1/(25-LV)))
6030 IF C(Z)<100#Z THEN C(Z)=100#Z
6040 IF C(Z)>200#Z THEN C(Z)=200#Z
6050 IF Z>6 AND C(Z)<1 THEN C(Z)=1
6060 IF Z>6 AND C(Z)>50 THEN C(Z)=50
6070 IF Z=6 AND C(6)>4000 THEN C(Z)=4000
6080 NEXT Z:FOR Z=1 TO 4:FOR Y=8 TO 12:K(Z,Y)=0:NEXT Y:NEXT Z:RE
TURN

```

Adjust date.

```

7000 DA=DA+1:IF DA=13 THEN DA=1:YE=YE+1
7010 RETURN

```

End of Game phase.

```

8000 IF YE=2051 AND DA>3 THEN 8500
8010 R=INT(RND(2)):IF R=1 OR YE=2050 THEN RETURN
8500 PRINT@978,"The INSPECTOR has arrived!":GOSUB20140

```

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



```

8510 FORZ=1TO900:NEXT
8520 PRINT@974,"Your performance is being assessed.":FOR Z=1 TO
  900:NEXT Z:IF PL=1 THEN 8600
8530 Z=-1:Y=0:FOR X=1 TO PL:IF K(X,0)=Z THEN Z=K(X,0):Y=Y+1
8540 NEXT X:IF Y=1 THEN 8590
8550 FOR X=1 TO Y-1:IF K(X,0)=Z THEN 8570
8560 NEXT X:GOTO 8590
8570 FOR X=1 TO Y:IF K(X,0)<0 THEN K(X,0)=0:NEXT X:GOTO 8530
8580 K(X,0)=K(X,16)*100+K(X,15)+K(X,14)+K(X,13)+RND(9):NEXT X:GO
  TO 8530
8590 PRINT@960,"The supervisor of";MID$(A$,Y#10,10);" has won th
  e right to mine TITAN!":GOTO8700
8600 IFK(1,0)>19 THEN 8630
8610 PRINT@960," You did NOT achieve enough status to mine al
  l of TITAN! ";GOTO8700
8630 PRINT@960,"Because of your status you have won the right to
  mine TITAN!";
8700 GOTO 8700

```

Initial values.

```

9000 DA=0:YE=2050
9010 C(1)=100:C(2)=200:C(3)=300:C(4)=400:C(5)=500:C(6)=2000:C(7)
  =5:C(8)=10:C(9)=100:C(10)=200:C(11)=300:C(12)=400
9020 EQ$="Drill Rig Robominer Deflector R&D Unit Refinery Ener
  gizer CONTINUE "
9030 A$=" ACTAEON BELLONA CHIMERA DAEDALUS "
9040 WC$="Labor PoolShift TimeWage ScaleRecreationSafety Bonu
  ses CONTINUE "
9050 EX$="CREDIT MANPOWER EFFICIENCY* VEINS * "
9060 RETURN

```

Data lines.

```

10000 DATA 17,121,45,14,255,33,1,1,45,122,237,97,67,16,254,237,1
  05,67,16,254,61,32,243,21,32,239,201
10010 DATA217,33,255,63,6,60,126,254,128,56,4,47,246,128,119,43,
  124,184,48,242,217,201
10015 DATA175,155,160,24,24,24,26,183,183,139,160,143,144,24,24,
  24,26,159,131,175,32,32,32,24,24,24,26,32,32,32
10020 DATA 1,2,3,4,5,6,67,68,131,132,195,196,1,1,1,1,1,162,30
10030 DATA 9,10,11,12,74,75,138,139,201,202,203,204,9,9,9,9,9,
  108,30
10040 DATA 15,16,17,18,19,20,81,82,145,146,209,210,15,15,15,15,1

```

```

5,15,81,45
10050 DATA 24,25,26,27,87,88,91,92,151,152,153,154,155,156,214,2
  15,220,221,64,20
10060 DATA 32,33,36,37,96,97,98,100,101,160,161,163,164,165,224,
  225,228,229,68,50

```

Sound routines.

```

20000 JM="":FORZ=1TO27:READY:JM=JM+CHR$(Y):NEXT
20010 IFPEEK(16396)=201THEN20030 ELSE CMD"T":U=VARPTR(JM):U=PEEK
  (U+2)*256+PEEK(U+1):IFU>32767THENU=U-65536
20020 DEFUSR0=U:RETURN
20030 U=VARPTR(JM):POKE16526,PEEK(U+1):POKE16527,PEEK(U+2):U=PEE
  K(U+2)*256+PEEK(U+1):IFU>32767THENU=U-65536
20040 RETURN
20100 POKEU+1,UN:POKEU+2,UL:US=USR(0):RETURN
20110 FORUZ=1TO4:UN=40:UL=9:GOSUB20100:UN=80:GOSUB20100:NEXT:RET
  URN
20120 OUT255,0:OUT255,10:RETURN
20130 UL=3:FOR UN=250TO100STEP-10:GOSUB20100:NEXT:RETURN
20140 FORUZ=1TO30:OUT255,0:OUT255,6:NEXT:RETURN

```

Flash screen routine.

```

30000 JF="":FORZ=1TO22:READY:JF=JF+CHR$(Y):NEXT
30010 IFPEEK(16396)=201THEN30030 ELSE U=VARPTR(JF):U=PEEK(U+2)*2
  56+PEEK(U+1):IFU>32767THENU=U-65536
30020 DEFUSR0=U:RETURN
30030 U=VARPTR(JF):POKE16526,PEEK(U+1):POKE16527,PEEK(U+2):U=PEE
  K(U+2)*256+PEEK(U+1):IFU>32767THENU=U-65536
30040 RETURN

```

Get keyboard input.

```

40000 I$=INKEY$:IFI$=""THEN40000ELSEI=ASC(I$):RETURN

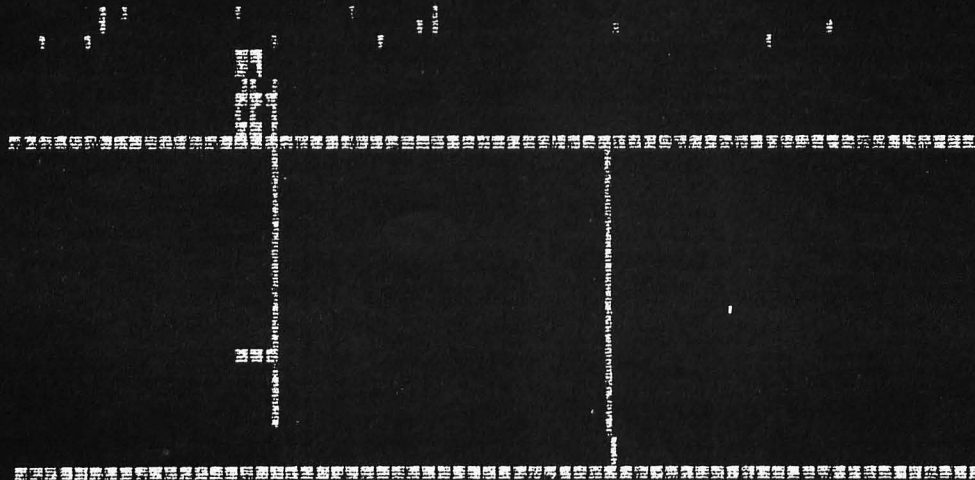
```

Initialize packed graphic strings.

```

50000 JR="":FORZ=1TO10:READY:JR=JR+CHR$(Y):NEXT
50010 JD="":FORZ=1TO10:READY:JD=JD+CHR$(Y):NEXT
50020 JB="":FORZ=1TO10:READY:JB=JB+CHR$(Y):NEXT:RETURN

```



EUREKA! YOU HAVE FOUND DILITHIUM!



ATARI-DV SURVIVE

by Randy Massey

Survive is a graphics action game for one or two players, requiring a 32K Atari, one or two joysticks, and a disk drive. It is included as a bonus program on this month's Atari Disk Version.

In the single-player version of this arcade-style game, you are fighting for survival against attacking aliens, in a space quadrant peppered with mines. Should you survive, you will face more aliens in 11 more quadrants, each more dangerous than the last.


The mines are actually your allies (as long as you don't collide with them yourself). You can lure an attacking alien to its destruction by positioning yourself on the opposite side of a mine, using the joystick. Colliding with a mine yourself will not affect your strength, but will reduce your score by ten points. Each quadrant has fewer mines than the previous one, making it more difficult to lure the aliens to their doom.

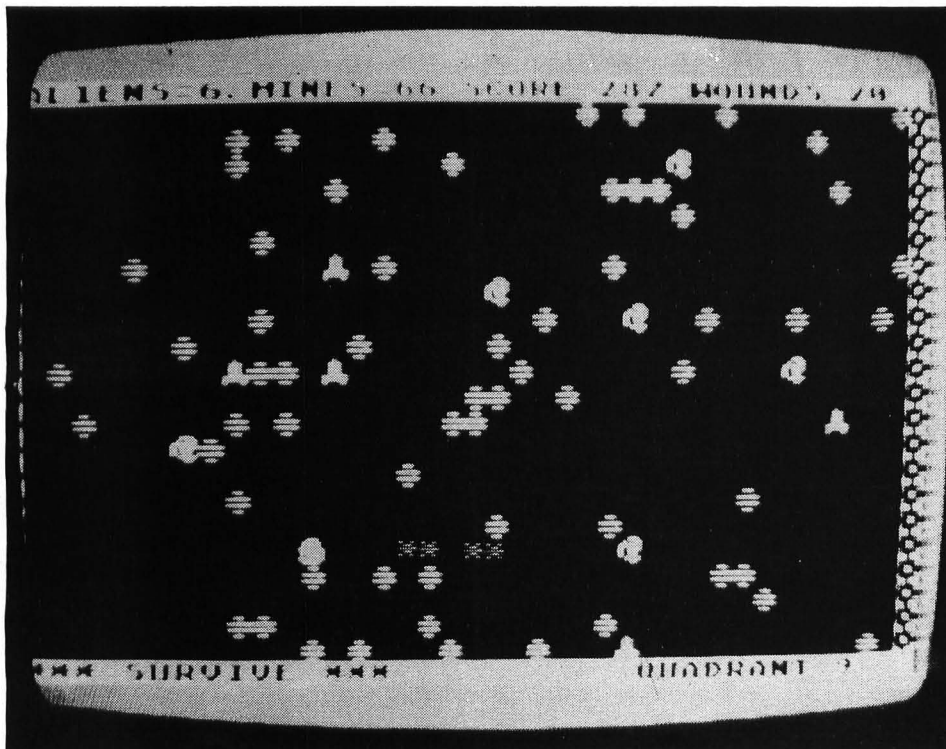
Both you and the aliens have laser beam weapons to fire at each other. You aim and fire by positioning the joystick in the desired direction and

pressing the fire button. If you hit an alien, it will be destroyed; if an alien hits you, you will incur injuries of 20%. When your injuries reach 100%, you've had it.

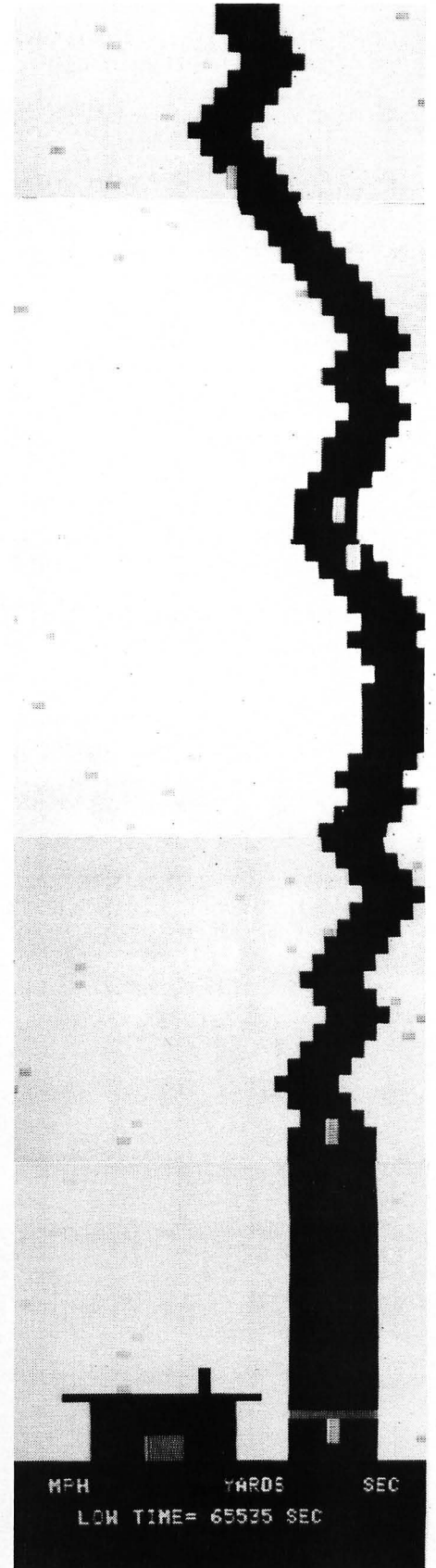
Every quadrant contains some starbases. Docking at one of these restores 5% of your injuries and adds 25 points to your score. Docking is accomplished by simply bumping into the base. If an alien destroys a starbase, you lose 25 points from your score.

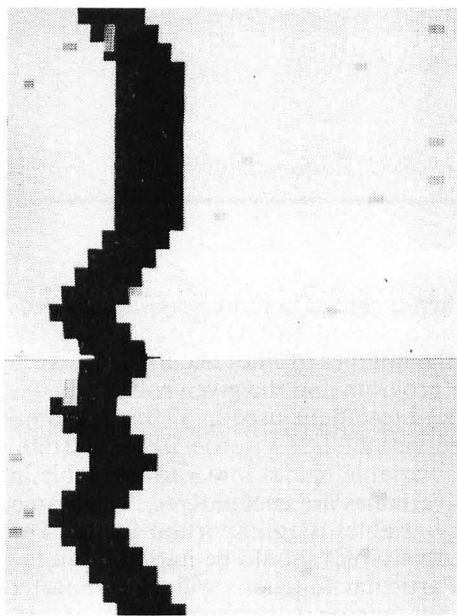
In the two-player version, the second player uses the second joystick to control the attacking alien force. This adds interest and challenge to the game, since the aliens can then be maneuvered more skillfully to avoid the mines. The aliens' firing is still controlled by the computer.

This program uses a modified character font, which is loaded at the beginning. This special font contains the peculiar-looking aliens and other special characters that are used throughout the game. In addition, a Machine Language routine, located in page six of memory, returns the joystick positions. 



APPLE-DV BOBSLEDDING





by Maxwell Su

Bobsledding is a graphics game for the Apple requiring 32K RAM, Applesoft, game paddles, and a disk drive. It is included as a bonus program on this month's Apple Disk Version.

Bobsledding originated in Switzerland at the beginning of the 19th Century and since then has been popular, especially in the European nations. The four-man bobsled competition was introduced in the Winter Olympic Games in 1924, and the two-man bobsled in 1932. This computerized version is a simulation of bobsledding, and reveals the reason for the excitement over this sport, where victory is measured in hundredths of a second.

This program employs a "real-time" clock which is reasonably accurate for a program running under Applesoft. The course consists of a two-mile track (3520 yards) which is randomly generated for each run. The slightest deviation from the course results in a crash and total destruction of the bobsled — and the end of that run! A low time is displayed, as well as the speed, distance travelled, and elapsed time. The low time is initially set to an arbitrarily large amount of time and, if your bobsled run is made in less time, it is then updated. You must finish the course (of course) to qualify for the low time.

There are nine levels of difficulty, 1 being the easiest and 9 the most difficult. After selecting the level of difficulty, press the button on Paddle 0 to begin the run. Paddle 0 is then used for steering, and Paddle 1 for braking. Dress warmly, and good luck!



S-80-DV

KONANE

by Norman Whaland

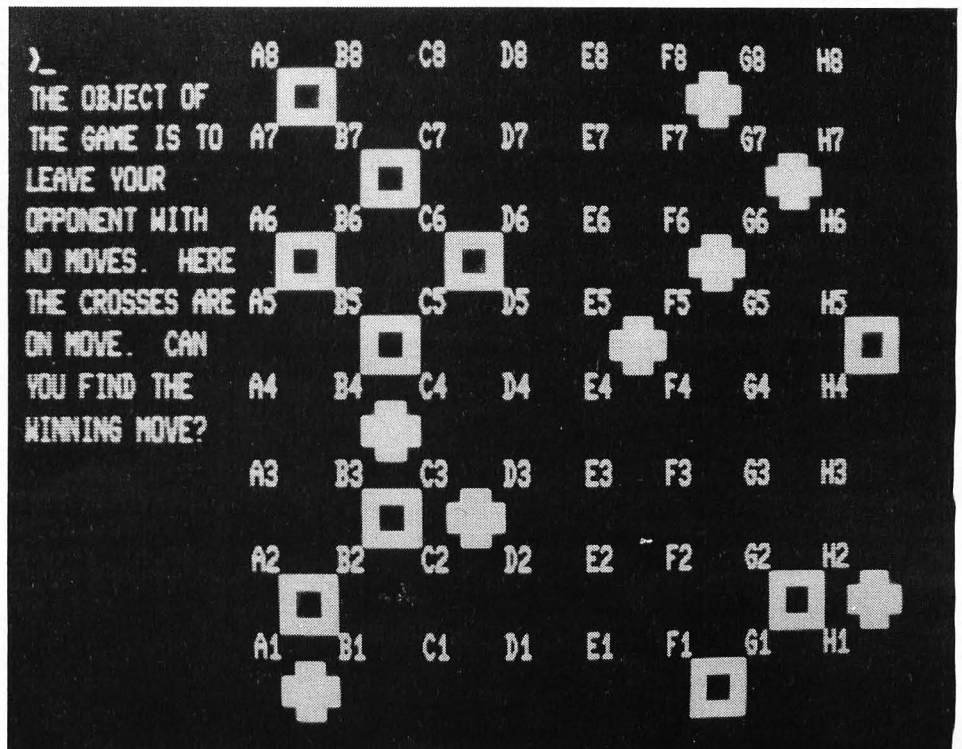
Konane is a TRS-80® Machine Language board game for one player requiring 32K RAM. It is included as a bonus program on this month's TRS-80® Disk Version.

In *Konane* (pronounced ko-NAH-nay), you play against the computer on an 8-by-8 board filled with equal numbers of both players' pieces. The goal of the game is to be the last player to make a move. With the exception of the first move, all moves are made by jumping. When a piece is jumped by one of the other player's pieces, it is removed from the board; in this respect, the game is similar to checkers. Multiple jumps are allowed, but only in a straight line. You can't zig-zag through a number of pieces.

The main game strategy is to isolate the other player's pieces, leaving them surrounded by empty squares so that they cannot move. When there are only a few pieces left on the board, each side tries to plan movements so that it has the last move, and thus wins the game.

The computer has four levels of play. Level 1 is the easiest, with the computer moving within a second; and Level 4 is the hardest, with the computer moving in about 15 seconds.

Konane is stored on the disk in two parts. Part 1 contains a thorough set of instructions, complete with graphics examples, and Part 2 is the actual game program.



K-Byters

Y-WING

A TRS-80® K-Byter by Roger W. Robitaille, Sr., *SoftSide*

Playing Instructions

There are five significant control keys on the TRS-80® keyboard, the four arrow keys and the space bar. The arrow keys control the screen action as though they are steering the target. If the target is below the cross-hair, you can make it rise by pressing the up-arrow. The arrow keys have to be struck repeatedly for substantial movement. The space bar acts as the trigger to fire the missiles.

Playing Considerations

When the program asks for your skill level, the value you declare is considered your handicap. Five different factors are set from this handicap. The first is the speed at which the target appears to move. Second is the delay time imposed between allowable missile shots and third, the number of missiles allowed for the mission. The fourth is the number of damage hits a fighter can sustain. And the fifth is the number of targets which must be destroyed. Their combined effects offer anything from a turkey shoot to a very sticky situation.

Program Type

The title of this program suggests the theme of one of the best-selling Ramware™ products, *X-Wing II*. This is not to suggest that *Y-Wing* is nearly as sophisticated as the *X-Wing II* product. It's simply a shoot-em-up fighter game.

Key Speed Points

Although BASIC is not known for its speed, an acceptable response speed is essential to make *Y-Wing* a satisfying game. The fighter movement routine is easily the most active routine in the program. By placing this routine at the beginning of the program and keeping the GOTOs and GOSUBs to a minimum, you can be assured that response

time will stabilize when included in more complicated programming. The time consumed on GOTOs and GOSUBs varies according to the number of lines occurring between the beginning of the program and the given routine.

Another useful speed technique used in *Y-Wing* is to initialize the most used variable, "I" in the DEFINT statement. BASIC stores variable names in a variable table in the order that these variables are encountered. The search time used in finding variables is an important factor. The most commonly used variables should be initialized in the order of their use. Particular functions will execute faster when dealing with an integer variable rather than the slower, single precision variable assumed by Level II when precision is unspecified.

Program Expansion

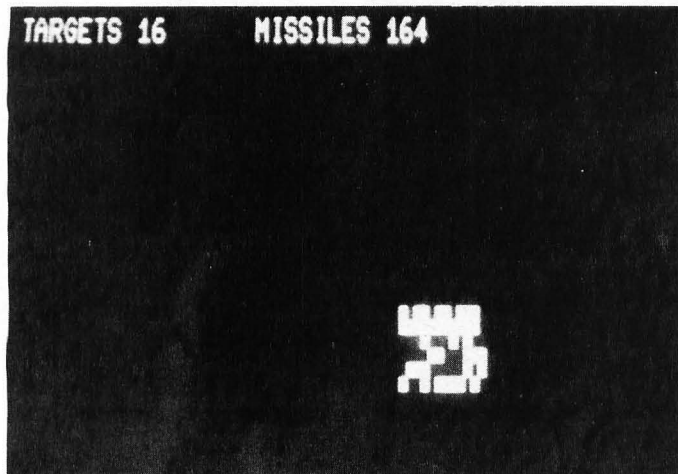
This program stands as a quality K-Byter. However, it has three intended purposes. The first purpose is obvious — to write a K-Byter. Second, this program is to serve as an example of a new, very intensive type of documentation in *SoftSide*. Third, *Y-Wing* will serve as an action subroutine for the Parsec™ Interpreter (*diversion's*™ first *Envyrment*™).

There is another reason for our presenting this little program. *Y-Wing* is very expandable. We are considering it an ongoing project and will be publishing expansion modules in the coming months and welcome your comments and input.

How would you like to try to add:

- return fire?
- intelligent return fire?
- a different type of ship graphics?
- a keyboard repeat?
- a landing procedure?
- a bail out procedure?
- a full instrument panel?
- different kinds of weapon systems?
- bombing runs on fixed targets?
- sound?

Let's hear your ideas!



Initialization.

```
100 CLS:
...Clear the screen.
DEFINT I:
...Define I as an integer for speed purposes.
DEFSTRA-C:
...Define all variables starting with letters A thru C to be
string variables to save typing "###"s. This is a K-Byter!
Y=140:
...Used as a constant to save bytes.
IM%=CHR$(157)+CHR$(174):
...Defines the appearance of the fighter craft.
I$=" ":
```

...The fighter's blanking variable. Erases the fighter's last image.
 CX=CHR\$(Y)+CHR\$(191)+CHR\$(Y);
 ...Defines the crosshairs graphic.
 X%=STRING\$(12,32)
 ...This prepares the variable for the explosion graphic. The LEFT\$ and RIGHT\$ functions used elsewhere would cause errors if it weren't filled with 12 characters first.

Skill Level Control.

105 PRINT@448,"HOW GOOD ARE YOU (1-100)";;
 ...Skill request message displayed near center screen.
 INPUTA;
 ...Actual request.
 I=VAL(A);
 ...For error protection, get input as a string even though a number is called for.
 T=30-INT(I/4)+1;
 ...The number of targets set to be 30, less one-quarter of declared skill.
 TP=T+1*3;
 ...The number of missiles allowed is set equal to the number of targets plus three times skill level.
 HM=INT(T/5)+2;
 ...The number of damage hits (near misses) the targets are allowed is equated to one-fifth the number of targets plus two.
 Z=10-INT(I/10);
 ...The game speed controller Z is set to be one-tenth of the skill level.
 RS=10-INT(I/10);
 ...The reload speed (pause before missile can be re-fired) set same as Z.
 CLS;
 ...Ready to go to work. Clean screen.
 PRINT@539,CX;;
 ...Display the cross-hairs.
 GOTO400
 ...Start play; launch first fighter.

Random Fighter Movement.

110 N=RND(3)-2+(64*(RND(3)-2));
 ...This function will only generate the numbers -65, -64, -63, -1, 0, 1, 63, 64, and 65. These are offset values which can be added to the fighter's present position to simulate movement to the 8 immediately adjoining positions.
 IFX+N<0ORX+N>1021THENGOTO110
 ...Must make sure that this randomly generated PRINT@ position is legal (0-1023). Further, position 1023 would generate screen roll and must be avoided.
 ELSE:
 FORI=1TOZ:NEXTI;
 ...Slow down loop; main skill control.
 IFRLOTHENPRINT@1010,"RELOADING";;
 ...If the reload cycle is still active, PRINT message to screen.

RL=RL-1;
 ...Decrement the reload cycle variable.
 PRINT@1010,CHR\$(30);
 ...Clean the reload message off the video.

Player Input.

120 B=INKEY\$:IFB<>" THEN200
 ...Simple live-key routine. If the player touches the keyboard, go check out what it's about.

Left/Right Screen Boundary Check.

130IF(X+N-(INT((X+N)/64)*64)=0)OR(X+N-(INT((X+N)/64)*64)>62)THEN
 110
 ...Loosely read: If the fighter image would be printed in the first column or the last two columns then try again.

Screen Refresh.

140 PRINT@X,I\$;;
 ...Erase the fighter image.
 PRINT@539,CX;;
 ...Print the cross-hairs, just in case.
 X=X+N;
 ...Add the offset to the present position, thus updating X.
 PRINT@X,IM\$;;
 ...Finally, print the fighter image.
 GOTO110
 ...Do it--again.

Utilities.

198 CLEAR50:PRINT38639-MEM-49:END
 199 SAVE"YWING/SUB":SAVE"YWING/SUB:1":END

Steering Control.

200IFB=CHR\$(8)ANDX-(INT(X/64)*64)>1N=-2
 ...Left Arrow pressed. Alter position two to the left.
 ELSE
 IFB=CHR\$(9)ANDX-(INT(X/64)*64)<61THENN=2
 ...Right Arrow pressed. Alter position two to the right.
 ELSE
 IFB=CHR\$(10)ANDX<958THENN=64
 ...Down Arrow pressed. Alter position one line down on the screen.
 ELSE
 IFB="["ANDX>63THENN=-64
 ...Up Arrow pressed. Alter position one line up on the screen.
 ELSEGOTO240
 ...The key pressed is not one of the arrows; keep looking at 240.
 205 GOTO140
 ...In all the cases where there were arrows depressed, go to line 140 for screen refresh.

Missile Fire.

```
240 IFB=" ANDTP>0ANDRL=0THEN
...If the player has pulled the trigger (space bar) and you
still have some missiles and you are not in reload you may...
TP=TP-1;
...Fire the missile (decrement TP).
RL=RS;
...Reset the reload cycle.
V=62;
...Initialize the right missile position variable.
FORI=960T0539STEP-60;
...Control loop for missile trace.
PRINT@I,CHR$(Y);:PRINT@I+V,CHR$(Y);;
...Print missile trace graphics, both left and right sides.
PRINT@I," ";:PRINT@I+V," ";;
...Erase missile trace graphics, both left and right sides.
V=V-9;
...Adjust right side controlling variable.
NEXTI:
...Loop.
PRINT@539,CX;:
...Reprint cross-hair graphic.
IFX>537ANDX<542THENGT0305
...If the target is in the damage zone, go to line 305 to
inflict damage.
290 GOTO110
...End of keyboard command options. If the player missed the
target or simply pressed the wrong key (or a key whose use has
not yet been defined), start over.
```

Fighter damage.

```
305 IFX=537ORX=541THEN320
...If the fighter's cursor position is either 537 or 541, it
has received wing damage. Go to line 320 for damage graphics
and assessment.
```

Fighter destruction.

```
310 FORI=1T050;
...Initialize a loop for fifty cycles.
X%=RIGHT$(X%,11)+CHR$(RND(64)+127);
...The method used to create destruct graphics is quite
primitive. The essential process is to randomize one of the 64
graphic characters and add it onto the end of a
twelve-character string. This string is split in two parts and
printed in two halves, one over the other. The effect is fairly
good for its simplicity.
PRINT@474,LEFT$(X%,6);;
...Print the top half of the destruct graphic.
PRINT@538,RIGHT$(X%,6);;
...Print the bottom half of the destruct graphic.
NEXTI:
...Continue animated effect.
GOTO400
...Animated graphic complete. Proceed to document and display
event at 400.
```

Damage routine.

```
320 FORI=1T010;
...This graphic is equally rough. For 10 cycles randomly
```

```
select a graphic character to print over the left side of the
fighter graphic.
PRINT@X,CHR$(RND(64)+127);;
...Print the random character.
NEXTI:
...Do it again, Sam.
H=H+1;
...Document the damage hit.
IFH=HMTHEINGOTO310
...If the number of damage hits is equal to the maximum
allowed, the fighter is destroyed anyway, so go to fighter
destruction section.
ELSEGOTO140
...Time to freshen the screen and get that fighter out of the
way.
400 T=T-1;
...Knock that victory; another one bites the dust.
IFT<160T0500
...Looks like we wiped them out; time to celebrate.
ELSEPRINT@0,"TARGETS";T,"MISSILES";TP;:
...Print a progress report for the player.
FORI=1T0500:NEXTI:
...A straight timing loop to allow player time to read message
before play proceeds. This is usually a bad technique because
there are other ways to let the player decide when he is done
with the information. It is used here because I think that it's
more important to keep it a challenge by keeping the game
pressing the player. We'll see.
H=0;
...Zero the accumulated hit counter.
X=RND(3)*64+RND(50);
...This is a special function to randomly devise a new
starting position for the next pidgeon--I mean fighter.
CLS:
...Wash the blood off the screen.
GOTO140
...and go freshen the picture one more time.
```

Victory Routine.

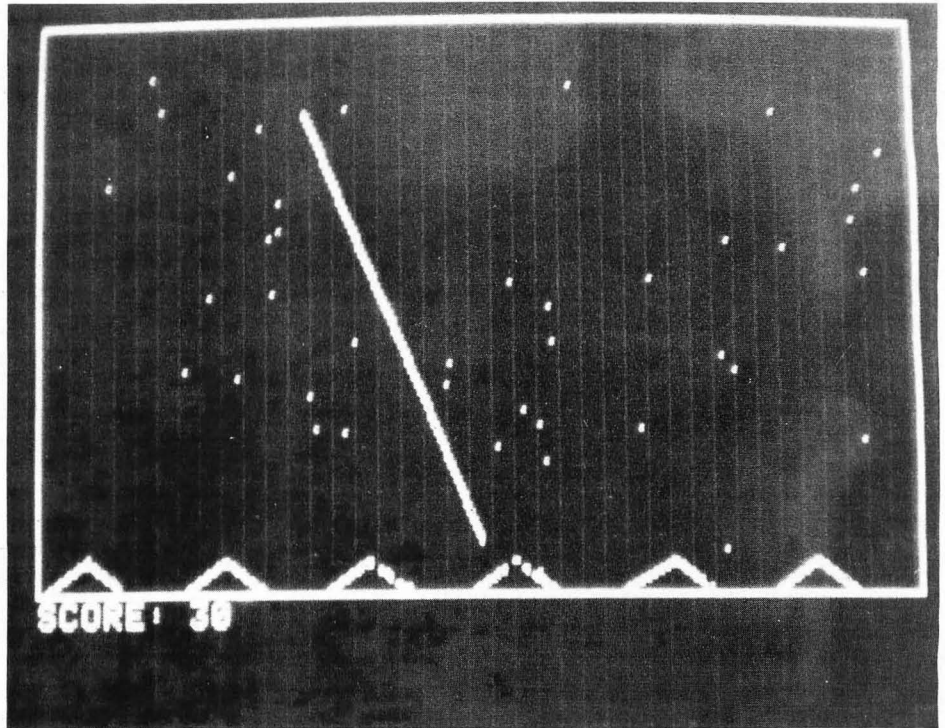
```
500 FORI=1T0100;
...Literally every other PRINT statement in this program
closes with a semicolon because of the necessity of keeping
close screen control. The one below does not, so that it will
trigger screen roll causing an effect coarsely similar to the
"Star Wars" introduction (this is a K-byter, remember that).
...One hundred "atta-boy's" coming up.
PRINT@980,"THE EMPIRE IS SAFE - THE FORCE BE WITH YOU":
...My Hero!!!!!!!!!!!!
FORJ=1T020;
...A probably unnecessary delay loop.
NEXTJ,I:
...A compound NEXT use. Otherwise, the ceremony continues till
completion.
CLS:
...Wash the screen of all the graffiti.
INPUT"NEW GAME (Y/N)";A;
...Get the Nielsen rating.
IFA="Y"GOTO100
...He loves me...he loves me not.
```


INVADER CATCH

An Applesoft K-Byter by Matt Laurence, Lincoln, MA

Some nasty aliens from the galaxy of Bombay are dropping bombs on you and your cities. Luckily for you, the bombing systems of Bombay are not very sophisticated, and can only drop one bomb at a time. The bombs, on the other hand, are fast, and will fall to the cities below before your neutralizer can move from one end of the screen to the other. In these cases, when you cannot reach the bomb in time to catch it, you may fire one of your phasers at it. Hitting a bomb with a phaser does not add any points to your score, but it does destroy the bomb.

You start out with 100 points. Each time a bomb gets past you and hits the ground, you lose 10; each time you catch one, you gain 10. If your score drops to zero, the game ends and you lose. But if your score reaches 200, the aliens will retreat, and you win. Be warned: You have only five phasers, so don't be trigger happy. Paddle 0 controls the movement of your bomb neutralizer and button 0 fires a phaser. Good luck!



