NUMBER 8

$2.50

A.N.A.L.O.G. COMPUTING

THE MAGAZINE FOR ATARI® COMPUTER OWNERS

BUDGET PROGRAMS REVIEW

CTIA AND GTIA GRAPHICS

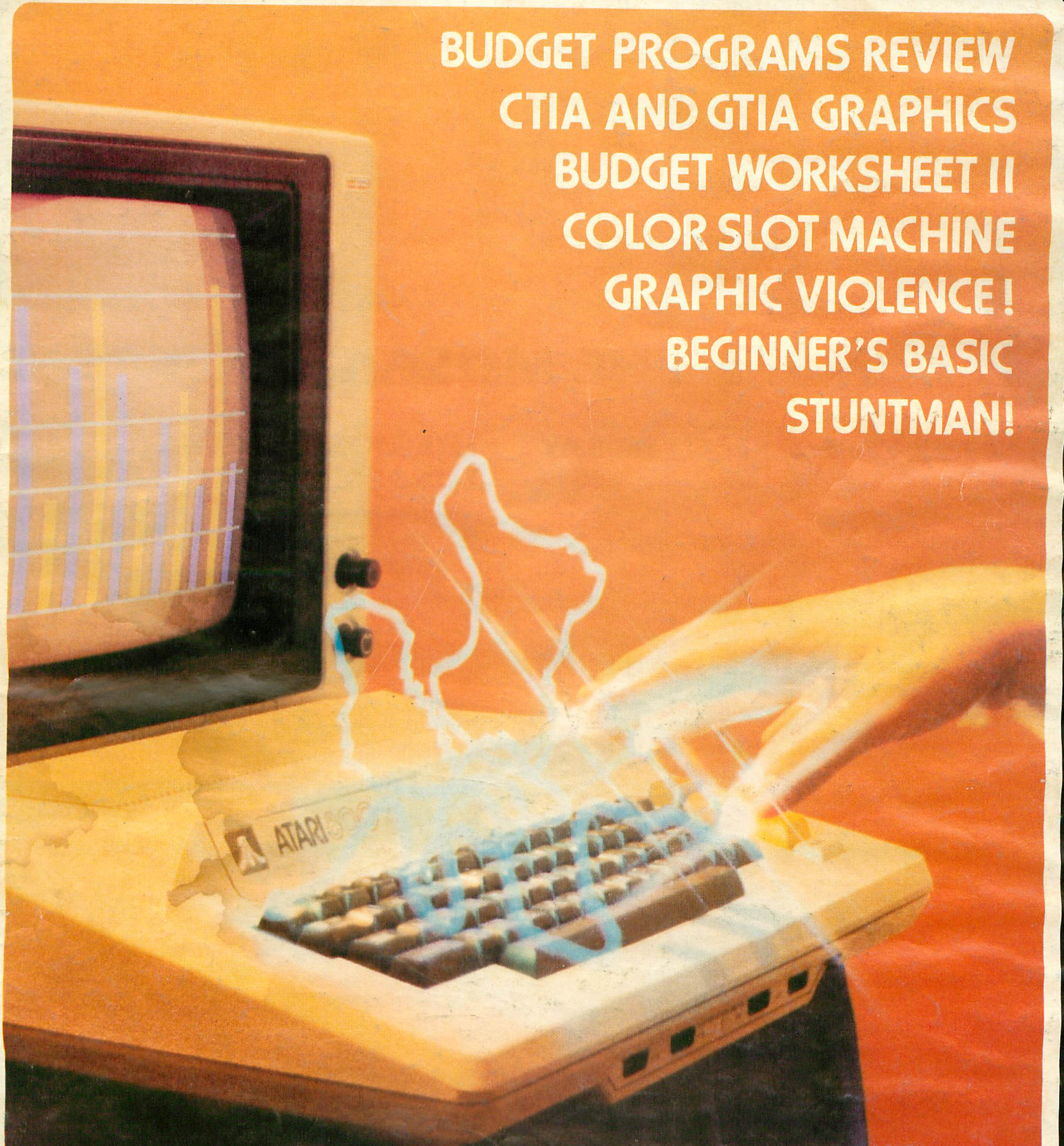
BUDGET WORKSHEET II

COLOR SLOT MACHINE

GRAPHIC VIOLENCE!

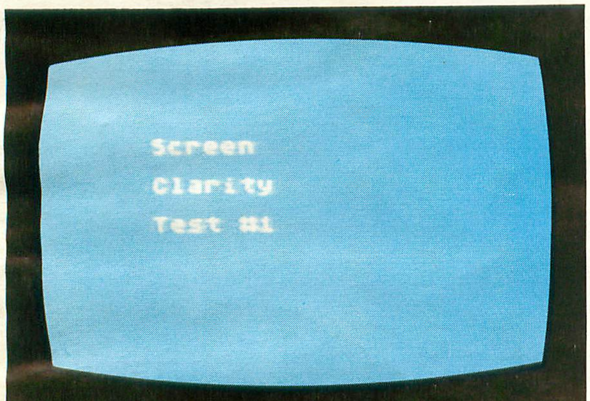
BEGINNER'S BASIC

STUNTMAN!



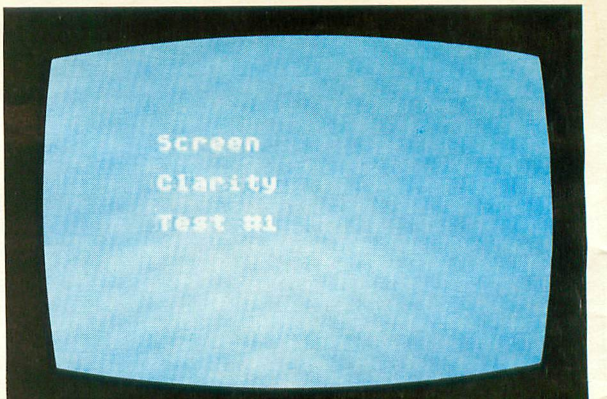
THE MOSAIC RAM SYSTEMS FOR ATARI*

CLEARLY THE BEST



MOSAIC 32K RAM

THE SCREEN CLARITY TEST



THE COMPETITION

YOU CAN SEE THE MOSAIC DIFFERENCE

WHAT THE EXPERTS HAD TO SAY

A.N.A.L.O.G. 400/800 MAGAZINE said in a 32K RAM board comparison test: "The Mosaic 32K RAM showed no sign of interference and gave the best screen clarity" and "Mosaic uses what we feel are the highest quality components which may improve reliability".

INTERFACE AGE said after replacing the Atari memory board with a Mosaic 32K RAM: "Once in place there is no noticeable change in screen clarity" and "in view of its excellent performance it should be a serious choice for those Atari owners intending to expand their memory."

Each Mosaic RAM board gives you more than just the best screen clarity but also the best in reliability, flexibility, and compatibility. The Mosaic RAM systems offer you the best in every way — these features prove it.

- Works in both the Atari 400*& 800*computers.
- 4 year guarantee.
- Complete instructions.
- Test cycled 24 hours for reliability.
- Gold edge connectors for better reliability.
- Quick no-solder installation.
- Full flexible memory configuration.
- Can be used with 8K, 16K and future products.
- Designed to take advantage of Atari 800's superior bus structure.
- Designed for inter-board communication in Atari 800.
- Allows Atari 800 to have 2 slots open for future expansion.
- Always the best components used for superior screen clarity and reliability.
- Low power design for safety and reliability.
- Available companion board ($5) to allow running The Mosaic RAM systems independent of other boards.

THE MOSAIC RAM SYSTEMS FOR THE SERIOUS ATARI OWNER.

THE MOSAIC 32K RAM

For the serious Atari owner. This is the 32K RAM board you've been hearing about. It has every feature you could want from a RAM board and more features than any other board offers. Each Mosaic 32K RAM comes with complete instructions so in a few minutes you will have expanded your Atari 800 to 48K RAM. The Mosaic 32K RAM works as well in the Atari 400, but we suggest the NEW Mosaic Expander.

THE MOSAIC EXPANDER

This is the most effective way you can expand to 32K RAM for your Atari 400 computer. And at almost the 16K price! The revolutionary Mosaic Expander is a memory board with 16K RAM in place and open slots to add 16K more from the Atari 16K board that comes with your Atari 400. Each board comes with complete instructions so in a few minutes you will have expanded to 32K RAM.

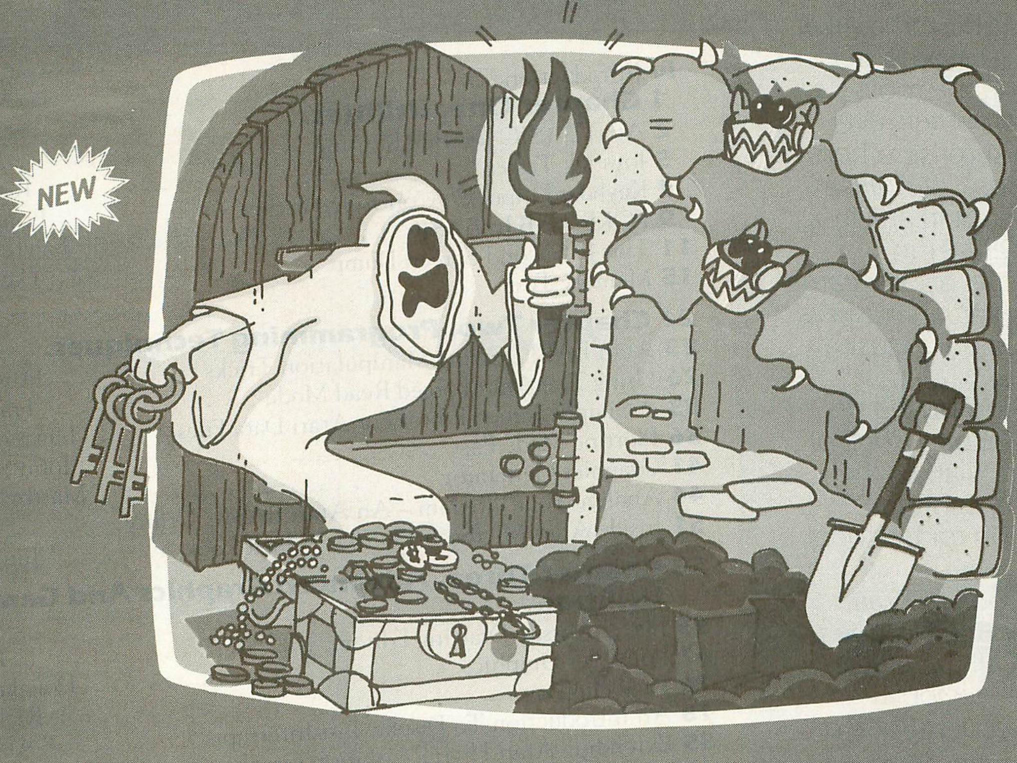
*Trademark of Atari, Inc.

MOSAIC
ELECTRONICS

P.O. Box 708 Oregon City, Oregon 97045
503/655-9574, Toll Free 1-800-547-2807

GHOST ENCOUNTERS™

A 16K Assembly Language Real-Time Adventure Game For ATARI® 400/800™ Computer



GHOST ENCOUNTERS™ is the second in a series of real-time, joystick controlled, adventure games following ACTION QUEST,™ presenting a brand new collection of dangers and puzzles. You, as a ghost, travel through a network of 30 rooms in search of valuable prizes while, at the same time, try to survive the many perils encountered. Of course, a mere ghost cannot overcome all the evil powers striving to block his journey. Luckily, you are not a mere ghost, but are equipped with the power of transmutation, allowing you to take on the form of other, sometimes more useful, objects. Who knows when a torch or a hammer might be just what the witch doctor ordered.

Locating all 30 rooms and the 20 prizes scattered throughout requires a player with fast response, physical coordination, and intellectual resourcefulness. Of the 20 prizes, no two can be acquired through exactly identical methods. Regardless of which factors most interest you in a real-time action game, you are sure to find them among the wide variety of challenges located in GHOST ENCOUNTERS.™

Available from your local Atari retailer or send $29.95 in check or money order (California residents add 6½% sales tax) to JV Software Inc.

ALSO AVAILABLE

Don't forget to ask for the original ACTION QUEST,™ with its own unique and imaginative assortment of puzzles, prizes, and perplexing predicaments. Another challenging journey through an entirely different network of rooms and prizes.

Also available in 16K for $29.95 from your local retailer or JV Software Inc.

Atari™ and 400/800™ are trademarks of ATARI, Inc.

JV SOFTWARE, INC
3090 MARK AVE. SANTA CLARA, CA 95051

COMPUTE!'s Second Book Of Atari

After only three years on the market, the Atari 400/800 microcomputers have become among the most popular personal computers ever made. So it was no surprise when COMPUTE!'s First Book of Atari, *a collection of the best Atari articles published during 1980-81 in* **COMPUTE!** *Magazine, also became a "bestseller" with Atari enthusiasts. The first printing sold out in just a few months.*

That's why we've followed up with COMPUTE!'s Second Book of Atari. *Available immediately, the* Second Book of Atari *continues* **COMPUTE!***'s tradition for personal computer users.*

But the Second Book of Atari *differs from the* First Book *in one important respect – all the articles are totally new and previously unpublished. The* Second Book of Atari *includes such interesting articles as "Page Flipping," "Fun With Scrolling," "Perfect Pitch," "Player-Missile Drawing Editor," and "TextPlot Makes a Game." Whole chapters are devoted to subjects such as "Advanced Graphics and Game Utilities," "Programming Techniques," and "Beyond BASIC." With 250 pages – more than 25 percent thicker than the* First Book *at the same price – the* Second Book of Atari *is crammed with information and ready-to-type program listings. And the book is spiral-bound to lie flat and is fully indexed for quick reference.*

Best of all, COMPUTE!'s Second Book of Atari, *like* **COMPUTE!** *Magazine itself, is written and edited to appeal to all computer enthusiasts – beginners and experts alike. Priced at only $12.95.*

Available at computer dealers and bookstores nationwide. To order directly call TOLL FREE 800-334-0868. In North Carolina call 919-275-9809. Or send check or money order to **COMPUTE! Books**, *P.O. Box 5406, Greensboro, NC 27403.*

Add $2.00 shipping and handling. Outside the U.S., add $4.00 for air mail, $2.00 for surface mail. All orders prepaid, U.S. funds only.

NOTE: ATARI® is a registered trademark of ATARI, INC., a Warner Communications Company and all references to ATARI® should be so noted.

ANALOG COMPUTING
THE MAGAZINE FOR ATARI® COMPUTER OWNERS

1982
NUMBER 8

A.N.A.L.O.G. STAFF

Editors/Publishers
MICHAEL DES CHENES
LEE H. PAPPAS

Managing Editor
JON A. BELL

Contributing Editors
TONY MESSINA
DR. GARY ROSE
MARCIA ROSE

Art Director
BOB DESI

Technical Division
CHARLES BACHAND
TOM HUDSON
CRAIG PATCHETT

Advertising Manager
MICHAEL DES CHENES

Distribution
PATRICK J. KELLEY

Typography
HENDRICKX &
LARRIVEE CO., Inc.

Production/Distribution
LORELL PRESS, INC.

Contributors
BOB CURTIN
WILLIAM HOUGH
MICHAEL A. IVINS
ALY KAHN
RICHARD KALAGHER
THOMAS KRISCHAN
ANDREW LIEBERMAN
ISTVAN MOHOS
BRIAN MORIARTY
CARL PATTERSON
STEVEN POGATCH
MARC ROSSEN
LARRY SEFTOR
MIKE SUEIRRO
STEPHEN A. VANCE
CRAIG WEISS
JERRY WHITE

A.N.A.L.O.G.
Magazine Corp. is in no way affiliated with ATARI.
ATARI is a trademark of ATARI, Inc.

FEATURES/ARTICLES

REVIEWS/PROFILES

PROGRAM LISTINGS

COLUMNS

A.N.A.L.O.G. COMPUTING (ISSN 0744-9917) is published bi-monthly for $12 per year by A.N.A.L.O.G. Magazine Corp., 565 Main Street, Cherry Valley, MA 01611, Tel. (617) 892-3488. Second-class postage paid at Worcester, MA and additional mailing offices. POSTMASTER: Send address changes to A.N.A.L.O.G. COMPUTING, P.O. Box 615, Holmes, PA 19043. No portion of this magazine may be reproduced in any form without written permission of the publisher. Program listings should be provided in printed form. Articles should be furnished as typed copy in upper and lower case with double spacing. By submitting articles to A.N.A.L.O.G. COMPUTING, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of A.N.A.L.O.G. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply self addressed, stamped envelope.

EDITORIAL

by Jon Bell

There have been many magazine articles and editorials written in the last few years about the so-called "computer whiz kids" of today. Almost all of these articles mention how wonderful it is that these kids are so proficient with computers, as well as mentioning the importance of computer literacy in the next decade or so. Educators beam and gush enthusiastically about how these kids (unlike many adults, or even the educators themselves) are not afraid of computers. Again, the phrase "computer literacy" enters into the conversation. Both the educators and the kids think it's wonderful.

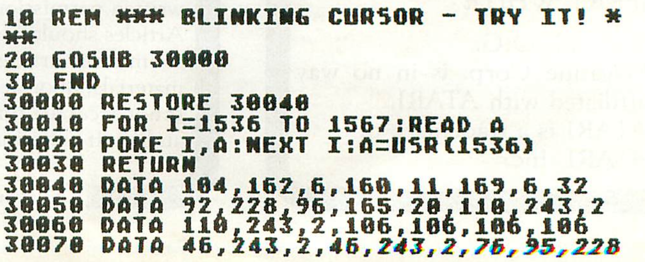
Personally, I think it's somewhat scary, and I'd like to explain why.

In the last year or so, I have met a number of kids half my age who know much more about programming than I. They have been brought up on video games and the technological god of rasterscan, and they have graduated to worshipping the higher order of the computer. I guess that there is a slight amount of envy here, since I perceive that these kids are going to become much more adept at programming than I will ever be. They had a head start on me, spending their adolescence (and in some cases, childhood) working hard, becoming computer literate.

What isn't so great is what I have seen in some (not all, thankfully) of these computer kids. I have met kids, who, in their quest to become computer literate, do not interest themselves in anything besides computers. I have read program submissions from kids who chronically misspell words; kids who have nice programs but cannot write documentation for their programs which can be published in A.N.A.L.O.G. unless the documentation is heavily edited. I have met kids, who, when they aren't in school, do not have any hobbies or play anything except for video or computer games. By sacrificing everything at the altar of the computer, these kids are becoming illiterate. It is my considered opinion that the only way to learn anything other than experiencing something directly is by reading. Reading is an interactive experience. When a child reads, his or her imagination interacts with the printed word and makes the events in the book come alive. Kids who won't read anything other than program documentation or the instructions included with a video game scare me. Or, more precisely, I feel incredibly sorry for anyone who chooses to be illiterate, or who chooses to remain uninformed about the rest of the world. As important as they are (and are becoming), computers aren't the only thing important on this planet. Basic reading and writing skills should come first.

It can be argued that one very good aspect of home computers and video games is that they can be a family activity, keeping the family unit together after dinner, promoting conversation and togetherness, an aspect of American home life that seems to be sadly lacking lately. Also, one can argue, working with computers is an interactive experience; a child is learning by doing. Fine, I agree with both of those points. However, other than the temporary enjoyment that a video game produces while you are playing it, what benefits does a video game provide? Publisher Lee Pappas mentioned to me the recent argument that playing video games increases eye-to-hand coordination and teaches a person how to be a good loser. (That is to say, it teaches persistence and determination; a person wants to top his high score.) I personally think those are rather specious benefits. What possible use is eye-to-hand coordination skill as applied to something other than the particular video game at which one is adept? Sorry, but I can't see that playing Star Raiders (my favorite game, incidentally) has made me more skilled at building models, driving my car or even pasting up the copy for this issue.

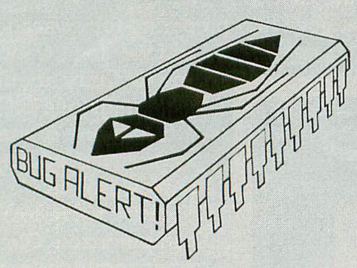
Lest many readers think that the previous statements seem surprising coming from the editor of a computer magazine, let me clarify my position by stating that I am not against computers or video games. (Obviously — otherwise, why would I be working here?) What I am against is the single-minded pursuit of any interest, to the exclusion of everything else. I believe that is called fanaticism. Parents—don't let your kids develop into "computer literate" illiterates. □



```
10 REM *** BLINKING CURSOR - TRY IT! *
**
20 GOSUB 30000
30 END
30000 RESTORE 30040
30010 FOR I=1536 TO 1567:READ A
30020 POKE I,A:NEXT I:A=USR(1536)
30030 RETURN
30040 DATA 104,162,6,160,11,169,6,32
30050 DATA 92,228,96,165,20,110,243,2
30060 DATA 110,243,2,106,106,106,106
30070 DATA 46,243,2,46,243,2,76,95,228
```

READER COMMENT

BUG ALERT!



Dear Editor:

Enclosed are details on a bug in the ATARI DOS (or possibly BASIC'S I/O code) which can clobber up to half of page 6, the "safe" RAM.

Since the operation of the bug isn't obvious, it can cause an incredible amount of frustration — I spent two days trying to find a non-existent bug in a USR sub, before I realized that DOS was the culprit.

Note that I am sending a copy of the information to several other magazines, so that it can be distributed as widely as possible.

Sincerely,
David H. Simmons
Redondo Beach, CA

BUG ALERT!!

RAM page six (1536-1791, $600-6FF) is NOT always "safe" memory.

A bug in either DOS (DOS I, DOS 2.0s, and OS/A tested), the BASIC cartridge, or the 10K OS ROM (Version A tested) causes the location of the disk's I/O buffer to move from around location 7000-8000 (decimal) to location 1408, which is only 128 bytes below location 1536, the beginning of page six.

This means that an input or output of more than 128 bytes will destroy some or all of the first 127 bytes of page six!

The program below demonstrates the change of address problem.

TESTING INSTRUCTIONS:
A: Type in program.
B: SAVE to disk as
"D:BUGTEST. BAS".
C: Type "RUN".

```
100 REM BUG DEMO
110 REM BY DAVID H. SIMMONS
120 DIM IN$(256)
130 OPEN #1,4,0,"D:BUGTEST.B
AS"
140 REM (OR OTHER 'TEXT RECO
RD' FILE)
150 PRINT "IOCB BUFFER ADDRE
SS: ";PEEK(852)+PEEK(853)*25
6
160 REM *(IOCB #1 ICBAL/ICBA
H)*
170 INPUT #1;IN$:INPUT #1;IN
$
180 PRINT "AFTER INPUT: ";PE
EK(852)+PEEK(853)*256
190 PRINT " +128= 1536, STAR
T OF PAGE SIX!":PRINT "I/O O
f more than 128 bytes WIPES
OUT anything in page six!"
200 CLOSE #1:END
```

Dear Editors:

The questions concerning the interference band which sometimes appears to roll up the screen on ATARI 400, 800, and VCS machines have an easy answer. ATARI failed to follow good engineering practice in the design of the power supply circuits. But, the deficiency is correctable for less than a dollar.

The interference is caused by the omission by ATARI of a component required in any power supply regulated by an integrated circuit. Without the omitted component, the regulator circuit can oscillate at a very high frequency. Such oscillation causes the interference band on the screen. ATARI uses only electrolytic capacitors to filter the power from the regulator. Electrolytic capacitors are only effective in eliminating low frequency oscillations.

So the corrective action required is to put in a filter to eliminate high frequency oscillations too. The component needed is a .01 micofarad ceramic capacitor available at a Radio Shack Store. One of them should be connected in parallel with each electrolytic capacitor in the power supply section of the console. The wires used should be cut as short as possible and connected by soldering. After these additions, the interference will be totally eliminated and the power supply will generate less heat.

Sincerely,
James Howard
Austin, Texas

Dear A.N.A.L.O.G.,

Congratulations are certainly in order to Michael A. Ivins for the magnificent program TRIPLE THREAT DICE, published in issue # 7. We enjoyed it tremendously.

The odds should be changed within the program so that (when the player wins), the player will receive his own money back, as well as the stated odds for that bet. For example, the even money bets in the program only returns the player's bet, if the player wins.

This can be corrected by changing the following lines:

```
265 WIN=WIN+(BET(0)*2):POS
ITION 6,21:? WIN;

280 WIN=WIN+(BET(1)*2):POS
ITION 6,21:? WIN;
```

Also, in the following lines, add one more count to the number following the asterisk.

How about a nice vacation on the beaches of France – for just $39.95?

At SSI, we think that our latest software for the Atari,® TRS-80® and Apple® — BATTLE FOR NORMANDY™ — is more than a great strategy game. We think of it as a great vacation package. After all, we are whisking you off to the northern coast of France for 25 days of fun and excitement (June 6 to 30, 1944) — all for just $39.95!

Of course, we're not promising peace and quiet. There is a war going on, you know....World War II!

We'll be taking you to the beaches of Normandy in style. Not on a mere jumbo 747 or an ocean liner, but an LCI (that's Landing Craft Infantry). Let's see the Joneses top that!

Like any good traveler, you need to plan ahead. What's the weather going to be like out there? Rough and stormy? Calm and gorgeous? It's hard to say, so you'd better be prepared for all kinds.

How about supplies? Well, we're a little tight on luggage space, so you'll have to juggle among the things you really need: fuel, general, and combat supplies.

How about some friendly sea bombardment to let the natives know you're coming? After all, you don't want trouble once you hit the beaches. The natives are a bit hostile at first, and a little naval artillery fire really helps to loosen them up.

And our service? Have we got good service — and plenty of it! Whole divisions and regiments of infantry, paratroopers, tank units, and commandoes are at your every beck and call. We do have to apologize for their different leadership and combat ratings. It's so hard to get good, consistent help these days.

You want sightseeing? You've got sightseeing! Nice historical towns like St. Lo, Cherbourg, and Caen — which you just have to take in (or take over, as the case may be.)

We know there's a lot that goes into preparing for a vacation, and we don't want you to worry about the details one bit.

SSI is one of the most advanced companies around because we're totally computerized. Our great computer program takes care of all the dirty work so all you do is enjoy. After all, you're on this trip for the fun and games, not work. And if you have a hard time finding friends to play with, who needs friends? You can play solitaire against the computer any time.

For complete details see the travel agent at your nearest game or computer store today!

SSI

STRATEGIC SIMULATIONS INC

As part of our demanding standards of excellence, we use maxell floppy discs.

If there are no convenient stores near you, VISA and M/C holders can order direct 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, 465 Fairchild Drive, Suite 108, Mountain View, CA 94043. (California residents, add 6.5% sales tax.)

Available in 32K cassette or 40K disc for the Atari® 400/800; 16K cassette for the TRS-80® Level II, Models I & III; and 48K disc for the Apple® II with Applesoft ROM card. Please specify which version when ordering.

WRITE FOR A FREE COLOR CATALOG OF ALL OUR GAMES.

Apple is a registered trademark of Apple Computer Inc. Atari is a registered trademark of Atari Inc. TRS-80 is a registered trademark of the Tandy Corporation.

```
295,310,2010,2110,2202,222
2,2232,2242,2252,2262,2272
```

Sincerely,
Henry L. Smith
Newburgh, N.Y.

Gentlemen:

We write in response to the question presented by Mr. Tucker, issue 7 or your magazine, how to draw a simple circle?

Presented below, while not exactly simple, is a routine which will draw a circle considerably faster than the one presented by the magazine.

This routine draws the circle 8 sides at a time by reflecting the calculated points of one seventh of the circle to the other seven sides.

Respectfully,

David M. Kampschafer
Programmer/Analyst

Lloyd D. Ollmann Jr.
Programmer-Consultant

Synergistic Software

```
10 XCENTER=310/2:YCENTER=
192/2
100 GRAPHICS 8
110 COLOR 1
120 ? "ENTER RADIUS:";:IN
PUT RADIUS
130 LET RADIUS=RADIUS+3-1
140 LET X=0
150 LET Y=RADIUS
160 LET DIAMETER=3-2*RADI
US
170 IF X<=Y THEN GOSUB 10
00:IF DIAMETER<0 THEN DIA
METER=DIAMETER+4*X+6:X=X+
1:GOTO 170
180 IF X>Y THEN END
190 DIAMETER=DIAMETER+4*(
X-Y)+10
200 Y=Y-1
210 X=X+1:GOTO 170
1000 REM
1010 PLOT XCENTER+X,YCENT
ER+Y
1020 PLOT XCENTER+Y,YCENT
ER+X
1030 PLOT XCENTER+Y,YCENT
ER-X
1040 PLOT XCENTER+X,YCENT
ER-Y
1050 PLOT XCENTER-X,YCENT
ER-Y
1060 PLOT XCENTER-Y,YCENT
ER-X
1070 PLOT XCENTER-Y,YCENT
ER+X
1080 PLOT XCENTER-X,YCENT
ER+Y
1090 RETURN
```

●

MANIAC IN 32K

We have received many requests for information on how to run issue no. 6's "Maniac" game in 32K machines. Stephen A. Vance, of Brooklyn, Ohio, has found a way to do just that. He writes:

"When Issue Number 6 of A.N.A.L.O.G. Magazine arrived in the mail, I was thrilled to see a machine language game in it. Unfortunately, the article states that the program requires 48K and I only have 32K. I decided to take a closer look to see if it really needed 48K. I discovered that (the program) was using address $8800 (34816 decimal) to $9000 (36864 decimal) for player-missile graphics. Since I only have 32K, the system can't use those addresses. Therefore, I went through the program changing the player-missile graphics to $6800 (26624 decimal) to $7000 (28672 decimal). After doing this, I rewrote it to tape again and it worked perfectly. This program was worth the effort."

Use the appropriate changes below to make "Maniac" run in 32K.

Note: the disk version will create a file called "MANIAC.OBJ". This is a binary file. Use DOS menu option "L" to load this file (no cartridge). The "Maniac" game will automatically start.

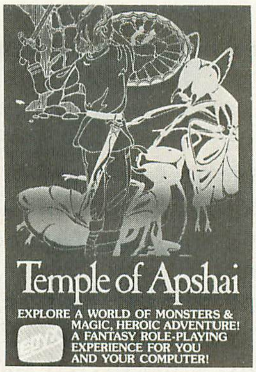
```
1 REM MANIAC 32K CASSETTE CHANGES
100 DIM X$(3984)
143 IF J=999 THEN 500
500 X$(184,184)=CHR$(104)
510 X$(1455,1455)=CHR$(105)
520 X$(1458,1458)=CHR$(105)
530 X$(1471,1471)=CHR$(105)
540 X$(1474,1474)=CHR$(105)
550 X$(1544,1544)=CHR$(105)
560 X$(1547,1547)=CHR$(105)
570 X$(2147,2147)=CHR$(104)
580 X$(2337,2337)=CHR$(105)
590 X$(2340,2340)=CHR$(106)
600 X$(2343,2343)=CHR$(106)
610 X$(2346,2346)=CHR$(107)
620 X$(2349,2349)=CHR$(107)
630 X$(3306,3306)=CHR$(106)
640 X$(3308,3308)=CHR$(106)
650 X$(3310,3310)=CHR$(107)
660 X$(3312,3312)=CHR$(107)
670 X$(3314,3314)=CHR$(105)
7490 DATA 999
```

●

```
1 REM MANIAC 32K DISK CHANGES
100 DIM X$(3984)
143 IF J=999 THEN 500
500 X$(184,184)=CHR$(104)
510 X$(1455,1455)=CHR$(105)
520 X$(1458,1458)=CHR$(105)
530 X$(1471,1471)=CHR$(105)
540 X$(1474,1474)=CHR$(105)
550 X$(1544,1544)=CHR$(105)
560 X$(1547,1547)=CHR$(105)
570 X$(2147,2147)=CHR$(104)
580 X$(2337,2337)=CHR$(105)
590 X$(2340,2340)=CHR$(106)
600 X$(2343,2343)=CHR$(106)
610 X$(2346,2346)=CHR$(107)
620 X$(2349,2349)=CHR$(107)
630 X$(3306,3306)=CHR$(106)
640 X$(3308,3308)=CHR$(106)
650 X$(3310,3310)=CHR$(107)
660 X$(3312,3312)=CHR$(107)
670 X$(3314,3314)=CHR$(105)
1000 OPEN #1,8,0,"D:MANIAC.OBJ"
5000 DATA 255,255,6,64,127,79,169,60,1
41,2,211,169,119,141,231,2,2031
7480 DATA 10,0,11,0,38,64,224,2,225,2,
38,64,0,0,0,0,678
7490 DATA 999
```

●

Slaying Monsters Should Be Mostly Fun and Games


Temple of Apshai
EXPLORE A WORLD OF MONSTERS & MAGIC, HEROIC ADVENTURE! A FANTASY ROLE-PLAYING EXPERIENCE FOR YOU AND YOUR COMPUTER!
EPYX

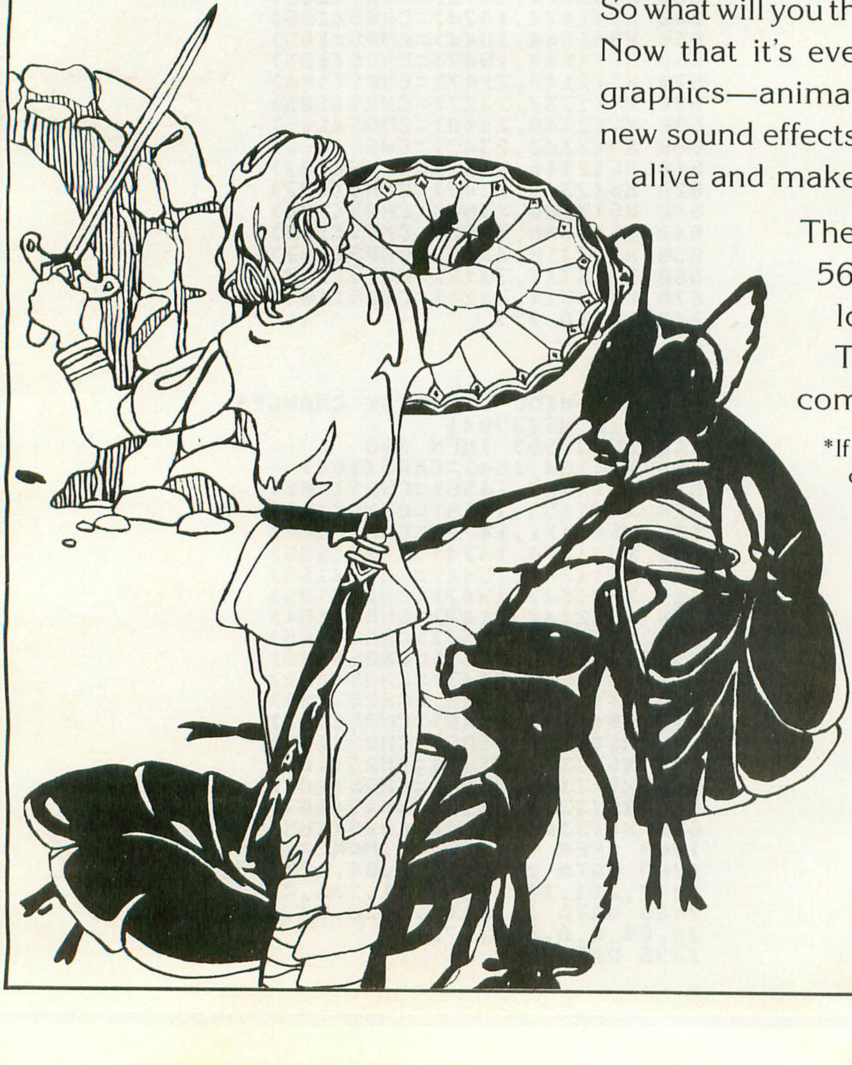
Be one of more than 16 million alter-egos that your computer can generate. Walk into a labyrinth filled with traps, treasures and monsters. There you'll test your strength, constitution, dexterity, intelligence . . . against thousands of monsters in over 200 caverns and chambers—growing in wealth, power and experience as you *progress* through the four levels of the dungeon.

Your character will do whatever you want him to do. Do battle—in real time—with the likes of giant ants, ghouls, zombies. . . Explore the various levels of the maze and discover the great treasures within.

The Game Manufacturer's Association named The Temple of Apshai the computer game of the year. The Temple is the very first computer game—ever to win the Hobby Industry award for excellence. There can be only one reason for that: it's a great game.

So what will you think of The Temple of Apshai now? Now that it's even better than ever. With better graphics—animated movement—and completely new sound effects that make your computer come alive and make The Temple even more fun.*

An EPYX game from Automated Simulations



The Temple has a superbly illustrated 56-page Book of Lore, and your local dealer has it for the ATARI,** TRS-80,** APPLE,** and IBM** computers.

*If you already have The Temple of Apshai, you can enjoy these great improvements. Just send us your original cassette or disk together with your check for $5.00 and we'll send you the brand new version—with better-than-ever graphics and sound.

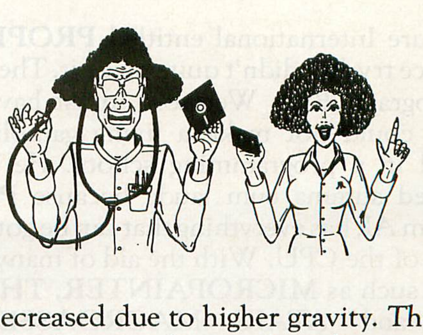
**ATARI, TRS-80, APPLE and IBM are trademarks of Atari, Inc., Tandy Corp., Apple Computer, Inc., and IBM, respectively.

©1981, **AUTOMATED SIMULATIONS, INC.,** P.O. Box 4247, Mountain View, Ca., 94040.

EPYX COMPUTER GAMES THINKERS PLAY

NEW PRODUCTS

by The Program Doctors

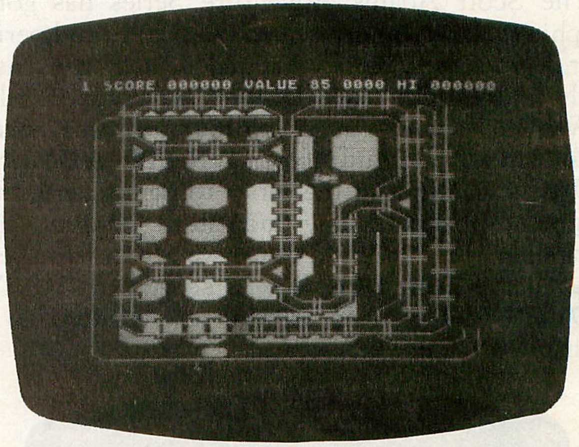


Things are happening fast in ATARI computerland. Business partnerships, new software, new joysticks, and an outstanding speech synthesizer have all come about since we last talked.

In a move to get a share of the burgeoning home computer market, CBS has signed a marketing agreement with K-Byte (**K-RAZY SHOOTOUT**). There will be a delay of about two months in the release of K-Byte's three new cartridges, **K-RAZY KRITTERS**, **K-RAZY ANTIKS**, and **K-STAR PATROL**, since all new packaging and advertising will be prepared by CBS. We've seen these games, and as we've told you before, they are well worth waiting for. This agreement, along with CBS's working agreement with Bally, is just the beginning of their move into the ATARI PCS market.

What would you do if you won $25,000? Put a down payment on a house, buy a Porsche, or pay your MasterCharge bill? After winning the first ATARI Star Award for **MY FIRST ALPHABET**, Fernando Herrera has done the only sensible thing and started his own software company. He has further capitalized on his achievement by naming his organization First Star Software. We have been told by a very reliable source that his **ASTRO CHASE** is dynamite. This all machine language arcade game is planned for release in late October.

The amount of new software that has recently been hitting the market is overwhelming and most of it is top-notch. One of the best comes to us from across the ocean. **AIRSTRIKE**, from English Software Company, has to be one of the hardest games we have ever played since Theo (for our new readers, Theo is our 800) became a member of our family. In a typical SYPFAD scenario (Save Your Planet From Absolute Destruction), this one or two player game artistically employs a horizontally scrolling screen as you fly your Mark V fighter-bomber across the Fortress of Gemini. You will encounter everything but the kitchen sink in this mission. Surface-to-air missiles attack you from below, meteor showers attack from above, fission bombs attack from everywhere and sliding airlocks attempt to block your path unless you can neutralize them with your laser cannon. Ammunition and fuel dumps dot the landscape, and you must destroy them to refuel your ship and reload your arsenal. (Your supplies are limited, naturally — we told you this game was hard!) As you progress to higher levels, the amount of fuel and ammo your ship can carry is decreased due to higher gravity. The key to this game is timing and control, as the speed of the enemy defense system is directly related to the speed of your fighter-bomber. This program can only be run with the data-lock key (included) that must be plugged into joystick port #4. This arcade quality game is outstanding in all respects with two exceptions. A joystick used in the conventional manner controls your ship, with the fire button shooting your laser cannons. But in order to drop a bomb you must hit the space bar. Unless you are a three-handed alien from Arcturus you will find this maneuver quite klutzy. Secondly, when you are hit, rather than starting your next fighter at the point of destruction, you are sent back to one of two starting points depending on how far you have progressed. This game is hard enough without these added frustrations.



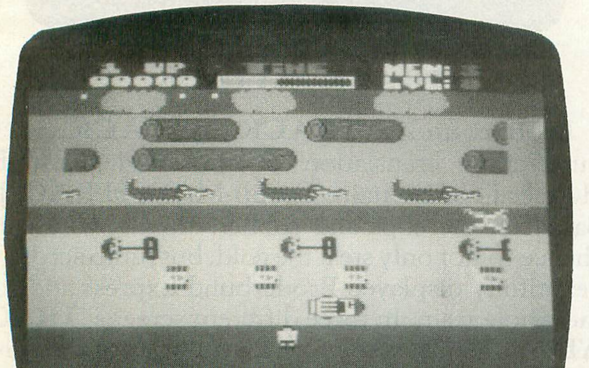
TRACK ATTACK

Broderbund's **TRACK ATTACK** is an outstanding translation of the original by Bill Hooper (The Solitaire Group-**MATCH RACER**). Based on the cops and robbers theme, you have a chance to not only steal the gold, but jump across the beautifully displayed Broderbund Express and steal the whole train. In three different screens, **TRACK ATTACK** boasts some of the finest graphics seen to date. Upcoming releases from Broderbund include **CHOPLIFTER, DAVID'S MIDNIGHT MAGIC, SEAFOX, STAR BLAZER, SERPENTINE, DUELING DIGITS, LABYRINTH,** and a 16K cassette version of **APPLE PANIC**.

Back in the dog days of the summer of '81, when we were dying for quality software, there was a football prediction program available from

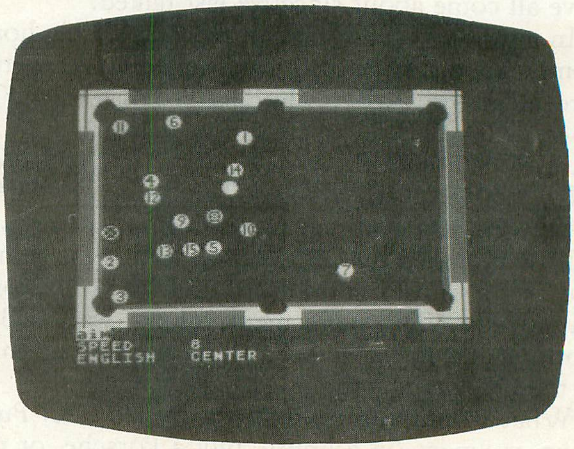
Adventure International entitled **PROPIX '81**. It was a nice try, but didn't quite make it. The author of that program, Russ Wetmore, must have realized that he could not make a living gambling so he enrolled in a programming school. He obviously graduated summa cum laude because **PREPPIE**, also from AI, has everything that can be gotten out of the bits of the CPU. With the aid of many excellent utilities such as **MICROPAINTER, THE NEXT STEP,** and **ATARI MACROASSEMBLER**, Wetmore has created a work of computer art that is destined to become a classic. In the guise of Wadsworth Overcash, a freshman fraternity pledge, you are sentenced to a full day of golf ball retrieval on the world's most dangerous course, the Nasty Nine. Hindering your progress are golf carts, lawn mowers, tractors, alligators, rolling logs, streams, and a huge killer frog. This program is outstanding in so many areas it must be seen to be fully appreciated. The playfield resembles a full-color cartoon and you will find yourself unconsciously humming the music. Obviously, you must avoid being hit by vehicles on the course, but we recommend you see it at least once to see how you look as a pancake. This game is so polished that even the instruction booklet must be mentioned. It's hysterical!

The Scott Adams Adventure Series has gone graphic!!! Now you can see all the puzzles and perils that have teased and frustrated you in the past. Unlike the On-Line Hi-Res adventures, the S.A.G.A. series uses animation, different perspectives, and full color. **PIRATE ADVENTURE** is the first of the series to be converted to *graphics. If this* is any indication of how the remaining eleven will be done, we are in for a real treat. Even if you have solved **PIRATE ADVENTURE** in its original text version, do it again! You'll enjoy it.



PREPPIE

The cartridge business is picking up. In the near future, all of the following programs will be available in ROM packs!: CROSSFIRE (On-Line); WIZARD OF WOR, DELUXE INVADERS, GORF (Roklan); **EMBARGO, FIREBIRD, RUSSKI DUCK** (Gebelli); and **SPEEDWAY BLAST** (I.D.S.I.). Already available is a ROM-crammed version of I.D.S.I.'s **POOL 1.5**, entitled **POOL 400**. The way that I.D.S.I. has taken their original 48K program, kept all the great features, and put it into a 16K cartridge is truly remarkable. **POOL 400** should satisfy all the Hoppes and Masconis who are memory and/or disk drive deficient. It is played with a joystick instead of a paddle; aligning shots and adding english is simple even for the youngest hustlers.

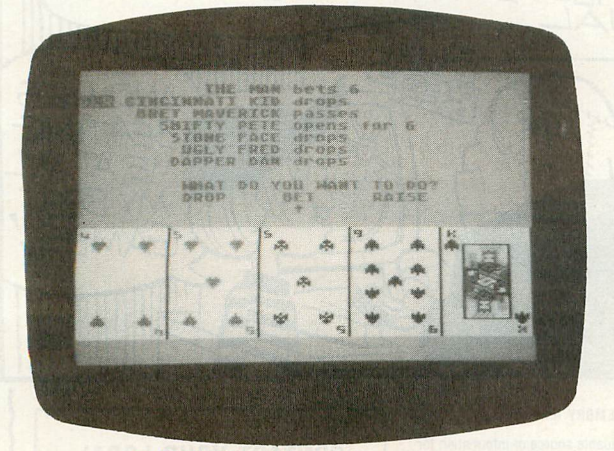


POOL 400

We heard a rumor that scented software (Eau do Cartridge) is on the way from...(are you ready for this!)...AVON! By this time next year expect a lot more cartridges from many companies, and probably a piece of hardware added to the ROM packs to stop those stupid cartcopy utilities (i.e. THE BLOCK - "back up your cartridges" - COME ON, BE SERIOUS! How many cartridges have you ruined?)

Jerry White hasn't been sleeping on his qwerty board. Since **TRIVIA TREK** he has translated **POKER TOURNEY** for Artworx, a five card draw tournament that will thoroughly relax the ardent arcaders. It's you against six other card sharks who each possess individual card playing characteristics. Based on the rules of the poker parlors in Gardenia, California, **POKER TOURNEY** is both challenging and stimulating. The cards appear as realistic as if you had pasted a deck on your TV.

Jerry has written many fine utility packages, but he has really outdone himself with **JERRY WHITE'S MUSIC LESSONS.** This outstanding tutorial from Swiftware will teach you everything you wanted to know about programming music and sound in ATARI Basic. There is a marvelous demo of special effects, and programs teaching sound generation, chording of all major scales, sound related poke locations, and a sing-a-long routine. Including **PLAYER PIANO**, (second in the series), *which* turns your computer into a twenty note mini-piano with a full, graphically accurate screen display of the piano keyboard, this package contains thirteen separate programs.



```
      THE MAN bets 6
CINCINNATI KID drops
 BRET MAVERICK passes
    SHIFTY PETE opens for 6
     STONE FACE drops
      UGLY FRED drops
     DAPPER DAN drops

  WHAT DO YOU WANT TO DO?
  DROP      BET      RAISE
```

POKER TOURNEY

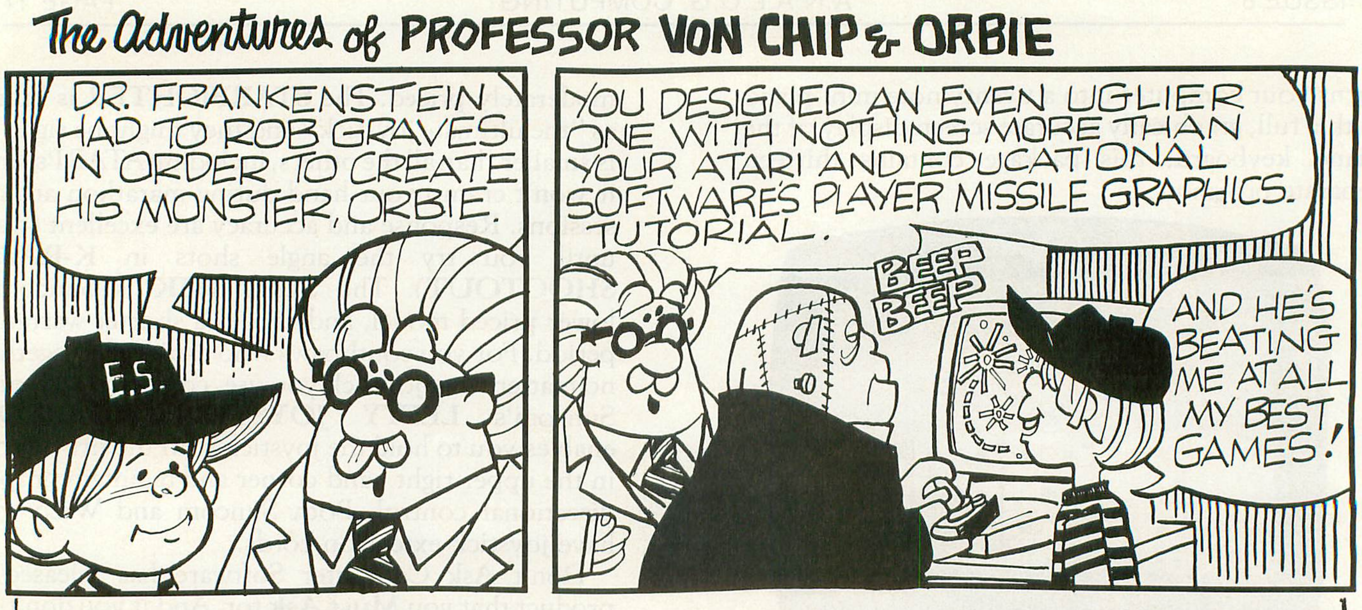
Several programs designed for the youngest members of your family have been released from Sub-Logic. These programs are geared for children ages 4-12, and include two arcade games and three adventure type games with graphics. **ROBBY, THE ROBOT CATCHER, SKY RESCUE, ADVENTURE ON A BOAT, GHOSTLY MANOR,** and **THE BLACK FOREST** are all simple enough for the kids to play well, yet they incorporate enough challenge to hold their attention. Sub-Logic has not forgotten about Mom and Dad and will be converting their popular pinball simulation **A2-PB1** in the very near future. With ten levels of play and 40-user adjustable parameters for creating your own customized games, this program promises to be something special.

Three companies have come to our rescue in the "when will somebody make a better joystick?" department. After thorough testing on several different types of games, we feel that two of these companies have come close but one of them really seems to know what you want. Wico, the world's largest manufacturer of commercial arcade controls, is introducing a line of joystick products called **COMMAND CONTROL**, the first of which is now available. This is a very impressive looking piece of equipment, however, it is very large, somewhat loose, and lacks consistent accuracy. Game-Tech's new joystick, **PRO-STICK**, lacks smooth cornering ability in the maze games, and directional control in firing is inconsistent. This product is also very tiring to use, due to the size of the base. It does not fit comfortably in your hand, and even placing it on a hard surface did not solve the problems. Both of these products surpass the ATARI joystick only in durability, and in comparison they are both rather expensive. But don't give up hope yet. Suncom has released a line of products that not only solve the above mentioned design problems, but are moderately priced. The **STARFIGHTER** is billed as "the ultimate joystick" and they might be right. It is smaller than all the others, including ATARI's, and it won't cramp your hand during marathon arcade sessions. Response and accuracy are excellent (wait until you try the angle shots in **K-RAZY SHOOTOUT**). The **SLIK STIK** is Suncom's lower priced model, and carries a shorter warranty period. For you southpaws who can't get it together no matter what joystick you use, complain no more. Suncom's **LEFTY JOYSTICK ADAPTOR** enables you to hold the joystick with the fire button in the upper right hand corner and maintain proper directional control. Both Suncom and Wico will have joystick extension cords.

Don't Ask Computer Software has released a product that you **Must Ask** for. And if you don't, it can ask for itself. We're talking about **S.A.M.**, the Software Automatic Mouth, a speech synthesizer on DISK!! No expensive hardware to buy! This exciting breakthrough gives you an unlimited vocabulary, full inflection control, separate speed and pitch control and more for only $59.95. Accessed easily from Basic, **S.A.M.** will amaze your friends and breaks the ground for an entire new generation of software. This program is a must for every ATARI computer owner.