```
0 REM "INVADER CATCHER"
  BY MATT LAURENCE
```

```
1 HOME : VTAB 12: PRINT TAB( 13
  );"INVADER CATCHER": FOR Q =
  1 TO 1000: NEXT Q: HGR : HOME
  : HCOLOR= 3: VTAB 21: GOSUB
  300
2 HCOLOR= 3: HPLLOT 0,0 TO 279,0 TO
  279,159 TO 0,159 TO 0,0:SH =
  5:X = 140:S = 100: GOSUB 100
  0
3 R = INT ( RND (1) * 240) + 20:
  HCOLOR= 3: HPLLOT 0,159 TO 2
  79,159
4 H = 10
5 Y = (( PDL (0) / 255) * 250) +
  20
6 HCOLOR= 3
8 IF X > 260 THEN X = 260
9 IF X < 20 THEN X = 20
10 GOSUB 100: FOR A = 1 TO 15: NEXT
  A: HCOLOR= 0: GOSUB 100: HCOLOR=
  3
```

```
15 GOSUB 500: FOR A = 1 TO 15: NEXT
  A: HCOLOR= 0: GOSUB 500
17 IF H = 145 AND (R < X + 8 AND
  R > X - 8) THEN HCOLOR= 0:S
  = S + 10: GOSUB 500: FOR A =
  1 TO 200: NEXT A: GOSUB 1000
  : GOTO 3
18 IF S < 1 THEN TEXT : HOME : VTAB
  21: PRINT "EARTH IS DEAD!": END
20 IF H > 150 THEN HCOLOR= 0: GOSUB
  500: HCOLOR= 3:S = S - 10: FOR
  A = 1 TO 200: NEXT A: GOSUB
  1000: GOTO 3
21 IF PEEK ( - 16287) > 127 AND
  SH > 0 THEN HCOLOR= 3: HPLLOT
  X,145 TO R,H: GOSUB 200: HCOLOR=
  0: HPLLOT X,145 TO R,H: GOSUB
  500:SH = SH - 1: GOTO 3
23 IF Y < 75 THEN X = X - 15
24 IF Y > 175 THEN X = X + 15
30 H = H + 15
40 GOTO 5
```

```
100 HPLLOT X - 8,145 TO X + 8,145
101 RETURN
200 FOR W = 1 TO 100:SO = PEEK
  ( - 16336) + PEEK ( - 16336
  ): NEXT W: RETURN
300 HPLLOT 0,160 TO 15,150 TO 30,
  160: HPLLOT 45,160 TO 60,150 TO
  75,160: HPLLOT 90,160 TO 105,
  150 TO 120,160: HPLLOT 135,16
  0 TO 150,150 TO 165,160: HPLLOT
  180,160 TO 200,150 TO 215,16
  0: HPLLOT 230,160 TO 245,150 TO
  260,160
310 FOR W = 1 TO 50: HPLLOT INT
  ( RND (1) * 270) + 10, INT (
  RND (1) * 150) + 10: NEXT W
  : RETURN
500 HPLLOT R,H TO R - 4,H - 9 TO
  R + 4,H - 9 TO R,H: RETURN
1000 HOME : VTAB 21: PRINT "SCOR
  E: "S: RETURN
```

SOFTSIDES FROM THE PAST

If you like what this issue of **SoftSide** has to offer, you should see what's waiting for you in **SoftSide** back issues! You may feel that you've missed out on many of our programs that appeared before you became a subscriber. It's not too late to do something about it. You won't have to miss a thing because we still have back issues available to complete your **SoftSide** library! But, order now as some of our more popular issues are already out of stock and others are dwindling quickly.

Listed below are all of our past issues with their **FEATURE** programs and the systems they're for. For a more complete

index of all the programs and articles offered in each of the back issues of **SoftSide** please refer to the May, 1981, issue. Each issue costs \$3.50 for the magazine only. Those issues marked with an asterisk are also available with the programs on cassette for \$9.95 or on disk for \$14.95 (except DV issues).

The enhanced Disk Versions (DV) contain an extra program for each system. The TRS-80® DV began with the September, 1981, issue. The Apple DV began October, 1981, and the Atari DV started in November, 1981. Each enhanced DV costs \$19.95.

SoftSide TRS-80® Edition:

- October 1978 — Not Available
- November 1978 — "End Zone"
- December 1978 — "Santa Paravia En Fiumaccio"
- *January 1979 — "Round the Horn"
- *February 1979 — "Form 1040"
- *March 1979 — "Personal Finance"
- *April 1979 — "Safari"
- *May 1979 — "Dog Star Adventure"
- *June 1979 — "Atlantic Balloon Crossing"
- *July 1979 — "All Star Baseball"
- *August 1979 — "Meltdown"
- September 1979 — Not Available
- *October 1979 — "Westward 1847"
- November 1979 — Not Available
- *December 1979 — "Oil Baron"
- *January 1980 — "Moving Maze"
- February 1980 — Not Available
- *March 1980 — "Broadway"
- April 1980 — Not Available
- *May 1980 — "Star Trek"
- *June 1980 — "Micro Millionaire"
- *July 1980 — Adventure Issue

PROG 80 (TRS-80® only)

- March 1979 — Not Available
- May 1979 — "Clock Routines"
- July 1979 — "Histogram and BASIC Statistics"
- September 1979 — "DOS-How to stop lost data errors"
- December 1979 — "Computer Telephone Dialing"
- February 1980 — "Hexmem"
- April 1980 — "Redefining Level II Keyboard"
- June 1980 — "Z-80 Disassembler"
- August 1980 — "Varlist"-A program to list variables

SoftSide: Apple Edition

- *January 1980 — "Dog Star Adventure"
- *February 1980 — "Connection"
- *March 1980 — "Treasure Hunt"
- *April 1980 — "Jig-Saw"
- *May 1980 — "Invaders"
- June 1980 — Not Available
- *July 1980 — "Pork Barrel"

SoftSide: Apple, Atari and TRS-80® Combined Edition

- *August 1980 — "Caribbean Cruising" — Apple
 - "Master's Golf" — Atari
 - "Sailplane" — TRS-80®
- September 1980 — Not Available
- *October 1980 — "Developing Data Base II" — All
 - "Moonlanding" — Apple
 - "World Series" — Atari
 - "Earth-Port II" — TRS-80®
- *November 1980 — "Developing Data Base III" — All
 - "Collision" — Apple
 - "Trench" — Atari
 - "Kriegspiel" — TRS-80®

- *December 1980 — "Developing Data Base IV" — All
 - "Baseball" — Apple
 - "Speedello" — Atari
 - "Kidnapped" — TRS-80®

- *January 1981 — "Developing Data Base V" — All
 - "Convoy" — Apple and S-80
 - "Angle Cannon" — Atari
 - "Ship Destroyer" — TRS-80®

- *February 1981 — "Developing Data Base VI" — All
 - "Miner" — All
 - "Mini-Golf" — Atari and TRS-80®
 - "Long Distance" — TRS-80®

- *March 1981 — "Developing Data Base VII" — All
 - "Strategy Strike" — Apple and TRS-80®
 - "Flags" — Atari
 - "Volcano" — TRS-80®

- *April 1981 — "Battle At Sea" — Apple
 - "Convoy" — Atari
 - "Dominoes" — TRS-80®

- *May 1981 — "Galaxia" — Apple
 - "Dodge" — Atari
 - "Orienteering At Jacque's Coulee" — TRS-80®

- *June 1981 — "Old Glory" — All
 - "Word-Search Puzzle Generator" — All
 - "Anallist" — TRS-80®

- *July 1981 — "Chemistry Drill" — All
 - "Kidnapped" — Apple and Atari
 - "Magic Paper Calculator" — TRS-80®

- *August 1981 — Not Available

- *September 1981 — Not Available

- *October 1981 — "Envyrn™" — TRS-80®
 - "Leyte" — All
 - "Developing Data Base" — Apple
 - "Character Generator" — Atari

- *November 1981 — "Titan" — All
 - "Aircraft Commander" — Apple
 - "Electronics Assistant" — TRS-80®
 - "Developing Data Base" — Atari

Use the bind-in card in this issue to order or send a list of the back issues you'd like, with payment of \$3.50 per magazine.

SoftSide Publications
515 Abbott Drive
Broomall, PA 19008

For the magazine/media combination, send \$9.95 per cassette and magazine, \$14.95 per disk and magazine, or \$19.95 per DV and magazine to:

SoftSide Publications
6 South Street
Milford, NH 03055

CHRISTMAS TREE

An Atari K-Byter by Gary Domrow,
Fort Wayne, IN

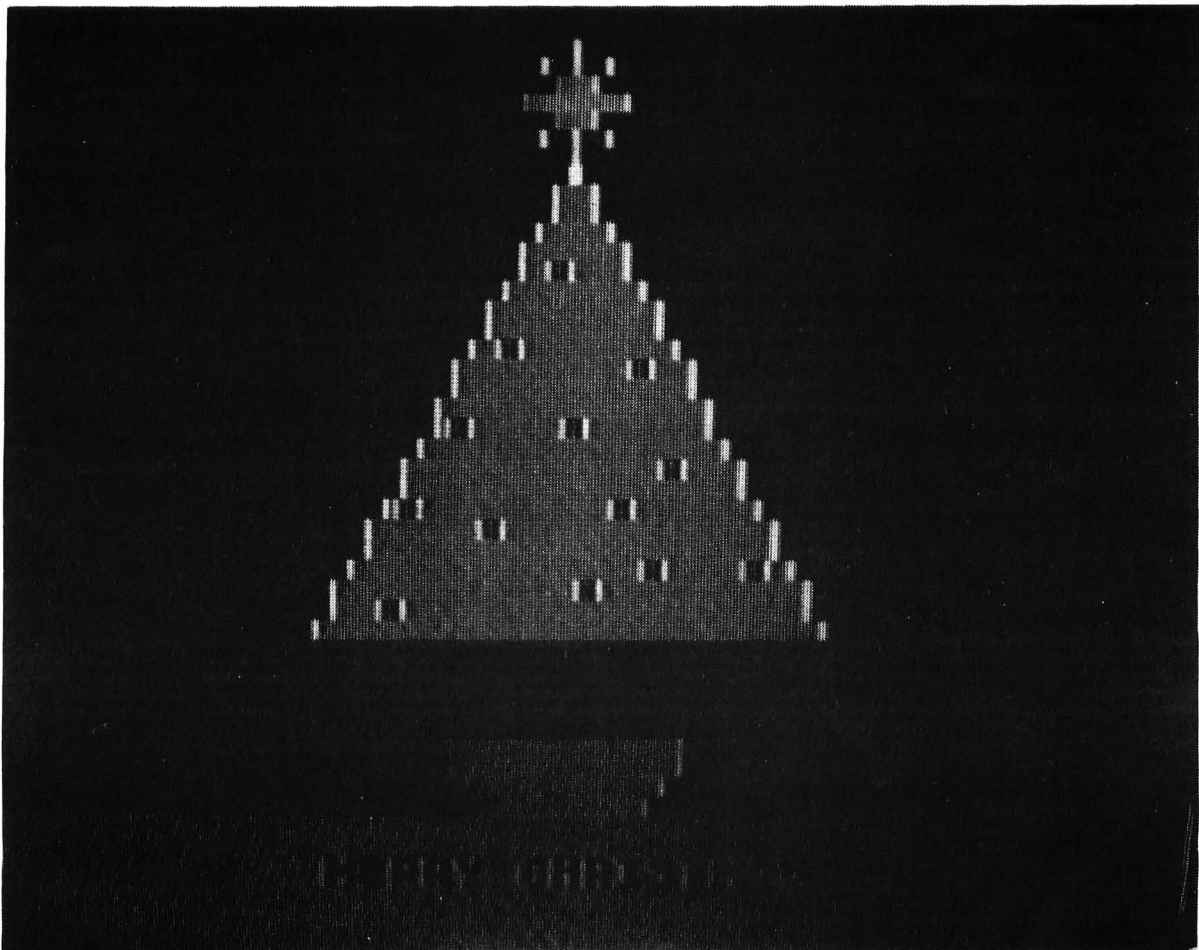
The message of this K-Byter is self-explanatory. It draws a brightly-colored Christmas tree (what other kind is there?) with flashing lights, and prints the message "Merry Christmas".

Line 10 sets up the graphics for mode 5 and begins to draw the tree. Line 20 fills in the tree. Line 30 draws the red pot at the bottom, and line 40 draws the star at the top. Line 50 draws the trunk of the tree and prints "Merry Christmas". Lines 60-80 repeatedly plot and erase the blinking lights of the tree.

Season's Greetings to all!

```
10 GRAPHICS 5:SETCOLOR 2,12,6:SETCOLOR
  1,3,4:SETCOLOR 0,1,2:COLOR 3:PLOT 55,
30:DRAWTO 40,7:POKE 752,1
20 POSITION 25,30:POKE 765,3:X10 18,#6
  ,0,0,"S":COLOR 2
30 PLOT 35,39:DRAWTO 45,39:PLOT 34,38:
DRAWTO 46,38:PLOT 33,37:DRAWTO 47,37:P
LOT 33,36:DRAWTO 47,36
40 PLOT 40,6:DRAWTO 40,0:PLOT 37,3:DRA
WTO 43,3:PLOT 38,1:DRAWTO 42,5:PLOT 38
,5:DRAWTO 42,1
50 COLOR 1:FOR I=39 TO 41:PLOT I,31:DR
AWTO I,35:NEXT I:? :? "      MERR
Y CHRISTMAS"
60 FOR B=2 TO 3:COLOR B:PLOT 44,17:PLO
T 29,29:PLOT 33,20:PLOT 30,24
70 PLOT 46,22:PLOT 39,12:PLOT 51,27:PL
OT 43,24:PLOT 35,25:PLOT 36,16:PLOT 40
,20:PLOT 45,27:PLOT 41,28
80 FOR W=0 TO 160:NEXT W:NEXT B:GOTO 6
0
```

5



EnvyrnTM

ENVYRNTM SCREEN CHOICES

by Roger W. Robitaille, Sr.

The TRS-80TM's graphics capability is, essentially, a special set of pixel combinations that are treated by the computer just as if they were letters, punctuation, or numbers. Many screen options are available to this group which could not be supported in computers with more complex and exclusive graphics capabilities. In other words, the TRS-80TM's graphics capability is primitive. However, it does present special opportunities, along with its restrictions. Much of the following will only pertain to computers which have a graphics character set accessible in text mode.

MAIN VIEWING SCREEN — Almost every EnvyrnTM application will make some use of its ability to tie map tiles together into some larger visual image. The Main Viewing Screen (MVS) on the TRS-80TM is one of four configurations by which related map tiles can be displayed together.

SCREEN PAGE — A Screen Page is a user-defined set of information windows which together will define what information will be presented on the screen, and where and how it will be shown.

There are five practical MVS choices in the TRS-80TM graphics structure. They are:

0% Text Screen — Text only, no main MVS.

10% Peek Screen — Small (18 chr x 6 line) MVS in the upper left corner of the screen.

30% Travel Screen — Medium size (30 chr x 10 line) MVS in the upper left corner of the screen

75% Look Screen — Large (48 chr x 16 line) MVS with a narrow (16 chr x 16 line) area still available to be assigned other information

95% Full Screen — Nearly the entire useable screen is dedicated to being the MVS. At (@) location #### is the information point of any keyboard activity.

As I examine the table in Figure 1, I note the following: The two tile sizes with the lowest resolution are completely efficient under all MVSs. The third lowest tile size averages over 90% efficiency. Since these may be the most commonly used tile sizes, I feel assured that these five MVSs are indeed the most practical to be selected. The larger tile sizes will be used primarily for special video displays so their efficiency is less significant. I hope that future EnvyrnTM enhancements will allow the MVSs to be completely user-defined.

Figure 2

MVS LAYOUT

```

18          30          48          60
1EEEEEEEEEEEEEEEE:TTTTTTTTTT;LLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
2EEEEEEEEEEEEEEEE:TTTTTTTTTT;LLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
3EEEEEEEEEEEEEEEE:TTTTTTTTTT;LLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
4EEEEEEEEEEEEEEEE:TTTTTTTTTT;LLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
5EEEEEEEEEEEEEEEE:TTTTTTTTTT;LLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
6EEEEEEEEEEEEEEEE:TTTTTTTTTT;LLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
7//////////////////:TTTTTTTTTT;LLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
8TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT;LLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
9TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT;LLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
10TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT;LLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
11//////////////////:LLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
12LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
13LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
14LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
15LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX
16LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL;FFFFFFFFFFFFXXX

```

Figure 1

SCREEN KEY Resolution Table for MVSs

Tile Size	Text	Peek	Travel	Look	Full
1x3	N/A	6x6*	10x10*	16x16*	16x20*
2x6	N/A	3x3*	5x5*	8x8*	8x10*
3x9	N/A	2x2*	3x3(90%)	5x5(91%)	6x5(82%)
4x12	N/A	1x1(47%)	2x2(63%)	4x4*	4x5*
5x15	N/A	1x1(69%)	2x2*	3x3(91%)	3x4*

* — 100% useful (nn%) — % useful

Key:

- E — In use under all MVSs.
- T — In use under all except Peek Screen.
- L — In use under Look and Full Screens.
- F — In use as MVS only under Full Screen.
- X — Always in use as Text.
- : — Border area where text will reject text windows and graphics will reject graphics windows.
- / — Border area where graphics warning will be offered.





Electronics Assistant

by John D. Adamson

Electronics Assistant is a TRS-80® utility program for helping in the design of electronic circuits. It requires 16K of RAM.

There are two main facets of a computer. One is the program that tells the computer how to do something. The other is the actual electronic circuitry that makes the computer understand and use the information provided by the program.

Electronics Assistant is a utility for the “other” side of computers, and for a variety of electronic design applications. Its main function is to help in the preparation of a circuit design by eliminating most of the lengthy and repetitive calculations that would otherwise take up a large amount of the designer’s time. The circuits it is programmed to handle range from attenuators to filters to coil design. There is a circuit here for almost any need. Enjoy!

Variables

All non-string variables except those listed below are integers and are used for display I/O.

C\$: String value of the INKEY\$ routine.

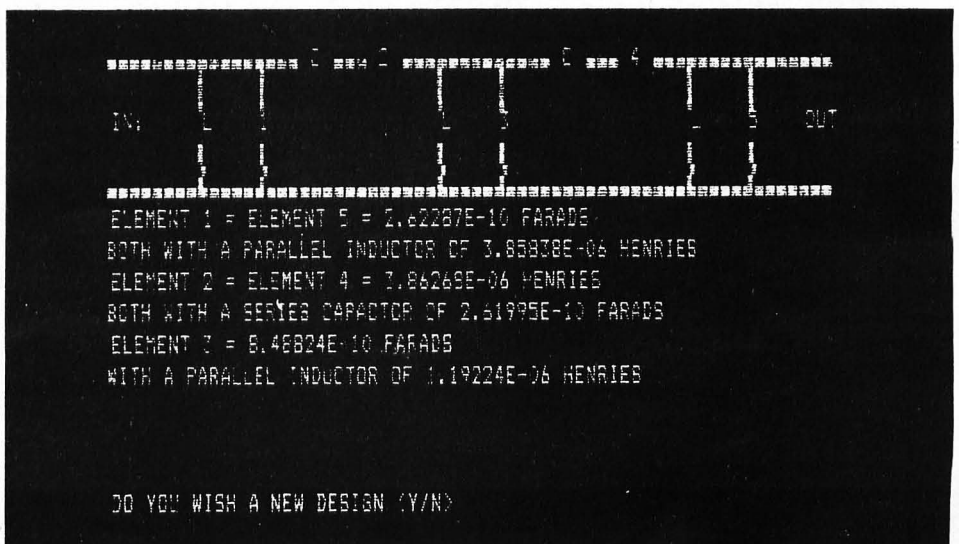
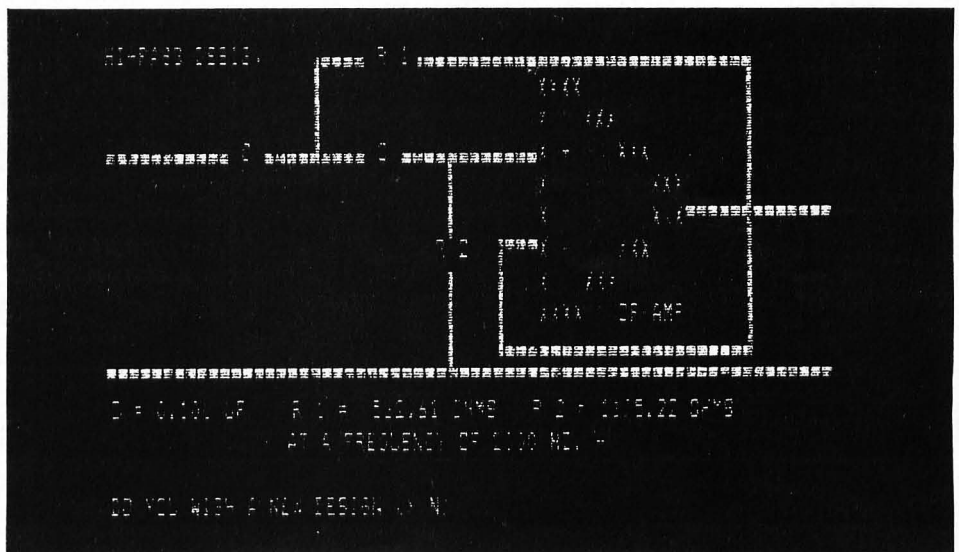
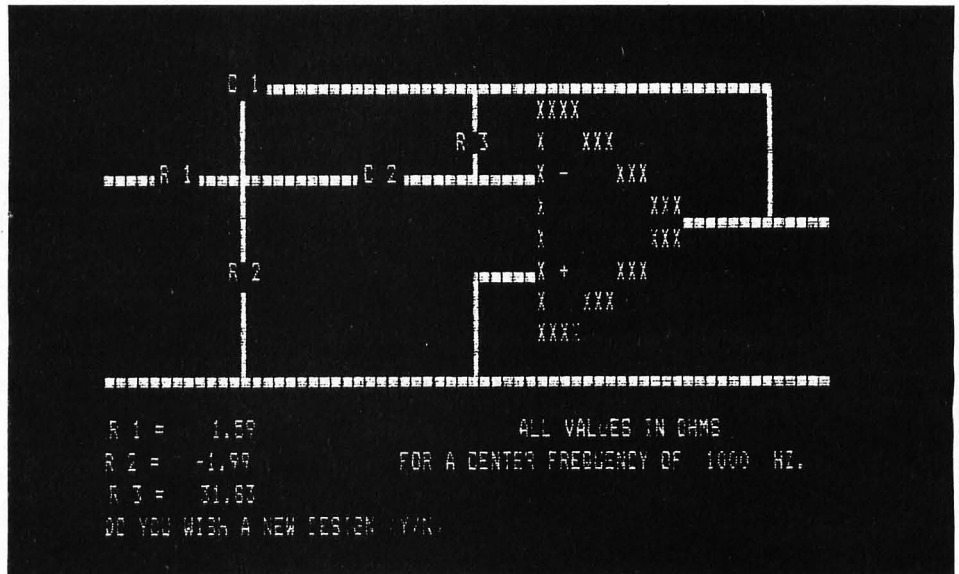
CC: Return from the INKEY\$ routine as the numeric value of the input.

CO\$: Used in the PRINT USING function. Set to “##.###”.

R\$: Used in the PRINT USING function. Set to “#####.##”.

F\$: Set to the word “FARADS”.

H\$: Set to the word “HENRIES”.



Program introduction.

```
100 CLS:PRINT@456,CHR$(23)"ELECTRONICS ASSISTANT":FORX=1TO1000:
NEXT
110 CLS:PRINT@338,CHR$(23)"PRESENTED BY"
120 PRINT@590,"JOHN D. ADAMSON"
160 FORX=1TO1200:NEXT
```

Program selection routine.

```
170 CLEAR100:CLS:PRINTTAB(23);"SELECTIONS":PRINTSTRING$(64,"=");
R$="#####.##";C0$="###.###";F$="FARADS":H$="HENRIES"
180 PRINT:PRINTTAB(10);"1 - ACTIVE BAND PASS FILTER DESIGN"
190 PRINTTAB(10);"2 - PI-NETWORK ATTENUATOR"
200 PRINTTAB(10);"3 - ACTIVE HIGH/LOW PASS FILTER DESIGN"
210 PRINTTAB(10);"4 - PASSIVE HIGH/LOW PASS FILTER DESIGN"
220 PRINTTAB(10);"5 - BAND-PASS FILTER DESIGN"
230 PRINTTAB(10);"6 - COIL DESIGN"
240 PRINTTAB(10);"7 - IMPEDANCE-MATCHING NETWORKS DESIGN - I"
250 PRINTTAB(10);"8 - IMPEDANCE-MATCHING NETWORKS DESIGN - II"
260 PRINTTAB(10);"9 - IMPEDANCE-MATCHING NETWORKS DESIGN - III"
270 PRINT@910,"ENTER YOUR SELECTION ?":GOSUB4210
280 CLS:IFCC<=0THEN170
290 ONCCGOTO300,630,870,1420,2130,2610,2940,3380,3820
```

Active Band-Pass filter design.

```
300 GOSUB4230
310 PRINT@457,"THIS PROGRAM IS FOR ACTIVE BAND PASS FILTERS"
320 GOSUB4250:GOSUB480
330 PRINT@640,"ENTER VOLTAGE GAIN ";:INPUTA
340 INPUT"ENTER Q":Q:V=(ABS(A)/Q)
350 INPUT"ENTER CENTER FREQUENCY <HZ>":F
360 INPUT"ENTER VALUE OF C1 & C2 IN <UF>":L,M
370 PRINT@576,"":PRINTCHR$(31)
380 L=(L*1E-06):M=(M*1E-06):W=6.2832*F
390 H=1/(V*L*W):I=1/(Q*(L+M)*W):J=(H*I)/(H-I):K=ABS(A)*H*(1+L/M)
400 PRINT"R 1 ="USINGR$;H
410 PRINT"R 2 ="USINGR$;J
420 PRINT"R 3 ="USINGR$;K
430 PRINT@740,"ALL VALUES IN OHMS";
440 PRINT@794,"FOR A CENTER FREQUENCY OF ";F;" HZ.";
450 GOSUB4260
460 IFC$="N"THEN170ELSEIFC$="Y"THENGOSUB480ELSE450
470 GOTO330
480 CLS:PRINT@11,"C 1";:PRINT@159,"R 3";:PRINT@102,"XXXX";
490 PRINT@197,"R 1";:PRINT@215,"C 2";:PRINT@166,"X XXX";
500 PRINT@230,"X - XXX";:PRINT@294,"X XXX";
510 PRINT@358,"X XXX";:PRINT@395,"R 2";
520 PRINT@422,"X + XXX";
530 PRINT@486,"X XXX";
540 PRINT@550,"XXXX";
550 Y=1:FORX=29TO117:SET(X,Y):NEXT
560 Y=10:FORX=0TO8:SET(X,Y):NEXT:FORX=17TO44:SET(X,Y):NEXT
570 FORX=53TO75:SET(X,Y):NEXT
580 Y=14:FORX=102TO127:SET(X,Y):NEXT:Y=19:FORX=65TO75:SET(X,Y):N
EXT
590 Y=29:FORX=0TO127:SET(X,Y):NEXT:X=24:FORY=3TO10:SET(X,Y):NEXT
Y
600 X=65:FORY=1TO5:SET(X,Y):NEXT:Y=8:SET(X,Y):Y=9:SET(X,Y)
610 X=117:FORY=1TO14:SET(X,Y):NEXT:Y=24:FORY=9TO17:SET(X,Y):NEX
TY
```

```
620 FORY=21TO28:SET(X,Y):NEXT:Y=X=65:FORY=19TO28:SET(X,Y):NEXT:R
ETURN
```

PI-Network Attenuator.

```
630 GOSUB4230
640 PRINT@462," PI-NETWORK ATTENUATOR PROGRAM"
650 GOSUB4250
660 GOSUB790
670 PRINT@656,"ENTER DESIRED ATTENUATION <DB>":INPUTW
680 PRINT@721,"ENTER IMPEDANCE OF THE PAD":INPUTD
690 Y=1:X=10:Y=(.05*W)
700 R1=D*((X*Y+1)/(X*Y-1))
710 R2=(D*R1*(X*Y-1))/(D+R1)
720 PRINT@512,CHR$(31)
730 PRINT@576,"R 1 ="USINGR$;R1;
740 PRINT@640,"R 2 ="USINGR$;R2;
750 PRINT@612,"<Z> IN = <Z> OUT =":D;:PRINT@676,"THE ATTENUATION
=";W;"DB";
760 GOSUB4260
770 IFC$="N"THEN170ELSEIFC$="Y"THEN780ELSE760
780 GOSUB790:GOTO670
790 CLS
800 PRINT@22,"PI-NETWORK CIRCUIT"
810 PRINT@256,"IN":PRINT@95,"R2";:PRINT@270,"R 1";
820 PRINT@304,"R 1 OUT"
830 Y=4:FORX=0TO59:SET(X,Y):NEXT:X:FORX=68TO127:SET(X,Y):NEXT
840 Y=22:FORX=0TO127:SET(X,Y):NEXT:X
850 X=30:FORY=4TO11:SET(X,Y):NEXT:Y:FORY=15TO22:SET(X,Y):NEXT
860 X=98:FORY=4TO11:SET(X,Y):NEXT:Y:FORY=15TO22:SET(X,Y):NEXT:RET
URN
```

Active High/Low Pass filter design.

```
870 CLS
880 GOTO1090
890 CLS
900 PRINT@102,"XXXX";
910 PRINT@166,"X XXX";
920 PRINT@230,"X + XXX";
930 PRINT@294,"X XXX";
940 PRINT@358,"X XXX";
950 PRINT@422,"X - XXX";
960 PRINT@486,"X XXX";
970 PRINT@550,"XXXX OP-AMP";
980 Y=1:FORX=37TO45:SET(X,Y):NEXT:X
990 FORX=55TO113:SET(X,Y):NEXT:X
1000 X=37:FORY=1TO10:SET(X,Y):NEXT:Y
1010 X=113:FORY=1TO28:SET(X,Y):NEXT:Y
1020 Y=10:FORX=0TO21:SET(X,Y):NEXT:X
1030 FORX=28TO45:SET(X,Y):NEXT:X
1040 FORX=52TO75:SET(X,Y):NEXT:X
1050 X=60:FORY=11TO17:SET(X,Y):NEXT:Y:FORY=21TO30:SET(X,Y):NEXT
1060 Y=15:FORX=102TO127:SET(X,Y):NEXT:X
1070 Y=18:FORX=69TO75:SET(X,Y):NEXT:X:Y=69:FORY=18TO28:SET(X,Y):N
EXT:Y
1080 Y=28:FORX=69TO113:SET(X,Y):NEXT:X:Y=30:FORX=0TO127:SET(X,Y):
NEXT:RETURN
1090 GOSUB4230
1100 PRINT@452,"THIS IS A HI-PASS/LOW-PASS ACTIVE FILTER DESIGN
PROGRAM"
1110 GOSUB4250
1120 CLS:PRINT@455," FOR HI-PASS ENTER 1, FOR LOW PASS ENTER
```

```

2";:GOSUB4210
1130 CLS:ONCCGOTO1140,1170:GOTO1120
1140 PRINT"HI-PASS DESIGN"
1150 PRINT@24,"R 1";:PRINT@204,"C";:PRINT@216,"C";:PRINT@413,"R
2";
1160 GOSUB900:GOTO1320
1170 PRINT"LOW-PASS DESIGN"
1180 PRINT@24,"C 1";:PRINT@204,"R";:PRINT@216,"R";:PRINT@413,"C
2";
1190 GOSUB900
1200 PRINT@704,"ENTER 3DB CUTOFF FREQUENCY <HZ>";:INPUTW
1210 PRINT@768,"ENTER VALUE OF C2 <UF>";:INPUTA
1220 Z=A*1E-06
1230 X=6.28319*W
1240 R=.707/(X*Z):B=2*A
1250 PRINT@704,CHR$(31)
1260 PRINT@704,"C 1 ="USINGC0$;B;:PRINT" UF";
1270 PRINT@720," C 2 ="USINGC0$;A;:PRINT" UF";
1280 PRINT@737," R ="USINGR$;R;:PRINT" OHMS";
1290 PRINT@782,"- AT A FREQUENCY OF";W;"HZ. -"
1300 GOSUB4260
1310 IFC$="Y"THEN1120ELSEIFC$="N"THEN170ELSE1300
1320 PRINT@704,"ENTER 3DB CUTOFF FREQUENCY <HZ>";:INPUTW
1330 PRINT@768,"ENTER VALUE OF C <UF>";:INPUTA
1340 Z=A*1E-06:X=6.28319*W
1350 R=.707/(Z*X):B=2*R
1360 PRINT@704,CHR$(31)
1370 PRINT@704,"C ="USINGC0$;A;:PRINT" UF";
1380 PRINT@720,"R 1 ="USINGR$;R;:PRINT" OHMS";
1390 PRINT@741,"R 2 ="USINGR$;B;:PRINT" OHMS";
1400 PRINT@782,"- AT A FREQUENCY OF";W;"HZ. -"
1410 GOTO1300

```

Passive High/Low Pass filter design.

```

1420 GOSUB4230
1430 PRINT@453,"THIS PROGRAM IS FOR PASSIVE HIGH/LOW PASS FILTER
S."
1440 GOSUB4250
1450 CLS:PRINT@456,"ENTER A <1> FOR HIGHPASS OR A <2> FOR LOWPAS
S";:GOSUB4210
1460 IFCC=2ORCC=<0THEN1450
1470 GOSUB2050
1480 IFCC=1THEN1780
1490 PRINT@448,"ENTER THE INPUT/OUTPUT IMPEDANCE IN OHMS":INPUTR
1500 PRINT@512,"ENTER THE 3DB FREQUENCY IN MHZ":INPUTF
1510 F=F*1E06
1520 X=R/(6.2832*F)
1530 Y=1/(6.2832*F*R)
1540 H=X:I=Y
1550 PRINT@576,"ENTER THE NUMBER OF POLES (2,3,4 OR 5)":INPUTQ
1560 IFQ<2GOTO1550
1570 IFQ>6GOTO1550
1580 PRINT@384,CHR$(31)

1590 X=H:Y=I
1600 ONQGOTO1550,1610,1640,1670,1720
1610 PRINT@512,"ELEMENT 1 ="(Y*1.414);F$
1620 PRINT@576,"ELEMENT 2 ="X*1.414;H$;
1630 PRINT@704,"AT A FREQUENCY OF";F/1E6;"MHZ":GOTO1760
1640 PRINT@512,"ELEMENT 1 = ELEMENT 3 ="Y;F$
1650 PRINT@576,"ELEMENT 2 ="2*X;H$

```

```

1660 PRINT@704,"AT A FREQUENCY OF";F/1E6;"MHZ":GOTO1760
1670 PRINT@512,"ELEMENT 1 ="Y*.7654;F$
1680 PRINT@576,"ELEMENT 2 ="X*1.848;H$
1690 PRINT@640,"ELEMENT 3 ="Y*1.848;F$

1700 PRINT@704,"ELEMENT 4 ="X*.7654;H$
1710 PRINT@768,"AT A FREQUENCY OF";F/1E6;"MHZ":GOTO1760
1720 PRINT@512,"ELEMENT 1 = ELEMENT 5 ="Y*.618;F$
1730 PRINT@576,"ELEMENT 2 = ELEMENT 4 ="X*1.618;H$
1740 PRINT@640,"ELEMENT 3 ="Y*2;F$
1750 PRINT@768,"AT A FREQUENCY OF";F/1E6;"MHZ"
1760 GOSUB4260
1770 IFC$="N"THEN170ELSEIFC$="Y"THEN1450ELSE1760
1780 PRINT@448,"ENTER THE INPUT/OUTPUT IMPEDANCE IN OHMS":INPUTR
1790 PRINT@512,"ENTER THE 3DB FREQUENCY IN MHZ":INPUTF
1800 F=F*1E06
1810 X=R/(6.2832*F)
1820 Y=1/(6.2832*F*R)
1830 H=X:I=Y
1840 PRINT@576,"ENTER THE NUMBER OF POLES (2,3,4 OR 5)":INPUTQ
1850 IFQ<2GOTO1840
1860 IFQ>6GOTO1840
1870 PRINT@384,CHR$(31)
1880 X=H:Y=I
1890 ONQGOTO1840,1900,1930,1960,2010
1900 PRINT@512,"ELEMENT 1 ="(1.414*X);H$
1910 PRINT@576,"ELEMENT 2 ="(1.414*Y);F$
1920 PRINT@704,"AT A FREQUENCY OF";F/1E6;"MHZ":GOTO1760
1930 PRINT@512,"ELEMENT 1 = ELEMENT 3 ="X;H$
1940 PRINT@576,"ELEMENT 2 ="2*Y;F$
1950 PRINT@704,"AT A FREQUENCY OF";F/1E6;"MHZ":GOTO1760
1960 PRINT@512,"ELEMENT 1 ="X*.7654;H$
1970 PRINT@576,"ELEMENT 2 ="Y*1.848;F$
1980 PRINT@640,"ELEMENT 3 ="X*1.848;H$
1990 PRINT@704,"ELEMENT 4 ="Y*.7654;F$
2000 PRINT@768,"AT A FREQUENCY OF";F/1E6;"MHZ":GOTO1760
2010 PRINT@512,"ELEMENT 1 = ELEMENT 5 ="X*.618;H$
2020 PRINT@576,"ELEMENT 2 = ELEMENT 4 ="Y*1.618;F$
2030 PRINT@640,"ELEMENT 3 ="X*2;H$
2040 PRINT@768,"AT A FREQUENCY OF";F/1E6;"MHZ":GOTO1760
2050 CLS:PRINT@86,"2
2060 PRINT@105,"4"
2070 PRINT@192,"INPUT"
2080 PRINT@203,"1":PRINT@223,"3";:PRINT@243,"5";:PRINT@249,"ETC:
";
2090 Y=4:FORX=1TO41:SET(X,Y):NEXTX:FORX=48TO79:SET(X,Y):NEXT:FOR
X=86TO121:SET(X,Y):NEXT
2100 Y=17:FORX=1TO121:SET(X,Y):NEXTX
2110 X=23:GOSUB2120:X=63:GOSUB2120:X=103:GOSUB2120
2120 FORY=4TO8:SET(X,Y):NEXTY:FORY=12TO17:SET(X,Y):NEXT:RETURN

```

Band-Pass filter.