SLIK STIK™
SLIK STIK
STARFIGHTER
THE ULTIMATE JOYSTICK
Starfighter™

By now we're sure you have noticed that our column has changed. With this new format, we can bring you more information, keep you more current with the industry and cover a *wider* range of products. We would appreciate any comments, suggestions or criticisms (take it easy on the bad stuff — our egos are very fragile). □


The Adventures of PROFESSOR VON CHIP & ORBIE
DR. FRANKENSTEIN HAD TO ROB GRAVES IN ORDER TO CREATE HIS MONSTER, ORBIE.
E.S.
YOU DESIGNED AN EVEN UGLIER ONE WITH NOTHING MORE THAN YOUR ATARI AND EDUCATIONAL SOFTWARE'S PLAYER MISSILE GRAPHICS TUTORIAL.
BEEP BEEP
AND HE'S BEATING ME AT ALL MY BEST GAMES!

TRICKY TUTORIALS (tm)

There are many things that the ATARI computers can do either better, or easier than other small computers. The following series of programs is designed for anyone who is at least familiar with BASIC programming. What each tutorial offers is similar to an extensive magazine article with all discussion in as simple language as possible, plus you get MANY examples already typed in and running. The instruction manuals range from 10 to 50 pages, and some tutorials fill up a complete tape or disk. There is little overlap in what is taught, so anyone wanting to know all they can should buy them all (my banker thanks you). ATARI buys these from us to use in training their own people! Rave reviews have been published in ANTIC, ANALOG, CREATIVE COMPUTING, and even INFOWORLD. You trust INFOWORLD, don't you?

TT #1: DISPLAY LISTS—This program teaches you how to alter the program in the ATARI that controls the format of the screen. Normally, when you say "Graphics 8", the machine responds with a large Graphics 8 area at the top of the screen and a small text area at the bottom. Now, you will be able to mix various Graphics modes on the screen at the same time. The program does all of the difficult things (like counting scan lines). You will quickly be able to use the subroutines included in your own programs.
16K Tape or 24K Disk. **$19.95**

TT #2: HORIZONTAL/VERTICAL SCROLLING—The information you put on the screen, either GRAPHICS or TEXT, can be moved up, down, sideways, or diagonally. We provide the basic methods and leave the rest up to your skill and imagination. Includes 18 examples to get you started, with several using a small machine language subroutine for smoothness.
16K Tape or 24K Disk. **$19.95**

TT #3: PAGE FLIPPING—Now you don't have to redraw the screen every time you change the picture or text. You will learn how to have the computer draw the next screen you want to see while you are still looking at the previous screen, then flip to it instantly. You won't see it being drawn, so a complicated picture can seem to just appear. Depending on your memory size and which graphics or text modes you are using, you can instantly look at up to 50 pages. The basic method takes only 9 lines and the usefulness is infinite.
16K Tape or 24K Disk. **$19.95**

TT #4: BASICS OF ANIMATION—This program shows you how to animate simple shapes (with some sound) using the PRINT and PLOT commands, and it also has a nice little PLAYER/MISSILE GRAPHICS game you can learn from. The P/M example is explained and will get you started on this complicated subject (more fully explained in TT #5). This would be an excellent way to start making your programs come alive on the screen with movement! Recommended for beginning users.
16K Tape or 24K Disk. **$19.95**

TT #5: PLAYER/MISSILE GRAPHICS—Learn to write your own games and other animated applications! The tutorial begins with many small examples that compliment the 50 page manual, then gradually builds up to a complete game where everything you need to know is fully explained. Also included are two machine language utilities that you can use to animate Players with from BASIC. Next we include two of the best editors currently available; one for editing playfield shapes (backgrounds); and one to edit your players, and all in glorious Technicolor!! Everything except the two editors run in 16K Tape or 32K Disk. **$29.95**

TT #6: SOUND AND MUSIC—Unless you have spent many years experimenting with the four voice channels, you will learn a lot from this one! Learn to play standard notes, chords, and whole songs using some simple "tricks". One of the nicest parts are the examples of special sound effects that you can refer to whenever you need a sound for a program or to impress a friend. This program will be of interest to all ages and levels of experience!
16K Tape or 24K Disk. **$19.95**

SPECIAL DISCOUNT

Order the first six tutorials in a 3-ring binder for $99.95, a $30.00 savings!

TT #7: DOS UTILITIES—We at Educational Software have been shocked by some of the prices others are charging to offer you small utilities to help in the use of your Disk Drive. We now offer you all of the following plus explanation as to how each was written, and how to use them: A UNIQUE MENU PROGRAM, AN AUTORUN.SYS BUILDER, DISK INSPECTOR (LOOK AT SECTORS), DISK JACKET PRINTER, AUTOMATIC FORMATTER, RECORD SAVE AND LOAD UTILITY.
32K Disk Only. **$29.95**

MASTER MEMORY MAP (tm)

This book is the most valuable source of information for your ATARI you can buy. It starts out by explaining how to PEEK and POKE values into memory, so that even new computer owners can use many of these "Tricks". Then you are given 32 pages of the memory locations that are the most useful, along with hints on how to use many of the locations. Finally, it includes hints on problems you may be having with the computer and discusses the new Graphics modes 9 to 11. Even ATARI buys this book from us! **$6.95**

USER SUBMITTED PROGRAMS

MINI-DATABASE/DIALER—stores and edits up to 8 lines of information such as names & addresses, phone numbers, messages, inventories, or anything you want. It has the usual sort, search, and print options, but it also has an unusual feature: If your file includes phone numbers and your phone company allows touch-tone phone signals, the program will DIAL THE PHONE NUMBER FOR YOU! 16K Tape or 24K Disk. **$24.95**

THE GRAPHICS MACHINE—allows the ATARI to act like more expensive graphics computers using simple commands like line, box, circle, polygon, fill, and savescreen to get a high resolution picture you can save on disk in only five seconds! Many more features! 48K Disk Only. **$19.95**

BOB'S BUSINESS—14 small business type programs accessed from a common menu. 16K Tape or 32K disk. **$14.95**

OUR NEWEST PROGRAMS

*** OUR BEST GAME ***

SPACE GAMES—Our family is being attacked by ALIENS, and only you can save us. A comic book manual will guide you through three games that test your ability in space skills. Includes ALIENS, SURVIVE, and ROBOT ATTACK, and is for all ages. The first two games require 16K for Tape. The last game and all Disk users need 32K. **$24.95**

MATHS FOR FUN—Another ENGLISH import teaching basic math skills. Very colorful and enjoyable to use. For ages 5 to 16. 16K Tape or 24K Disk. **$19.95**

TT #10: SOUND EFFECTS—From laser blasts to ringing phones, this tutorial will show you how to make unique sound effects in all of your programs! 16K Tape or 24K Disk. **$19.95**

CONTACT YOUR LOCAL DEALER or ORDER BELOW

We have other fine programs for ATARI computers.
Write for a catalog.
Send us your programs to sell too!

MINI WORDPROCESSOR—A simple text editor to write, save, and print several pages at a time. 32K Tape or Disk. **$19.95**

KID'S #1—Includes a MATH QUIZ, a children's TREASURE HUNT, and a DIALOGUE program. 16K Tape or 24K Disk. 3 for . . . **$14.95**

KID'S #2—SPELLING BEE, WORD SCRAMBLE, and TOUCH. 16K Tape or 24K Disk. 3 Educational games for . . . **$14.95**

PLAYER PIANO—Turns your keyboard into a mini-piano and more. 24K Tape or 32K Disk. **$14.95**

DOG DAZE—Two cute little doggies race for the fire hydrants, shoot their bones, and just have a lot of fun! A fast action program for all ages. 8K Tape or 16K Disk, in machine language. **$16.95**

GRAPHIC SYMBOL LABELS—for your keyboard to remind you of the built-in Graphics symbols. 2 complete sets for . . . **$2.95**

MARATHON—This is a unique math quiz for one or two players. You are in a race to move your runner across the screen first! There are four levels of play with five modes of operation for each. The game uses joysticks for all input, so play is easy for young children. This wonderful learning tool is imported from ENGLAND for your learning pleasure. Your kids will never even notice they are playing an EDUCATIONAL program. 16K Tape or 24K Disk. **$19.95**

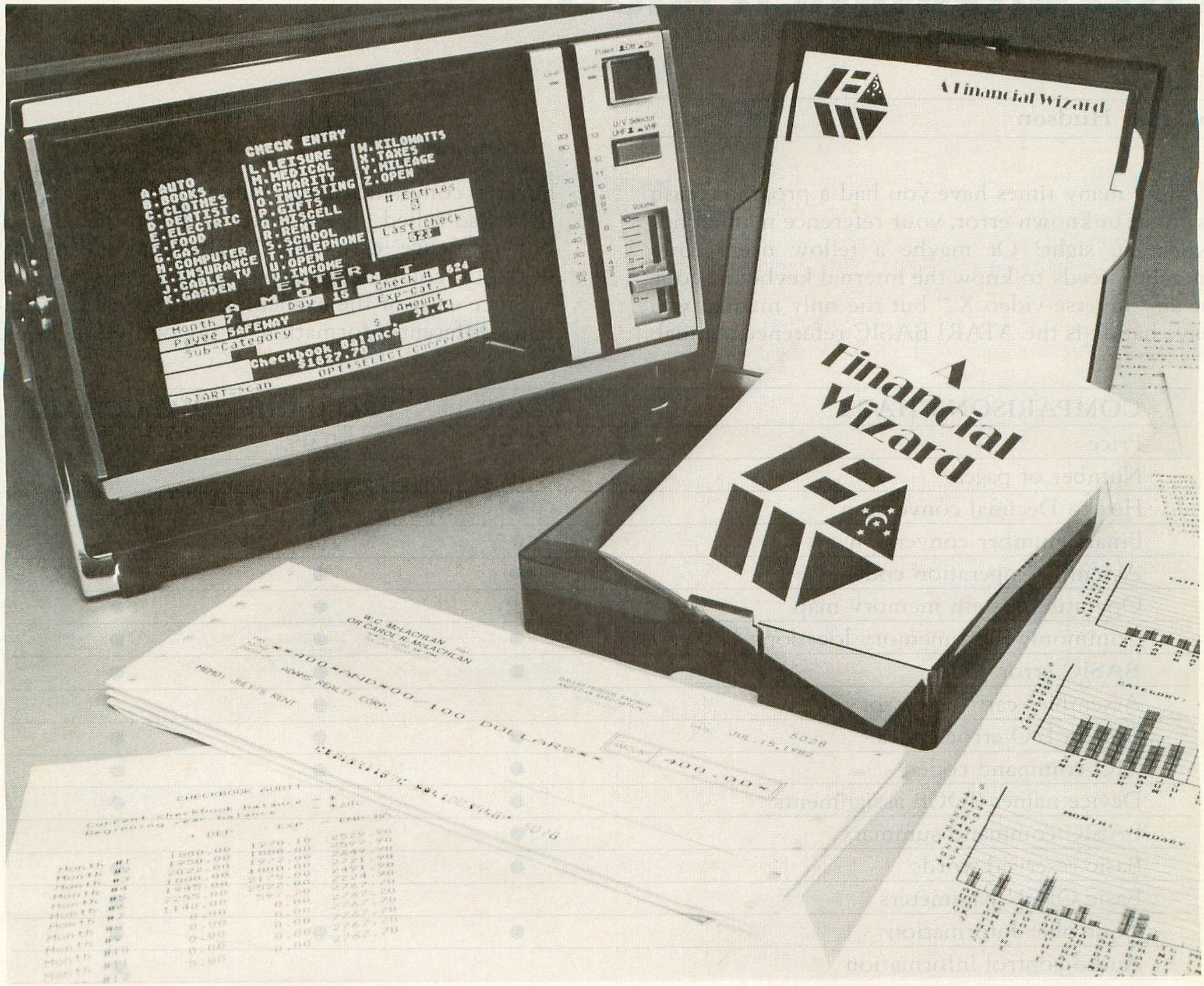
TT #8: CHARACTER GRAPHICS—Character Graphics is the best way to animate your ATARI! Make letters look like space monsters, gunfighters, or a myriad of other shapes. Use our editor to create these multicolor shapes and then we'll show you how to move them around the screen. This tutorial even shows how our Space Games were written! 16K Tape or 24K Disk. **$19.95**

To order COD, VISA or MasterCard call — **(408) 476-4901**

By mail include $3.00 postage (USA only) or $1.50 Memory Map only—
California residents add 6.5% TAX.—Specify Tape or disk.

Educational
Software inc.

4565 Cherryvale Avenue
Soquel, Ca. 95073
(408) 476-4901



CHECK ENTRY
A.AUTO
B.BOOKS
C.CLOTHES
D.DENTIST
E.ELECTRIC
F.FOOD
G.GAS
H.COMPUTER
I.INSURANCE
J.CABLE TV
K.GARDEN
L.LEISURE
M.MEDICAL
N.CHARITY
O.INVESTING
P.GIFTS
Q.MISCELL
R.RENT
S.SCHOOL
T.TELEPHONE
U.OPEN
V.INCOME
W.KILOWATTS
X.TAXES
Y.MILEAGE
Z.OPEN
# Entries
Last Check 623
ENTER AMOUNT
Check # 624
Month 7
Day 15
Exp-Cat. F
Amount 39.44
Payee SAFEWAY
Sub-Category
Checkbook Balance $1627.70
START-Scan
OPT/SELECT-Correction
A Financial Wizard
W.C. McLACHLAN OR CAROL R. McLACHLAN
ADAMS REALTY CORP.
400AND**00/100 DOLLARS**
JUL.15,1982
5028
MEMO JULY'S RENT
CHECKBOOK AUDIT

PROGRAMMER AIDS

by Tom Hudson

How many times have you had a program crash with an unknown error, your reference manual nowhere in sight? Or maybe a fellow user group member needs to know the internal keyboard code of an "inverse-video X," but the only manual you have handy is the ATARI BASIC reference manual.

Several companies have followed the example of IBM's so-called "green card" and introduced fold-out reference cards specifically for the ATARI 400/800 computer systems.

Since each of the available reference guides contains different information, in my opinion it would

COMPARISON CHART	ACE	PRO CARD	WALL CHART
Price	$9.95	$9.95	$16.95
Number of pages	16	13	1
Hex to Decimal conversion	●	●	
Binary number conversion	●		
Assembler operation codes	●	●	
Operating system memory map		●	●
Commonly used memory locations	●	●	●
BASIC error codes	●	●	●
Assembler cartridge error codes		●	
System I/O error codes	●	●	●
XIO command codes	●	●	●
Device names/IOCB assignments	●		●
BASIC command summary	●		
Basic reserved words	●		●
Basic OPEN parameters			●
Controller information	●	●	●
Audio control information			●
Audio frequency/note conversion	●		●
ATASCII values	●	●	●
Internal keyboard codes	●	●	
Character set displacements		●	
BASIC/O.S. graphics modes	●	●	●
Graphics point sizes	●		
Graphics screen limits	●		●
Bytes per graphics line	●		
Graphics total RAM requirements	●		
Color assignments for graphics modes	●	●	
Color value information	●		●
Display list format		●	
Player-missile memory map	●	●	
Console switches		●	●
LIST/SAVE command formats	●		●
ATARI 825 printer codes		●	
Epson MX-80 printer codes		●	
NEC 8023A printer codes		●	

be unfair to the companies involved to make a judgment based on my personal needs. Therefore, I will present a comparison chart describing the contents of each guide, allowing each reader to make a decision based on his or her requirements.

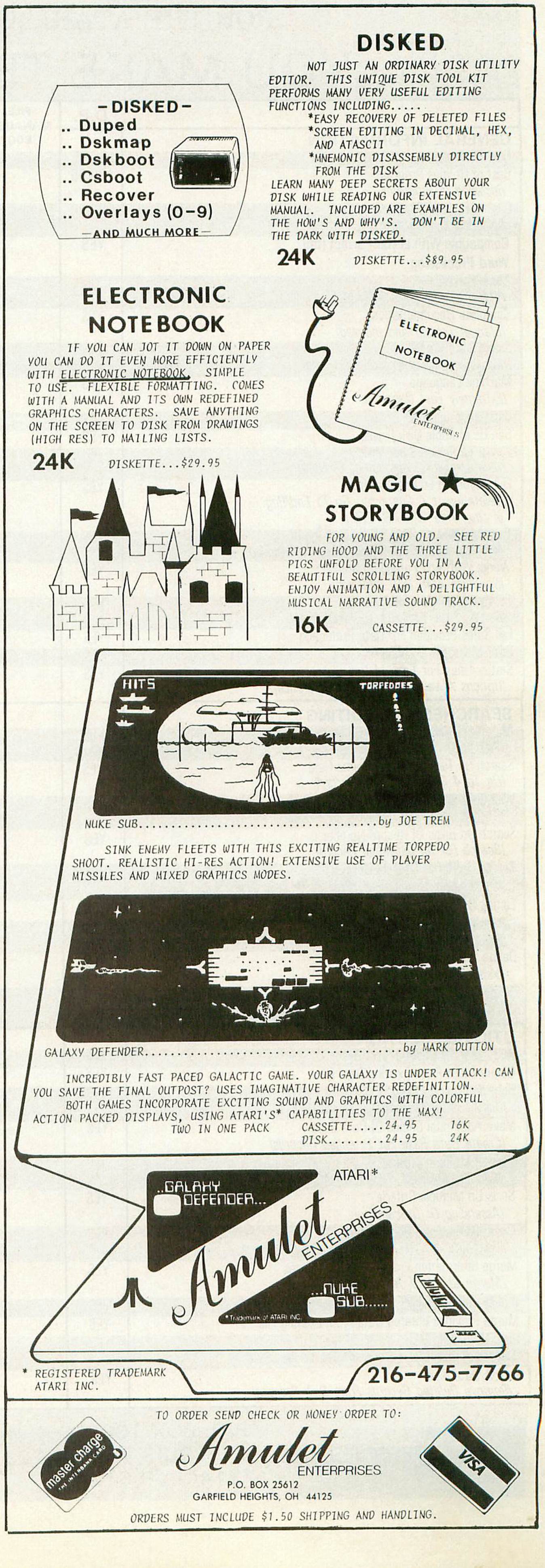
The three programmer's guides that will be considered here are the Pocket Reference Card from Advanced Computing Enterprises, the Pro Card from On Line Computing centers OKC, and the Programmer's Guide wall chart from The Computer Center. The Pocket Reference Card and the Pro Card are both similar to IBM's "green card." That is, they both are pocket-sized, fold-out reference guides made from heavy card stock. The Programmer's Guide wall chart is a huge poster-sized sheet meant for "at-a-glance" reference above the computer.

As one can easily see by studying the chart, each one of the available programmer's aids has both strong and weak points. Hopefully, this comparison will make ATARI computer owners more aware of such differences. □

Advanced Computing Enterprises (ACE)
5516 Rosehill, Shawnee, KS 66216

On Line Computing Centers OKC
10944 A North May Ave.
Oklahoma City, OK 73120

The Computer Center
P.O. Box 171, Old Saybrook, CT 06475



DATA PERFECT

FOR THE ATARI 400 AND 800 COMPUTERS

YOU MAKE THE COMPARISON

	D.P.	FILE MANAGER "800"
GENERAL INFORMATION		
Cost of Program	$99.95	
Cost of Utilities Program *(Included In Program)*	$00.00	
Cost of Reports Program *(Included In Program)*	$00.00	
Compatible With Letter Perfect (tm) Word Processing	YES	
Menu Driven *(Very User Friendly)*	YES	
Complete Documentation *(Manual Tabbed And Indexed)*	YES	
Single Load Program *(No Swapping Of Program Diskette)*	YES	
Machine Language *(Extremely Fast Operation)*	YES	
Can Use Single Disk Drive	YES	
Can Us Multiple Disk Drives	YES	
Ability To Design Screen Mask *(User Designs Arrangement Of Data)*	YES	
Full Keyboard Editing Available *(Delete/Insert A Character; Go To End/Beg. of Line; Fine 'n', TAB, ETC.)*	YES	
Compatible With Bit 3 80-Column Board *(40-Column and 80-Column Version Available)*	YES	
Works With Any Parallel Printer *(Supports Atari 850 Interface)*	YES	
Totals Of Numeric Field *(Return Total And Average Value/Field)*	YES	
Fail Safes Provided For Data Protection	YES	
Error Messages Displayed	YES	
Status Lines For Ease of Use *(Options Always Available For Reference)*	YES	
SEARCHES AND EDITING		
Multiple Searches Allowed On Same Record *(Search On 9 Criteria Per Record)*	YES	
Search On Two Criteria In Same Field *(Up To 4 Fields In Single Record)*	YES	
Wild Card Searches *(And/Or, Include, Character, Or Block)*	YES	
Search On Basis Of Record Number *(Search For An Individual Record)*	YES	
Search On Range Of Data Desired *(Dates, Numbers, Values, Greater Or Less Than, Equal To, etc.)*	YES	
Editing Of Records Individually	YES	
Editing Records Globally *(Verification Allowed)*	YES	
Delete Records Individually *(Verification Allowed)*	YES	
Deleting Records Globally *(Verification Allowed)*	YES	
UTILITIES SECTION		
Add Fields To Existing Data Base	YES	
Delete Fields From Existing Data Base	YES	
Reformat A Data Base *(Copy Format Of Existing Data Base)*	YES	
Make Additional Copies Of Data Base *(Create Data Base For Extended Records)*	YES	
Sort on Multiple Criteria *(Sort On Basis Of 4 Fields In A Sort)*	YES	
Sorts On Multiple Criteria *(Assending Or Descending)*	YES	
Depth Of Sort Can Be Changed *(Designate Number Of Charters Deep To Sort)*	YES	
Merge Information From Other Data Bases *(Merge Standard Text Files)*	YES	
Add Or Delete Fields From Data Base	YES	
Merge Previous Entered Data From Existing File	YES	
Back Up A Data Base *(Make A Back Up Of Current Source Data)*	YES	
Pack A Data Base *(Remove Deleted Records From Disk Storage)*	YES	

	D.P.	FILE MANAGER "800"
REPORT GENERATOR		
Design Report To User Specifications	YES	
Level Breaks Allowed At Users Option *(Up To 4 Level Breaks Per Report)*	YES	
Designate Font To Be Used In Report	YES	
Boldfacing Allowed In A Report *(With Dot Matrix Printer)*	YES	
Mathematical Formulas Allowed In Report *(Example, Field 'x' + Field 'y' = Field 'z')*	YES	
Auto Page Number Allowed In Report	YES	
Auto Date Entering Allowed In Report	YES	
Repeating Characters Allowed	YES	
Optional Level Breaks and Page Breaks When Sort Values Change	YES	
Up To 7 Lines Allowed For Header on Each Report	YES	
Up To 2 Lines Allowed For Detail Information On A Report	YES	
Variable Spacing Allowed Between Data On Items In A Report	YES	
Multiple Fields Allowed In A Report *(Number, Date, Alpha, Formula)*	YES	
Search Criterian Allowed On Report *(Same Criteria As In Editor)*	YES	
Ability To Have "Literal" Data *Printed In A Report*	YES	
Ability To Have "Conditional" Data Printed In A Report	YES	
Use A Default Date Field	YES	
Designate Default Value For Specific Fields	YES	
LABELS REPORT GENERATOR		
Mailing Labels Allowed *(Specifically Designed For Labels)*	YES	
User Designs Data Placement On Label *(One Across Label Design)*	YES	
Multiple Fields Allowed On Label *(Date, Alpha, Numeric, Formula)*	YES	
Repeating Characters Allowed	YES	
Front Designation Allowed	YES	
Print Labels On A Conditional Basis	YES	
Search Criteria Valid On Label *(Same Search Criteria As Editing)*	YES	
MATHEMATICAL ABILITIES		
Basic Math Calculation Addition, Substraction, Multiplication, Division	YES	
Built In Calculator (Automatic) *(Use In Editing, Or Adding Data)*	YES	
Find the Integer Value Of A Numeric Expression	YES	
Find The Log Base 'e' Of 'x'	YES	
Find The Log Base '10' Of 'x'	YES	
Find The Absolute Value Of 'n'	YES	
Exponentential Notation Used	YES	
Find The Square Root Of 'n'	YES	
Formulas Allowed Between Fields [Field × /+ − */) Field y = Field z] [Field × /+ − */) N = Field Y]	YES	
SPECIFICS		
Maximum Number Of Fields Per Record	32	
Maximum Number Of Formulas In A File	16	
Maximum Length Of A Field	127	
Maximum Record Length	511	
Maximum Number Of Level Breaks	4	
Records Per Diskette *(Depends On Length And Number Of Fields)*	VAR.	
Data Bases Allowed On Each Diskette *(Can Be Expanded To Additional Diskettes)*	ONE	
Form Letter Capability *(Compatible With Letter Perfect)*	YES	

LJK ENTERPRIES, INC.

P.O. BOX 10827
ST. LOUIS, MO. 63129
[314] 846-6124

Atari - A Trademark of Atari Inc., Full View 80 a Trade Mark of Bit 3 Computers, Fill Manager 800 A Trademark of Synapse Software, Letter Perfect, Data Perfect A Trademark of LJK Enterprises, Inc.

D:CHECK

16K DISK

by Istvan Mohos

EDITOR'S NOTE:

For the last few issues, A.N.A.L.O.G. has been printing program listings in a 38-column format for easier entry into reader's computers. We have found that errors can still sneak in (especially when typing marathon-length programs such as issue no. 6's "Maniac"). In order to help those readers with disk drives check to see if they have made any typos, with this issue we are introducing "D:CHECK", a debugging aid. It will be used for those listings that are longer than average, or have many DATA statements. Such program listings will have a table of values listed after them. This is a "checksum" table, explained in the following article. Unlike other debugging aids, D:CHECK will tell you the exact line in which an error resides. It can also be used with BASIC A+ with no modification.

Boot the system with DOS II and "D:CHECK" SAVEd onto disk. Type in your program to be debugged. LIST the program to disk. Type: RUN "D:CHECK". At the prompt type in the name of your program.

Your program will now be read 3 consecutive times from the disk. The first reading (lines 190-200 of 'CHECK') counts the lines of your program, establishing the variable 'LINECOUNT'. The second reading (lines 240-310 of 'CHECK') tabularizes every 15th actual line number of your program: R(0) contains your first line number, R(1) holds your 16th line number, R(2) holds your 31st line number, etc.

Unlike the line-by-line input of the first two readings, the third reading of your program is byte-by-byte (lines 330-360 of 'CHECK'). Beginning with the line number of the first line of your program, the ATASCII equivalent of each letter (including spaces, special characters, inverse video) are summed up. The resulting 'CHECKSUM' undergoes a module division which strips off digits higher than hundreds. The remaining 3-digit number represents uniquely, the first line of your program. The process repeats, resulting in a checksum for each line of your program, contained by array C(LINECOUNT).

Lines 370-450 of 'CHECK' write a program listing to disk under the name 'BUG'. 'BUG' is a series of data statements. The first line of 'BUG' might look like this:

```
2001 DATA 389,972,454,31,721,555,127,1
26,323,442,453,571,10,920,865,6959
```

where 2001 is the starting line of your program, 389 is the checksum of the first line of your program, 972 is the checksum of the second line of your program, etc., 6959 is the total of 15 checksum numbers of data line 2001: use it to check a range of 15 lines at once.

The line number of the second line of 'BUG' (2095 DATA for example) coincides with the 16th line of your program: the numbers in '2095 DATA' are checksums of the 16th to the 30th lines of your program.

When the 'CHECK' run is finished, you are instructed to wipe the memory by typing 'NEW' and then ENTER "D:BUG". Type 'LIST' to view the data statements, use 'CONTROL 1' to stop scrolling if your program had more than 165 lines.

The published source of the program you typed in should have an identical set of data statements listed after it. This was produced by a run of 'CHECK' reading the original listing of the published program.

Compare screen versus print, the last number of the first statement. If it matches, you have 15 perfect lines: continue with the last number of the next statement. If there is a discrepancy, check each number in that data statement. If the fifth number in screen line 2164 (for example) does not match with the fifth number of the printed data line 2164, the bug will be found in the fifth line of the typed-in program, counting from line 2164.

Note all the discrepancies, then ENTER "D:your prog" (the typed-in program) to make the necessary corrections. Your program will automatically wipe the 'BUG' data by writing its own lines over it. Re-LIST your program to disk after you have the corrections, then re-RUN "D:CHECK" for a double-check. □

```
100 REM CHECK DEBUGGING AID
    BY  ISTVAN MOHOS
110 GRAPHICS 0:? :? "This run will LIS
T data statements    with the name: BU
G, to the disk."
120 ? :? "The BUG DATA is created by e
valuating each character of a user pro
gram,      LISTed to disk.":?
130 ? "Replace the word 'USER' in line
 150    with the name of your program;
        then type RETURN CONT RETURN."
135 PIK=PEEK(559):Z=0:REM constants
140 POSITION 2,15:LIST 150:POSITION 20
,11:STOP
150 OPEN #1,4,0,"D:USER"
160 ON X GOTO 230,330
170 ? "K":? "DISABLING SCREEN...STAND
BY...":FOR I=1 TO 800:NEXT I:POKE 559,
Z:REM debug before poking
180 LINECOUNT=Z:DIM I$(126)
190 TRAP 210:INPUT #1;I$:LINECOUNT=LIN
ECOUNT+1
200 GOTO 190
210 CLOSE #1:Q=INT(LINECOUNT/15):DIM C
(LINECOUNT),R(Q),S$(5):IF (LINECOUNT=Z
 OR I$="") THEN 560
215 IF ASC(I$(1,1))<48 OR ASC(I$(1,1))
>57 THEN 560
220 X=1:GOTO 150
230 RANGE=Z:LINE=Z:FOR I=1 TO 5:S$(I,I
)=" ":NEXT I
240 COUNT=Z
250 INPUT #1;I$:T=1:COUNT=COUNT+1
260 IF I$(T,T)<>" " THEN S$(T,T)=I$(T,
T):T=T+1:GOTO 260
270 LINE=VAL(S$)
280 R(RANGE)=LINE:RANGE=RANGE+1
290 TRAP 320:INPUT #1;I$
300 COUNT=COUNT+1:IF COUNT=15 THEN 240
310 GOTO 290
320 CLOSE #1:X=2:GOTO 150
330 FOR I=1 TO LINECOUNT:CHECKSUM=Z
340 GET #1,NUMBER:PRODUCT=X*NUMBER:CHE
CKSUM=CHECKSUM+PRODUCT:X=X+1:IF X=4 TH
EN X=1
345 IF NUMBER=155 THEN 360
350 GOTO 340
360 CHECKSUM=CHECKSUM-1000*INT(CHECKSU
M/1000):C(I)=CHECKSUM:NEXT I
370 CLOSE #1:OPEN #1,8,0,"D:BUG":LINE=
R(Z):ITEM=Z
380 COUNT=15:TOTAL=Z:IF LINECOUNT<15 T
HEN COUNT=LINECOUNT
390 PRINT #1;LINE;" DATA ";
400 FOR I=1 TO COUNT:DATUM=C(15*ITEM+I
):PRINT #1;DATUM;",";:TOTAL=TOTAL+DATU
M:NEXT I
410 PRINT #1;TOTAL
420 ITEM=ITEM+1:LINECOUNT=LINECOUNT-15
:IF LINECOUNT<1 THEN 450
430 LINE=R(ITEM)
440 GOTO 380
450 CLOSE #1:POKE 559,PIK
460 ? "K↰To check BUG data against pri
nted      data statements, type NEW. Th
en type:"
470 ? "ENTER ";CHR$(34);"D:BUGRETURN .
                   Type LIST after the
READY prompt."
480 ? :? "The line number of each data
 statementcoincides with the first lin
e of the"
490 ? "user program which the data sta
tement evaluates."
500 ? "Numbers within each data statem
ent     represent consecutive lines of
the     user program."
510 ? "The last number is the total."
520 ? :? "Check the last number of eac
h state-  ment against the printed ver
sion;"
530 ? "only in case of a discrepancy c
heck    each number in the data stateme
nt."
540 ? "Make note of the lines containi
ng the bugs. Then ENTER ";CHR$(34);"D:
yourprogRETURN"
550 ? "to make the corrections.":END
560 POKE 559,PIK:? "K↰":? "Your typed-
in program was not properlyLISTed to d
isk."
570 ? :? "Please LIST your program to
disk, thenRUN ";CHR$(34);"D:CHECK";CHR
$(34);" again.":CLR :END
```

●

UTILITY#3: DISK TOOL

16K DISK

by Tony Messina

Prologue

I'd like to start this issue's utility by thanking all of you who have taken the time to write and express your ideas/comments about this series. I've been forced to purchase a larger mailbox to handle the volume of mail I'm getting. I LOVE IT!! Your responses to this series really make it a pleasure for me and make everything worthwhile. Many readers have asked me, "Why are you doing this?" The most asked question is, "What is the purpose of this series?" Well . . . as I've mentioned before, you could run out and buy these utilities and spend anywhere between $10.00-$50.00. I like to think that the purpose of a magazine such as *A.N.A.L.O.G.* is to inform the reader as well as provide useful software. (Have I avoided the question?) Really though . . . the intent of any article I write, especially my utilities, is multipurpose.

1) Provide something useful
2) Explain how and why it works
3) Present ideas clearly
4) Enlighten the reader on the internal workings of the ATARI
5) Encourage improvements on the basic idea and programs

My routines may not be elegant and I rarely use any tricks. This seems to cause heartburn with some readers, as I get nasty letters — such as, "Why didn't you do X instead of what you did?" or "You could have made this a subroutine and saved some space!!" My first answer would be a very grateful "Thank you for showing me an alternate method." My second reply would be that, in the interest of clarity, I coded the program the way it was presented. I do tend to sacrifice coding efficiency for coding clarity. I'm simply following guidelines 3, 4, and 5. If the reader can understand the ideas and concepts behind the utility, he or she can certainly improve upon the concept by altering routines, adding error checking, improving output, making the program user friendly, etc. . . . An average assembly language programmer could take the whole program and convert it all to assembly language. Again . . . the concept, the ideas . . . that is what I am presenting. For those who want to improve things, please be my guest. Those who don't care will get a reasonable utility to use and will get some understanding on how the ATARI operates. I encourage any questions or comments by letter or phone. If you've improved upon my utilities, I'd appreciate your input and a source listing. Thanks for all the letters and cards. Keep those ideas rolling!!

Back to Utility #3 Part 1

I have received many letters asking about disk structures. "AHA!" (my devious mind senses another topic for a tutorial/utility.) Thus was born the DISK TOOL article.

Disk Tool History

My need for a disk utility made its appearance shortly after my disk drive arrived in March, 1981. (Yes . . . I have the old clunker ROM version.) I was plagued with disk link errors and crashed files all over the place. To put it mildly, "Boy, was I really mad!" It was then I decided to write a program that would allow me access to any sector on the disk. To make a long story short, I got a copy of the DOS 1 source listing and ATARI Tech Manual. I then locked myself in the den and proceeded to work. 50 gallons of coffee, two power outages and 5 billion phone calls to ATARI later, I emerged victorious . . . I had actually managed to READ and WRITE to a disk sector without using the File Management System (FMS) or Utility Code in DOS 1. Yaaaayy!!

When DOS 2 arrived on the scene, I converted the TOOL. I've been fiddling with it off and on for the past year and a half. Anyway, now that you know my life history (at least the last year or so), I'll move into the meat of this article, DISK STRUCTURES. One last thing . . . To keep the article as short as possible (there's really a lot to cover), I've broken it into two parts. This part will cover Disk Organization, Disk Sector Structure, Disk File Types and Disk Directory Structure. The DISK TOOL program is virtually unchanged from when I wrote it way back when. AND away we go . . .

Disk Sector Structure

The ATARI disk drive, in conjunction with the File Management System (FMS), organizes data on a diskette into blocks called sectors. There are 720 sectors (numbered from 0-719) on each diskette after it is formatted by the Disk Operating System. The sectors are laid out on what are known as tracks. There are 40 tracks per diskette, each containing 18 sectors. To clarify the last two statements, I have my patented "Formatted Diskettes are *like onions*" dog and pony show. Next time you cut an onion in half (when you make onion rings, mushrooms and onions, or whatever), lop off a hunk in the middle about 1/4 inch wide. Now turn the onion so that the big round part faces you. Each individual ring of that onion is exactly similar to a track on the diskette. Go ahead, pull off the outer ring. Now if you cut that ring into 18 equal pieces, each piece would represent a sector. The outer ring is track 0. As you move inward, the next ring is track 1 and so forth, until you reach track 39. Each track would contain 18 sectors. Track 0 contains sectors 0-17; track 1 has sectors 18-35; etc....So now you have the idea of how a diskette is organized. DISK TOOL is designed to work at the sector level. Although there are 720 sectors to each diskette, not all sectors are available to you, the user.

You Wondered Where the Sectors Went?

You've just formatted a diskette. Ahhh, the feeling of power, 720 sectors to store all of your programs. You hit the A OPTION in DOS (just to see that magic number, 719). Upon hitting RETURN, the number 707 appears. What??!! What happened? Well, it's quite simple, friends. Although there are 720 sectors, only 707 are available for your use. The other sectors are allocated for use by DOS. The Disk Directory steals 8 sectors, starting at sector 361 and running to 368. One sector (360) is allocated for the VTOC (Volume Table of Contents, pronounced "Vee-Talk"). We'll talk about the VTOC next issue. The Boot portion of FMS also occupies 3 sectors (1, 2, 3). That's what happened to your 12 missing sectors, so don't be alarmed. With that out of the way, it's time to discuss the different types of sectors. Yes...I know it sounds confusing...after all, isn't a sector a sector? The answer is yes. Each sector is capable of holding 128 Bytes of Data. The manner in which the Data is structured on a sector is dependent on a particular sector's purpose or type. I like to define the sectors as being of 4 types:

1) Data Sector: Containing program information, text files, etc.
2) Boot Sector: Containing ML Program Data
3) Directory Sector: Containing program names and associated Data
4) VTOC Sector: See next issue

Let's take a look at the differences and similarities of each.

Data Sectors

These are the most common type of sector on your disk. Technically, all sectors are Data sectors. I use this name only to distinguish its format from other types of sectors. Whenever you use the commands: SAVE "D:XXX", LIST "D:XXX" or invoke the Binary Save Option from DOS, the actual programs are written to the disk in Data Sector Format. The format is quite simple. Bytes 0-124 contain Actual Program Data. Bytes 125-127 contain Sector Identity Data or Link Data. Figure 1 illustrates this type of format.

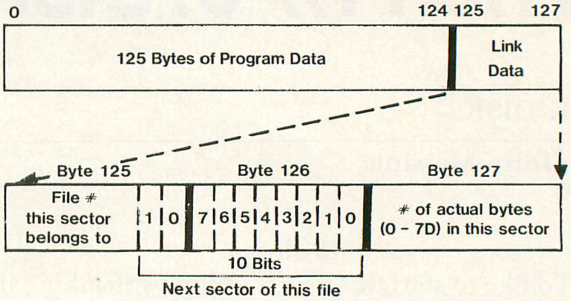

0
124 125
127
125 Bytes of Program Data
Link Data
Byte 125
Byte 126
Byte 127
File # this sector belongs to
1 0
7 6 5 4 3 2 1 0
# of actual bytes (0 - 7D) in this sector
10 Bits
Next sector of this file

FIGURE 1

Notice that the lower 2 Bits of Byte 125 and all of Byte 126 are combined to point to the next physical sector of this file. A zero (0) indicates that this is the last sector of a file. One variation in Data Sector Format occurs when the Binary Save Option is used to save an area of memory to the disk. The variation occurs with the first 6 Bytes of the first sector of the Binary File Data. Those 6 Bytes are commonly referred to as the "Binary File Header." The Header is formatted as per Figure 2.

Byte	0	1	2	3	4	5
	FF	FF	LSB Start addr	MSB Start addr	LSB End addr	MSB End addr

FIGURE 2

If, for example, you answer the Binary Save Prompt in DOS with "MLPROG,0600,065F", then the first 6 Bytes of the first sector of disk storage for this program would look like Figure 3.

Byte	0	1	2	3	4	5
	FF	FF	00	06	5F	06
	Binary file definition bytes		Start addr LSB/MSB		End addr LSB/MSB	

FIGURE 3

In our example above, only one sector would be needed to save our program, since it isn't longer than 125 Bytes. Now that we know about Data Sectors, let's move on to Directory Sectors.

Directory Sectors

There are 8 Directory Sectors, starting at sector 361 and running sequentially to sector 368. The Directory contains the names of all the programs on the diskette, along with other information about the program. Each Directory entry, program name and miscellaneous information uses 16 Bytes. There is enough room to hold 8 program names (and associated data) on 1 sector. (16 Bytes * 8 names = 128 Bytes or 1 sector.) Therefore, with 8 sectors available, we can have (8 sectors * 8 names per sector) = 64 possible file names total. Of course, the size of each file will vary, so several large files may take up the entire disk. Anyway, the Directory sectors do not have any link Data. On a Directory read, DOS starts at sector 361 and keeps reading sectors until there are no more names. Directory entries have the following format:

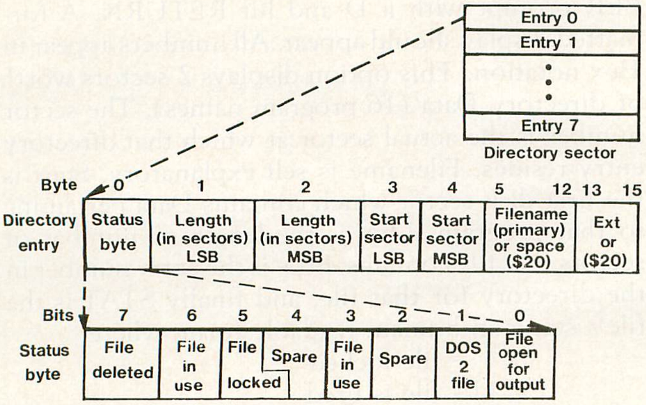


FIGURE 4

STATUS NOTE: Bits set (=1) indicate condition listed.

Bits 1, 5, 6 set.

Bits can be combined for multiple status.

Example: Bits 1, 2, 3, 4, 5 set would mean file was created by DOS 2; it is in use and locked.

Here is a quick reference to the possible status values.

$00 = file never used
$01 = file open for output
$02 = file created by DOS 2 (if Bit not set, assume DOS 1)
$20 = file locked
$40 = file in use
$80 = entry deleted

That's all there is to Directory Sectors. How does DOS use the information we have discussed so far? In simple terms, when you type in LOAD "D:XXX" or RUN "D:XXX" in BASIC, the FMS opens the Directory for input, reads in the Directory sectors starting at sector 361 and searches for a match. When it finds a Directory Entry that matches the program name you asked it for, the FMS (File Management System) extracts the starting sector from Bytes 3 and 4 of the entry and also the length from Bytes 1 and 2. FMS then positions the READ/WRITE head of the disk drive at that sector, reads in the sector, extracts the link information (to find the next sector) and checks to see if this sector actually belongs to the file you wanted. If it does, then FMS checks to see if this is the last sector to load. (Remember, the next sector to load is in the link Bytes.) FMS keeps loading until the next sector to load is 0. If, during this process, the file number of the sector just loaded does not match the one you are looking for, a File Number Mismatch Error occurs. This usually means that either the disk link information of the previous sector was incorrect, or possibly the link Data of the current sector is incorrect. We'll discuss how to fix this in the follow up article next issue.

Boot Sectors

I use the term Boot Sector Format in referring to files which start at sector 1 and run contiguously to sector X, where sector X is the ending sector. These files do not need any language cartridges or DOS and are completely self contained programs which load and execute upon powering up the computer. Do not confuse these with AUTORUN.SYS files. Remember the header Bytes for Binary files saved using the Binary Save Option of DOS? Well, Boot Sectors have a similar structure. Sector 1 of the disk contains the magic header information which is structured as per Figure 5.

Byte	0	1	2	3	4	5
	0 by tradition	# of sectors to load	LSB Load addr	MSB Load addr	LSB Init addr	MSB Init addr

Boot sector header (sector 1)

FIGURE 5

Whenever you turn on your computer, a check is made to verify if any cartridges are present. If a cartridge is present, the "Allow Disk Boot Bit" (Bit 0 of location $BFFD) is checked. If it is zero (as it would be if no cartridge were present), then the ROM Boot Routine is invoked. This routine goes out to sector 1 of the disk, reads in the Data contained there and interprets it. Byte 1 tells the system how many sectors to read. Byte 2 and 3 tell the system where to load the Date, and Byte 4 and 5 tell the system where to start executing the ML program once it is loaded. Boot sectors do not have any link Data, and consequently each boot type sector can contain 128 Bytes of program information. I said "can" because the last sector may be a short sector (contain less than 128 Bytes). The FMS for DOS 2 contains 3 boot sectors worthy of program Data.

Well, that about wraps up our discussion on disk Data structures. I realize I've clobbered you with many new concepts and material. The best way to digest this information is to use the DISK TOOL experiments which follow. First take a break, then we'll discuss DISK TOOL and do some "hands on" experimenting.

Disk Tool Structure

You may have noticed that DISK TOOL consists of 3 programs . . . an Autorun.Sys creator program, a Machine Language Loader and the Actual BASIC Code. Why 3?? Well, in order to make DISK TOOL fit into a 16K disk drive system, I decided to load the ML portion separately from the BASIC program. That really is the only reason. I didn't want any disk drive owners left out because he or she didn't have more than 16K or memory available. DISK TOOL sets itself up as follows:

1) Protect 1500 Bytes of low end memory and disable the break key (via Autorun.Sys)
2) Load the ML portion of DISK TOOL into the protected area and load DISK TOOL BASIC Program
3) Execute DISK TOOL BASIC

Since I believe it is more important (at least this issue) to know how to use this utility, I won't get into a long winded dissertation about how it works. (As if I haven't been long winded in this article.) If you study the listings, you should be able to get a fairly good understanding of what is going on. If time and space permit, I will discuss the program function at the conclusion of the next issue's article.

Warnings:

DISK TOOL will happily allow you to wipe out your directory, the VTOC, DOS Boot Sectors or any other sector on your disk, even if the file is locked!! It will ask you to verify prior to writing, but once a sector has been written, it may be too late. You don't need to be an advanced systems programmer to use DISK TOOL...only a careful programmer. It is suggested that you don't even use the write function until next issue, when we'll discuss things you can do with DISK TOOL. OK, warnings are behind us; let's move on.

Getting Things Together

The first thing to do is to get a new diskette, format it and write out DOS 2 to the diskette. Type the listings in order from program 1 to 3. Save all programs to the disk. Suggested names follow:

1) Autorun.Sys Maker....MAKAUTO.UTL
2) ML Loader............DSKTOOL.RV3
3) Disk Tool BASIC DSKTOOL.PT2

These are only suggestions. If you decide to rename the BASIC DISK TOOL portion, you must change the RUN command in the ML LOAD program so that you don't get a file not found error. Run the Autosys maker first so it can create the Autosys file. Power down, power up with the same disk and type RUN"D:DSKTOOL.RV3".

Using DSKTOOL...(finally!!)

I know everyone has DSKTOOL running. (Those of you who don't, keep trying.) The first thing you will see is the Command Menu and a "COMMAND OR SECTOR NUMBER" prompt. To examine any sector, just type in the number and hit RETURN. Only sectors 1-720 can be examined. Any number 1 or 720 will generate an error message. Sector numbers can be entered in either Decimal or Hex (if preceded by a $). Let's try it out. Put in any of your program diskettes.

Experiment 1 (look at directory sector)

Answer the prompt with 361 and hit RETURN. You will see the first sector of the directory. Compare each entry with the format of Figure 4. Once you feel comfortable with the format of the directory, move on to the next experiment.

Experiment 2
(look at formatted directory output)

Answer the COMMAND OR SECTOR NUMBER prompt with a D and hit RETURN. A formatted display should appear. All numbers appear in Hex notation. This option displays 2 sectors worth of directory Data (16 program names). The sector number is the actual sector at which that directory entry resides. Filename is self explanatory. Start is the first disk sector which contains Data pertaining to that program. LEN is the length or number of sectors that file contains. FIL# is the entry number in the directory for that file, and finally STAT is the file's status in human, readable form, where:

* = file locked
U = file is used
D = file has been deleted
1 = file created by DOS 1
2 = file created by DOS 2

To examine more directory sectors, hit + and press RETURN. The new sectors will appear. To abort the directory format, just hit RETURN and our friend "COMMAND OR SECTOR NUMBER" will appear.

Experiment 3 (trace/examine a file)

Perform Experiment 2 and abort the directory format simply by hitting RETURN. The COMMAND OR SECTOR NUMBER prompt should appear. Now find a file you want to examine from the directory listing. (Try one other than DOS.SYS or DUP. SYS.) Find the start sector number for that file under the start column. Since the start number is in Hex, type & followed by the number. You don't need to type in leading zeros. If, for example, the start number was 00BF, then type $BF, for 01CD type $1CD, etc. Then hit RETURN. The sector will appear in HEX/ATASCI format along with the sector number, next sector and file information. Sector number indicates the current sector number being displayed. Next sector points to the next sector containing Data for this file. File number is the file number to which this sector belongs. The next sector does not have to be the current sector number +1. If a diskette has had files deleted and then new files added, a particular file's sectors may be scattered all over the disk. When you are ready to look at the next sector, you can enter the number and hit RETURN. If the next sector happens to be the current sector +1 just hit RETURN or + and RETURN. If you want to look at the current sector -1 type - and RETURN. Trace your file, examining the format of the Data. Remember Figure 1. Try to look at all types of files, Binary, Save Files, ASCII files etc. and compare these with the appropriate figures. When you hit the end of a file, you'll see that the next sector pointer will equal zero.

Experiment 4 (change Bytes)

Call up sector 720 on the disk. If it is all zero's, then you can use it. If it isn't all zero's, type – and hit RETURN until a sector is displayed with all zero's. At the prompt COMMAND OR SECTOR NUMBER, type in C and hit RETURN. The screen should change to yellow and a prompt should appear. Move the cursor (ctrl up, down, left, right, arrow etc) to the 1st Hex value in the Byte 00 line. Replace the 00 values with the following:

47 4F 4F 44 20 4A 4F 42

Then hit RETURN. Make sure you overwrite each value of 00 and space between each Byte. If you have done everything correctly, you should see a "secret message". Change Byte only changes memory locations. Nothing has been written to the disk.

Experiment 4A (Change Bytes ATASCII METHOD)

Follow the procedure in EXPERIMENT 4. To change bytes, move the cursor over to the hex parameter you wish to modify. Hit the space bar to blank out the first parameter of the hex number. Now type the ATASCI letter or number that you want. Continue with the rest of the line, always remembering to precede the character you want with a space. Hit return and check your work. If the ATASCI column does not reflect your desires, try again.

Experiment 5 (writing to the disk)

As I mentioned previously, writing to the disk can be dangerous. Be careful!! Sector 720 should be safe. Why? Well, there is a bug in DOS. DOS can only handle sector numbers from 0 to 719. The disk drive, however, will only accept commands for sectors 1-720. Some software developers have taken advantage of this useful quirk. Don't write to 720 if something was there. If all was OK and you did experiment 4, push RETURN. Now type W and hit RETURN. The screen will turn red and a verify prompt will appear. Answer Y to the prompt if you are sure you want to write to the disk. When the write is complete, the screen will turn green again and we're back to the COMMAND OR SECTOR NUMBER prompt. Recall sector 720, or the sector which you wrote, just to check what was written.

What can we do right now?

One item you can fix right now and the procedure using DISK TOOL is outlined below.

PROBLEM: Two files on a disk with the same *name.*

REASON: You renamed a file to a name that already existed.

FIX: Call up the directory of the disk using the D command. Find the file you want to change the name of. Hit RETURN. Get the sector number of the directory where the file name resides, from the SEC # column. Call up this sector by typing $XXXX where XXXX is the sector number. Find the file name ATASCI. Hit C for change Byte and RETURN. Move the cursor up to the line of the file name Hex values. Using the ATASCI change procedure from EXPERIMENT 4A, change the file name and hit RETURN. Check to see that the name is correct. When everything is to your satisfaction, write out the sector to the disk.

That's all for now. We've covered a lot of ground. One suggestion is to play around, examining sectors, formats and structures. Refer to the Figures and enjoy. Next issue we will discuss the VTOC. We'll also add a File Sector Trace, Sector Allocation Map Dump, Screen Dump to Printer and Change Links Function to make DISK TOOL more useful. Till next issue. . . Keep hacking and may all your sectors be full. □

PROGRAM 1

```
10 GRAPHICS 2+16
15 ? #6;" ++++++++++++++++++"
20 ? #6;" + ANALOG 400/800 +"
25 ? #6;" +  DSKTOOL.RV3   +"
30 ? #6;" +  autorun.sys   +"
35 ? #6;" + CREATOR PROG.  +"
40 ? #6;" +   for dos ii   +"
45 ? #6;" ++++++++++++++++++"
50 ? #6;"   hit any key to":? #6;" cre
ate AUTORUN.SYS":? #6;"          file"
60 OPEN #1,4,0,"K:"
65 GET #1,A
70 CLOSE #1
75 ? #6;"   creating file"
80 OPEN #1,8,0,"D:AUTORUN.SYS"
85 PUT #1,255:REM HEADER $FF
90 PUT #1,255:REM HEADER $FF
100 PUT #1,0:REM LOAD START LSB $00
105 PUT #1,6:REM LOAD START MSB $06
110 PUT #1,74:REM LOAD END   LSB $4A
115 PUT #1,6:REM LOAD END   MSB $06
120 READ A:IF A=999 THEN GOTO 140
123 REM ** NOW PUT OUT REST OF PROG **
125 PUT #1,A
130 GOTO 120
140 CLOSE #1
160 POSITION 3,10:? #6;" FILE WRITTEN"
170 GOTO 170
1000 DATA 24,173,231,2,105,220,141,231
,2,173
1002 DATA 232,2,105,5,141,232,2,169,0,
133
1004 DATA 8,32,27,6,76,0,160,120,173,2
2
1006 DATA 2,141,60,6,173,23,2,141,61,6
1008 DATA 169,52,141,22,2,169,6,141,23
,2
1010 DATA 88,96,72,173,14,210,16,4,104
,76
1012 DATA 59,6,169,127,141,14,210,165,
16,141
1014 DATA 14,210,104,64,0,226,2
1016 DATA 227,2,0,6,224,2,225,2,0,6
1018 DATA 999
1020 REM ******************
1022 REM * END AUTORUN.SYS*
1024 REM *   LOADER PROG  *
1026 REM ******************
```

●

D: CHECK DATA

```
10 DATA 442,478,615,797,549,86,737,481
,771,206,465,558,511,194,597,7487
90 DATA 511,325,357,409,318,353,235,49
5,698,762,8,510,567,276,235,6059
1006 DATA 52,273,359,628,586,942,728,9
60,537,285,978,6328
```

●

PROGRAM 2

```
5 GRAPHICS 2+16:POKE 712,14:POKE 709,1
02:POKE 708,202
10 ? #6;"  ++++++++++++++++"
20 ? #6;"  +ANALOG 400/800+"
30 ? #6;"  +   presents   +"
40 ? #6;"  +  utility #3  +"
50 ? #6;"  +   DISK TOOL  +"
60 ? #6;"  ++++++++++++++++"
70 ? #6;"            by"
80 ? #6;"    TONY MESSINA"
90 ? #6;" reading ml program"
100 AREA=7420:REM **ML SAVE AREA **
110 POKE 711,14:READ X:IF X=999 THEN P
OKE 755,2:GOTO 125
120 POKE 711,0:POKE AREA,X:AREA=AREA+1
:GOTO 110
125 ? #6;" loading dsktool.utl":RUN "D
:DSKTOOL.PT2"
130 DATA 32,83,228,48,51,173,142,29,20
8,69
132 DATA 32,145,29,32,50,30,173,34,30,
32
134 DATA 27,30,32,187,29,162,0,160,0,1
89
136 DATA 253,3,32,255,29,32,234,29,32,
36
138 DATA 30,32,190,29,32,43,30,200,192
,8
140 DATA 240,29,232,76,25,29,140,138,2
9,173
142 DATA 122,4,72,41,3,141,143,29,104,
74
144 DATA 74,141,144,29,160,0,140,34,30
,104
146 DATA 96,142,35,30,138,56,233,7,170
,238
148 DATA 35,30,189,253,3,32,207,29,142
,141
150 DATA 29,32,164,246,174,141,29,232,
236,35
152 DATA 30,208,235,169,155,32,164,246
,174,35
154 DATA 30,224,128,176,190,32,50,30,3
2,18
156 DATA 30,32,187,29,160,0,174,35,30,
76
158 DATA 25,29,0,0,0,0,0,0,0,162
160 DATA 0,189,166,29,240,13,142,139,2
9,32
162 DATA 164,246,174,139,29,232,56,176
,238,96
164 DATA 125,66,89,84,69,35,127,127,72
,69
166 DATA 88,127,127,65,84,65,83,67,73,
155
168 DATA 0,32,190,29,169,32,32,164,246
,96
170 DATA 41,15,201,10,48,2,105,6,105,4
8
172 DATA 96,201,32,144,20,201,125,144,
18,201
174 DATA 128,144,12,201,155,144,10,201
,160,144
176 DATA 4,201,253,144,2,169,46,96,32,
36
178 DATA 30,173,254,29,32,164,246,173,
253,29
180 DATA 32,164,246,32,43,30,96,0,0,72
182 DATA 74,74,74,74,32,196,29,141,254
,29
184 DATA 104,32,196,29,141,253,29,96,1
73,34
186 DATA 30,24,105,8,141,34,30,32,255,
29
188 DATA 32,234,29,96,0,0,142,139,29,1
40
190 DATA 140,29,96,174,139,29,172,140,
29,96
192 DATA 169,62,32,164,246,169,36,32,1
64,246
194 DATA 96,104,104,133,206,104,133,20
5,160,2
196 DATA 177,205,72,200,177,205,32,149
,30,141
198 DATA 148,30,104,32,149,30,10,10,10
,10
200 DATA 13,148,30,170,24,105,8,141,14
1,29
202 DATA 200,200,200,177,205,201,32,20
8,5,200
204 DATA 177,205,208,21,72,200,177,205
,32,149
206 DATA 30,141,148,30,104,32,149,30,1
0,10
208 DATA 10,10,13,148,30,157,253,3,232
,236
210 DATA 141,29,144,213,72,76,6,29,0,5
6
212 DATA 233,48,201,10,144,2,233,7,96,
76
214 DATA 52,29,32,255,29,32,234,29,96,
32
216 DATA 83,228,48,241,169,253,133,205
,169,3
218 DATA 133,206,160,5,162,0,177,205,1
57,215
220 DATA 31,200,232,224,11,144,245,160
,0,177
222 DATA 205,141,230,31,200,177,205,14
1,229,31
224 DATA 200,177,205,141,228,31,200,17
7,205,141
226 DATA 227,31,200,177,205,141,226,31
,44,230
228 DATA 31,16,8,169,68,141,231,31,76,
24
230 DATA 31,80,37,169,85,141,231,31,16
9,32
232 DATA 44,230,31,240,5,169,42,141,23
3,31
234 DATA 169,2,44,230,31,240,8,169,50,
141
236 DATA 232,31,76,24,31,169,49,141,23
2,31
238 DATA 173,11,3,32,162,30,173,10,3,3
2
240 DATA 162,30,32,190,29,162,0,189,21
5,31
242 DATA 32,36,30,32,164,246,32,43,30,
232
244 DATA 224,11,144,239,32,187,29,173,
226,31
246 DATA 32,162,30,173,227,31,32,162,3
0,32
248 DATA 187,29,32,190,29,173,228,31,3
2,162
250 DATA 30,173,229,31,32,162,30,32,18
7,29
252 DATA 32,190,29,173,237,31,32,162,3
0,32
254 DATA 187,29,162,2,189,231,31,32,36
,30
256 DATA 32,164,246,32,43,30,169,32,15
7,231
258 DATA 31,202,16,236,238,237,31,238,
236,31
260 DATA 169,8,205,236,31,240,22,165,2
05,24
262 DATA 105,16,133,205,144,2,230,206,
169,155
264 DATA 32,164,246,32,198,31,76,182,3
0,169
266 DATA 155,32,164,246,173,10,3,24,10
5,1
268 DATA 141,10,3,144,3,238,11,3,162,0
270 DATA 142,236,31,173,235,31,208,17,
238,235
272 DATA 31,76,169,30,162,11,169,32,15
7,215
```

```
274 DATA 31,202,208,250,96,206,235,31,
76,75
276 DATA 29,0,0,0,0,0,0,0,0,0
278 DATA 0,0,0,0,0,0,32,32,32,32
280 DATA 32,0,0,0,0,0,999
```

●

D: CHECK DATA

```
5 DATA 355,844,811,79,882,872,711,341,
260,440,347,8,84,917,437,7388
132 DATA 508,322,250,556,640,222,576,7
47,397,738,929,654,336,485,693,8053
162 DATA 783,490,416,436,287,724,799,5
32,885,240,574,671,205,561,672,8275
192 DATA 748,981,703,367,562,920,675,5
46,683,81,523,325,757,771,711,9353
222 DATA 811,950,887,281,630,672,572,4
20,268,682,509,887,369,860,375,9173
252 DATA 619,459,553,705,854,686,920,5
62,249,682,896,597,161,482,733,9158
```

●

PROGRAM 3

```
10 REM ********************************
15 REM *  DISK TOOL BASIC PROGRAM    *
20 REM *   BY TONY MESSINA (C)1982   *
25 REM ********************************
30 POKE 82,0:REM **LFT MAR TO 0 ***

50 DUNIT=769:DCOMM=770:DAUX1=778:DAUX2
=779:GSEC=82:PSEC=87:DBYHI=777:DBYLO=7
76
70 DIM A$(29),AN$(1)
110 TRAP 710
130 REM **** SET UP DISK VECTOR TABLE
****
150 POKE DUNIT,1:REM ** DRIVE 1

170 POKE DCOMM,GSEC:REM ** FOR READ

190 POKE DAUX1,1:REM ** SECTOR 1 **

210 POKE DAUX2,0:REM ** NULL **

230 POKE 772,253:REM ** LOW BUF ADR ($
FD)**
250 POKE 773,3:REM **HI BUF ADR ($03)
**
270 POKE DBYLO,127:REM ** GET 128 BYTE
S (1 SECTOR) **
290 POKE DBYHI,0:REM ** NO HI **

300 GOTO 2000
330 REM *** OK..CLR SCREEN AND ASK FOR
 SECTOR ***
370 ? "▸COMMAND OR SECTOR NUMBER";:INP
UT A$
371 GRAPHICS 0:REM **CLEAR SCREEN**
372 POKE 712,10:POKE 709,0:POKE 710,21
4
380 IF A$="H" THEN GOTO 2000
390 IF A$="+" OR A$="" THEN SECNUM=SEC
NUM+1:GOSUB 630:GOTO 490
410 IF A$(1,1)="-" THEN SECNUM=SECNUM-
1:GOSUB 670:GOTO 490
430 IF A$="W" THEN GOSUB 750:GOTO 370
450 IF A$="C" THEN GOSUB 1030:GOTO 370
452 IF A$="D" THEN GOTO 5000
460 IF A$(1,1)="$" THEN GOSUB 4000
470 SECNUM=VAL(A$):IF SECNUM<1 OR SECN
UM>720 THEN ? "      INVALID SECTOR  RAN
GE IS (1-720)":GOTO 370
490 SECHI=INT(SECNUM/256):SECLOW=INT(S
ECNUM-(SECHI*256))
510 POKE DAUX1,SECLOW:POKE DAUX2,SECHI
530 X=USR(7420):REM ** GO DO IT **
550 IF PEEK(7562) THEN ? "CAN'T READ S
ECTOR";SECNUM:POKE 7562,0:GOTO 370
560 IF SECNUM<369 AND SECNUM>360 THEN
? "▸  DIRECTORY SECTOR ";SECNUM:GOTO 3
70
570 ? "SECTOR ==>";SECNUM;"▸▸NEXT SEC=
=>";(PEEK(7567)*256)+PEEK(1147)
590 ? "FILE#==>";PEEK(7568)
610 GOTO 370
630 IF SECNUM>720 THEN SECNUM=1
650 RETURN
670 IF SECNUM<1 THEN SECNUM=720
690 RETURN
710 ? "▸ILLEGAL INPUT!!":POKE 709,0:PO
KE 710,214:TRAP 710:GOTO 370
730 REM *** WRITE SECTOR TO DISK ***

750 POKE 710,64:POKE 709,10
755 X=USR(7430)
760 GOSUB 3000
770 ? "KCURRENT SECTOR IS ==>";SECNUM
790 ? "KSURE ABOUT WRITE(Y/N)";:INPUT
AN$
810 IF AN$<>"Y" AND AN$<>"N" THEN ? "▸
DANGEROUS INPUT!!":GOTO 950
830 IF AN$="N" THEN ? "WRITE ABORTEDKK
":GOTO 950
850 POKE DCOMM,PSEC:POKE 7566,1
870 X=USR(7420)
890 ? "SECTOR ";SECNUM;" WRITTEN"
910 IF PEEK(7562) THEN ? "DISK WRITE E
RROR!!!K":POKE 7562,0
930 POKE DCOMM,GSEC:POKE 7566,0
950 POKE 709,0:POKE 710,214
970 AN$=""
990 RETURN
1010 REM *** CHANGE BYTES ROUTINE ***

1030 POKE 710,26
1031 X=USR(7430)
1032 GOSUB 3000
1040 ? "KMOVE CURSOR TO BYTES,CHANGE,H
IT RETURN"
1050 INPUT A$
1070 IF A$="" OR LEN(A$)<26 THEN ? "IL
LEGAL INPUT!!PRESS RETURN";:GOTO 1170
1090 POKE 710,26
1110 LINBUF=ADR(A$)
1130 X=USR(7741,LINBUF)
1150 ? "DATA CHANGED--HIT RETURN TO CO
NTINUEKK";
1170 INPUT A$
1190 POKE 710,214
1210 RETURN
2000 GRAPHICS 1:POKE 710,214:? #6;"   
                            ":? #6;" disk tool co
mmands"
2005 ? #6;"                    "
2010 ? #6;"+ READ NEXT SECTOR":? #6;"-
 READ PREVIOUS SEC":? #6;"WRITE SECTOR
 TO DSK"
2020 ? #6;"CHANGE SECTOR BYTES":? #6;"
HELP(SHOW COMMANDS)"
2030 ? #6;"DIRECTORY LIST"
2090 ? "K":GOTO 370
3000 POSITION 0,17
3010 ? "↑↑↑↑↑"
3030 POSITION 0,17:RETURN
4000 N=0
4003 FOR I=2 TO LEN(A$)
4005 IF A$(I,I)<"0" THEN GOTO 4100
4010 IF A$(I,I)<="9" THEN N=N*16+VAL(A
$(I,I)):GOTO 4050
4015 IF A$(I,I)<"A" OR A$(I,I)>"F" THE
N 4100
4020 N=N*16+ASC(A$(I,I))-ASC("A")+10
4050 NEXT I
4060 A$=STR$(N):RETURN
4100 ? "INVALID HEX PARAMETER":POP :GO
TO 370
5000 POKE DAUX2,1:POKE DAUX1,105:REM *
* SET SECTOR 361 FOR READ
5002 SECNUM=361
5010 ? "KSEC# FILENAME/EXT START LENGT
H FIL# STAT"
5020 X=USR(7849)
5022 SECNUM=SECNUM+1
5030 ? :? "      HIT RETURN TO STOP, + T
O CONT.KK";
5040 INPUT AN$
5050 IF AN$="+" THEN 5010
5060 AN$="":POKE 8173,0
5065 ? "▸COMMAND OR SECTOR NUMBER";
```

```
5067 INPUT A$
5068 IF A$="" THEN GOTO 5075
5070 IF A$(1,1)="W" OR A$(1,1)="C" THE
N ? "}IMPROPER SCREEN CONDITION<<":GOT
O 5065
5075 GOTO 380
```

●

D: CHECK DATA

```
10 DATA 771,708,417,788,809,486,463,51
0,770,963,175,386,429,885,494,9054
270 DATA 292,555,940,727,323,866,535,5
85,980,345,406,510,582,932,488,9066
490 DATA 738,363,112,715,464,302,301,5
05,398,814,240,708,73,433,175,6341
755 DATA 106,851,491,382,314,238,27,10
7,209,674,99,504,311,717,670,5700
1030 DATA 25,63,929,168,817,643,43,460
,936,197,789,991,786,239,548,7634
2010 DATA 261,286,788,331,412,791,712,
173,899,963,659,281,948,495,45,8044
4100 DATA 18,958,305,363,164,952,506,7
96,77,908,243,831,364,338,788,7611
```

●

Next issue (A.N.A.L.O.G. Computing No. 9) we will be presenting Part 2 of "Disk Tool," the third in a series of utility programs by Tony Messina.

AUDIO IN YOUR PROGRAMS

by Marc Rossen

For those of you with program recorders, this article will teach you how to access one of its hidden features, playing audio (e.g., music, your voice, etc.) through your TV speaker via the 410 program recorder. You may have seen this feature, if you've seen ATARI's Invitation To Programming series, which uses audio to help explain the lessons.

Using this feature is very simple. Just put an audio cassette in your program recorder and type in the following program:

```
10 POKE 54018,52
20 FOR A=1 TO 2000:NEXT A
30 POKE 54018,60
```

Now press the play button on your program recorder, type RUN and press the RETURN key.

You should be hearing audio playing through your TV speaker. If you don't hear anything, turn up the volume on your TV.

What you are actually doing is turning the motor on and off in your program recorder. Line 10 instructs the computer to turn on the program recorder. Line 20 is a delay to keep the motor on for a specified period of time. To change the length of time the motor is on, simply change the value of the FOR/NEXT loop in line 20. Line 30 turns off the motor in the program recorder. It is important to turn off the motor after turning it on, so that it can be turned on again later in the program.

In order to use this feature in your programs, first record the program onto a cassette and then record the audio portion of the program on the tape right after the digital part (your program).

You will need to use a regular cassette recorder to record the audio for your programs because the ATARI 410 program recorder will not record audio. □

BUDGET PROGRAMS REVIEW

Review by The Program Doctors

"Budget programs? How boring!!!" said the Program Doctors, when approached by our two illustrious editors to write this article. (The surgeon general was right — L & M can be hazardous to your health.) But we do use Theo (our computer's name, remember?) for more than just playing games, as most of you probably do, and with this in mind we decided it was important that you be made aware of the different types of budget programs available, to make sure you buy the one best suited to your needs. (Also to get your budgeting chores done quickly and easily so you have more time to play games!!!)

The programs we will discuss are **Personal Financial Management System** from ATARI, **Personal Finance for the ATARI** (new version to be released as **A Financial Wizard**) from Computari, and **Budgetmaster** from Sunrise Software. **Money Manager** from Acorn Software, while not really a budget program, does deal with your checkbook in an outstanding manner and will be outlined for you.

All three programs begin with the basic premise of setting up a budget, helping you follow it, and giving you an idea of where you are spending your hard-earned dollars. Let's first look at what these programs have in common.

To begin with, you must set up the categories that best apply to your spending habits. Then you enter your best estimate of what you will spend for each month. As you make entries of what you actually spent (your checks), the amounts are compared and analyzed for you, and you can chart a single category for many months, or many categories for one month. With the exception of Budgetmaster, you can also have this information in the form of bar graphs. All three programs offer a check reconciliation feature. Sounds simple enough, doesn't it? But the way each of these programs handles the above functions, the additional individual features, and the ease of use vary greatly.

Personal Finance for the ATARI from Computari is by far the best of these programs (why save the best for last?) and will be the standard of comparison for the others. There are 26 expense categories available that are easily adapted to your personal requirements; 21 are regular expense accounts, 1 is reserved for salary and 4 are usage categories for record keeping, such as gas and electric usage. You then input your budgeted amounts. After you get January done, if your expenses run much the same month to month, you just tell the computer to copy January for February, March, April, etc. If you have certain expenses that are different from month to month, or expenses that occur only once in a while, you can easily adjust only those without having to re-enter everything.

The check entry mode is very simple to use. After asking you what month you are entering, the program prompts you to enter the check information, including whether or not it is tax deductible. When tax time rears its ugly head, you will **really** appreciate this function and the Check Search mode which will search by Name, Category, Check # or tax deductible checks.

The way Personal Finance displays and handles your tabulations is excellent. You can chart your actual expenses vs. your budget by month, by category or Year to Date. This information is obtained by the computer from your check input. Tabulations by month give you a list of all categories, how much you spent, how much you budgeted, the dollar amount plus or minus your budget, and the percentage of your total income you are spending on each category. The tabulation by category also gives you actual expense vs. budget, the difference, and the average amount you are spending. This information will be very helpful to you in adjusting your budget if need be. (Come on, you know you'll have to adjust it!!) Besides the charts, you can also look at your expenses vs. budget in bar graph form, again by month or by category. There it is in black & white (and blue and gold); the amount you budgeted vs. the amount you spent. A very depressing picture indeed, but an excellent feature of this program.

Everything about this program is excellent, but where it **really** outshines the rest is in the Check Reconciliation. In effect, it gives you your bank statement on the screen, a complete list by month of all your checks and deposits. All you do is run down the column and type an asterisk next to everything that has been cleared. You can clear about 40 checks in 2 minutes. Then sit back, and the computer will balance your checkbook, and report how many checks and deposits are still outstanding, and whether or not you balance.

The version of **Personal Finance** that was reviewed is version 1.3. In speaking with the author, Bill McLachlan, we have been told that a version 1.5 is coming out. (As mentioned earlier, this program will also be re-named **A Financial Wizard**.) This newest version will be enhanced in a few ways. There will be a check writer option. You enter your checks as if they have already been written — the program will perform all of the previously mentioned functions and, if you have a printer, will print out your checks. You just sign and mail. Bank compatible checks will be available from Computari; ordering information will be in the package. There will also be an audit feature. As the program stands now, if for some reason, you absolutely, positively cannot balance your checkbook, you can easily adjust your balance. In version 1.5 the only way to adjust your balance will be to run an audit. The computer will run through all your checks and give you totals for each month of your checks and deposits that you can check against your bank statements. At least you'll know it's not a math error. As stated in the instructions, "Computers don't make missteaks." While this may seem like a lot of extra work, it actually is an excellent safeguard for those of you who tend to adjust your checkbook balance too readily. After all, even though you may not want to admit it, the bank is usually ıight. However, if after the audit you still cannot reconcile, you will have the opportunity to adjust your balance. Check Entry has also been enhanced. You will have the option to scan all your checks in the month you are working on without leaving the Check Entry mode. Check Search has been expanded so you can pinpoint a check within six parameters at once.

By the time this article is published, **A Financial Wizard** should be on the market, and we strongly recommend this program. User compatibility is excellent, and is set up with most of the instructions on the screen, so you are not constantly referring to the instruction manual. However, as in all of these programs, it is IMPERATIVE that you thoroughly read the manual to get the most from the program.

ATARI's **Personal Financial Management System** is a program that many people might be tempted to buy because of the superb packaging job. This program consists of 3 diskettes and a 3-ring binder instruction manual written with ATARI's typical step-by-step thoroughness. But, *caveat emptor*. While this program has most of the features of Computari's program, and even a few nice additional ones, this program is very awkward to use.

Check entry is slow. Rather than having the categories displayed in front of you as you enter checks, you must input the category name. If you had to abbreviate a category name, and don't remember it exactly, the program will ask you if it should create a new category. This creating during entry is time consuming and frustrating, especially if it's just a matter of not remembering how you coded in the names. If a check is tax deductible, you must enter it in the tax deductible category, as opposed to just answering "yes" or "no." There is a provision for comments in the check entry mode. One nice addition to this program is the Credit Card categories. When you use your credit cards, you enter the transaction and the corresponding budget category. When you write a check for a credit card payment, you enter it in the transfer category. That way you don't have a double calculation for the same expense. This is a good function because it gives you a more accurate accounting of where your dollars are spent. There is also an automatic deductions option and savings account record keeping ability.

When it comes time to analyze your budget, this program also offers you charts and bar graphs. You have the options to look at expense vs. budget in the same way as the other programs — by month, by category, etc. If you choose to look over one month's expenses, you get a partial listing and must hit return for another screenful. At the end you get totals, but unless you have a printer, you really can't study your complete budget all at one time.

The checkbook balancer is also **very** awkward to use. You must enter the check you want, wait for it to be found and displayed, then clear it. You must choose different menu options for clearing checks, deposits or automatic deductions.

ATARI offers one more feature that is different from the others — the budget forecaster. This will give you a projection of your anticipated expenditures over a selected period of time. The more data you have entered, the more accurate your projections will be. The forecasts are presented in one of three ways: by averaging your spending patterns, by es̃tablishing a trend of your habits, or seasonally. Again, you can get charts or bar graphs. While some may feel this is a necessary addition to the program, this reviewer thinks it is superfluous. After all, this is depressing enough without having to be told how much worse it's going to get.

Budgetmaster from Sunrise Software is really two separate programs, **Budgetmaster** and **Checkbalancer**. The cassette versions may be purchased separately; the disk version gives you both programs on one disk. This program is more than adequate for those of you who simply want to set up a budget and see where your money is going. There are 26 categories which are easy to personalize, and as you enter your budget amounts for each category, you can input an amount for all 12 months, or you can change or delete the amount for one or more months. For example, if your car loan is the same every month — say, $200.00 — you just budget 200 and when prompted for the month, type 13. But if you only have insurance payments in February and August, then when asked for the months you type in 2 and 8. This makes setting up your budget for an entire year quite a simple task. It is very similar to the way Computari's **Personal Finance** does it.

To input your expenses and reconcile your checkbook, you need to load **Checkbalancer**. Again your categories are displayed and you input the check number, amount, category and whether the check was cleared. There is no place to enter who a check was written to. It is recommended that check entry be done at the time you receive your bank statement, as you can enter and reconcile at the same time. However, because of this you really cannot use this program to keep your checkbook current, or possibly even eliminate keeping it by hand at all, as you can with Computari's program. If you do not purchase **Checkbalancer**, your expense entry is rather different. You must manually add up all expenses in a category for one month and make a single entry of that amount.

When you are finished, you select Save Data on the menu, and the information is written to the **Budgetmaster** files. While **Budgetmaster** does not give you bar graphs, it does analyze your budget in chart form by either month or category.

You must remember to use the Save Data option after any entering is done. It is not automatic (as with the other programs) and leaves a lot of room for error. The reason for this is that you must indicate whether you are saving to cassette or disk. For those of you with only a cassette these drawbacks will be easily overlooked because this is something you can use very well with your system.

As said earlier, **Money Manager** from Acorn Software is not a budget program, nor does it claim to be. It is a checkbook manager and this it does very well. You set up categories and input your check data. You have full search capabilities, by entry, by month, by category, etc., but instead of a chart or graph, you get a listing of all the requested data, with totals. When you define your categories, the program assigns them each a number. Entering data is simple, however, you will need a hard copy of your categories and their assigned numbers, since the program does not show them to you on the screen as you are entering your checks. **Money Manager** allows multiple entries of the same check number and has provisions for tax deductible checks.

Check reconciliation is quick and simple. The program searches through your monthly files, displays outstanding checks one at a time, and prompts you to clear or not to clear. It then informs you of what your balance should be. This is an excellent program if you do not want to keep track of your budget in a formal way as in the other programs.

Because of the capacity of the ATARI disk, some choices had to be made — and Computari, Sunrise and ATARI all made different ones. While the ATARI on the surface may seem to offer more functions and a more complete package, the fact that you have to constantly change disks to accommodate everything is very annoying. You insert a disk to load the check entry program, change disks to enter data, change disks for months, change disks for budget analysis and forecasting, re-insert data disks. **Personal Finance** has one disk that does everything, and **Budgetmaster** has two programs on one disk. While it would be nice to have the credit card files included, you can do something similar, since **Personal Finance** will accept duplicate check numbers with different categories (and in Version 1.5 there is a way to enter credit card transactions without subtracting the amounts from your balance), and **Budgetmaster** allows you to break down the dollar amount of one check into several categories. This is a small price to pay for having all your information on one disk.

Graphics, while really not a factor in the quality of programs of this type, do make your budgeting chores a little more pleasant. Again **Personal Finance** comes out on top. **Budgetmaster** and **ATARI** use graphics mode 0, with no colors but blue. **Personal Finance** uses different background colors, depending on what option you are working with, and nice touches here and there to jazz up the program. **Money Manager** also makes excellent use of ATARI's graphic capabilities. It is this little extra effort that shows you a programmer is thinking of you, the user, when he designs a program.

So, dear readers, there you have it. In making a selection of which program to use, everyone will have different requirements and wants. We hope we have made your decision a little easier. Now the choice is yours (and we can go play some games!). □

(**NOTE:** Some little bugs have been discovered in the ATARI **Personal Financial Management System**, and we have learned that this program will be taken off the market temporarily for corrections.)

FINANCIAL MANAGEMENT COMPARISON CHART

	Money Manager	Personal Finance	Budget Master	ATARI PFMS
Price	**$39.95**	**$49.95**	**$34.95**	**NLA**
System Requirements	32K Disk	24K Disk	32K Disk/Cass	32K Disk
Printer Optional	Yes	Yes	Yes	Yes
Budget Entry	No	Yes	Yes	Yes
Budget Analysis	No	Yes	Yes	Yes
Check Entry	Yes	Yes	Yes	Yes
Check Search	Yes	Yes	No	Yes
Check Reconciliation	Yes	Yes	Yes	Yes
Bar Graphs	No	Yes	No	Yes
Number of Categories	98	26 (add'l 36 sub-categ. Vers. 1.5)	26	128

STUNTMAN!

16K CASSETTE 24K DISK

by Steven Pogatch

Your stunt man has been hired to climb to the top of every building he can find. This is not as easy as it may seem, though, because the tenants of the buildings will do anything to get you off it. There are six (6) levels to each building, each progressing in difficulty.

In the first section, windows constantly close to keep you from getting past them. Next, men stick their heads out of the windows, trying to get in your way. After that, flower pots fall from the window ledges, closing all windows in their way. After passing this section, a crazy bird drops girders on you. Be careful here, they can be deadly if they hit you on the head. Once you get past the bird, you have to avoid King Kong, waiting for you on his part of the building. He is very angry and is throwing down anything he can find on top of you. Last (but not easiest), girders (3 lanes wide) come crashing down from the building. Look out!

If you are lucky enough to get through all of this, there will be a brief intermission telling you to go on to the next building. On the top left corner of the screen are three numbers. The first one represents the section, the second represents the building number, and the third represents the number of men you have left. If you manage to score 10,000, 30,000 or 50,000 points, you will be rewarded a free stunt man. The score is displayed in the lower left hand corner. You can move left, right and up with the joystick. For every movement you make, you are rewarded 50 points. You start out with 6 stunt men. Good luck climbing — you'll need it! □

Here is the documentation of the program:

Line #	Description
1 - 30	Initialization
40 - 1000	Movement of a player, activate obstacle(s)
1000 - 2000	Death (fall) of stunt man
2000 - 3000	Section 1 (windows & men)
3000 - 4000	Section 2 (flower pots)
4000 - 5000	Section 3 (bird)
5000 - 6000	Section 4 (King Kong)
6000 - 10000	Section 5 (girders)
10000 - 11000	Bonus stunt man
11000 - 32000	Go on to next building (intermission)
32000 - 32500	Redefines character set
32500 - 32700	Title
32700 - 32750	End of game

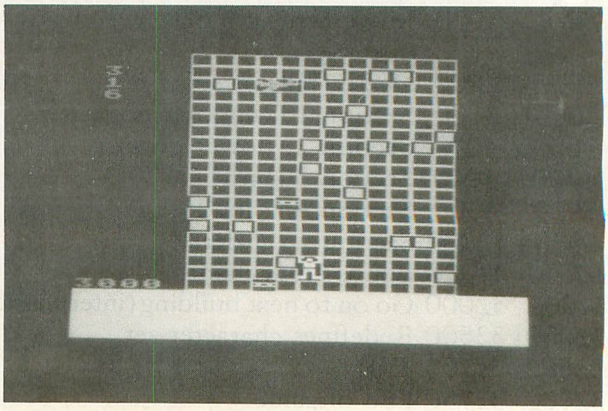
```
1 GOSUB 32000:CLR
2 GOSUB 32500:SH=6:B=1
5 GRAPHICS 1:POKE 756,PEEK(106)+1
10 SETCOLOR 2,0,0:POKE 710,94:POKE 711
,45:FOR A=0 TO 19:POSITION 5,A:? #6;"e
eeeeeeeeeee":NEXT A
20 FOR A=5 TO 16:F=RND(0)*19:IF F>1 TH
EN POSITION A,F:? #6;"f":NEXT A
30 X=10:Y=18:OX=X:OY=Y
40 POSITION OX,OY:? #6;"e":POSITION OX
,OY+1:? #6;"e"
41 LOCATE X,Y,Z:IF Z=102 OR Z=225 OR Z
=66 THEN GOSUB 1000
42 POSITION X,Y:? #6;"G":POSITION X,Y+
1:? #6;"H"
43 IF SC=10000 OR SC=50000 OR SC=10000
0 THEN SH=SH+1:GOSUB 10000
44 IF Y=0 THEN L=L+1:GOTO 5
45 OX=X:OY=Y
46 SOUND 0,Y+20,3,15:FOR A=1 TO 15:NEX
T A:SOUND 0,0,0,0
47 G=RND(0)*4:IF G>3.7 THEN FOR A=5 TO
 16:POSITION A,RND(0)*18:? #6;"f":NEXT
 A
48 ON L GOSUB 2000,3000,4000,5000,6000
:IF L=6 THEN L=0:B=B+1:GOSUB 11000
49 POSITION 0,19:? #6;SC:POSITION 1,1:
? #6;L:POSITION 1,2:? #6;B:POSITION 1,
3:? #6;SH
50 IF STICK(0)=14 AND Y>0 THEN Y=Y-1:S
C=SC+50:GOTO 40
60 IF STICK(0)=11 AND X>5 THEN X=X-1:S
C=SC+50:GOTO 40
70 IF STICK(0)=7 AND X<16 THEN X=X+1:S
C=SC+50:GOTO 40
100 GOTO 42
1000 SOUND 0,40,6,10:FOR A=1 TO 25:NEX
T A:SOUND 0,0,0,0:SOUND 1,0,0,0
1010 FOR A=Y TO 18:POSITION X,A:? #6;"
G":POSITION X,A+1:? #6;"H":POSITION X,
A-1:? #6;"e":POSITION X,A:? #6;"e"
1020 SOUND 0,A+20,10,10:NEXT A:SOUND 0
,0,0,0:SH=SH-1
1030 IF SH<0 OR SH=0 THEN GOTO 32700
1040 GOTO 10
2000 REM DOCTORS
2010 C=C+1:IF C=11 THEN C=1:DC=RND(0)*
10
2020 IF DC>6 THEN POSITION C+5,RND(0)*
18:? #6;"a"
2040 RETURN
3000 REM POTS
3010 D=D+1:IF D=11 THEN D=1:DC=RND(0)*
10
3020 IF DC>7 THEN FOR A=2 TO Y:POSITIO
N D+5,A:? #6;"B":POSITION D+5,A-1:? #6
;"f":NEXT A:POSITION D+5,Y:? #6;"e"
3030 RETURN
4000 REM BIRD
4010 BD=BD+1:IF BD=11 THEN BD=1
4020 POSITION BD+5,2:? #6;"C":POSITION
 BD+6,2:? #6;"D":POSITION BD+4,2:? #6;
"e":BDD=RND(0)*10
4022 IF BD=1 THEN POSITION 15,2:? #6;"
ee"
4025 IF BDD>3 THEN 4030
4027 RETURN
4030 FOR BDDA=4 TO 19:POSITION BD+5,BD
DA:? #6;"em":POSITION BD+5,BDDA-1:? #6
;"ee"
4040 LOCATE X,Y,Z:IF Z=109 THEN GOSUB
1000
4050 SOUND 0,BDDA+100,10,8:NEXT BDDA:S
OUND 0,0,0,0
```

```
4060 RETURN
5000 REM KONG
5010 KN=KN+1:IF KN=12 THEN KL=KL+1:KN=
1
5015 IF KN=1 THEN KN=KN+1:IF KL=12 THE
N KL=1
5020 POSITION KN+5,KL+1:? #6;"█":POSIT
ION KN+5,KL+2:? #6;"K":POSITION KN-1+5
,KL+1:? #6;"e":POSITION KN-1+5,KL+2:?
#6;"e"
5025 POSITION 16,KL+1:? #6;"e":POSITIO
N 16,KL+2:? #6;"e"
5026 REM FOR A=1 TO 12POSITION 15,A? #
6;"e"NEXT A
5027 LOCATE X,Y,Z:IF Z=235 THEN GOTO 1
000
5030 QQQ=RND(0)*10+1
5040 IF QQQ<6 THEN RETURN
5045 IF QQQ>6 AND QQQ<7 THEN 5050
5046 IF QQQ>7 AND QQQ<8 THEN 5200
5047 IF QQQ>8 AND QQQ<9 THEN 5300
5048 IF QQQ>9 AND QQQ<10 THEN 5400
5050 FOR A=KL+4 TO 17
5060 POSITION KN+5,A:? #6;"B":POSITION
 KN+5,A-1:? #6;"e":LOCATE X,Y,ZZ:IF ZZ
=66 THEN 1000
5070 SOUND 0,A+200,10,8:NEXT A:SOUND 0
,0,0,0:POSITION KN+5,17:? #6;"e"
5080 RETURN
5200 FOR A=KL+4 TO 17
5210 POSITION KN+5,A:? #6;"I":POSITION
 KN+5,A-1:? #6;"e":LOCATE X,Y,ZZ:IF ZZ
=73 THEN 1000
5220 SOUND 0,A+200,10,8:NEXT A:SOUND 0
,0,0,0:POSITION KN+5,17:? #6;"e"
5230 RETURN
5300 FOR A=KL+4 TO 17
5310 POSITION KN+5,A:? #6;"M":POSITION
 KN+5,A-1:? #6;"e":LOCATE X,Y,ZZ:IF ZZ
=77 THEN 1000
5320 SOUND 0,A+200,10,8:NEXT A:SOUND 0
,0,0,0:POSITION KN+5,17:? #6;"e"
5330 RETURN
5400 FOR A=KL+4 TO 17
5410 POSITION KN+5,A:? #6;"A":POSITION
 KN+5,A-1:? #6;"e":LOCATE X,Y,ZZ:IF ZZ
=65 THEN 1000
5420 SOUND 0,A+200,10,8:NEXT A:SOUND 0
,0,0,0:POSITION KN+5,17:? #6;"e"
5430 RETURN
5500 RETURN
6000 REM GIRDERS ARE MEAN
6005 GG=RND(0)*10+1:IF PL=1 THEN PL=19
6010 IF GG<4 THEN RETURN
6020 TTT=RND(0)*12+1
6030 IF TTT<5 THEN RETURN
6040 FOR A=2 TO 19:POSITION TTT,A:? #6
;"MMM":POSITION TTT,A-1:? #6;"eee":SOU
ND 0,A,TTT,12:SOUND 1,TTT,A,12
6045 LOCATE X,Y,ZZ:IF ZZ=77 THEN GOTO
1000
6047 NEXT A
6050 SOUND 0,0,0,0:SOUND 1,0,0,0
6060 POSITION TTT,19:? #6;"eee"
6070 RETURN
10000 RESTORE 10500
10010 READ SO:IF SO=-1 THEN SOUND 0,0,
0,0:RETURN
10020 SOUND 0,SO,10,14:FOR A=1 TO 2:NE
XT A:GOTO 10010
10500 DATA 243,4,162,4,121,6,96,2,102,
4,243,4,162,4,121,6,81,2,60,8,-1
11000 GRAPHICS 0:POKE 752,1:POSITION 1
,1:? "        GO ON TO BUILDING ";B:FOR
A=1 TO 3:FOR Q=1 TO 50:SOUND 0,Q,10,8
11005 NEXT Q:NEXT A:SOUND 0,0,0,0
11010 FOR A=0 TO SC STEP 150:SOUND 0,1
9,92,8:FOR Q=1 TO 20:NEXT Q:SOUND 0,0,
0,0:POSITION 10,5:? "SCORE:";A:NEXT A:
GOTO 5
31999 END
32000 POKE 106,PEEK(106)-5:GRAPHICS 0:
START=(PEEK(106)+1)*256:POKE 756,START
/256:POKE 752,1
32010 DIM XFR$(38):XFR$="h)♥.K.M)♦.N%j
┴i├.L ♥1M.KHP♥fL fN%NIdP.♦"
32020 Z=USR(ADR(XFR$)):RESTORE 32100
32030 READ X:IF X=-1 THEN RESTORE :RET
URN
32040 FOR Y=0 TO 7:READ Z:POKE X+Y+STA
RT,Z:NEXT Y:GOTO 32030
32100 DATA 264,60,126,219,255,231,189,
195,126
32101 DATA 272,106,50,36,255,126,126,1
26,126
32102 DATA 280,30,207,255,255,127,15,3
0,60
32103 DATA 288,30,26,255,254,224,0,0,0
32104 DATA 296,255,129,129,129,129,129
,129,255
32105 DATA 304,0,126,126,126,126,126,1
26,0
32106 DATA 312,195,153,153,231,60,60,6
0,0
32107 DATA 320,60,60,36,36,36,231,231,
0
32108 DATA 328,60,36,60,8,24,16,24,8
32109 DATA 336,60,90,126,129,165,129,1
26,60
32110 DATA 344,60,102,165,165,165,219,
60,231
32111 DATA 352,126,102,102,126,8,8,40,
56
32112 DATA 360,0,0,255,102,255,0,0,0
32113 DATA -1
32500 GRAPHICS 17
32510 FOR A=1 TO 22:POSITION 0,A:? #6;
"0*0*0*0*0*0*0*0*0*0*":SOUND 0,A+25,10
,12:NEXT A:POSITION 4,10
32511 SOUND 0,0,0,0
32512 ? #6;"..stunt.man.."
32520 POSITION 2,13:? #6;"by STEVEN PO
GATCH":POSITION 5,14:? #6;"PRESS START
"
32525 POSITION 7,18:? #6;"(c)1982"
32530 IF PEEK(53279)=6 THEN RETURN
32540 GOTO 32530
32700 GRAPHICS 18:POSITION 1,2:? #6;"G
AME  OVER":POSITION 1,5:? #6;"SCORE=";
SC
32710 POSITION 1,6:? #6;"PRESS START "
32715 IF PEEK(53279)<>6 THEN 32715
32755 CLR :GOTO 2
```

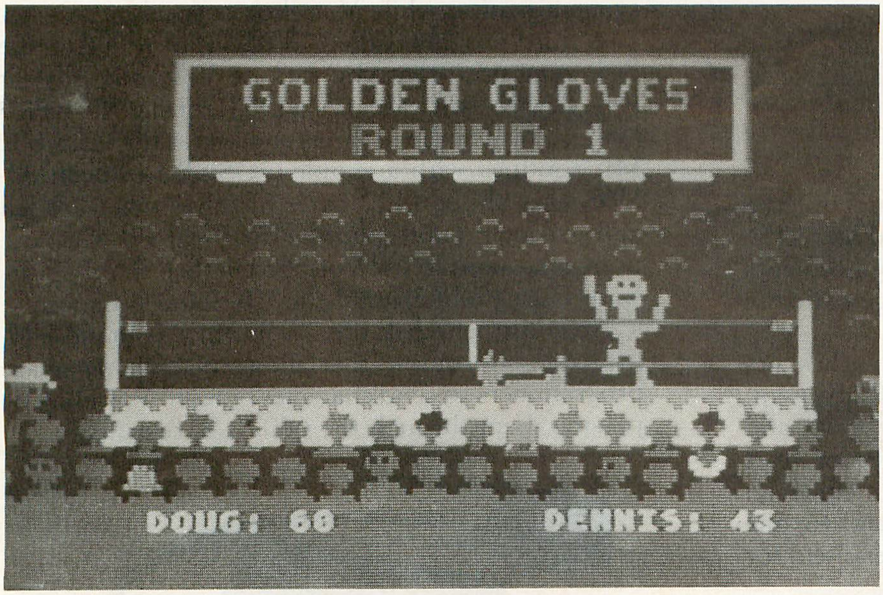
●

D: CHECK DATA

```
1 DATA 405,862,266,737,109,88,847,358,
465,380,757,766,893,250,529,7712
49 DATA 651,821,816,783,492,194,283,35
5,459,802,275,731,243,641,81,7627
3010 DATA 814,802,639,5,863,397,285,59
,644,289,767,315,649,45,903,7476
5015 DATA 475,635,35,101,239,438,357,1
96,575,12,498,586,954,593,656,6350
5200 DATA 694,516,469,796,844,154,541,
966,579,957,600,649,802,490,636,9693
6010 DATA 517,696,775,661,718,783,34,1
66,974,422,267,498,329,809,81,7730
11010 DATA 684,280,284,314,875,528,960
,108,861,386,45,170,363,354,269,6481
32108 DATA 209,607,818,288,887,619,312
,711,269,836,11,98,343,224,289,6521
32710 DATA 906,306,478,1690
```



ARTWORX SCORES ANOTHER TECHNICAL KNOCKOUT.



Scene from GOLDEN GLOVES

HODGE PODGE: by Marsha Meredith (Atari and Apple)
NOW AVAILABLE FOR ATARI!!! This captivating program is a marvelous learning device for children from 18 months to 6 years. HODGE PODGE consists of many cartoons, animation and songs which appear when any key on the computer is depressed. A must for any family containing young children.
PRICE . . . . **$19.95** diskette

BETA FIGHTER: by Douglas McFarland (Atari, 16K)
See who will be the ace gunner in this action game set on a spectacular Martian landscape. BETA FIGHTER can be played with one or two players and uses player/missile graphics and delightful sound effects.
PRICE . . . . **$16.95** cassette **$20.95** diskette

DRAWPIC: by Dennis Zander (Atari 16K)
DRAWPIC provides the user with an unbelievably easy way to create screens in graphics modes 3-7. Just sit back with your joystick and use POINT PLOT, DRAW LINE, RUBBER BAND fill and COLOR SET to create beautiful images on your Atari. Full or partial screen images are saved as string data in the program and can be instantly recalled and combined into new images using machine language subroutines. These graphic images can be easily incorporated into your own programs. The images of HODGE PODGE and the landscape of BETA FIGHTER were made using DRAWPIC.
PRICE . . . . **$29.95** cassette **$33.95** diskette

□ **ROCKET RAIDERS** by Richard Petersen (Atari 24K)
Defend your asteroid base against pulsar bombs, rockets, lasers, and the dreaded "stealth saucer" as aliens attempt to penetrate your protective force field. Precise target sighting allows you to fire at the enemy using magnetic impulse missiles to help protect your colony and *its vital structures*.
PRICE . . . . **$19.95** cassette **$23.95** diskette

□ **FOREST FIRE TWO:** by Richard Petersen (Atari 24K)
FOREST FIRE has been enhanced and now offers a two player mode for head to head competition to see who can survive, suffer the least damage and put their fire out first. User input now determines landscape, wind and weather conditions, offering limitless game variation. FOREST FIRE's excellent color graphics have been made even better, turning your computer into a super-detailed fire scanner.
PRICE . . . . **$16.95** cassette **$20.95** diskette

□ **FORM LETTER SYSTEM:** (Atari, North Star and Apple)
This is the ideal program for creating personalized form letters! FLS employs a simple-to-use text editor for producing fully justified letters. Addresses are stored in a separate file and are automatically inserted into your form letter along with a personalized salutation. Both letter files and address files are compatible with **ARTWORX** MAIL LIST 3.0 and TEXT EDITOR programs.
PRICE . . . . **$39.95** diskette

□ **PILOT:** by Michael Piro (Atari, 16K)
Pilot your small airplane to a successful landing using both joysticks to control throttle and attack angle. PILOT produces a true perspective rendition of the runway, which is constantly changing. Select from two levels of pilot proficiency.
PRICE . . . . **$16.95** cassette **$20.95** diskette

□ **TEXT EDITOR:** (Atari and North Star)
This program is very "user friendly" yet employs all essential features needed for serious text editing with minimal memory requirements. Features include common sense operation, two different justification techniques, automatic line centering and straightforward text merging and manipulation. TEXT EDITOR files are compatible with **ARTWORX** FORM LETTER SYSTEM.
PRICE . . . . **$39.95** diskette

□ **MAIL LIST 3.0:** (Atari, Apple and North Star)
The very popular MAIL LIST 2.2 has now been upgraded. Version 3.0 offers enhanced editing capabilities to complement the many other features which have made this program so popular. MAIL LIST is unique in its ability to store a maximum number of addresses on one diskette (typically between 1200 and 2500 names!). Entries can be retrieved by name, keyword(s) or by zip codes. They can be written to a printer or to another file for complete file management. The program produces 1, 2 or 3-up address labels and will sort by zip code (5 or 9 digits) or alphabetically (by last name). Files are easily merged and MAIL LIST will even find and delete duplicate entries! The address files created with MAIL LIST are completely compatible with **ARTWORX** FORM LETTER SYSTEM.
PRICE . . . . **$49.95** diskette

□ **THE VAULTS OF ZURICH:** by Felix and Greg Herlihy (Atari, 24K, PET)
Zurich is the banking capital of the world. The rich and powerful deposit their wealth in its famed impregnable vaults. But you, as a master thief, have dared to undertake the boldest heist of the century. You will journey down a maze of corridors and vaults, eluding the most sophisticated security system in the world. Your goal is to reach the Chairman's Chamber to steal the most treasured possession of all: THE OPEC OIL DEEDS!
PRICE . . . . **$21.95** cassette **$25.95** diskette

□ **BRIDGE 2.0** by Arthur Walsh (Atari (24K), Apple TRS-80, PET, North Star and CP/M (MBASIC) systems)
Rated #1 by Creative Computing, BRIDGE 2.0 is the only program that allows you to both bid for the contract and play out the hand (on defense or offense!). Interesting hands may be replayed using the "duplicate" bridge feature. This is certainly an ideal way to finally learn to play bridge or to get into a game when no other (human) players are available.
PRICE . . . . **$17.95** cassette **$21.95** diskette

□ **ENCOUNTER AT QUESTAR IV:** by Douglas McFarland (Atari, 24K)
As helmsman of Rikar starship, you must defend Questar Sector IV from the dreaded Zentarians. Using your plasma beam, hyperspace engines and wits to avoid Zentarian mines and death phasers, you struggle to stay alive. This BASIC/Assembly level program has super sound, full player missile graphics and real time action.
PRICE . . . . **$21.95** cassette **$25.95** diskette

NEW PROGRAMS!

GOLDEN GLOVES: by Douglas Evans (Atari 24K)
Use your joystick to jab, block and duck as each player attempts to land the knockout punch. This unique real-time program brings all of the excitement of ringside to your Atari. GOLDEN GLOVES is a one or two-player game, or you can be a spectator as the computer controls both fighters.
PRICE . . . . **$22.95** cassette **$26.95** diskette

CRAZITACK: by Peter Adams (Atari 16K)
The Crazies are attacking us and the only defenses are three MX bases. Missiles can be launched singly or in a salvo, but it is doomsday when you run out of missiles.
PRICE . . . . **$17.95** cassette **$21.95** diskette

DOMINATION: by Alan Newman (Atari 24K)
Between one and six players compete for power via economic, diplomatic and military means in this award-winning game. You must make decisions quickly, exercise skillful hand-eye coordination, outguess your opponents and cope with random events.
PRICE . . . . **$17.95** cassette **$21.95** diskette

POKER TOURNEY: by Edward Grau (Atari 32K, Northstar)
You are entered in a high stakes Draw Poker Tournament facing six opponents including Lakewood Louie, Shifty Pete and Dapper Dan. Each has his own style of play and of bluffing. POKER TOURNEY utilizes the Joker, has true table stakes play and each hand is played based on pot odds. The Atari version's graphics and sound are superb of course (programmed by Jerry White) making POKER TOURNEY the class program of its type.
PRICE . . . . **$18.95** cassette **$22.95** diskette

HAZARD RUN: by Dennis Zander (Atari, 16K)
The sheriff has spotted you and you must make the treacherous run through Crooked Canyon past Bryan's Pond to the jump at Hazard Creek and safety. You can even put the joystick-controlled GEE LEE car up on two wheels to make it through some tight spots. A lead foot is not always the answer as you dodge trees, rocks and chickens in this nerve-racking game. HAZARD RUN employs full use of player/missile graphics, re-defined characters and fine scrolling techniques to provide loads of fast action and visual excitement.
PRICE . . . . **$27.95** cassette **$31.95** diskette

ORDERING INFORMATION

Call ARTWORX toll-free number to order direct:
800-828-6573
In New York, Alaska, Hawaii call:
(716) 425-2833

All orders are processed and shipped within 48 hours.

Shipping and handling charges:
Within North America: Add $2.00
Outside North America: Add 10% (Air Mail)
New York State residents add 7% sales tax

Quantity Discounts:
Deduct 10% when ordering 3 or more programs

Ask for ARTWORX at your local computer store.

Write for FREE Catalogue listing more information about these and other quality ARTWORX programs.

Artworx
150 North Main Street Fairport, NY 14450

HARDWARE REVIEW: THE VOICEBOX

THE ALIEN GROUP
27 WEST 23rd STREET
DEPT. AL-1
NEW YORK, NY 10010

$169.00 RETAIL

by Brian Moriarty

Speech synthesis is a great way to improve the human engineering of your software. Used correctly, it can keep mundane prompts and announcements off the TV screen, and give you extra space to do the thing ATARI does best — graphics!

There are a couple of good voice synthesizers available for use with the ATARI 400/800 System. Unfortunately, these devices (from Votrax and Street Electronics) both carry list prices of over $350. In addition, your system must include an ATARI 850 Interface Module ($220 extra) for the units to work properly. This is a steep investment for the average hobbyist.

A few months ago, a company called The Alien Group in New York introduced their "Voicebox" Speech Synthesizer. For $169, you get a metal box slightly larger than a pack of cigarettes, a disk (or cassette) full of software and a stapled sheaf of 11 typewritten pages. This modest package is capable of producing high-quality synthetic gab at a fraction of the cost and trouble of competitive units. Best of all, it can run **without** an 850 Interface, external speakers or special equipment of any kind.

How To Use It

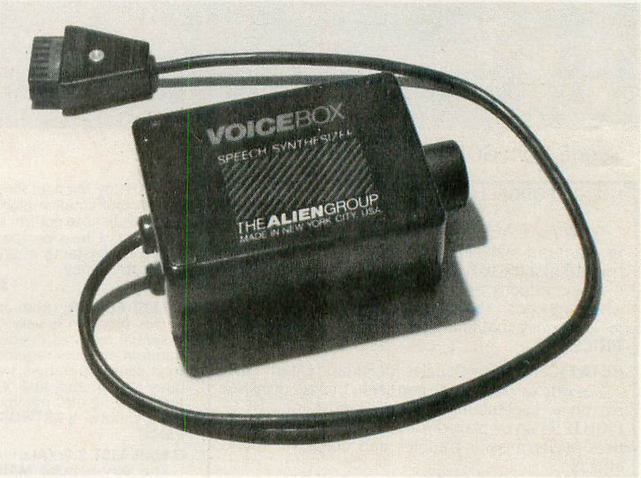
The Voicebox connects to your ATARI computer by plugging it into the serial I/O jack normally used to attach a cassette or disk drive. (If you already have a disk attached, the unit can be connected to the extra I/O jack on the back of the drive. Cassette users must disconnect the tape drive.) Voicebox steals all the power it needs directly from the computer, so there are no batteries or AC wires to fool around with.