```

2130 GOSUB4230
2140 PRINT@392,"THIS PROGRAM IS FOR DESIGNING BAND-PASS FILTERS.
"
2150 PRINT@519,"THE FILTER'S BAND WIDTH SHOULD BE BETWEEN 15 & 5
0 %"
2160 PRINT@589,"OF THE CENTER FREQUENCY OF THE FILTER."
2170 GOSUB4250
2180 GOSUB2490:PRINT@384,

```

```

2190 INPUT"ENTER BAND WIDTH MHZ";F
2200 INPUT"ENTER THE CENTER FREQUENCY OF THE FILTER MHZ";C
2210 C=C*1E06:F=F*1E06
2220 INPUT"ENTER THE INPUT/OUTPUT IMPEDANCE";R
2230 INPUT"ENTER THE NUMBER OF POLES <3,4 OR 5>";N
2240 IFN<3ORN>=6THEN2230
2250 PRINT@320,CHR$(31)
2260 X=R/(6.2832*F)
2270 Y=1/(6.2832*R*F):D=CC2
2280 ONNGOTO2230,2230,2290,2330,2410
2290 PRINT@320,"ELEMENT 1 = ELEMENT 3 =" ;Y;F$
2300 PRINT"BOTH HAVE A PARALLEL INDUCTOR OF";.0253/(D*Y);H$
2310 PRINT"ELEMENT 2 =" ;2*X;H$
2320 PRINT"WITH A SERIES CAPACITOR OF";.0253/(D*(2*X));F$:GOTO24
70
2330 PRINT@320,"ELEMENT 1 =" ;.7654*Y;F$
2340 PRINT"WITH A PARALLEL INDUCTOR OF";.0253/(D*(Y*.7654));H$
2350 PRINT"ELEMENT 2 =" ;1.848*X;H$
2360 PRINT"WITH A SERIES CAPACITOR OF";.0253/(D*(1.848*X));F$
2370 PRINT"ELEMENT 3 =" ;1.848*Y;F$
2380 PRINT"WITH A PARALLEL INDUCTOR OF";.0253/(D*(1.848*Y));H$
2390 PRINT"ELEMENT 4 =" ;.7654*X;H$
2400 PRINT"WITH A SERIES CAPACITOR OF";.0253/(D*(.7654*X));F$:GO
T02470
2410 PRINT@320,"ELEMENT 1 = ELEMENT 5 =" ;.618*Y;F$
2420 PRINT"BOTH WITH A PARALLEL INDUCTOR OF";.0253/(D*(.618*Y));
H$
2430 PRINT"ELEMENT 2 = ELEMENT 4 =" ;1.618*X;H$
2440 PRINT"BOTH WITH A SERIES CAPACITOR OF";.0253/(D*(1.618*X));F
$
2450 PRINT"ELEMENT 3 =" ;2*Y;F$
2460 PRINT"WITH A PARALLEL INDUCTOR OF";.0253/(D*(2*Y));H$
2470 GOSUB4260
2480 IFC$="N"THEN170ELSEIFC$="Y"THEN2180ELSE2470
2490 CLS
2500 PRINT@18,"C    2"
2510 PRINT@40,"C    4"
2520 PRINT@128,"IN:   L    1"
2530 PRINT@157,"L    3"
2540 PRINT@179,"L    5    OUT"
2550 Y=1:FORX=0T033:SET(X,Y):NEXTX:FORX=40T045:SET(X,Y):NEXT
2560 FORX=52T077:SET(X,Y):NEXTX:FORX=84T089:SET(X,Y):NEXT
2570 FORX=96T0127:SET(X,Y):NEXTX
2580 Y=13:FORX=0T0127:SET(X,Y):NEXTX
2590 X=16:GOSUB2600:X=27:GOSUB2600:X=58:GOSUB2600:X=69:GOSUB2600
:X=102:GOSUB2600:X=113:GOSUB2600
2600 FORY=1T05:SET(X,Y):NEXTY:FORY=9T013:SET(X,Y):NEXT:RETURN

```

Coil design.

```

2610 CLS:GOSUB4230
2620 PRINT@332,"THIS PROGRAM IS USED FOR DESIGNING COILS"
2630 PRINT@461,"AFTER THIS PAGE CLEARS, ENTER EITHER -"
2640 PRINT@516,"A <1> KNOWING THE RADIUS, LENGTH & THE # OF TURN
S OR A <2>"
2650 PRINT@580,"KNOWING THE RADIUS, # OF TURNS & THE L <UH>."
2660 GOSUB4250
2670 CLS:GOSUB4210
2680 IFCC=2ORCC=0THEN2610

```

```

2690 IFCC=2THEN2810
2700 PRINT@192,;INPUT"ENTER THE COIL RADIUS <INCHES>";A
2710 INPUT"ENTER THE COIL LENGTH <INCHES>";B
2720 INPUT"ENTER THE NUMBER OF TURNS";N:CLS
2730 R=A/2:S=N/2:T=R*S
2740 L=T/((9*A)+(10*B))
2750 PRINT@192,
2760 PRINT"THE COIL ="USINGR$;L;PRINT" UH."
2770 PRINT:PRINT"USING";N;" TURNS OF WIRE."
2780 PRINT"WITH A LENGTH OF";B;" INCHES."
2790 PRINT"AND A RADIUS OF";A;" INCHES."
2800 GOTO2920
2810 PRINT@192,
2820 INPUT"ENTER THE COIL RADIUS <INCHES>";A
2830 INPUT"ENTER THE COIL LENGTH <INCHES>";B
2840 INPUT"ENTER THE INDUCTANCE OF THE COIL <UH>";L
2850 PRINT@192,
2860 P=((9*A)+(10*B)):R=A/2
2870 X=(L*P)/R:Y=SQR(X)
2880 PRINT"THE NUMBER OF TURNS NEEDED IS "USINGR$;Y
2890 PRINT:PRINT"WITH A COIL RADIUS OF";A;" INCHES"
2900 PRINT"AND A LENGTH OF";B;" INCHES"
2910 PRINT"WITH AN INDUCTANCE OF";L;"UH."
2920 GOSUB4260
2930 IFC$="Y"THEN2610ELSEIFC$="N"THEN170ELSE2920

```

Impedance-Matching Network design I.

```

2940 GOSUB4230
2950 PRINT@323,"THIS PROGRAM IS FOR DESIGNING IMPEDANCE-MATCHING
NETWORKS"
2960 PRINT@451,"THIS NETWORK IS THE CLASSIC <T> TYPE. IT CAN BE
USED FOR "
2970 PRINT@513,"MATCHING <R1> AMOUNTS WHICH ARE ABOVE & BELOW TH
E VALUE OF <RL>"
2980 GOSUB4250
2990 GOSUB3210
3000 PRINT:PRINT
3010 INPUT"INPUT <R1>";R
3020 INPUT"INPUT <CO> IN (PICO-FARADS)";C
3030 INPUT"INPUT THE LOADED <Q> OF THE CIRCUIT";Q
3040 INPUT"INPUT THE OPERATING FREQUENCY <MHZ.>";F
3050 INPUT"INPUT <RL>";S
3060 PRINT@448,"":PRINTCHR$(31)
3070 C=C*1E-12:F=F*1E06
3080 P=1/(6.28*F*C)
3090 H=(Q*R)+P
3100 A=R*(Q/2+1)
3110 Y=SQR((A/S)-1)
3120 J=A/(Q+Y):I=S*Y
3130 PRINT"<< ALL VALUES IN OHMS >>"
3140 PRINT"X <L1> ="USINGR$;H
3150 PRINT"X <L2> ="USINGR$;I
3160 PRINT"X <C1> ="USINGR$;J
3170 PRINT@673,"AT A FREQUENCY OF";F/1E06;"MHZ.";
3180 GOSUB4260
3190 IFC$="Y"THEN3200ELSEIFC$="N"THEN170ELSE3180
3200 GOSUB3210:GOTO3000
3210 CLS
3220 PRINT@23,"MATCHING NETWORK"

```



```

3230 PRINT@70,"C 0"
3240 PRINT@86,"L 1"
3250 PRINT@111,"L 2"
3260 PRINT@192,"R 1"
3270 PRINT@226,"C 1"
3280 PRINT@252,"R L"
3290 Y=4:FORX=3T010:SET(X,Y):NEXTX:FORX=19T042:SET(X,Y):NEXT
3300 FORX=51T092:SET(X,Y):NEXTX:FORX=101T0123:SET(X,Y):NEXT
3310 Y=16:FORX=3T0123:SET(X,Y):NEXTX
3320 X=3:GOSUB3370:X=71:GOSUB3370:X=123:GOSUB3370
3330 Y=1:FORX=0T023STEP3:SET(X,Y):NEXTX
3340 X=23:FORY=1T019STEP2:SET(X,Y):NEXTY
3350 Y=19:FORX=23T00STEP-3:SET(X,Y):NEXTX
3360 PRINT@397,"J---DEVICE TO BE MATCHED":RETURN
3370 FORY=4T08:SET(X,Y):NEXTY:FORY=12T016:SET(X,Y):NEXT:RETURN

```

Impedance-Matching Network design II.

```

3380 GOSUB4230
3390 PRINT@387,"THIS PROGRAM IS FOR DESIGNING IMPEDANCE-MATCHING
NETWORKS"
3400 PRINT@513,"THIS NETWORK IS BEST SUITED TO MATCHING COLLECTO
R IMPEDANCES"
3410 PRINT@577,"OF LESS THAN 50 OHMS TO LOAD IMPEDANCES OF 50 OH
MS OR GREATER."
3420 GOSUB4250
3430 GOSUB3640
3440 PRINT@512,":INPUT"INPUT <R1>";R
3450 INPUT"INPUT <C0> IN (PICO-FARADS)";C
3460 INPUT"INPUT THE LOADED <Q> OF THE CIRCUIT";Q
3470 INPUT"INPUT THE OPERATING FREQUENCY <MHZ.>";F
3480 INPUT"INPUT <RL>";S
3490 PRINT@384,CHR$(31)
3500 C=C*1E-12:F=F*1E06
3510 P=1/(6.28*F*C)
3520 H=(Q*R)+P
3530 B=R*((Q*Q)+1)
3540 X=((R*((Q*Q)+1))/S)-1
3550 Y=SQR(X):J=Y*S
3560 I=B/(Q+Y)
3570 PRINT:PRINT"ALL VALUES IN OHMS"
3580 PRINT"X <L1> ="USINGR$;H
3590 PRINT"X <C1> ="USINGR$;I
3600 PRINT"X <C2> ="USINGR$;J
3610 PRINT@675,"AT A FREQUENCY OF";F/1E06;"MHZ.";
3620 GOSUB4260
3630 IFC$="N"THEN170ELSEIFC$="Y"THEN3430ELSE3620
3640 CLS
3650 PRINT@23,"MATCHING NETWORK"
3660 PRINT@70,"C 0"
3670 PRINT@86,"L 1"
3680 PRINT@111,"C 2"
3690 PRINT@192,"R 1"
3700 PRINT@226,"C 1"

3710 PRINT@252,"R L"
3720 Y=4:FORX=3T010:SET(X,Y):NEXTX:FORX=19T042:SET(X,Y):NEXT
3730 FORX=51T092:SET(X,Y):NEXTX:FORX=101T0123:SET(X,Y):NEXT
3740 Y=16:FORX=3T0123:SET(X,Y):NEXTX
3750 X=3:GOSUB3810:X=71:GOSUB3810
3760 X=123:GOSUB3810
3770 Y=1:FORX=0T023STEP3:SET(X,Y):NEXTX

```

```

3780 X=23:FORY=1T019STEP2:SET(X,Y):NEXTY
3790 Y=19:FORX=23T00STEP-3:SET(X,Y):NEXTX
3800 PRINT@397,"J--- DEVICE TO BE MATCHED":RETURN
3810 FORY=4T08:SET(X,Y):NEXTY:FORY=12T016:SET(X,Y):NEXT:RETURN

```

Impedance-Matching Network Design-III.

```

3820 GOSUB4230:GOTO3990
3830 PRINT:PRINT:INPUT"INPUT <R1>";T
3840 INPUT"INPUT <RL>";R
3850 INPUT"INPUT THE LOADED <Q> OF THE CIRCUIT";Q
3860 PRINT@448,CHR$(31)
3870 F=F*1E06:C=T/Q
3880 X=(T/R)/((Q*Q)+1)-(T/R)
3890 D=R*SQR(X):V=Q*T
3900 U=T*(R/D)
3910 E=(V+U)/(Q*Q+1)
3920 PRINT"ALL VALUES IN OHMS":PRINT
3930 PRINT"X <C1> ="USINGR$;C
3940 PRINT"X <C2> ="USINGR$;D
3950 PRINT"X <L1> ="USINGR$;E
3960 GOSUB4260
3970 IFC$="N"THEN170ELSEIFC$="Y"THEN60SUB4030ELSE3960
3980 GOTO3830
3990 PRINT@387,"THIS PROGRAM IS FOR DESIGNING IMPEDANCE-MATCHING
NETWORKS"
4000 PRINT@516,"THE IMPEDENCE OF <R1> SHOULD BE LARGER THE THE L
OAD <RL>"
4010 PRINT@578,"IF NOT THE VALUE OF <L1> BECOMES ALMOST IMPOSSIB
LE TO REALIZE"
4020 GOSUB4250:GOSUB4030:GOTO3830
4030 CLS:PRINT:PRINT
4040 PRINT"MATCHING NETWORK"
4050 PRINT@94,"L 1"
4060 PRINT@192,"R 1"
4070 PRINT@205,"C 1"
4080 PRINT@239,"C 2"
4090 PRINT@252,"R L"
4100 Y=4:FORX=3T058:SET(X,Y):NEXTX
4110 FORX=67T0123:SET(X,Y):NEXTX
4120 Y=16:FORX=3T0123:SET(X,Y):NEXTX
4130 X=3:GOSUB4190:X=29:GOSUB4190
4140 X=97:GOSUB4190:X=123:GOSUB4190
4150 Y=1:FORX=0T015STEP3:SET(X,Y):NEXTX
4160 X=15:FORY=1T019STEP2:SET(X,Y):NEXTY
4170 Y=19:FORX=15T00STEP-3:SET(X,Y):NEXTX
4180 PRINT@393,"J---DEVICE TO BE MATCHED":RETURN
4190 FORY=4T08:SET(X,Y):NEXTY
4200 FORY=12T016:SET(X,Y):NEXTY:RETURN

```

Inkey\$ routine.

```

4210 C$=INKEY$:IFC$=""THEN4210
4220 CC=VAL(C$):RETURN
4230 PRINT:PRINTSTRING$(64,140):PRINT@896,"";
4240 PRINTSTRING$(64,140):RETURN
4250 PRINT@785,"TO CONTINUE, PRESS ANY KEY";GOSUB4210:RETURN
4260 PRINT@896,"DO YOU WISH A NEW DESIGN <Y/N>";GOSUB4210:RETUR
N

```

The most important book ever published for the Apple.

The most comprehensive description of Apple II firmware and hardware ever published — all in one place.

What's Where in the Apple?

- Guides you — with a numerical Atlas and an alphabetical Gazetteer — to over **2,000** memory locations of **PEEKs**, **POKEs**, and **CALLs**.
- Gives names and locations of various **Monitor**, **DOS**, **Integer BASIC**, and **Applesoft** routines — and tells you what they're used for.
- Helps BASIC users to speed up their programs.
- Enables assembly language programmers to simplify coding and interfacing.

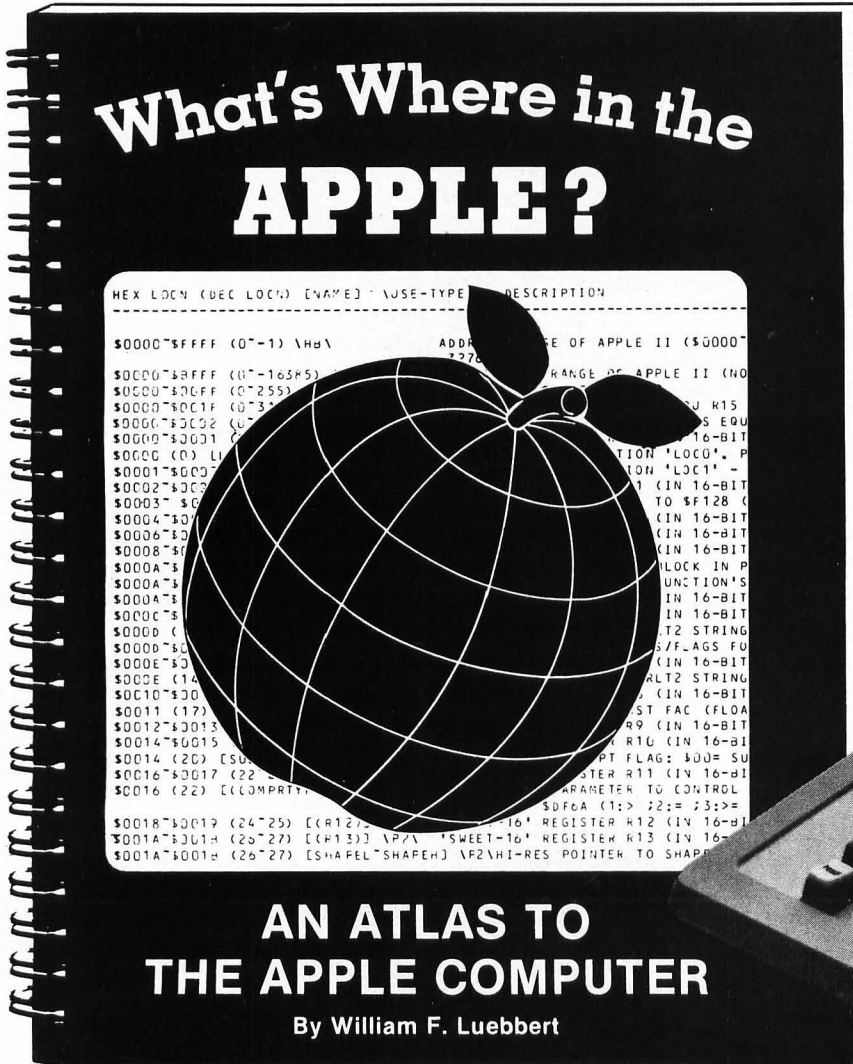
All Apple users will find this book helpful in understanding their machine, and essential for mastering it!

Ask for it at your computer store

128 pages, 8½ × 11 inches, cardstock cover, Wire-O binding.

\$14.95

ISBN: 0-938222-07-4



ORDER TOLL-FREE TODAY **800-227-1617** EXT. 564
(in California 800-772-3545 Ext. 564)

Yes! Please send me _____ copies of *What's Where in the Apple?* at \$14.95 each (in U.S. plus shipping).

Name _____

Address _____

City _____ State _____ Zip _____

Check for \$ _____ enclosed. (Add \$2.00 surface shipping for each copy.) Massachusetts residents add 5% sales tax.

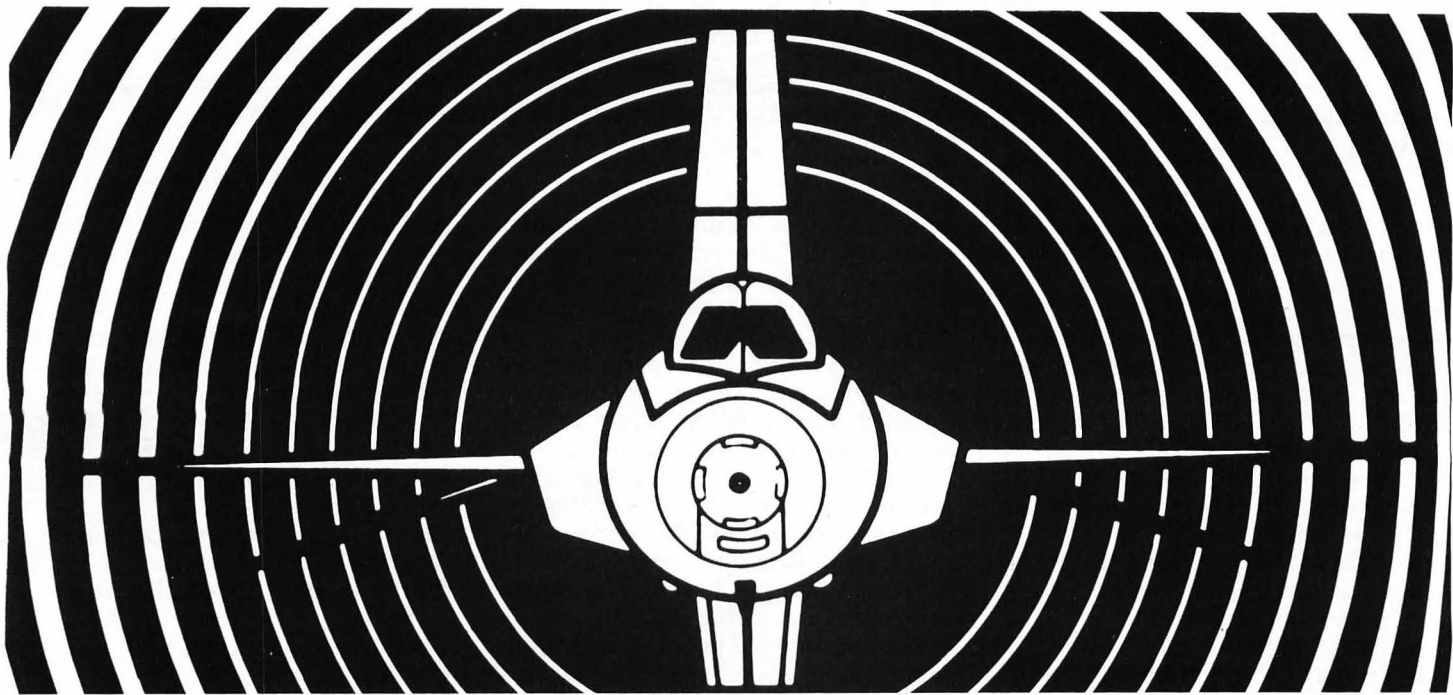
VISA MasterCard

Acct. # _____ Expires _____

Signature _____

MICRO INK, Inc., 34 Chelmsford Street, P.O. Box 6502, Chelmsford, MA 01824

AIRCRAFT



COMMANDER

BY WILLIAM J. EDMUNDS

Aircraft Commander is a Hi-Res flight instrument simulator program for an Apple II with Applesoft in ROM and 24K RAM.

This program puts you in the cockpit of a single-engine jet fighter. The flight instruments are displayed on the Hi-Res screen, with additional information concerning communications, fuel, and missile status in the text area at the bottom of the screen. You, as pilot, can practice take-off and landing, flight, and air-to-air intercept. There is no ground navigation or through-the-window view.

The Flight Instruments

At the top center is the Attitude Indicator, sometimes called the "artificial horizon." Because it represents the horizon and not the wings, it moves opposite to the motion of the aircraft. Entering a left bank, the horizon appears to turn right; if you pull back on the control stick and pitch up, the horizon appears to fall.

At the top left is the Airspeed Indicator, which gives indicated airspeed from about seventy knots up. At sea level, indicated airspeed is the same as true airspeed, which is displayed in the text area on the left. As altitude increases the air becomes thinner, and for the same true airspeed the indicated or effective airspeed is less. Airspeed translates into lift, and if airspeed drops below 90 knots you may stall. There is an audible warning when you are approaching a stall or doing something that can cause a stall.

Below the right-hand corner of the Airspeed Indicator is the Landing Gear Indicator. An "O" represents the gear being "out" or down, and an "I" means that it is "in" or up.

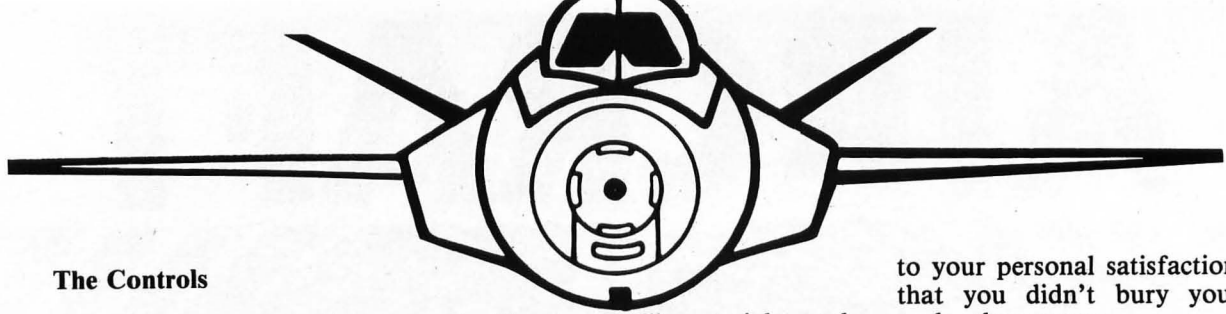
At the top right is the Altimeter. Each number represents 100 feet, with each full revolution being 1000 feet. The number of total revolutions is displayed in the text area on the right as ALT X1000.

At the lower left is the Compass. This pointer operates as a magnetic

compass, with the heading displayed below it as a number of degrees clockwise from north.

At the lower center, the Radar Screen displays range vertically, and relative azimuth horizontally. It has a range of ten miles, with the center vertical row of dots representing one-mile increments dead ahead. The other vertical rows each represent 15-degree increments port and starboard (left and right). The only thing which can be displayed on the Radar Screen are the targets, as small horizontal lines.

At the lower right is the Engine Tachometer. Jet engines idle at 50% RPM and develop maximum thrust at 100% RPM or Full Military Power. Jet fighters also have an afterburner which injects fuel into the exhaust port to increase exit velocity and thereby gain additional thrust. Afterburn is very inefficient and is used only for takeoff and where maximum acceleration is required. Afterburn is indicated on the tachometer by an RPM greater than 100%.



The Controls

The following is a summary of the controls; their use is illustrated more fully in the "Operation" section.

Paddle 0 controls bank or roll.

Paddle 1 controls pitch.

Right-arrow and left-arrow keys increase or decrease RPM by 1%. These keys set a flag to determine if RPM is increasing or decreasing; once the flag is set, the number keys can be used to make larger changes.

Number keys 1-9 are used for changing RPM, up to 9% at once.

Number key 0 toggles RPM to 100%, afterburn, or idle.

G raises or lowers landing gear. Gear can be used as a speed break.

A arms a missile.

F fires an armed missile.

C opens the communications channel.

Spacebar is used to acknowledge the first communications.

ESC ejects you from the aircraft (exits program).

Operation

1. Takeoff:

Starting at idle, press the right-arrow key. This sets the flag for increasing throttle position and moves the RPM to 51%.

Press 0 to increase RPM to 100%.

Press 0 again to engage the afterburner. At 100 knots start pitching up with Paddle 1. At approximately 120 knots liftoff will occur, and the altimeter will start to wind upward. Try to hold your pitch about 15 to 20 degrees nose-up.

Press G to raise the landing gear. The indicator will change from Out to In.

At around 200 knots press the left-arrow key to disengage the afterburner and set throttle flag for decreasing RPM. Continue your climb to the desired altitude and level off.

Press 9, reducing RPM to 91%.

Press 9 again, reducing RPM to 82%. Cruise.

2. Air-to-air intercept:

Once airborne you can simulate an air-to-air intercept. It will be your mission to intercept and shoot down a

target. The target flies straight and level, and does not try to evade or attack.

To start, type C. This opens the communication channel, and ground control will give you an absolute bearing to the target. Pressing the spacebar again will give target heading, range, speed, and altitude in thousands of feet. Pressing the spacebar once more will give you a continuously updated display of the target's range and its bearing relative to you.

When you have the target on your screen, arm a missile by pressing A. The range of the missiles is about five miles, and they can steer up to 15 degrees from the heading on which they are fired. Differences in altitude are not considered in computing radar or missile range, but when the missile is fired the difference between target elevation and aircraft pitch must be less than 15 degrees. The easiest way to achieve this is to fly at the same altitude as the target.

Steer to put the target on the center line of your radar screen. If the target is approaching head-on, you can fire once the target is about five miles out. If you are chasing the target you need to get within two or three miles depending on target speed. The missiles travel at 1200 knots or about twice the speed of the fastest target. When you are ready, press F to fire the missile.

Once the missile is fired, it can take up to 30 seconds to reach the target. At the point of contact a "HIT" will be displayed in the missile status, and the target will disappear. If the target was not in range then a "MIS" is displayed instead.

You have only two missiles, but landing the aircraft will load two new missiles and additional fuel if needed.

3. Landing:

Landing speed is about 100 knots. With gear lowered (using the G key), and pitch about 0 to 10 degrees nose-up, you will need 78% to 80% thrust. Your rate of descent will be displayed at touchdown. A good landing would be with wings level and at a descent rate of less than 400 feet per minute.

Landing is the most difficult part of flying an aircraft. However, a good landing is not essential to the enjoyment of the program. It just adds

to your personal satisfaction to know that you didn't bury your aircraft under the runway.

Variables

A: Missile-armed flag.

AA: Angle of attack.

AB: Landing gear flag.

ALT: Altitude.

AR: Rotation of altimeter pointer.

AX,AY: Coordinates of altimeter indicator center.

B: Bank.

C: Communication counter.

CB: Cosine of bank.

CP: Cosine of pitch.

CR: Compass pointer rotation.

D: Drag.

DP: Delta pitch (pitch difference).

F: Fuel.

F1: Fuel consumption.

G: Gravity.

GH,GV: Coordinates of attitude indicator center.

GX,GY: Horizon line x and y positions.

H: Heading.

IAS: Indicated airspeed.

L: Lift.

M: Missile number.

MF: Missile counter.

MH: Missile heading.

MR: Missile range.

MX,MY: Missile x and y starting position.

OS: Radar on-screen flag.

P: Pitch.

PI: Radians in a circle.

R: Temporary variable.

RB: Target relative bearing.

RG: Horizon line rotation.

SB: Sine of bank.

SP: Sine of pitch.

SR: Airspeed pointer rotation.

SX,SY: Target position on radar screen.

T: Thrust.

TAS: True airspeed.

TB: Target bearing.

TC: Thrust control flag.

TH: Target heading.

TR,TS,TU: Target range, speed, and altitude.

TT: Engine RPM in percent.

TX,TY: Target x and y range.

TW: The number 2.

U: The number 1.

VV: Vertical velocity.

X,Y: Temporary variables.

XI,YI: Target x and y increment.

Z: The number 0.

Set LOMEM to protect Hi-Res graphics display.

10 LOMEM: 16385: GOTO 3000

Beginning of main subroutine to update aircraft parameters. Read paddles and compute pitch and bank angles.

100 DP = (PDL (U) - 90) / 1000:P
= P + DP * CB:CP = COS (P)

110 IF SGN (CP) < > 1 THEN X =
TAN (B):Y = ATM (X):X = PI
/ TW - TW * Y:B = B + X:P =
SGN (P) * PI / TW - P:H = H
+ X:CP = COS (P)

120 SP = SIN (P):X = (PDL (Z) -
PB) / PB

130 B = B + X ^ TW * SGN (X) + D
P * SP * SB:SB = SIN (B):CB
= COS (B)

140 IF ABS (B) > PI / TW THEN B
= (ABS (B) - PI) * SGN (B
)

150 IF ALT = Z THEN B = Z: IF TA
S < PB THEN P = Z:SB = Z:SP =
Z:CP = Z:CB = Z

Update attitude indicator.

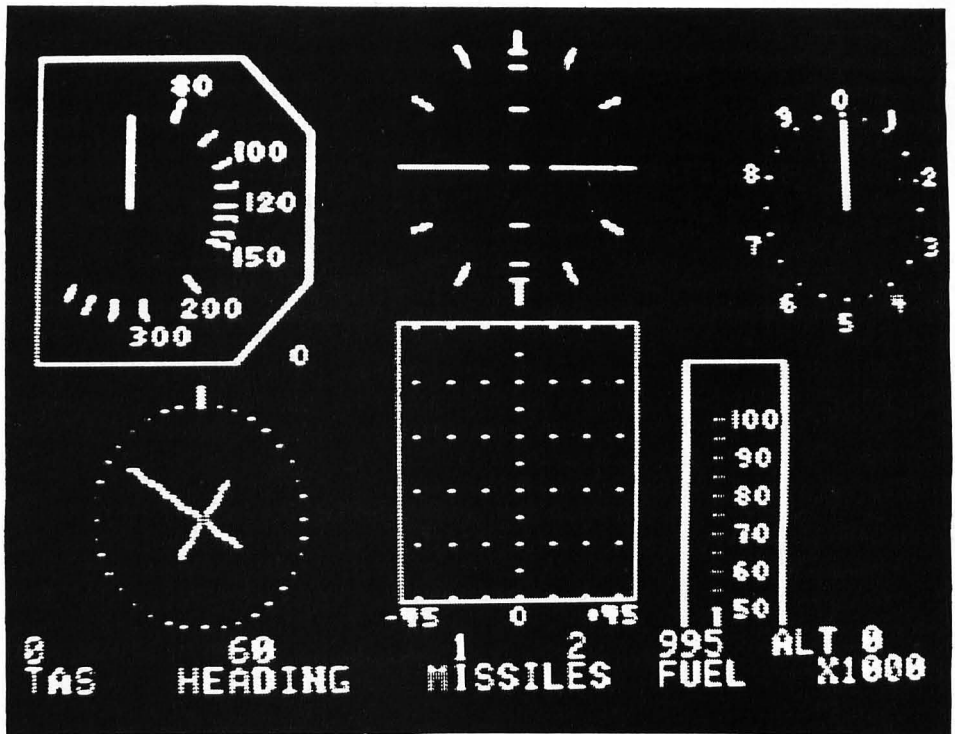
200 X = 6H + 30 * SP * SB
210 Y = 6V + 30 * SP * CB
220 R = PB - B * 64 / PI
230 ROT= R: SCALE= 10
240 XDRAW 11 AT X,Y
250 ROT= RG
260 XDRAW 11 AT 6X,6Y
270 6X = X:6Y = Y:RG = R

Compute heading.

300 H = H + SB * (DP + .05236)
320 IF H > PI THEN H = H - PI
325 IF H < Z THEN H = H + PI
330 HTAB 9: PRINT INT (H * 360 /
PI); " ";

Compute true air speed and indicated air speed.

400 TAS = TAS - D + SP * 6 * (ALT
> Z) + T * SQRT (IAS / TAS)
410 IF TAS < U OR (T < U AND TAS



< TW) THEN TAS = U
415 HTAB 1: PRINT INT (TAS * 0.
6); " " ;
420 IAS = TAS * 18500 / (ALT + 18
500)
430 IF ALT > 30100 THEN IAS = IA
S * 20800 / (ALT - 9250)

Compute altitude.

600 L = (IAS * .02) ^ TW
605 AA = 33 / (IAS + 21) + DP / T
W
610 D = AA * L * (AB + .6)
625 IF AA > .19 AND ALT > Z THEN
X = - 16336:Y = PEEK (X) +
PEEK (X) + PEEK (X) + PEEK
(X)
630 VV = L * CB * CP + 6 + TAS *
SIN (P - AA)
635 IF ALT > Z AND ALT + VV < =
Z THEN GOSUB 2600
637 IF ALT = Z AND VV > Z THEN HOME
: VTAB 21
640 ALT = ALT + VV
650 IF ALT < Z THEN ALT = Z: IF
P < Z THEN P = Z
670 IF IAS < PB AND ALT > Z THEN
P = P - (P + 1.5) / 5

Update altimeter.

800 X = ALT / 1000
810 R = INT ((X - INT (X)) * 64
)

820 IF R < > AR THEN ROT= AR: SC
ALE= 24: XDRAW 12 AT AX,AY:
ROT= R: XDRAW 12 AT AX,AY:AR = R
830 HTAB 36: PRINT INT (X); " " ;

Decrease fuel and check for
keypress.

900 F = F - F1: IF PEEK (- 1638
4) > PB THEN GOSUB 2000
990 RETURN

Beginning of main program loop.
Update compass.

1000 GOSUB 100
1010 R = PB - INT (H * 64 / PI):
IF CR < > R THEN ROT= CR:
SCALE= 12: XDRAW 13 AT 50,1
30: ROT= R: XDRAW 13 AT 50,1
30:CR = R
1015 IF C = 3 THEN GOSUB 2500
1020 GOSUB 100
1030 R = Z: IF IAS > 122 THEN R =
INT (LOG (IAS / 20 - 5) *
10)

Update air speed indicator.

1040 IF R < > SR THEN ROT= SR:
SCALE= 24: XDRAW 12 AT 30,5
0: ROT= R: XDRAW 12 AT 30,5
:SR = R
1050 HTAB 27: PRINT INT (F); " "
;

Compute target position and update radar.