The system works by translating ASCII character bytes into sound. To get a line of speech, you set a BASIC text string (A$) equal to the phrase you want to hear (for example, A$="HELLO"). You then pass the address of A$ to a machine-language program which sends each character in the string to the Voicebox unit. Voicebox converts the characters to an appropriate audio waveform and feeds it back into the ATARI, which reproduces the sound through its regular TV output channel. There is no interference with the ATARI's built-in sound generators or the TV picture.

Both the cassette and disk versions of the Voicebox software come with two demonstration programs written in ATARI BASIC. "VB" (16K) lets you type in and listen to sentences and compile your own pronunciation "dictionaries" (more on these later). "VBRF" is an expanded 32K version of VB which includes a random sentence generator and an amusing "talking face" routine that produces lip-sync animation while Voicebox is speaking.

A third program called "VBS" is included on the disk version so that you can use Voicebox in your own BASIC software. It's a LIST-format file which lets you read in dictionaries and access the Voicebox ML routines with a simple GOSUB to line 1000. VBS uses about 4400 bytes of RAM when loaded and executed (not counting dictionaries). Cassette users who want to use Voicebox in their BASIC software must build around the VB demo program.

An additional disk file, "VB.OBJ," contains a relocatable subroutine that allows machine-language programmers to send character bytes to Voicebox by passing them through the 6502's X-register.



VB, VBRF and VBS let you type in straight English sentences and hear a somewhat recognizable speech output. You can also use special phonetic symbols to get better definition of the words. Voicebox automatically distinguishes between English and phonetic syntax in the same line, making speech composition fast and painless. You can raise or lower the pitch of the reproduced sound (all within a single word, if you like) and add pauses of any length for greater clarity. A complete list of phonetic control symbols is included with the documentation.

"The Rain In Spain..."

The Voicebox software also lets you create "dictionaries" of phonetic equates. These dictionaries can be individually saved onto cassette or disk and recalled for later use.

For example, suppose you want Voicebox to say the word "ATARI." If you just type in ATARI, you'll get an output that sounds something like "aytayree." The phoneme string AH-TAH-RE produces a more natural-sounding response, but you don't want to type the whole thing every time. So you type:

AH-TAH-REE = ATARI

and Voicebox will deliver your special pronunciation every time you enter the word ATARI. Simple, right? You can add or remove entries from a dictionary or review its entire contents at any time. Disk users can create lots of dictionaries on a single diskette and retrieve any one of them in seconds. And the software includes a handful of sample dictionaries to get you started.

Marvelous, but...

As a piece of hardware, Voicebox is a delight. It's compact, inexpensive, easy to connect and relatively foolproof. The BASIC demonstration programs aren't coded very efficiently, but they do a fairly good job of showing off the system's features and possibilities.

The least attractive feature of the Voicebox package is the documentation. It's disappointing. There are no printed listings of the programs and no explanation of the machine-language routines that drive the Voicebox unit. And the instructions for making Voicebox work with your own software are skimpy at best.

It isn't too difficult for an experienced programmer to wade through the source code and figure out what's going on. But beginners will probably be discouraged by the strange character strings that make up Voicebox's machine-language software interface. Later releases of Voicebox will hopefully include a more complete explanation of the software and (dare we hope?) a technical description of the system.

The Bottom Line

For the majority of ATARI users who want to add speech to their programs, The Alien Group's Voicebox is probably the best choice. It delivers reasonably well-articulated speech with a minimum of fuss and cost. Voicebox isn't as versatile or well documented as the Votrax Type 'N Talk or Street Echo-GP units, but for sheer dollar value it's a tough act to beat. □

SOFTWARE REVIEW: TRIVIA TREK

Swifty Software
64 Broad Hollow Rd.
Melville, NY 11747

$29.95 32K Disk

Review by the Program Doctors

TRIVIA TREK, the newest release from Swifty Software, written by Jerry White, is just that — a fun trek through massive amounts of information. This game is reminiscent of one of the very first computerized games to infiltrate the pinball arcades. Remember, for a quarter you could pick one of the categories and test your knowledge on sports, movie stars, TV, etc. Well, the computer got smaller and better and the game got bigger and bigger.

TRIVIA TREK has fifty different categories each containing ten multiple choice questions. You have the option to choose your own category or have the computer do it for you. Since this game can be played by one or two players, the random selection of categories will no doubt head off any arguments about picking only subjects the players know well. The categories cover a range of subjects from the expected (TV, Movie Stars, Pop Music, World Capitals) to the unexpected (Bathroom, TV Soaps, Body Language, Shopping) and are interesting and imaginative. The questions are all multiple-choice and the answer choices (four for each question) are interspersed with Jerry White's warped sense of humor.

The only drawback to this program is that once you have gone through all the questions you *will* know most of the answers, you will not want to play any more and no one will want to play with you. Well, guess what? It's not a drawback at all because the best feature of this program (at least in our opinion) is you can create and edit your own trivia files.

This function has been designed to be simple to use, and it is a very easy (and fun) task to compile your own questions and answers. We made up three so far, but instead of using "standard" trivia categories, we chose to personalize them even more. We wrote a file about our son, one about our friends, and one about A.N.A.L.O.G. Magazine. Use your imagination and like us, you too can make your parents feel inadequate, antagonize your friends, and possibly get fired from the A.N.A.L.O.G. staff.

Once again, Jerry White has shown how much can be accomplished with graphics and sounds using BASIC. This is a great party game and a good way to get "non-computer" people addicted to your computer. This program is a real treat and provides some welcome relief for those of you who are "arcaded out".

Sample Trivia Trek Question

The Program Doctors think that Jerry White's TRIVIA TREK is:

1. Fun & addicting
2. Well written and well-documented
3. A relief from shooting *invaders*, pounding apples and eating dots
4. All of the above and more

That's all, folks! □

MIXING CTIA AND GTIA GRAPHICS

by William W. Hough

The GTIA television interface chip provides expanded color graphics capability for the ATARI 400 and 800 computers. My first introduction to GTIA style graphics was an article by Clyde Spencer in the July, 1981 issue of COMPUTE ("ATARI Graphics: 16 Colors!"). Before I had my ATARI television interface converted from a CTIA to a GTIA chip, I used the techniques in that article, and a lot of experimentation, to get eight distinctly colored boxes on one horizontal line. Five of the eight boxes, however, were distinguishable only because of alternating colors within the box.

When I first ran my program on an ATARI equipped with a GTIA chip, I expected to see a greatly different graphics presentation. To my surprise, there was no difference. After obtaining **"De Re ATARI"**, and reading the appendix on GTIA graphics, I understood why. The GTIA is upward compatible with the CTIA, and I needed to do a bit of POKEing around to activate GTIA graphics. After a few modifications, my program produced seven lovely, controllable colors (keep reading to find out what happened to the eighth). However, my program contained a custom display list and presented all three styles of text on the screen as well as the colored boxes. With GTIA graphics enabled, my text became colored blobs. This led to a learning experience, which in turn prompted me to write this article. Should you ever need to mix GTIA map mode graphics with text, I hope it will help you avoid some of the pitfalls I encountered.

If you don't know whether your ATARI has a CTIA or GTIA chip, type in and RUN the following program:

```
10 GRAPHICS 10
20 GOTO 20
```

If the screen turns and remains black, you have GTIA; if it returns to blue, you've got CTIA.

We will start with a Graphics 8 display, and modify it until we have mixed graphics 10 and zero on the screen. In between, there will be plenty of room for experimentation. If your ATARI has a CTIA rather than a GTIA chip, take heart, as this article will show you how to get the most out of it. You will be able to see examples of many colors on the screen simultaneously, and the alternating color effect described by Spencer. I also promise to tell you when to stop reading.

After a program segment has been RUN and you are tired of looking at the screen, program lines may be listed and changed directly in the text window. You should enter the suggested direct commands in the text window, as they will change the colors already on the screen. For major program additions, and when you GTIA owners can't read the text because it has turned to blobs, press SYSTEM RESET or type "GR.0" to return to a mode zero display. Here is the starting program:

```
10 GRAPHICS 8
19 REM : POKE SOME COLORS INTO ALL 9 C
OLOR REGISTERS
20 FOR I=0 TO 8:POKE 704+I,10+30*I
30 NEXT I
69 REM : DRAW SOME BOXES ON THE SCREEN
70 FOR I=0 TO 8
80 COLOR I
90 FOR K=1 TO 6
100 PLOT 9*I+K,10:DRAWTO 9*I+K,150
110 NEXT K:NEXT I
119 REM : PRINT SOME MODE 0 TEXT
120 ? "DEMO OF MIXED CTIA & GTIA GRAPH
ICS"
140 END
```

You should see four black (really, very dark blue-green) vertical bars on the left of the screen. As the book says, Graphics 8 is a one color, two luminance mode. The luminance of the bars is from color register 709 (in this case, 160, or very dark), and the background is from 710 (in this case, 190, or light blue-green). Mode zero text uses these same registers in a like way. Try POKEing 709 with 174; this sets the luminance of the bars and the text to the same as the background. Why only four bars? There really are nine. Try replacing "I" in line 80 with "1" (odd number), and you will see all nine. Then try "2" (even number), and they will all disappear.

If you would like to observe the color artifact phenomenon at this point, change the "2" in line 80 to "1", and change the second "9*I+K" in line 100 (in the DRAWTO statement) to "9+I*K". You should now see two new colors where the bars overlap. For a pretty combination, type the direct statement: "POKE 709,30". You can try other POKEs to 709 and 710.

We shall now tell the operating system to interpret the screen memory as Graphics 10. Return the program to its original form (change line 80 to "COLOR I" and line 100 to "DRAWTO 9*I+K"), and add the following:

```
49 REM : TELL OS YOU WANT GRAPHICS 10
50 POKE 87,10
```

What magic! More colors, wider bars. All but two are broken, and you can't see the first because it is the same color as the background. Change "I" in line

80 to "5+5*I". Now the first three bars are distinct solid colors, and all nine are present. You can try some POKES to 709 and 710, but until you add the next program segment, the full flexibility of CTIA graphics won't be apparent.

To this point, we have been using the two-color ANTIC mode 15 (Basic mode 8) with a little trickery thrown in. The only involved color registers (actually their shadow registers) are 709, 710, and the background register, 712. We can go after more colors by invoking ANTIC mode 14, which is a four-color mode not supported by BASIC. This takes some modification to the display list. Add the following subroutine and its call:

```
40 GOSUB 200
198 REM : SUBROUTINE TO ENABLE ANTIC M
ODE 14
199 REM : FIND BEGINNING ADDRESS OF DI
SPLAY LIST
200 DL=PEEK(560)+256*PEEK(561)
209 REM : TURN ANTIC OFF
210 POKE 559,0
219 REM : SET OP CODES IN DL FOR ANTIC
 MODE 14
220 POKE DL+3,78:POKE DL+99,78
230 FOR I=DL+6 TO DL+98
240 POKE I,14:NEXT I
250 FOR I=DL+102 TO DL+166
260 POKE I,14:NEXT I
269 REM : TURN ANTIC ON
270 POKE 559,34:RETURN
```

When the new program is RUN, you should have a different set of colors. You should play with POKEing 708 through 710 and 712 with any number between 0 and 255. This is the type of experimenting I did to obtain eight distinguishable colored boxes.

We have not yet taken advantage of your GTIA chip, if indeed you have one. That's next. (If you have CTIA, stop here and save the rest of the article until you have your computer upgraded with a GTIA chip.)

First, remove line 40 and change line 80 back to "COLOR I". The available documentation (eg. **De Re ATARI**) tells us that the high-order bits of the PRIOR register must be set to invoke GTIA graphics. For Graphics 10, we need to set the highest order bit to 1. We will do this through PRIOR's shadow register, GPRIOR. Add the following:

```
59 REM : SET HIGH BIT OF GPRIOR TO INV
OKE GTIA GRAPHICS
60 POKE 623,128
```

You now have lots of solid colors, the first being the same as the background. But the text has turned to those colored blobs. The next addition is a subroutine and its call that adds a display list interrupt to clear the high-order bit of PRIOR just in time for the Graphics 0 (CTIA) text:

```
40 GOSUB 300
298 REM : SUBROUTINE TO CLEAR HIGH BIT
 OF PRIOR
299 REM : FIND BEGINNING ADDRESS OF DI
SPLAY LIST
300 DL=PEEK(560)+256*PEEK(561)
309 REM : TURN ANTIC OFF
310 POKE 559,0
319 REM : SET DLI BIT ON LINE BEFORE M
ODE 0 TEXT
320 POKE DL+166,143
329 REM : POINT DLI VECTOR TO ROUTINE
ON PAGE 6
330 POKE 513,6:POKE 512,0
339 REM : POKE IN DLI ROUTINE
340 FOR I=1536 TO 1546
350 READ A:POKE I,A:NEXT I
360 DATA 72,169,0,141,10,212,141,27,20
8,104,64
369 REM : ENABLE DLI
370 POKE 54286,192
379 REM : TURN ANTIC ON
380 POKE 559,34:RETURN
```

The text is back. You can POKE any of the color shadow registers, 704 through 712, with any number between 0 and 255. You will see that each bar corresponds to a color register (the first bar and the background are colored by 704). Be careful of 709 and 710, lest your text disappear into the background.

OK, now what was all this stuff about only seven colors and pitfalls? The seven color problem occurs if one wants, as I did, the same colored background with both the CTIA and GTIA modes. The GTIA mode takes its background color from shadow register 704, and the CTIA mode from shadow register 712. If they are to be the same, then the last color will vanish. POKE 712 with 10 and you'll see what I mean. If I hadn't come to really respect the designers of the ATARI for its marvelous capacities, I'd call this a design defect. Let's just say I don't understand why they used two different registers for backgrounds. The solution is to change the CTIA background register on the fly with the same display list interrupt that switches to a GTIA mode. But be careful, you must change the actual hardware register, 53274, and not its shadow, 712.

In the demo, we started each screen with a GTIA mode, and switched to a CTIA mode. This enabled us to invoke GTIA graphics through GPRIOR. This only works for the top-most mode on the screen. For subsequent changes, you must toggle the bit or bits of the hardware register PRIOR at address 53275. This is exactly what the sequence 141,27,208 in the DLI routine does. It tells the 6502 to put the contents of its accumulator, in this case 0, into address 27+208*256=53275. To go the other direction (CTIA to GTIA), you would change the sequence 169,0 to 169,128.

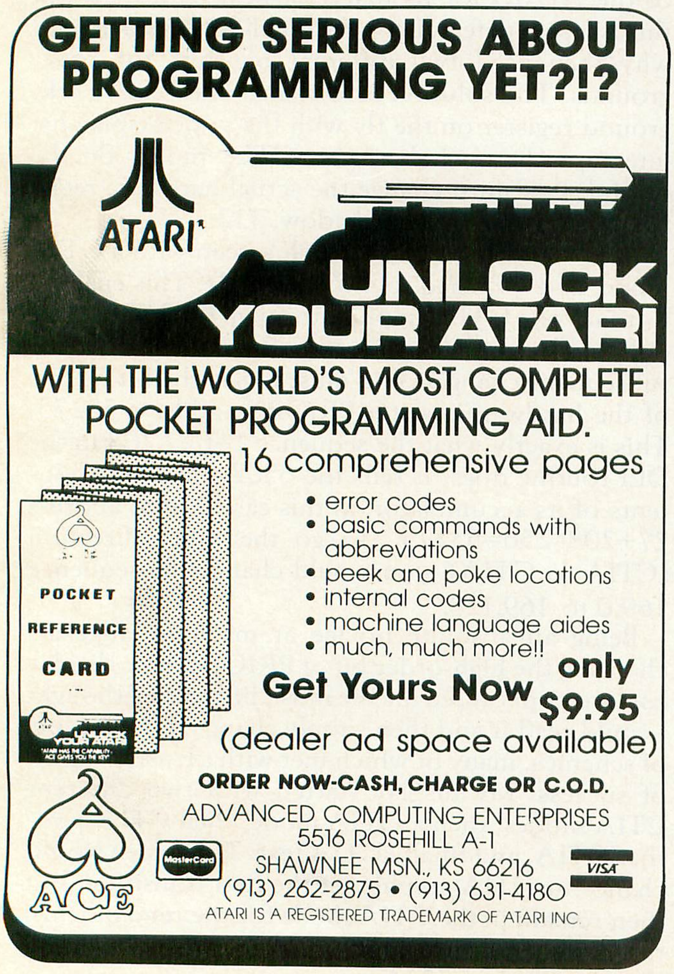
Being an absolute novice at machine language, changing the high-order bit of PRIOR with a display list interrupt caused me the most difficulty. I thought I could read it and then simply change it. I tried lots of schemes, many of which met with a frustrating bit of success. My display started with two different CTIA modes, then a GTIA mode, then CTIA, another GTIA, and finally CTIA text. The screen would change to GTIA graphics at the first transition, but then remain in the GTIA mode for the remainder of the screen. I spent a long time looking at colored blobs instead of CTIA text on the lower portion of the screen.

You experts out there have probably already deduced my error. All my schemes involved reading

PRIOR and then changing that high-order bit. Well, you can't read PRIOR. It is a hardware register, and if you try to read it, it returns decimal 15 independent of its contents. While I was setting the high-order bit at the first transition from a CTIA to a GTIA mode, I was also setting it for each successive transition until the vertical blank interrupt came along and restored it to zero.

The solution was simple. I took a suggestion from **De Re**, set the bit, and as the last part of the DLI, changed the DLI pointer to point to another DLI routine. In this routine, I cleared the bit and set the pointer back again.

In case machine language scares you like it scared me before I was forced to learn a little bit about it for these DLI routines, don't let it. I would recommend the book "6502 Assembly Language Programming" by Lance Leventhal (OSBORNE/McGraw-Hill). It has everything you need to know to write short routines like the one used in the demo without an Assembler/Editor cartridge. □



SOFTWARE REVIEW: DATA-SOFT LISP

Data-Soft
19519 Business Drive
Northridge, CA 91324

$149.95 Retail

by Carl J. Patterson

Introduction:

Writing a review of Lisp is a little difficult when not too many people know about Lisp in the first place. So this article will start out with Lisp in general, and then review a specific implementation. There are many conceptions about what Lisp is like — an artificial intelligence language, a memory waster, a group of parentheses for syntax. So, in part, I'll have to give you my own feelings about the langauge.

On Lisp:

First of all, I like Lisp. The ideas behind it, its applications and its elegance outweigh all those parentheses, and its speed. Lisp is not solely another programming language, but also a programming environment, due to its extensibility and power.

Lisp is short for **LISt Processing** language. Its strength is derived from the representation of both programs and data as lists, and from operators that work on any list structured data — possibly a program. You can, by looking at listings 1 and 2, get a quick flavor of the language. Oh yes, SETQ takes the first argument and gives it the value of the second and COND is the same as CASE statement. Listings 1, 2, and 3 all are the same program; listing 1 is a recursive version, listing 2 is a straightforward version, and listing 3 is a version in BASIC.

Why aren't more people using this language? It looks strange. Availability is certainly one problem, and memory is another, (Data-Soft requires 48K and a disk). It is a tad slow — depending on the application, and well, it does have many parentheses. But these are trade-offs in any language, BASIC vs. Assembler, for example. Once one understands the inherent limitations of any language, including its implementation, the reasons for using it are also clearer.

Don't get impatient, we will illustrate where Lisp might be used. The field of artificial intelligence is one place, where one would like to construct "expert" systems that can draw hypotheses from principles and data. In fact, some oil companies use such a system to keep track of the confusion of interrelated geological facts and rules so they can decide if a piece of land may have oil under it. I'm looking at another application, working on a program to do manipulation of symbolic expressions for formal logic. If I can understand how to make it general enough, I'll also use it for matrix manipulation, tensor calculation and other goodies. In short, I want a general purpose mathematical assistant.

On Data-Soft's Lisp:

Data-Soft has implemented Lisp for the ATARI with 48K and a single drive. It comes with an 86 page manual, a single disk, and a book — *Lisp*, by Winston and Horn. The book is so good that I had bought it before I bought Lisp. It is the single best book on the subject for a beginner, and an excellent reference. The system itself is successfully modelled after Inter-Lisp with such ATARI specific commands as sound, color, stick, trigger and peek/poke. In addition all the graphics modes are available.

On booting up, I was in control of Lisp version 2.1, so without reading the manual, (version 2.0), I typed in:

Listing 1.

```
(DEFINEQ FIBO (N)
   (COND ( (EQ N 2) 1)
      ( (EQ N 1) 1)
      (T (+ (FIBO( SUB N 1) ) (FIBO( SUB N
      2) ) ) )
   )
)
```

which defines a program called FIBO recursively, that is, in terms of itself. It worked, generating the Nth Fibonacci number. (i.e. a number from the sequence 1 1 2 3 5 8 13 21 34 etc. Do you know the rule for generating this sequence?)

With a slightly more complex program it becomes necessary to use the editor. The commands take a single evening to understand and use. Written in Lisp, the editor is a joy. (Take it from someone who tried, be careful editing the editor). It can step in to different levels of parentheses, remove or insert parentheses or expressions, and pretty-print listings as the above program was done. This last feature helps to find both logical and structural faults. The editor also has features to save and retrieve files, which like almost everything else in Lisp, is a list.

Standard Lisp I/O functions of READ, READA and READC are implemented as are PROG, PROGN, MACRO, DEFINE and DEFINEQ. For those acquainted with MACLISP there is a package that simulates the actions of many MACLISP functions, (including property lists — for Inter-Lisp uses A-lists). Also included are: a program for a light show demo; and RPN calculator program; yet another version of ELIZA and Towers of Hanoi; CLISP, a program which changes algebraic expressions into valid Lisp expressions; and a worthwhile utilities package.

The manual is a pleasant surprise. Looseleaf bound with plenty of examples and a clarity usually reserved for a cold drink on a hot summer day, it is a refreshingly complete reference for this implementation of Lisp.

Listing 2.

```
(DEFINEQ FIBO (LAMBDA (N)
  (PROG (XX N1 N2 COUNT)
     (SETQ XX 1)
     COND ( (EQ N 2) RETURN XX) )
          ( ( EQ N 1) (RETURN XX) ) )
    (SETQ COUNT 2)
    (SETQ N2 1)
    (SETQ N1 1)
 LOOP
    (SETQ X (+ N1 N2) )
    SETQ N2 N1)
    (SETQ N1 X)
    SETQ COUNT (+ COUNT 1) )
    (COND ( (EQ COUNT N) (RETURN XX) ) )
    (GO LOOP) ) ) )
```

A non recursive version of the program in Listing 1.

Listing 3.

```
10  INPUT N
20  N1=1:N2=1
30  IF N=1 OR N=2 THEN X=1:GOTO 90
40  FOR I=3 TO N
50  X=N1 + 2
60  N2=N1
70  N1=X
80  NEXT I
90  PRINT X
```

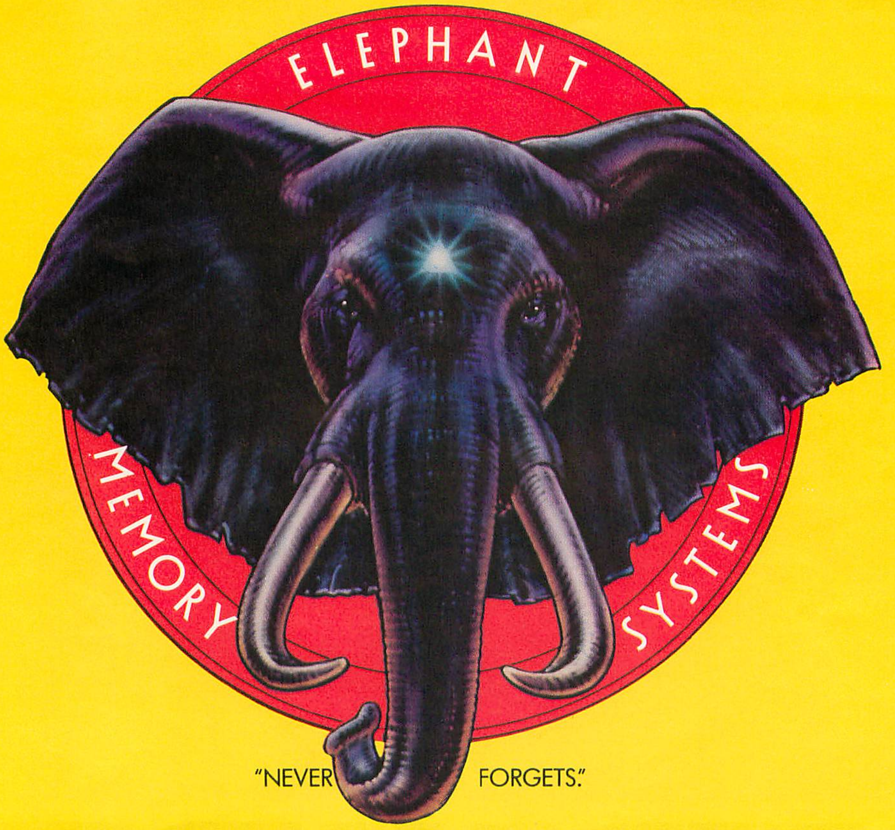
The problems that exist are minor, the worst being that GENSYM from the MACLISP package does not work. Although peek and poke are implemented there is no simple way to transfer control to a machine language program and this makes the problem of speed painful. The execution speeds of the above programs are 16, 1, and 1/6 seconds respectively, (which says the algorithm used affects speed greatly). But speed of execution is not the reason one would use Lisp anyway, (or BASIC for that matter). Rather, applicability to a given problem is the reason for using Lisp. Consider a program like ELIZA. It can be written in Assembly language and run much faster than Lisp. But the time to write or modify the program far outweighs the advantages in speed.

The final problem seems to result from some difficult choices facing the people when they wrote the language. Lisp usually has infinite, (or large), precision arithmetic. But this version uses the routines in the OS rom. This saves space — in not having to write another routine — but means it has the same limits of accuracy as ATARI BASIC, and its always floating arithmetic.

Summary:

In summary, Lisp is not a language for everyone. But if you have the application Data-Soft Lisp is a good choice. Do I like the product? Emphatically, yes — and if I didn't have it, I'd buy it! □

REMEMBER:



MORE THAN JUST ANOTHER PRETTY FACE.

Says who? Says ANSI.

Specifically, subcommittee X3B8 of the American National Standards Institute (ANSI) says so. The fact is all Elephant™ floppies meet or exceed the specs required to meet or exceed all their standards.

But just who is "subcommittee X3B8" to issue such pronouncements?

They're a group of people representing a large, well-balanced cross section of disciplines—from academia, government agencies, and the computer industry. People from places like IBM, Hewlett-Packard, 3M, Lawrence Livermore Labs, The U.S. Department of Defense, Honeywell and The Association of Computer Programmers and Analysts. In short, it's a bunch of high-caliber nitpickers whose mission, it seems, in order to make better disks for consumers, is also to make life miserable for everyone in the disk-making business.

How? By gathering together periodically (often, one suspects, under the full moon) to concoct more and more rules to increase the quality of flexible disks. Their most recent rule book runs over 20 single-spaced pages—listing, and insisting upon—hundreds upon hundreds of standards a disk must meet in order to be blessed by ANSI. (And thereby be taken seriously by people who take disks seriously.)

In fact, if you'd like a copy of this formidable document, for free, just let us know and we'll send you one. Because once you know what it takes to make an Elephant for ANSI . . .

We think you'll want us to make some Elephants for you.

ELEPHANT.™ HEAVY DUTY DISKS.

For a free poster-size portrait of our powerful pachyderm, please write us.
Distributed Exclusively by **Leading Edge Products, Inc.,** 225 Turnpike Street, Canton, Massachusetts 02021
Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.

Your greatest challenge lies ahead-and downwards.
ZORK I
An INTERLOGIC prose adventure
INFOCOM
The next step downward to danger.
ZORK II
An INTERLOGIC prose adventure
INFOCOM
It all comes down to this.
ZORK III
An INTERLOGIC prose adventure
INFOCOM
A LOCKED DOOR. A DEAD MAN.
And 12 Hours to solve the murder.
DOCUMENTARY EVIDENCE
DEADLINE
First of the INTERLOGIC Mystery Series from Infocom
The challenge was issued eons ago from light years away
STARCROSS
INFOCOM

WE'RE WRITING

THEM AS FAST AS WE CAN!

At the rate we're going, we'll have these pages filled by 2083. And by 2084, people will be clamoring for the next Infocom creation.

We hate to disappoint our public. So we keep you waiting. Because while the software factories are cranking out arcade game after arcade game, pulpy adventure after trite fantasy, we're writing and rewriting, honing and perfecting. Before a single person enters one of Infocom's worlds, it must be crafted into a living, riveting, definitive experience.

Judging from the public's reaction, it's worth the wait. For instance, *Creative Computing* welcomed DEADLINE™ as "thoroughly engrossing and realistic," while a *Softalk* readers' poll recently voted ZORK™ I and ZORK II the most popular adventures of 1981.

And now, for the moment, your wait is over. ZORK III, your final step in the underground trilogy, and STARCROSS,™ an exploration of a new dimension in science fiction, are ready for you.

Look at them up there, the little worlds of Infocom. As our universe expands, companions will come to help fill that vast expanse of white space. Till then, they'll continue to stand alone as the best of all possible worlds.

INFOCOM™
55 Wheeler Street, Cambridge, MA 02138

Infocom's worlds are available for Apple,® Atari,® IBM, TRS-80,® Commodore, NEC, Osborne, CP/M,® and DEC.™

Apple is a registered trademark of Apple Computer, Inc. Atari is a registered trademark of Atari, Inc. TRS-80 is a registered trademark of Tandy Corporation. CP/M is a registered trademark of Digital Research, Inc. DEC is a trademark of Digital Equipment Corporation.

RELIEVE YOUR FLOATING POINT BLUES

by Mike Sueirro

You say you've got a great idea for a new wargame but the 40 X 40 dynamic battlefield matrix blew the socks off your memory? Or perhaps you have what could be THE benchmark for a chess program in BASIC, but the three 8 X 8 board matrices only leave enough memory to look one move ahead? These are typical (if somewhat exaggerated) problems associated with matrices and arrays in ATARI BASIC. BASIC arrays and matrices simply devour memory, and even constants are hungry critters if they're not managed carefully. The source of this problem is the fact that all numbers in ATARI BASIC are represented in floating point — this includes constants, variables, arrays — all of them. To represent a floating point number requires 7 bytes for a constant and 6 bytes for each element in an array or matrix. What we need are a variety of methods that allow us to handle these mind blowers without sacrificing any more memory real estate than necessary. So, let's look at some ways to really cut down your memory needs. But hang on! The ride might get a little bumpy.

Constant Pain

The BASIC Reference Manual shows us one way out of the forest. On page 62 it says that a constant requires 7 bytes of memory at each use, but a variable only needs one byte each time it's used (except for an 8 byte overhead when we initialize the variable). That means we save 6 bytes every time we can use a variable in place of a constant. Terrific! In one program I recently played with, the constant zero was used 82 times. By replacing each constant with a variable whose value was zero I was able to save nearly 500 bytes of memory. And that was only one of many constraints used in the program.

If you find you're having trouble keeping track of all these variables, you might consider setting up a numbering scheme. One method is to let A=1, B=2, . . . , I=9, and J=0. Or, if you like to use J as a dummy variable, then let Z=0. Now your variables become easy to construct and interpret. For example, the value 150 becomes AEZ while the line number 5240 becomes EBDZ. Piece of cake, right?

There are only two constraints to using this technique. First, due to the 8 byte overhead for a variable, be sure you use the constant at least twice, preferably three times before you substitute a variable. Second, the ATARI BASIC Variable Table has room for only 128 variables. If you approach this limit, you'll have to be more selective in choosing your converts. Till then, though, it's happy hunting. And don't forget that you can use these variables for line numbers in GOTO's and GOSUB's as well.

An Array of Possibilities

Converting constants to variables is a neat trick, but it still doesn't help us deal with those greedy arrays and matrices. We still have to find a way to overcome paying the floating point overhead when we use these tools. In Issue 7 of *A.N.A.L.O.G.* (RESTORE Your Mental Health) I showed you how to reduce memory consumption in your numeric and string arrays and matrices. There was one hitch, though. The data in the arrays had to be constant. A good deal if you can use it, but it still doesn't solve the problems of a dynamic array or matrix.

Here's how we handle those beasties. If you're using integer data only, or if all your numbers can be scaled to integer values, then we can convert your numeric array or matrix to a string. One hitch, though. We're going to convert your numeric data to ATASCII characters and store them in a string. This means that your data must fall into the range of 0-255, or be scaled into that range, i.e. 6325-6580 would fit since we can scale the data into the range 6325 + 0 to 6325 + 255. But don't run off if your data exceeds this range. There's hope for you too.

Within the ATARI all characters are represented by an internal code. However, the ATARI can also represent these characters in ATASCII code. Given the string functions available to us, the ATASCII code is what's important for our application. What we need to do is develop a method that allows us to address the elements of a string as if they are part of an array or matrix. We also need a technique for converting our data to character codes and back again. Since it's the easiest, let's try the last item first.

Switching between numbers and string characters is fairly simple. We just use the string functions CHR$(X) and ASC(X). The CHR$(X) function identifies our number as the ATASCII code for a particular character that we can store in our string. The ASC(X) takes the ATASCII code number and reidentifies it as a numeric quantity. We don't care one whit what those characters are as long as we can use them to store our numeric data. We could print out our string if we wish to see the characters but it would just be gibberish. But then, my wife says that's representative of many of my programs anyway.

The more difficult problem is in treating the string as an array or matrix. Again, we'll take the simplest problem first. The structure of an array is essentially the same as a string. If we want the Xth element in array FUDGE, then we simply call FUDGE(X). For a string we must cite both the first and last string elements we want. Since we're only interested in one element at a time, we get the Xth element of the string FUDGE$ by calling FUDGE$(X,X). Easy enough, right? This is the procedure for both reading and writing into the string representing an array.

Enough of the simple stuff. Now let's look at matrices. The memory in your ATARI is actually a linear address space, just like your string is linear. When you create a matrix the BASIC interpreter just sets up a series of routines that will identify the address of the desired element when you cite the element's matrix coordinates. In other words, the interpreter simply converts your table of data into a list. All we have to do is figure out how to do the same thing!

Our matrix FUDGE has coordinates X and Y. X will be the row number in the matrix and Y will be the column number. Let's say our matrix has a total of R rows and C columns. In our string, we'll list all the elements in the first row, then all of those in the second row, and so on. So, if we want the element that's in the 4th row, 3rd column, that is FUDGE (4,3) (see Figure), and our matrix is 6 by 5, then we have to go down our string past the data for the first 3 rows to get to the row we're interested in. In this case we have 5 elements in each row so we pass over the first 15 elements in our string. This puts us into the data for the 4th row. Now we go down our string 3 elements farther. This puts us at the element that is in the 4th row, 3rd column of our table but it's the 18th element in our string.

In general, we convert our matrix location, FUDGE(X,Y), to a string position FUDGE$(Z,Z), bu passing over (X-1) rows. Since there are a known number of columns, say 5, then we pass over (X-1)*5 elements of our string. We then pass down our row to the Yth column to reach the string element corresponds to the matrix element at position X,Y. This means that our string position is the element at position (X-1)*5+Y. To be even more general, if we dimension our matrix as (R,C) then our string position of the X,Y element is (X-1)*C+Y. Did you follow all that?

I can hear all of you hot shots in the background yelling, "What about the zero elements of the matrix?" All right, I admit that the scheme I've shown ignores the zero elements of the matrix. In fact, I'm really treating the matrix as a table with no zero coordinates. If you're the conscientious type that uses the zero elements in a matrix, then make the appropriate changes to your string dimensions and the position formula (it comes out (X*C)+Y+1). This is really only necessary though if you're converting an existing program. If you're using this string method when you start writing a program, the original formula works fine. And if you're converting an existing matrix that doesn't use the zero elements, the original formula not only works fine but also gives additional memory savings by not wasting those bytes assigned to the unused zero elements.

So, if we want to read or write to the X,Y element of matrix FUDGE, we just go to the string element FUDGE$((X-1)*C+Y,(X-1)*C+Y). Note that I had to use the position formula twice to get only the one element I wanted in the string.

O.K. We've taken something relatively simple and turned it into a bit of a mess. What did we gain? Since each matrix and array element uses 6 bytes of memory while a string element uses only one, we've saved 5 bytes of memory for every element that we converted into our string. In our 40 X 40 dynamic battlefield matrix that amounts to a savings of 8K bytes! Not bad, huh? Now maybe we can get the game into our 16K machine.

To briefly recap what we've done, if we want to store the number 12 at the X,Y position of our table whose dimensions are R,C, then we use:

```
FUDGE$((X-1)*C+Y,(X-1)*C+Y) = CHR$(12)
```

When we want to get the number back out, we use

```
FUDGE = ASC(FUDGE$((X-1)*C+Y,(X-1)*C+Y))
```

You can see that this isn't the kind of thing you want strewn all through your program or even want to poke in more than once. Put these statements into subroutines and just call them each time you need to read or write to the string.

Gluttony

Whew! Well, I hope you're still with me. I promised earlier to provide help for those gluttons whose data exceed the range 0-255, so here goes. If we take two string elements together and use them to represent a single number, we can still save 4 bytes per element. We define the first string element to be our low order element and then we constrain that element to the range 0-99. We let the second, high order element run its full range of 0-255. Then to reconstruct our number we pull out the high order element, multiply it by 100 and add it to the low order element. Presto, our range now goes 0-25599 and we still save 4 bytes per element.

Keeping track of where we are in our string now that we need to take two elements at a time is not too tricky, but it does introduce another complication on all that we've already done. A simpler method is to use two strings, one for the low order elements and one for the high order. We do eat up a bit more memory due to the initialization and naming of the second string, but that cost is minimal (9 bytes plus the name length plus the DIM statement) and the ease of conversion of our data is justification enough to make this trade off.

Here's the method we use to store and retrieve our number using two strings, HIFUDGE$ and LOFUDGE$. To store the number X in our strings we use:

```
    HIFUDGE$(Z.Z) = CHR$(INT(X/100))
    LOFUDGE$(Z,Z) = CHR$(X-INT(X/100))
```

To retrieve X is just as easy, like this:

```
X = ASC(HIFUDGE$(Z,Z))*100 + ASC(LOFUD
GE$(Z,Z))
```

Want to extend that range a little further? Remember that we constrained the low order element between 0 and 99? If we let the low order element range between 0-199 then we can let the value of our number be negative when the low order element is greater than 99 and positive when it is less than 99 (or vice versa). Now our data can range from –25599 to +25599. And we still save 4 bytes per element. One method we can use to store our numbers is:

```
HIFUDGE$(Z,Z) = CHR$(INT(ABS(X/100)))
IF SGN(X)=-1 THEN LOFUDGE$(Z,Z) = CHR$
(ABS(X) - INT(ABS(X/100)) + 100):RETUR
N
LOFUDGE$(Z,Z) = CHR$(X - INT(X/100)):R
ETURN
```

We can use the following statements to retrieve our number:

```
IF LOFUDGE$(Z,Z) > 99 THEN X=-1*(ASC(H
IFUDGE$(Z,Z))*100 + ASC(LOFUDGE$(Z,Z))
 - 100):RETURN
X = ASC(HIFUDGE$(Z,Z))*100 + ASC(LOFUD
GE$(Z,Z))
```

Pack It In

If you're like me, you've just flopped back in your chair and are ready to pack it in, right? But wait! You say your data has a very limited range but there's gobs of it and you want to handle it all in even less space then I've shown you? Bloody ingrate, if you ask me! But we'll try. Since we're using string values limited to 255 max, we don't have much leeway on the methods we can use to pack data. However, if your data can be kept in the range 0-9, then there may be hope.

The trick to this packed data method, if you haven't already guessed, is to use the least significant digit to hold one number, and use the second digit to hold the second number. The most significant digit only has a range of 0-2 so it isn't of much value. We can use the most significant digit along with the middle digit to get a range of 0-24 for one number and 0-9 on the second number, but such an unbalanced data scheme probably has very limited applicability. We won't address this option further, but it is available.

As far as packing the numbers themselves, it's just a case of multiplying the second number by 10 and adding it to the first number. Unpacking is almost as easy. We get the second number by dividing the element by 10 and taking the INT function to strip off the fraction. These are all positive numbers so we don't have any problems with this. We get the first number by multiplying the second number by 10 and subtracting it from the string element value. To put it more clearly, we build the string value like this:

```
    FUDGE$(Z,Z) = CHR$(X1+X2*10)
```

where X1 and X2 are our two numbers. We unpack the data like this:

```
    X2 = INT(ASC(FUDGE$(Z,Z))/10)

    X1 = ASC(FUDGE$(Z,Z))-X2*10
```

That's it. We can now store two numbers (0-9) that previously would have taken 12 bytes into a single string that uses only one. A phenomenal saving of 11 bytes for every two numbers. In our chess game this means that where our original three 8 X 8 matrices took 1152 bytes, we can now store the data in a piddling 96 bytes! Not too shabby, is it?

Flotsam and Jetsam

There you have it, Lesson 2 in Dr. Mike's Memory Improvement Course. If those won't help you, I'm afraid you'll just have to spring for more memory or learn Assembly language. These techniques really do work and they result in significant improvements in terms of efficient memory usage. But, like everything else, this improvement has its price. The price for converting arrays and matrices into strings is an increase in program complexity and the additional processing time required to make conversions. Whether these tradeoffs are acceptable is solely up to you and your application. One thing is certain, though. You may have the best program on the block but if it won't fit in memory, who cares? □

String FUDGE$(Z)

X,Y	Z
1,1	1
1,2	2
1,3	3
1,4	4
1,5	5
2,1	6
2,2	7
2,3	8
2,4	9
•	•
•	•
•	•
3,6	15
4,1	16
4,2	17
$\boxed{4,3}$	$\boxed{18}$
4,4	19
•	•
•	•
•	•

Matrix FUDGE(X,Y)

		Y				
		1	2	3	4	5
	1	x	x	x	x	x
	2	x	x	x	x	x
	3	x	x	x	x	x
X	4	x	x	$\boxed{x}$	x	x
	5	x	x	x	x	x
	6	x	x	x	x	x

The Official FROGGER™ by SEGA®

The Popular Arcade Game
Now Available For Home Computers

FROGGER runs on any 48K APPLE II with DOS 3.3 or ATARI 400/800 with 32K and a Diskdrive, 16K and cassette. Frogger is available for $34.95 at your local computer store or order directly from . . .

All orders by VISA, Mastercard, check, or COD. Add one dollar for shipping. California residents please include 6% sales tax.

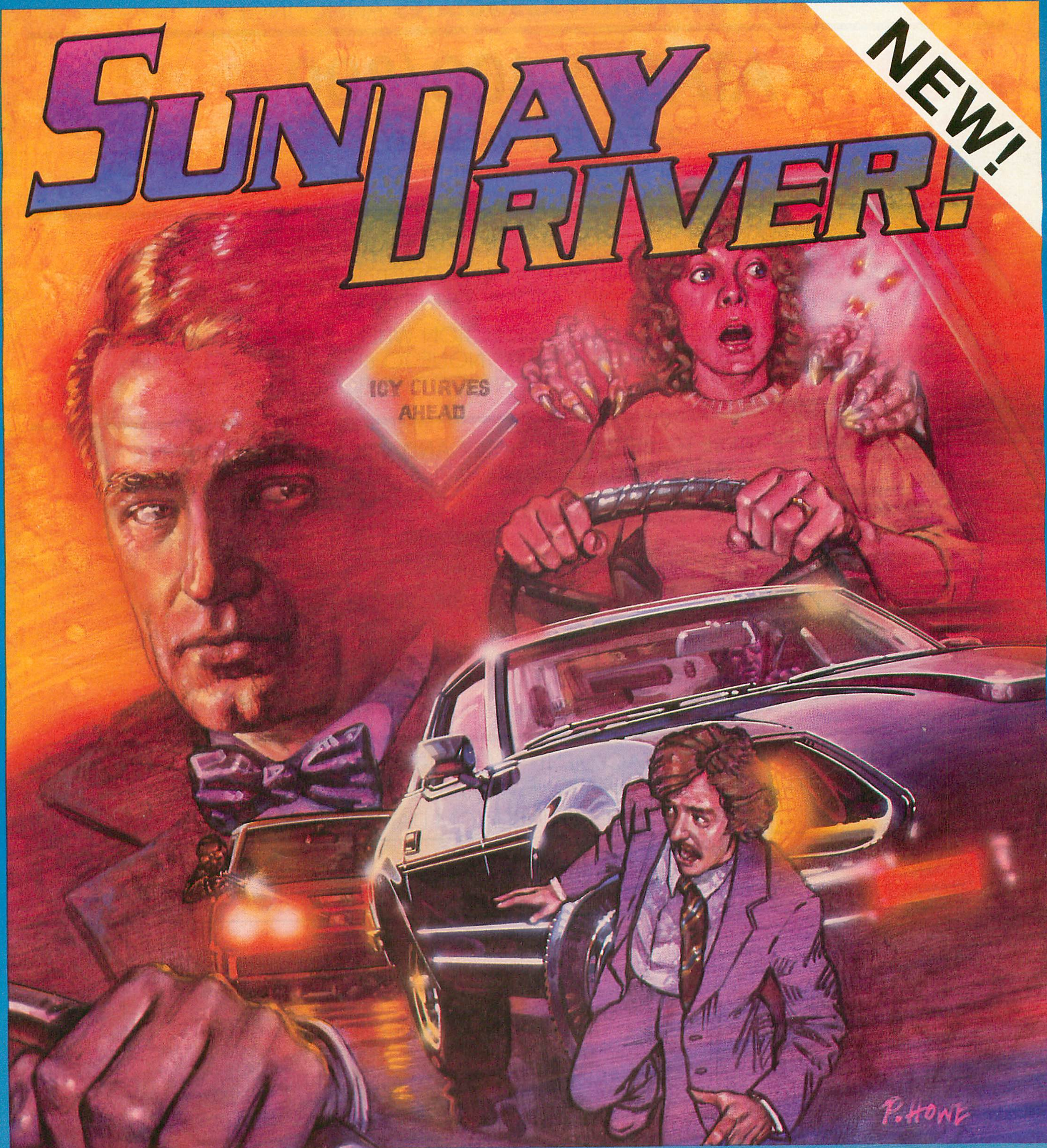
On-Line systems
36575 Mudge Ranch Rd., Coarsegold, CA 93614
(209) 683-6858

® and TM designates a trademark of Sega Enterprises, Inc.

Your mission: pilot your spaceship through a dangerous asteroid belt in a hostile, uncharted solar system while dodging zooming comets. Avoid the interstellar obstacles if you can, but beware — you may find yourself under attack by your opponent's well-armed battle cruiser! Try to escape your foe through the threatening universe of anti-matter! Options let you select from any of 128 different versions of the game. Written entirely in machine language with multi-color player/missile type graphics and outer space sound effects!

RACE IN SPACE
THE BEST-SELLING GAME FROM
A.N.A.L.O.G. SOFTWARE
16K cassette or disk $24.95

SEND CHECK OR MONEY ORDER TO:
A.N.A.L.O.G. Software
P.O. Box 23 Worcester, MA 01603
MasterCard and VISA accepted (617) 892-3488



SUNDAY DRIVER gives you four scenarios to choose from. You must beat the clock as you drive along while avoiding pedestrians, other cars, and obstacles. In other versions it's winter and you're on ice-slicked roads. In game three *it's nighttime (don't hit the ghosts).* If this sounds too easy try the 007 option — it's you against **them** on twisty roads.

SUNDAY DRIVER — the ultimate driving game.
SEND CHECK OR MONEY ORDER TO:

A.N.A.L.O.G. Software
P.O. Box 23 Worcester, MA 01603
MasterCard and VISA accepted (617)892-3488
16K cassette or disk $29.95.

AUSTIN FRANKLIN ASSOCIATES ANNOUNCES

32K - ATARI® 400/800 - 48K

$\$94^{95}$

2-YEAR FULL WARRANTY

HIGHEST QUALITY AND RELIABILITY POSSIBLE

$\$124^{95}$

(WITH 16K IN TRADE)

BOTH 32K AND 48K WORK IN THE 400 AND 800 - FULLY ILLUSTRATED INSTRUCTIONS

32K — FITS IN ANY CONFIGURATION ANY OTHER BOARD WILL
REQUIRES NO MODIFICATIONS TO THE 400

48K — AUTOMATIC MEMORY DE-SELECTION WHEN CARTRIDGE USED
4-WIRE INSTALLATION TO 400 - NONE TO 800
¼ THE POWER CONSUMPTION OF OTHER BOARDS

WE GUARANTEE THESE TO BE THE FINEST 32K AND 48K BOARDS AVAILABLE AT ANY PRICE — AND HERE'S WHY:

□ GOLD EDGE CONNECTOR — BEST CONNECTION TO BACK PLANE
TAKES FULL ADVANTAGE OF ATARI'S BUSS STRUCTURE — INSURES TOTAL COMPATABILITY

□ COMPUTER GENERATED PC BOARDS — MOST ACCURATE

□ 100 HOUR MEMORY TEST BURN-IN — GUARANTEES PERFECTLY OPERATING BOARDS

□ ALL IC'S FULLY DECOUPLED (MANY CAPACITORS) — INSURES GLITCH-FREE OPERATIONS AND NO SCREEN INTERFERENCE

□ NOT SOCKETED — SOCKETS ARE UNRELIABLE AND COST MORE

□ PARTS PURCHASED IN MASS QUANTITY — SAVINGS REFLECTED IN PRICE

WARRANTY: 2-YEAR FULL REPLACEMENT — JUST RETURN YOUR BOARD AND WE WILL SEND YOU A NEW ONE WITHIN 10 DAYS.

10-DAY FULL REFUND FROM DATE OF PURCHASE

THESE BOARDS DO EVERYTHING THE COMPETITORS WILL DO — BUT COST YOU MUCH LESS.

48K (WITHOUT 16K TRADE) **$149.95**

32K LOOPBACK — ALLOWS 32K TO BE USED ALONE IN THE 800 . . **$5.00**

48K LOOPBACK — ALLOWS 48K TO BE USED ALONE IN THE 800 . . **$5.00**

SEND CHECK OR MONEY ORDER TO:

AUSTIN FRANKLIN ASSOCIATES
43 GROVE STREET, AYER, MA 01432

FOR C.O.D., MASTERCARD OR VISA, CALL
(617) 772-0352

DEALER INQUIRIES AVAILABLE

STATIC DAMAGE IS NOT COVERED BY WARRANTY.

BUDGET WORKSHEET II

16K DISK

by Aly Kahn

Sooner or later, most home computer users come to the startling realization that the cassette tape is much too slow to allow for the efficient usage of time spent with their computer.

When I wrote the article BUDGET WORKSHEET (*A.N.A.L.O.G. #7*), I envisioned that I would continue to be ecstatic about the cassette world. Well, horror of horrors, the WORKSHEET worked so well that I soon had enough money for one ATARI 810 disk drive. My wife, as a peace offering to the computer room, even made a cover for the orphan cassette player!

After setting up the disk drive, I had a horrible thought — BUDGET WORKSHEET was written for tape users. After 50 quick disk accesses one is no longer a tape user.

Thankfully, the original program needed a minimal amount of restructuring to work on a disk system. Two new features were added to the original program to enhance its use.

The first enhancement was the inclusion of percentages of the amounts budgeted compared to the total amount. The percentage is saved in the same fashion as the screen.

The second, far more important enhancement was the ability to randomly access up to 999 individual budget screens from the program.

The random access routine uses a simple procedure to its best advantage. The standard filename has the form: D:XXXXXXXX.YYY. In this form the D: must always be present to specify the device to be opened; the x's represent the filename, and the y's represent the optional extender. By making the filename a fixed entity (D:BUDGET.), I was able to dimension a numeric array to change the extender based upon user input. Since D:BUDGET. occupies 9 spaces (including the period) and the extender occupies 3, I dimensioned the array FR$(12). The extender would therefore occupy FR$(10,12). If we then let the extender equal N$, the user can input the extender as a three-digit filename from 000 to 999 inclusive. On writing or reading of the budget file this is the user-accessible part.

The program is fairly straight-forward. The title page is displayed, followed by a prompt with two choices. If the user wishes to create a budget he types save and hits the return key. He will then be asked to input the month (alphabetically), day, and year. The screen then clears and asks the user to enter eight (or less) budget categories. Upon completion of each category the user must hit the return key for the next prompt. After the eighth category is entered and the return key depressed, the screen again clears and the user inputs the dollar and cents amounts of each category. At the final input he hits return and the complete budget is displayed in columnar structure with a date on the bottom and the total of the budget.

The whole program has been error-trapped to make it "semi-user-friendly." (It barks once in a while!) As usual I want to thank John and Anne Smith of the Computer Shoppe, Patchogue, New York. Their patience during the search for a working 32K board and my transition to a disk system was really appreciated.

Happy hacking!...□

(Editor's note: Since some readers may not have a copy of A.N.A.L.O.G. Computing #7 in which we printed the cassette version of BUDGET WORKSHEET, we elected to reprint the entire program for disk users, not ju st the cassette version modifications. Program lines changed from the cassette version are preceded by an asterisk. Line 200 of the cassette version has been deleted. Line 1002 has been added to the disk version.)

```
 0 GRAPHICS 2+16:SETCOLOR 2,4,4:CLR
 1 POKE 764,255:REM BUDGET WORKSHEET

   REV.3.D 6/30/82
 2 POSITION 5,1:? #6;"BUDGET"
 3 POSITION 5,3:? #6;"WORKSHEET"
 4 POSITION 5,6:? #6;"BY aly kahn"
 5 POSITION 5,8:? #6;"  1982  "
 6 FOR X=1 TO 200 STEP 0.1:NEXT X
*7 DIM MENU$(4),FR$(12),N$(3)
 8 ? "TO LOAD DATA TYPE LOAD":? "TO BUD
GET TYPE SAVE"
 9 TRAP 8:INPUT MENU$:IF MENU$="LOAD" T
HEN 1000:IF MENU$="SAVE" THEN 10
10 DIM A$(10),A(6),B$(10),B(6),C$(10),
C(6),D$(10),D(6),E$(10),E(6),F$(10),F(
6)
11 DIM MONTH$(9),DATE(2),YEAR(4)
12 DIM G$(10),G(6),H$(10),H(6)
13 TRAP 13:? "INPUT MONTH":INPUT MONTH
$:? "INPUT DAY":INPUT DATE:? "INPUT YE
AR":INPUT YEAR:TRAP 40000
14 ? CHR$(125)
15 ? "Type budget category- up to 9 le
tters"
16 ? "(UP TO 8 CATEGORIES,PLEASE)"
17 POKE 752,1
18 FOR L=1 TO 10:SOUND 2,INT(RND(1)*25
5)+1,10,10:NEXT L:SOUND 2,0,0,0
20 TRAP 20:INPUT A$:TRAP 40000
30 TRAP 30:INPUT B$:TRAP 40000
40 TRAP 40:INPUT C$:TRAP 40000
50 TRAP 50:INPUT D$:TRAP 40000
60 TRAP 60:INPUT E$:TRAP 40000
70 TRAP 70:INPUT F$:TRAP 40000
72 TRAP 72:INPUT G$:TRAP 40000
74 TRAP 74:INPUT H$:TRAP 40000
75 ? CHR$(125)
```

```
77 ? "Enter amount after category-up t
o 6 digits"
78 ?
79 ? "IF CATEGORY IS BLANK ,PLEASE ENT
ER 0"
80 TRAP 80:? A$;"=";:INPUT A:TRAP 4000
0
90 TRAP 90:? B$;"=";:INPUT B:TRAP 4000
0
100 TRAP 100:? C$;"=";:INPUT C:TRAP 40
000
110 TRAP 110:? D$;"=";:INPUT D:TRAP 40
000
120 TRAP 120:? E$;"=";:INPUT E:TRAP 40
000
130 TRAP 130:? F$;"=";:INPUT F:TRAP 40
000
132 TRAP 132:? G$;"=";:INPUT G:TRAP 40
000
134 TRAP 134:? H$;"=";:INPUT H:TRAP 40
000
140 ? CHR$(125)
141 ? "EXIT TO MENU HIT "M""
*142 ? "To save data to disk,hit return
*"
*143 ? :? :? :TOTAL=(A+B+C+D+E+F+G+H)
*145 ? A$,A,INT(A/TOTAL*100);"%"
*150 ? B$,B,INT(B/TOTAL*100);"%"
*160 ? C$,C,INT(C/TOTAL*100);"%"
*170 ? D$,D,INT(D/TOTAL*100);"%"
*180 ? E$,E,INT(E/TOTAL*100);"%"
*190 ? F$,F,INT(F/TOTAL*100);"%"
*192 ? G$,G,INT(G/TOTAL*100);"%"
*194 ? H$,H,INT(H/TOTAL*100);"%"
201 FOR Z=1 TO 10:SOUND 0,TOTAL,10,10:
NEXT Z:SOUND 0,0,0,0
208 ? :?
210 ? "Total",TOTAL
213 ? :? MONTH$;" ";DATE;",";" ";YEAR
217 OPEN #3,4,0,"K:"
218 GET #3,A:IF A=155 THEN 226
219 IF A=77 THEN CLOSE #3:GOTO 259
220 GOTO 218
*226 ? "File number ?(3 digits)":TRAP 2
26:INPUT N$:FR$="D:BUDGET.":FR$(10,12)
=N$:OPEN #4,8,0,FR$:TRAP 40000
228 FOR Y=0 TO 23
230 FOR X=0 TO 39
232 LOCATE X,Y,Z
233 PUT #4,Z
234 NEXT X
236 NEXT Y
240 CLOSE #4:CLOSE #5:CLOSE #3
259 POKE 764,255:GRAPHICS 0:SETCOLOR 2
,4,4
260 ? "TO START A NEW WORKSHEET, HIT S
TART"
263 ?
264 ? "TO EXIT PROGRAM,HIT OPTION"
265 ? "TO LOAD PROGRAM PUSH SELECT"
270 IF PEEK(53279)=6 THEN 0
290 IF PEEK(53279)=3 THEN ? CHR$(125):
FOR R=1 TO 10:SOUND 1,INT(RND(1)*255)+
1,10,10:NEXT R:SOUND 1,0,0,0:END
291 IF PEEK(53279)=5 THEN GOSUB 1000
295 GOTO 270
*1000 ? "K";:CLR :DIM FR$(12),N$(3):? "
File #?":INPUT N$:FR$="D:BUDGET.":FR$(
10,12)=N$:TRAP 1001:OPEN #4,4,0,FR$
*1001 IF PEEK(195)=170 THEN CLOSE #4:TR
AP 40000:POKE 195,0:GOTO 0
*1002 POKE 82,0
1005 OPEN #5,12,0,"E:"
1010 OPEN #2,8,0,"S:"
1020 FOR I=1 TO 960
1030 GET #4,A:PUT #2,A
1040 NEXT I
1050 CLOSE #4:CLOSE #5:CLOSE #2
1052 ? "FOR MENU HIT ANY KEY"
1054 IF PEEK(764)<>255 THEN POKE 764,2
55:GOSUB 259
1055 GOTO 1054
```

●

D: CHECK DATA

```
0 DATA 438,952,653,506,25,770,395,251,
371,139,858,38,362,443,767,6968
15 DATA 339,249,919,203,849,153,791,86
1,174,812,880,201,817,772,816,8836
79 DATA 822,898,57,129,137,145,153,169
,185,705,969,770,452,582,675,6848
160 DATA 898,582,705,925,601,741,592,3
,298,263,367,719,334,792,973,8793
228 DATA 293,285,214,628,448,777,817,4
42,310,908,998,640,452,711,536,8459
295 DATA 799,281,937,978,610,695,401,6
24,488,34,26,359,14,6246
```

NOREM

By Jerry White

NOREM reads a BASIC program in list format, deletes all REM statements, and writes the remless program to disk in list format. I'm sure many will find this to be a useful programmers tool. □

```
10 REM NOREM BY JERRY WHITE
20 GRAPHICS 0:SETCOLOR 2,0,0:? :? "
   DELETE REM STATEMENTS ":? :? :?
30 DIM LI$(200),LO$(200),FI$(15),FO$(1
5),U$(12)
40 ? :? "INPUT FILE MUST BE IN LIST FO
RMAT.":? :? "OUTPUT FILE WILL BE LISTE
D TO DISK."
50 ? :? "TYPE INPUT FILE NAME";:INPUT
U$
60 FI$="D:":FI$(LEN(FI$)+1)=U$
70 ? :? "TYPE OUTPUT FILE NAME";:INPUT
 U$
80 FO$="D:":FO$(LEN(FO$)+1)=U$
90 ? "K":? :? FO$(3,LEN(FO$));" LISTIN
G:"
100 OPEN #1,4,0,FI$:OPEN #2,8,0,FO$:TR
AP 240
110 INPUT #1,LI$:LI=LEN(LI$)
120 IF LI>4 THEN 140
130 GOTO 160
140 FOR ME=1 TO LI-2:IF LI$(ME,ME+3)="
REM " THEN LO$=LI$(1,ME-2):POP :GOTO 1
80
150 NEXT ME
160 LO$=LI$
170 ? LO$:PRINT #2;LO$:GOTO 110
180 IF VAL(LI$)<10 AND LI$(3,5)="REM"
THEN 110
190 IF VAL(LI$)<100 AND LI$(4,6)="REM"
 THEN 110
200 IF VAL(LI$)<1000 AND LI$(5,7)="REM
" THEN 110
210 IF VAL(LI$)<10000 AND LI$(6,8)="RE
M" THEN 110
220 IF LI$(6,8)="REM" /HEN 110
230 GOTO 170
240 CLOSE #1:CLOSE #2:? CHR$(253):? "R
EM STATEMENTS HAVE BEEN DELETED.":END
```

●

MICROSOFT•MICROPRO•LIGHTNING SOFTWARE•SIRIUS•LOGICAL SYSTEMS•SUPERSOFT

•EDUCATIONAL COURSEWARE•BRODERBUND•COMPU-THINGS•THE CORNSOFT GROUP•DATASOFT•DAKIN5

EVERY NINE MINUTES SOMEONE JOINS AMERICAN SOFTWARE CLUB

HERE ARE 10 REASONS WHY:

1. PRICE: ASC promises that members prices are the lowest available. If, within 30 days of your purchase you find a lower, nationally advertised price, we will gladly credit your account with the difference.

2. NO MINIMUM: As a Club member you are never obligated to buy anything you don't want, unlike other software clubs which sometimes require as many as three purchases every six months.

3. FAST SHIPMENT: The products we offer in our catalog are kept in stock, so orders are usually shipped within 24 hours. We also offer air shipment for those members who want it.

4. PRODUCT DESCRIPTIONS: ASC's bimonthly Compendium contains extensive product descriptions on every offering so that both novice and expert alike know what they are buying and can be sure that it's right for them and their computers.

5. WIDE SELECTION: ASC carries hundreds of programs as well as hardware and supplies for APPLE, ATARI, CP/M, IBM PC, and TRS-80 computers. Program categories include Business, Education, Entertainment and Languages/Utilities.

6. PRODUCT EVALUATION: Each product sold by ASC must first pass rigorous evaluation before it is included among the Club's offerings. Products must demonstrate both their claimed capabilities and represent sound purchases prior to their inclusion.

7. PRODUCT SUPPORT: We believe it is as important to support products as it is to sell them, so we stand behind every product we sell, and only offer merchandise from manufacturers who are committed to living up to their warranties.

8. EASE OF ORDERING: An order form and envelope is provided with every catalog, and to make ordering even more convenient, the Club also provides a toll free number for 24 hour, 7-days-a-week service.

9. CONSUMER SERVICES: The Club provides at member request, information and suggestions as to product suitability and compatibility, and when necessary can call on the assistance of additional outside sources.

10. SPECIAL OFFERS: Club members will from time to time receive special exclusive offerings that greatly enhance the value of membership.

• PROGRAM DESIGN •I.U.S.•ADVENTURE INTERNATIONAL•ALLENBACH INDUSTRIES•BIG FIVE •

•DATAMOST•ARTIFICIAL INTELLIGENCE•EDU-WARE•GALACTIC SOFTWARE •ANCHOR AUTOMATION •C. ITOH•

ASC is America's oldest and largest commercial software club.

FOR A SIX-MONTH TRIAL MEMBERSHIP WITH NO FEE OR OBLIGATION, FILL OUT THE COUPON OR CALL OUR TOLL FREE NUMBER: 1-800-431-2061
(N.Y. Residents call 914-762-5644)

AMERICAN SOFTWARE CLUB, INC.
Millwood, New York 10546

Please begin my free six-month trial membership as outlined in this ad.

Name ______________________________

Address ______________________________

City/State/Zip ______________________________

My computer is: circle

APPLE ATARi IBM PC

TRS-80 (Mods. 1, 2, 3) CP/M (8" or 5¼")

BEGINNER'S BASIC

by Thomas M. Krischan

A few days ago my neighbor Kevin finally broke down and decided to buy an ATARI 800. After opening up the box Kevin confessed to me, "Well, I'm counting on you to help me through programming. I guess you've adopted a son!" I've also adopted a great opportunity to share with the readers the trials and tribulations of the beginning programmer. After the initial ooh's and aah's of playing with his ATARI, Kevin started on a serious project, his first program. The next day or so, I stopped over to see how Kevin was progressing. He met me at the door with a smile. "Have I got something to show you!" We went upstairs to the spare bedroom, which had now become the computer room, and I sat in front of the TV. Kevin typed in RUN, a line was drawn from the top center of the screen to a random point along the base, a whistle sounded and the program repeated this sequence 24 times. Kevin then listed the program on the screen. It was 15 lines long and worked just fine. But, it was somewhat awkwardly written.

Let's spend some time looking over Kevin's original program and see if we can't remove some of this "awkwardness."

FIGURE 1. ORIGINAL PROGRAM

```
5 X=1
10 GRAPHICS 3
12 X=X+1
15 B=RND(0)*39
20 FOR M=0 TO 3
21 FOR R=1 TO 100 STEP 4
22 SOUND 0,R,10,8
25 COLOR M
30 PLOT 18,0
35 DRAWTO B,19
36 NEXT R
40 NEXT M
90 IF X=25 THEN 200
95 GOTO 12
200 END
```

There are several improvements which can be made. The FOR...NEXT loop in line 21 executes 25 times. R becomes 1,5,9,...,93, 97 but it never becomes 100, and 100 is the upper limit of the loop. This may seem nit picky, but that's what good programming is all about. We must either make the upper limit 97 or make the starting value 4. There are exceptions to this rule, however this program does not demonstrate any of them. The loop executes four statements; SOUND, COLOR, PLOT, and DRAWTO. After executing the loop all 25 times, M is still M, and B is still B. That is, the variables of the COLOR and DRAWTO statements remain unchanged. PLOT contains no variables and likewise remains unchanged. Therefore, these statements do not require this multiple execution, once is enough. More than once wastes time. Let's move the COLOR, PLOT, and DRAWTO statement in front of the loop. Also let's use a colon and put multiple statements on the same line. Here's what the first revision looks like.

FIGURE 2. FIRST REVISION

```
5 X=1
10 GRAPHICS 3
12 X=X+1
15 B=RND(0)*39
20 FOR M=0 TO 3:PLOT 18,0:COLOR M:DRAW
TO B,19
21 FOR R=4 TO 100 STEP 4:SOUND 0,R,10,
8:NEXT R
40 NEXT M
90 IF X=25 THEN 200
95 GOTO 12
200 END
```

The FOR...NEXT loop in line 20 executes 4 times. M becomes 0,1,2, and 3. M also designates the COLOR variable used by the PLOT and DRAWTO. But, COLOR 0 is the same color as the background and consequently can't be seen. So, simply eliminate zero from the loop. We can incorporate lines 5,12,90 and 95 into a simple FOR...NEXT loop, where the starting value is 2 and the upper limit is 25.

We can again use colons for multiple statements and also renumber each line into increments of ten. This is easier on the eye and allows more space for future updates. Lastly, we will reverse the order of the multiplicands for determining B in line 15. While the product is identical under either form, the latter is quicker.

FIGURE 3. FINAL REVISION

```
10 GRAPHICS 3:FOR X=2 TO 25:B=39*RND(0
)
20 FOR M=0 TO 3:PLOT 18,0:COLOR M:DRAW
TO B,19
30 FOR R=4 TO 100 STEP 4:SOUND 0,R,10,
8:NEXT R
40 NEXT M:NEXT X:END
```

What have we really accomplished? A lot! First, we've reduced Kevin's 15 line program into a 4 liner, which is much more readable. Second, we've reduced the amount of required memory by 29%. And last, we've increased the speed of the program by over 350%! □

Attack at EP-CYG-4©

a space adventure from BRAM *Inc.*

*For the ATARI*400/800*

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

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

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

*Atari is a trademark of Atari, Inc.

* ©Art Copyright R.L. Kothenbeutel

3 levels of difficulty
Advanced joystick control capabilities
Free poster included

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

© Copyright 1982 BRAM Inc.

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

PEARSON ADVENTURE SERIES

Med Systems is proud to announce the Jyym Pearson Adventure Series for the Atari computer. These adventures feature multi-word input, graphic and sound enhancements, complex story lines, and scenarios up to 30% larger than the current popular 24K series. And ours fit in 16K!

Cassettes load in 5 minutes, not 20 minutes as with our competitors. Our duplication produces reliable tapes. Diskettes boot in under 20 seconds, and save games in less than 5 seconds.

The Institute: Not just one adventure, but five! You must negotiate not only the institution, but four other scenarios in your dreams. A huge statue, a prehistoric forest, an ancient temple, and the Titanic ocean liner all conceal objects and clues to help you escape the nightmare of the Institute.
The Institute 16K cassette or diskette $19.95

Lucifer's Realm: Entering the kingdom of Satan, you discover a revolution in the making, headed by Adolf Hitler. Travel through the bowels of Hades, dealing with the most evil mortals of all time. By cunning and strategy, you can bring Hitler to his final doom and gain your escape from the fiery Pit.
Lucifer's Realm 16K cassette or diskette . $19.95

The Paradise Threat: Successfully emerging from Lucifer's Realm, you find that even heaven is not immune from harm. Your own previous actions in Hell have lead to a threat within Heaven itself. Winston Churchill, Abe Lincoln, and Groucho Marx guide you to ways to remedy your mistakes and save Paradise.
Paradise Threat 16K cassette or diskette . $19.95

Special Game Disk
(all three Pearson Adventures) $49.95

Med Systems Software • P.O. Box 3558 • Chapel Hill, NC 27514
To order, call 1-800-334-5470, or see your dealer

GRAPHIC VIOLENCE!

16K CASSETTE OR DISK

by Tom Hudson

When writing game programs, many programmers automatically choose assembly language over BASIC because of the obvious speed advantage. This can sometimes be a mistake, since BASIC offers some functions (such as sine, square root, etc.) not easily written in assembler. One way to take advantage of the convenience of BASIC and the speed of assembler is to combine the two languages. ATARI BASIC allows the user to "call" machine-language subroutines, which can be many times faster than the same routine in BASIC. A.N.A.L.O.G. Software's shooting gallery game "Carnival" uses this technique. Fine screen scrolling, music generation and other functions which must be performed quickly are done in assembler, while simpler operations are handled by BASIC.

In order to assist those game programmers who would like to have dramatic explosion effects in their BASIC programs, I have developed Graphic Violence, a group of assembly-language subroutines. These routines allow BASIC to generate up to 20 simultaneous explosions in GRAPHICS 7. They can optionally generate sound effects as well as "cycle" the colors of the explosions for an interesting "radioactive glow" effect.

The first half of this article is a non-technical explanation of how to use Graphic Violence. The second half is an in-depth discussion of the actual assembly language code for those interested in the inner workings of the subroutines.

USING GRAPHIC VIOLENCE

Listing 1 is the BASIC language code necessary to set up the Graphic Violence subroutine. This code should be placed in any program that is to use the explosion generator. After typing this program in, SAVE it immediately, BEFORE RUNNING IT! The routine has some safeguards against typing errors in the DATA statements, but if it is executed with bad DATA, the system may crash and it will be necessary to re-type the program.

After the program is typed and SAVEd, RUN it. If it is typed correctly, the program will run for several seconds before anything happens. The screen colors will begin cycling quickly. If not, an error was made somewhere, and you should re-boot your system, load the SAVEd program, find the mistake, SAVE it and try again.

If a message such as "COORD1 ERR" occurs, you have made a mistake typing in the DATA statements. "COORD1 ERR" indicates that an error was made in the COORD1 DATA, "INIT ERR" is an error in the INITIALIZATION CODE, etc. Find the error, fix it and re-RUN the program.

Once the computer starts cycling colors, press SYSTEM RESET before doing anything else. Whenever operating any program using the Graphic Violence subroutine, you MUST use the SYSTEM RESET key to terminate the program. The subroutine automatically disables the BREAK key since typing commands in immediate mode while the subroutine is in operation will usually cause a system crash. Pressing SYSTEM RESET will correctly terminate the subroutine and avoid any problems.

At this point, you should have a correctly operating Graphic Violence initialization subroutine SAVEd on tape or disk.

PROGRAM 1 FLOW

Line 80 - GOSUBs to line 10000 to initialize the subroutine.
Line 10010 -Dimensions the strings needed by Graphic Violence and RESTOREs the DATA pointer.
Line 10020-10060 -READs DATA statements into the strings used by the routine.
Line 10080 -POKEs graphics PLOT values into Graphic Violence.
Line 10100 -Calls the machine-language initialization routine. It is of the form:

```
A=USR(ADR(INIT$),ADR(MAIN$),ADR(COORD1
$),ADR(COORD2$),COLOR,SOUND)
```

The COLOR value tells whether or not you want the color of the explosions to cycle. In the program listing, this value is set to 1, indicating that cycling is desired. If you do not want cycling, place a 0 here.

The SOUND value tells whether or not you want the routine to generate sounds with the explosions. In the listing it is a 1, indicating that we want sound. If sound is not desired, place a 0 here.

Line 10110 -This line simply returns from the subroutine to the main program.

A SHORT DEMONSTRATION OF GRAPHIC VIOLENCE

With listing #1 in your computer, *add listing #2 to* the original program and RUN it. This is a short demonstration routine which simply places an explosion at the center of the screen, then repeats.

By looking at this short routine, you will notice the USR call in line 220. This is the command which starts an explosion. Once the Graphic Violence machine-code subroutine is set up, this short operation is all you need to generate explosions.

Remember to stop the program by pressing SYSTEM RESET.

PROGRAM 2 FLOW

Line 190 - Set up a full-screen graphics mode 7.

Line 220 - Call the explosion-starting machine language routine. This line actually starts the explosion. It is of the form:

```
A=USR(ADR(EXPL$),X,Y)
```

X and Y are the screen coordinates of the center of the explosion. In the listing, X=80 and Y=48, placing the explosion at the center of the screen.

This statement is the heart of the Graphic Violence routine. Once this statement is executed, it starts off an explosion while BASIC continues with whatever it is doing. In addition, the explosion handler can operate up to 20 explosions simultaneously, while BASIC does its own processing!

Line 240 - This line is a simple delay loop which allows an explosion to dissipate before generating another.

Line 260 - This line goes to start a new explosion after the wait.

A LONGER DEMONSTRATION

In the previous example, we generated one explosion at the center of the screen, just to keep things simple. In this example, we will see how the Graphic Violence routine will handle up to 20 simultaneous explosions without the programmer having to worry about what's going on inside the explosion handler! All the programmer needs to do is send the explosion coordinates to the routine via the USR command and let the computer do the rest. (What could be simpler?)

With listing #1 in your computer, add listing #3 to the original program and RUN it. The program will fill up most of the screen with graphics, then start dropping "bombs" from the top of the screen. As they hit the graphics area, they will explode violently, "eating" away the graphics. As soon as one of the bombs falls off the bottom of the screen, an end message will be displayed and subsequently destroyed by a number of explosions. The program will run continuously and MUST be stopped by pressing SYSTEM RESET.

PROGRAM 3 FLOW

Line 190 - Sets up graphics mode 7 and sets COLOR #2 (the explosion color) to maximum brightness.

Line 210 - Fills up the bottom section of the screen with COLOR 1 graphics.

Line 230 - Makes sure any error will cause the program to continue at line 320 (the "THE END" routine). This TRAP statement will take effect when a bomb falls off the bottom of the screen.

Line 250 - Gets the X and Y coordinates where the bomb will start its drop.

Line 270 - Erases old bomb position (using COLOR 0) and increments Y position so that bomb will "fall" toward bottom of screen.

Line 290 - Uses the LOCATE command to see if the bomb has hit anything. If the bomb hits color 1, an explosion is started at the X and Y coordinates and a new bomb is randomized.

Line 310 - If no hit is detected, the bomb is plotted in color 2, the program waits a fraction of a second, then continues at line 270.

Line 330 - When a bomb falls off the bottom of the screen, the error is TRAPped here. At this time, the computer sets up a new graphics 7 screen, sets the explosion brightness, and selects COLOR 1.

Line 350 - This line RESTOREs the DATA pointer to line 400 (THE END shape data), then reads from and to plot data and draws the THE END message on the screen.

Line 370 - This line sets off 200 explosions, which destroy the THE END message. Note that the explosion USR call has random number functions for X and Y coordinates of the explosion center. There is also a 40 count delay after each explosion is started for a more interesting display.

Line 390 - After all explosions are generated, wait a few seconds and GOTO line 190 to re-run the demonstration continuously.

Line 410-430 - These lines contain PLOT data for the words "THE END". Each line in the letters is represented by 4 values, made up of 2 sets of X and Y coordinates, the line endpoints.

SUMMARY

The Graphic Violence explosion generator subroutine will operate in almost any game using graphics 7. Explosions overlapping the edges of the screen are automatically "clipped," but the program has minimal error-trapping. The user should take care to make sure that the coordinates supplied to the routine do not exceed the graphics 7 screen limits. The routine uses sound channel 1 when the sound generation option is requested. The Explosions use COLOR 3 (SETCOLOR 2), and will cycle the color only (not brightness) if color cycling is requested. Any program using the Graphic Violence routine must be terminated with SYSTEM RESET to avoid a system crash.

The following section contains a discussion of the assembly-language routines that make up Graphic Violence. This information is not necessary to use the subroutine, but may assist those interested in assembly language and the inner workings of the ATARI computers.

GENERAL BACKGROUND INFORMATION

The Graphic Violence subroutine is made up of three program segments and two data tables. These five modules work together to provide a machine-language explosion generator for BASIC.

The first assembly program (listing #4) is the Graphic Violence initialization subroutine. It is stored in the BASIC string variable INIT$. Its function is to accept the locations of the main program module and the explosion plot coordinates, start up the main module, and accept the color cycling and sound generation options.

Remember that this is the routine called in the BASIC statement:

```
A=USR(ADR(INIT$),ADR(MAIN$),ADR(COORD1
$),ADR(COORD2$),COLOR,SOUND)
```

PROGRAM 4 FLOW

Line 230 - This line arbitrarily sets the location counter to $6000. Since this routine will be fully relocatable and stored in a BASIC string, this address does not matter.

Line 240 - This PLA instruction pulls the first argument off of the stack. In a BASIC USR call, this argument is always the number of arguments passed to the machine language routine. We do not use it in this case, and it is discarded.

Line 250-270 - This section zeroes out the explosion ready flag and the explosion counter.

Line 280-330 - This section pulls the low and high bytes of the address of the main routine (ADR MAIN$), transfers them to the X and Y registers, then puts a 7 in the accumulator and jumps to the SETVBV subroutine. This tells the system that we are using a vertical blank interrupt. The 7 indicates that it is a "deferred" vertical blank routine, that is, it operates after the system's vertical blank operation.

Line 340-410 - This section pulls the low and high bytes of the two sets of plot coordinates (COORD1$ and COORD2$, 4 PLA's total) and stores them on page zero ($CB-$CE) for later use by the main module.

Line 420-440 - This section pulls the color cycle indicator (COLOR) from the stack. Since this is a one-byte indicator and the system sends a two-byte argument, the first byte (high byte) is discarded and the second is stored in CYCFLG.

Line 450-470 - This section is the same as lines 420-440, except that it stores the sound indicator (SOUND) in SNDFLG.

Line 480 - This RTS (Return from Subroutine) returns control to your BASIC program after the initialization is complete.

The second assembly language program (Listing #5) is the explosion start routine. It is called by the BASIC statement:

```
A=USR(ADR(EXPL$),X,Y)
```

This routine simply accepts the coordinates of the explosion from BASIC. If there are 20 explosions active, it will ignore the request, otherwise it will send the coordinates to the main module, which is executing in the deferred vertical blank.

PROGRAM 5 FLOW

Line 200 - Once again, this listing has its location counter set to $6000. It makes no difference, since this routine is fully relocatable.

Line 210 - As in the previous listing, this line discards the first item on the stack (the number of arguments passed to the assembly routine).

Line 220-240 - These lines check the variable EXPCNT to make sure the new explosion can be started. If there are less than 20, control is passed to EXPOK (explosion OK).

Line 250-290 - These lines are used if there are already 20 explosions. The remaining 4 bytes are pulled from the stack and discarded, and the program returns to BASIC. No explosion is generated.

Line 300-350 - In a manner similar to the COLOR and SOUND parameters in listing #4, this routine pulls the X and Y coordinates off of the stack and places the values in NEWX and NEWY for use by the main module.

Line 360-370 - This section places a 1 in the READY flag, which tells the main interrupt routine that a new explosion is ready to start.

Line 380 - This RTS instruction simply returns control to BASIC. In this way, the interrupt can start the explosion graphics while BASIC keeps running normally.

The third assembly language listing (LISTING #6) is the vertical blank interrupt routine, stored in MAIN$. It does all the color cycling, sound, and graphics for the explosions. Since it is an interrupt-driven program, it operates independently of BASIC, allowing BASIC to continue processing normally while the vertical blank does all the explosion work.

Since this program is stored in a BASIC string, any program editing or immediate mode operations in BASIC while the vertical blank routine is running will cause a system crash. This is due to the fact that BASIC moves its variables around in memory during editing of programs, and such movement of the interrupt routine will confuse the system. To help avoid such a problem, the Graphic Violence interrupt routine disables the break key, making it necessary to press SYSTEM RESET to stop program execution. This is only a partial solution, however, since if the programmer allows his program to end with the READY prompt, then enters a program line, the crash will still occur.

The interrupt routine performs several functions. First, it disables the BREAK key and cycles the color of playfield type 2 if necessary. Next it processes sound, if required, using sound channel 1. The last major function it performs is that of explosion graphics generation.

Each explosion graphic is made up of 89 separate pixels. The routine uses the specified centerpoint of each explosion and adds X and Y offset values, which are stored in the BASIC string variables COORD1$ and COORD2$. Each of the 89 pixels are first turned on, one pixel at a time, resulting in a "growing" appearance. After all 89 pixels are on, the routine turns off one pixel at a time, causing the explosion to dissipate. Each active explosion has a pixel either turned on or off each time the interrupt is performed. Since this happens 60 times a second, each explosion takes roughly 3 seconds to expand and dissipate ((89*2)/60). Explosions are independent of each other because of three tables. The X and Y coordinates of each explosion are stored in the XPOS and YPOS tables. The third table, CNT, holds the number of the pixel which will be turned on or off next for each explosion. This value ranges from 0 to 88 for "on" pixels, and 89 to 177 for "off" pixels. If the CNT value for an explosion exceeds 177, the explosion has dissipated completely and its values are removed from the explosion tables by a "repack" operation. That is, if explosion number 2 is finished, explosion 3 will move back to 2, 4 to 3, etc.

PROGRAM 6 FLOW

Line 500 - Clears decimal mode. This instruction is vital when writing subroutines for BASIC that do any binary arithmetic.

Line 510-540 - Disables the BREAK key by altering POKMSK and IRQEN, the interrupt request enable. This prevents the BREAK key from generating an interrupt.

Line 550-640 - Cycles colors if CYCFLG is not zero.

Line 650-770 - Processes explosion sound if SNDFLG is not zero.

Line 780-940 - Monitors the READY flag to see if there is a new explosion. If not, the program checks for any old explosions at MAIN. If there is a new explosion, the routine sets up the XPOS, YPOS and CNT tables with the new information.

Line 950 - Zeroes out COUNTR, the variable indicating which explosion is being processed.

Line 960-1000 - Increments the explosion counter. If the counter is greater than the current number of explosions active (EXPCNT), the routine jumps to XITVBV, the vertical blank exit vector. Otherwise control is passed to INDEX.

Line 1130-1350 - This section repacks the XPOS, YPOS and CNT tables to eliminate a "dead" explosion. It then branches back to RUNLP to handle the next explosion.

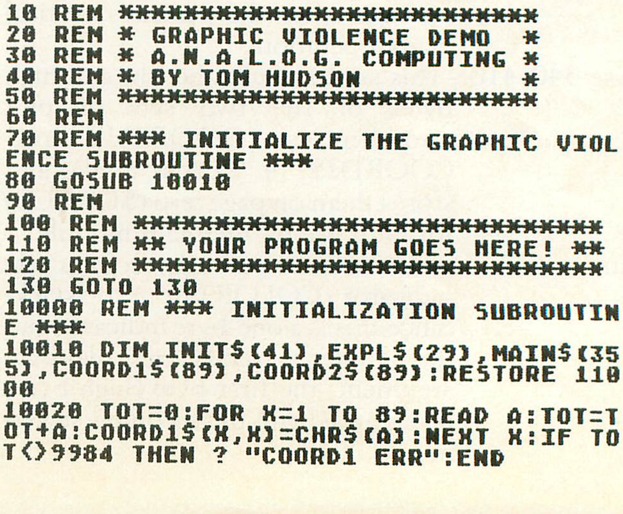
Line 1360-2350 - This routine turns explosion pixels on or off, depending on the PLOTCLR setting. If the pixel is off the screen, the plot is abandoned by a branch to RUNLP.

By expanding the XPOS, YPOS and CNT tables and altering the explosion call routine (Listing #5), advanced users can enable the Graphic Violence routine to handle many more explosions than it can now. However, 20 explosions are more than enough for most applications, and the routine should serve well as is.

I hope that ATARI programmers will see by this example that it is not always necessary to write game programs completely in assembly language. Just use BASIC for complicated functions difficult to write in assembler, and use assembler for things BASIC is too slow to do.

And by the way, have fun causing graphic havoc on your computer! □

LISTING 1



```
10 REM **************************
20 REM * GRAPHIC VIOLENCE DEMO  *
30 REM * A.N.A.L.O.G. COMPUTING *
40 REM * BY TOM HUDSON          *
50 REM **************************
60 REM
70 REM *** INITIALIZE THE GRAPHIC VIOL
ENCE SUBROUTINE ***
80 GOSUB 10010
90 REM
100 REM *****************************
110 REM ** YOUR PROGRAM GOES HERE! **
120 REM *****************************
130 GOTO 130
10000 REM *** INITIALIZATION SUBROUTIN
E ***
10010 DIM INIT$(41),EXPL$(29),MAIN$(35
5),COORD1$(89),COORD2$(89):RESTORE 110
00
10020 TOT=0:FOR X=1 TO 89:READ A:TOT=T
OT+A:COORD1$(X,X)=CHR$(A):NEXT X:IF TO
T<>9984 THEN ? "COORD1 ERR":END
```

```
10030 TOT=0:FOR X=1 TO 89:READ A:TOT=T
OT+A:COORD2$(X,X)=CHR$(A):NEXT X:IF TO
T<>9984 THEN ? "COORD2 ERR":END
10040 TOT=0:FOR X=1 TO 41:READ A:TOT=T
OT+A:INIT$(X,X)=CHR$(A):NEXT X:IF TOT<
>4237 THEN ? "INIT ERR":END
10050 TOT=0:FOR X=1 TO 29:READ A:TOT=T
OT+A:EXPL$(X,X)=CHR$(A):NEXT X:IF TOT<
>2198 THEN ? "EXPL ERR":END
10060 TOT=0:FOR X=1 TO 355:READ A:TOT=
TOT+A:MAIN$(X,X)=CHR$(A):NEXT X:IF TOT
<>36691 THEN ? "MAIN ERR":END
10070 REM *** SET UP PLOT BITS ***
10080 POKE 1568,192:POKE 1569,48:POKE
1570,12:POKE 1571,3
10090 REM *** INITIALIZE GRAPHIC VIOLE
NCE ROUTINE AND RETURN ***
10100 A=USR(ADR(INIT$),ADR(MAIN$),ADR(
COORD1$),ADR(COORD2$),1,1)
10110 RETURN
11000 REM *** COORD1 DATA ***
11010 DATA 0,1,255,0,255,0,255,2,1,1,0
,254,255,1,0,1,254,254,2,0,1,255,2,2,2
,255,254,1,253,3,3,4,252,253,254
11020 DATA 255,254,2,3,3,253,0,0,0,4,4
,252,255,2,0,3,2,1,253,254,254,252,253
,3,253,252,251,251,252,4,3,4,255
11030 DATA 5,5,5,253,1,254,0,255,252,2
53,251,253,252,3,4,3,1,255,1,2,4
12000 REM *** COORD2 DATA ***
12010 DATA 0,255,1,2,254,255,0,1,254,0
,1,0,255,1,253,253,2,255,255,254,2,3,2
,0,254,2,1,3,254,1,254,255,0,1,253
12020 DATA 253,254,3,2,0,3,252,4,3,0,2
,2,4,4,5,3,253,252,0,3,4,254,252,252,2
,1,1,0,255,254,255,1,251
12030 DATA 0,255,1,4,4,252,251,252,253
,253,255,255,3,253,253,4,251,5,5,252,3
13000 REM *** INITIALIZATION CODE ***
13010 DATA 104,169,0,141,0,6,141,1,6,1
04,170,104,168,169,7
13020 DATA 32,92,228,104,133,204,104,1
33,203,104,133,206,104,133,205
13030 DATA 104,104,141,11,6,104,104,14
1,12,6,96
14000 REM *** EXPLOSION CALL CODE ***
14010 DATA 104,173,1,6,201,20,48,5,104
,104,104,104,96,104,104
14020 DATA 141,2,6,104,104,141,3,6,169
,1,141,0,6,96
14990 REM *** MAIN INTERRUPT CODE ***
15000 DATA 216,165,16,41,127,133,16,14
1,14,210,173,11,6,240,20
15010 DATA 173,14,6,24,105,16,141,14,6
,173,198,2,41,15,13
15020 DATA 14,6,141,198,2,173,12,6,240
,22,173,13,6,240,17
15030 DATA 56,233,1,141,13,6,74,74,74,
141,1,210,169,40,141
15040 DATA 0,210,173,0,6,240,31,238,1,
6,174,1,6,173,2
15050 DATA 6,157,64,6,173,3,6,157,85,6
,169,127,141,13,6
15060 DATA 169,0,157,106,6,141,0,6,141
,5,6,238,5,6,173
15070 DATA 1,6,205,5,6,16,3,76,98,228,
174,5,6,169,0
15080 DATA 141,4,6,189,106,6,201,89,48
,51,238,4,6,56,233
15090 DATA 89,201,89,48,41,138,168,232
,236,1,6,240,2,16,21
15100 DATA 189,64,6,153,64,6,189,85,6,
153,85,6,189,106,6
15110 DATA 153,106,6,200,208,227,206,1
,6,206,5,6,169,0,240
15120 DATA 176,254,106,6,168,189,64,6,
24,113,203,141,6,6,201
15130 DATA 160,176,159,189,85,6,24,113
,205,141,7,6,201,96,176
15140 DATA 146,10,133,207,169,0,240,2,
240,137,133,208,165,207,10
15150 DATA 133,207,165,208,42,133,208,
165,207,10,133,207,141,9,6
15160 DATA 165,208,42,133,208,141,8,6,
165,207,10,133,207,165,208
15170 DATA 42,133,208,165,207,10,133,2
07,165,208,42,133,208,165,207
15180 DATA 24,109,9,6,133,207,165,208,
109,8,6,133,208,165,88
15190 DATA 24,101,207,133,207,165,89,1
01,208,133,208,173,6,6,41
15200 DATA 3,168,190,32,6,142,10,6,173
,6,6,74,74,24,101
15210 DATA 207,133,207,165,208,105,0,1
33,208,160,0,173,4,6,208
15220 DATA 11,173,10,6,81,207,145,207,
169,0,240,132,173,10,6
15230 DATA 73,255,49,207,145,207,169,0
,240,241
```

●

D: CHECK DATA

```
10 DATA 280,584,117,872,443,39,803,938
,42,503,948,583,773,754,453,8132
10020 DATA 663,814,280,321,554,442,562
,706,821,779,332,901,124,164,556,8019
12010 DATA 151,34,36,734,44,907,884,83
1,254,592,785,494,170,812,217,6945
15040 DATA 696,834,922,296,863,301,996
,27,550,398,634,887,664,954,370,9392
15190 DATA 817,737,509,518,917,3498
```

●

LISTING 2

```
130 REM ***************************
140 REM *  GRAPHIC VIOLENCE DEMO  *
150 REM *        NUMBER 1         *
160 REM ***************************
170 REM
180 REM *** SET UP GRAPHIC MODE 7 ***
190 GRAPHICS 7+16
200 REM *** SET OFF AN EXPLOSION ***
210 REM *** AT SCREEN CENTER     ***
220 A=USR(ADR(EXPL$),80,48)
230 REM *** WAIT A FEW SECONDS ***
240 FOR WAIT=1 TO 2000:NEXT WAIT
250 REM *** DO EXPLOSION AGAIN ***
260 GOTO 220
```

●

LISTING 3

```
130 REM ***************************
140 REM *  GRAPHIC VIOLENCE DEMO  *
150 REM *        NUMBER 2         *
160 REM ***************************
170 REM
180 REM *** SET UP GRAPHICS 7 FULL SCR
EEN AND EXPLOSION COLOR ***
190 GRAPHICS 7+16:SETCOLOR 2,15,15
200 REM *** DRAW THE 'GROUND' ***
210 COLOR 1:FOR Y=20 TO 95:PLOT 0,Y:DR
AWTO 159,Y:NEXT Y
220 REM *** TRAP ANY ERRORS TO 'THE EN
D' ROUTINE ***
230 TRAP 320
240 REM *** RANDOMIZE START POINT FOR
DROPPING BOMBS ***
250 X=5+RND(0)*149:Y=RND(0)*3
260 REM *** ADVANCE THE BOMB AS IT DRO
PS ***
270 COLOR 0:PLOT X,Y:Y=Y+3
280 REM *** IF THE BOMB HITS COLOR 1,
SET OFF EXPLOSION ***
290 LOCATE X,Y,Z:IF Z=1 THEN A=USR(ADR
(EXPL$),X,Y):GOTO 250
300 REM *** NO HIT, CONTINUE DROP ***
310 COLOR 2:PLOT X,Y:FOR DELAY=1 TO 10
:NEXT DELAY:GOTO 270
320 REM *** 'THE END' ***
330 GRAPHICS 7+16:SETCOLOR 2,15,15:COL
OR 1
340 REM *** PLOT 'THE END' ***
350 RESTORE 400:FOR X=1 TO 22:READ FRX
,FRY,TUX,TUY:PLOT FRX,FRY:DRAWTO TUX,T
UY:NEXT X
360 REM *** SET OFF 200 RANDOM EXPLOSI
ONS ***
370 FOR EXPL=1 TO 200:A=USR(ADR(EXPL$)
,40+RND(0)*75,20+RND(0)*55):FOR DELAY=
1 TO 40:NEXT DELAY:NEXT EXPL
```

```
380 REM *** LET EXPLOSIONS DIE, THEN R
E-RUN THE DEMO ***
390 FOR DELAY=1 TO 2000:NEXT DELAY:GOT
O 190
400 REM *** 'THE END' DATA ***
410 DATA 50,25,67,25,59,25,59,45,72,25
,72,45,72,35,88,35,88,25,88,45,93,25,9
3,45,93,25,109,25,93,35,109,35
420 DATA 93,45,109,45,50,50,50,70,50,5
0,67,50,50,60,67,60,50,70,67,70,72,70,
72,50,72,50,88,70,88,70,88,50
430 DATA 93,50,93,70,93,50,102,50,102,
50,109,56,109,56,109,64,109,64,102,70,
102,70,93,70
```

●

LISTING 4

```
0100 ; GRAPHIC VIOLENCE
0110 ;
0120 ; A.N.A.L.O.G. COMPUTING #8
0130 ;
0140 ; INITIALIZATION CODE
0150 ;
0160 READY = $600
0170 EXPCNT = $601
0180 CYCFLG = $60B
0190 SNDFLG = $60C
0200 COORD1 = $CB
0210 COORD2 = $CD
0220 SETVBV = $E45C
0230  *= $6000
0240 INIT PLA              ;DISCARD
0250  LDA #0               ;ZERO OUT:
0260  STA READY            ;READY FLAG
0270  STA EXPCNT           ;# OF EXPL.
0280  PLA                  ;INTERRUPT HI
0290  TAX                  ;PUT IN X
0300  PLA                  ;INTERRUPT LO
0310  TAY                  ;PUT IN Y
0320  LDA #7               ;DEFERRED VBI
0330  JSR SETVBV           ;SET IT!
0340  PLA                  ;COORD1 HI
0350  STA COORD1+1         ;SAVE IT
0360  PLA                  ;PULL COORD1 LO
0370  STA COORD1           ;SAVE IT
0380  PLA                  ;PULL COORD2 HI
0390  STA COORD2+1         ;SAVE IT
0400  PLA                  ;PULL COORD2 LO
0410  STA COORD2           ;SAVE IT
0420  PLA                  ;DISCARD
0430  PLA                  ;PULL COLOR CYCL
E FLAG
0440  STA CYCFLG           ;SAVE IT
0450  PLA                  ;DISCARD
0460  PLA                  ;PULL SOUND FLG
0470  STA SNDFLG           ;SAVE IT
0480  RTS                  ;FINISHED!
0490  .END
```

●

LISTING 5

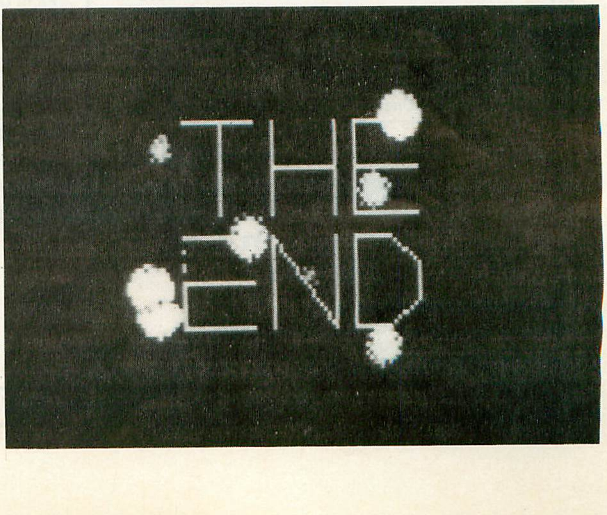
```
0100 ; GRAPHIC VIOLENCE
0110 ;
0120 ; A.N.A.L.O.G. COMPUTING #8
0130 ;
0140 ; EXPLOSION CALL ROUTINE
0150 ;
0160 READY = $600
0170 EXPCNT = $601
0180 NEWX = $602
0190 NEWY = $603
0200  *=$6000
0210  PLA                  ;DISCARD
0220  LDA EXPCNT           ;# OF EXPL.
0230  CMP #20              ;20 ACTIVE?
0240  BMI EXPOK            ;NO, IT'S OK!
0250  PLA                  ;YES, DISCARD
0260  PLA                  ;BOTH COORDS
0270  PLA
0280  PLA
0290  RTS                  ;AND EXIT
0300 EXPOK PLA             ;DISCARD HIGH
0310  PLA                  ;GET X-COORD
0320  STA NEWX             ;STORE IT
0330  PLA                  ;DISCARD HIGH
0340  PLA                  ;GET Y-COORD
0350  STA NEWY             ;STORE IT
0360  LDA #1               ;TELL INTERRUPT
0370  STA READY            ;WE'RE READY!
0380  RTS                  ;AND EXIT BACK
                            TO BASIC!
0390 ;
0400  .END
```

●

LISTING 6

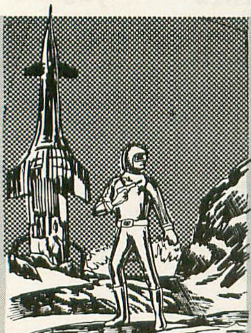
```
0100 ; GRAPHIC VIOLENCE
0110 ;
0120 ; A.N.A.L.O.G. COMPUTING #8
0130 ;
0140 ; VBLANK INTERRUPT ROUTINE
0150 ;
0160 READY = $600
0170 EXPCNT = $601
0180 NEWX = $602
0190 NEWY = $603
0200 PLOTCLR = $604
0210 COUNTR = $605
0220 PLOTX = $606
0230 PLOTY = $607
0240 HIHLD = $608
0250 LOHLD = $609
0260 PLOTBYT = $60A
0270 CYCFLG = $60B
0280 SNDFLG = $60C
0290 SNDCNT = $60D
0300 COLOR = $60E
0310 PLOTBL = $620
0320 XPOS = $640
0330 YPOS = XPOS+21
0340 CNT = YPOS+21
0350 LO = $CF
0360 HI = $D0
0370 COORD1 = $CB
0380 COORD2 = $CD
0390 ;
0400 ;SYSTEM EQUATES
0410 ;
0420 XITVBV = $E462
0430 COLPF2 = $2C6
0440 AUDC1 = $D201
0450 AUDF1 = $D200
0460 SAVMSC = $58
0470 POKMSK = $10
0480 IRQEN = $D20E
0490  *=$6000
0500  CLD                  ;CLEAR DECIMAL
0510  LDA POKMSK           ;GET IRQ INT.
0520  AND #$7F             ;NO BREAK KEY
0530  STA POKMSK           ;THE BREAK KEY
0540  STA IRQEN            ;IS NOW OFF!
0550  LDA CYCFLG           ;CYCLING COLOR?
0560  BEQ CONT             ;NO, CONTINUE
0570  LDA COLOR            ;GET LAST COLOR
0580  CLC                  ;INCREMENT IT
0590  ADC #16              ;BY 16
0600  STA COLOR            ;AND SAVE IT
0610  LDA COLPF2           ;GET COLOR REG.
0620  AND #$0F             ;GET BRIGHTNESS
0630  ORA COLOR            ;ADD THE COLOR
0640  STA COLPF2           ;AND SAVE IT!
0650 CONT LDA SNDFLG       ;SOUND ON?
0660  BEQ GO               ;NO, SKIP IT!
0670  LDA SNDCNT           ;MORE SOUND?
0680  BEQ GO               ;NO, SKIP IT!
0690  SEC                  ;DECREMENT THE
0700  SBC #1               ;SOUND COUNTER
0710  STA SNDCNT           ;AND STORE IT
0720  LSR A                ;SHIFT DOWN TO
0730  LSR A                ;DERIVE VOLUME
0740  LSR A                ;FROM COUNTER
0750  STA AUDC1            ;SET UP SOUND
0760  LDA #40              ;CHANNEL 1...
0770  STA AUDF1            ;FINISHED!
0780 GO LDA READY          ;NEW EXPLOSION?
0790  BEQ MAIN             ;NO, CONTINUE
0800 ;
0810 ;AT THIS POINT, THERE IS A
0820 ;NEW EXPLOSION!
0830 ;
0840  INC EXPCNT           ;ONE MORE EXPL
```

```
0850  LDX EXPCNT      ;PUT IN INDEX
0860  LDA NEWX        ;GET X-COORD,
0870  STA XPOS,X      ;PUT IN TABLE
0880  LDA NEWY        ;GET Y-COORD,
0890  STA YPOS,X      ;PUT IN TABLE
0900  LDA #127        ;INITIALIZE THE
0910  STA SNDCNT      ;SOUND COUNTER
0920  LDA #0          ;INIT COUNTER
0930  STA CNT,X       ;FOR EXPL IMAGE
0940  STA READY       ;AND READY FLAG
0950 MAIN STA COUNTR  ;ZERO COUNTER
0960 RUNLP INC COUNTR ;NEXT EXPLOSION
0970  LDA EXPCNT      ;GET # OF EXPL.
0980  CMP COUNTR      ;ANY MORE EXPL?
0990  BPL INDEX       ;YES, CONTINUE
1000  JMP XITVBV
1010 INDEX LDX COUNTR ;GET INDEX
1020  LDA #0          ;SET PLOTCLR
1030  STA PLOTCLR     ;0=PLOT A BLOCK
1040  LDA CNT,X       ;GET COUNTER
1050 ;                 FOR EXPLOSION
1060  CMP #89         ;ALL DRAWN?
1070  BMI DOPLOT      ;NO, DO IT NOW
1080  INC PLOTCLR     ;1=ERASE BLOCK
1090  SEC             ;GET READY FOR
1100  SBC #89         ;ERASE CYCLE
1110  CMP #89         ;ERASE DONE?
1120  BMI DOPLOT      ;NO,ERASE BLOCK
1130  TXA             ;MOVE INDEX
1140  TAY             ;TO Y REGISTER
1150 ;
1160 ;THE FOLLOWING ROUTINE REPACKS
1170 ;THE EXPLOSION TABLE TO GET RID
1180 ;OF EXPLOSIONS THAT ARE DONE.
1190 ;
1200 REPACK INX       ;NEXT EXPLOSION
1210  CPX EXPCNT      ;DONE?
1220  BEQ RPK2        ;NO,REPACK MORE
1230  BPL RPKEND      ;YES, EXIT!
1240 RPK2 LDA XPOS,X  ;NO, START RPK
1250  STA XPOS,Y      ;MOVE BACK X
1260  LDA YPOS,X
1270  STA YPOS,Y      ;MOVE BACK Y
1280  LDA CNT,X
1290  STA CNT,Y       ;MOVE BACK CNT
1300  INY
1310  BNE REPACK      ;NEXT REPACK
1320 RPKEND DEC EXPCNT ;DEC POINTERS
1330  DEC COUNTR      ;DUE TO REPACK
1340  LDA #0          ;FORCE BRANCH
1350  BEQ RUNLP       ;TO NEXT EXPL.
1360 DOPLOT INC CNT,X ;INC COUNTER
1370  TAY             ;EXP PHASE IN Y
1380  LDA XPOS,X      ;GET X-COORD
1390  CLC
1400  ADC (COORD1),Y  ;ADD X OFFSET
1410  STA PLOTX       ;STORE IT
1420  CMP #160        ;OFF SCREEN?
1430  BCS RUNLP       ;YES,DON'T PLOT
1440  LDA YPOS,X      ;GET Y-COORD
1450  CLC
1460  ADC (COORD2),Y  ;ADD Y OFFSET
1470  STA PLOTY       ;STORE IT
1480  CMP #96         ;OFF SCREEN?
1490  BCS RUNLP       ;YES,DON'T PLOT
1500 ;
1510 ;THE FOLLOWING SECTION IS A
1520 ;DEDICATED MULTIPLY ROUTINE
1530 ;WHICH MULTIPLIES THE A REGISTER
1540 ;BY 40, WITH RESULT IN LO & HI
1550 ;
1560   ASL A
1570   STA LO
1580   LDA #0
1590   BEQ X2
1600 JRUNLP BEQ RUNLP
1610 X2 STA HI          ;*2
1620   LDA LO
1630   ASL A
1640   STA LO
1650   LDA HI
1660   ROL A
1670   STA HI           ;*4
1680   LDA LO
1690   ASL A
1700   STA LO
1710   STA LOHLD
1720   LDA HI
1730   ROL A
1740   STA HI
1750   STA HIHLD        ;*8
1760   LDA LO
1770   ASL A
1780   STA LO
1790   LDA HI
1800   ROL A
1810   STA HI           ;*16
1820   LDA LO
1830   ASL A
1840   STA LO
1850   LDA HI
1860   ROL A
1870   STA HI           ;*32
1880   LDA LO
1890   CLC
1900   ADC LOHLD
1910   STA LO
1920   LDA HI
1930   ADC HIHLD
1940   STA HI           ;+*8=*40
1950 ;
1960 ;AT THIS POINT, THE MULTIPLY BY
1970 ;40 IS FINISHED, AND WE NEED TO
1980 ;GET AN OFFSET INTO THE SCREEN
1990 ;MEMORY
2000 ;
2010   LDA SAVMSC       ;ADD THE DISPLAY
2020   CLC              ;ADDRESS TO GET
2030   ADC LO           ;THE ACTUAL
2040   STA LO           ;ADDRESS OF THE
2050   LDA SAVMSC+1     ;BYTE THAT WILL
2060   ADC HI           ;BE ALTERED FOR
2070   STA HI           ;THE PLOT.
2080   LDA PLOTX        ;MASK PLOTX FOR
2090   AND #3           ;THE PLOT AND
2100   TAY              ;PLACE IN Y REG
2110   LDX PLOTBL,Y     ;GET PLOT BITS,
2120   STX PLOTBYT      ;AND SAVE!
2130   LDA PLOTX        ;GET PLOTX AND
2140   LSR A            ;DIVIDE
2150   LSR A            ;BY 4
2160   CLC              ;AND ADD TO
2170   ADC LO           ;PLOT ADDRESS
2180   STA LO           ;FOR FINAL PLOT
2190   LDA HI           ;ADDRESS.
2200   ADC #0
2210   STA HI
2220   LDY #0           ;ZERO OUT Y REG.
2230   LDA PLOTCLR      ;ERASING?
2240   BNE CLEARIT      ;YES,GO CLEAR IT
2250   LDA PLOTBYT      ;GET PLOT BITS,
2260   EOR (LO),Y       ;ALTER DISPLAY,
2270   STA (LO),Y       ;AND PLOT IT!
2280   LDA #0           ;FORCE BRANCH
2290 JRUNLP2 BEQ JRUNLP ;AND EXIT!
2300 CLEARIT LDA PLOTBYT ;PLOT BITS
2310   EOR #$FF         ;FLIP 'EM
2320   AND (LO),Y       ;ALTER DISPLAY
2330   STA (LO),Y       ;AND ERASE IT!
2340   LDA #0           ;FORCE BRANCH
2350   BEQ JRUNLP2      ;AND EXIT!
2360   .END
```



Great Autumn Offerings

the PROGRAM STORE



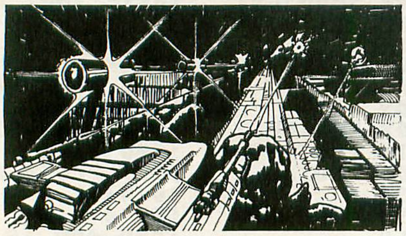
PROBE ONE

From Synergistic Software

High-res action game. You're the commander of the Terran space ship PROBE ONE during the war between the Terran Confederation and the Drelgan Hegemony. Only you can save the human race from extinction by the Drelgans...if you can find the matter transmitter and bring it to Terra. You'll need a quick mind and fast reflexes to fight off the guard robots as you search. Multiple skill levels. BASIC required; paddles or joystick.