```
1100 IF NOT C THEN 1000
1110 GOSUB 100
1120 TX = TX + XI - TAS * COS (H
) / 2000; TY = TY + YI - TAS *
SIN (H) / 2000; IF TX = Z THEN
TX = U / PB
1130 TR = SQR (TX ^ 2 + TY ^ 2)
1140 TB = ATN (TY / TX); IF TX <
Z THEN TB = TB + PI / TW
1150 IF TB < Z THEN TB = TB + PI

1155 GOSUB 100
1160 ROT= 16; SCALE= 5; IF OS THEN
XDRAW 12 AT SX,SY; OS = Z
1170 X = TB - H; IF ABS (X) = <
PI / TW THEN RB = X
1180 IF ABS (X) > PI / TW THEN
RB = ( ABS (X) - PI) * SGM
(X)
1190 IF TR < 10 AND ABS (RB) <
.78 THEN OS = U; SX = 138 + R
B * 240 / PI; SY = 151 - TR *
7; XDRAW 12 AT SX,SY
1200 IF M > TW THEN IF NOT OS THEN
C = 0
```

Compute missile position and check for hit.

```
1300 IF NOT MF THEN 1000
1310 MF = MF + U; IF MF > 6 THEN
MF = Z; HTAB 12 + M * 5; PRINT
"MISS"; GOSUB 2650; GOTO 100
0
1320 GOSUB 100
1330 R = MR:MR = SQR ((MX + MF *
(XI - COS (MH))) ^ 2 + (MY
+ MF * (YI - SIN (MH))) ^
2)
1340 IF MR > R THEN X = ABS (MH
- ATN ((MY + MF * YI) / (M
X + MF * XI))); MF = Z; IF X <
.26 OR ABS (PI / 2 - X) < .
26 OR ABS (PI - X) < .26 THEN
HTAB 12 + M * 5; PRINT "HIT
"; C = 0; GOSUB 2650; IF OS THEN
ROT= 16; SCALE= 5; XDRAW 12
AT SX,SY; OS = 0
1400 GOTO 1000
```

Interpret keypress.

```
2000 X = PEEK (- 16384); POKE -
```

```
16368,0: IF X = 155 THEN TEXT
: HOME : END
2100 IF X = 136 THEN TC = - 1: GOTO
2300
2110 IF X = 149 THEN TC = 1: GOTO
2300
2115 IF X = 176 AND TT = 100 AND
TC = 1 THEN TT = 110; HCOLOR=
3: HPLLOT 200,105: GOTO 2360
2120 IF X = 176 THEN TT = 100 *
TC: GOTO 2300
2130 IF X > 176 AND X < 186 THEN
TT = TT + (X - 177) * TC: GOTO
2300
2200 IF X = 160 THEN HOME : VTAB
21: IF C THEN 2500
2210 IF X = 193 THEN IF NOT A AND
ALT > 0 AND M < 2 THEN A = 1
:MF = 0;M = M + 1; FLASH : HTAB
13 + M * 5: PRINT M;: NORMAL
2220 IF X = 199 THEN ROT= 0: SCALE=
1: XDRAW 2 - AB AT 80,90;AB =
NOT AB: XDRAW 2 - AB AT 80,
90
2230 IF X = 195 THEN GOSUB 2500
2240 IF X = 198 THEN IF A AND C
THEN MF = 1;A = 0: HTAB 13 +
M * 5: PRINT " ";MX = TX:MY
= TY:MR = TR;MH = H: IF ABS
(P - ATN ((TU / 6 - ALT / 6
000) / TR)) > 0.26 THEN MF =
7
2290 RETURN
```

Update tachometer and compute thrust and fuel consumption.

```
2300 TT = TT + TC
2310 IF TT < 50 THEN TT = 50
2320 IF TT > 100 THEN TT = 100
2330 IF F < 1 THEN TT = 41;TC =
- 1
2340 IF TC > = Z THEN HCOLOR=
3: HPLLOT 200,155
2350 IF TC < Z THEN HCOLOR= 0: HPLLOT
200,95
2360 HPLLOT TO 200,205 - TT
2370 T = (TT - 50) / 5
2380 F1 = ((TT - 40) / 50) ^ 2 /
4
2390 RETURN
```

Communications.

```
2500 C = C + U; HOME : INVERSE : IF
M > TW THEN C = 5
2505 ON C GOSUB 2520,2540,2510,2
550,2590
```

```
2510 NORMAL : VTAB 21: RETURN
2520 X = INT ( RND (1) * 3 + 1):
ON X GOSUB 2700,2750,2800: GOSUB
2900
2530 PRINT C;"THIS IS GROUND CON
TROL, I HAVE"
2535 PRINT "A TARGET BEARING "; FN
Q(TB);" DEGREES";: RETURN
2540 PRINT "TARGET HEADING IS ";
FN Q(TH);" RANGE "; INT (T
R + 0.5);" MILES"
2545 PRINT "SPEED ";TS;" KNOTS A
T ALT. ";TU;" THOUSAND FT";:
RETURN
2550 PRINT C;"TARGET RANGE IS ";
INT (TR + 0.5);" MILES,"
2555 PRINT "RELATIVE BEARING "; ABS
(FN Q(RB));" DEGREES ";
2560 X = .087; IF RB < - X THEN
PRINT "PORT";
2565 IF RB > X THEN PRINT "STAR
BOARD";
2570 C = 3: RETURN
2590 PRINT : PRINT C;"RETURN TO
BASE.";:C = 0: IF OS THEN C =
4
2595 RETURN
```

Landing check; crash routine.

```
2600 X = ABS ( INT (VV * 30)): HOME
2605 IF X > 600 OR ABS (B) > .2
8 OR TAS > 300 OR NOT AB THEN
FLASH : PRINT " C R A S H
": NORMAL : FOR TT = 1 TO 4
9:Y = PEEK (- 16336): NEXT
:TC = - 1: GOSUB 2350;TA =
1
2607 IF NOT AB THEN PRINT "GEA
R NOT DOWN.";: GOTO 2620
2608 IF ABS (B) > .28 THEN PRINT
"UNUSUAL ATTITUDE";: GOTO 26
20
2610 PRINT "RATE OF DESCENT ";X;
" FT./MIN.";
2620 VTAB 21: HTAB 17
2625 PRINT " 1 2 ";M = 0:A =
0
2630 IF F < 300 THEN F = F + 300
2640 RETURN
2650 IF M = 2 THEN M = 3
2690 RETURN
Target generator
2700 GOSUB 2850;TB = R
2710 R = TB + PI / 2: GOSUB 2860:
TH = R
2720 TR = 15 + RND (1) * 10
2730 TU = INT (10 + INT ( RND (
1) * 3) * 2.5)
```

```

2740 RETURN
2750 GOSUB 2850:TB = R:TH = R
2760 TR = 11 + RND (1) * 5
2770 TU = INT (5 + INT ( RND (1)
) * 9) * 2.5)
2780 RETURN
2800 X = SGN ( RND (1) - 0.5)
2810 R = H + RND (1) * X: GOSUB
2860:TB = R
2820 R = H + PI / 2 + PI / 4 * X:
GOSUB 2860:TH = R
2830 TR = 12 + RND (1) * 10
2840 TU = INT (5 + INT ( RND (1)
) * 11) * 2.5)
2845 RETURN
2850 R = H + RND (1) * 3 - 1.5
2860 IF R > PI THEN R = R - PI
2870 IF R < .175 THEN R = R + PI

2880 RETURN
2900 TX = COS (TB) * TR:TY = SIN
(TB) * TR
2910 Y = SQR (TU) * 15
2930 XI = COS (TH) * Y / 200:YI =
SIN (TH) * Y / 200
2940 TS = INT (Y * 0.6) * 10
2950 DEF FN Q(X) = INT ((X + 0
.087) * 36 / PI) * 10
2960 RETURN
Initialization
3000 HOME
3010 GOSUB 9100
3020 R = 0:Z = 0:U = 1:PI = 6.283
18531
3030 PB = 128.5:TW = 2
3040 GOSUB 4000
3050 VTAB 22
3060 PRINT "TAS HEADING MISS
ILES";
3070 PRINT " FUEL X1000";
3080 VTAB 21: HTAB 1: PRINT "0";

3090 HTAB 9: PRINT "60";
3100 HTAB 18: PRINT "1 2";
3110 HTAB 32: PRINT "ALT 0";
3120 POKE 34,22: HOME : POKE -
16368,0
3130 TAS = 1:G = - 16:F = 995.9
3140 C% = "ECHO 1, "
3150 VTAB 21: GOTO 1000
Draw Hi-Res cockpit. Attitude indicator.
4000 HGR : HCOLOR= 3
4020 GH = 140:GV = 40:RG = 0:GX =
GH:GY = GV
4030 FOR R = 0 TO PI - .1 STEP P
I / 12
4040 HPLLOT 30 * COS (R) + GH,30
* SIN (R) + GV TO 35 * COS
(R) + GH,35 * SIN (R) + GV

```

```

4050 IF R > = PI / 4 AND R < =
3 * PI / 4 THEN HPLLOT GH -
2,30 * SIN (R) + GV TO GH +
2,30 * SIN (R) + GV
4060 NEXT
4070 SCALE= 10: ROT= R6
4080 XDRAW 11 AT 6X,6Y
Altimeter.
4100 AX = 240:AY = 50:AR = 0: SCALE=
1
4110 FOR R = 0 TO 9.5 STEP 0.5
4120 X = SIN (R * PI / 10):Y = COS
(R * PI / 10)
4130 HPLLOT AX + 25 * X,AY - 25 *
Y
4140 IF R = INT (R) THEN DRAW
R + 1 AT AX + 2 + 30 * X,AY +
3 - 30 * Y
4150 NEXT
4160 ROT= AR: SCALE= 24: XDRAW 1
2 AT AX,AY
Airspeed indicator.
4200 GOTO 4240
4210 R = LOG (X / 12 - 5) * PI /
6.4
4220 HPLLOT 30 + 25 * SIN (R),50
- 25 * COS (R) TO 30 + 30 *
SIN (R),50 - 30 * COS (R)
4230 RETURN
4240 FOR X = 80 TO 150 STEP 10
4250 GOSUB 4210: NEXT
4260 FOR X = 200 TO 600 STEP 100
4270 GOSUB 4210: NEXT
4280 SCALE= 24:SR = 0: ROT= 0
4290 XDRAW 12 AT 30,50
4300 SCALE= 1: DRAW 9 AT 46,21: DRAW
1 AT 52,21
4310 DRAW 2 AT 63,38: DRAW 1 AT
67,38: DRAW 1 AT 73,38
4320 DRAW 2 AT 66,51: DRAW 3 AT
70,51: DRAW 1 AT 76,51
4330 DRAW 2 AT 63,65: DRAW 6 AT
67,65: DRAW 1 AT 73,65
4340 DRAW 3 AT 49,78: DRAW 1 AT
55,78: DRAW 1 AT 61,78
4350 DRAW 4 AT 35,85: DRAW 1 AT
41,85: DRAW 1 AT 47,85
4360 DRAW 1 AT 80,90:AB = 1
4370 HPLLOT 5,12 TO 60,12 TO 80,3
2 TO 80,70 TO 60,90 TO 5,90 TO
5,12
Compass.
4500 HPLLOT 50,96 TO 50,101
4510 FOR R = 0 TO PI STEP PI / 3
6
4520 HPLLOT 50 + 29 * SIN (R),13
0 - 29 * COS (R)
4530 NEXT

```

```

4540 SCALE= 12:H = PI / 6 + .009
4550 CR = 64 - INT (H * 64 / PI)
: ROT= CR
4560 XDRAW 13 AT 50,130
Engine tachometer.
4600 FOR X = 199 TO 201 STEP 2
4610 FOR Y = 105 TO 155 STEP 5
4620 HPLLOT X,Y: NEXT Y,X
4630 SCALE= 1: ROT= 0: FOR Y = 0
TO 4:R = 157 - Y * 10
4640 DRAW Y + 6 AT 210,R: DRAW 1
AT 216,R: NEXT
4650 DRAW 2 AT 208,107: DRAW 1 AT
212,107: DRAW 1 AT 218,107
4660 HPLLOT 190,159 TO 190,90 TO
220,90 TO 220,159
4670 HPLLOT 200,159 TO 200,155
Radar screen.
4700 HPLLOT 105,80 TO 175,80 TO 1
75,152 TO 105,152 TO 105,80
4710 FOR X = 110 TO 171 STEP 10:
FOR Y = 81 TO 152 STEP 14
4720 HPLLOT X,Y: NEXT Y,X
4730 SCALE= 1: ROT= 0: DRAW 1 AT
142,158
4740 FOR X = 110 TO 170 STEP 60
4750 DRAW 5 AT X,158: DRAW 6 AT
X + 6,158: NEXT
4760 HPLLOT 101,156 TO 103,156: HPLLOT
161,156 TO 163,156: HPLLOT 16
2,155 TO 162,157
4770 FOR Y = 88 TO 145 STEP 14: HPLLOT
140,Y: NEXT
4800 RETURN
Data for shape table.
9000 DATA 13,0,28,0,36,0,41,0,5
0,0,60,0,67,0,76,0
9010 DATA 84,0,91,0,100,0,109,0
,114,0,116,0,27,231
9020 DATA 36,12,173,54,6,0,219,3
6,36,4,0,59
9030 DATA 63,12,12,12,28,191,6,0
,24,23,63,12
9040 DATA 40,12,28,63,7,0,27,36,
229,252,54,45
9050 DATA 0,27,63,76,225,63,36,4
5,45,0,27,63
9060 DATA 104,225,63,100,45,0,27
,36,12,60,63,7
9070 DATA 0,24,30,63,96,101,28,1
91,6,0,27,103
9080 DATA 33,63,28,12,173,6,0
9090 DATA 59,79,9,45,0,4,0,79,23
,36,4,0
9100 FOR X = 768 TO 888: READ Y:
POKE X,Y: NEXT : POKE 232,0
: POKE 233,3
9110 RETURN

```



by Alan J. Zett

PART II: THE ATARI CASSETTE

Every now and then, we here at *SoftSide* receive a call about CLOADing troubles on the Atari. The most common errors generated are: 138 (Device timeout), 140 (Serial bus input framing error), and 143 (Serial bus data frame checksum error). One general cure-all is to type LPRINT with no printer attached. This clears the cassette buffer after a bad load so you can start from scratch. Another good idea is to press SYSTEM RESET before loading (or saving) to restore the system to its initialized condition.

The Atari is an extremely well-built computer and as such, almost all cassette loading problems are tape-related. But if typing PRINT PEEK(65528) from BASIC displays a value of 255, then you have what Atari calls the "OPERATING SYSTEM 255" ROM. It means that you must read and write to cassette using a special procedure.

To save or load programs or data, place the tape in the recorder and press REWIND. When it has finished, take the cassette out and advance the tape (a ball point pen inserted in the take-up hub will help immensely) to the point where the splice between the plastic tape and the magnetic tape is positioned in the center slot of the cassette shell. Re-insert the tape in the recorder, reset the tape counter, and CSAVE or CLOAD.

But in the case where the computer is not at fault, cassette I/O errors are caused by one thing: improper positioning of the tape at the beginning of the "program leader." The program leader is a high-pitched tone that tells the Atari that a program is about to start on the tape. It also serves to drown out any noise on the tape that might otherwise be mistaken as program data.

The solution is easy if you have another cassette recorder. All you have to do is listen to the tape, and when the leader starts, stop the tape and put it back into the Atari recorder. Press PLAY and type CLOAD and your troubles are over.

There is no way to listen to a tape with the Atari cassette recorder alone.



The speaker on the Atari recorder is permanently disconnected, so you cannot hear what is on the tape — at least not directly. But try this: Before loading any program from cassette, type POKE 54018,54. Now turn up the speaker on your TV set, and type CLOAD — but DON'T enter the line with a RETURN.

Press the PLAY button on the cassette and listen. You will be able to hear what is on the tape through the TV, but you will still have complete control over the computer; all commands can be executed normally. As soon as you hear a clear portion of the leader, type RETURN to start CLOADing. In most cases there will be a blank gap on the tape between each program, which you will want to skip over. (All *SoftSide* cassette tapes are indented about ten turn counts from the beginning of the tape and there is a three-count gap between programs.) You should then hear the start of the

program leader, which may be very faint and is usually accompanied by a few pops and clicks caused by "glitches" at the recording head when the cassette is first started. When the leader settles down to a smooth tone, then the time is right to start CLOADing.

I'm not saying that this method is REQUIRED to properly load a program from cassette; I am simply saying that this method is almost fool-proof, even for tapes that are otherwise difficult to load. In fact, if you have the "OPERATING SYSTEM 255" ROM, this may be the only way to load in commercial tapes.

Another advantage of this POKE is that you can listen to any tape through the TV speaker. The computer will ignore the cassette completely unless a normal load or save command is executed. So if you would like a little background music while using your computer, just POKE YOUR ATARI! ☺



by Mark Pelczarski and Paul Marentette

The "Developing Database" series is copyrighted 1981 by Mark Pelczarski. It may be reprinted with written permission from the author.

Finally! At last! Here's a complete listing of the Atari version of the developing database program, courtesy of Paul Marentette of New York.

One constant in working with programs across a few computers is that there's always a better way to do it on another machine. (Never mind the parts that require 15 doses of frustration to translate features that are quite different on the receiving machine.) A literal translation of a program will usually work, given enough patience and a little elbow grease, but a translation that takes the spirit of a program and applies it to the features of a different computer is usually far superior. That's where Paul Marentette comes in. The Atari database as published in the series worked all right after some tinkering here and there, but we had early plans for a version written exclusively for the Atari. The invasion of the bugs, along with assorted commitments on my part, slowed things down a bit when along came Paul to the rescue. (Actually, my main obstacle was a RETURN statement in the rewrite that insisted on changing the value of a variable. Unless I misunderstand Atari's conception of the BASIC language, that's not supposed to happen. I've had various and sundry hypothetical explanations, along with a few good ones I made up, but nothing that fully explained the alleged bug in the BASIC.)

Anyway, Paul's version of the database does justice to the Atari's capabilities, and makes the program look pretty good (which is good, because maybe some people will accidentally think it's my fault). So, following is Paul's explanation of the changes to the program. The instructions for using the database accompany the listings of the Apple version in the October *SoftSide*.

```
Line 1700 I$(LEN(I$)+N1)=" "
:I$(I*RL+N1)=I$(I*RL+N1+
NH*IL+IL):IF I$=" "
THEN I$="":GOTO 1710
1705 I$=I$(N1,LEN(I$)-N1)
```

Documentation

More than 2000 bytes were trimmed from this expanded version of the program by using multiple statements per line number and by converting commonly-used numeric constants (i.e., 1, 2, 3, etc.) to numeric variables (N1, N2, N3 . . .). This latter step achieves a tremendous saving in memory.

Unfortunately, the use of multiple statement lines makes the program somewhat less readable, but the trade-off for memory savings more than justifies the tighter coding. Documentation can accompany the program in text form.

Improved Menu System

The main menu now has a title banner which gives a constantly updated report on the current file name, the number of records in memory, and the number of records that can still be fit into memory. This "records-free" parameter replaces the characters-free category in the original version. It is more valuable to know how many complete records you can add to your database. The value is calculated by dividing the total number of bytes allotted to the I\$ string by the record length of one record (RL).

When the program is first run, the file name is set to NULL and the records-free parameter is not displayed until the user has designed the headings and stated the maximum item length. Thereafter, the status of the current file is updated after every sequence of commands.

New I/O Routines

The input and output routines have been radically altered. Here are some of the major changes:

1. When first run, the program prompts for either loading a saved file or initializing a new one. If the LOAD option is chosen and if there is a disk device connected (this is automatically determined when the program is initialized), the program automatically gives a listing at the bottom of the screen of the data files (.DAT only) in the disk directory. If none are found,

the "load file" option is cancelled. If .DAT files do exist, the user is prompted to enter the file name he wants to retrieve. The file is then loaded and the menu is displayed.

2. If the user chooses the INITIALIZE A NEW FILE option, he is immediately prompted for a file name. The name is checked for validity (limit of eight characters and no file extension) and .DAT is appended to the name. The name is then displayed at all times next to the "FILE:" heading of the banner.

3. When the SAVE FILE option is chosen from the menu, the user is given the opportunity to rename the file before it is saved. There is a single save routine for disk and cassette. The appropriate device is automatically selected based on the HASDOS variable. If the cassette is in use, the file name is written to the tape so that each tape data file has an indentifying header.

4. Disk users have the F option in the menu (it is not listed for cassette users). This option can be chosen at any time to display all of the data files on the disk. The file names are printed below the menu area in four-across fashion.

5. When a cassette user is loading a tape file, he is asked for the file name. There is a single read routine for disk and cassette, but with cassette use the file name is checked against the file name header on the tape. If they do not match, the file is not read, and a message states that the file was not found. Of course, with disk use, the file name is always checked against the directory and an error message is displayed if the file is not found.

6. The save/load routines for format files have been modified so that a .FMT extension is appended to each file name. The string .FMT is used to hold the current format name.

Uncluttered Video Screens

Whenever possible, the video display has been "cleaned up." After almost every command, the screen is cleared and the banner is redisplayed. Suboptions have been listed under one another in menu fashion. Instead of: (S) SCREEN or (P) PRINTER? the revised program offers:

(S) SCREEN
(P) PRINTER ?

This format is maintained throughout the program.

If an input error is made, the screen will usually clear and the choices will be reprinted on a clean screen, instead of having past work scroll off the top of the screen.

Print Routine Changes

1. If the video is chosen as the print device, the screen editor is opened for output (line 2210) with the symbol "E:". The "S:" symbol in the original program caused all control characters such as tabs and cursor controls to be printed rather than acted upon. Sending these characters to the screen editor allows the effective use of these controls when they are buried in formatting strings. The "E:" allows more creative screen formatting.

2. If the printer is chosen as the print device, and if there is no printer on-line, an error message is given (line 2230) and the video editor is opened as the default.

3. When printing to the screen, only one record is printed on the screen at a time. Hitting any key (RETURN is suggested) causes the message SEARCHING... to be printed at the bottom of the screen since the search may take some time before finding the next record. When the next record is found, the screen is cleared, and the new record is printed on a clean screen. If no records are found in a search (i.e., search criteria not met) a message to that effect is printed (line 2500). Like in the original version, the user has the option between records of hitting ESC to abort the routine and return to the menu. When the end of file is reached, a buzzer sounds, and an appropriate message is printed.

Modified Search Routine

Because of the lack of string arrays in Atari BASIC, there was an important error in the Atari version of the search routine. In the Apple and TRS-80® versions, the asterisk could be used as a "wild card" in a search ("MA**")

would search for any item beginning with "MA"). This worked in the Atari version too, but a search for "MA" without the asterisk would also turn up every item beginning with "MA". In fact, if we search for items equal to "MA" without the wild card, we should only find items equal to "MA", not equal to or greater than. (There is a separate > = search option.)

The revised version (lines 2320 to 2340) makes the "equal to" type search work as it was intended. Only items strictly equal to the value specified will be found by the search unless the wild card * is used.

Conclusions

Although the original algorithms have been kept, this revision of the database represents a whole new look. The program functions more smoothly, input/output routines are much better debugged and are simpler to use, and the search routine now works the way the way it was designed to operate.

```
10 REM SOFTSIDE DATABASE PROGRAM
20 REM AUTHOR - MARK PELCZARSKI
30 REM ATARI TRANS. - R. BOUCHARD
40 REM
50 REM REVISED BY PAUL MARENTETTE
60 REM 708 WASHINGTON ST. #6B
70 REM NEW YORK, N.Y. 10014
80 REM PHONE (212) 929-6804
90 REM CURRENT REVISION -- 7/7/81
100 REM SAFE REENTRY = GOTO 330
110 DIM A$(260):X=INT(FRE(0)*0.8):NB=8
:DIM H$(X*0.1),I$(X*0.9),FI$(14),FN$(N
B),FMT$(NB)
120 NO=0:N1=1:N2=N1+N1:N3=N1+N2:N4=N2+
N2:N6=N3+N3:N7=N1+N6:GOTO 180
130 FOR W=N1 TO 400:NEXT W:RETURN
140 ? CHR$(125);"
FILE: ";FN$:REM ctrl-Q, 17 ctrl-R's,
ctrl-E, 3 spaces
150 ? "ISOFTSIDE DATABASE! RECS: ";N
I+N1:?" ";REM ctrl
-Z, 17 ctrl-R's, ctrl-C
151 REM ! in above line is shift==
160 IF RL>NO THEN ? " FREE: ";INT(X*
0.9/RL)-NI+N1;
170 ? :? :? :RETURN
180 DIM C1(N7),C2(N7),F$(400),B$(100),
C$(100):HASDOS=NO:NI=-N1
190 OPEN #N2,N4,NO,"K":GRAPHICS NO:SE
TCOLOR N2,N7,NO:SETCOLOR N4,N7,NO
200 FI$="C":FOR I=NO TO N6+N6:IF PEEK
(794+N3*I)=ASC("D") THEN HASDOS=N1:FI$
="D:"
```

```
210 NEXT I:GOTO 230
230 FN$="NULL":GOSUB 140:?"(I) INITIA
LIZE A NEW FILE"
240 ? "(L) LOAD A SAVED FILE?":? :? "S
ELECT? ";GET #N2,A:IF CHR$(A)<>"L" AN
D CHR$(A)<>"I" THEN 230
250 GOSUB 140:GOTO 290
260 FN$="":? " ENTER FILE NAME ";:INPU
T FN$:IF FN$="" THEN POSITION N6,22:GO
TO 260:REM shift-BACK S before ENTER
270 FI$(N3)=FN$:FI$(LEN(FI$)+N1)=".DAT
":FOR I=N1 TO LEN(FN$):IF FN$(I,I)=".
" THEN POSITION N6,22:GOTO 260
280 NEXT I:RETURN
290 IF CHR$(A)="L" THEN GOSUB 700:GOTO
310
300 IF CHR$(A)="I" THEN GOSUB 860:GOTO
330
310 IF SS=N1 THEN 330
320 GOTO 230
330 TRAP 330:CLOSE #N3:GOSUB 140:?"(S
) SAVE CURRENT FILE"
340 ? "(P) PRINT DATA"
350 ? "(A) ADD DATA"
360 ? "(C) CHANGE A RECORD"
370 ? "(D) DELETE A RECORD"
380 ? "(T) SORT"
390 IF HASDOS THEN ? "(F) LIST FILE NA
MES"
400 ? "(N) START NEW FILE"
410 ? "(Q) QUIT":?
420 ? :? :? "SELECT ";
430 GET #N2,A:?" CHR$(A)
```

```
440 IF CHR$(A)="S" THEN GOSUB 950:GOTO
330
450 IF CHR$(A)="P" THEN GOSUB 1130:GOT
O 330
460 IF CHR$(A)="A" THEN GOSUB 1450:GOT
O 330
470 IF CHR$(A)="C" THEN SB=N3:GOSUB 19
80:GOTO 330
480 IF CHR$(A)="D" THEN SB=N4:F5=N1:GO
SUB 1980:GOTO 330
490 IF CHR$(A)="T" THEN GOSUB 1740:GOT
O 330
500 IF CHR$(A)="F" AND HASDOS THEN GOS
UB 590:GOTO 330
510 IF CHR$(A)="Q" OR CHR$(A)="N" THEN
530
520 GOTO 330
530 AA=A:IF SS THEN 570
540 ? " CURRENT FILE IS NOT SAVED.":?
"CANCEL COMMAND? (Y/N) ";:GET #2,A:?" C
HR$(A):REM ctrl-- before CURRENT
550 IF CHR$(A)="Y" THEN 330
560 IF CHR$(A)<>"N" THEN 540
570 IF CHR$(AA)="N" THEN CLR :RUN
580 TRAP 40000:GRAPHICS NO:END
590 POSITION N2,16:?" DATA FILES":TR
AP 600:OPEN #N1,N6,NO,"D:*.DAT":GOTO 6
10:JJ=NO:REM shift-BACK S before DATA
600 POSITION N4,20:?"CAN'T READ DISK
DIRECTORY.":CLOSE #N1:GOSUB 130:RETURN
610 INPUT #N1,A$
620 FOR I=17 TO 20:FOR J=N2 TO 29 STEP
```

```

9:IF A$(N1,N1)<>" THEN 670
630 POSITION J,I: A$(N3,10):JJ=JJ+N1:
INPUT #N1,A$:NEXT J:NEXT I
640 IF A$(N1,N1)<>" THEN 670
650 POSITION 8,23: ? "PRESS RETURN FOR
MORE ";GET #N2,A
660 FOR I=23 TO 17 STEP -N1:POSITION N
0,I: ? " ";NEXT I:GOTO 620:REM shift-B
ACK S
670 TRAP 330:IF JJ=NO THEN ? : ? "NO DA
TA FILES IN DIRECTORY":GOSUB 130
680 CLOSE #N1:POSITION N6,23:IF CHR$(A
)<>"L" THEN ? "PRESS RETURN TO CONTINU
E ";GET #N2,A
690 RETURN
700 IF HASDOS THEN GOSUB 590:IF JJ=NO
THEN RETURN
730 POSITION N6,22:GOSUB 260:IF HASDOS
=NO THEN ? "HIT ENTER TO PLAY ";:INPUT
A$
740 GOSUB 140: ? : ? "LOADING ";FN$:;TRA
P 850:OPEN #N1,N4,N0,FI$:IF HASDOS THE
N 750
745 INPUT #N1,A$:IF A$<>FI$ THEN 850
750 INPUT #N1,NH,NI,HL,IL:IF IL>NO THE
N DIM T$(NH*IL+IL+N1):RL=(NH+N1)*IL
760 SEG=N1
770 INPUT #N1,A$:IF LEN(A$)=NO THEN 79
0
780 H$(SEG)=A$:SEG=SEG+250:GOTO 770
790 IF NI=-N1 THEN 830
800 SEG=N1
810 INPUT #N1,A$
820 I$(SEG)=A$:IF LEN(A$)=250 THEN SEG
=SEG+250:GOTO 810
830 CLOSE #N1
840 SS=N1:RETURN
850 ? " ";FN$;" NOT FOUND":A=70:GOSUB
680:SS=NO:RETURN :REM shift-BACK S in
1st quotes; ctrl-2 and space in 2nd
860 GOSUB 260: ? : ? "HOW MANY HEADINGS
";:INPUT NH:IF NH<N1 THEN 860
870 ? : ? "ENTER MAXIMUM HEADING LENGTH
";:INPUT HL:IF HL<N1 THEN 870
880 NH=NH-N1:NI=-N1:H$="":I$="":GOSUB
140
890 FOR I=NO TO NH
900 ? "HEADING #";I+N1;" ";:INPUT A$
910 IF LEN(A$)>HL THEN ? "MAXIMUM LENG
TH IS ";HL;" ". REENTER":GOTO 900
920 IF LEN(A$)<HL THEN A$(LEN(A$)+N1)=
" ":GOTO 920
930 H$(LEN(H$)+N1)=A$:NEXT I
940 SS=NO:RETURN
950 GOSUB 140: ? "IS FILE NAME - ";FN$;
" - OK? ";:GET #N2,A:IF CHR$(A)="N" TH
EN GOSUB 260
960 IF HASDOS THEN 980

```

```

970 ? : ? "HIT ENTER TO RECORD ";:INPUT
A$
980 TRAP 1110: ? : ? "SAVING ";FN$:;OPEN
#N1,NB,NO,FI$:IF HASDOS=NO THEN PRINT
#N1;FI$
990 PRINT #N1;NH: ? #N1;NI: ? #N1;HL: ? #
N1;IL:SEG=N1
1000 IF LEN(H$)<=SEG+249 THEN PRINT #N
1;H$(SEG,LEN(H$)):GOTO 1030
1010 PRINT #N1;H$(SEG,SEG+249):SEG=SEG
+250:IF SEG>LEN(H$) THEN 1030
1020 GOTO 1000
1030 PRINT #N1;" "
1040 IF NI=-N1 THEN 1100
1050 SEG=N1
1060 IF LEN(I$)<=SEG+249 THEN PRINT #1
;I$(SEG,LEN(I$)):GOTO 1090
1070 PRINT #N1;I$(SEG,SEG+249):SEG=SEG
+250:IF SEG>LEN(I$) THEN 1090
1080 GOTO 1060
1090 PRINT #N1;" "
1100 CLOSE #N1:SS=N1:RETURN
1110 ? "OUTPUT ERROR":GOSUB 130:CLOSE
#N1:RETURN
1120 REM PRINT SUBROUTINE VERS.3
1130 FS=N1:IF NI=-N1 THEN GOSUB 2520:R
ETURN
1140 GOSUB 140: ? "FORMATTED OUTPUT? (Y
/N) ";:GET #N2,A: ? :IF CHR$(A)="Y" THE
N GOSUB 2540:FS=N2:GOTO 1170
1150 IF CHR$(A)<>"N" THEN 1120
1160 FS=N1:GOTO 1180
1170 IF FS=N2 AND FMT$="" THEN 1120
1180 GOSUB 140:SB=N1: ? "(S) SCREEN or"
: ? "(P) PRINTER ?";:GET #2,A: ? :IF CHR
$(A)="P" THEN SB=N2:GOTO 1200
1190 IF CHR$(A)<>"S" THEN 1180
1200 GOSUB 1990:CLOSE #N3:POKE 752,N0:
RETURN
1210 REM PRINT ONE RECORD VERS. 4
1220 ON FS GOSUB 1270,1310:POSITION N2
,22: ? "PRESS RETURN (ESC FOR MENU)";
1230 GET #N2,A:IF A=27 THEN RS=N1
1240 RETURN
1250 REM PRINT ONE RECORD TO PRINTER,
VERS.4
1260 ON FS GOSUB 1270,1310:RETURN
1270 IF SB=N1 OR SB=N4 THEN ? CHR$(125
)
1280 PRINT #N3:PRINT #N3:"RECORD ";I+N
1:PRINT #N3:FOR J=NO TO NH
1290 PRINT #N3;H$(J*HL+N1,J*HL+HL),I$(
I*RL+N1+J*IL,I*RL+J*IL+IL):NEXT J:RETU
RN
1300 REM PRINT ONE FORMAT
1310 J=N1:T=NO:B$="":IF F$(LEN(F$))<>"
6" THEN F$(LEN(F$))="6"
1320 J1=VAL(F$(J,J)):J=J+N1:IF J1<5 TH
EN N=VAL(F$(J,J+N1)):J=J+N2

```

```

1330 ON J1 GOTO 1340,1350,1360,1370,13
90,1440
1340 A$=H$(N*HL+N1,N*HL+HL):GOTO 1420
1350 A$=I$(I*RL+N1+N*IL,I*RL+N*IL+IL):
GOTO 1420
1360 FOR QQ=N1 TO N:B$(LEN(B$)+N1)=" "
:NEXT QQ:GOTO 1430
1370 PRINT #N3;B$:IF N>N1 THEN FOR J2=
N2 TO N:PRINT #N3:NEXT J2
1380 B$="":GOTO 1430
1390 J2=J
1400 IF F$(J2,J2)<>"!" THEN J2=J2+N1:G
OTO 1400
1410 A$=F$(J,J+(J2-J)-N1):J=J2+N1
1420 B$(LEN(B$)+N1)=A$
1430 GOTO 1320
1440 PRINT #N3;B$:RETURN
1450 GOSUB 140:SS=NO:IF IL>NO THEN 148
0
1460 ? "ENTER MAXIMUM ITEM LENGTH ";:I
NPUT IL:IF IL<N1 THEN IL=NO:GOTO 1460
1470 DIM T$(NH*IL+IL):RL=(NH+N1)*IL:G0
SUB 140
1480 NI=NI+N1: ? : ? "RECORD NUMBER ";NI
+N1;" (ENTER ! TO END)": ? : ? :FOR J=N
0 TO NH
1490 ? H$(J*HL+1,J*HL+HL)"; " : :INPUT
A$:IF A$="!" THEN NI=NI-N1:J=NH:NEXT
J:RETURN
1500 IF LEN(A$)>IL THEN ? "TOO LONG. M
AXIMUM SIZE IS ";IL;" ". REENTER":GOTO 1
490
1510 IF LEN(A$)<IL THEN A$(LEN(A$)+N1)
=" ":GOTO 1510
1520 I$(LEN(I$)+N1)=A$:NEXT J:GOTO 145
0
1530 REM CHANGE SUBROUTINE VERS.2
1540 GOSUB 140: ? "(C) CHANGE ITEM, (K)
KEEP ITEM, OR": ? "(R) KEEP REMAINDER
OF RECORD"
1550 ? : ? "RECORD ";I+N1:CS=N1:RS=NO:F
OR J=0 TO NH: ? : ? H$(J*HL+N1,J*HL+HL);
" : ";I$(I*RL+1+J*IL,I*RL+J*IL+IL)";
";
1560 IF RS=N1 THEN PRINT :GOTO 1640
1570 GET #N2,A:IF CHR$(A)<>"C" AND CHR
$(A)<>"K" AND CHR$(A)<>"R" THEN 1570
1580 ? CHR$(A):IF CHR$(A)="K" THEN 164
0
1590 IF CHR$(A)="R" THEN RS=N1:GOTO 16
40
1600 ? H$(J*HL+N1,J*HL+HL)"; " : :INPU
T A$
1610 IF LEN(A$)>IL THEN PRINT "TOO LON
G. MAXIMUM SIZE IS ";IL;" ". REENTER":G0
TO 1600
1620 IF LEN(A$)<IL THEN A$(LEN(A$)+N1)
=" ":GOTO 1620
1630 I$(I*RL+N1+J*IL,I*RL+J*IL+IL)=A$:

```

```

CS=NO
1640 NEXT J:RS=NO:IF CS=NO THEN SS=NO
1650 RETURN
1660 REM DELETE SUBROUTINE VERS.2
1670 ? :POKE 752,NO?: "DELETE THIS REC
ORD? ";
1680 GET #N2,A:IF CHR$(A)<>"Y" AND CHR
$(A)<>"N" THEN 1680
1690 ? CHR$(A):POKE 752,N1:IF CHR$(A)=
"N" THEN 1720
1700 I$(LEN(I$)+N1)=" ":I$(I*RL+N1)=I$
(I*RL+N1+NH*IL+IL):I$=I$(N1,LEN(I$)-N1
)
1710 NI=NI-N1:SS=NO:I=I-N1
1720 RETURN
1730 REM SORT ROUTINE
1740 IF NI=-N1 THEN GOSUB 2520:RETURN
1750 GOSUB 140:FOR J=NO TO NH?: "(;J+
N1;)" :H$(J*HL+N1,J*HL+HL):NEXT J
1760 ? :? "SORT WHICH HEADING ";:INPUT
J1:J1=J1-N1
1770 IF J1<NO OR J1>NH THEN RETURN
1780 GOSUB 140?: "(A) ASCENDING, OR (D
) DESCENDING":GET #N2,A:IF CHR$(A)="A"
THEN A=N1:GOTO 1810
1790 IF CHR$(A)="D" THEN A=N2:GOTO 181
0
1800 GOTO 1780
1810 ? :? "SORTING...":FOR I=NO TO NI
-1
1820 T=I:FOR I1=T+N1 TO NI
1830 ON A GOTO 1840,1860
1840 IF I$(I1*RL+N1+J1*IL,I1*RL+J1*IL+
IL)<I$(T*RL+N1+J1*IL,T*RL+J1*IL+IL) TH
EN T=I1
1850 GOTO 1870
1860 IF I$(I1*RL+N1+J1*IL,I1*RL+J1*IL+
IL)>I$(T*RL+N1+J1*IL,T*RL+J1*IL+IL) TH
EN T=I1
1870 NEXT I1
1880 IF T=I THEN 1900
1890 T$=I$(T*RL+N1,T*RL+RL):I$(T*RL+1,
T*RL+RL)=I$(I*RL+1,I*RL+RL):I$(I*RL+1,
I*RL+RL)=T$
1900 NEXT I
1960 ? "COMPLETED.":GOSUB 130:SS=NO:RE
TURN
1970 REM SEARCH SUBROUTINE, VERS. 2
1980 IF N1=-N1 THEN GOSUB 2520:RETURN
1990 I1=NO:I2=N1:J=NO:C1(NO)=-N1:BS=N1
:NF=N1
2000 GOSUB 140?: "SEARCH CRITERIA:":?
?: "0) RECORD NUMBER"
2010 FOR I=NO TO NH:PRINT I+N1;") ";H$
(I*HL+N1,I*HL+HL):NEXT I?: ? NH+N2;")
BEGIN"
2020 POSITION N2,20?: "WHICH FIELD: ";
:INPUT I:IF I<NO OR I>NH+N2 THEN 2020
2030 IF I=NH+N2 THEN C1(J)=-N1:GOTO 21
70

```

```

2040 C1(J)=I-N1
2050 POSITION N2,21?: "(1) <= (2)
= (3) >=" :;:INPUT A:C2(J)=A:I
F C2(J)<N1 OR C2(J)>N3 THEN 2050
2060 POSITION N2,22?: "VALUE:":;:IF C1(
J)=-N1 THEN 2110
2070 PRINT " ";:INPUT A$:IF LEN(A$)>IL
THEN ? "TOO LONG. MAXIMUM LENGTH IS "
;:IL:GOTO 2060
2080 IF LEN(A$)<IL THEN A$(LEN(A$)+N1)
=" ":GOTO 2080
2090 C$(J*IL+N1,J*IL+IL)=A$:J=J+N1:IF
J>N7 THEN 2180
2100 GOTO 2000
2110 PRINT " ";:INPUT I:IF I<N1 OR I>N
I+N1 THEN 2110
2120 I=I-N1
2130 IF C2(J)=N1 THEN I2=I
2140 IF C2(J)=N2 THEN I1=I:I2=I
2150 IF C2(J)=N3 THEN I1=I
2160 GOTO 2000
2170 IF J<N2 THEN 2200
2180 POSITION N2,21?: " 1) ITEM MUST M
EET ALL CONDITIONS?: " 2) ITEM MAY ME
ET ANY CONDITION ";:INPUT BS
2181 REM shift-BACK S before 1) and 2)
in above line
2190 IF BS<N1 OR BS>N2 THEN 2180
2200 TRAP 2230:RS=NO:IF SB=N2 THEN OPE
N #N3,N8,NO,"P:":TRAP 330:GOTO 2240
2210 OPEN #N3,N8,NO,"E:":POKE 752,N1:T
RAP 330:SETCOLOR N2,N7,NO:SETCOLOR N4,
N7,NO
2220 POSITION N2,22?: "SEARCHING...":;
GOTO 2240
2230 CLOSE #N3?: "PRINTER NOT ON LINE"
:GOTO 2510
2240 I=I1-N1:FOR I9=I1 TO I2:I=I+N1
2250 AS=NO:FOR J=NO TO N7
2260 IF C1(J)=-N1 THEN J=N7:GOTO 2410
2270 B$=C$(J*IL+N1,J*IL+IL)
2280 IF LEN(B$)>NO THEN IF B$(LEN(B$),
LEN(B$))=" " THEN B$=B$(N1,LEN(B$)-N1)
:GOTO 2280
2290 ON C2(J) GOTO 2300,2320,2360
2300 IF I$(I*RL+N1+C1(J)*IL,I*RL+C1(J)
*IL+LEN(B$))<=B$ THEN 2380
2310 GOTO 2400
2320 A$=" ":A$(IL-N1)=" ":A$(2)=A$:A$(
N1,LEN(B$))=B$:IF I$(I*RL+N1+C1(J)*IL,
I*RL+C1(J)*IL+IL)=A$ THEN 2380
2330 IF B$(LEN(B$),LEN(B$))<>"#" THEN
2400
2340 T=LEN(B$)-N1:IF I$(I*RL+N1+C1(J)*
IL,I*RL+C1(J)*IL+T)=B$(N1,T) THEN 2380
2350 GOTO 2400
2360 IF I$(I*RL+N1+C1(J)*IL,I*RL+C1(J)
*IL+LEN(B$))>=B$ THEN 2380
2370 GOTO 2400

```

```

2380 IF BS=N2 THEN AS=N1:J=N7
2390 GOTO 2410
2400 IF BS=N1 THEN AS=N2:J=N7
2410 NEXT J
2420 IF AS=NO AND BS=N1 THEN 2440
2430 IF AS=NO OR AS=N2 THEN 2490
2440 NF=0:IF SB=1 THEN GOSUB 1220
2450 IF SB=N2 OR SB=N4 THEN GOSUB 1260
2460 IF SB=3 THEN GOSUB 1540
2470 IF SB=N4 THEN GOSUB 1670
2480 IF RS=N1 THEN I9=I2
2490 POSITION N2,22?: "SEARCHING...
";:NEXT I9:POKE 7
52,NO:IF RS THEN RETURN
2500 IF NF THEN POSITION N2,20?: "NO M
ATCHES FOUND";
2510 POSITION N2,22?: CHR$(253);"END O
F FILE...RETURN FOR MENU":;GET #N2,A:R
ETURN
2520 ? CHR$(253);"NO DATA IN MEMORY.":
GOSUB 130:RETURN
2530 REM PRINT FORMATTING, V.1
2540 GOSUB 140:IF FMT$="" THEN 2570
2550 ? "SAME FORMAT? ";:GET #N2,A:PRIN
T :IF CHR$(A)="Y" THEN RETURN
2560 IF CHR$(A)<>"N" THEN 2540
2570 ? "(L) LOAD FORMAT or?": "(C) CRE
ATE FORMAT ";:GET #N2,A?: :IF CHR$(A)=
"C" THEN 2670
2580 IF CHR$(A)<>"L" THEN 2570
2590 TRAP 2650
2600 GOSUB 2890:IF FMT$="" THEN RETURN
2610 OPEN #N1,N4,NO,A$:IF HASDOS THEN
2630
2620 INPUT #N1,A$:IF A$<>FMT$ THEN 265
0
2630 F$=""
2640 INPUT #N1,NF:FOR J=NO TO NF:INPUT
#N1,A$:F$(LEN(F$)+N1)=A$:NEXT J:GOTO
2660
2650 ? "FORMAT ";FMT$;" NOT FOUND":GOS
UB 130:FMT$=""
2660 CLOSE #N1:RETURN
2670 NF=NO:J=NO:F$="" :GOSUB 140?: "STA
RT IN THE UPPER LEFT CORNER AND"? "WO
RK ACROSS EACH LINE.":?
2680 ? "1:HEADING":? "2:ITEM":? "3:TAB
":? "4:NEXT LINE":? "5:STRING":? "6:EN
D":?
2690 INPUT J1:IF J1<N1 OR J1>N6 THEN 2
690
2700 F$(LEN(F$)+N1)=STR$(J1):J=J+N1
2710 ON J1 GOTO 2720,2720,2750,2750,27
90,2810
2720 GOSUB 140:FOR T=NO TO NH:PRINT T+
N1;") ";H$(T*HL+N1,T*HL+HL):NEXT T
2730 ? :? "WHICH? ";:INPUT T:T=T-N1:IF
T<NO OR T>NH THEN 2730

```

```

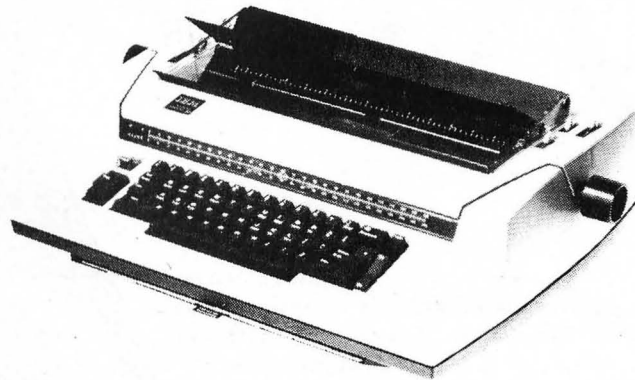
2740 GOTO 2760
2750 ? "HOW MANY? ";:INPUT T:IF T<N1 O
R T>99 THEN ? "OUT OF RANGE.":GOTO 275
0
2760 A$=STR$(T):IF T<10 THEN A$="0":A$
(LEN(A$)+N1)=STR$(T)
2770 F$(LEN(F$)+N1)=A$:J=J+N2
2780 GOTO 2800
2790 ? "STRING: ";:INPUT A$:A$(LEN(A$)
+N1)="!":F$(LEN(F$)+N1)=A$:J=J+LEN(A$)
2800 GOSUB 140:GOTO 2680
2810 TRAP 2880:GOSUB 2890:IF FMT$="" T
HEN RETURN
2820 OPEN #N1,NB,NO,A$:IF HASDOS THEN
2840
2830 PRINT #N1;FMT$
2840 PRINT #N1;INT((LEN(F$)-N1)/250)
2850 FOR J=N0 TO INT((LEN(F$)-N1)/250)
:IF LEN(F$)<J*250+250 THEN ? #N1;F$(J*
250+N1,LEN(F$)):GOTO 2870
2860 PRINT #N1;F$(J*250+N1,J*250+250):
NEXT J
2870 CLOSE #N1:RETURN
2880 ? "OUTPUT ERROR":GOSUB 130:CLOSE
#N1:GOTO 2810
2890 GOSUB 140:FMT$="":? "ENTER FORMAT
NAME ";:INPUT FMT$:IF LEN(FMT$)=NO TH
EN RETURN
2900 FOR J=N1 TO LEN(FMT$):IF FMT$(J,J
)="." THEN ? :? "INVALID NAME":GOSUB 1
30:GOTO 2890
2910 NEXT J
2920 IF HASDOS THEN A$="D":A$(LEN(A$)
+N1)=FMT$:A$(LEN(A$)+N1)="."FMT":RETURN
2930 A$="C":RETURN

```

WHEN YOU SPEND SO MUCH FOR A PRINTER,
YOU SHOULD HAVE ONE THAT YOU CAN USE---

Introducing....

THE IBM TOTAL PRINTER/TYPEWRITER



FEATURES:

10 and 12 Pitch, Proportional Space, Full Typewriter Use, Auto Correcting, Sound Cover, "Smart" Keyboard

SPECIFICATIONS:

200 WPM Throughput, Either Serial OR Parellel, Self-Test, Lowest On-Site Maintenance, IBM Backed Printer. Cables stocked for all Apple, TRS(I, II, III), RS-232 systems.

PRICE: ONLY \$1995 (With 30 Day IBM Service Agreement)

CONTACT:

ICOM 11 N. Main, Lombard, Illinois 60148 (312) 932-1766

16K Memory Upgrade Kits for Apple™ or TRS-80®

Everything you need to up-grade to a 16K system

Each kit is 100% guaranteed against failure. Add high quality, high density memory for less than you would expect to pay!

- 20 nano seconds
- 8 tested and guaranteed 16K RAM's
- Easy-to-follow instructions
- Only tool required is a household screwdriver

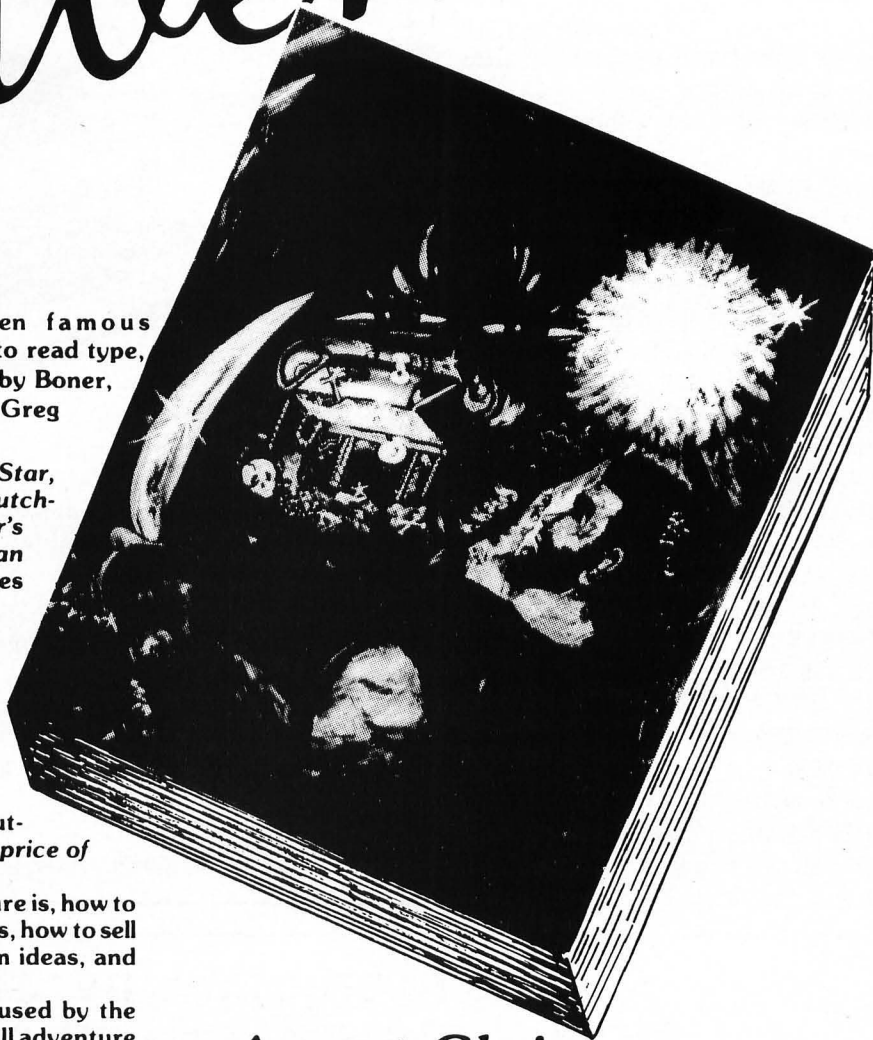
16K Memory Upgrade Kit \$19.95

Offer ends January 15, 1982



The Captain 80 Book of

BASIC Adventures



- Eighteen famous adventures, listed in large easy to read type, ready to be keyed in. Adventures by Boner, Kepner, Powers, Micklus, Forsythe, Greg Hassett and others.

- Includes *Temple of the Sun, Dog Star, Lost Ship, Spider Mountain, Lost Dutchman's Gold, Thunder Road, Sorcerer's Castle, Deadly Dungeon, Atlantean Odyssey* and others. These adventures would cost over \$200 if purchased individually!

- Includes a unique *Adventure Generator* program - not available anywhere. YES, this program will actually write another BASIC adventure program, which you may then run. Not even the author of the generator program knows the outcome! *This program alone is worth the price of the entire book!*

- Includes chapters on what an adventure is, how to play adventures, how to write adventures, how to sell your adventure, ten adventure program ideas, and more!

- Study the techniques and methods used by the masters of adventure program writing. All adventure programs listed in this book are in their original form and in full length. Although specifically written for the TRS-80 Model I and III, these programs are adaptable to other computers using Microsoft's BASIC.

- Available at your dealer approximately November 15th, 1981. The suggested retail price of this book is \$19.95.

- The BASIC Adventure Book was compiled and edited by Bob Liddil, of *The Programmers Guild*. It measures 8½ by 11 inches and contains 256 pages.

• **ORDER YOURS TODAY!**

Dealer discounts available - please inquire

TRS-80® is a registered trademark of the Tandy Corp.

A neat Christmas gift idea!

Send To: 80-U.S. Journal
3838 South Warner Street
Tacoma, Washington 98409

Please send my copy of the BASIC Adventure Book. My check or Money Order for \$22.00 (\$19.95 plus \$2.05 postage and handling) is enclosed.

Name _____

Address _____

City _____ State _____ Zip _____

Visa/MC _____

Exp. Date _____ Signature _____

(206) 475-2219 Phone orders accepted for Visa/MC only.

Order#SS

SoftSide



by Richard Lamb

Design Master is graphics program for a 16K Atari.

This program can draw thousands of geometrical designs, most of which are symmetrical in nature. There are nine different types of basic designs and all can be varied by the parameters keyed in. The parameters for each design are all explained on the screen and some of the designs, namely the Spiral, Oscilloscope, and Epicycloid, will display a larger variety of patterns so that the more you experiment, the more you get.

I have included default parameter values for each design so that if you enter 0 for the first parameter, a representative pattern for a design will

be displayed. When you are finished viewing an Oscilloscope or Epicycloid design, hit "S" to stop the display and return to the main menu. The Oscilloscope and the Epicycloid are the only two designs where you can hit "S" in the middle of the design generation because you may want to interrupt these displays while the design is being drawn. I set the background color to dark red to produce a nice variety of colors for graphics 24. However, the background color can be changed by hitting "C" in all but the Oscilloscope and Epicycloid designs.

Most of the designs were written us-

ing a conversion of Cartesian Coordinates (X,Y) to Polar Coordinates ($X = \text{radius} * \text{COS}(\text{angle})$, $Y = \text{radius} * \text{SIN}(\text{angle})$) which, in effect, allows you to generate designs symmetrically by varying the degree of angle and length of the radius from the center of the screen.

If you want to see whether you are interested in these designs, you can code in just one of them (for example, lines 100-190, 990-999 changing 994 to GOTO 100). If you are bold, then copy over one of the designs and experiment with your own idea.

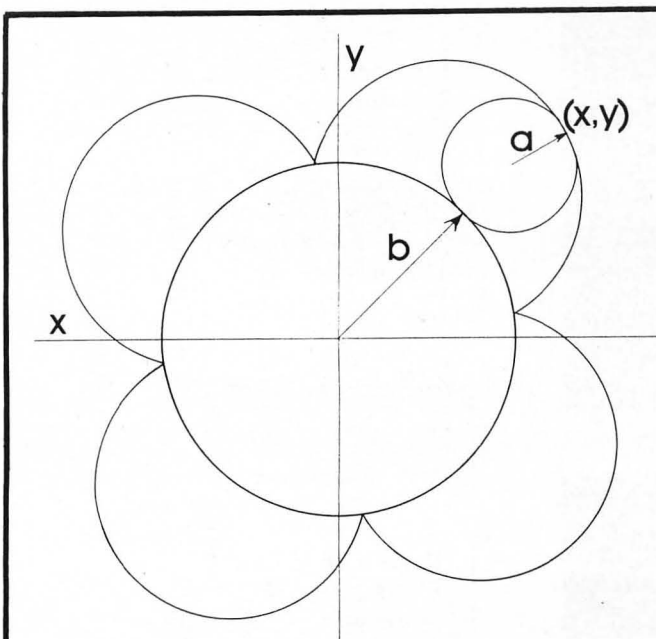


Figure 1

An added explanation of two terms unfamiliar to most:

An epicycloid is defined as the points generated from a point on a circle of radius "a" which is rolled around the outside of a circle of radius "b".

where:

$$x = (a + b) * \text{COS}(t) - a * \text{COS}((a + b)/a) * t$$

$$y = (a + b) * \text{SIN}(t) - a * \text{SIN}((a + b)/a) * t$$

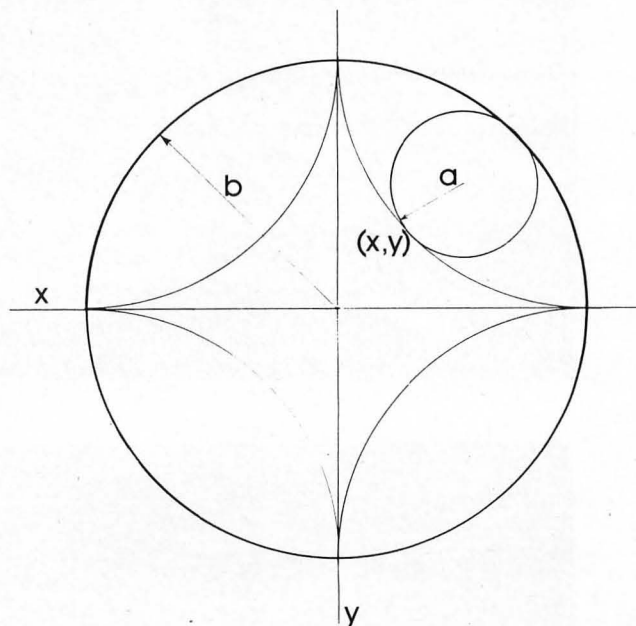


Figure 2

A hypocycloid of four points is defined as the points generated from a point on a circle of radius "a" which is rolled around the inside of a circle of radius "b" = 4 * a.

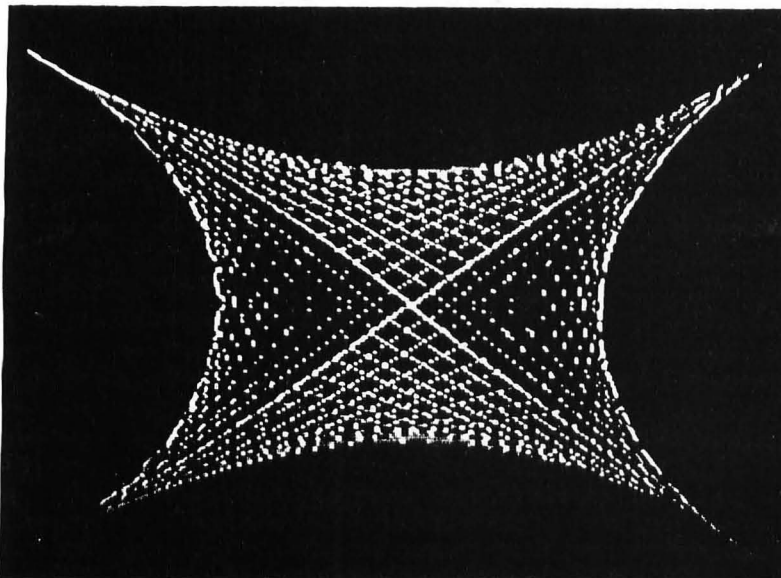
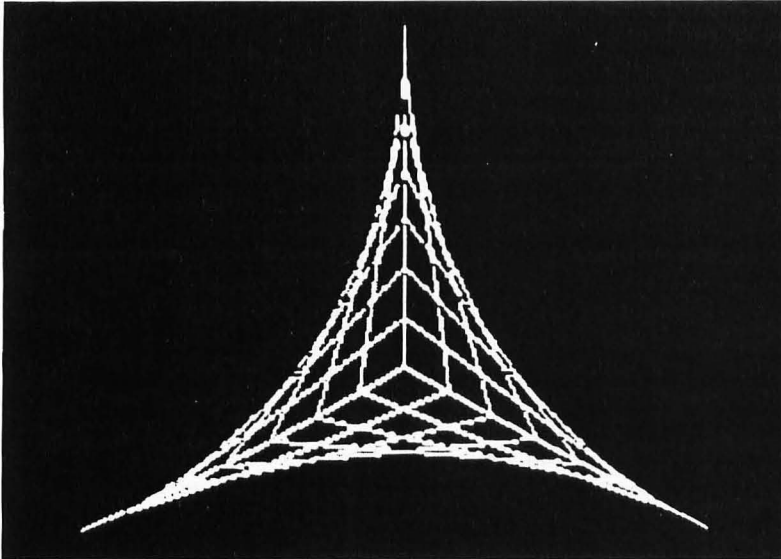
where:

$$x = a * \text{COS}^3(t)$$

$$y = a * \text{SIN}^3(t)$$

Additional examples of creating the many designs available:

Design	Parameters	Description
1	3, 103, 95 150, 159, 95	Odd-looking circle/ellipse Hold down "C" = Psychedelic Egg
2	1.01, 91, 5 1.001, 15, 20	Squares around Gun sight if you can wait
3	1.5, 10, 28 1.5, 10, 24	Pillow for the couch Fish
4	20 5	Nice to watch draw Real Star
5	3 20	Top of pyramid Nice curves
6	49, 2 20, 20	Porcupine Slow explosion
7	10, 10, 52 38, 29, 40	Apple on edge Pretty
8	10, 3 55, 4	Arrow head Nice Squares
9	20, 1, 60 40, 40, 110	Sunflower Two-headed star



Variables

A: Number of design (0-95); number of radii (100-195, 400-495); length of line multiplier (200-295); number of lines per cycle (300-395); number of lines per angle (500-595); number of points in star (600-695); size of cycloid (700-795); number of cycloids (800-895); number of pedals (900-995).

ANG: 120 degrees.

B: Number of degrees in DRAWTO (200-295); number width of design (300-395); counter for DRAWTO (400-495); indicator for odd/even number of points (600-695); distance from the center (700-795); number of lines per cyloid (800-895); number of lines per pedal (900-995).

C: Width of radius (100-195); length from center from first DRAWTO (200-295); degree phase shift on X axis (300-395); counter for DRAWTO (400-495); column (500-595); column limiter (600-695); number of lines per cyloid (700-795); size limiter (900-995).

CE1, CE2: Column endings.

D: Height of design.

E: Function of number of peaks.

F: Degree phase shift on Y axis.

LC: Increment length for column.

LR: Increment length for row.

N: Counter for radius from center (200-295); number of lines per star point (600-695).

R: Radius from center (200-295); row (500-595); row limiter (600-695).

RE1, RE2: Row endings.

T: Angle.

W: X coordinate for DRAWTO.

X: Column.

Y: Row.

Z: Y coordinate for DRAWTO.

Main menu selection.

10 REM LINE DESIGNS

15 DIM LC(50),LR(50):H=4

20 GRAPHICS 0:COLOR 1:SETCOLOR 2,0,0

25 PRINT " GEOMETRIC LINE DESIGN

S "

30 X=C*SIN(N*T)*COS(T)+160

35 PRINT " ENTER 5 TO STOP AFTER DISPLAY OF ";PRINT " DESIGN AND RETURN TO THIS MENU"

40 PRINT :PRINT " 1. CIRCLE AND ELLIPSES

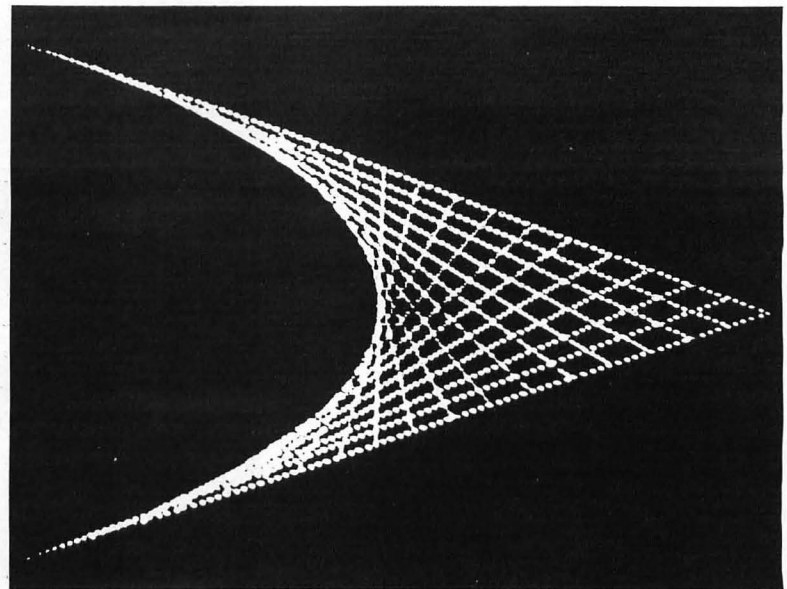
45 PRINT :PRINT " 2. SPIRAL CURVES

50 PRINT :PRINT " 3. OSCILLOSCOPE PATTERNS