40K Tape, ~~$34.95~~ Save 15%!

Now Thru Oct. 15 You Pay Just $29.71



CAVERNS OF MARS

By Greg Christensen from APX

An award-winning arcade game that will challenge your skill for many, many plays. You command the fighter Helicon VII on a mission to invade the Martian high command. The headquarters are located deep in the caverns, so you must descend through radar, rockets, laser gates, and space mines to reach your destination. Once there, you activate the fusion bomb and try to get out before it detonates. Good luck! Joystick required.

16K Tape, $34.95 (WHILE THEY LAST!!

16K Disk, $37.95

DE RE ATARI

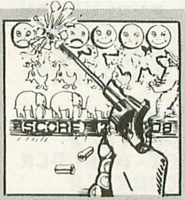
From APX

Translated from Latin, the title of this book is "All About Atari" and it means what it says! Used in combination with Atari's Technical Reference Manual, advanced programmers will be able to learn to exploit the many hardware and operating system features that make the Atari 400/800 so tremendously versatile. Includes a useful discussion of the new GTIA chip. Once you know Atari BASIC and assembler, this book is a must.

Loose leaf (binder not supplied), $19.95

From DataSoft Inc.

Games, Graphics, Word Processing Programs



SHOOTING ARCADE

From DataSoft

Step right up, ladies and gentlemen, and try your luck! It's your favorite amusement park game right in your own home! Aim at the elephants, ducks, bunnies and more. Every time you hit the target you score. But aim carefully—don't waste bullets. Fire at a frowning face and your bullets turn to blanks; shoot the smiling face and win BIG! If you make it to the 10th round, there's a ***surprise.*** Requires joystick.

16K Disk, ~~$29.95~~ Save 15%!

Now Thru Oct. 15 You Pay Just $25.46

CLOWNS AND BALLOONS

From DataSoft by Frank Cohen

A huge bunch of balloons is stuck at the top of the circus tent. How do you get them down? By bouncing a clown on a trampoline, high enough to burst them, of course! You must break the balloons in order, one row at a time—if you miss any, the whole row fills up again. Aim your clowns where you want them by moving the trampoline to just the right position. 1 or 2 players; multi-skill levels; joystick or paddle required.

16K Tape or Disk, $29.95

CANYON CLIMBER

From DataSoft by Tim Ferris

You're at the bottom of the Grand Canyon, trying to scale the world's toughest wall. If that's not enough of a problem, there are three challenges to face along the way: angry sheep, threatening Indians and attacking birds. Action-packed arcade game you'll really enjoy. Requires joystick.

16K Tape or Disk, $29.95

SPELL WIZARD

From DataSoft

SPELL WIZARD is a proof-reading package, compatible with Atari's word processor and Text Wizard. It has a dictionary of 20,000 words; you can add up to 5,000 more. Insert your letter, term paper or the latest chapter of your novel, and SPELL WIZARD will proofread it for you. Your copy will be perfect.

32K Disk, $79.95

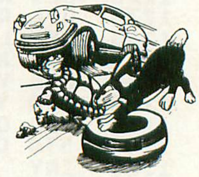
MICROPAINTER

From DataSoft by Bob Bishop

Create your own color graphics with this "electronic coloring book" for all ages and abilities. The special features include: draw mode, diamond fill, microscope mode for "close-up" viewing, inversing, rubberband line draw, and joystick-controlled easel for selecting pattern, colors and intensities.

48K Disk, ~~$34.95~~ Save 15%!

Now Thru Oct. 15 You Pay Just $29.71



PACIFIC COAST HIGHWAY

From DataSoft by Ron Rosen

Fast-action game with good graphics for 1 or 2 players. You're a tortoise, trying to cross the Highway without getting hit by the rush hour traffic. Once across, you're at the shore, and must leap from boats to rafts on the Pacific Ocean. Be careful 'cause you can't swim! Requires joystick.

16K Tape or Disk, $29.95

TEXT WIZARD 2.0

From DataSoft

This is the newest version of the Atari word processing package TextWizard, so popular with writers, students annd small business people. Like the original, you can learn and use TextWizard 2.0 in about an hour; write copy, then edit; underline, insert or replace words or paragraphs (in one place or **throughout** text); print in 1 or 2-column format; get multiple copies, print in boldface, elongated or condensed lettering with proportional spacing. It even warns you about errors ***before*** you make them!

TEXTWIZARD 2.0 now has file compatibility with File Manager 800 so you can change your text with information taken from File Manager files; address your form letters from File Manager's mailing lists, etc. It's also compatible with: EPSON MX-80 with Graftrax Plus ROMS; NEC 8023-A; Okidata and Pro-Writer. Instruction manual with 3-ring binder.

32K Disk $99.95

Over 1500 Programs for TRS-80, ATARI 400/800, APPLE & IBM.

For Information Call
202-363-9797

Visit our other stores:
Seven Corners Center, Falls Church, VA
W. Bell Plaza, 6600 Security Blvd., Baltimore MD
Coming Soon to Columbus, OH and Greensboro, NC

the PROGRAM STORE™

To Order Call Toll-Free
VISA **800-424-2738** MasterCard

MAIL ORDERS: Send check or M.O. for total purchase price, plus $1.00 postage & handling. D.C., MD. & VA.: add sales tax. Charge cards: include all embossed information on the card.

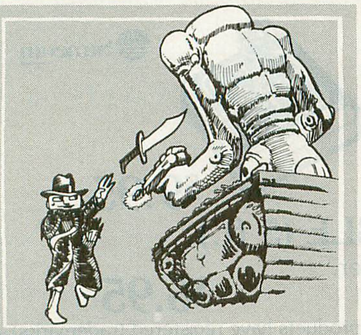
4200 Wisconsin Avenue, NW, Dept. A209 Box 9609 Washington, D.C. 20016

For Your ATARI™ 400/800

the PROGRAM STORE

From Synapse Software

Arcade-Type Action Games, Application Software



SHAMUS

From Synapse Software

It's the 21st century, and you're the SHAMUS, looking for your arch-enemy, the Shadow, to destroy him. You're armed with Ion-Shivs, the most powerful weapons in the galaxy. Can you handle the Shadow's henchmen: Robo-Droids, Whirling Drones and Snap-Jumpers, all armed and **evil**? Can you find the Shadow in his lair of 4 levels with 32 rooms each . . . every one of them dangerous? Intensive arcade action; requires joystick.

16K Tape or Disk, ~~$29.95~~ *Save 15%!*

Now Thru Oct. 15 You Pay Just $25.46

PROTECTOR

From Synapse Software

The planet's inhabitants are endangered by a malevolent alien that beams them to his ship and transports them to an active volcano. You must pick them up one-by-one with your rescue/attack ship and transport them to another city while dodging lasers and rough terrain. After you save as many as possible, the volcano erupts. You must then move each person to a volcano-proof vault in the mountains! Great graphics and sound in this arcade game. Requires joystick.

32K Tape or Disk, $29.95

FILE MANAGER 800

From Synapse Software

An extremely powerful and versatile database manager for use in both professional and personal applications. You define the format of the records to be filed and FILE MANAGER 800 gives you full control over sorting, searches, and retrieval.

You can store up to 1000 records on each disk and the program will allow up to four drives. Access to any record takes less than 1.5 seconds, and most commands can be entered with self-prompting single key strokes. Works with Text Wizard 2.0 to make editing easier.

40K *Disk*, $99.95

FORT APOCALYPSE

From Synapse Software

The Warlords of Kraltha have constructed a prison deep inside Earth—the terrifying FORT APOCALYPSE. All those who have tried to rescue their slave/captives have vanished without a trace. ***Now it's your turn!*** Can you descend thru the Kralthian disruptor fields and penetrate the vast underground Vaults of Draconis? Can your Rocket-Copter fight off the Wormlings, Servo-Tank Interceptors and Robo-Choppers? Multi-player game with arcade action; requires joysticks.

32K Tape or Disk, ~~$29.95~~ *Save 15%!*

Now Thru Oct. 15 You Pay Just $25.46

SLIME

SLIME

From Synapse by Mike Hales

Large drops of "Slime" fall from the sky onto your ships. If ***one drop*** hits your ship, it will sink. Use the triangular diverters to aim Slime into buckets on the sides of the screen. If Slime falls into the ocean, the level of the ocean rises. When the ocean reaches the top of the screen, the game is over, and your score is calculated. One-player game with excellent graphics.

16K Tape or Disk, $29.95 Requires joystick.

CHICKEN

From Synapse Software

There's trouble in the barnyard, and this machine-language, arcade game will challenge even the most experienced arcade player. You try to help Ma Hen save the eggs and chickens from the wily fox. The action gets faster and faster as eggs turn into chicks, feathers fly, chickens squawk, and all bedlam breaks loose. You'll really have to think fast to outwit this fox.

Requires paddles.

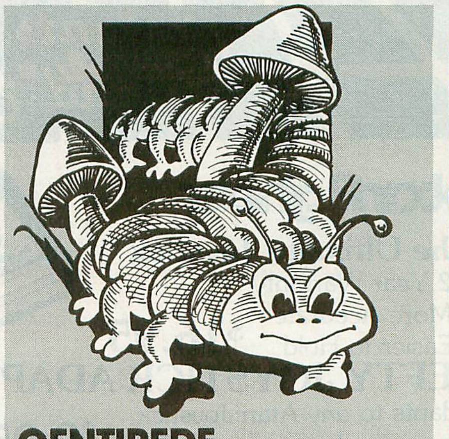
16K Tape, $29.95

NAUTILUS

From Synapse by Mike Potter

Two-player interaction combat game: the underwater cities and their destroyers against the helicopters and submarines. The helicopters bomb the ships; the subs fire torpedoes at the cities and surface to fire at the destroyers. The destroyers fire at the helicopters and drop depth charges on the subs. They can also lower men into the ocean to rebuild cities hit by torpedoes. Top and bottom halves of screen scroll separately for maximum field of play. Great graphics. Requires joystick.

16K Tape or Disk, $29.95



CENTIPEDE

From Atari

Very fast arcade game for the nimble-fingered. The green centipede is streaking across the screen, trying to get from the top to the bottom row. If you're fast enough, you can shoot it—but don't hit it in the ***wrong spot***. Gain points by killing the spiders and mushrooms, too. Requires joystick.

ROM Cartridge, ~~$42.95~~ *Save 15%!*

Now Thru Oct. 15 You Pay Just $36.51

FROGGER

From On Line Systems

Based on arcade game Frogger. Dodge traffic and jump the moving logs in the stream to get home. Fantastic graphics. Requires joystick.

16K Tape or 32K Disk, $34.95

TEMPLE OF APSHAI

From Automated Simulations

The first of the DujonQuest series, and still one of the most popular. In exploring over 200 rooms in the magical labyrinth, you will encounter more than 30 kinds of fearsome monsters guarding over 70 treasures. Some of the treasures will help you in your quest, but you must still watch out for the many monsters and traps that spring out from the walls and shadows.

32K Tape or Disk, $39.95

THE PROGRAM STORE • Dept. A209 • Box 9609 • 4200 Wisconsin Avenue, NW • Washington, D.C. 20016

Item Tape/Disk/Book Price

____

____

____

____

Postage $1.00

Total ______

□ CHECK □ VISA

□ MASTERCARD

Name ______________________

Address ______________________

City ________ State ________ Zip ________

Card# ______________ Exp ________

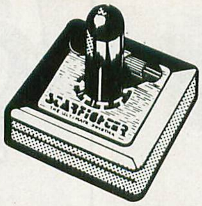
ComputAbility presents the Newest in Joysticks and Software Products

Starfighter™

The Ultimate Joystick

- 2 Year Warranty
- More Accurate
- Easier to Hold

$16.95



SLIK STIK™

- 90 Day Warranty
- Same as Starfighter except made of softer materials.

Suncom

$9.95

LEFTY JOYSTICK ADAPTOR

Adapts to *any* Atari Joystick.
Moves fire button to top Right. $9.95

EXTENSION CABLE (5 ft.)

Adapts to *any* Atari controller.

$6.95

ALL JOYSTICKS WORK WITH: Atari VCS, Sears Telegame, Commodore VIC-20, Atari 400/800 (All products have registered trademarks)

30 Day Money Back Guarantee

SUPER SPECIALS

CHOPLIFTER
$22.95 disk

RASTER BLASTER
$19.95 disk

K-RAZY SHOOTOUT
$31.95 cartridge

FROGGER
$24.50 disk/tape

CANYON CLIMBER
$19.95 disk/tape

Prices effective only through November 30, 1982.

Buy any 3 programs and receive a FREE* SLIK STIK

ATARI	
CONVERSATIONAL LANGUAGES - T	43.95
INVITATION TO PROGRAMMING 2 & 3 - T	21.95
MUSIC COMPOSER - C	32.95
MY FIRST ALPHABET - D	28.95
TOUCH TYPING - T	18.95
WORD PROCESSOR - D	112.95
BOOKKEEPER - D	112.95
HOME FILING MANAGER - D	37.95
MAILING LIST - T	18.95
ASTEROIDS - C	26.95
CAVERNS OF MARS - D	28.95
CENTIPEDE - C	32.95
COMPUTER CHESS - C	26.95
MISSILE COMMAND - C	26.95
PAC-MAN - C	32.95
SPACE INVADERS - C	26.95
SUPER BREAKOUT - C	26.95
STAR RAIDERS - C	32.95
ASSEMBLY EDITOR - C	44.95
BASIC - C	44.95
MACROASSEMBLER - D	65.95
MICROSOFT BASIC -D	65.95
PILOT (HOME PACKAGE) - C	58.95

APX

ALL ITEMS - 20% OFF RETAIL. WE CARRY THE ENTIRE APX CATALOG!

DATASOFT	
SHOOTING ARCADE - D/T	23.95
PACIFIC COAST HIGHWAY - D/T	23.95
CLOWNS & BALLOONS - D/T	23.95
TUMBLEBUGS - D	23.95
MICROPAINTER - D	27.95
BASIC COMPILER - D	74.95
TEXT WIZARD - D	74.95

ON-LINE	
JAWBREAKER - D/T	23.95
ULTIMA I - D/T	31.95
CROSSFIRE - D/T	23.95
ULYSSES & GOLDEN FLEECE - D	27.95
SOFTPORN ADVENTURE - D	23.95
THE NEXT STEP - D	31.95
MISSION: ASTEROIDS - D	20.95
THE WIZARD & THE PRINCESS - D	25.95
THRESHOLD - D	31.95
ULTIMA II - D	44.95
MOUSKATTACK - D	27.95

BRODERBUND	
APPLE PANIC - D/T	23.95
STELLAR SHUTTLE - D/T	23.95
DAVID'S MIDNIGHT MAGIC - D	27.95
STAR BLAZER - D	25.50
TRACK ATTACK - D	23.95
LABYRINTH - D	23.95
SERPENTINE - D	23.95
DUELING DIGITS - D	23.95
DEADLY SECRETS - D	27.95

SYNAPSE	
PROTECTOR - D/T	23.95
SLIME - D/T	23.95
CHICKEN - D/T	23.95
SHAMUS - D/T	23.95
FORT APOCALYPSE - D/T	23.95
FILE MANAGER 800+ - D	79.95
DISK MANAGER - D	23.95
PAGE 6 - D	23.95

ROKLAN	
DELUXE INVADERS - D	27.95
GORF-D/C	31.95
WIZARD OF WOR -D/C	31.95

ANALOG	
RACE IN SPACE-D/T	20.95
CARNIVAL-D/T	20.95
SUNDAY DRIVER-D/T	23.95
CRASH DIVE!-D/T	23.95

ADVENTURE INTERNATIONAL	
REAR GUARD - D	20.95
PREPPIE - D/T	23.95
S.A.G.A. ADVENTURES - D	23.95

AUTOMATED SIMULATIONS	
TEMPLE OF APSHAI - D/T	31.95
UPPER REACHES OF APSHAI - D/T	16.95
CRUSH, CRUMBLE & CHOMP - D/T	23.95
RICOCHET - D/T	16.95
CRYPT OF THE UNDEAD - D/T	23.95

I.D.S.I.	
POOL 1.5 - D	27.95
POOL 400 - C	31.95
SPEEDWAY BLAST - C	31.95

GEBELLI	
EMBARGO - C	39.95
FIREBIRD - C	39.95

SIRIUS	
SNAKE BYTE - D	23.95
SPACE EGGS - D	23.95
CYCLOD - D	23.95
SNEAKERS - D	23.95

QUALITY	
ALI BABA - D	26.50

THORN

CALL FOR ITEMS AND PRICES

D - Disk T - Cassette C - Cartridge

WE CARRY HUNDREDS OF ITEMS FOR ATARI 400/800, if you don't see it here, please call.

* COUPON PROGRAM

The purchase of each program (with the exception of Super Specials, Atari and APX) will earn you 1 COMPUTABILITY DIVIDEND COUPON. Save 3 coupons and redeem them for your choice of Slik Stik, a Lefty Adaptor, or an Extension Cable. You pay only a $2.50 shipping and handling charge.

ORDERING INFORMATION

To order by mail send money order, certified check or personal check (allow 14 days to clear) to COMPUTABILITY. Include $2.00 shipping on software orders and $2.50 shipping on hardware orders (FREE OR PURCHASED). Mastercard & VISA please include card number and expiration date. WI residents please add 5% sales tax. Outside of continental U.S.A. please add 15% shipping (U.S. Funds only).

Prices subject to change without notice.

Order Hours:
12 pm - 9 pm
C.S.T.

Computability
P.O. Box 17882
Milwaukee, WI
53209

Mastercard/VISA
Order Toll Free

800-558-0003

In Wisc. Call
414/351-2007

COLOR GRAPHICS IN MODE Ø

24K CASSETTE OR DISK

by Michael A. Ivins

In issue #5 we were introduced to the concepts and techniques for re-defining the character set and using it to create graphics. The re-defined character set is a powerful tool which can be used in many different ways. These can be used for special animation effects. They are especially useful when combined with certain types of modified display lists. Finally, they can be used to create colorful graphic displays in the text mode, GR.0. This last application is the subject of this article.

If you have ever done much playing around with GR.8 you know that getting four colors in this mode is not as difficult as you might expect. For the newcomers I include example program 1 to show what I mean.

```
10 GRAPHICS 8:SETCOLOR 2,0,15:SETCOLOR
 1,0,0:COLOR 1
20 FOR X=0 TO 200 STEP 2
30 PLOT X,0:DRAWTO X,10
40 NEXT X
50 FOR X=1 TO 201 STEP 2
60 PLOT X,20:DRAWTO X,30
70 NEXT X
80 FOR X=0 TO 200
90 PLOT X,40:DRAWTO X,50
100 NEXT X
```

This may seem to have little to do with re-defined character sets, but bear with me, I'm coming to it. The example should show what appear to be three bars on a white background with blue at the top, red next and black at the bottom. The program was supposed to draw two sets of evenly spaced vertical lines and one solid bar, so what happened? You would expect the bottom bar to be black. The only differences between the top two bars is in the positioning of the vertical lines, yet we get the two colors.

This effect is due to a curious property of the graphics screen whose technical name is "artifacting." Simply stated, the principle is that a single pixel of GR.8 (the smallest the ATARI will generate) will be one color while another pixel one space or any odd number of spaces away will have a different color. Put these two dots side by side and they will take on the designated foreground color. By now you are probably asking, "If this guy wants to talk about re-defined character sets, why all this stuff about colors in GR.8?" Every character has eight bytes associated with it, and the pattern made up by those bits which are ones determines the shape of the character. Two examples of this are shown in FIGURE ONE with one A showing the bit pattern of the letter "A" and one B showing a percent sign. Each pixel has a GR.0 character, whether it be text or control graphic, is identical to a single pixel of GR.8. By applying the same techniques which gave us colors in GR.8 to re-defining characters, we can get many kinds of colored graphic characters.

A	B
00000000	00000000
00011000	01100110
00111100	01101100
01100110	00011000
01100110	00110000
01111110	01100110
01100110	01000110
00000000	00000000

FIGURE ONE

There is one important factor which should be mentioned at this point. The colors you can get from your special characters (or a GR.8 display) will depend on the chosen background color and chosen luminosity of the foreground. For your own applications you should experiment with the combinations of foreground and background color which gives the effect you want most. For the purpose of this article and the game program which accompanies it I use a white background (SETCOLOR 2,0,15) and a black foreground (SETCOLOR 1,0,0).

I give two examples in figures two and three. For greater ease of use I have enlarged the pattern of bits so you can see them better than in the previous example. I have also labeled the values of the bits and given the decimal values that you would poke into the character table to make the changes. With the specified colors, the character defined in figure two will give you a solid blue block while the one in Figure three will make a solid red block.

FIGURE TWO (BLUE BLOCK)

128	64	32	16	8	4	2	1	Decimal value
1	0	1	0	1	0	1	0	170
1	0	1	0	1	0	1	0	170
1	0	1	0	1	0	1	0	170
1	0	1	0	1	0	1	0	170
1	0	1	0	1	0	1	0	170
1	0	1	0	1	0	1	0	170
1	0	1	0	1	0	1	0	170
1	0	1	0	1	0	1	0	170

FIGURE THREE (RED BLOCK)

128	64	32	16	8	4	2	1	Decimal value
0	1	0	1	0	1	0	1	85
0	1	0	1	0	1	0	1	85
0	1	0	1	0	1	0	1	85
0	1	0	1	0	1	0	1	85
0	1	0	1	0	1	0	1	85
0	1	0	1	0	1	0	1	85
0	1	0	1	0	1	0	1	85
0	1	0	1	0	1	0	1	85

Simple red and blue blocks alone make for rather dull graphics, but I'm sure you can see that by clever arranging of the dots you can create many interesting shapes. If the shape you want is too large to fit into a single 8x8 grid then use two, three or even more characters. To give one example of the kinds of things that can be done with color graphics characters and hopefully have a little fun at the same time, I include my program for Color Slot Machine.

Before getting into a description of the game itself there is a comment I would like to make. Calculating out all the numbers for special characters you have drawn on graph paper is very slow work and it tends to be boring. Fortunately this is the sort of task which lends itself to being "computerized." There are, in fact, many character editor programs on the market as well as some which have been published in magazines. These all allow you to make changes in an enlarged matrix and see the effects of these changes on the normal sized character. They let you save the special character set or "font" for use with your own programs. I used such a program which went as far as writing the actual subroutine that does the work in the Slot Machine game. If the hint isn't obvious yet, why doesn't some enterprising and ingenious programmer out there write up a good character editor program and send it in to help the readers of A.N.A.L.O.G.? You would be helping the other readers and making money for your trouble.

THE GAME

After the title display, the program will draw a colorful slot machine on the screen making use of several kinds of colored graphics characters. There are two ways to play which you choose by pressing the OPTION button anytime there is no bet placed. For those who might be unfamiliar with slot machines I will describe the options. Single line play uses only those symbols which line up in the center of the pay windows. In this version additional "coins" bet give bigger payouts when a winning combination comes up. The five line version gives more ways to win by adding top, bottom and diagonal paylines for the number of coins played. Single line play can pay more when it pays but the five line version can give more ways to win so you win more often.

Playing the game itself is simple. To enter a single coin bet, press the trigger button of the joystick and release. If you wish to bet the maximum bet of five, simply hold the trigger button down until the beeps stop. In the single line version the paychart changes to reflect payout for the size of the bet, while the five line version employs line pointers to indicate how many lines are in play. When you have made your desired bet, move the joystick in any direction to spin the reels. More details will be given in the program documentation. Happy gambling! □

PROGRAM DOCUMENTATION

The first thing the program does is to jump to the routine which alters the character set and since that is the main thing I wish to illustrate, I will cover this first.

32000 The first step resets RAMTOP. Next a graphics command to set the new top of memory. Now we poke the location of the new character set.

32005 This defines a machine language routine which will copy the old character set out of ROM into the protected area of RAM.

32010-32015 These lines are here to give you something to look at while the character set initializes. You won't see anything at this point since the area pointed to by CHBAS (location 764) is blank.

32020 This executes the machine language routine so that the material printed in the previous lines can now be seen on the screen.

32030-32040 Now we make the actual changes. We first read the number that tells where to start and then put in the new numbers. Some of the characters look a bit funny (like a cluster of cherries with a blue leaf or a purple bell) but this is the best I could do with these colors.

Now we return to the main program and from this point I will take things in the sequence they are shown in the program.

10-11 Here we dimension variables and asign other variables to frequently used subroutines.

15-25 These set up the reels of 30 "symbols" on each. If you wished to change the odds of the game, this is the place to do it. You could make it harder to win by changing the symbols or by adding no pay symbols or blanks. If you wanted to, you could set up the reels so that you would win on every play, which I could consider to be boring.

30 Jumps to the routine which draws the machine.

40-100 Here we set the initial values for game counters and display them. This also lets you know you are playing the one line version.

120 This displays the betting prompt.

125 If the bet is the maximum or the bankrool is zero then the betting routine is skipped.

130 For trigger press increment bet, decrement bankroll and start sound. Also gosub to change the

paychart.

135 Jump to the five line version if OPTION is pressed and bet is zero.

145 Erase play prompt if bet reaches maximum.

150 A delay to slow betting.

160-168 Display bet, shut off sound, display bet.

170-175 Return to betting loop if stick not moved or if stick moved but bet zero.

180 Zero out the attract mode.

290-310 Jumps to the routine that animate the handle and spin the reels.

311-315 Reads the symbols on the payline and jumps to payroutine.

320-327 Calculates the proper length of windsound and jumps to that routine.

330-340 Resets bet to zero and if any money is left you are returned to the betting routine.

350-420 This is the routine that is activated if you go broke. It resets the left margin, erases the paychart and then gives you your quit or start over options. Starting over redraws the machine and paychart and resets all values to beginning levels. Quitting naturally ends the game.

500-590 The functions here are similar to the betting loop of the single line version. The main difference is in setting line pointers instead of changing the paychart.

600-610 This reads all payable locations. Caution should be noted here. Be sure when you type these lines in that you use the abbreviation LOC. for LOCATE and POS or you won't get everything in on those lines.

620-676 This section checks for winning combinations and jumps to the payout routine if one is present. I originally tried to make this section more brief but kept getting errors.

680-698 Checks for win and goes to sound routine if appropriate.

700-720 Resets bet, erases line pointers and jumps back to betting routine if not bankrupt.

1000-1055 This creates the siren for winning and is a simple tone with rising and falling pitch.

1300-1360 This is the routine which resets the paychart according to the size of the bet in the one line play version.

2000-2090 This is the routine which actually draws the machine. Notation should be made here that the paychart is not truly complete. Most combinations will pay with a bar (single or double) on the last reel. I did not have room to fit this on the screen.

2300-2390 This animates the handle by first erasing the knob and redrawing it lower and doing the reverse by drawing a section of the handle and the knob one space higher. This routine also clears any old wins.

2400-2495 This is the payout routine for the single line version.

2600-2690 This is the super jackpot routine and is triggered if three of the "seven" symbols appear on the center line with maximum bet in the one line version and on the fifth line with maximum bet in the five line version. This has first an explosion sound, a slower siren than the regular windsound followed by another explosion. Then the words "SUPER JACKPOT!!!" are flashed. I suggest that you type in "GOSUB 2600" from the direct mode as you won't be seeing much of this routine unless you change the odds.

2950-3180 This animates the spin of the reels. I had originally tried to make the reels turn two full spins plus a random bit extra, but this slowed down the action of the game too much, doing it from BASIC. Therefore we just assume those spins without showing them and take a certain number plus a random amount. From this point the reels are moved visibly by a fixed number of spaces for each reel.

4000-4080 This is the pay routine for the five line version.

```
1 REM COLOR SLOT MACHINE
2 REM BY MICHAEL A. IVINS
4 GOSUB 32000
10 DIM L$(60),M$(60),R$(60),PAY$(9):OP
EN #2,12,0,"S:":WINSOUND=1000:FPAY=400
0:SPIN=2950
11 PAY$=" h▙▁-●●2H":CH=97:DB=146:SB=16
0
15 L$="abcdefgh██abghefcd▬ghcd██efijc
dghcdgh▬cdefcdcdabghefcdcdefgh"
20 M$="ab██efghefabghefcd▬efcdabefghi
jefabcdgh▁efabef██ghabcdefcd"
25 R$="ghcdef▬ghefghefcd▬cdcdef██cdi
jgh██cdgh▬cdcdefghef▬efefcd"
30 BANKROLL=100:BET=0:WIN=0:L=1:M=1:R=
1
40 GOSUB 2000:REM DRAW MACHINE
90 POSITION 20,20:? "1 LINE PLAY"
100 POSITION 20,22:PRINT "BANKROLL:";B
ANKROLL:POSITION 20,23:PRINT "BET:";BE
T;:POSITION 30,23:PRINT "WIN:";WIN;
120 IF BET<5 THEN POSITION 20,21:PRINT
 "PLAY 1 TO 5 COINS";
125 IF BANKROLL=0 OR BET=5 THEN 145
130 IF STRIG(0)=0 THEN BET=BET+1:GOSUB
 1300:SOUND 0,50,10,14:BANKROLL=BANKRO
LL-1
135 IF PEEK(53279)=3 AND BET=0 THEN 50
0
145 IF BET=5 THEN POSITION 20,21:? "
                 ";
150 FOR DELAY=1 TO 5:NEXT DELAY
160 POSITION 24,23:PRINT BET;
165 SOUND 0,0,0,0
168 POSITION 29,22:PRINT BANKROLL;" ";
170 IF STICK(0)=15 THEN 120
175 IF BET=0 THEN 120
180 POKE 77,0
290 GOSUB 2300:REM PULL HANDLE
310 GOSUB SPIN
311 LOCATE 5,8,LM:POSITION 5,8:? CHR$(
LM)
312 LOCATE 8,8,MM:POSITION 8,8:? CHR$(
MM)
313 LOCATE 11,8,RM:POSITION 11,8:? CHR
$(RM)
315 GOSUB 2400
320 IF WIN>0 AND WIN<BET*10 THEN DUR=2
:GOSUB WINSOUND
325 IF WIN>=BET*10 AND WIN<BET*25 THEN
 DUR=3:GOSUB WINSOUND
326 IF WIN>=BET*25 AND WIN<=BET*50 THE
N DUR=5:GOSUB WINSOUND
327 IF WIN>BET*50 AND WIN<2000 THEN DU
R=10:GOSUB WINSOUND
330 BET=0:POSITION 24,23:PRINT BET;" "
;
340 IF BANKROLL>0 THEN 120
350 POKE 82,20
360 FOR I=0 TO 23:POSITION 20,I:? "
                   "::NEXT I
```

```
370 POSITION 20,0:? "I'M SORRY":? "YOU
 HAVE GONE BROKE":? "IF YOU WISH TO BU
Y MORE":? "CHANGE PRESS START"
380 ? "PRESS SELECT IF YOU":? "WISH TO
 QUIT"
390 IF PEEK(53279)<>6 AND PEEK(53279)<
>5 THEN 390
400 IF PEEK(53279)=6 THEN POKE 82,2:GO
TO 11
420 POSITION 20,18:? "THANK YOU":? "FO
R PLAYING, BETTER":? "LUCK NEXT TIME":
END
500 POSITION 20,20:? "5 LINE PLAY";
510 BET=1:GOSUB 1300:BET=0
520 POSITION 20,22:? "BANKROLL:";BANKR
OLL;:POSITION 20,23:? "BET:";BET;:POSI
TION 30,23:? "WIN:";WIN;
530 IF BET<5 THEN POSITION 20,21:PRINT
 "PLAY 1 TO 5 COINS";
532 FOR DELAY=1 TO 5:NEXT DELAY
535 IF BANKROLL=0 OR BET=5 THEN 560
540 IF STRIG(0)=0 THEN BET=BET+1:BANKR
OLL=BANKROLL-1:SOUND 0,50,10,14
545 IF PEEK(53279)=3 AND BET=0 THEN 90
550 IF BET=1 THEN POSITION 4,8:PRINT "
■";
552 IF BET=2 THEN POSITION 4,6:PRINT "
■";
554 IF BET=3 THEN POSITION 4,10:PRINT
" ■";
556 IF BET=4 THEN POSITION 4,4:PRINT "
▁";
558 IF BET=5 THEN POSITION 4,12:PRINT
"▁";
560 POSITION 29,22:PRINT BANKROLL;" ";
:POSITION 24,23:PRINT BET;
562 IF BET=5 THEN POSITION 20,21:PRINT
 "                 "
565 FOR DELAY=1 TO 20:NEXT DELAY
566 SOUND 0,0,0,0
570 IF STICK(0)=15 THEN 530
575 IF BET=0 THEN 530
580 GOSUB 2300
590 GOSUB SPIN
600 LOCATE 5,8,LM:POSITION 5,8:? CHR$(
LM):LOCATE 8,8,MM:POSITION 8,8:? CHR$(
MM):LOCATE 11,8,RM:POSITION 11,8:? CHR
$(RM)
605 LOCATE 5,6,LT:POSITION 5,6:? CHR$(
LT):LOCATE 8,6,MT:POSITION 8,6:? CHR$(
MT):LOCATE 11,6,RT:POSITION 11,6:? CHR
$(RT)
610 LOCATE 5,10,LB:POSITION 5,10:? CHR
$(LB):LOCATE 8,10,MB:POSITION 8,10:? C
HR$(MB):LOCATE 11,10,RB:POSITION 11,10
:? CHR$(RB)
620 IF (LM=CH AND MM<>CH) OR (LM=CH AN
D MM=LM) THEN F=LM:S=MM:T=RM:GOSUB FPA
Y
621 IF LM=DB AND MM=SB AND (RM=DB OR R
M=SB) THEN F=LM:S=MM:T=RM:GOSUB FPAY
622 IF LM=MM AND RM=MM THEN F=LM:S=MM:
T=RM:GOSUB FPAY
623 IF LM=DB AND (MM=DB OR MM=SB) AND
RM=SB THEN F=LM:S=MM:T=RM:GOSUB FPAY
624 IF LM<>CH AND LM<>DB AND LM<>SB AN
D LM<>105 THEN IF LM=MM AND (RM=DB OR
RM=SB) THEN 629
625 IF LM=SB AND MM=DB AND (RM=DB OR R
M=SB) THEN F=LM:S=MM:T=RM:GOSUB FPAY
626 IF LM=SB AND (MM=DB OR MM=SB) AND
RM=DB THEN F=LM:S=MM:T=RM:GOSUB FPAY
628 GOTO 630
629 F=LM:S=MM:T=RM:GOSUB FPAY
630 IF BET=1 THEN 680
631 IF LT=DB AND MT=SB AND (RT=DB OR R
T=SB) THEN F=LT:S=MT:T=RT:GOSUB FPAY
632 IF (LT=CH AND MT<>CH) OR (LT=CH AN
D MT=CH) THEN F=LT:S=MT:T=RT:GOSUB FPA
Y
633 IF LT=DB AND (MT=SB OR MT=DB) AND
RT=SB THEN F=LT:S=MT:T=RT:GOSUB FPAY
634 IF LT=MT AND RT=MT THEN F=LT:S=MT:
T=RT:GOSUB FPAY
635 IF LT=SB AND MT=DB AND (RT=DB OR R
T=SB) THEN F=LT:S=MT:T=RT:GOSUB FPAY
636 IF LT<>CH AND LT<>105 AND LT<>DB A
ND LT<>SB THEN IF LT=MT AND (RT=DB OR
RT=SB) THEN 640
637 IF LT=SB AND (MT=SB OR MT=DB) AND
RT=DB THEN F=LT:S=MT:T=RT:GOSUB FPAY
638 GOTO 642
640 F=LT:S=MT:T=RT:GOSUB FPAY
642 IF BET=2 THEN 680
643 IF LB=DB AND MB=SB AND (RB=DB OR R
B=SB) THEN F=LB:S=MB:T=RB:GOSUB FPAY
644 IF (LB=CH AND MB<>CH) OR (LB=CH AN
D MB=CH) THEN 652
645 IF LB=DB AND (MB=DB OR MB=SB) AND
RB=SB THEN F=LB:S=MB:T=RB:GOSUB FPAY
646 IF LB=MB AND RB=MB THEN 652
647 IF LB=SB AND MB=DB AND (RB=DB OR R
B=SB) THEN F=LB:S=MB:T=RB:GOSUB FPAY
648 IF LB<>CH AND LB<>105 AND LB<>DB A
ND LB<>SB THEN IF LB=MB AND (RB=DB OR
RB=SB) THEN 652
649 IF LB=SB AND (MB=DB OR MB=SB) AND
RB=DB THEN F=LB:S=MB:T=RB:GOSUB FPAY
650 GOTO 654
652 F=LB:S=MB:T=RB:GOSUB FPAY
654 IF BET=3 THEN 680
655 IF LT=DB AND MM=SB AND (RB=DB OR R
B=SB) THEN F=LT:S=MM:T=RB:GOSUB FPAY
656 IF (LT=CH AND MM=CH) OR (LT=CH AND
 MM<>CH) THEN 664
657 IF LT=DB AND (MM=DB OR MM=SB) AND
RB=SB THEN F=LT:S=MM:T=RB:GOSUB FPAY
658 IF LT=MM AND RB=MM THEN 664
659 IF LT=SB AND MM=DB AND (RB=DB OR R
B=SB) THEN F=LT:S=MM:T=RB:GOSUB FPAY
660 IF LT<>CH AND LT<>105 AND LT<>DB A
ND LT<>SB THEN IF LT=MM AND (RB=DB OR
RB=SB) THEN 664
661 IF LT=SB AND (MM=DB OR MM=SB) AND
RB=DB THEN F=LT:S=MM:T=RB:GOSUB FPAY
662 GOTO 665
664 F=LT:S=MM:T=RB:GOSUB FPAY
665 IF BET=4 THEN 680
666 IF LB=105 AND MM=LB AND RT=MM THEN
 WIN=WIN+2000:GOSUB 2600
667 IF LB=DB AND MM=SB AND (RT=DB OR R
T=SB) THEN F=LB:S=MM:T=RT:GOSUB FPAY
668 IF (LB=CH AND MM<>CH) OR (LB=CH AN
D MM=CH) THEN 676
669 IF LB=DB AND (MM=DB OR MM=SB) AND
RT=SB THEN F=LB:S=MM:T=RT:GOSUB FPAY
670 IF LB<>105 AND LB=MM AND RT=MM THE
N 676
671 IF LB=SB AND MM=DB AND (RT=DB OR R
T=SB) THEN F=LB:S=MM:T=RT:GOSUB FPAY
672 IF LB<>CH AND LB<>105 AND LB<>DB A
ND LB<>SB THEN IF LB=MM AND (RT=DB OR
RT=SB) THEN 676
673 IF LB=SB AND (MM=DB OR MM=SB) AND
RT=DB THEN F=LB:S=MM:T=RT:GOSUB FPAY
674 GOTO 680
676 F=LB:S=MM:T=RT:GOSUB FPAY
680 BANKROLL=BANKROLL+WIN:POSITION 29,
22:? BANKROLL;
685 IF WIN>0 AND WIN<10 THEN DUR=2:GOS
UB WINSOUND
690 IF WIN>=10 AND WIN<25 THEN DUR=3:G
OSUB WINSOUND
691 IF WIN>=25 AND WIN<=50 THEN DUR=5:
GOSUB WINSOUND
695 IF WIN>50 AND WIN<2000 THEN DUR=10
:GOSUB WINSOUND
700 POSITION 4,4:? "┘";:POSITION 4,6:?
 "┘";:POSITION 4,10:? "┘";:POSITION 4,
12:? "├";
705 BET=0:POSITION 24,21:PRINT BET;
710 IF BANKROLL>0 THEN 530
720 GOTO 350
1000 REM WINNER SOUND
1010 FOR I=1 TO DUR
1015 FOR S=40 TO 90 STEP 5
1020 SOUND 0,5,10,10
1025 NEXT S
1030 FOR S=90 TO 40 STEP -5
1035 SOUND 0,5,10,10
1040 NEXT S
1050 NEXT I
1055 SOUND 0,0,0,0:RETURN
1300 PS=2
1310 FOR I=1 TO 8
```

```
1320 POSITION 34,P5:? ASC(PAY$(I,I))*B
ET;" ";
1325 P5=P5+2
1330 NEXT I
1340 IF BET<5 THEN POSITION 34,18:? AS
C(PAY$(9,9))*BET;" ";:RETURN
1350 IF BET=5 THEN POSITION 34,18:? AS
C(PAY$(9,9))*10;
1360 RETURN
2000 POKE 752,1:? CHR$(125):POSITION 2
,2:PRINT "├├├├├├├├├├├├├├├    ab       P
AYS 2"
2005 POKE 756,PEEK(106)+1:SETCOLOR 1,0
,0:SETCOLOR 2,0,15
2010 PRINT "├├├├├├├├├├├├├├├├"
2012 PRINT "├┘┘┘┘┘┘┘┘┘┘┘┘├├    ab ab
 PAYS 5"
2014 PRINT "├├┘  ┘   ┘   ┘├├ "
2016 PRINT "├├┘  ┘   ┘   ┘├├    cd cd cd
 PAYS 10"
2020 PRINT "├├┘  ┘   ┘   ┘├├ █"
2025 PRINT "├├▪  ▪   ▪   ▪├├ ▌  ef ef ef
 PAYS 14"
2030 PRINT "├├┘  ┘   ┘   ┘├├ ▌"
2035 PRINT "├├┘  ┘   ┘   ┘├├ ▌  gh gh gh
 PAYS 18"
2040 PRINT "├├┘  ┘   ┘   ┘├├ ▌"
2045 PRINT "├├├├├├├├├├├├├├├├ ▌  ███ ███ ███
 PAYS 20"
2046 PRINT " ||||||||||||||▐▌"
2048 PRINT " ├┤├┤├┤├┤├┤├┤├┤█     MIX BARS
 PAYS 20"
2050 PRINT " H┫┫┫┫┫┫┫┫┫┫┫┫┫ ▀▀"
2055 PRINT " h▪▪▪▪▪▪▪▪▪▪▪▪▪ ◤   ═══ ═══ ═══
 PAYS 50"
2060 PRINT " H▪▀▀▀▀▀▀▀▀▀▀▀┫ |"
2065 PRINT " h▪▪▪▪▪▪▪▪▪▪▪┫ |     ij ij ij
 PAYS 200"
2066 PRINT " H┫┫┫┫┫┫┫┫┫┫┫┫┫|"
2068 PRINT " ├┤┤┤┤┤┤┤┤┤┤┤┤┤|"
2069 PRINT "├├├├├├├├├├├├├├├├"
2070 PRINT "├├/          \├├"
2080 PRINT "├├├├├├├├├├├├├├├├";
2090 RETURN
2100 POKE 752,1:FOR I=19 TO 23
2110 POSITION 20,I
2120 PRINT "                ";
2130 NEXT I
2140 RETURN
2200 FOR I=1 TO 5
2210 POSITION 20,16:PRINT "PLAY 1 TO 5
 COINS";
2220 FOR DELAY=1 TO 10:NEXT DELAY
2230 POSITION 20,16:PRINT "PLAY 1 TO 5
 COINS";
2240 FOR DELAY=1 TO 10:NEXT DELAY
2250 NEXT I
2260 RETURN
2300 POKE 752,1:POSITION 17,7
2310 FOR I=1 TO 5
2320 PRINT " ↓←█←";
2325 FOR DELAY=1 TO 20:NEXT DELAY
2330 NEXT I
2340 FOR I=1 TO 5
2350 PRINT "▌↑←█←";
2355 FOR DELAY=1 TO 20:NEXT DELAY
2360 NEXT I
2370 WIN=0:POSITION 34,23:? WIN;"  ";
2390 RETURN
2400 IF LM=CH AND MM<>CH THEN WIN=BET*
2
2410 IF LM=CH AND MM=CH THEN WIN=BET*5
2420 IF LM=99 AND MM=LM AND RM=MM THEN
 WIN=BET*10
2425 IF LM=99 AND MM=99 AND (RM=DB OR
RM=5B) THEN WIN=WIN*10
2430 IF LM=101 AND MM=LM AND RM=MM THE
N WIN=BET*14
2435 IF LM=101 AND MM=101 AND (RM=DB O
R RM=5B) THEN WIN=BET*14
2440 IF LM=103 AND MM=LM AND RM=MM THE
N WIN=BET*18
2445 IF LM=103 AND MM=103 AND (RM=DB O
R RM=5B) THEN WIN=BET*18
2450 IF LM=DB AND MM=LM AND RM=MM THEN
 WIN=BET*50
2452 IF LM=5B AND MM=LM AND RM=MM THEN
 WIN=BET*20
2453 IF LM=DB AND MM=5N AND (RM=DB OR
RM=5B) THEN WIN=BET*20
2454 IF LM=DB AND (MM=DB OR MM=5B) AND
 RM=5B THEN WIN=BET*20
2455 IF LM=5B AND MM=DB AND (RM=DB OR
RM=5B) THEN WIN=BET*20
2456 IF LM=5B AND (MM=DB OR MM=5B) AND
 RM=DB THEN WIN=BET*20
2460 IF LM=105 AND MM=LM AND RM=MM AND
 BET<5 THEN WIN=BET*200
2470 IF LM=105 AND MM=LM AND RM=MM AND
 BET=5 THEN WIN=BET*2000:GOSUB 2600
2480 POSITION 34,23:PRINT WIN;" ";:BAN
KROLL=BANKROLL+WIN
2490 POSITION 29,22:PRINT BANKROLL;" "
;:IF BANKROLL>100000 THEN 2700
2495 RETURN
2600 FOR I=0 TO 200 STEP 5
2605 SOUND 0,I,0,15
2610 NEXT I
2615 FOR I=1 TO 5
2620 FOR 5=40 TO 90 STEP 2
2625 SOUND 0,5,10,10
2630 NEXT 5
2640 FOR 5=90 TO 40 STEP -2
2645 SOUND 0,5,10,10
2650 NEXT 5:NEXT I
2655 FOR I=1 TO 20
2660 FOR I=0 TO 200 STEP 5
2665 SOUND 0,I,0,15
2670 NEXT I:SOUND 0,0,0,0
2672 FOR I=1 TO 10
2673 FOR DELAY=1 TO 40:NEXT DELAY
2674 POSITION 20,20:? "SUPER JACKPOT!!
!";
2675 FOR DELAY=1 TO 20:NEXT DELAY
2676 FOR DELAY=1 TO 20:NEXT DELAY
2678 POSITION 20,20:? "
 ";
2680 NEXT I
2685 POSITION 20,20:PRINT "
        ";
2690 RETURN
2800 FOR I=1 TO 200 STEP 25
2810 SOUND 0,I,6,8
2820 NEXT I
2830 SOUND 0,0,0,0:RETURN
2950 L=L+INT(RND(0)*6)*2:IF L>59 THEN
L=L-60
2960 M=M+16+INT(RND(0)*6)*2:IF M>59 TH
EN M=M-60
2970 R=R+22+INT(RND(0)*6)*2:IF R>59 TH
EN R=R-60
3000 POKE 77,0:FOR X=1 TO 15
3010 POSITION 11,10:PRINT R$(R,R+1):R=
R+2:IF R>59 THEN R=1
3020 POSITION 11,8:PRINT R$(R,R+1):R=R
+2:IF R>59 THEN R=1
3030 POSITION 11,6:PRINT R$(R,R+1)
3040 R=R-2:IF R<1 THEN R=R+60
3045 IF X=11 THEN GOSUB 2800
3050 IF X>10 THEN 3110
3060 POSITION 8,10:PRINT M$(M,M+1):M=M
+2:IF M>59 THEN M=1
3070 POSITION 8,8:PRINT M$(M,M+1):M=M+
2:IF M>59 THEN M=1
3080 POSITION 8,6:PRINT M$(M,M+1)
3100 M=M-2:IF M<1 THEN M=M+60
3105 IF X=6 THEN GOSUB 2800
3110 IF X>5 THEN 3160
3120 POSITION 5,10:PRINT L$(L,L+1):L=L
+2:IF L>59 THEN L=1
3130 POSITION 5,8:PRINT L$(L,L+1):L=L+
2:IF L>59 THEN L=1
3140 POSITION 5,6:PRINT L$(L,L+1)
3150 L=L-2:IF L<1 THEN L=L+60
3160 NEXT X
3165 GOSUB 2800
3170 L=L-2:IF L<1 THEN L=L+60:M=M-2:IF
 M<1 THEN M=N+60:R=R-2:IF R<1 THEN R=R
+60
3180 RETURN
4000 IF CHR$(F)="a" AND CHR$(5)<>"a" T
HEN W=2
4010 IF F=CH AND S=CH THEN W=5
```