```

55 PRINT :PRINT " 4. POLYGONS -ALL VER
TICES CONNECTED"
60 PRINT :PRINT " 5. ANGLES AND CURVES
"
65 PRINT :PRINT " 6. STARS AND CURVES
"
70 PRINT :PRINT " 7. EPICYCLOIDS
"
75 PRINT :PRINT " 8. HYPOCYCLOIDS WITH
4 POINTS "
80 PRINT :PRINT " 9. ROSE PEDALS
"
85 PRINT :PRINT " ENTER THE NUMBER YO
U WANT"
90 TRAP 995:INPUT A
95 ON A GOTO 100,200,300,400,500,600,7
00,800,900

```

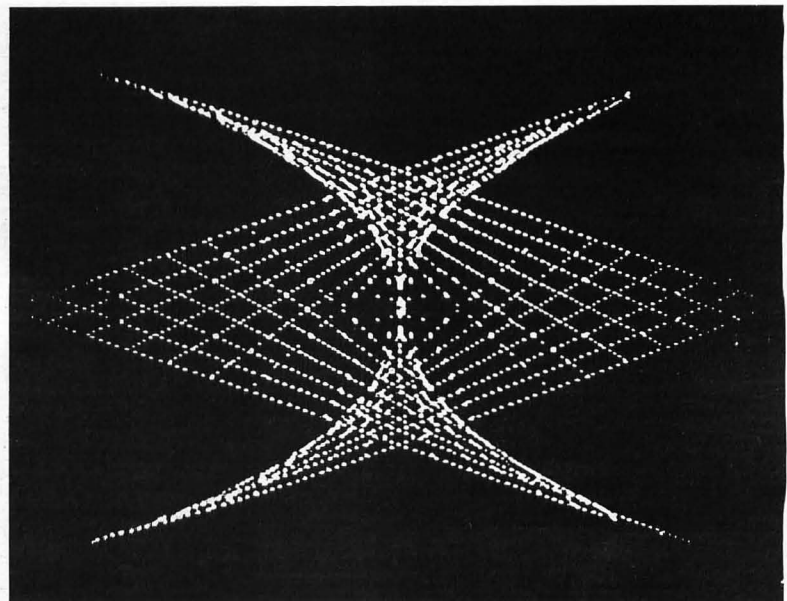
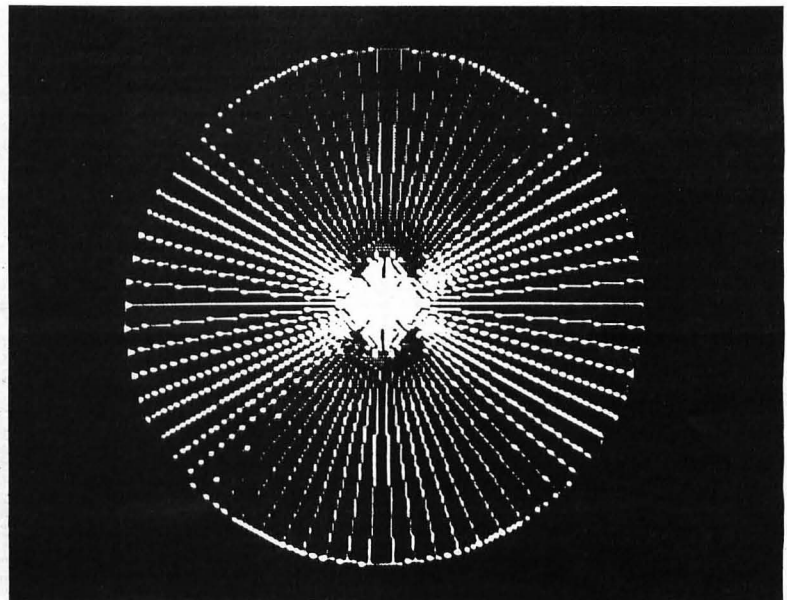


Circles and Ellipses.

```

100 REM CIRCLE RADII
105 GRAPHICS 0:COLOR 1:SETCOLOR 2,0,0
110 PRINT :PRINT " CIRCLE/ELLIPTIC
AL PATTERNS":? :? "ENTER 0 FOR DEFAULT
PARAMETERS -":PRINT " 70, 103, 90"
115 PRINT :PRINT :PRINT "HOW MANY SPOK
ES ON YOUR WHEEL?"
120 TRAP 995:PRINT " EXAMPLE 1, 2, 5,
10, OR 1.3, ETC.":INPUT A:IF A<>0 THE
N 130
125 A=70:B=103:C=90:GOTO 150
130 PRINT :PRINT "HOW WIDE THE WHEEL?"
135 TRAP 995:PRINT " A CIRCLE = 103,
MAX = 159 ":INPUT B
140 PRINT :PRINT "HOW TALL THE WHEEL?"
"
145 TRAP 995:PRINT " A CIRCLE = 90,
MAX = 95 ":INPUT C
150 GRAPHICS 24:COLOR 1:SETCOLOR 2,4,0
:DEG
155 ANG=360/A
160 FOR N=1 TO A:PLT 160,96:T=N*ANG
165 X=(B*COS(T)+160):Y=(C*SIN(T)+96)
170 DRAWTO X,Y:NEXT N
175 FOR N=1 TO A:PLT X,Y:T=N*ANG
180 X=(B*COS(T)+160):Y=(C*SIN(T)+96)
185 DRAWTO X,Y:NEXT N
190 GOTO 990

```



Spirals.

```

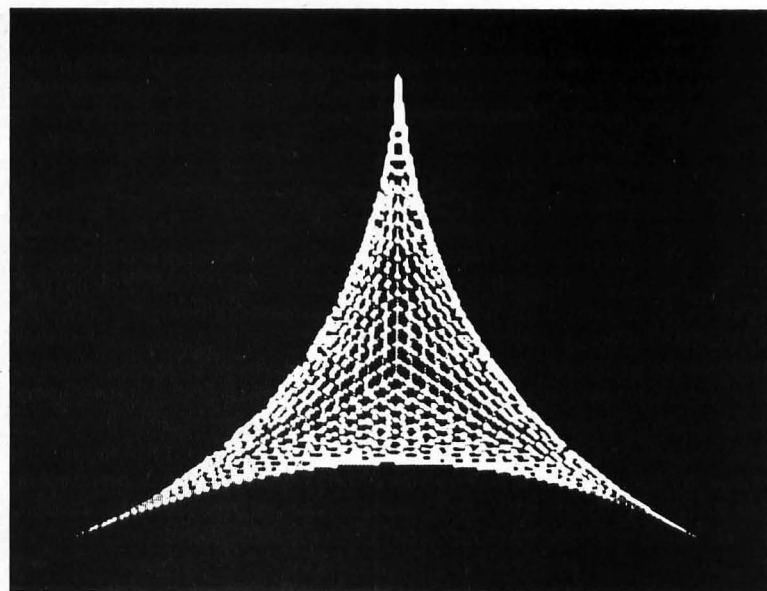
200 REM SPIRAL PATTERNS
205 GRAPHICS 0:SETCOLOR 2,0,0:COLOR 1:
DEG
210 PRINT :PRINT " SPIRAL PATTERNS"
215 PRINT :PRINT "ENTER 0 FOR DEFAULT
PARAMETERS -":PRINT " 1.007, 44, 10"
220 PRINT :PRINT "MULTIPLIER TO INCREA
SE THE LENGTH "

```

```

225 PRINT "OF EACH LINE DRAWN
"
230 TRAP 995:PRINT " EXAMPLE 1.1, 1.0
1, 1.05, ETC.":INPUT A:IF A<>0 THEN 24
0
235 A=1.007:B=44:C=10:GOTO 255
240 PRINT :PRINT "NUMBER OF DEGREES FO
R EACH TURN "
245 TRAP 995:PRINT " EXAMPLE 10, 20,
45, 110, ETC.":INPUT B
250 TRAP 995:PRINT :PRINT "LENGTH FROM
CENTER OF SCREEN ":PRINT "EXAMPL
E 1, 5, 10 ETC.":INPUT C
255 GRAPHICS 24:COLOR 1:SETCOLOR 2,4,0
:DEG
260 N=0:R=C:PLOT 160,96
265 R=A*R:T=N*B:N=N+1
270 X=R*COS(T)+90:Y=R*SIN(T)+160
275 IF X>191 OR X<0 THEN 990
280 IF Y>319 OR Y<0 THEN 990
285 DRAWTO Y,X:GOTO 265

```

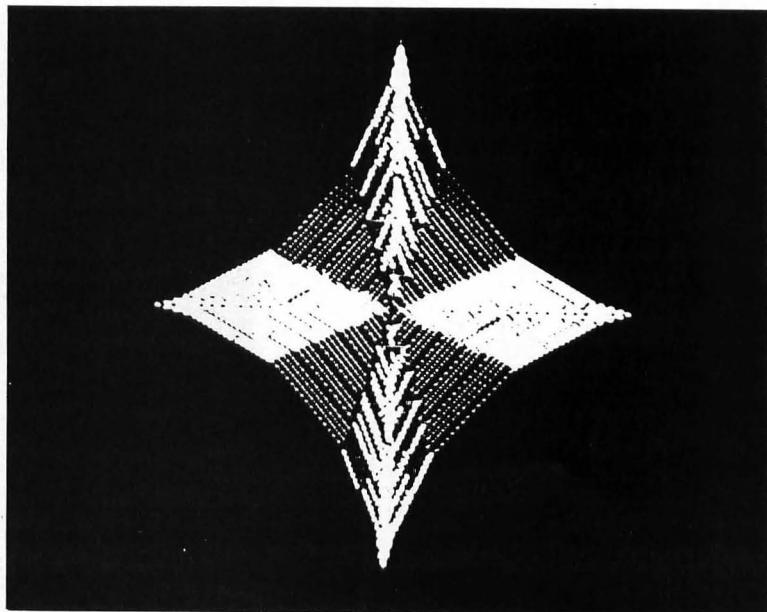
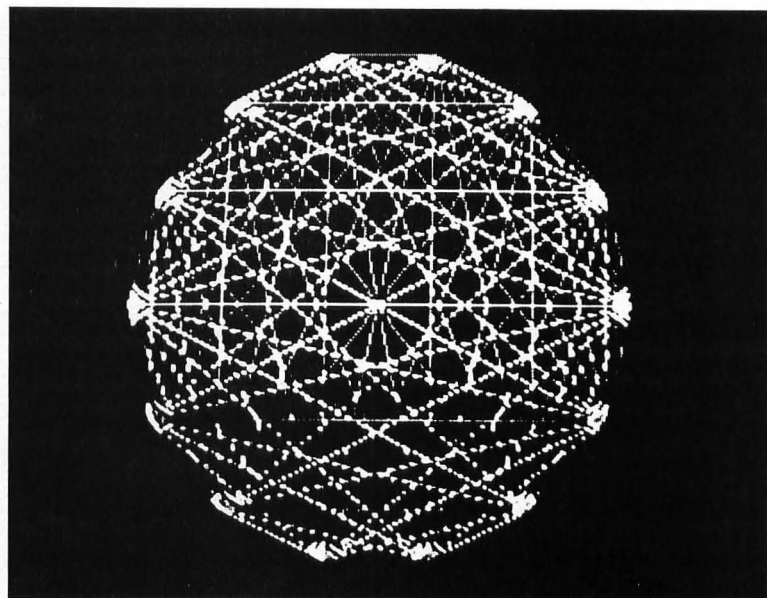


Oscilloscope.

```

300 REM *** OSCILLOSCOPE ***
305 GRAPHICS 0:COLOR 1:SETCOLOR 2,0,0
310 PRINT :PRINT " OSCILLOSCOPE PA
TTERNS ":? :? "ENTER 0 FOR DEFAULT
PARAMETERS -:PRINT " 15, 2, 90"
315 ? :? :? "A FUNCTION OF NUMBER OF P
EAKS ":PRINT " EXAMPLE 1, 2, 4, 10, 0
R .8, 1.5 ETC"
320 TRAP 995:INPUT E:IF E<>0 THEN 330
325 E=15:G=2:C=90:GOTO 350
330 PRINT :PRINT "SMOOTHNESS OF CURVE
"
335 TRAP 995:PRINT " EXAMPLE 1=SMOOTH
EST, 2,4,ETC.":INPUT G
340 PRINT :PRINT "DEGREE PHASE SHIFT 0
F X COORDINATE"
345 TRAP 995:PRINT " EXAMPLE 0, 30, 6
0, 90, ETC ":INPUT C
350 GRAPHICS 24:COLOR 1:SETCOLOR 2,4,0
:DEG
355 B=159:D=95:F=0:T=1
360 X=B*SIN(T+C)+160:Y=D*SIN(E*T+F)+96
:PLOT X,Y
365 FOR T=1 TO 10000 STEP G
370 X=B*SIN(T+C)+160:Y=D*SIN(E*T+F)+96
375 DRAWTO X,Y
380 IF PEEK(764)<>62 THEN NEXT T
385 POKE 764,33:GOTO 20

```



Polygons — all vertices connected.

```

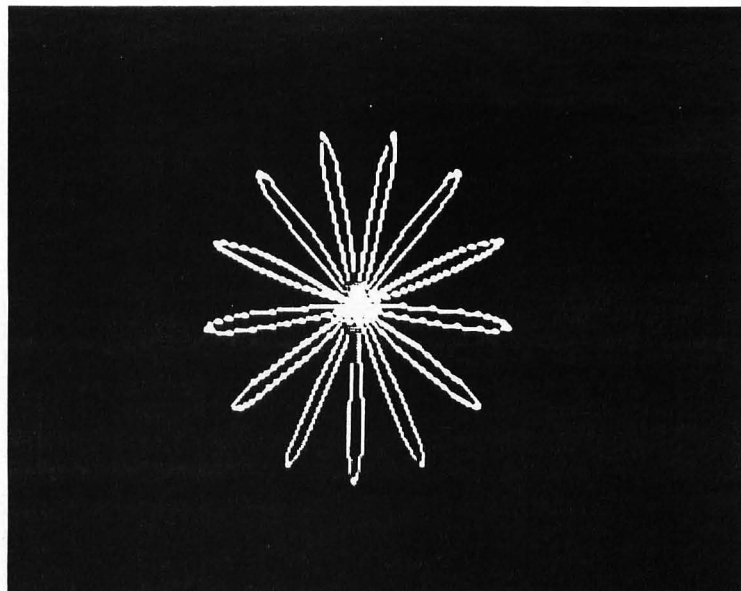
400 REM **** POLYGONS ****
405 GRAPHICS 0:COLOR 1:SETCOLOR 2,0,0
410 PRINT :PRINT " POLYGONS - ALL VE
RTICES CONNECTED":? :? "ENTER 0 FOR DE

```

```

FAULT PARAMETER -:PRINT " 14 "
415 PRINT :PRINT :PRINT "HOW MANY VERT
ICES DO YOU WANT?"
420 TRAP 995:PRINT " LESS THEN 20 IS
BEST
":INPUT A:IF A<>0 THE
N 430
425 A=14
430 GRAPHICS 24:COLOR 1:SETCOLOR 2,4,0
:DEG
435 ANG=360/A:C=2:FOR N=1 TO A:T=N*ANG
:X=101*COS(T)+160:Y=90*SIN(T)+96:PLOT
X,Y
440 FOR B=C TO A:PLOT X,Y:T=B*ANG
445 W=101*COS(T)+160:Z=90*SIN(T)+96:DR
AWTO W,Z
450 NEXT B:C=C+1:NEXT N:GOTO 990

```

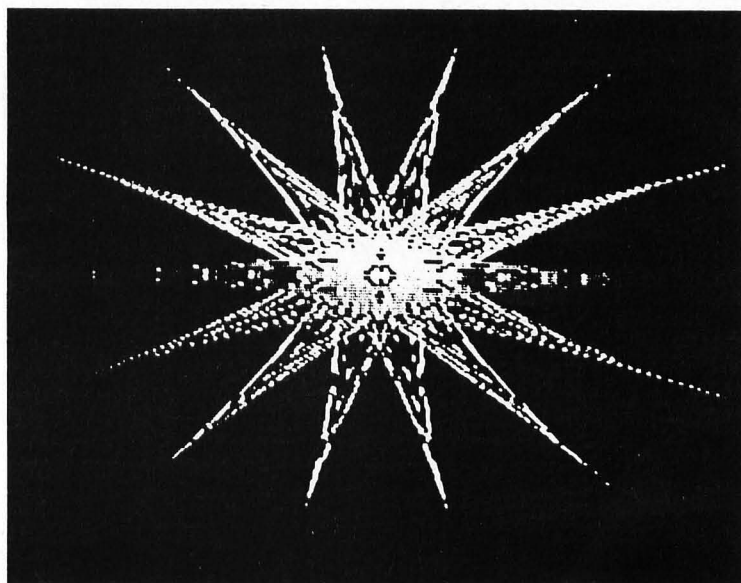
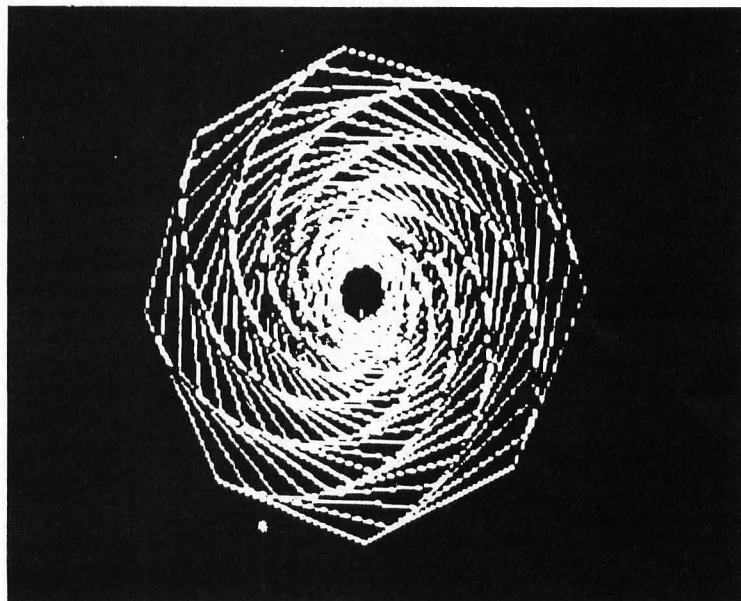


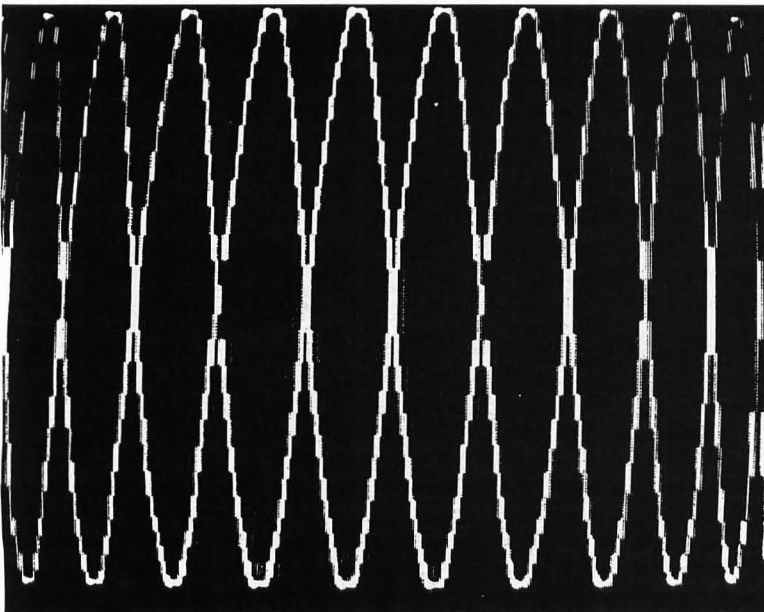
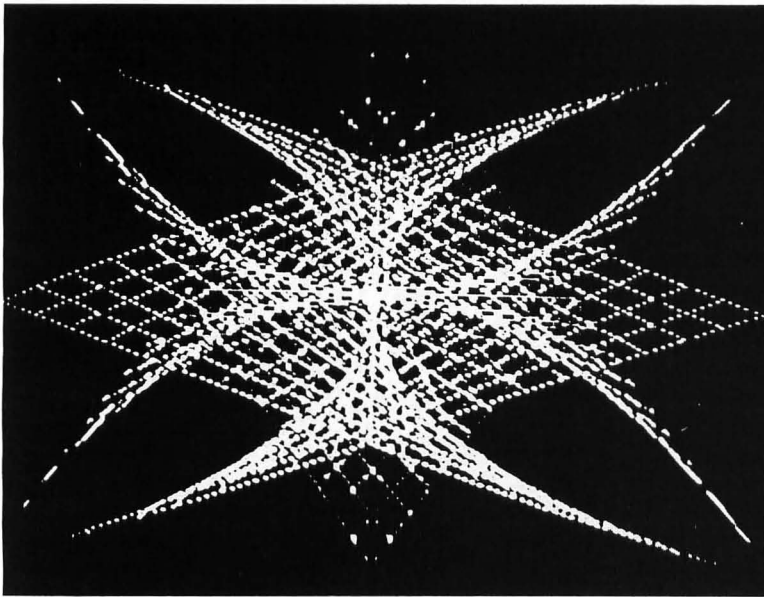
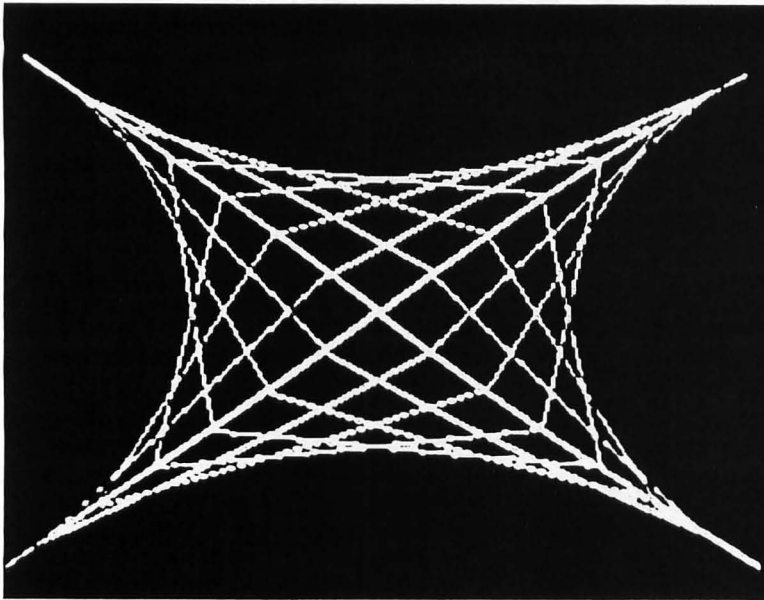
Angles and Curves.

```

500 REM *** ANGLES AND CURVES
505 GRAPHICS 0:COLOR 1:SETCOLOR 2,0,0
510 PRINT :PRINT " ANGLES AND CURVE
PATTERNS":? :? "ENTER 0 FOR DEFAULT
PARAMETER -:PRINT " 20 "
515 PRINT :PRINT "THERE WILL BE THREE
DESIGNS DISPLAYED"
520 PRINT " SO HOLD YOUR HORSES UNTIL
THEY ARE ":PRINT " ALL FINISHED"
525 TRAP 995:PRINT :PRINT :PRINT "HOW
MANY LINES TO FORM THE ANGLES?":INPUT
A:IF A<>0 THEN 532
530 A=20
532 REM *** 1ST DESIGN ***
535 GRAPHICS 24:COLOR 1:SETCOLOR 2,4,0
:DEG :B=150:C=118:ANG=120
540 FOR N=1 TO A+1:T=N*ANG+150
542 X=B*COS(T)+160:Y=C*SIN(T)+120
544 LC(N)=(160-X)/A:LR(N)=(120-Y)/A:NE
XT N
546 FOR J=1 TO 3:N1=A:FOR K=0 TO A
548 PLOT 160+LC(J)*K,120-LR(J)*K
550 DRAWTO 160+LC(J+1)*N1,120-LR(J+1)*
N1:N1=N1-1:NEXT K:NEXT J:GOSUB 598
552 REM *** 2ND DESIGN ***
554 GRAPHICS 24:COLOR 1:SETCOLOR 2,4,0
556 N1=A:C=4:R=94:LC=(310-C)/A:LR=R/A
558 FOR K=1 TO A+1:K1=K-1
560 PLOT C+LC*K1,R-LR*K1
562 DRAWTO C+LC*N1,R+LR*N1
564 N1=N1-1:NEXT K:GOSUB 598
566 N1=A:C=310:R=94:LC=(C-4)/A:LR=R/A
568 FOR K=1 TO A+1:K1=K-1:PLOT C-LC*K1
,R-LR*K1:DRAWTO C-LC*N1,R+LR*N1:N1=N1-
1:NEXT K:GOSUB 598
570 N1=A:C=155:R=4:LC=C/A:LR=(191-R)/A
:FOR K=1 TO A+1:K1=K-1
572 PLOT C-LC*K1,191-LR*K1:DRAWTO 2*C-
LC*K1,R+LR*K1:N1=N1-1:NEXT K:GOSUB 598
574 N1=A:C=155:R=4:LC=C/A:LR=(191-R)/A
:FOR K=1 TO A+1:K1=K-1

```





```

576 PLOT C-LC*K1,R+LR*K1:DRAWTO 2*C-LC
*K1,191-LR*K1:N1=N1-1:NEXT K:GOSUB 598
578 REM **** 3RD DESIGN ****
580 CE1=0:CE2=310:RE1=0:RE2=187:CE=CE1
582 GRAPHICS 24:COLOR 1:SETCOLOR 2,4,0
584 N1=A:R=94:LC=(CE-155)/A:LR=R/A:FOR
K=0 TO A
586 PLOT 155+LC*K,R-LR*K:DRAWTO 155+LC
*N1,R+LR*N1:N1=N1-1:NEXT K:IF CE=CE2 T
HEN 588
587 CE=CE2:GOTO 584
588 RE=RE1
589 LC=155/A:LR=(RE2-RE1)/(2*A):FOR K=
0 TO A:PLOT INT(155-LC*K),INT(95-LR*K)
:DRAWTO INT(310-LC*K),INT(RE++LR*K)
590 NEXT K
591 IF RE2<RE1 THEN 990
592 RE=RE2:RE2=RE1:RE1=RE:GOTO 589
598 FOR JJ=1 TO 200:NEXT JJ:RETURN

```

Stars and Curves.

```

600 REM *** STARS WITH CURVES ***
605 GRAPHICS 0:COLOR 1:SETCOLOR 2,0,0
610 PRINT :PRINT " STAR WITH CURVES
":? :? "ENTER 0 FOR DEFAULT
PARAMETERS -:PRINT " 14, 8 "
615 ? :? :? "HOW MANY POINTS IN THE ST
AR?":? " EVEN NUMBER MAKES N POINTS":
? " ODD NUMBER MAKES 2 X N POINTS"
620 PRINT " MAX = 49"
625 TRAP 995:INPUT A:IF A<>0 THEN 635
630 A=14:N=8:GOTO 650
635 PRINT :PRINT "HOW MANY LINES PER P
DINT?"
640 TRAP 995:PRINT " EXAMPLE 2,3,5,10
, < 30 IS BEST":INPUT N
650 GRAPHICS 24:COLOR 1:SETCOLOR 2,4,0
:DEB :ANG=360/A:B=1
655 FOR J=1 TO A+1:T=J*ANG
660 C=150*COS(T)+160:R=95*SIN(T)+96
665 LC(J)=(160-C)/N:LR(J)=(96-R)/N:NEX
T J:IF A-2*(INT(A/2))=0 THEN B=2
670 FOR L=1 TO A/B:N1=N:FOR K=0 TO N
675 PLOT 160+LC(L+1)*K,96-LR(L+1)*K
680 DRAWTO 160-LC(L)*N1,96+LR(L)*N1
685 PLOT 160+LC(L)*K,96-LR(L)*K
690 DRAWTO 160-LC(L+1)*N1,96+LR(L+1)*N
1:N1=N1-1:NEXT K:NEXT L:GOTO 990

```

Epicyploids.

```

700 REM *** EPICYCLOIDS *****
705 GRAPHICS 0:COLOR 1:SETCOLOR 2,0,0
710 PRINT :PRINT " EPICYCLOID PATT
ERNS. ":? :? "ENTER 0 FOR DEFAULT
PARAMETERS -:PRINT " 20, 32, 20 "
712 IF E=1 THEN PRINT " PARAMETERS TOO
LARGE, TRY AGAIN"

```

```

715 PRINT :PRINT :PRINT "DISTANCE OF C
YCLOID FROM CENTER?"
720 TRAP 995:PRINT " EXAMPLE 5,10,20
,30, < 50 IS BEST ":PRINT "
":INPUT B:IF B<>0 THEN GOTO 730
725 B=20:A=32:C=30:GOTO 750
730 PRINT :PRINT :PRINT "SIZE OF CYCLO
ID?"
735 TRAP 995:PRINT " EXAMPLE 5,10,20,
< 50 ALSO IS BEST":INPUT A
740 PRINT :PRINT :PRINT "SMOOTHNESS OF
LINES PER CYCLOID?"
745 TRAP 995:PRINT " EX 1=SMOOTHEST,
2, 5,10, 20, 50 ETC":INPUT C
750 GRAPHICS 24:COLOR 1:SETCOLOR 2,4,0
:DEG :T=0:E=0
755 X=(A+B)*COS(T)-A*COS(((A+B)/A)*T)+
160:Y=(A+B)*SIN(T)-A*SIN(((A+B)/A)*T)+
95
760 PLOT X,Y:FOR T=0 TO 50000 STEP C
765 X=(A+B)*COS(T)-A*COS(((A+B)/A)*T)+
160:Y=(A+B)*SIN(T)-A*SIN(((A+B)/A)*T)+
95
767 IF X<320 AND X>0 AND Y>0 AND Y<192
THEN 770
768 E=1:GOTO 700
770 DRAWTO X,Y
775 IF PEEK(764)=62 THEN 990
780 NEXT T:GOTO 990

```

Hypocycloid with four points.

```
800 REM ** HYPOCYCLOIDS ***
```

```

805 GRAPHICS 0:COLOR 1:SETCOLOR 2,0,0
810 PRINT :PRINT " HYPOCYCLOIDS
":? :? "ENTER 0 FOR DEFAULT P
ARAMETERS -:PRINT " 25, 20 "
815 PRINT :PRINT :PRINT "NUMBER OF CYC
LOIDS?"
820 TRAP 995:PRINT " EXAMPLE 1, 2, 5,
10, 20, MAX=95 ":INPUT A:IF A<>0 THE
N 830
825 A=25:B=20:GOTO 850
830 PRINT :PRINT "HOW MANY LINES PER C
YCLOID ?"
835 TRAP 995:PRINT " SMOOTHNESS, EX.
20=SMOOTH, 5=ROUGH":INPUT B
850 GRAPHICS 24:COLOR 1:SETCOLOR 2,4,0
:DEG :ANG=360/B
860 FOR K=1 TO A
861 C=COS(ANG):R=SIN(ANG):M=95/A
863 PLOT K*M*C*C+160,K*M*R*R+95
864 FOR N=0 TO B:T=N*ANG
865 C=COS(T):R=SIN(T):M=95/A
870 DRAWTO K*M*C*C+160,K*M*R*R+95:
NEXT N:NEXT K
875 GOTO 990

```

Rose petals.

```

900 REM ** ROSE PEDALS ***
905 GRAPHICS 0:COLOR 1:SETCOLOR 2,0,0
910 PRINT :PRINT " ROSE PEDALS
":? :? "ENTER 0 FOR DEFAULT
PARAMETERS -:PRINT " 8, 2, 90 "

```

```

915 PRINT :PRINT :PRINT "NUMBER OF PED
ALS?"
920 TRAP 995: ? " ODD NUMBER PRODUCES
N PEDALS ":? " EVEN NUMBER PRODUCES
2 X N PEDALS":INPUT A:IF A<>0 THEN 930
925 A=8:B=2:C=90:GOTO 950
930 PRINT :PRINT "HOW MANY LINES PER P
EDAL?"
935 TRAP 995:PRINT :PRINT " EX. 1=SMO
OTHEST, 2,5,10- ROUGHER ":INPUT B
940 PRINT :PRINT "HOW LARGE IS YOUR FL
OWER?"
945 TRAP 995:PRINT " MAX SIZE IS 95
, WHICH IS BEST ":INPUT C
950 GRAPHICS 24:COLOR 1:SETCOLOR 2,4,0
:DEG :D=360
952 PLOT 160,95
955 IF A-2*(INT(A/2))=1 THEN D=180
960 FOR T=0 TO D STEP B
965 X=C*SIN(A*T)*COS(T)+160:Y=C*SIN(A*
T)*SIN(T)+95
970 DRAWTO X,Y:NEXT T:GOTO 990
990 IF PEEK(764)<>18 THEN 993
991 H=H+1:POKE 710,H*16:IF H=14 THEN H
=-1
992 POKE 764,255:GOTO 990
993 IF PEEK(764)<>62 THEN 990
994 POKE 764,255:GOTO 20
995 TRAP 40000:GOTO 256*PEEK(187)+PEEK
(186)

```

Bugs, Worms

and other undesirables



In the TRS-80® version of *Melody Dice* (November 1981), line 7062 should read as follows:

```
7062 RESUME 5090
```



Kidnapped has a few problems too. Here is a line to insert in the program to fix the jump command:

Apple version:

```
805 IF A<>51 THEN 730
```

Atari version:

```
2075 IF A<>51 THEN 2040
```



The memory requirements for Quest 1 are not accurate. The tape version needs 32K and the disk version 40K. For those with 24K cassette systems the program will load, but may crash with an Error 2 (memory insufficient) or just hang up. To fix this, load in the program and delete lines 2 and 3. Change line 1 to read:

```
1 GOSUB 30000:OPEN#3,4,0,"K"
```

This will give you a few bytes to spare for future enhancements.

In the Atari program *Music Programmer* (November, 1981), page 72 should be where page 74 is, the next to the last page in the program.

Atari One Liners

```

1 ? " ":POKE 752,1:POSITION 2,RND(0)*1
8: ? " " " " " ":FOR X=1
TO 30: ? " " :SOUND 0,99,6,15-X/3:NEX
T X:POKE 77,128:GOTO 1
2 REM Esc-Ctrl keys for above quotes:
3 REM 1st: "<"
4 REM 2nd: "==" FMMGNNN"
5 REM 3rd: "ZR.RRR.C"
6 REM 4th: ">->"

```

Sheldon Leemon
14400 Elm
Oak Park, MI 48237

```

1 N=PEEK(560)+256*PEEK(561)+3:POKE N,1
7:POKE 19,90: ? CHR$(125);"22222222222
10"

```

Tilden A. Smith,
Casilla 101-D,
Punta Arenas, Chile



VOYAGE OF THE VALKYRIE

By Leo Christopherson (Advanced Operating Systems) System requirements: 16K TRS-80® Model I or III; or 48K Apple II or Apple II Plus with paddles or a joystick. Suggested retail price: tape — \$34.95, disk — \$39.95

Reviewed by Alan J. Zett

Recipe: Take one part adventure, one part war game, one part arcade shoot-em-up, and one part graphics and sound a la Leo Christopherson. Combine these into one action-packed simulation and you have *Voyage of the Valkyrie*, a hybrid BASIC and Machine Language program of exceptional quality.

If you are an adventurer, war gamer, arcade freak, or even a computer neophyte, this game has something to hold your interest. The combination of sound and graphics blend together with the smoothness and style that Leo Christopherson has mastered so well.

At the start of the game, you will find yourself piloting the attack ship "Valkyrie," hovering over a mountain peak that serves as a refueling base. The peak is the only safe place on the entire island for the moment. Your goal is to conquer the imaginary island of Fugloy. (Fugloy is Norwegian for "bird island.") The island is mapped out in an adventure-type layout with "rooms" and directions in which you can travel. You can win by capturing the ten cities that constitute the island kingdom and by stocking up on the wealth they contain. Each of these cities has a castle that is protected by a fleet of birds which constitute the Fugloy Air Defense. The birds are drawn in finely detailed graphics and have a number of life-like mannerisms, especially the way they flap their wings.

The overall movement and actions are classic examples of the quality present in all of Leo Christopherson's work, but I think he has really gone all-out this time and produced a fine piece of graphics animation. Even the rooms in this adventure are displayed with graphics. There are rooms with mountains, ocean views and castles. In order

to defeat a city, all of the birds must be killed or made to surrender. This is accomplished by aiming the cross-hairs of your ship's weapon at the birds' midsections and shooting them down as they fly; not an easy task when they are firing back!

There are five different types of birds. Each has a different level of fighting ability and a unique method of attack. Hawks and Ospreys are easy to kill, Condors are medium, but Falcons and Eagles are very hard. In addition



to the birds that defend the cities, there are several Eagles that roam the map in search of you. They are known as the Fugloy Air Force. They love to ambush you, and they will frequently hide in a location directly between you and a refueling base.

In addition to sound effects there are also musical selections from Richard Wagner's operas, which can optionally be turned off in the interest of quickening the pace of the game. For instance, *The March of the Tannhauser* starts off the game, followed by *The Ride of the Valkyrie* every time you capture a castle. If you become careless you will hear another selection from *die Walkure*, but the game will be over. If you should win, you are rewarded with the *Prelude from Act III of Lohengrin*. All music is done extremely well using a multiple-voice technique that sounds as if two notes are being played at the same time. The Apple version has

stereo sound effects: Music is played through the cassette port and through the internal speaker, giving four-voice sound.

All this sounds interesting (pun intended), but the best part is in the play of the game. There is a definite strategy required to play this game and win. For example, every time the birds attack, they fire bolts of energy at you. Each shot drains your shields according to the type of bird which is attacking. Your shields will recharge, but only at a set rate. If you deplete your energy supply, the shields collapse, the ship stalls, and you die. Keeping a supply of energy is just as important as capturing a city; in fact, it may be the only way of surviving an attack when a battle is in progress. When your energy levels drop too low, it is necessary to return to a refueling base to replenish the spent fuel. If you have not established a second refueling base, you must return to the mountain peak to refuel. So you can see that the location and amounts of your energy resources determine how far you'll get in the game.

This is where the strategy comes into play. When you defeat a castle, you are awarded all the gold contained in the castle. This is also the means of getting a promotion. Rank and wealth, along with energy reserves, are crucial to your survival in this fantasy world. But if you force the castle you are attacking to surrender, you receive no gold. Instead, the city is turned into a refueling base. Proper planning of which castles to use as bases and which to plunder may well decide the outcome of the game. In addition, when you defeat the roaming band of Eagles, the map location in which the battle occurs is also turned into a refueling base. When your rank advances to Colonel, the Air Force Eagles will attack you whenever you hover over a deserted location. By using this method you can choose the place to defeat them.

The Apple version has a few unique features that are not present in the TRS-80® version. In addition to stereo sound, there are provisions for saving a game in progress; and when the island is totally subjugated, the Apple will treat you to a Hi-Res light show!

FUGLOY BIRDS BIRDS DOWN: BIRDS LEFT: FUGLOY AIR FORCE EAGLES:		TOTAL ENERGY	
FIGHTING ABILITY HAWKS: LOW OSPREYS: LOW CONDORS: MEDIUM FALCONS: HIGH EAGLES: HIGH		STATUS REPORT: MAP LOCATION: 45 VIEW: NORTH WEAPONS: OFF	
ANGREP CONDORS DRAGE OSPREYS FRYKT FALCONS GEVAER EAGLES JAMMER HAWKS LUFTIG FALCONS NEDGAA OSPREYS TORDEN CONDORS VAKKER HAWKS AEKEL HAWKS		PROGRESS REPORT: GOLD	
SELECT LEVEL: (EASY) 0-1-2-3-4-5-6-7-8-9 (HARD)			

Above: The player's home base, as represented in the TRS-80® version of *Voyage of the Valkyrie*. From this castle the player can monitor the progress of the game. The center column shows how many birds have been killed. The upper left quadrant gives you vital information concerning the Fugloy Air Force, whose birds will ambush you on your way to a refueling base. The other quadrants display status information relative to one's play.

You may be saying to yourself, "So, it's great. So what, it's still an adventure. When you're done playing it, it's useless!" I say, "Not so!" There are ten levels of difficulty in this game. Mapping out the adventure area is a required part of the game before you can start a serious attempt at winning. There are even blank mapping sheets included, to aid you in drawing the map (which, when finished, may be sent to Advanced Operating Systems in return for a full-color wall poster entitled "Valkyrie"). But the continuing challenge lies in the varying numbers of different obstacles to be overcome; the variety of their types and locations require a different winning strategy for each game. Each game is a test of your arcade battling skills, logical deduction, and reflexes. No two games are the same. For this reason alone it is more than an adventure. The action is constant, and the pace is fast. It is the type of game you can play for hours on end and lose all track of time. It's like a good novel that you just can't put down. Boredom never has a chance to set in.

The only gripe I have about this program is that it is virtually impossible to play the TRS-80® version without a numeric keypad. Here the author really struck out. You'd have to be an

octopus to fight the fleet and win. He uses the arrangement of the numbers on the keypad to denote the direction of the cross-hairs for aiming. Trying to remember this when using the top row of numbers on a normal keyboard is extremely difficult. If there had only been a provision for using a similar arrangement of letter keys or even making use of the arrow keys, I would have given the program an A+. Instead, I must give it an A — as in adventuring, astounding, and amazing.

RACE FOR MIDNIGHT

from Avant-Garde Creations. System requirements: Apple II Plus with 48K RAM (Applesoft ROM), and disk drive. Suggested retail price: \$29.95

Reviewed by Ulf Lindmark

It was close to midnight, the hair on my arms was growing, an essential ingredient of the potion was missing, and I was stuck in a room with an ogre. What to do? I simply typed SAVE GAME, and then QUIT. I was playing *Race for Midnight*, a new Hi-Res adventure just released by Avant-Garde Creations.

This adventure has a very interesting

goal. The player must find a way to make a potion to prevent him from becoming a werewolf and there is a time limit in which the player must complete the necessary tasks. The scenario is fully explained in the instructions, which are more complete than those in many other adventures.

Now, to the game itself. Each room is very quickly drawn in beautiful full color. The monsters and other objects were well drawn. Some monsters were humorous, others menacing. *Race for Midnight* also has a feature that I have rarely seen in adventure games: sound effects. There are a lot of them, ranging from a screaming noise when you kill a monster to a rendition of a funeral march when you are killed. The sounds are not just limited to combat, though. There are many others too. Another "special effect" that I really like is when you kill a monster. The picture of the room turns a reddish color, which is really eerie.

The text portion is very good also. Pressing the RETURN key toggles between text and graphics. In the text mode, you see a room description at the top of the screen, followed by your last several commands. It's very similar to Scott Adams' split-screen style.

The program accepts the standard two-word commands. The vocabulary is quite sufficient, although the author can not anticipate every possible response. Most commands had at least two possibilities; for instance, SEARCH and EXAMINE. This held true for the more esoteric verbs as well. Another thing adventurers may be glad to hear is that this is really not a linear adventure. This means that you do not have to solve a really difficult puzzle just to go to a different location. At certain points, going in the wrong direction can lead you off into a completely different part of the adventure. There is no set "path" that you must follow.

Of course, there are one or two minor quirks. Occasionally, the program misses filling a small corner of the screen, but the picture still looks fine. The only other problem that I found was that the program stopped once or twice for about 15 seconds. The instructions said that this would happen, however, so I was not worried.

If you enjoy good Hi-Res graphics, unusual sound effects, or exciting adventures, I would suggest *Race for Midnight*. Its price, \$29.95, is about the same as other Hi-Res adventures presently available for the Apple. I hope that the author produces another adventure soon; I really enjoyed the 14th century.

MISSILE COMMAND

from Atari, Inc. System requirements: Atari 400 or Atari 800 with 8K RAM. Uses cartridge, one or two joysticks. Suggested retail price: \$39.95

Reviewed by Dean F. H. Macy

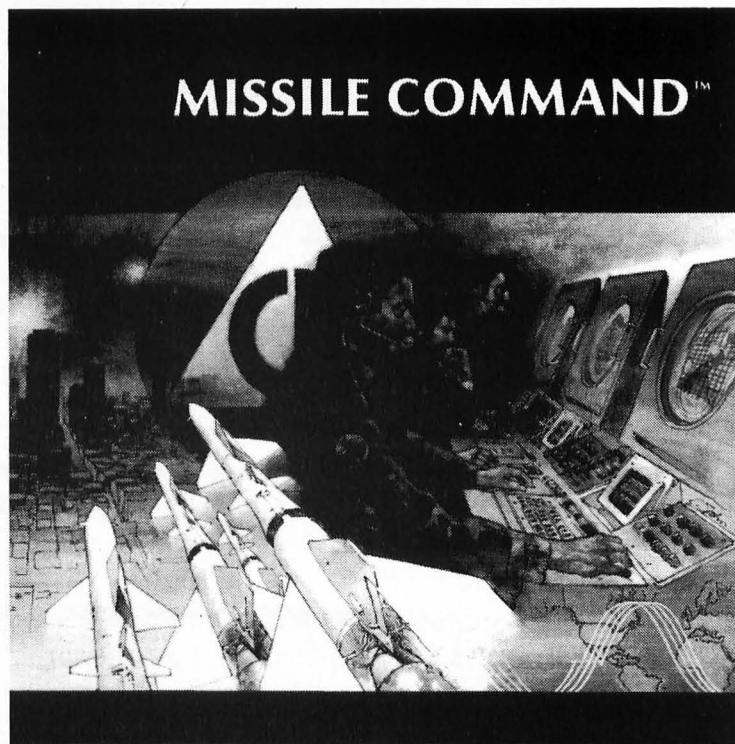
As missile base commander, you occupy the "hot seat" in Missile Control. For weeks now, Satellite Surveillance has been sending you photographs showing unusual arms buildup and launch base activity in enemy territory. Today, the analysts assure you, the enemy is in battle-ready posture. "Could they be mistaken?" you wonder. "Could this buildup really be exercises? War games?" Don't kid yourself, this is the real thing! It's only a question of time now before the enemy launches ICBMs targeted for the remaining free cities of the world. "How much time?" you ask yourself. "Will it be tonight?" Instinctively your hand moves to the hot line. "Ring, blast you! Get it over with!"

"Better cool it. Think about defense preparations," you tell yourself. When the attack comes you'll have only seconds to ready your antiballistic missiles. The enemy is powerful. Intelligence has kept you posted on the enemy's secret stockpiling of ICBMs in violation of international agreements. His long-range bombers and so-called "killer" satellites aren't news to you either. No, your real . . . (say it) . . . fear is that the reports of enemy smart missiles may be true. Those treacherous babies can dodge antiballistic missiles and home in on the target. What are the chances the enemy will drop smart missiles? "Pretty good," you think. "Pretty good!"

The situation is desperate. You know it. You're going to have to rely more on tactics than sheer fire power. You'll need all your cunning, all your courage, and more than a little luck. You're suddenly aware you're sweating and freezing at the same time. How long now? How much longer?

"RED ALERT! RED ALERT!" Screams the hot line. The sudden shock dislodges fear, starts the adrenaline flowing. You lift the receiver and hear "Enemy ICBMs on a heading for . . ." That's it! Move! You press a button on the console in front of you. The base alarm shrieks the alert. The ABMs are in place now. Seconds to go. "Stand by! Ready to launch!" Enemy on the monitor screen. "Fire!"

Missile Command is the newest in Atari's line of game software using superior graphics and sound and requiring skill on the part of the players.



How much skill? After playing 11 games for three hours (it seemed like a week) I managed to rack up 48,655 points. Not being much of a joystick expert, I was pleased. Then I read page 16 of the manual which states, "Every 10,000 points earns you a bonus city and a musical salute. Your bonus city is placed on a vacant site chosen at random . . . Bonus cities will be reserved for you until your score reaches one million points, at which time you lose all cities held in reserve. After all, such a talented player really doesn't need bonus cities." Obviously Atari expects players to exceed one million points. Fat chance!

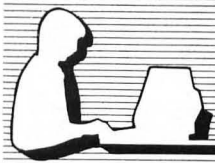
Upon power-up *Missile Command* is similar to the arcade version: three cities, a missile base, and three more cities sit at the screen bottom. The display at top shows highest score and game score. Attacks come in waves, with successive ones more difficult and challenging. On the sixth wave the enemy launches smart missiles which tend to bounce off heat waves from your ABMs and home-in on your cities and base. Only nearly direct hits effect them.

You can lose three cities per wave and your missile base with direct hits. To protect your cities you must judiciously use the 30 ABMs per wave assigned to your care. After you have used 24 ABMs your aiming cursor shrinks letting you know there are only six missiles left. At the end of a wave your score, if any, depends on the number of ABMs and cities remaining. If you have no cities left the screen displays "THE END".

Atari has made provisions for several game options. One or two players can match skills against the computer enemy. If you enjoy fighting smart missiles you can elect to dispense with other forms of oblivion and just shoot the smarties. Perhaps the expert player wants to skip waves one or two or more at the beginning. He would enter the number of skips and get annihilated faster. Players can also opt for no bonus cities. Pick your poison. There is a very good option I use. The player can press any key and freeze all action. Very handy when the telephone rings or you need a snack. Another key starts the action.

One operating hint for a high cheating score. If you elect to skip nine waves and shoot only smart missiles and you are an expert with a joystick, you can rack up points very quickly. Even I scored 78,000 this way. I allowed my wife (forced is a better word) to play *Missile Command* shooting only smart missiles and she scored over 9,000. (She said the game was a frustrating and nerve-racking experience.)

As a game requiring steady hands and eyes, and for sheer excitement and entertainment, I recommend that you rush out and purchase *Missile Command*. The first player to send me a photo of himself or herself alongside a CRT screen displaying over one million points will receive a prize from the *SoftSide* Library. Address your entries to Library Editor, *SoftSide* Magazine, 6 South Street, Milford, NH 03055. Get ready! Aim! Fire!



My Side of the Page

by Lance Micklus

GETTING A BIT SERIOUS — PART 6

The saga of *The Mean Craps Machine* continues. . . If you remember, I had said that the program had just been sent down to Adventure International for evaluation. Some time ago, I told you that it can take twice as long as forever to get a product out to the market after it's finished because of all of the delays down at the distributor. Well, this is a perfect example. I don't mean to imply anything bad against Scott Adams. All I do mean to say is that taking forever seems to be a way of life.

Also, I did not mean to put Scott Adams on the spot — although I freely admit that I did and I apologize for that. I wanted Scott to make his decision freely without feeling that he might embarrass me should he say YUK instead of YOHO to *The Mean Craps Machine*.

There were a few things I didn't share with you last time because I wanted to save them for now. Actually, the real benefit of writing *The Mean Craps Machine* will come regardless of what Scott Adams does. YUK or YOHO, *The Mean Craps Machine* had to be written even if nobody wanted it.

The reason is that people learn things much better when there is a goal. The best way to learn how to program a computer is to decide on a program as a project and then write it. It takes time and money to learn how to be a good programmer. When you have somebody working for you who is very talented, you have to invest time and money in that person to develop his or her skills.

In my office, that person is Diane Bernard, who we affectionately refer to around here as Diane #1 because she has only one N in her name. This avoids confusion with Dianne #2, whose last name is Micklus. Diane (with one N) has a lot of experience with mainframe computers but much less on TRS-80® machines. Her only claim to fame is that she is the person who modified my *Deluxe Personal*

Finance program so that it would run on the Model II. Making *The Mean Craps Machine* run on the TRS-80® Color Computer was an investment on my part to learn the machine. However, I let Diane do the Model I/III modifications so that she could get some experience with fast graphics and Machine Language. It is the learning experience that is the pay-off to *The Mean Craps Machine*. If Adventure International decides to market the thing and we make some money on it, all it means is that besides getting a more highly-skilled employee, I will also be able to offset my training expenses. Indeed, the difference between making or losing money often rests with your ability to make things you had to do anyway, also pay off.

There was a very good reason why I had to improve Diane's skills in the fast graphics and Machine Language area. The reason is called *Star Trek III* for the Color Computer. *Star Trek III* is one of my most profitable programs — having sold over 6,000 copies to date. Based on its track record, it's the kind of a program that's well worth a large investment in time, money, and training. Although, to be quite honest with you, I underestimated the amount of effort it was going to take to convert it over to the Color Computer.

In the first place, I didn't know too much about the 6809 and Diane didn't have very much Machine Language experience with any chip. That much I had taken into account. Where I miscalculated was in the limited size of the 16K Color Computer with Extended BASIC. At first, I thought we could cut *Star Trek III* down to make it fit. But after Diane had worked on it full-time for almost three weeks, it was clear that it wasn't going to work that way.

We now had three choices. First, we could strip it down and make it a *Star Trek II½*. I didn't like that idea because I think a poorly-written program is worse than no program at all. Second, we could just give up and say it can't be done. I didn't like that because I felt we had already put too

much time and effort into it to just throw it away. Third, we could make a 32K version and hope that there are enough extra memory machines out there to make the thing worth selling. After thinking it over, I sat down with Diane and laid it on the line.

My decision was to go with a 32K version. Since there weren't going to be very many 32K Color Computers, the finished product had to be better than the usual good — it had to be outstanding. It had to be so good, that even people with 16K machines would be willing to plunk down \$100 and void the warranty on their machines just to be able to play this one game. In short, this must be a game that every Color Computer owner must have, no matter what the cost, no matter what it takes to upgrade the machine so it will run.

That's a tall order.

Poor Diane worked and slaved over this thing all summer. At first I think she thought that I had given her a project that was beyond what she could handle. But she never felt that it couldn't be done — only that it was going to be very difficult. That's a key point I'll get back to later. Another thing Diane had to get used to is the idea that some experimenting and failure is necessary in order to achieve success. In fact, sometimes your failures can turn out to be very successful.

One incident that comes to mind is the warp out effect. Diane was having a lot of trouble understanding the 6809 at first. Her original warp effect Machine Language program didn't warp out. It just made an effect that looked like a confetti snow storm. When I saw it, I jumped up and down and said, "Fantastic!!" Poor Diane thought I was nuts until I suggested that she save her routine and use it for a special effect when the short-range sensors fail. Had it not been for a stupid mistake, we would have never thought of this idea and a great effect would have been lost.

Some other mistakes also paid off. The shaking of the ship after it has

been hit made an unusual line in the display. It looks like an invisible wave that moves back and forth across the image of the "Enterprise." It is the result of the frequency of the shifting image and the interference of the television sync frequencies. Just an accident, but it looks terrific. (Maybe I shouldn't be telling all of you about this. Then you'd think we planned it and would praise us for being so smart instead of lucky).

One of the hardest effects we had to work on was the crash sound. We tried all kinds of things for two weeks and came up with nothing even close. Finally, we took apart *Star Fighter* on the Model I to see how the author made his crash. Boy, you should have heard me swear at myself for being so stupid. What we really needed was a random number generator. It had never occurred to me that the Color Computer (Model I also) has a ROM with 12,000 random numbers in it. All you have to do is pick a starting address and you get a whole bunch of numbers in no particular order — a very clever idea I had never run across before.

Luck was definitely on our side. By the time *Star Trek III* for the Color Computer was done, Radio Shack announced that they would now offer a 32K upgrade. Perfect!!!

Unlike *The Mean Craps Machine*, I know I'll have no trouble getting somebody to market *Star Trek III*. I am totally pleased with the finished product and I really do believe that any Color Computer owner will have a hard time passing up this one. Now that we're done, we've finally figured out how to make a *Star Trek II* for 16K Color Computers. Originally, I didn't want to consider such a program because I didn't want a stripped-down *Star Trek III*. Now we know how to produce a variation of our *Star Trek III* that will still be a quality product so you don't feel you're getting half of a program. It will be a slightly different game, but a good one.

What this all means is that for all of our efforts, we end up with not one, but two *Star Trek* games. Just that alone makes it worth the effort. But if we can also end up marketing the *Craps* game, then that would be a third game which would really make all the effort worthwhile.

(NOTE: At press time, Adventure International had decided to market the Color Computer version of *The Mean Craps Machine*. The Model I/III version may be offered later, although

Lance Micklus, Inc. will sell Model I, Model III, and CC versions directly, themselves.)

My Speech to Prospective Employees

Like every employer, I get people who call looking for a job as a programmer. If I run across somebody interesting, I have a little speech I give them and I'd like to share it with you. Actually, it's simply a list of things I strive for and that I expect the people who work for me will strive for also.

I might start my speech by telling of an incident that happened a few months ago. My supply of pads for jotting down phone messages ran out. I went to my office supply store here in Burlington, VT, to pick up some more but they didn't have any more of the kind I usually use. So, I bought a different brand.

The first time I used them, I discovered something interesting. There was a place for the caller's name, address, the time they called, who took the message, and whether or not I'm supposed to call them back. Nowhere was there a place for the caller's phone number. Now I've got to believe that if the company who made these phone message pads had ever tried to use them, they would have noticed immediately that there's no space for a phone number. I've got to conclude that whoever designed these phone pads never actually used them.

Speaking of people who produce things they, themselves, apparently never use, I am absolutely convinced people who design automobiles never smoke with the air conditioner turned on. If they did, they wouldn't put the ash tray just below the air vent. I also think that people who work in soda factories never drink soda. If they did, they would put a cap on the bottle that can be taken off without requiring the use of large tools. On the other hand, there is no doubt in my mind that people who work in beer factories drink beer.

I'm sure we could all have a lot of good laughs telling stories like this but it's not funny to go and buy an expensive product and wonder if the company that made it ever tried to use it. If it wasn't good enough for them to use, then why do they expect me to buy it? When I put a product on the market, I want people to feel that it's a product I used myself and that they might find

useful. So far, that hasn't been too hard to do because every program I've ever released for market has been a program I've used myself. I think it shows, too.

An interesting thing about people is that they're all little kids inside, and like all little kids, they enjoy a pleasant surprise. In all of my programs, I try to put some little thing in it that makes it do something extra special that wasn't expected by the user. It can't be some kind of a cheap gimmick. It has to be something that makes the user feel that whoever wrote the program said, "Gee, wouldn't it be nice if it also did this." When people buy a program like that and run into these little touches, it makes them feel that the product does more than they expected — and they like that.

You are your own worst enemy. The best way to defeat yourself is to believe that you "can't." If Diane Bernard believed that she couldn't make my *Star Trek* game into the world's greatest *Star Trek* game ever written for any computer by anybody in the world, we would have released no *Star Trek* game at all for the Color Computer. The only things you can't do are the things you believe you can't do. When I started my company, I had every reason in the world to believe that it would fail. Had I really believed that, this space would probably be filled with an advertisement for a printer that most of you either don't need or can't afford. As for me, I'd still be working at our local public television station helping them make shows which explain to people how easy it is to heat your home with wood.

One of my biggest complaints about a lot of programs is that I get the feeling that the author is trying to say, "Look at what I can do. See how clever I am." The truth of the matter is that there are only two people who care about what you can do — your boss and your spouse. I care little about what your boss thinks you can do unless I happen to be your boss. In any event, I certainly don't care about what your spouse thinks you can do.

I think the single most overworked command in the BASIC language is INKEY\$. It's a very powerful instruction that a lot of people overuse just to be fancy. A perfect example comes to mind because at the moment I'm writing this, I'm training a new person, Kim Foshier, to be my office manager. Kim has never used a computer before in her life. Keys like ENTER and CONTROL are all strange to her. Now

put yourself in her place and imagine that besides all of the other things you're trying to learn about running this place, you're also trying to figure out how to run a *General Ledger* program. You are told that you enter a line of data and then when you're done, you press this special key that says ENTER on it. Kim's a smart woman and although she doesn't totally understand why you press ENTER she'll do it because that's what she's supposed to do. Now we come to another part of the program and this time you press a key but you do NOT press ENTER afterwards. At this point, Kim has got to think computer people are a little nuts. First, she's supposed to press the ENTER key, then I tell her not to press the ENTER key — make up your mind! If the programmer who wrote this *General Ledger* program had thought more about people like Kim — who is typical of the type of person who is going to be using this product — and less about impressing people like me with the mastery of INKEY\$ (I'm not impressed), it would have made this *General Ledger* program a lot easier to learn.

One other example of this sort of thing comes to mind right now. If any of you have terminal programs and have called some of the bulletin board systems around the country, I think you'll agree that the BEEP is a little overworked. One system I logged onto the other night BEEPed so much, that it locked up my MX-80 printer. I don't know who the authors of these bulletin board systems think they are going to impress, but I'd sure wish they'd cut it out.

Another thing I always point out to prospective employees is the difference between what they've been doing in the past for programming and what they'll be doing now. In the past, the stuff they've written just had to work. If you wrote a payroll program, it could be the worst payroll program in the world. But, so long as it worked and the boss was happy with it, you got your raise and a pat on the back. In this business it has to do more than just work. The reason is that there are a lot of other people around who are just as capable of writing programs as you are — maybe even better. Anything you write might easily be marketed in competition with somebody else. Your work must be superior if the product you create is to survive.

Of all of the things I look for in a prospective employee, there is one thing that I definitely feel is the most

important. It's attitude. You see, anything you don't know you can be taught. But no matter how smart you are, if your attitude is bad, it will probably make it difficult to use your talents — great as they may be.

Attitude means a lot of different things to different people. Computer programmers and hardware designers share a lot in common with artists. The common ground is that they are creative people who are trying to make something that has never been made before. The purpose of creating anything is to have something that can be shared and enjoyed by others. What purpose would it serve to paint a beautiful picture if no one would ever be able to see it? What would be the point of having an orchestra play a beautiful symphony if there were no one who came to listen to it? Show me a person with the heart of an artist — who is humble, and enjoys the work — and I'll show you a person with the right attitude. Skill? No problem. As long as the ability exists, the skills can be taught.

Keep Your Options Open and Plan Ahead

If I had to pick a motto to hang on my wall, it would say, "KEEP YOUR OPTIONS OPEN." What this means is that you want to keep yourself in the position where you will have as many choices as possible in the future. I'm certain that if you ever follow my work and look at my decisions, you'll see that it is a recurring theme.

In my last column, I came down pretty hard on exclusive rights contracts. They have their place and I understand that. The trouble I have with them is that they limit your options and that rubs me the wrong way. I feel that an exclusive rights contract is like driving a car that has only one gear. When you're rolling along on level ground it's fine. But when you come to a hill you find yourself in a lot of trouble because you've got nothing to shift.

Contracts aren't the only place where I apply this philosophy. You'll also find it in the way I write my programs. I avoid features which may be particular to one machine. I may use such a feature, but if I do, I usually have a pretty good idea how I will han-

dle that problem on another system. One place where this shows up is the line lengths. I seldom write BASIC programs that have more than 128 characters in a program line. That way, I can always go to Microsoft's CP/M BASIC if I ever want to. That thinking paid off when BASCOM came out for the TRS-80® machines.

Another way to keep your options open is to plan ahead for the things that you might want to do in the future. If you have some idea where you're going now, you can start building your bridges before you come to the streams. This is exactly what was done with the *ST80* terminal programs. Before anyone had ever heard of a Model II or Model III, I had already planned in my head that *ST80-III* would be capable of running on those systems. In other words, it was designed to be modified for computers other than the Model I where it started. Although it is written for the Z-80 chip, its design makes it easy to rewrite the whole thing for another chips like the 6502 — and it didn't stop there! I also gave consideration to the possibility of writing my own bulletin board programs. Thus, while I was building a communications system, my competitors were trying to write a terminal program. That's the difference between the way other people think and the way I think. It's probably the single most important reason why I've been so successful.

Don't be a Copycat

One of my favorite stories is about Mark Twain, master of four-letter words which are unprintable here. One morning in the bedroom something went wrong and Twain began to cuss and swear up a storm. When he stopped to come up for air, his wife sat up on the bed and she began to mimic him. She cussed and swore right back using all the four-letter words her husband had just used. When she stopped, Twain looked her straight in the eye and said, "Honey, you've got the words but you don't have the melody."

It is one thing to borrow an idea from somebody else — very few people come up with a truly original idea — but it is something else to imitate another person. The copy is never as good as the original. Yet an awful lot of the stuff you see on the market is just an imitation of somebody else's work. If a programmer is an artist, and the job of an artist is to create that

which has never existed before, I must assume that those who write these cheap imitations are not programmers. They're hacks.

Look at business programs. They're all pretty much alike. I find it hard to believe that there are only one or two ways to write business computer software. Maybe I'm wrong, but it sure looks that way. That's why one of the projects I'll be working on is some business software. We have a lot of new and interesting ways of handling things. In fact, I'm surprised somebody hasn't thought of some of my ideas already.

On the other hand, maybe I shouldn't be surprised. Most people writing programs today are so busy trying to imitate the work of others, they don't have time for new ideas — even obvious ones. I, on the other hand, prefer to spend my time trying to think of new ideas and see no point in doing what's already been done.

Star Fighter

One of the best programs I've seen to date is *Star Fighter* by Sparky Starks (Adventure International). It is a perfect example of many of the things I've been talking about.

In the first place, the game is very original. It is the work of an artist because the author has made something that had never been made before. Sure, you could pick it apart and find all kinds of ideas Starks borrowed from somebody else. I certainly get the feeling that *Star Wars* was somewhat of an inspiration. But it isn't imitating *Star Wars*, *Space Invaders*, or *Star Trek*. Best of all, it has a lot of fresh ideas. People are always interested in what's new. Successful people almost always are people who have come up with something different that works. That's why they attract attention.

Earlier, I talked about attitude. I have no idea who Sparky Starks is, but I like his attitude. When I play *Star Fighter*, I get the feeling that the author created something he enjoyed and wanted to share with me. In other words, he published the program because of what it would do for me — which is to give me hours of great entertainment. Nowhere do I get the feeling that he's trying to show off. I don't doubt for a moment that Starks is an excellent Machine Language programmer but he isn't trying to prove it to me.

Then there are the little touches. One night, my *Star Fighter* was getting a lit-

tle beat up and I needed to stop at the repair station at Landbase 1. I wasn't too concerned. I figured I'd get around to it when I raised a little more bounty money. Kicking in my Hyperdrive, I started zooming through the Cosmos. Suddenly, out of my speaker, I hear a big backfire and I'm stopped in midspace with a drive malfunction. The drive malfunction I expected. The backfire sound was a total surprise. It makes me wonder what other unexpected pleasures await me. According to Scott Adams, I won't be disappointed. There are a lot of unexpected pleasures(???)

Finally, unlike a lot of products I've seen, *Star Fighter* is one of those programs you know is a game that the author himself played and enjoyed. After all, if the author of the program didn't like it enough to play it himself and enjoy it, why should anybody else? No, Starks has probably played this game to death, and it shows.


At any rate, *Star Fighter* is an excellent example of how somebody else has used the same rules I do and come up with an excellent product that deserves a great deal of recognition.

It's Really an Art

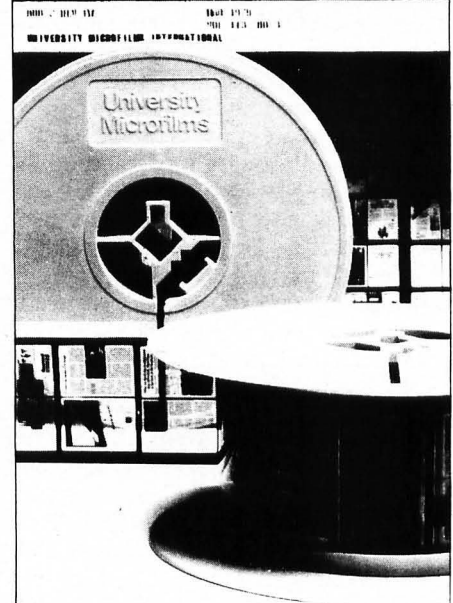
There is a great fear among many people in this market that the big guys will take over. While the fear is justified, the little guy who writes programs as an independent really has the advantage.

Programming is an art far more than a science. Art almost never comes from big factories. As long as it is possible for small independent programmers to find a way to market their products, the best programs will always come from the independents.

If I were a big computer manufacturer, like Radio Shack or Apple, the way I would go about making sure there was a lot of good software to support my system would be to support the arts — specifically the art of computer programming. All this means is that you just make sure independents have a way to get their products to the market. You definitely do not try to compete with the independents to put them out of business.

There's certainly a lot of difference between Radio Shack and Apple computers, and software for Radio Shack and Apple computers. Next time you're trying to figure out why these differences exist, I think you'll find the answer in the way these two companies support the art of computer programming. 

This publication is available in microform.



University Microfilms International

Please send additional information
for _____

Name _____

Institution _____

Street _____

City _____

State _____ Zip _____

300 North Zeeb Road
Dept. P.R.
Ann Arbor, Mi. 48106
U.S.A.

30-32 Mortimer Street
Dept. P.R.
London W1N 7RA
England



User-Defined Functions: Easier Than They Look

by David A. Gash

In this article the TRS-80® “DEFFN” feature is explored as a time- and work-saving programming aid.

A major goal in computer programming, as any textbook will confirm, is the performance of a specific task with as little code and in as short a time as possible. In the field of microcomputers, this is especially desirable, since they not only have less memory but also perform slower than their mainframe counterparts.

To this end, we, as programmers, employ many methods of code reduction and streamlining. Some of these are simply coding conventions already built into the programming language we are using; others are little tricks we pick up through experience which may bend the rules a bit, but produce faster code; still others are completely separate and distinct programs, or utilities. One of TRS-80® BASIC’s handiest features (belonging to the first group of already-existing tools), is the User-Defined Function (DEFFN). However, in talking to other programmers, I have found that the majority of them don’t use it, giving as their reason, “I don’t understand it.” All things considered, “DEFFN” is easier to use than it is to say! When properly used, this feature can become an extremely effective code- and time-saver. Let’s take a look at it.

What is a Function?

For all practical purposes, it is a built-in subroutine. User-Defined Functions (let’s call them UDFs from now on) enable us to obtain certain results without writing an equivalent BASIC subroutine (which takes up our time), and also without the actual transfer of control from the calling line to the subroutine and from the RETURN statement back to the calling line (which takes up the computer’s time). Some examples of BASIC’s existing functions are ASC(string), CHR\$(number), and LEFT\$(string, number).

Each of these functions contains the two required parts: a KEYWORD, by which the function is recognized, and one or more ARGUMENTS, by which data is passed to the function. In general, functions fall between straight code and subroutines in terms of complexity. That is, they are used most often for computations which are longer or more complex than we want

to code repeatedly, but which are not so lengthy or difficult as to require a multi-line subroutine. For example, we need not write a UDF to multiply two numbers together; regular BASIC code can accomplish the same thing easier and faster. On the other side of the coin, a UDF won’t take a string of mixed alphabetic and numeric characters and, say, count the number of “A”s within it; for that, we would need a subroutine. For problems in that scope, UDFs are exactly what we need.

Why Use Functions?

The obvious reason is, of course, speed. BASIC’s intrinsic functions operate much faster than equivalent code; UDFs may or may not operate faster than the same code in the mainline of the program, depending on the nature of the function. Therefore, the BASIC programmer’s primary reason for using UDFs will be convenience. This convenience, in terms of the reduced amount and complexity of code, generally outweighs their relative speed or lack of it. First, it is vastly easier to code a UDF call a dozen times in the body of a program than a long line of code the same number of times. Second (and most important), since there is less code, there are fewer bugs. Period. Once the UDF is coded correctly, it can be used repeatedly, with the certainty that it will operate without error each time, thus reducing overall debug time on your program.

Types of Functions

There are two basic (sorry) types of functions, numeric and string. A function’s type is determined solely by the type of value it returns (i.e., what the function name’s variable type is), not by what type arguments it has. CHR\$(number) and RIGHT\$(string, number) are both string functions, as designated by the type identifier “\$”; ABS(number) and VAL(string) are numeric functions, since there is no type identifier. (Actually, a numeric UDF can have any of the three type identifiers “%”, “!”, or “#” denoting integer, single-precision, or double-precision, respectively; the value returned is the same precision as the function’s type identifier. As with any numeric variable, the default type is single-precision; for our purposes, a standard non-identified numeric func-

tion will do.) Wherever BASIC syntax calls for a numeric expression, a numeric function may be used; wherever a string expression may appear in a line of code, so may a string function.

Well — enough discussion. Let’s code some UDFs and see how they work in actual use.

Examples and Explanations

The TRS-80® Model II has a very nice PRINT@ feature; rather than hard-coding a print position, such as PRINT@915, Model II BASIC allows us to code the relative line on the screen (0-23) followed by the relative cursor position (0-79), like this: PRINT@(11,35), “(text here)”. Quite a shortcut from trying to figure out all those screen positions! However, Model I BASIC doesn’t include this feature. Rats! Right? Wrong! A simple UDF can do the same thing. First, define the function with DEFFN and a name (let’s call it AT, since that’s the Model II function we’re emulating — note that because we want a number as the returned value, there is no type identifier); follow that with the parameter to be passed to the function, in parentheses (call them L and P, for Line and Position); follow that with an equals sign and the actual computation using the parameters, and we’re through. Our complete definition looks like this:

```
DEFFN AT(L,P) = L * 64 + P
```

Now, to use the function, just call it by FN and its name and pass it the required parameters (either the actual numbers or variables containing valid values). Note that these variables need NOT have the same names as those in the DEFFN statement, but they must be of the same type. That is, to print something at position 30 on the seventh line on the Model I screen, we can either calculate the position and use PRINT@ 478, “(text)”, or, using our UDF, we can simply say PRINT@ FNAT(7,30), “(text)”. Obviously, for this single PRINT, the UDF takes more code and time; but if we’re doing a lot of PRINTing in our program it becomes quite easy to get used to the UDF, since we don’t have to consult a video display worksheet for every PRINT. Here’s how this UDF works: When BASIC encounters the keyword FN, it calls the function, AT, and gives

One problem I run into constantly in trying to format a pleasing, balanced screen display is the centering of titles, headings, etc... A simple User-Defined Function solved that problem for me, by automatically centering the string passed to it, regardless of the length.

the variable L a value of 7 and P a value of 30. It then multiplies 64 (the length of the screen line) by 7, obtaining 448, and adds 30, getting a result of 478. It replaces the function call with this calculated value, and executes the PRINT@ statement normally. It now becomes a simple matter to print anywhere on the screen we want. Try this program:

```
10 DEFFN AT(L,P) = L * 64 + P
20 INPUT "LINE, POSITION";A,B
30 CLS:PRINT@ FNAT (A,B),
  "* < = = POSITION" FNAT (A,B)
40 GOTO 20
```

Note line 30, which demonstrates that calling the function doesn't really DO anything except obtain a number, by using it as a field in the PRINT statement. Also note that values for A (or L) other than the standard line numbers 0-15 will cause an ?FC (Illegal Function Call) error, as UDFs do no error checking; it's up to us, the programmers, to supply valid information to the UDF through the parameters.

Let's examine a string function in detail. (Actually, as a general rule, I find string functions handier than numeric functions, but maybe that's just my sloth showing. . .) One problem I run into constantly in trying to format a pleasing, balanced screen display is the centering of titles, headings, etc. I used to spend a lot of time figuring the print position, testing, deciding I didn't like the heading, changing it, re-adjusting the print position for the new header because of the difference in length, testing, deciding I didn't like it, and so forth, ad nauseum. A simple UDF solved that problem for me, by automatically centering the string passed to it, regardless of the length. This is it:

```
DEFFN CN$ (A$) = STRING$
(32-LEN(A$)/2,32) + A$
```

The name of the function is CN\$ (for "CeNter"); it carries the type identifier "\$", indicating that we want a string as the returned value; the argument is any string variable. Working

from the inner parentheses out, the UDF first uses an intrinsic function to obtain the LENgth of the string to be printed, and divides that by 2. It then subtracts that figure from 32 (half the length of the screen line). This tells it how many extra spaces it has on half the line (in effect, how far to "tab" to the right). It constructs a string of that many blanks (the ASCII code for a space is 32), concatenates that string of blanks onto the left end of the original string, and returns that as the final string value. Once this is set up, centering becomes a breeze! Again, try this short test program:

```
5 CLEAR 1000
10 DEFEN CN$(A$) =
STRING$(32-LEN(A$)/2,32) + A$
20 INPUT "TYPE ANY STRING";P$
30 CLS:PRINT@ 448, FNCN$(P$)
40 PRINT:GOTO 20
```

Now it's getting easy! In fact, let's replace that PRINT@ 448 in line 30 with our other UDF; using the two functions together, centering gets easier all the time. Try:

```
5 CLEAR 1000
10 DEFFN AT(L,P) = 64 * L + P
20 DEFFN CN$(A$) = STRING$
(32-LEN(A$)/2,32) + A$
30 INPUT "STRING TO BE
CENTERED";X$
40 INPUT "LINE TO CENTER IT
ON";Y
50 CLS:PRINT@ FNAT(Y,0),
FNCN$(X$)
60 PRINT:GOTO 30
```

Note that because the CN\$ function must begin at the left margin to work correctly, we need only pass the AT function the line number (to multiply by 64), and a 0 for the "offset" parameter, and it works fine. As you can see, it doesn't take long before UDFs start saving us a lot of programming and debugging time, not to mention frustration!

The two examples above are UDFs I use in almost every program I write. Then, for each individual program, I usually end up writing a special function or two. The following examples

should be helpful in understanding how to define and use UDFs.

Date formatting can be cumbersome and inconvenient for the programmer. Here's a program with a function which returns a string-type date when a numeric-type date is passed to it:

```
5 CLEAR 1000
10 DEFFN DT$(M,D,Y) = MID$
("JANFEBMARAPRMYJUN
JULAUGSEPOCTNOVDEC",M*
3-2,3) + "." + STR$(D) + "," +
STR$(1900 + Y)
20 INPUT "MONTH, DAY, YEAR (2
DIGIT)";MO,DY,YR
30 PRINT "THAT DATE IS:
"FNDT$(MO,DY,YR)
40 GOTO 20
```

This one takes the month value (1-12), multiplies it by 3 (the length of each abbreviation), backs up 2 to get to the first character of the abbreviation, and extracts that portion of the string (with MID\$). It then concatenates a period onto the month abbreviation, the day in string form, a comma, and the year in string form (after adding 1900). Granted, the definition itself is long, but we need only code it once. After that, any date we need is obtained easily, as demonstrated in line 30.

Many times in a menu-driven system, menu options will be selected by the first letter of the descriptions, rather than by number. A simple UDF for "emphasizing" the first letter of any string follows:

```
5 CLEAR 1000
10 DEFFN EM$(A$) = "<" +
LEFT$(A$,1) + ">" +
RIGHT$(A$, LEN(A$) - 1)
20 INPUT "TYPE ANY STRING";S$
30 PRINT TAB(25); FNEM$(S$)
40 GOTO 20
```

This handy little rascal takes a string (say, "ADD EMPLOYEE RECORD"), surrounds the first letter with the signs "<" and ">", and tacks on the rest of the string, like this:

```
< A > DD EMPLOYEE RECORD
```

I do keep a "library" of my favorite User-Defined Functions on disk and merge them into new programs when I begin writing them. In this way, my User-Defined Functions have become as familiar and easy-to-use as BASICs, and because of this, my program entry time is reduced...

This makes menu printing easy; we never have to type the "emphasizing" characters — just pass it the string, and the UDF does the rest!

Often, names may be in records in the form "LASTNAME, FIRST-NAME", but we may want to display it in the form "FIRSTNAME LASTNAME" on the screen or on a report. Usually, a subroutine would be written to do this, but a UDF will get us the same result. This program will demonstrate:

```
5 CLEAR 1000
10 DEFFN NM$(A$) = RIGHT$(
A$,LEN(A$)-INSTR(A$,","))
+ " " + LEFT$(A$,INSTR
(A$,",")-1)
20 LINEINPUT"LAST NAME,
FIRST NAME? ";LF$
30 PRINT"THE NAME IS: ";
FNNM$(LF$)
40 GOTO 20
```

This function uses a name in the form "LASTNAME, (comma) FIRST-NAME", hence the LINEINPUT. (Non-Disk BASIC users would need to use two INPUTs and concatenate a comma between them to obtain the same result, but we're assuming the name would actually come from a disk file, anyway.) The name is passed to the UDF, which uses INSTR to find the position of the comma within the string. Having found this number, it subtracts it from the length of the entire string to obtain the number of characters to the right of the comma, then strips those off and saves them (the first name). It adds a blank to the first name, then (again) finds the comma, strips off the characters to the left of it (the last name), and finally adds that to the first name and the blank. The result from (for example) "SMITH,JOHN Q." is then "JOHN Q. SMITH". This has come in handy more than once when dealing with employee names in compressed disk files.

UDFs can also be very useful for verification of input. In the example using the "LASTNAME, FIRST-NAME" name format, the absence of

a blank after the comma was important to the correct operation of the UDF; if there is a blank in front of first name, it winds up as an extra space in front of the completed string (" JOHN Q. SMITH"). In some programs, the opposite is the case; that is, there must be a blank between the comma and the first name. Ordinarily, an INSTR check for a blank would suffice, but not in the case of an unusual name like "VAN SCHWARTZ, IRVING"; the blank following "VAN" would satisfy the INSTR function, and if the blank following the comma were not there, it would not be detected. However (you guessed!), a UDF can take care of it nicely:

```
5 CLEAR 1000
10 DEFFN VN$(A$,C) = ASC
(MID$(A$,C+1,1))
20 LINEINPUT"LAST NAME,
FIRST NAME?";LF$
30 X = INSTR(A$,",")
40 IF FNVN(LF$,X) = 32 THEN
PRINT "VALID" ELSE PRINT
"INVALID"
50 GOTO 20
```

One other technique worth mentioning is that of nesting UDFs. We often nest BASIC's intrinsic functions, such as: ASC(MID\$(X\$,10,1)) or MID(A\$,VAL(X\$),5). UDFs can be nested the same way. In fact, they can be nested two different ways! Remember the centering UDF and the emphasizing UDF? Those can be nested explicitly, like this:

```
PRINT FNCN$(FNEM$( "DELETE A
RECORD" ))
```

or we can define a new function which is actually an implicit nesting of two (or more) other UDFs.

```
DEFFN CE$(A$) = FNCN$(FNEM$(
A$))
```

then call the new function just as we would either of the other two:

```
PRINT FNCE$( "CHANGE A
RECORD" )
```

In both cases, BASIC works from the inside out, and simply makes whatever calls are necessary before proceeding; the outer function then operates on the returned value from the inner function. Of course, with both methods, the original two functions are always unchanged and available.

Summary

The main points of this article may be summed up with a few DOs and DON'Ts:

- DO use DEFFN when a string or numeric formula is complex and likely to cause typing errors, increasing both entry time and debug time.

- DON'T use it if a subroutine will do the job better or more efficiently than a ridiculously complicated UDF.

- DO use it in conjunction with BASIC's existing functions to create new, specialized functions to suit your specific programming needs.

- DON'T use it if an intrinsic function already exists which does almost exactly the same thing.

- DO use it to clarify common operations which, while not necessarily complex, are clearer or easier to understand with a UDF.

- DON'T use it if the operation is so simple it requires only a line of ordinary BASIC code.

In short, use it, but don't overuse it. DEFFN can be an enormous time-saver when developing a program, but only to the point where we spend less time using the UDFs than it took to set them up! In practice, I have found that most of my UDFs are string functions, to simplify editing, formatting concatenating, etc., as most standard arithmetic may be done with BASIC's very good mathematical functions. I do keep a "library" of my favorite UDFs on disk and merge them into new programs when I begin writing them. In this way, my UDFs have become as familiar and easy-to-use as BASICs, and because of this, my program entry time is reduced even further, as is testing and debugging time. In other words, less time and trouble between concept and completed program — and that's what it's all about! ☺



Hardware Corner

by Edward E. Umlor

As promised in last month's column, I am going to start a series on diskettes and disk drives. The first thing we should look at is the diskette. The same principles apply to both the floppy diskette and the hard disk. The major difference between them is that the hard disk read/write head never comes in contact with the disk itself, which is rotating at 3000+ RPM, while the read/write head of a floppy diskette is in contact with the diskette (turning at 300 RPM) when accessed. With that in mind we will start to lay the groundwork for our little project.

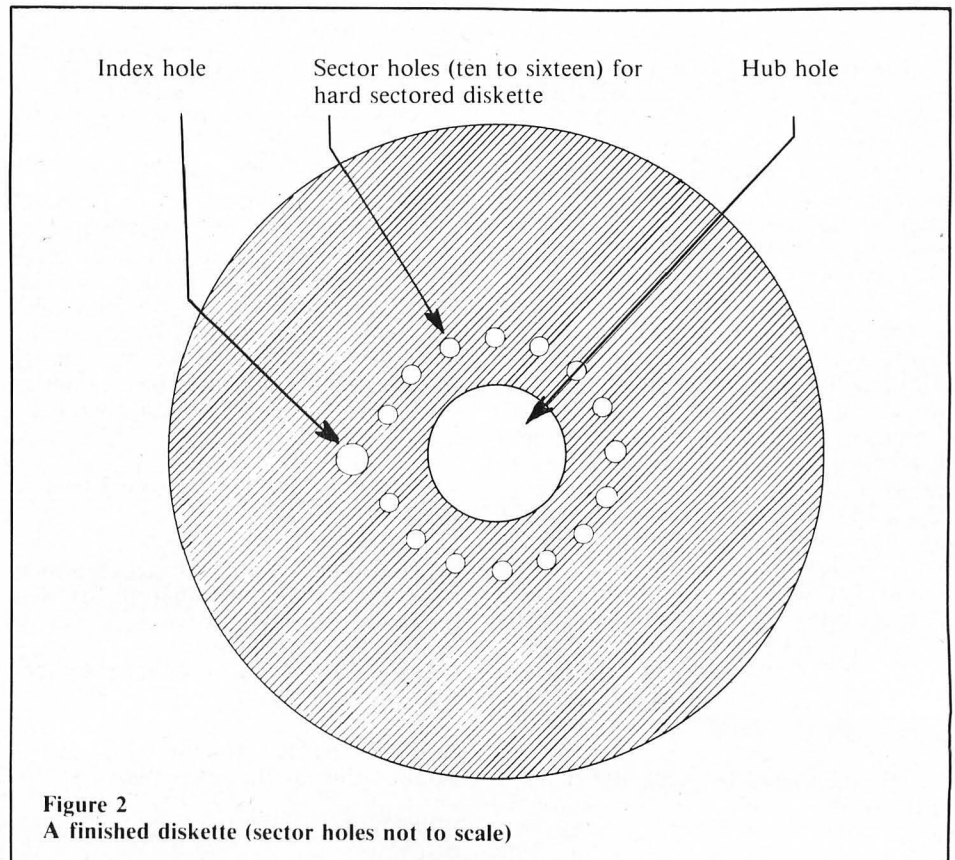
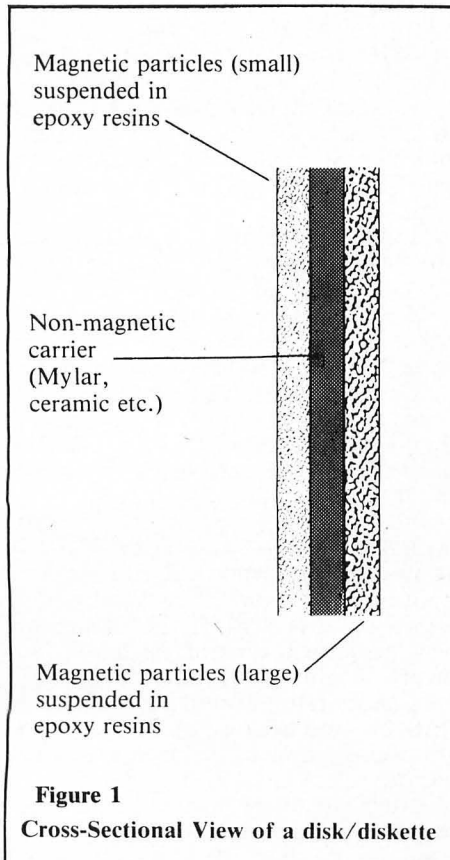
The floppy diskette has a flexible mylar material in the center. Magnetic material is bonded to both sides of the mylar with a formula that lets it remain flexible when cured. This bonding material is one of the epoxy resins into which the actual magnetic material is mixed before coating the mylar. The hard disk is made the same way except the center is a rigid plate of non-magnetic material and the bonding material can be rigid when cured. See

Figure 1 for a cross-sectional view of a diskette or disk. The magnetic material can be one of a vast variety such as rust, special alloys, or combinations. The surprise is that rust was used for early recording tapes and devices. It was cheap, available, and it is magnetic in nature. Since the early days, there have been a lot of various materials and formulations that have been tried, used and discarded.

We now proceed to the next step that makes a magnetic recording medium better. When the magnetic material is ground up, the average size of the particles is controlled. The smaller the particle, the less magnetizing energy is necessary to magnetize it. Each molecule of iron (or other material that can be magnetized) is a small magnet itself. Normally the molecules are randomly oriented, so that individual magnetic fields cancel each other out. A piece of iron (your car or engine) is not a magnet. However, an external magnetizing force can be applied, which the molecules react to and align

themselves. When this occurs, the individual fields now aid each other and you have a magnet. The second benefit of using very small particles of magnetic materials is that you can have a higher percentage of magnetic material to binder. This is called increasing the magnetic density of the recording tape. The tapes that advertise high density are made with extremely fine powdered materials. We started off with a cross section of a diskette or disk in Figure 1. This same coating process is used for hard disks, floppy diskettes, and magnetic drums as well as magnetic recording tapes. All the same principles apply.

We are now ready for the next step in the process of putting our floppy diskette together. We now have to prepare the surface of the diskette for the read/write head. We don't want a rough surface in contact with the disk drive head. After all we don't want to sandpaper the head too fast. Wear on the head will never be completely eliminated, but it can be greatly re-



duced. To do this there are two popular methods. You can highly polish the surface of the tape or sheets of disk material, or highly polish the surface of some rollers and then roll the material flat between the rollers. Both methods are in use and both can produce a high quality product. Those that polish the surface of the floppy diskette usually wait until the diskette has been punched out of the sheet of coated mylar, and then polish only the area of the diskette that will be used for data storage.


Producing the little round diskette is done with a press and die method. That good old chunka . . . chunka . . . chunka method. On each chunka of the press, a diskette is formed with the center hole and the index hole (the little hole just out from the center hole) as well as all the sector holes if it is to be a hard sectored diskette (see Figure 2). The product could be used right now, but would have a very short life span due to injury. The surface of the diskette could be scratched or contaminated or the diskette bent, folded, or mutilated quite easily if left as is from this process.

In order to protect it, the manufac-

turers decided to enclose the diskette in an envelope (see Figure 3). The protective envelope is composed of a strong but flexible outer plastic layer with a soft lint-free liner. The liner lessens the wear on the diskette surface as well as keeping the working surface free of dust. The front and back halves of the envelope are ultrasonically bonded to each other after the diskette has been placed inside. The technique of ultrasonic bonding keeps the energy and heat confined to a very small area and prevents damage to the diskette during bonding.

We now have our completed diskette. The envelope has an oval opening from the center toward the outside edge. This is the opening where the read/write head contacts the disk. The hole just to the outside of the hub hole (center hole) is for the index/sector holes, and of course the write/protect notch is on the outer edge of the envelope. We will be going into more detail about how the diskettes are organized for data storage next month. This also brings up a sore point with me concerning disk drive advertisements. The unformatted storage figure quoted by some is absolutely meaningless. The formatted

storage capacity of a diskette with any given drive is dependent upon the disk operating system, density, and actual number of tracks being utilized. A 40-track drive can be used as a 35-track drive and thus lose its full data storage capacity. A drive fully capable of double density can be run in single-density format and reduce the data capacity. I can sympathize with the drive manufacturers up to a point, but only to a point. They should state the maximum formatted storage capacity of their drives for both single- and double-density operation. These figures have true useful meaning to the disk drive purchaser.

I had best stop here or I will be into next month's column. In January we will be discussing how the disk is organized (formatted) to receive data, some of the requirements of disk drives, double/single density operation, and some of the disk operating systems (DOSs) that are available. Well, that is about all for now from the old Granite Knoggin' for this month. Remember — if you have any questions about the information in this column, please write and we will have a question-answering session at the end of following columns. 

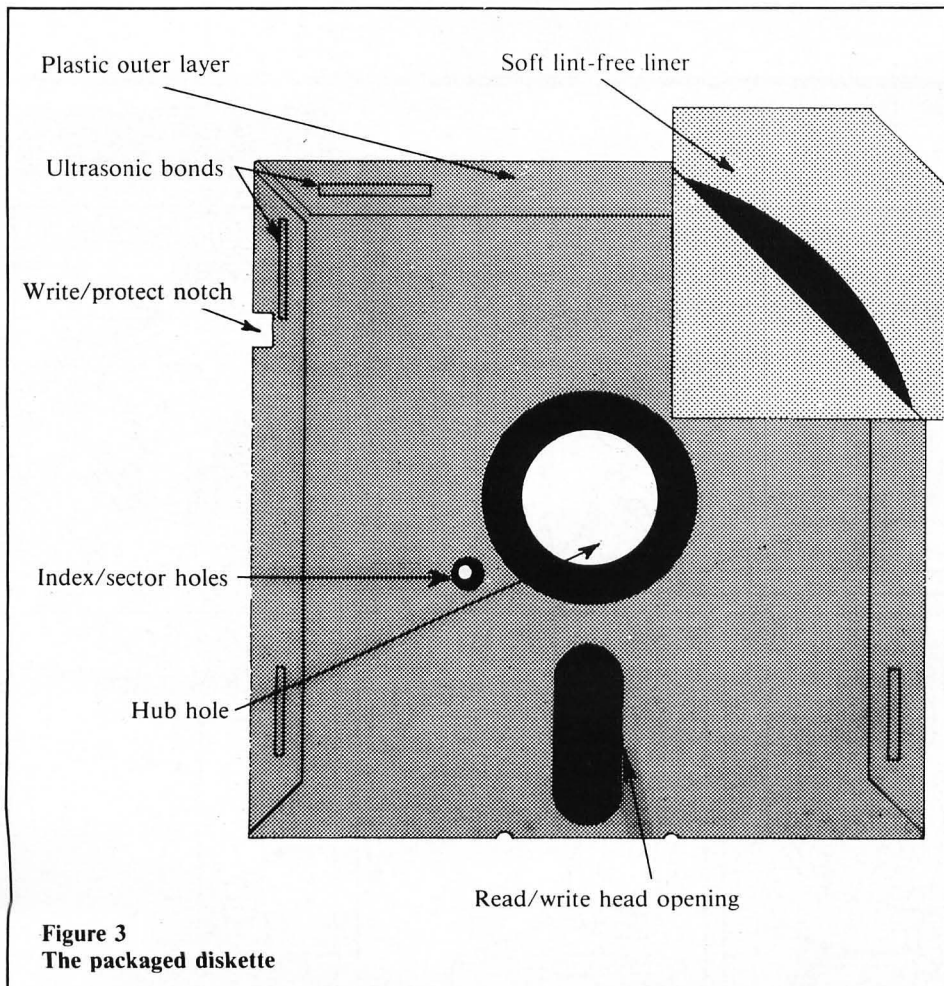


Figure 3
The packaged diskette



duplicating service

307 West Main Street
Maple Shade, NJ 08052

(609) 667-1667

- AMP "Data-sette" blank cassettes for digital use
- Cassette Storage Boxes
- Cassette Labels - Custom printing & blank
- Custom Record Album production from your tapes
- Stereo and Spoken Word cassette duplication

Call or write to:

Jerry

for more information.

All cassette work at AMP R. & D. is custom work to fit your needs.

Coming next month in

SoftSide

GAMBLER

Next month it's a sure bet. Play the ponies, spin the wheel of fortune, and more in these games for up to four players.

APPLE CAPTURE

Your hungry worm searches the orchard for apples to chomp.

PLAZA HOTEL

You are the house detective looking for a bomb hidden in one of the rooms.

Plus:

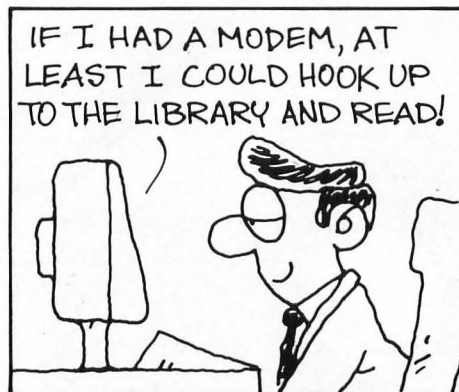
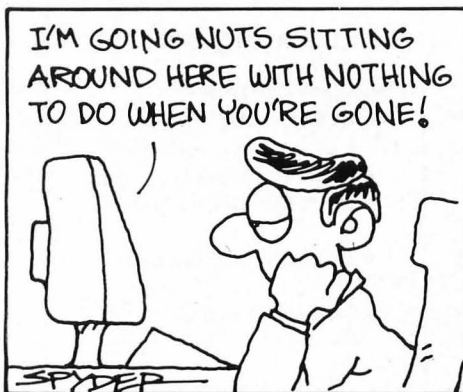
Word Wars, Number Race, and much, much more.

Advertiser's Index

Adventure International	Cover III
Amp Recording	95
Computer Shows	16
Continental Adventures	24
Icom	75
Micro Ink	64
Muse	17, 27
SoftSide Selections	Cover II, 1, 4, 9, 12, 13, 20, 32, 39, 40, 48, 56, 75, 90,
Spectrum Computers	25
Strategic Simulations	Cover IV
80-U. S. Journal	76

MACHINE HEAD

BY SPYDER



ADVENTURE has gone GOLD! AND WE WANT YOU TO CELEBRATE WITH US.*

The 12 Scott Adams' Adventures are presented in our **Limited Gold Edition**. Each tape and disk is individually numbered, and guaranteed until July 10th, 2001. Each package contains a certificate of authenticity, a registration card and an autographed, (rather interesting) photo of the author. The 12 Adventures normally retail for \$239.40 individually on tape and \$159.80 for 4 triple-pack disks.

The **Limited Gold Edition** is yours ... forever ... for \$100.00 tape or disk, value for value. To Order: The **Limited Gold Edition** is available in very limited quantity, from interested Software Retailers. Ask your local dealer. If he does not have The **Limited Gold Edition** ... Then call toll free 1-800-421-5770. In California 1-800-262-4242 (Local 213-670-9461) ... And we will direct you to a dealer who does have the **Limited Gold Edition**. *Supply Is Limited!*



The **Limited Gold Edition** is available in the following different configurations.

- Apple 2 Disk with TRS-80 Disk in same package
- Atari Tape
- Atari Disk
- TRS-80 Tape

\$100.00 Each

*In keeping with the spirit of **Adventure**, \$1,000.00 in Gold Coins is hidden within The **Limited Gold Edition**. Its waiting for you.

The **Limited Gold Edition** from Adventure International is distributed exclusively by:

SOFTSEL

8295 SO. LA CIENEGA BLVD. • INGLEWOOD, CA 90301

TWO NEW GAMES FROM SSI FOR THE

APPLE® AND THE TRS-80®!

We know it hasn't been easy for you TRS-80® owners to see so many great made-for-Apple-only games from SSI pass you by. But then, it hasn't been easy for us to design games for a 16K cassette format good enough to meet our critical standards.

After all, we've got a reputation to protect, a reputation in strategy gaming for unsurpassed sophistication, innovation, realism, and playability.

Well, our designers have been hard at work, and we've not only met but surmounted the challenge. We're delighted to announce two historical wargames — deserving of the SSI label — for both the Apple® and the TRS-80® (16K cassette for the TRS-80 Model I and III; 48K disc for Apple II with Applesoft ROM card).

Combining our extensive war-game-design experience and superior programming techniques, we've given a fresh new look and feel to these favorite classic battles.

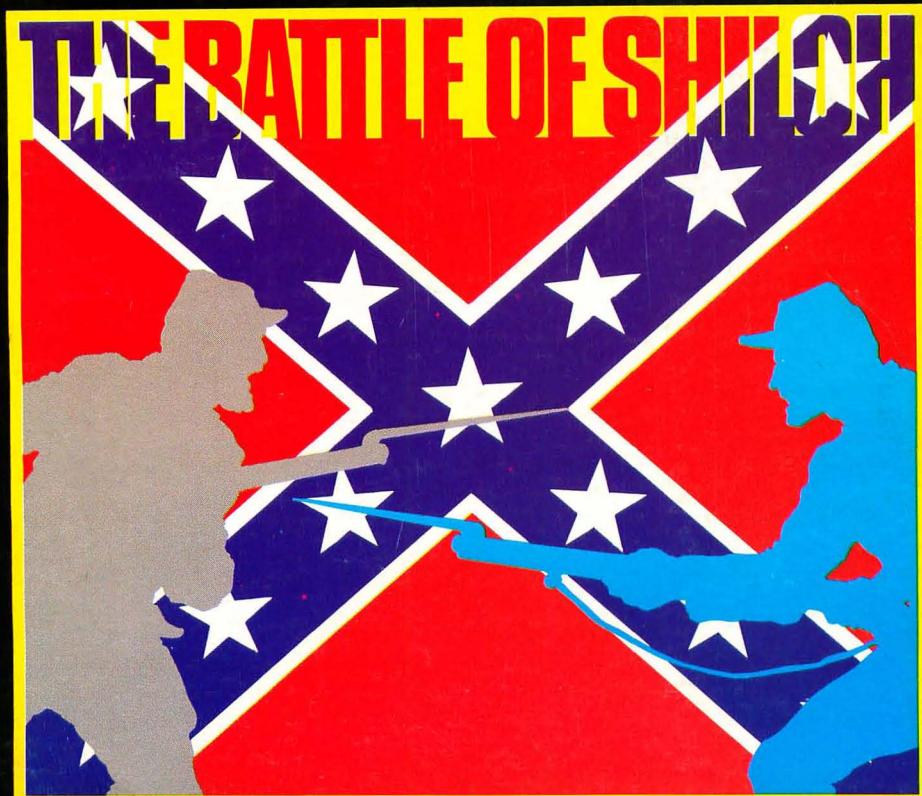
At \$24.95 each for TRS-80 cassette and \$39.95 each for Apple disc, these are extraordinary games at quite an ordinary price.

So head on down to your local store and check them out today!

VISA and M/C holders can order by calling 800-227-1617, ext. 335 (toll free). In California, call 800-772-3545, ext. 335.

To order by mail, send your check to: Strategic Simulations Inc, Dept. SS-1, 465 Fairchild Drive, Suite 108, Mountain View, California 94043.

All our games carry a 14-day money-back guarantee.



THE BATTLE OF SHILOH: A brigade-level simulation of the first grand battle of the Civil War, pitting the Confederate Army against Grant's troops and Union gunboats.

THE BATTLE OF THE BULGE:



TIGERS IN THE SNOW: Ghostlike Nazi Tiger tanks and infantry sweep across the dark, frozen forests of the Ardennes against a surprised U.S. force in this division/regiment-level simulation of Hitler's last desperate attack.



STRATEGIC SIMULATIONS INC

As part of our demanding standards of excellence, we use **MAXELL** floppy discs.

Apple is a registered trademark of Apple Computer Inc.

TRS-80 is a registered trademark of the Tandy Corporation.