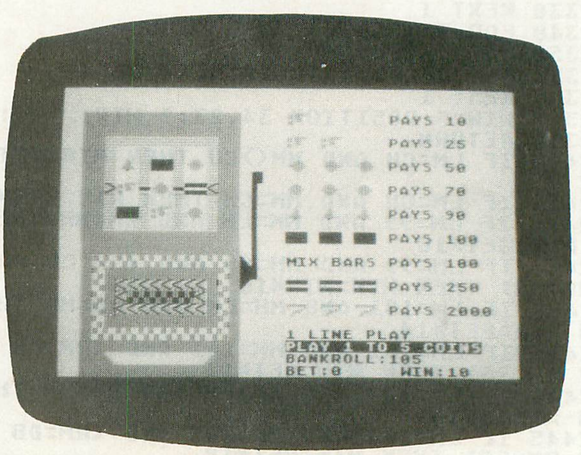
```
4020 IF F=99 AND S=99 AND T=99 THEN W=
10
4025 IF F=99 AND S=99 AND (T=DB OR T=D
B) THEN W=10
4030 IF F=101 AND S=101 AND T=101 THEN
 W=14
4035 IF F=101 AND S=101 AND (T=DB OR T
=SB) THEN W=14
4040 IF F=103 AND S=103 AND T=103 THEN
 W=18
4045 IF F=103 AND S=103 AND (T=DB OR T
=SB) THEN W=18
4050 IF F=DB AND S=F AND T=S THEN W=20
4052 IF F=DB AND S=SB AND (T=DB OR T=S
B) THEN W=20
4053 IF F=DB AND (S=DB OR S=SB) AND T=
SB THEN W=20
4054 IF F=SB AND S=DB AND (T=DB OR T=S
B) THEN W=20
4055 IF F=SB AND (S=DB OR S=SB) AND T=
DB THEN W=20
4058 IF F=SB AND S=F AND T=S THEN W=20
4060 IF F=105 AND S=F AND T=S THEN W=2
00
4065 WIN=WIN+W:POSITION 34,23:PRINT WI
N;
4080 RETURN
10000 V=0:FOR I=0 TO 200 STEP 25
10005 SOUND 0,I,0,15
10006 SOUND 1,I,2,15:SOUND 2,I,4,15
10010 NEXT I
10015 SOUND 0,0,0,0:SOUND 1,0,0,0:SOUN
D 2,0,0,0
10090 STOP
20000 FOR I=1 TO 5
20005 FOR S=0 TO 200 STEP 5
20010 SOUND 0,S,8,15
20015 NEXT S
20020 FOR S=200 TO 0 STEP -5
20025 SOUND 0,S,8,15
20030 NEXT S
20035 NEXT I
20040 SOUND 0,0,0,0
20045 STOP
32000 POKE 106,PEEK(106)-5:GRAPHICS 2:
START=(PEEK(106)+1)*256:POKE 756,START
/256:POKE 752,1
32005 DIM XFR$(38):XFR$="h]♥,K,M)♦,N%j
┻i ,L♥1M,KHPyfLfN%NIdP,♦"
32010 ? #6;"     *************"
32011 ? #6;"     * COLOR     *"
32012 ? #6;"     * slot      *"
32013 ? #6;"     * machine   *"
32014 ? #6;"     *************"
32015 ? "BY MICHAEL A. IVINS"
32020 Z=USR(ADR(XFR$)):RESTORE 32100
32030 READ X:IF X=-1 THEN RESTORE :RET
URN
32040 FOR Y=0 TO 7:READ Z:POKE X+Y+STA
RT,Z:NEXT Y:GOTO 32030
32100 DATA 520,170,170,170,170,170,170
,170,170
32101 DATA 528,170,85,170,85,170,85,17
0,85
32102 DATA 536,170,0,170,0,170,0,170,0
32103 DATA 544,160,160,160,160,10,10,1
0,10
32104 DATA 552,80,80,80,80,5,5,5,5
32105 DATA 560,128,128,160,160,168,168
,170,170
32106 DATA 568,2,2,10,10,42,42,170,170
32107 DATA 584,234,184,46,139,46,186,2
24,170
32108 DATA 600,167,28,114,200,114,156,
39,170
32109 DATA 608,170,0,170,255,255,170,0
,170
32110 DATA 616,1,171,7,175,31,191,127,
255
32111 DATA 624,255,127,191,159,175,167
,171,169
32112 DATA 776,2,82,82,81,1,81,80,80
32113 DATA 784,170,168,128,64,64,64,0,
0
32114 DATA 792,1,5,5,21,21,5,5,1
32115 DATA 800,64,80,80,84,84,80,80,64
32116 DATA 808,2,10,10,42,42,10,10,2
32117 DATA 816,128,160,160,168,168,160
,160,128
32118 DATA 824,1,2,1,2,5,10,21,3
32119 DATA 832,0,128,64,128,64,160,80,
128
32120 DATA 840,85,85,64,0,1,5,4,20
32121 DATA 848,85,84,4,16,80,64,0,0
32122 DATA -1
```

●

D: CHECK DATA

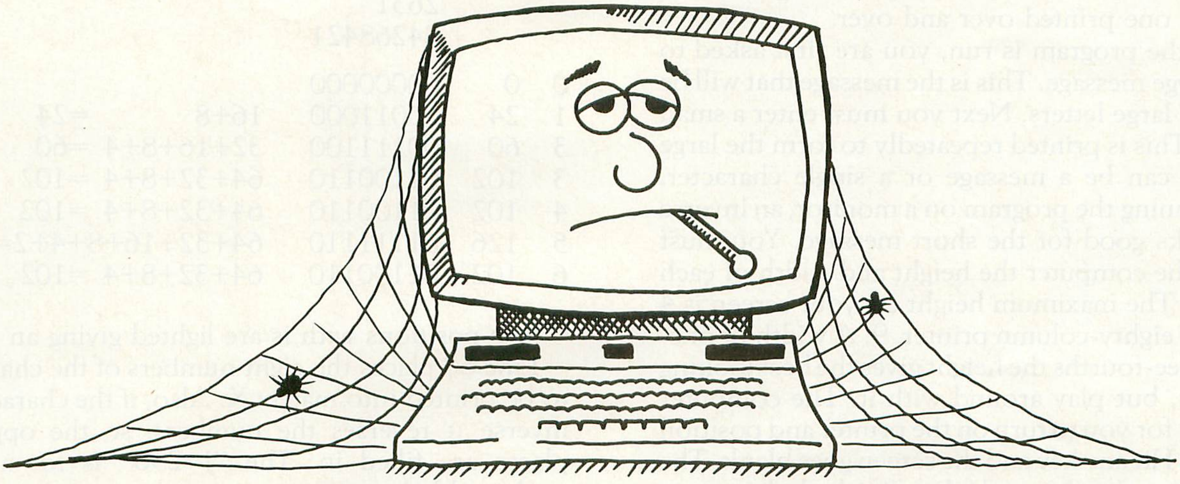
```
1 DATA 536,499,899,760,612,466,678,105
,546,19,143,396,55,17,548,6279
130 DATA 158,933,697,360,244,341,374,3
35,797,598,134,35,354,376,338,6074
315 DATA 806,170,901,19,506,56,364,786
,155,407,340,655,516,575,667,6923
510 DATA 586,851,653,224,552,940,534,1
28,130,457,454,285,808,353,297,7252
566 DATA 133,434,622,825,96,930,995,30
6,27,725,148,803,578,361,100,7083
628 DATA 526,175,819,735,163,883,351,7
28,546,945,734,190,788,198,859,8640
645 DATA 654,857,646,390,959,524,111,8
31,595,872,811,137,315,541,742,8985
662 DATA 823,403,644,995,720,954,105,6
29,628,365,825,752,166,201,797,9007
690 DATA 936,245,94,807,780,492,798,93
0,744,423,644,806,257,413,518,8887
1050 DATA 489,246,280,173,873,577,496,
408,611,794,925,732,697,14,187,7502
2016 DATA 760,436,815,167,822,168,181,
736,991,278,542,614,501,363,807,8181
2069 DATA 57,157,977,656,616,747,946,4
92,788,170,331,373,983,375,580,8248
2260 DATA 962,985,342,237,621,791,351,
237,627,800,668,969,932,865,610,9997
2425 DATA 472,248,664,267,687,646,409,
229,633,520,947,809,263,418,411,7623
2495 DATA 812,237,360,505,170,97,432,5
37,289,566,774,193,156,571,234,5933
2672 DATA 441,409,971,401,637,318,812,
27,972,291,320,580,483,399,695,7756
2970 DATA 731,427,351,178,607,832,535,
650,246,299,629,757,313,576,320,7451
3130 DATA 126,99,504,542,960,708,794,5
66,829,983,465,46,231,399,305,7557
4050 DATA 847,411,497,417,488,886,62,8
69,655,951,446,245,662,595,647,8678
20000 DATA 481,208,531,689,297,486,678
,938,250,640,237,816,778,290,315,7634
32013 DATA 583,782,846,875,528,960,853
,606,110,666,650,918,17,702,619,9715
32109 DATA 597,529,954,959,491,625,205
,753,127,634,597,658,955,621,8705
```

●



QUICKSOFT™

CURES THE #1 TERMINAL ILLNESS:



S.D.S.

(SLOW DELIVERY SYNDROME)

QUICKSOFT'S cure doesn't come in a bottle, but it does come in handy when your computer needs software delivered promptly.

24 HOUR SHIPPING: QUICKSOFT is the software-house with a difference: **SPEED!!** When you place a phone order with **QUICKSOFT we guarantee that it will be shipped within twenty-four hours.***

WIDE SELECTION: QUICKSOFT carries thousands of software disks, tapes and cartridges for every purpose. We have business systems to cure those overflowing files, educational disks to relieve the worry of a D in chemistry, and adventure games to aid a case of the blues.

QUICKSOFT HAS THEM ALL, and right now we're having specials on these items from our "most wanted" list:

SHAMUS	**21.95**
SPACE EGGS	**21.95**
CHICKEN	**21.95**
TRACK ATTACK	**21.95**
SPEED READ +	**47.95**
ZORK I or II	**32.95**
TEXT WIZARD	**79.95**
CURSE OF RA	**16.95**
MOSAIC 32K RAM	**139.95**
DISK DETECTIVE	**21.95**

OFFER EXPIRES NOVEMBER 30, 1982.

FREE DELIVERY: QUICKSOFT offers free delivery on *all* orders of $50.00 or more. We not only ship within 24 hours but we also pay the freight! (For orders of less than $50 add $1.50 for shipping and handling.)

CALL TOLL FREE ANYTIME. QUICKSOFT'S toll free number is available anytime of the day or night. For placing an order or for requesting our **FREE CATALOG** simply dial:

1-800-547-8009

IN OREGON CALL 1-342-1298

* **GUARANTEE: QUICKSOFT** guarantees shipment within 24 hours on each phone order. If the item ordered is not readily available and cannot be shipped on time, you will receive an additional 5% OFF the regular **QUICKSOFT** price.

VISA-MASTERCHARGE and C.O.D. Welcome

QUICKSOFT™

Department 97, P.O. Box 10854   Eugene, OR 97440

A BANNER BANNER PROGRAM

16K CASSETTE OR DISK

by Andrew Lieberman

C'mon now, another program to print banners? Yes! But this one prints all 256 ASCII characters, is only 19 lines long, and doesn't have a single DATA statement. Also, the letters can be of different heights and widths, and your large message is made of a small one printed over and over.

When the program is run, you are first asked to input a large message. This is the message that will be printed in large letters. Next you must enter a small message. This is printed repeatedly to form the large letters. It can be a message or a single character. When running the program on a monitor, an inverse space looks good for the short message. You must now tell the computer the height and width of each character. The maximum height on your screen is 4 and on an eighty-column printer, 9. A width of one-half to three-fourths the height gives the best looking characters, but play around with it. The computer now waits for you to turn on the printer and position the paper. Hit any key and the screen goes blank. The banner starts printing and when finished, the screen comes back to life. You are now asked if you want to run the program again.

If you want the banner to fly along the screen instead of the printer you must make two small changes. First delete the "POKE 559,0" from line 50. Second, change the "LPRINT" in line 80 to "PRINT". You may want to save this version separately.

Lines 1 to 45 are preparatory. Line 15 sets SP$ equal to all spaces for use in the program. The maximum lengths of L$ and S$ are set to 128, but can be increased by changing the DIM statement.

Line 50 first turns off the ANTIC chip by POKing location 559 with a 0. This causes the screen to go blank, but greatly increases execution speed. Next, a loop is started to print each character of the large message. A variable, X, is set equal to the ASCII value of the letter and is checked to see if it is an inverse character. If it is an inverse letter, the value is lowered by 128, making it the normal version, and a flag, IN, is set equal to 1.

Line 55 changes X from the ASCII value of the desired letter to the internal character set value of the letter.

Before I explain line 60, let me give you some information on the ATARI's character set. Each character is stored in ROM as eight numbers per character. When these numbers are converted from decimal to binary they show which pixels should be lighted. Take as an example the letter "A". Its eight numbers are 0,24,60,102,102,126,102,0. When changed to binary it looks like:

		1 2631 84268421		
0	0	00000000		
1	24	00011000	16+8	=24
3	60	00111100	32+16+8+4	=60
3	102	01100110	64+32+8+4	=102
4	102	01100110	64+32+8+4	=102
5	126	01111110	64+32+16+8+4+2	=126
6	102	01100110	64+32+8+4	=102

The positions with is are lighted giving an "A".

Line 60 places the eight numbers of the character to be printed into matrix X. Also, if the character is inverse it reverses the numbers, so the opposite places are filled in. The "J=256" is for a loop explained below.

Line 70 first divides J in half. It then starts a loop which checks each member of matrix X in a backwards order. If that number is less than J, spaces are left in PR$ and the count of PR$, CO, is advanced.

If the computer did not skip from line 70 to line 80 then a printed section should be put into PR$. This is done by advancing the count of S$, C, and loading it into PR$.

Line 80 prints PR$ a number of times equal to the width. It then sets the count of PR$ to 1 and sets PR$ equal to all spaces.

Line 85 checks for the end of loop J.

Line 90 turns the ANTIC chip back on and checks to see if you want to run the program again.

So, there you have it. I hope you could understand all the little tricks I used, but whether or not you do, have fun with this program. □

```
1 ? "K          A BANNER BANNER PROGRAM"
2 ? "            BY ANDREW LIEBERMAN":?
10 DIM S$(128),L$(128),SP$(160),PR$(16
0),X(7):C=0:CO=1:OPEN #1,4,0,"K:"
15 FOR A=1 TO 160:SP$(A,A)=" ":NEXT A
20 ? :? "LARGE MESSAGE";:INPUT L$:LL=L
EN(L$)
30 ? "SMALL MESSAGE";:INPUT S$:LS=LEN(
S$)
40 TRAP 40:? "HEIGHT";:INPUT H:IF H<1
OR H>20 THEN 40
42 TRAP 42:? "WIDTH";:INPUT W:IF W<1 O
R W>20 THEN 42
45 ? "POSITION PAPER--HIT ANY KEY":GET
 #1,A
```

```
50 POKE 559,0:FOR I=1 TO LL:IN=0:X=ASC
(L$(I,I)):IF X>127 THEN X=X-128:IN=1
55 X=X-32*(X<96 AND X>31)+64*(X<32)
60 FOR A=0 TO 7:X(A)=ABS(IN*255-PEEK(5
7344+X*8+A)):NEXT A:J=256
70 J=J/2:FOR K=7 TO 0 STEP -1:IF X(K)<
J THEN PR$(CO,CO+H-1)=SP$:CO=CO+H:GOTO
 80
75 X(K)=X(K)-J:FOR A=1 TO H:C=C+1-LS*(
C>=LS):PR$(CO,CO)=S$(C,C):CO=CO+1:NEXT
 A
80 NEXT K:FOR A=1 TO W:TRAP 99:LPRINT
PR$(1,H*8):NEXT A:CO=1:PR$=SP$
85 IF J<>1 THEN 70
90 NEXT I:POKE 559,34:? :? :? "AGAIN?"
;:GET #1,A:IF A=89 THEN C=0:CO=1:GOTO
20
92 END
99 POKE 559,34:? :? "YOUR PRINTER IS N
OT ON, TRY AGAIN":C=1:CO=1:GOTO 45
```

●

D: CHECK DATA

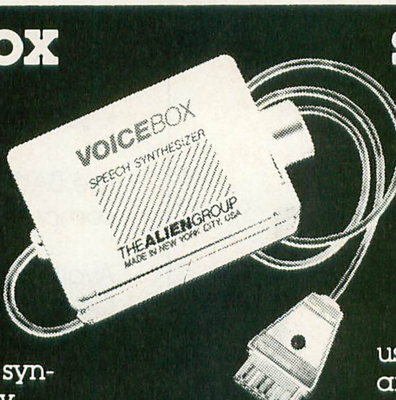
```
1 DATA 640,325,961,426,578,399,545,856
,498,353,293,834,934,935,889,9466
85 DATA 197,924,24,550,1695
```

WIN $5,000 Plus Royalties!*

For the best Talking Game

For the Atari®

Using the VOICE BOX Speech Synthesizer



Now you can make your Atari® 400/800 games and other programs come alive with the VOICE BOXTM by the Alien Group—the first low-cost, smart speech synthesizer with unlimited vocabulary.

.Add jokes to your programs. Insults. Compliments. Help messages. Stories. Alien voices. Animal roars. Have your computer talk to the fire department or police in emergencies. To kids. Or blind people. Teach touch typing with immediate spoken feedback. Or just about any other subject — the fun way. Or help a speech-impaired friend communicate . . . the possibilities are limitless.

The VOICE BOX plugs into your Atari's serial port. And talks directly through your TV set. No speaker, amplifier, power supply, special interfaces, or cables needed.

Just select from its simple screen menu. A dictionary with thousands of common words (on diskette or cassette) automatically translates your text into speech. It's that easy.

But don't let its friendliness fool you. The VOICE BOX has all 64 phonemes (basic sounds, like "ah") built in. So you can precisely create any word or sound you can imagine. And store it all on diskette or tape. Names or foreign language words, for example. Or wierd non-human languages.

The VOICE BOX is creative too. It will crack you and your friends up with non-stop random, grammatically correct sentences, using words you specify. It also has an amusing talking face with lip-sync animation — a real crowd-stopper. Best of all, you can call the VOICE BOX from any BASIC program and make your program really hum — literally!

Don't confuse the VOICE BOX with "dumb" speech synthesizers that can't learn new words. Or software-based ones with lower speech quality — and an annoying tendency to blank out the display when they talk. The VOICE BOX is a true breakthrough in speech synthesis. Small wonder thousands of Atari owners have already bought the VOICE BOX.

The VOICE BOX is available now at leading computer stores throughout the world. Or direct from the Alien Group, with 10-day money back guarantee if you're not completely satisfied.

VOICE BOX For Atari. $169.00
16K and 32K versions included
(Specify diskette or cassette).

Enclose check or money order, Visa or Mastercard #. Please include expiration date.

CHECK YOUR LOCAL COMPUTER STORE FOR TALKING "VOICE BOX" VERSIONS OF YOUR FAVORITE GAMES FROM LEADING GAME COMPANIES

*Win a $5,000 prize — plus royalties — for the best Atari 400/800 game using the VOICE BOX. Deadline: May 30, 1983. Write for contest details.

Please mail to: The Alien Group, Department AL-2 , 27 W. 23 St., N.Y., N.Y. 10010
Or call in order to (212) 741-1770

Atari is a registered trademark of Atari Inc. VOICE BOX is trademark of the Alien Group.

ON CONVERTING MICROSOFT BASIC

by Richard J. Kalagher

The purpose of this article is to point out some pitfalls that await the programmer who has learned to program with ATARI BASIC and wants to switch to ATARI's new Microsoft BASIC. I do not intend to "review" ATARI Microsoft BASIC or catalog all the differences between the two versions of BASIC. Instead I will try to help you avoid some errors that can occur without you even knowing it.

Let's start with a simple example. The statement PRINT (½)*2 should give an answer of 1. Not in ATARI Microsoft BASIC, however. You will get 0. To understand why this happens you have to read between the lines in the ATARI Microsoft BASIC manual. ATARI Microsoft BASIC allows three types of numbers: integer, single precision, and double precision. When you assign numbers to a variable, they will be stored according to how you defined the variable. In the example above, however, the ATARI Microsoft BASIC interpreter saw two constants inside of the parenthesis. Since these constants did not contain decimal points, the interpreter assumed they were integers. Thus (½) evaluates to 0 in ATARI Microsoft BASIC because division of two integers gives only the integer result and not the remainder.

To avoid errors in your programs you must be very careful to either assign the constants to single or double precision variables or to put a decimal point in at least one of the numbers. For example, PRINT (1./2)*2 will give you the correct answer. It is unfortunate that the ATARI Microsoft BASIC manual does not warn you of such problems, even in the section on converting from ATARI BASIC to ATARI Microsoft BASIC.

There are more problems with numbers in ATARI Microsoft BASIC, however. For example, consider the following simple program:

```
10 X=0
20 FOR N=1 TO 10000
30 X=X+0.1
40 NEXT N
50 PRINT X
```

You would expect (and you would get in ATARI BASIC) a result of 1000. However ATARI Microsoft BASIC gives you a result of 1000.28! The culprit here is the way ATARI Microsoft BASIC stores numbers. In good old ATARI BASIC all numbers are stored in Binary Coded Decimal (BCD) format. This means that each decimal digit is represented exactly in binary format. In ATARI Microsoft BASIC, however, the numbers are stored directly in binary. Since many decimal numbers can not be exactly represented in binary, you will get errors in the conversion process.

The reason that ATARI Microsoft BASIC uses binary representation is to conserve memory and to execute faster. Personally, I would rather have slightly slower execution and use slightly more memory in order to get the accuracy I am used to with ATARI BASIC. To be fair, even the IBM Personal Computer as well as the Apple and the TRS-80 represent numbers in binary format and have the same potential accuracy problems.

This problem is much more annoying when using double precision numbers in ATARI Microsoft BASIC. The computer seems to want to give you sixteen digits even when you manipulate numbers with one or two digits. In most cases the extra digits are just a nuisance which you can easily avoid using the PRINT USING command. By the way, if I sound negative toward ATARI Microsoft BASIC let me tell you that PRINT USING alone makes ATARI Microsoft BASIC a wonderful language to work with. It's just too bad ATARI did not allow a BCD option with the language.

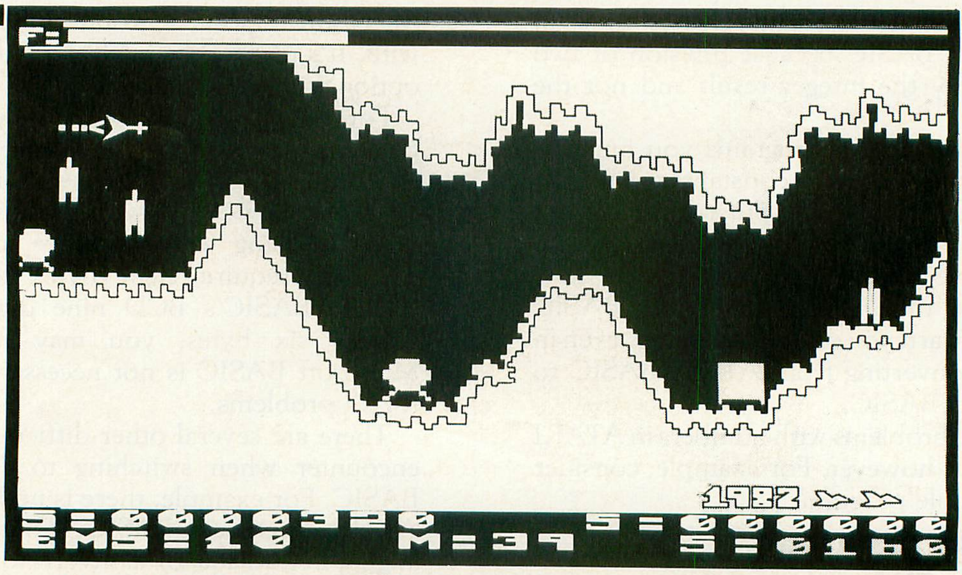
For most financial problems, by the way, you will probably need to use double precision. Since single precision is only useful to six digits, any number greater than 9999.99 will drop the least significant digits causing errors. Since double precision numbers require eight bytes for storage while ATARI BASIC's BCD nine digit numbers only require six bytes, you may find that ATARI Microsoft BASIC is not necessarily more efficient for all problems.

There are several other difficulties that you may encounter when switching to ATARI Microsoft BASIC. For example, there is no DEG statement in ATARI Microsoft BASIC. Thus you must enter angles in radians or convert to radians in your program. The ATARI Microsoft BASIC manual could have told you that 1 degree = .01745 radians but you are just greeted with a blank in Appendix J on converting from ATARI 8K BASIC to ATARI Microsoft BASIC.

At first glance, the change from the "POINT #iocb sector,byte" to "INPUT #iocb AT(sector,byte)" seems like a better way to do it. But I have at least one program that I wanted to convert to ATARI Microsoft BASIC that was not easy without the POINT command. In this program I had a disk file containing my checkbook account. In order to avoid reading the whole file each time I wanted to find something, I used NOTE to mark the beginning of each month. By storing just 12 pairs of numbers I could use POINT to jump to the beginning of any month. Since I was using POINT just to move the file pointer followed by a loop to read the file, I could no longer just move the pointer without also reading a record. Although you can do it with INPUT AT the code is very awkward.

Most reviewers of ATARI Microsoft BASIC hail the string array feature. I will mention only one case where the old ATARI BASIC strings have an advantage. It is sometimes convenient to print to the screen by dimensioning a string of 38 characters and selectively inserting information into the string. Something like A$(10,13)="DATE" can be used to do this. With ATARI Microsoft BASIC it is very awkward to do this. You must take the string apart with LEFT$, RIGHT$, and MID$ and then put it back together again.

ATARI Microsoft BASIC is really a superior version of BASIC. It could have been a lot better if ATARI had incorporated some of the nice features of the old 8K BASIC into ATARI Microsoft BASIC. But particularly in the area of potential problems with integer division and binary storage, ATARI has done a diservice to their customers in not even mentioning these factors in the ATARI Microsoft BASIC manual. □



Dungeons and Dragons® Character Generator

24K CASSETTE 32K DISK

by Bob Curtin

When I first bought my ATARI, one of the things I put high on my list of priorities was to try one of the computer adventure games on the market. I wasn't impressed with the game, but I was impressed with the ease of play. Pressing a few buttons took care of movement, combat, encumbrance, game time, and all the rest, and it dawned on me that my computer could be a big help to me in my ongoing DUNGEONS & DRAGONS campaign. I set to work writing a series of utility programs for it. This, the first, generates both player and non-player characters in an average of about four minutes. Normally, it takes anywhere from twenty to forty minutes to generate a character "by hand," and then there's a strong possibility of missing a few modifiers along the way. The computer always remembers.

Though the program was written to take the work out of generating characters, the Dungeon Master and players are still left with choices to make. As in D&D, the player still has choice of name, gender, race, class, and character level. Those categories greatly affect the final character statistics, and it would be an injustice to randomly choose them for the player. By the same token, there are certain minimum ability scores, or racial requirements, which must be met to assume the role of a particular race or class. The user doesn't have to know or worry about it; the computer will figure it all out and tell the player if he or she doesn't measure up. The player may continue to choose alternatives until one of his choices meets all requirements. The program will then continue on.

The system used is straight, unmodified, ADVANCED DUNGEONS & DRAGONS. There is an omission, however — by choice, not error. I didn't incorporate the maximum level restrictions imposed on certain races, such as an Elf being able to rise no higher than 7th level as a fighter. If a Dungeon Master wants to adhere to those limits, it's a simple matter to just look it up; while it's not so simple to get the computer to do what it's told not to. For those of you who want to ignore the limits, the computer doesn't know any better. Indulge.

I fudged a couple of other values, too. For instance, line 195 contains the random number generator for the characters' basic abilities. Notice that variables A and C have a +2 for the add-on number. I did this to give the players a break. All you hardline Dungeon Masters out there gnashing your teeth can switch back to +1 if you want. (Essentially, they're now rolling 3D6+2).

THE PROGRAM

As I said, there are five inputs. They are, in order: name, gender, race, class, and character level. Here is an explanation of each.

Name — after the title, the computer will ask for a character name. This is the only "open" input, and — although you have to work at it — it can be screwed up. For example, entering a couple of the current controls through the escape key will cause some grief later on down the line. Other than that, anything but an input of YES, NO, Y, or N will be taken as the character name. If you don't want a name, just hit the return key. Entering NO or N will fetch a list of names from memory as suggestions to the player.

GENDER — The computer will only accept M or F. Lower case letters will not work.

RACE and CLASS — Only the exact initials listed in parentheses on the respective menus will be accepted.

CHARACTER LEVEL — Any level between 1 and 18 (inclusive) will be accepted. If a value below 1 is entered, the value will be upped to 1. If a value over 18 is entered, a short message will be displayed and the program will loop back for another input. Any illegal entry, such as a letter instead of a number, will also cause the loop back for re-entry.

As the character builds, the computer does the appropriate calculations, comparisons, and modifications between inputs and then displays the results. After the information has been copied from the screen, the player may continue the program by pressing any key.

PROGRAM OUTLINE

5 - 26	Initialization
50 - 75	Character Race Modifier Routine
80 - 82	Custom Display List
88 - 86	Title (So, my vanity's showing.) This can be deleted by eliminating lines 80 through 96 and changing the last statement in line 26 to GOTO 100.
100 - 111	Thief, Magic-user, and Monk Data
159 - 179	Name Input
180 - 187	Gender Input
190 - 192	Race Menu
195	Basic Ability Scores
200 - 225	Race Input
226 - 229	Ability Score Display
235 - 243	Class Menu and Class Input
245 - 254	Class Trigger and Gold Piece Generator
263 - 269	Exceptional Strength Routine
276 - 332	Hit, Damage, Armor Class, and Dexterity Modifiers
335 - 341	Modifier Display
345 - 374	Height and Weight Routine (modified by race and gender)
375 - 438	Hit Point Generation Routine (modified by race and ability)
460 - 475	Thieves Abilities
500 - 530	Magic-user Abilities
550 - 599	Monk Abilities
2000-2020	Name List
2550-2730	Race Limitations
5000-5975	Class Limitations
6132-6200	Thief Abilities Modifiers (by race and ability scores)
7000-7055	Psionics Routine
8000-8020	Input Error Routine

A FEW SUGGESTIONS FOR THE DM

Never lose sight of the fact that the only reason a player will participate in one of your D&D sessions is to have FUN! Nothing will dampen the enthusiasm of a new player faster than being forced to assume the role of a character too weak to take any kind of initiative, do any exploring, or even stand fast with the rest of the party. Force your players into a position of constant impotence and you'll soon find your dungeon devoid of adventurers.

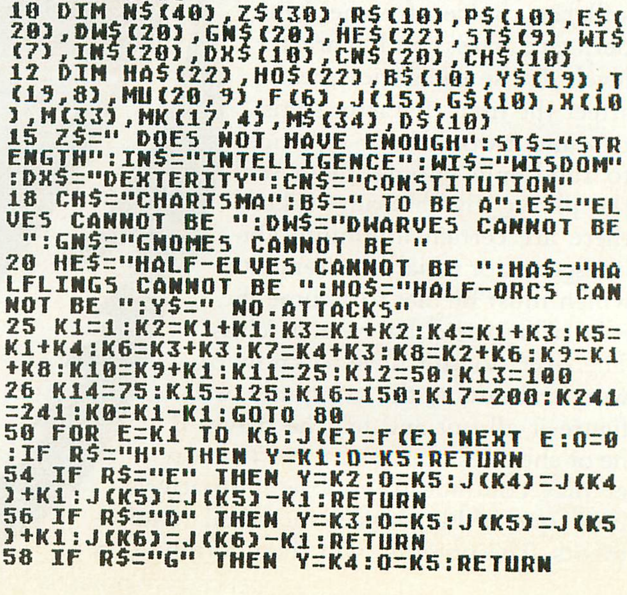
Although I'm certainly not in favor of the give-away dungeon, killer dungeons are, if not worse, at least as bad. Surviving and advancing up the ladder of experience — **developing** a character is what D&D is all about. To have a developed character snuffed out by the undetectable, unseeable, or unknowable is bound to cause you to gain a reputation as a "cheap shot" dungeon master. Having a character killed because of one's own recklessness or bad luck or **bad choice between alternatives** can be lived with. But the skewering of some hapless player for no rhyme or reason is unforgivable.

Give your players a break. Pick a number — I use five — and let each player run off that many characters. The player can then choose one of them to start the game with, and should that character come to an untimely end, there are four more from which to choose. That way, no more valuable playing time is taken up generating characters.

Normally, novice players start at level one. However, after a player has campaigned for some time, it's usually the practice to let him or her start higher than that. If they have a character killed off, you could, for instance, have them start a couple of levels lower than the character who was killed. Another way is to roll a six or eight-sided die.

Above all, be fair. Remember that you, and consequently all of the creatures you control, have perfect intelligence. Your players do not; they only know what you tell them. It behooves you to give that little extra. If a player can't see something, don't wait for him to ask; tell him.

Good luck. Good dungeoning. □



```
5 TRAP 8000
10 DIM N$(40),Z$(30),R$(10),P$(10),E$(
20),DW$(20),GN$(20),HE$(22),ST$(9),WI$
(7),IN$(20),DX$(10),CN$(20),CH$(10)
12 DIM HA$(22),HO$(22),B$(10),Y$(19),T
(19,8),MU(20,9),F(6),J(15),G$(10),X(10
),M(33),MK(17,4),M$(34),D$(10)
15 Z$=" DOES NOT HAVE ENOUGH":ST$="STR
ENGTH":IN$="INTELLIGENCE":WI$="WISDOM"
:DX$="DEXTERITY":CN$="CONSTITUTION"
18 CH$="CHARISMA":B$=" TO BE A":E$="EL
VES CANNOT BE ":DW$="DWARVES CANNOT BE
 ":GN$="GNOMES CANNOT BE "
20 HE$="HALF-ELVES CANNOT BE ":HA$="HA
LFLINGS CANNOT BE ":HO$="HALF-ORCS CAN
NOT BE ":Y$=" NO.ATTACKS"
25 K1=1:K2=K1+K1:K3=K1+K2:K4=K1+K3:K5=
K1+K4:K6=K3+K3:K7=K4+K3:K8=K2+K6:K9=K1
+K8:K10=K9+K1:K11=25:K12=50:K13=100
26 K14=75:K15=125:K16=150:K17=200:K241
=241:K0=K1-K1:GOTO 80
50 FOR E=K1 TO K6:J(E)=F(E):NEXT E:O=0
:IF R$="H" THEN Y=K1:O=K5:RETURN
54 IF R$="E" THEN Y=K2:O=K5:J(K4)=J(K4
)+K1:J(K5)=J(K5)-K1:RETURN
56 IF R$="D" THEN Y=K3:O=K5:J(K5)=J(K5
)+K1:J(K6)=J(K6)-K1:RETURN
58 IF R$="G" THEN Y=K4:O=K5:RETURN
```

```
60 IF R$="HE" THEN Y=K5:O=K5:GOTO 75
62 IF R$="HA" THEN Y=K6:O=K5:J(K1)=J(K
1)-K1:J(K4)=J(K4)+K1:RETURN
64 IF R$="HO" THEN Y=K7:O=K5:J(K1)=J(K
1)+K1:J(K5)=J(K5)+K1:J(K6)=J(K6)-K2:RE
TURN
75 RETURN
80 POKE 712,128:? "K":DL=PEEK(560)+256
*PEEK(561):POKE 752,K1:POKE 559,K0
81 Z1=PEEK(DL+K4):Z2=PEEK(DL+K5):POKE
DL+K3,71:POKE DL+K4,Z1:POKE DL+K5,Z2:P
OKE DL+K6,K7:POKE DL+K7,K6
82 POKE DL+K8,K6:POKE DL+K9,K6:POKE DL
+K10,K6:POKE DL+K11+K2,65:POKE DL+K11+
K3,PEEK(560):POKE DL+29,PEEK(561)
88 POKE 82,0:POKE 559,34:POKE 710,128:
? "K DUNGEONS & DRAGONS":? "  RANDOM C
HARACTER":? " GENERATION PROGRAM"
92 ? "                BY BOB CURTIN"
93 ? :? :? "   THIS PROGRAM WAS WRITTE
N TO TAKE":? "     SOME OF THE BURDEN
OFF OF THE"
94 ? "    USUALLY HARRIED DUNGEON MAST
ER."
95 ? :? "     PLEASE BE SURE TO PRESS R
ETURN   ":? "             AFTER EACH INPU
T.":?
96 ? :? :? "      GOOD LUCK!  GOOD DUNG
EONING!":FOR E=K1 TO K10^3*5:NEXT E
100 FOR I=K1 TO K8:FOR X=K1 TO K10+K8:
READ N:T(X,I)=N:NEXT X:NEXT I
101 FOR I=K1 TO K3:FOR X=K9 TO K10+K9:
READ N:MU(X,I)=N:NEXT X:NEXT I
102 FOR I=K1 TO K4:FOR X=K1 TO K10+K7:
READ N:MK(X,I)=N:NEXT X:NEXT I
103 DATA 30,35,40,45,50,55,60,65,70,80
,90,100,105,110,115,125,125,125,25,29,
33,37,42,47,52,57,62,67,72,77,82,87
104 DATA 92,97,99,99,20,25,30,35,40,45
,50,55,60,65,70,75,80,85,90,95,99,99,1
5,21,27,33,40,47,55,62,70,78,86,94,99
105 DATA 99,99,99,99,99,10,15,20,25,31
,37,43,49,56,63,70,77,85,93,99,99,99,9
9,10,10,15,15,20,20,25,25,30,30,35,35
106 DATA 40,40,50,50,55,55,85,86,87,88
,90,92,94,96,98,99,99.1,99.2,99.3,99.4
,99.5,99.6,99.7,99.8,0,0,0,20,25,30
107 DATA 35,40,45,50,55,60,65,70,75,80
,80,80,35,45,45,45,55,55,65,65,75,85,9
5,4,5,5,5,6,6,7,7,8,9,10,6,7,7,7,9,9
108 DATA 11,11,14,18,99
109 DATA 10,9,8,7,7,6,5,4,3,3,2,1,0,-1
,-1,-2,-3,150,160,170,180,190,200,210,
220,230,240,250,260,270,280,290,300
110 DATA 320,1,1,1,54,54,32,32,32,2,2,
52,52,52,3,3,4,4,13,14,16,16,27,28,39,
212,312,313,413,416,517,520,624
111 DATA 530,832
159 POKE 82,2:GRAPHICS 1:POKE 752,1:PO
KE 712,128:POKE 710,128
160 RESTORE :? #6:? #6:? #6;" DUNGEONS
 & DRAGONS":? #6;"CHARACTER GENERATION
"
170 ? "HAVE YOU THOUGHT OF A NAME":? "
FOR YOUR CHARACTER";:INPUT N$
175 IF N$="YES" OR N$="Y" THEN ? "KWEL
L, WHAT IS IT";:INPUT N$
179 IF N$="NO" OR N$="N" THEN GRAPHICS
 0:POKE 710,6:POKE 709,0:POKE 752,1:GO
SUB 2000
180 ? "KWHAT GENDER IS ";N$;" (M/F)";:
INPUT G$:O=0:IF G$="M" OR G$="F" THEN
O=K5
187 IF O<>K5 THEN ? "KM/F ONLY, PLEASE
!":FOR E=K1 TO 1500:NEXT E:GOTO 180
190 ? #6:? #6:? #6;"  HUMAN        (H)"
:? #6;"  ELF          (E)":? #6;"  DWA
RF       (D)"
192 ? #6;"  GNOME        (G)":? #6;"
HALFLING   (HA)":? #6;"  HALF-ELF  (HE
)":? #6;"  HALF-ORC  (HO)"
195 FOR E=K1 TO K6:A=INT(K6*RND(K1)+K2
):B=INT(K6*RND(K1)+1):C=INT(K6*RND(K1)
+K2):D=A+B+C:F(E)=D:NEXT E:GOTO 205
200 POP :?
205 ? "WHAT RACE";:INPUT R$:GOSUB K12
210 IF O<>K5 THEN ? "KINITIALS ONLY, P
LEASE!":GOTO 205
215 GOSUB 2550
220 FOR E=K1 TO K6:IF J(E)>K9+K9 THEN
J(E)=K9+K9
224 IF J(E)<K3 THEN J(E)=K3
225 F(E)=J(E):NEXT E
226 GRAPHICS K1:POKE 712,50:POKE 710,5
0:? #6:? #6:? #6
227 ? #6:? #6:? #6:? #6;"  STRENGTH
   ";F(K1):? #6;"  INTELLIGENCE  ";F(K
2):? #6;"  WISDOM        ";F(K3)
228 ? #6;"  DEXTERITY      ";F(K4):? #6
;"  CONSTITUTION  ";F(K5):? #6;" CHAR
ISMA       ";F(K6):? #6:? #6
229 ? #6;" BASIC ABILITIES":POKE 752,
1:? "   PRESS ANY KEY TO CONTINUE"
230 OPEN #1,4,0,"K:":GET #1,E:CLOSE #1
:IF E>0 THEN 235
235 GRAPHICS 1:POKE 709,96:POKE 710,16
8:POKE 712,98:POKE 752,1
236 ? #6:? #6:? #6
237 ? #6;"  FIGHTER        (F)":? #6;"
PALADIN         (P)":? #6;"  RANGER
 (R)":? #6;"  CLERIC          (C)"
238 ? #6;"  DRUID          (D)":? #6;"
MONK            (M)":? #6;"  THIEF
 (T)":? #6;"  ASSASSIN        (A)"
239 ? #6;"  MAGIC-USER  (MU)":? #6;"
ILLUSIONIST  (I)":? #6:? #6:? #6;" CH
OOSE FROM":? #6;" THE ABOVE LIST"
240 GOTO 243
241 POP :?
243 Z=K0:O=K0:ES=K0:? "WHAT CLASS";:IN
PUT P$
245 IF P$="F" THEN O=K5:Z=K1:GP=INT(15
0*RND(1)+50)
246 IF P$="R" THEN O=K5:Z=K2:GP=INT(K1
6*RND(K1)+50)
247 IF P$="P" THEN O=K5:Z=K3:GP=INT(K1
6*RND(1)+50)
248 IF P$="C" THEN O=K5:Z=K4:GP=INT(K1
6*RND(1)+30)
249 IF P$="D" THEN O=K5:Z=K5:GP=INT(K1
6*RND(1)+30)
250 IF P$="T" THEN O=K5:Z=K6:GP=INT(K1
3*RND(1)+20)
251 IF P$="A" THEN O=K5:Z=K7:GP=INT(K1
3*RND(1)+20)
252 IF P$="MU" THEN O=K5:Z=K8:GP=INT(6
0*RND(1)+20)
253 IF P$="I" THEN O=K5:Z=K9:GP=INT(60
*RND(1)+20)
254 IF P$="M" THEN O=K5:Z=K10:GP=INT(1
5*RND(1)+5)
255 IF O<>K5 THEN ? "CORRECT INITIALS
ONLY, PLEASE!":? :GOTO 243
262 GOSUB 5000
263 IF P$="F" OR P$="R" OR P$="P" THEN
 IF F(K1)=K10+K8 THEN 265
264 GOTO 276
265 GRAPHICS 2+16:POKE 711,4:? #K6:? #
K6:? #K6:? #K6;" ";N$;" HAS ":? #K6;"
EXCEPTIONAL":? #K6;" STRENGTH"
269 ? #6:ES=INT(K13*RND(K1)+K1):? #6;"
 E.S.RATING  18/";ES:FOR E=K1 TO 2000:
NEXT E
276 MH=0:MD=0:MA=0:MR=0:K325=325:K335=
335
310 IF ES=K13 THEN MH=K3:MD=K6:GOTO K3
25
311 IF ES>=K13-K9 THEN MH=K2:MD=K5:GOT
O K325
312 IF ES>=3*K11+K1 THEN MH=K2:MD=K4:G
OTO K325
313 IF ES>=K12+K1 THEN MH=K2:MD=K3:GOT
O K325
314 IF ES>=K1 THEN MH=K1:MD=K3:GOTO K3
25
315 IF A=K9+K9 THEN MH=K1:MD=K2:GOTO K
325
316 IF A=K10+K7 THEN MH=K1:MD=K1:GOTO
K325
317 IF A=K10+K6 THEN MD=K1:GOTO K325
318 IF A=K3 THEN MH=-K3:MD=-K2:GOTO K3
25
319 IF A=K4 THEN MH=-K2:MD=-K2:GOTO K3
25
320 IF A<=K6 THEN MH=-K1:GOTO K325
```

```
325 IF D=K9+K9 THEN MR=K3:MA=-K4:GOTO
K335
326 IF D=K9+K8 THEN MR=K2:MA=-K3:GOTO
K335
327 IF D=K8+K8 THEN MR=K1:MA=-K2:GOTO
K335
328 IF D=K7+K8 THEN MA=-K1:GOTO K335
329 IF D=K6 THEN MA=K1:GOTO K335
330 IF D=K5 THEN MR=-K1:MA=K2:GOTO K33
5
331 IF D=K4 THEN MR=-K2:MA=K3:GOTO K33
5
332 IF D=K3 THEN MR=-K3:MA=K4:GOTO K33
5
335 GRAPHICS 1:POKE 712,128:POKE 708,2
2:POKE 709,22:POKE 752,1:POKE 710,128
336 ? #6:? #6;" ";N$;"'S"
337 ? #6;" MODIFIERS:":? #6
338 ? #6;" HIT            ";MH
339 ? #6;" DAMAGE         ";MD
340 ? #6;" A/C ADJUSTMENT ";MA
341 ? #6;" R/A BONUS      ";MR
345 X(5)=INT(7*RND(1)):X(6)=INT(9*RND(
1)):X(7)=INT(11*RND(1)):X(8)=INT(13*RN
D(1)):X(9)=INT(25*RND(1))
346 X(6)=INT(9*RND(1))
350 M(5)=INT(40*RND(1)):M(6)=INT(30*RN
D(1)):M(7)=INT(20*RND(1)):M(8)=INT(50*
RND(1)):M(9)=INT(66*RND(1))
355 IF G$="F" THEN 365
356 IF Y=K3 THEN H=K2*K11-K6+X(K7):W=K
13+K11+K9+M(K5)
357 IF Y=K2 THEN H=K12+K6+X(K7):W=K13-
K10+M(K6)
358 IF Y=K4 THEN H=K12-K10-K1+X(K5):W=
K3*K11-K3+M(K7)
359 IF Y=K5 THEN H=K12+K10+X(K8):W=110
+M(K5)
360 IF Y=K7 THEN H=K12+K10+K2+X(K6):W=
K16+M(K8)
361 IF Y=K6 THEN H=K11+K10+K1+X(K6):W=
80+M(K5)
362 IF Y=K1 THEN H=K12+K10+X(K9):W=K13
+K11+K5+M(K9)
363 GOTO 372
365 IF Y=K3 THEN H=42+X(K6):W=K14+K4+M
(K6)
366 IF Y=K2 THEN H=K12+X(K7):W=K13-K5+
M(K7)
367 IF Y=K4 THEN H=K6*K6+X(K5):W=K6*K1
0+K7+M(K7)
368 IF Y=K5 THEN H=K12+K6+X(K8):W=K8*K
10+M(K6)
369 IF Y=K7 THEN H=K12+K9+X(K5):W=K14+
K5+M(K8)
370 IF Y=6 THEN H=30+X(5):W=42+M(7)
371 IF Y=K1 THEN H=K12+K6+X(K4):W=K14+
M(K9)
372 Q1=INT(H/12):Q2=Q1*12:Q3=H-Q2
373 ? #6:? #6:? #6;" HEIGHT   ";Q1;"'"
;Q3;CHR$(34)
374 ? #6;" WEIGHT   ";W;"LBS."
375 HPT=K0:O=K0:GOTO 400
380 HP=INT(K4*RND(K1)+K2):RETURN
385 HP=INT(K6*RND(K1)+K2):RETURN
390 HP=INT(K8*RND(K1)+K2):RETURN
395 HP=INT(K10*RND(K1)+K2):RETURN
400 ? "WHAT LEVEL IS ";N$;:INPUT L:IF
Z=K2 THEN L=L+K1
406 IF L>18 THEN ? "KYOU CAN'T START A
 CHARACTER":? "OVER LEVEL 18.  TRY AGA
IN.":? :GOTO 400
407 IF L<=0 THEN L=1
408 FOR J=K1 TO L:IF Z=K1 OR Z=K3 THEN
 GOSUB 395
410 IF Z=K2 OR Z=K4 OR Z=K5 THEN GOSUB
 390
415 IF Z=K6 OR Z=K7 THEN GOSUB 385
420 IF Z=K8 OR Z=K9 OR Z=K10 THEN GOSU
B 380
422 HPT=HPT+HP:NEXT J:GOTO 431
427 IF E=K9+K9 THEN HPT=HPT+(L*K4):GOT
O 438
428 IF E=K9+K8 THEN HPT=HPT+(L*K3):GOT
O 438
429 IF E=K8+K8 THEN HPT=HPT+(L*K2):GOT
O 438
430 GOTO 432
431 IF E>=K8+K8 THEN HPT=HPT+(L*K2):GO
TO 438
432 IF E=K9+K6 THEN HPT=HPT+L:GOTO 438
433 IF E=K3 THEN HPT=HPT-(L*K2):GOTO 4
38
434 IF E<K8 THEN HPT=HPT-L
438 ? #6:? #6;" HIT POINTS    ";HPT
440 IF Z=K1 OR Z=K3 THEN IF L>=12 THEN
 ? #6;Y$;"     2/1":GOTO 456
445 IF Z=K1 OR Z=K3 THEN IF L>=K6 THEN
 ? #6;Y$;"     3/2":GOTO 456
446 IF Z=K2 THEN IF L>=16 THEN ? #6;Y$
;"     2/1":GOTO 456
447 IF Z=K2 THEN IF L>=K7 THEN ? #6;Y$
;"     3/2"
456 IF Y=K1 OR Y=K3 OR Y=K6 THEN GOSUB
 7000
457 ? "K";N$;" HAS ";GP;" GOLD PIECES"
458 GOSUB 6130:? :? "   PRESS ANY KEY
TO CONTINUE"
459 OPEN #1,4,0,"K:":GET #1,I:CLOSE #1
:IF I>0 THEN 460
460 B5=INT(L/K4)+K2:IF Z=K6 OR Z=K7 TH
EN 462
461 GOTO 500
462 GRAPHICS 1+16
463 ? #6:? #6:? #6:? #6
465 ? #6;"     BS - - - - X";BS
466 ? #6;"     PP - - - - ";T(L,K1)
467 ? #6;"     LOCKS  - - ";T(L,K2)
468 ? #6;"     TRAPS  - - ";T(L,K3)
469 ? #6;"     MS - - - - ";T(L,K4)
470 ? #6;"     HS - - - - ";T(L,K5)
471 ? #6;"     HEAR - - - ";T(L,K6)
472 ? #6;"     CLIMB  - - ";T(L,K7)
473 ? #6;"     LANGUAGES ";T(L,K8)
474 ? #6:? #6:? #6;" THIEVES ABILITIES
"
475 FOR I=K1 TO K10^3*K5:NEXT I
500 IF Z=K8 OR Z=K9 THEN 505
501 IF Z=K10 THEN 550
502 GOTO 4999
505 GRAPHICS 2+16:POKE 712,160
510 ? #6:? #6:? #6:? #6;" CHANCE TO KN
OW   ";MU(B,1)
515 ? #6:? #6;" MINIMUM SPELLS  ";MU(B
,2)
520 ? #6:? #6;" MAXIMUM SPELLS  ";MU(B
,3)
530 FOR I=K1 TO 4000:NEXT I
535 GOTO 4999
550 GRAPHICS 1+16:POKE 712,212:POKE 71
0,224
551 ? #6:? #6:? #6:? #6;"   MONKS TABL
E":? #6
552 ? #6:? #6;" ARMOR CLASS      ";MK(L
,1)
553 ? #6;" MOVE             ";MK(L,2);"'"
554 IF MK(L,K3)=K1 THEN M$="1"
555 IF MK(L,K3)=54 THEN M$="5/4"
556 IF MK(L,K3)=32 THEN M$="3/2"
557 IF MK(L,K3)=K2 THEN M$="2"
558 IF MK(L,K3)=52 THEN M$="5/2"
559 IF MK(L,K3)=K3 THEN M$="3"
560 IF MK(L,K3)=K4 THEN M$="4"
561 ? #6;" ATTACKS/ROUND ";M$
562 IF MK(L,K4)=13 THEN D$="1D3"
563 IF MK(L,K4)=14 THEN D$="1D4"
564 IF MK(L,K4)=16 THEN D$="1D6"
565 IF MK(L,K4)=27 THEN D$="1D6+1"
566 IF MK(L,K4)=28 THEN D$="2D4"
567 IF MK(L,K4)=39 THEN D$="3D3"
568 IF MK(L,K4)=212 THEN D$="2D6"
569 IF MK(L,K4)=312 THEN D$="3D4"
570 IF MK(L,K4)=413 THEN D$="3D4+1"
571 IF MK(L,K4)=416 THEN D$="4D4"
572 IF MK(L,K4)=517 THEN D$="4D4+1"
573 IF MK(L,K4)=520 THEN D$="5D4"
574 IF MK(L,K4)=624 THEN D$="6D4"
575 IF MK(L,K4)=530 THEN D$="5D6"
576 IF MK(L,K4)=832 THEN D$="4D8"
577 ? #6;" DAMAGE/ATTACK ";D$
578 ? #6:? #6:? #6;" SEE THE PLAYERS"
579 ? #6;" HANDBOOK FOR"
580 ? #6;" SPECIAL ABILITIES"
599 FOR I=K1 TO 5000:NEXT I
1999 GOTO 4999
```

```
2000 ? "↓↓IF YOU'RE HAVING TROUBLE PIC
KING":? "A NAME FOR YOUR CHARACTER, PE
RHAPS"
2005 ? "YOU'D LIKE A FEW SUGGESTIONS.
"
2010 ? "YOU'RE WELCOME TO USE ONE OF T
HESE:"
2015 ? "↓↓SETH THE HUGE","BUCKTHORN"
2016 ? "AARON THE SWIFT","ELLIDE"
2017 ? "BRIAN OF BLACKMOOR","JANO"
2018 ? "ALONSO THE HOOK","TAPHENESE"
2019 ? "SIR BAGLEY","BAAREN SATO"
2020 ? "↓↓IF YOU WANT ONE OF THESE, JU
ST":? "TYPE IN THE NAME AND PRESS RETU
RN."
2022 ? "↓IF YOU DON'T, TYPE 'NO' AND P
RESS":? "RETURN.":? "↓NAME";:INPUT N$:
IF N$="NO" OR N$="N" THEN N$="WHOOZIT"
2028 GRAPHICS 1:POKE 708,40:POKE 752,1
:RETURN
2550 A=J(K1):B=J(K2):C=J(K3):A1=J(K4):
B1=J(K5):C1=J(K6):? "↰"
2555 ON Y-K1 GOTO 2600,2580,2630,2650,
2670,2700
2576 RETURN
2580 IF A<K8 THEN ? N$;Z$:? ST$;B$;" D
WARF.":GOTO K17
2585 IF B1<K6*K2 THEN ? N$;Z$:? CN$;B$
;" DWARF.":GOTO K17
2590 IF G$="F" THEN IF J(K1)>K9+K8 THE
N J(K1)=K10*K7
2595 IF J(K4)>K9+K8 THEN J(K4)=K9+K8
2597 IF J(K6)>K8+K8 THEN J(K6)=K8+K8
2599 RETURN
2600 IF B<K8 THEN ? N$;Z$:? IN$;B$;"N
ELF.":GOTO K17
2605 IF A1<K7 THEN ? N$;Z$:? DX$;B$;"N
 ELF.":GOTO K17
2610 IF B1<K6 THEN ? N$;Z$:? CN$;B$;"N
 ELF.":GOTO K17
2615 IF C1<K8 THEN ? N$;Z$:? CH$;B$;"N
 ELF.":GOTO K17
2620 IF G$="F" THEN IF J(K1)>K8+K8 THE
N J(K1)=K8+K8
2625 RETURN
2630 IF A<K6 THEN ? N$;Z$:? ST$;B$;" G
NOME.":GOTO K17
2635 IF B<K7 THEN ? N$;Z$:? IN$;B$;" G
NOME.":GOTO K17
2640 IF B1<K8 THEN ? N$;Z$:? CN$;B$;"
GNOME.":GOTO K17
2645 IF G$="F" THEN IF J(K1)>K3*K5 THE
N J(K1)=K3*K5
2648 RETURN
2650 IF B<K4 THEN ? N$;Z$:? IN$;B$;" H
ALF-ELF.":GOTO K17
2655 IF A1<K6 THEN ? N$;Z$:? DX$;B$;"
HALF-ELF.":GOTO K17
2660 IF B1<K6 THEN ? N$;Z$:? CN$;B$;"
HALF-ELF.":GOTO K17
2665 IF G$="F" THEN IF J(K1)>K9+K8 THE
N J(K1)=K9+K8
2668 RETURN
2670 IF A<K6 THEN ? N$;Z$:? ST$;B$;" H
ALFLING.":GOTO K17
2675 IF B<K6 THEN ? N$;Z$:? IN$;B$;" H
ALFLING.":GOTO K17
2680 IF A1<K8 THEN ? N$;Z$:? DX$;B$;"
HALFLING.":GOTO K17
2685 IF B1<K10 THEN ? N$;Z$:? CN$;B$;"
 HALFLING.":GOTO K17
2690 IF G$="M" THEN IF J(K1)>K9+K8 THE
N J(K1)=K9+K8
2694 IF G$="F" THEN IF J(K1)>K7+K7 THE
N J(K1)=K7+K7
2695 IF J(K3)>K9+K8 THEN J(K3)=K9+K8
2696 RETURN
2700 IF A<K6 THEN ? N$;Z$:? ST$;B$;" H
ALF-ORC.":GOTO K17
2705 IF B1<K6+K7 THEN ? N$;Z$:? CN$;B$
;" HALF-ORC.":GOTO K17
2710 IF J(K2)>K9+K9 THEN J(K2)=K9+K8
2715 IF J(K3)>K7+K7 THEN J(K3)=K7+K7
2720 IF J(K4)>K7+K7 THEN J(K4)=K7+K7
2725 IF J(K6)>K6+K6 THEN J(K6)=K6+K6
2730 RETURN
4999 GRAPHICS 1:SETCOLOR 2,L,4:POKE 75
2,1:SETCOLOR 4,L,4:GOTO 160
5000 A=F(K1):B=F(K2):C=F(K3):D=F(K4):E
=F(K5):F=F(K6):? "↰"
5005 ON Z GOTO 5100,5200,5300,5400,550
0,5600,5700,5800,5900,5950
5055 RETURN
5100 IF A<K9 THEN ? N$;Z$:? ST$;B$;" F
IGHTER.":GOTO K241
5105 IF E<K7 THEN ? N$;Z$:? CN$;B$;" F
IGHTER.":GOTO K241
5110 RETURN
5200 IF A<K10+K3 THEN ? N$;Z$:? ST$;B$
;" RANGER.":GOTO K241
5205 IF B<K10+K3 THEN ? N$;Z$:? IN$;B$
;" RANGER.":GOTO K241
5210 IF C<K10+K4 THEN ? N$;Z$:? WI$;B$
;" RANGER.":GOTO K241
5215 IF E<K10+K4 THEN ? N$;Z$:? CN$;B$
;" RANGER.":GOTO K241
5220 IF Y=K3 THEN ? DW$;"RANGERS.":GOT
O K241
5225 IF Y=K2 THEN ? E$;"RANGERS.":GOTO
 K241
5230 IF Y=K4 THEN ? GN$;"RANGERS.":GOT
O K241
5235 IF Y=K6 THEN ? HA$;"RANGERS.":GOT
O K241
5240 IF Y=K7 THEN ? HO$;"RANGERS.":GOT
O K241
5245 RETURN
5300 IF A<K10+K2 THEN ? N$;Z$:? ST$;B$
;" PALADIN.":GOTO K241
5305 IF B<K9 THEN ? N$;Z$:? IN$;B$;" P
ALADIN.":GOTO K241
5310 IF C<K10+K3 THEN ? N$;Z$:? WI$;B$
;" PALADIN.":GOTO K241
5315 IF E<K9 THEN ? N$;Z$:? CN$;B$;" P
ALADIN.":GOTO K241
5320 IF F<K9+K8 THEN ? N$;Z$:? CH$;B$;
" PALADIN.":GOTO K241
5325 IF Y<>K1 THEN ? "ONLY HUMANS CAN
BE PALADINS.":GOTO K241
5330 RETURN
5400 IF C<K9 THEN ? N$;Z$:? WI$;B$;" C
LERIC.":GOTO K241
5405 IF Y=K6 THEN ? HA$;"CLERICS.":GOT
O K241
5410 RETURN
5500 IF C<K10+K2 THEN ? N$;Z$:? WI$;B$
;" DRUID.":GOTO K241
5505 IF F<K10+K5 THEN ? N$;Z$:? CH$;B$
;" DRUID.":GOTO K241
5510 IF Y=K3 THEN ? DW$;"DRUIDS.":GOTO
 K241
5515 IF Y=K2 THEN ? E$;"DRUIDS.":GOTO
K241
5520 IF Y=K4 THEN ? GN$;"DRUIDS.":GOTO
 K241
5525 IF Y=K7 THEN ? HO$;"DRUIDS.":GOTO
 K241
5530 RETURN
5600 IF D<K9 THEN ? N$;Z$:? DX$;B$;" T
HIEF.":GOTO K241
5605 RETURN
5700 IF A<K10+K2 THEN ? N$;Z$:? ST$;B$
;"N ASSASSIN.":GOTO K241
5705 IF B<K10+K1 THEN ? N$;Z$:? IN$;B$
;"N ASSASSIN.":GOTO K241
5710 IF D<K10+K2 THEN ? N$;Z$:? DX$;B$
;"N ASSASSIN.":GOTO K241
5720 IF Y=K6 THEN ? HA$;"ASSASSINS.":G
OTO K241
5725 RETURN
5800 IF B<K9 THEN ? N$;Z$:? IN$;B$;" M
AGIC-USER.":GOTO K241
5805 IF D<K6 THEN ? N$;Z$:? DX$;B$;" M
AGIC-USER.":GOTO K241
5810 IF Y=K3 THEN ? DW$;"MAGIC-USERS."
:GOTO K241
5815 IF Y=K4 THEN ? GN$;"MAGIC-USERS."
:GOTO K241
5820 IF Y=K7 THEN ? HO$;"MAGIC-USERS."
:GOTO K241
5825 IF Y=K6 THEN ? HA$;"MAGIC-USERS."
:GOTO K241
5830 RETURN
5900 IF B<K10+K5 THEN ? N$;Z$:? IN$;B$
```

```
;"N ILLUSIONIST.":GOTO K241
5905 IF D<K10+K6 THEN ? N$;Z$:? DX$;B$
;"N ILLUSIONIST.":GOTO K241
5910 IF Y=K3 THEN ? DW$;"ILLUSIONISTS.
":GOTO K241
5915 IF Y=K2 THEN ? E$;"ILLUSIONISTS."
:GOTO K241
5920 IF Y=K7 THEN ? HO$;"ILLUSIONISTS.
":GOTO K241
5925 IF Y=K6 THEN ? HA$;"ILLUSIONISTS.
":GOTO K241
5930 RETURN
5950 IF A<K10+K5 THEN ? N$;Z$:? ST$;B$
;" MONK.":GOTO K241
5955 IF C<K10+K5 THEN ? N$;Z$:? WI$;B$
;" MONK.":GOTO K241
5960 IF D<K10+K1 THEN ? N$;Z$:? DX$;B$
;" MONK.":GOTO K241
5965 IF E<K10+K1 THEN ? N$;Z$:? CN$;B$
;" MONK.":GOTO K241
5970 IF Y<>K1 THEN ? "ONLY HUMANS CAN
BE MONKS.":GOTO K241
5975 RETURN
6130 IF D=18 THEN T(L,K1)=T(L,K1)+K10:
T(L,K2)=T(L,K2)+15:T(L,K3)=T(L,K3)+K5:
T(L,K4)=T(L,K4)+10:T(L,K5)=T(L,K5)+10
6131 IF D=K10+K7 THEN T(L,K1)=T(L,K1)+
K5:T(L,K2)=T(L,K2)+K10:T(L,K4)=T(L,K4)
+K5:T(L,K5)=T(L,K5)+K5
6132 IF D=K10+K6 THEN T(L,K2)=T(L,K2)+
K5
6133 IF D=K10+K2 THEN T(L,K4)=T(L,K4)-
K5
6134 IF D=K10+K1 THEN T(L,K1)=T(L,K1)-
K5:T(L,K3)=T(L,K3)-K5:T(L,K4)=T(L,K4)-
K10
6135 IF Y=K3 THEN T(L,K2)=T(L,K2)+K10:
T(L,K3)=T(L,K3)+15:T(L,K7)=T(L,K7)-K10
:T(L,K8)=T(L,K8)-K5
6136 IF Y=K2 THEN T(L,K1)=T(L,K1)+K5:T
(L,K2)=T(L,K2)-K5:T(L,K4)=T(L,K4)+K5:T
(L,K5)=T(L,K5)+K10:T(L,K6)=T(L,K6)+K5
6137 IF Y=K4 THEN T(L,K2)=T(L,K2)+K5:T
(L,K3)=T(L,K3)+K10:T(L,K4)=T(L,K4)+K5:
T(L,K5)=T(L,K5)+K5:T(L,K6)=T(L,K6)+10
6138 IF Y=K4 THEN T(L,K7)=T(L,K7)-K15
6139 IF Y=K5 THEN T(L,K1)=T(L,K1)+K10:
T(L,K5)=T(L,K5)+K5
6140 IF Y=K6 THEN T(L,K1)=T(L,K1)+K5:T
(L,K2)=T(L,K2)+K5:T(L,K3)=T(L,K3)+K5:T
(L,K4)=T(L,K4)+K10
6141 IF Y=K6 THEN T(L,K5)=T(L,K5)+K10+
K5:T(L,K6)=T(L,6)+K5:T(L,K7)=T(L,K7)-K
10+K5:T(L,K8)=T(L,K8)-K5
6142 IF Y=K7 THEN T(L,K1)=T(L,K1)-K5:T
(L,K2)=T(L,K2)+K5:T(L,K3)=T(L,K3)+K5
6143 IF Y=K7 THEN T(L,K6)=T(L,K6)+K5:T
(L,K7)=T(L,K7)+K5:T(L,K8)=T(L,K8)-K10
6200 RETURN
7000 AI=INT(2.5*B-16):AW=INT(1.5*C-16)
:AC=INT(0.5*F-16)
7001 IF AI<0 THEN AI=0
7002 IF AW<0 THEN AW=0
7003 IF AC<0 THEN AC=0
7004 AT=AI+AW+AC
7005 PS=INT(K13*RND(K1)+AT+1):IF PS>=K
13 THEN ? #6:? #6;" ";N$;" HAS":? #6;"
 PSIONIC ABILITY"
7010 AI=B-12:AW=C-12:AC=F-12:IF AI<0 T
HEN AI=K0
7011 IF AW<K0 THEN AW=K0
7012 IF AC<K0 THEN AC=K0
7013 AT=AI+AW+AC
7015 MP=K0:OT=K0:IF B>16 THEN OT=OT+K1
7020 IF C>16 THEN OT=OT+K1
7025 IF F>16 THEN OT=OT+K1
7030 IF OT=K2 THEN MP=K2
7035 IF OT=K3 THEN MP=K4
7040 PST=INT(K13*RND(K1)+K1)+AT*MP
7045 IF PS>=K13 THEN ? "↰PSIONIC ABILI
TY = ";PST*K2
7050 IF PS>=K13 THEN ? "PSIONIC STRENG
TH = ";PST:FOR I=1 TO 2000:NEXT I
7055 RETURN
8000 ERLN=256*PEEK(187)+PEEK(186)
8010 CUR=PEEK(90):? "↰":? "INPUT ERROR
 -- TRY AGAIN!":FOR I=1 TO 50:SOUND 0,
I+50,10,8:NEXT I:SOUND 0,0,0,0
8020 TRAP 8000:GOTO (ERLN)
```

●

D: CHECK DATA

```
5 DATA 400,646,24,906,119,911,716,75,1
64,290,307,670,642,113,136,6119
75 DATA 779,348,859,149,820,400,652,75
8,186,896,850,242,202,207,917,8265
105 DATA 903,725,461,652,209,769,870,7
01,764,756,574,291,872,160,303,9010
192 DATA 58,726,366,178,968,815,74,543
,819,281,817,316,185,228,554,6928
236 DATA 17,592,960,448,787,640,77,379
,339,65,53,57,65,49,159,4687
253 DATA 226,202,157,821,503,806,892,7
77,10,461,944,962,666,39,207,7673
316 DATA 446,723,17,108,729,278,386,70
9,448,93,33,372,872,107,590,5911
337 DATA 299,96,73,810,477,173,779,881
,793,703,44,547,489,863,598,7625
362 DATA 448,527,415,170,209,482,846,5
14,477,233,684,438,301,550,309,6603
390 DATA 435,380,205,562,632,334,255,6
76,600,392,577,408,730,506,605,7297
432 DATA 984,355,701,260,832,914,524,5
02,421,705,217,512,613,796,508,8844
463 DATA 627,210,381,674,664,399,486,5
55,539,959,323,115,500,807,680,7919
505 DATA 780,690,788,762,468,16,267,66
6,897,20,554,820,803,813,739,9083
559 DATA 708,559,582,799,803,810,269,6
0,69,860,995,100,69,311,859,7853
574 DATA 10,68,884,452,399,29,871,478,
942,502,57,867,756,106,427,6848
2018 DATA 596,287,108,594,857,950,185,
815,380,387,238,613,447,823,75,7355
2605 DATA 76,45,48,13,973,873,846,383,
996,986,143,774,674,402,667,7899
2670 DATA 713,688,778,956,37,390,436,8
20,746,687,405,379,571,412,805,8823
4999 DATA 583,814,27,652,770,728,791,3
22,301,306,300,601,446,527,514,7682
5240 DATA 534,807,719,189,228,728,81,9
55,647,880,818,643,951,893,497,9570
5515 DATA 236,480,637,651,451,978,841,
812,812,892,983,664,686,152,76,9351
5820 DATA 356,159,814,266,308,433,256,
427,602,659,565,363,805,544,474,7031
5975 DATA 831,175,836,891,881,505,266,
487,429,495,796,449,977,965,273,9256
6200 DATA 637,648,782,802,981,106,280,
900,132,998,251,913,365,378,102,8275
7035 DATA 469,373,54,238,654,33,99,250
,2170
```

●

FIRST and FINEST

In Systems Software for Atari and Apple

MAC/65

First we delivered Atari's Assembler/Editor (the cartridge).

Then we produced our enhanced "EASMD."

Now OSS is introducing the **finest** integrated assembly language development system yet!

In addition to being ideal for writing small, "quick and dirty" subroutines and programs, **MAC/65** shows its full power and speed when used with even the most complex of large assembly language source files.

Naturally, **MAC/65** is upward compatible with both EASMD and the Atari cartridge. And, of course, the object code output is also compatible with OS/A+, Atari DOS, and/or Apple DOS, as appropriate.

MAC/65 .... **$80.00***

OS/A+

Optimized Systems Software — the group that produced **both** the **first** Apple DOS **and** the **first** Atari DOS — now brings you OS/A+, which combines the **finest** features of these and other successful personal computer operating systems.

OS/A+ is the **first** and **finest** operating system available for **both** Apple II and Atari computers and features a keyboard-driven, easy-to-use command processor. In addition to several simple resident commands, OS/A+ allows logical and readable requests for even the most sophisticated utility commands. In fact, the user can even add system commands as desired.

But the real power and flexibility of OS/A+ is its ability to easily interface to devices and **disk drives** of virtually any kind and size. File compatibility (with Apple DOS or Atari DOS, as appropriate), device independence, batch processing, easy of use — OS/A+ truly brings the **finest** in operating systems to **your** computer.

AND NOW OS/A+ (for standard Atari or Apple drives) is included as a part of **every** standard OSS language package. Versions of OS/A+ for some higher capacity drives available at extra cost.

Unless otherwise noted, all OSS products require 48K and at least one disk drive. We recommend 64K for the Apple version of OS/A+.

SpeedRead+

The **first** and still **finest** speed reading tutor designed for **you** to use on **your** computer is available **only** from OSS.

SpeedRead+ uses time-proven techniques to train **you** to instantly recognize words and phrases, and yet it goes far beyond what mere mechanical devices are capable of.

SpeedRead+ exercises your peripheral vision, improves your eye movement and timing, and generally works **with you** at your pace... now and in the future.

NOTE: The Atari version of SpeedRead+ needs only 16K of RAM.

SpeedRead+ **$59.95**

tiny c

As a product of Tiny C Associates, **tiny-c** was the **first** structured language interpreter for microcomputers. Now OSS brings this innovative interpretive language to **your** home computer. While not having the speed and power a true **C** compiler, **tiny-c** is an excellent choice for the programming student who is ready to begin learning the valuable techniques of structured languages.

tiny-c provides an easy-to-use, easy-to-modify environment that encourages experimentation while promoting proper programming style. The **tiny-c** package includes not only a comprehensive and instructional user manual but also **complete source**.

tiny-c **$99.95***

C/65

NOW AVAILABLE!

The **first** native mode **C** compiler ever produced for Atari **and** Apple computers.

C/65 supports a very usable subset of the extremely powerful and popular **C** language. Just as **C** is used by the most sophisticated programmers from the professional and academic communities, so shall **C/65** prove to be a powerful and much-needed tool for 6502 software developers.

C/65 supports integer and character types (and arrays), pointers, fully recursive functions, and much more.

NOTE: C/65 requires MAC/65 or an equivalent assembler. Two disk drives recommended but not required.

C/65 .... **$80.00***

BASIC A+

"From the authors of Atari BASIC..."

It's a fact! OSS gave you that **first** and **most popular** language for Atari Home Computers. But why be content with the **first** when you can have the **finest**?

BASIC A+ is the **only** logical upgrade available to the Atari BASIC programmer. While retaining **all** the features which make Atari BASIC so easy to use, we've **also** given BASIC A+ features that place it at the forefront of modern interpretive languages. BASIC A+ will let **you** explore the worlds of structured programming, superior input/output, helpful programming aids, and even a very comprehensive PRINT USING command. **And**, exclusively for the Atari computer, an almost unbelievable array of PLAYER/MISSILE GRAPHICS commands and functions.

BASIC A+ **$80.00***

***REMEMBER:** Standard OS/A+ is **included** at no extra charge with BASIC A+, MAC/65, C/65, and tiny-c.

ATARI, APPLE II, and TINY C are trademarks of Atari, Inc., Apple Computer, Inc., and Tiny C Associates, respectively. ***SpeedRead***+, MAC/65, C/65, BASIC A+ and OS/A+ are trademarks of Optimized Systems Software, Inc.

Optimized Systems Software, Inc., 10379 Lansdale Ave., Cupertino, CA 95014, (408) 446-3099

# NEW FOR ATARI
FROM
MMG MICRO SOFTWARE

*********NECESSITIES*********

DISK COMMANDER - Just save this program on your BASIC disks and it will autoboot and automatically list all programs from the disk into your screen. Simply run any program by typing in a number.
Requires 16K, Disk Only $24.95

BASIC COMMANDER - This all machine language program is an absolute requirement for ATARI BASIC programmers. Single keystroke DOS and BASIC commands, plus; AUTONUMBER, RENUMBER, BLOCKDELETE and much more!
Requires 16K, Disk Only $34.95

RAM TEST - The most thorough and fastest memory test available for the ATARI. This all machine language program takes 4 min. to test 48K. It's the only program that tests the cartridge area of RAM. Good for new 400/800 computer owners and for testing new RAM boards.
Requires 8K - Disk or Cassette $24.95

******** BUSINESS/HOME*********

MAILING LIST - Extremely fast BASIC and machine language program. Each data disk holds over 500 files. Sort on any of 6 fields at machine language speed. Use any size labels or envelopes.
Requires 48K, Disk Only $39.95

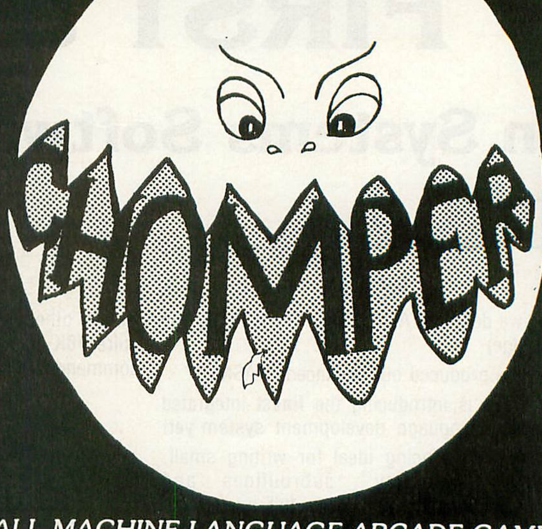
*********** TUTORIALS***********

ASTEROID MINERS - A unique game tutorial. A 32K BASIC game utilizing over 25 players in player-missile graphics, machine language subroutines, a redefined character set, multiprocessing utilizing the vertical blank interrupt interval, and much more! Comes with a book documenting each part of the entire program, and fully documented source code for both the BASIC and assembly language parts of the program. Use these routines in your own programs. These examples will make it easy.
Requires 32K, 1 Joystick - Cassette or Disk $34.95

Dealers and Distributors Contact:

CLASSIC SOFTWARE, INC.
RD1 - 3D •HIGHWAY 34 • MATAWAN, NJ 07747

(201) 566-5007



ALL MACHINE LANGUAGE ARCADE GAME WITH INTELLIGENT MONSTERS!

This arcade style game is sure to become an ATARI classic. Chomper requires 16K RAM, 1 joystick and nerves of steel.
Available on Disk or Cassette $29.95

Available at your favorite computer store or
Send a check or money order directly to:

MMG MICRO SOFTWARE
P.O. BOX 131 • MARLBORO, NJ 07746
or call **(201) 431-3472**
for MasterCard, Visa or COD deliveries

ATARI is a registered trademark of ATARI, INC.   N.J. Residents add 5% sales tax.

THE MONKEY WRENCH™ FOR ATARI

A BASIC and machine language programmers aid for 800 users. Plugs into right slot and works with ATARI BASIC. Adds 9 new direct mode commands including auto line numbering, delete lines, change margins, memory test, renumber BASIC, hex/dec conversion, cursor exchange, and machine language monitor.

$49.95
(See the many reviews.)

The monitor contains 15 commands used to interact with the 6502. Some are display memory/registers, disassemble, hunt, compare, hex/dec convert, transfer memory. Uses screen editing.

MACHINE LANGUAGE MONITOR FOR ATARI USERS

A must for anyone interested in machine language and wanting to know more about how the computer works. Provides 21 easy to use commands. Uses screen editing. Cassette – $24.95 Diskette – $29.95

TYPING EXERCISE

A great educational program. Consists of typing drills and timed typing tests. Diskette – $12.95

ATARI AND PET EPROM PROGRAMMER

Programs 2716 and 2532 EPROMs. Includes hardware and software. PET = $75.00 – ATARI (includes sophisticated machine language monitor) = $119.95

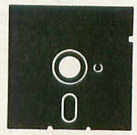
EPROM BOARD FOR ATARI

The EPROM board plug into the cartridge slots. Uses 2716, 2532, and other EPROMs. May contain up to 8K. A must for making your own machine language programs. $19.95

EPROMS – HIGH QUALITY, NOT JUNK

Use with PET, APPLE, ATARI, SYM, AIM, etc. 450 ns. $6.50 for 2716, $12.50 for 2532. We sell EPROM programmers for PET and ATARI – see above.

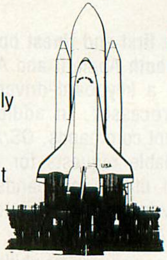
5¼ INCH SOFT SECTORED DISKETTES



Highest quality. We use them on our PETs, APPLEs, ATARIs, and other computers.

$22.50/10 or $44.50/20

MAE SOFTWARE DEVELOPMENT SYSTEM
More than just an Assembler/Editor!



It's a Professionally Designed Software Development System

MAE for PET APPLE ATARI $169.95

Blast off with the software used on the space shuttle project!

- Designed to improve Programmer Productivity.
- Similar syntax and commands –-No need to relearn peculiar syntaxes and commands when you go from PET to APPLE to ATARI.
- Coresident Assembler/Editor – No need to load the Editor then the Assembler then the Editor, etc.
- Also includes Word Processor, Relocating Loader, and much more.
- Options: EPROM Programmer, unimplemented opcode circuitry
- STILL NOT CONVINCED: Send for free spec sheet!

MEMORY TEST

Avoid problems with ATARI RAM cartridges. Don't let your guarantee run out without a test. Cassette – $6.95, Disk – $9.95

Eastern House

3239 Linda Dr.
Winston-Salem, N.C. 27106
(919) 924-2889 (919) 748-8446
Send for free catalog!

VISA MasterCard

THE LEARNING CENTER

"Perhaps the finest educational software that I have ever had the pleasure of reviewing. It's easy for kids to use and effective in teaching basic concepts and skills.... My kids are learning with it right now!"

Fred D'Ignazio, Educational Editor—*Compute!*, Associate Editor—*Softside*,
Author of bestseller—*Katie and the Computer*

We hope you've been using your ATARI for more than just games ... it is, and can be, a valuable educational tool for you and your children.

Bruce and Dianne Mitchell realized this potential and designed a series of programs for use in their Small World Preschool & Kindergarten located in Durham, N.C. Presented on TV's *PM Magazine* these unique educational programs will introduce your 3-9 year olds to the era of learning with computers. Using the graphics and sound capabilities of the Atari, each program develops a particular skill and reinforces correct responses with happy faces and music.

Beginning with basic concepts such as colors, shapes and the alphabet, your child will progress to an understanding of counting, arithmetic, and language skills.

Widely acclaimed, classroom designed and tested, these unique educational tools are now available to you ... *for your children.*

SPECIAL SKILLS	Preschool	K	1	2
* Color For The Non-Reader	●	●		
* Name That Color		●	●	
* Like Shape Identification	●	●	●	
* Different Shape Identification	●	●	●	
Cave Game	●	●	●	●
MATH AND NUMBER SKILLS				
Count With Me	●	●	●	
Number Recognition	●	●	●	●
Addition		●	●	
Subtraction		●	●	
Add.—Vertical/Horizontal		●	●	
Sub.—Vertical/Horizontal		●	●	
Advanced Addition/Subtraction		●	●	●
Ones and Tens			●	●
LANGUAGE SKILLS				
* Alphabet Recognition			●	●
* Letter Sequence	●	●	●	
* Like Symbol Discrimination	●	●		
* Different Symbol Discrimination	●	●		

Pricing Information	Cass.	Disk		Cass.	Disk
Single Program	6.95	9.95	Pre-School	36.95	41.95
Special Skills	24.95	29.95	Kindergarten	54.95	59.95
Language Skills	19.95	24.95	First Grade	49.95	54.95
Math and Number Skills	34.95	39.95	Second Grade	21.95	26.95
Complete Set	59.95	64.95			
Edumate™ Light Pen	19.95		Add $2.00 for postage & handling		

* Compatible with our Edumate™ Light Pen
NOTE: All software requires 8K cassette/16K disk

Start competing with your 4-year-old for computer time. Enroll in THE LEARNING CENTER, it could be the best investment you ever make ... *in your children's future.* See your local dealer or order direct.

Available now for the
ATARI 400/800
COMING SOON for the
TRS-80 Model I, III, Color
Apple, VIC, TI-99
Free Catalog Upon Request

PI®

THE PROGRAMMER'S INSTITUTE
—a division of **FUTUREHOUSE™**
P.O. Box 3191 Dept L
Chapel Hill, NC 27514

1-919-489-2198 10 am - 9 pm, Monday - Saturday

MISSING CAPABILITIES IN ATARI BASIC

16K DISK

by Larry Seftor

INTRODUCTION

For those ATARI users whose only experience has been with the ATARI, there must be a sense of mystery about ATARI's so-called "limited" string handling capability. After all, ATARI BASIC will create a very long string and manipulate it with agility. Other ATARI owners have no doubt tried to convert programs from Microsoft BASIC to ATARI BASIC, and have stumbled onto these limitations.

This article will first describe the string handling abilities which the ATARI lacks, and will present a number of possible ways to supply these missing talents. One of these methods will be described in detail, and a machine language program will be presented which will add new capabilities to ATARI BASIC.

MISSING CAPABILITIES

While a string in ATARI BASIC must be dimensioned, it is not a true array of strings. That is, a true string array has a number of elements, each of which is a string itself. As an example of this capability, a teacher might want to write a program in which array N$(30) contains the first names of her 30 pupils. To print the name of the Ith pupil, the program need only have a statement:

Print N$(I).

While the BASIC in this case is very simple, the internal work involved is great. Consider that each of the 30 elements might possibly have a different length. Somehow, a lot of bookkeeping must be performed to determine where in memory each element is stored. ATARI BASIC simply does not have the internal code to perform this work.

POSSIBLE SOLUTIONS

It is clear that anything done in one BASIC can be done in another. The resulting code may, however, be inefficient, wasteful, and messy. Therefore, while three possible solutions are discussed below, only one is really attractive.

It is useful at this point to discuss ATARI arithmetic. All arithmetic in ATARI BASIC is done in floating point; there is no facility for doing integer arithmetic. When integers are required, for an array index for example, a floating point to integer conversion must be performed. Since floating point operations are much more complicated than integer operations, any BASIC code to calculate these indices is inherently inefficient.

Solution I

The first possible solution for creating a multi-element string array is to take a large ATARI string array and subdivide it into a number of equally sized sections. The programmer then does the bookkeeping explicitly. For example, if the large string is S$, and the length of each section is L, then the Ith element can be written as:

S$((I-L) *L+1, I*L).

This method is probably the most commonly used and straightforward approach. Nevertheless, it has two fundamental problems. First, 4 arithmetic operations (in floating point) are required to find the two indices. Second, every substring must have the length of the longest substring. This is undesirable for a number of reasons. It is wasteful since any substring of length less than L will have unused bytes of memory. In the example of children's names given above, one Clementine in the class would have a dramatic effect on the amount of string space required. Each element will be 10 bytes long, even for Al or Bob.

Having substrings of equal length also complicates coding. Each element must be tested upon entry to make sure it is not too long. Next, if its length is less than L, it must be padded with blanks. And finally, before it can be used, the blanks need to be stripped off.

Solution II

The second method one can use to mock up an array of strings is to maintain a table of information about where each substring lies in a large string array. Such a table might be a numeric array which contains the starting location of each substring in the large array. For example, if S$ is the large string array and POS is an array of starting locations, the Ith substring could be written:

S$ (POS (I), POS (I+1) -1).

On the surface this is an elegant solution. Each substring can be any length, with no string space wasted. However, one must consider the space required for the table. Six bytes in the table are required for each substring. Also, there is the cost of doing ATARI arithmetic. One subtraction must be performed in floating point, and then two floating point numbers must be converted to integers.

The lack of flexibility in this approach turns out to be more of a problem. Going back to our example, if pupil #2, Larry, decides that he wants to be called

Laurence, things get complicated. All the elements in the large string array have to be shifted using BASIC operations, and the values in the numeric array have to be adjusted.

Nevertheless, if one must deal with substrings of varied lengths, and if there is no need to overwrite an element, the method could have advantages over method one.

Solution III

The third possibility, and the one developed here, is based on a specific data structure. The extra information (data descriptors) required to have a string array will be embedded within the string data. Two bytes of overhead will be required for each substring, but there will be no cost for unused array elements. As will be seen below, this approach provides the user with a great degree of flexibility. More importantly, this approach doesn't suffer from the disadvantages of the two methods described earlier.

This approach could be used either from BASIC or by use of a machine language routine. For reasons to be described below, the machine language routine is much more desirable. Indeed, the required manipulations would be most cumbersome, if performed from BASIC.

The data structure is shown in figure 1. The first byte (N_1) specifies which of the array substrings is to follow. This value can run from 1 to 254. The second byte (L_1) provides the length of the substring (up to 253). These two overhead bytes are followed by the actual L_1 bytes of string data. This is one complete data element. It may be followed by additional data elements, or by an end-of-string mark.

Example 1 shows a string array with two elements. Element 88 is 8 characters long and consists of the string 'LAURENCE'. The second substring is element 3, the 5 byte lone string 'LINDA'.

If element 88 is overwritten, example 2 results. The original string 88 is destroyed and everything else, including data descriptors, is shifted 10 bytes ($10 = L_1 + 2$) toward the front of the string space. The new string 88, 'NEIL', is placed at the end of the string space. In the process of going to a shorter substring, 4 bytes of memory are recovered.

IMPLEMENTATION

There is the obvious advantage of speed in writing any software in machine language. This is particularly true in this case, since many of the operations required are simpler when written in assembler. Perhaps the greatest advantage, however, lies in the ease of use. No special BASIC code must be written for these operations.

The program is designed to be placed in page six of memory. Since BASIC doesn't know about page six, no memory is removed from program space. In addition, once the program is put into place, unless deliberately written over, it will remain until the computer is turned off. Hitting 'System Reset' or typing NEW or CLOAD will not erase the program.

Listing 1 contains a BASIC program which will load the machine language array handler. Those readers interested in details of the machine language will have no trouble disassembling the program from this listing.

In typical use, the program of listing 1 will be loaded and RUN. After completion the user can type NEW and enter a new program, or may CLOAD a program from tape. Listing 2 shows a sample program which demonstrates use of the array handler.

This listing contains all aspects of using the machine language string array program. It, therefore, merits close study. Line 20 sets the values of a number of variables. It is here that both the location of the machine language subroutines, and the size of the string space are set. Line 30 tells BASIC to reserve a certain region of memory for use by the array handler. An initialization call is performed in line 40 which tells the machine language program where this space is located.

Line 50 is the start of a simple program which will either read data into the string array or print onto the screen one of the array substrings. Line 85 contains the machine language call for a write, while the read call is in line 140. Since characters entered on input go directly into memory, the usual editing (e.g., use of the cursor keys) on input is not available. To correct an incorrectly input character, type 'DEL'. This will print the deleted character in inverse video. If the delete key backs the input behind the starting position, input will be terminated and a null string written. This can be used to eliminate any unwanted strings, for the recovery of string space.

The string defined in line 30 will not be a true string from the view of ATARI BASIC. the 'LEN' function will not work, and no error checking is performed. Therefore, if one is not careful, it is possible to write beyond the end of the string space as reserved by BASIC, doing unknown damage. For this reason, line 200 is important. The variables SPACELEFT and SPACEUSED are defined according to their names. If the programmer checks the value of SPACELEFT before each read, to determine whether adequate space is left, the danger of crashing storage will become greatly reduced. Lines 210 and 220 print out these two quantities, for demonstration purposes.

While the usefulness of the array handler is apparent from running the above program, there is an important application which is as yet unaddressed. Suppose that in the classroom example above, the teacher didn't want to type in the pupils' names each tme the program was run. In this case, it would be useful to have the required information in DATA statements.

Because of size limitation, the machine language program cannot handle such input. But it is possible to use a BASIC subprogram to set up a proper initial configuration of the string array. It should be stressed that this is for initialization, and cannot be used subsequently in the program. Listing 3 is an example of this technique. Lines 70 through 140 consist of the actual array initialization routine.

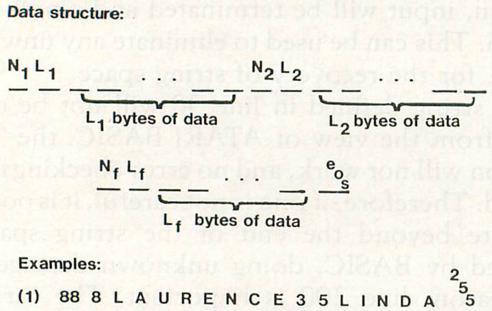
CONCLUSIONS

The BASIC interpreter proves a means to protect the programmer from the pitfalls of using a microprocessor. If one makes a mistake, the computer will usually print a diagnostic message rather than crash. Unfortunately, because the array handler must fit into 256 bytes, there is no room for such error checking capability. The programmer must take care. Some obvious errors are:

(1) writing beyond the end of the string space,
(2) entering a line larger than 253 bytes,
(3) specifying an element which doesn't lie between 1 and 254.

There are no doubt others. Careful use, however, will avoid such pitfalls.

There are still some array functions missing. For example, what is the third letter in the 5th string? Fortunately, there is a natural extension of this array handling method which will handle such tasks. However, since page six is full, the implementation will have to be handled differently. Such a program is currently under development. □



LISTING 1

```
10 FOR I=0 TO 251
20 READ V
30 POKE 1536+I,V
40 NEXT I
1000 DATA 216,104,24,165,140,109,254,6
,133,212
1010 DATA 165,141,109,255,6,133,213,10
4,133,217
1020 DATA 104,133,216,104,104,133,214,
32,61,6
1030 DATA 108,216,0,216,104,104,133,21
3,104,133
1040 DATA 212,56,229,140,141,254,6,165
.213,229
1050 DATA 141,141,255,6,160,0,169,255,
145,212
1060 DATA 96,160,0,177,212,201,255,208
,2,56
1070 DATA 96,197,214,208,2,24,96,160,1
,24
1080 DATA 177,212,105,2,144,3,230,213,
24,101
1090 DATA 212,133,212,144,222,230,213,
76,61,6
1100 DATA 176,22,160,1,177,212,133,215
,200,177
1110 DATA 212,162,22,132,216,32,125,6,
164,216
1120 DATA 198,215,208,240,96,168,189,1
,228,72
1130 DATA 189,0,228,72,152,96,160,0,13
2,216
1140 DATA 176,30,200,177,212,168,200,2
00,162,0
1150 DATA 177,212,201,255,240,16,129,2
12,24,165
1160 DATA 212,105,1,133,212,144,239,23
0,213,76
1170 DATA 150,6,169,63,162,22,32,125,6
,160
1180 DATA 0,165,214,145,212,200,132,21
6,162,36
1190 DATA 32,125,6,201,126,208,13,164,
216,177
1200 DATA 212,105,127,198,216,240,38,7
6,220,6
1210 DATA 201,155,240,14,230,216,164,2
16,145,212
1220 DATA 162,22,32,125,6,76,188,6,162
,22
1230 DATA 32,125,6,160,1,198,216,165,2
16,145
1240 DATA 212,230,216,230,216,164,216,
169,255,145
1250 DATA 212,96
```

●

D: CHECK DATA

```
10 DATA 65,340,92,378,767,874,763,993,
851,838,535,375,578,778,833,9060
1110 DATA 812,34,630,799,904,64,439,54
,774,827,241,573,927,246,893,8217
```

●

LISTING 2

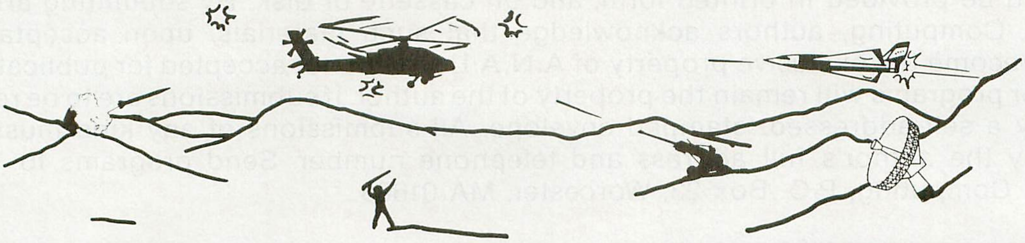
```
20 SIZE=2000:SET2=1569:WRITE=1636:REED
=1672
30 DIM S$(SIZE)
40 X=USR(SET2,ADR(S$))
45 TRAP 50
50 ? "K↓↓ The Seftor ARRAY HANDLER"
60 ? "↓↓  Input(I) or Output(O)";
65 L=PEEK(764):IF L=255 THEN 65
70 POKE 764,255:IF L<>8 AND L<>13 THEN
 65
75 ? :? "↓  Which element";:INPUT I:GO
TO L*10
80 ? "↓↓  The Message is:":?
85 X=USR(1536,WRITE,I):? :GOTO 200
130 ? "↓  Enter your message:↓"
140 X=USR(1536,REED,I):? :?
200 SPACEUSED=USR(1536,WRITE,0)-ADR(S$
):SPACELEFT=SIZE-SPACEUSED
210 ? "↓↓  You have used ";SPACEUSED;"
 bytes."
220 ? "  There are ";SPACELEFT;" left.
"
230 ? "↓↓  Press any key to CONTINUE";
240 L=PEEK(764):IF L<>255 THEN POKE 76
4,255:GOTO 50
250 GOTO 240
```

●

D:CHECK and other programs continued on page 93.

CHOPPER RESCUE

Zip up your flying suit and strap yourself into your sleek assault chopper: CHOPPER RESCUE challenges your skill and daring. Pilot your nimble craft through the treacherous underground labyrinth. Search out and rescue the captives trapped inside. Danger approaches from all sides as defenders fire winged missiles and sparkling energy bursts at your fragile craft. But you are not defenseless, your explosive tipped rapid-fire missles dart out to blast a path through the dense minefields. Thread your bomb directly into the heart of their pulsating nuclear furnace. And just when you think the path is clear, a swift patrol drone appears from behind a ledge, spitting missles as you scramble for a firing position. Immerse yourself in the non-stop excitement of CHOPPER RESCUE, you'll keep coming back for more!



CHOPPER RESCUE is a brand new game from MicroProse Software. Fully animated graphics, music, sound effects, and scrolling make this a showpiece for the ATARI's leading edge capabilities. CHOPPER RESCUE features three scenarios multiple skill levels, and competition for 1 or 2 players. No two games are ever alike. MicroProse is dedicated to bringing you games which challenge you far beyond the first few plays. We promise you hours of excitement and pleasure.

CHOPPER RESCUE (ATARI 32K, disk) $29.95

Other MicroProse games . . .

HELLCAT ACE - A real time, three dimensional WWII flying simulation with a unique pilot's eye view $29.95

FLOYD OF THE JUNGLE - A fun game for up to four players. Guide Floyd through the jungle hazards and reunite him with lovely Janice $29.95

To Order C.O.D., Visa or Master Card Call:
(301) 357-4920 or Send Check or Money Order To:

MicroProse Software
One Caribou Court
Parkton, MD. 21120

(Add $2.00 for Postage & Handling for Each Order)

ATARI is a Register Trademark of ATARI, Inc.

WHY PAY MORE?

FOR ENTERTAINING SOFTWARE FOR YOUR ATARI 400/800*

compare prices!

ALL SOFTWARE AT LEAST 20% OFF RETAIL PRICES — ALL THE TIME! 30-40% OFF ON SPECIAL SALE SOFTWARE!

compare selection!

MORE THAN 100 DIFFERENT GAMES, SIMULATIONS, PUZZLES, & ADVENTURES FROM ARTWORX, ADVENTURE INTERNATIONAL, EPYX, AVALON HILL, ONLINE, DATASOFT, ARCADE PLUS, IDSI, GEBELLI & MORE!

compare service!

FREE CATALOG · FREE NEWSLETTER · FREE SHIPPING ON ORDERS OVER $150 · FREE MASTERCARD AND VISA USE · MOST ORDERS SHIPPED SAME DAY RECEIVED OR NEXT DAY · ONLY $2.00 SHIPPING AND HANDLING PER ORDER

BOX 247 NEW FLORENCE, PA 15944
(412) 235-2970

mideastern software

PA residents add 6% sales tax

*trademark of Atari Inc.

ATTENTION PROGRAMMERS!

A.N.A.L.O.G. Computing is interested in programs, articles, and software review submissions dealing with the ATARI 400 and 800 home computers. If you feel that you can write as well as you can program, then submit those articles and reviews that have been floating around in your head, awaiting publication. This is your opportunity to share your knowledge with the growing family of ATARI computer owners. A.N.A.L.O.G. pays comparable to many other computer magazines. All submissions for publication must be typed, upper and lower case with double spacing. Program listings should be provided in printed form, and on cassette or disk. By submitting articles to A.N.A.L.O.G. Computing, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of A.N.A.L.O.G. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply a self-addressed, stamped envelope. All submissions of any kind must be accompanied by the author's full address and telephone number. Send programs to: Editor, A.N.A.L.O.G. Computing, P.O. Box 23, Worcester, MA 01603.

```
0 REM ***********************
1 REM *                     *
2 REM *    ATARI SYMBOL     *
3 REM *  BY CRAIG WEISS     *
4 REM *                     *
5 REM ***********************
10 GRAPHICS 24:COLOR 1:POKE 559,0
20 R=0
24 REM
25 REM *** PLOT STRAIGHT LINES ***
26 REM
30 READ W,X,Y,Z:PLOT W,X:DRAWTO Y,Z:R=
R+1
100 DATA 144,13,144,76,144,13,156,15,1
44,13,128,28,156,15,156,88,160,16,156,
20,160,16,160,176
110 DATA 160,16,180,20,180,20,180,176,
184,21,180,24,184,21,194,24,194,24,194
,84,240,154,240,172
120 DATA 240,172,220,172,180,176,160,1
76,160,176,144,176,144,176,144,144,88,
180,68,180,88,180,88,160
130 DATA 68,180,68,160,68,160,88,160,1
28,28,128,76,184,21,184,84
134 REM
135 REM *** PLOT FALSE CURVES ***
136 REM
140 DATA 128,77,126.5,94,126.5,94,124,
108,124,108,120,122,120,122,112,137,11
2,137,104,145
150 DATA 104,145,96,150,96,150,88,155,
88,155,80,158,80,158,72,160
160 DATA 144,76,142.5,94,142.5,94,140,
108,140,108,135,122,135,122,126,137,12
6,137,120,145
170 DATA 120,145,114,151,114,151,108,1
55.5,108,155.5,100,158.5,100,158.5,88,
160
180 DATA 156,88,153.5,112,153.5,112,15
0,128,150,128,144,144,143,144,136,156,
136,156,124,168
190 DATA 124,168,112,176,112,176,102,1
79,102,179,96,180,96,180,88,180
200 DATA 194,84,194,92,194,92,198,112,
198,112,208,130.5,208,130.5,216,141,21
6,141,224,148
210 DATA 224,148,232,152,232,152,240,1
54
220 DATA 184,84,186,104,186,104,189.5,
120,189.5,120,196,136,196,136,204,148
230 DATA 204,148,216,160,216,160,228,1
68,228,168,240,172
240 DATA 182,122,184,132,184,132,188,1
40,188,140,196,152,196,152,208,164,208
,164,220,172
250 IF R<68 THEN 30
260 IF R=68 THEN 500
310 REM
320 REM FILL
330 REM
400 Q=0
500 READ A,B,C,D:PLOT A,B:POSITION C,D
:Q=Q+1
900 POKE 765,1
910 XIO 18,#6,0,0,"S:"
1000 DATA 144,13,144,76,144,76,142.5,9
4,142.5,94,140,108,140,108,135,122,135
,122,126,137,126,137,120,145
1010 DATA 120,145,114,151,114,151,108,
155.5,108,155.5,100,158.5,100,158.5,88
,160,88,160,88,180
1020 DATA 160,16,160,176,184,21,184,84
1030 DATA 184,84,186,104,186,104,189.5
,120,189.5,120,196,136,196,136,204,148
1040 DATA 204,148,216,160,216,160,228,
168,228,168,239,171
2000 IF Q<20 THEN 500
2010 IF Q=20 THEN 2800
2500 REM
2510 REM *** MACHINE LANGUAGE ***
2520 REM
2800 POKE 559,34:FOR X=1 TO 1000:NEXT
X
3000 FOR I=1664 TO 1673:READ A:POKE I,
A:NEXT I
3010 DATA 232,142,10,212,142,24,208,76
,128,6
3020 ? USR(1664):RETURN
3030 RETURN
```

●

FOR SOME INTERESTING VARIATIONS ON THE LOGO, TRY THESE LINES:

```
10 GRAPHICS 24:SETCOLOR 1,0,0:COLOR 1:
POKE 559,0
```

OR:

```
3000 FOR I=1664 TO 1683:READ A:POKE I,
A:NEXT I
3010 DATA 232,138,41,15,168,138,41,240
,142,10,212,141,24,208,140,23,208,76,1
28,6
```

TO SEE LOGO BEING DRAWN, REMOVE THE 'POKE 559,0' IN LINE 10.

D: CHECK DATA

```
0 DATA 552,194,141,406,200,562,472,130
,43,298,271,873,436,686,546,5810
130 DATA 721,191,487,322,290,982,325,1
47,291,330,204,231,938,956,994,7409
250 DATA 655,676,302,657,87,225,960,96
4,541,600,560,34,78,951,592,7882
2010 DATA 625,292,270,138,870,767,617,
161,957,4697
```

●

D: CHECK DATA

```
20 DATA 775,925,214,358,293,916,760,92
7,690,329,482,724,710,754,677,9534
220 DATA 692,567,382,502,2143
```

●

LISTING 3

```
20 SIZE=2000:SET2=1569:WRITE=1636:REED
=1672
30 DIM S$(SIZE)
40 X=USR(SET2,ADR(S$))
50 REM
60 REM ***INPUT FROM DATA STATEMENTS
65 REM
70 I=0:DIM T$(15)
80 I=I+1
90 READ T$:IF T$="@" THEN 140
100 S$(LEN(S$)+1)=CHR$(I)
110 S$(LEN(S$)+1)=CHR$(LEN(T$))
120 S$(LEN(S$)+1)=T$
130 GOTO 80
140 S$(LEN(S$)+1)=CHR$(255)
150 REM ***
155 TRAP 210
160 REM PRINT THE INITIAL CONFIGURATIO
N
170 GOSUB 300
175 REM ***CHANGE IF REQUESTED
180 ? "Which element would you like to
 change (0 to END)";:INPUT CH
190 IF CH=0 THEN END
195 IF CH<1 OR CH>30 THEN 210
200 ? :? "Input new element ";CH;:X=US
R(1536,REED,CH)
210 TRAP 210:GOSUB 300
220 GOTO 180
300 REM
310 REM ***PRINT THE NAMES
320 REM
325 ? "K↓The student's names are:↓"
327 POKE 752,1
330 FOR J=1 TO I-1 STEP 3
335 ? :? ;:? J;" ";:X=USR(1536,WRITE,J
):ROW=PEEK(90)
340 POSITION 15,ROW:? J+1;" ";:X=USR(1
536,WRITE,J+1)
345 POSITION 29,ROW:? J+2;" ";:X=USR(1
536,WRITE,J+2)
350 NEXT J
355 ? :? :POKE 752,0:RETURN
400 DATA NEIL,LARRY,ALLISON,CARLA,JERR
Y,CAROL
405 DATA CLEMENTINE,DAVID,STEVE,GORDON
,MATT,ALEX
410 DATA DERICK,JOSH,BRYAN,JASON,BRITT
ON,JAN
415 DATA KEN,ERIC,KEVIN,JEFFREY,ADAM,C
HRIS
420 DATA SUSAN,CHAD,LINDA,ROBIN,STEPHA
NIE,MINDA
425 DATA @
```

●

D: CHECK DATA

```
20 DATA 775,925,214,38,207,49,228,194,
721,870,874,587,661,130,341,6814
155 DATA 715,917,801,298,848,908,662,9
45,875,780,300,636,304,868,972,10829
330 DATA 113,217,885,672,403,847,271,7
79,226,44,652,358,5467
```

●

PCA

PRESENTS

LABYRINTHS

FOR ATARI® 400/800

A

MAGICAL,

MYSTICAL,

AND THOROUGHLY ENTERTAINING WORLD OF MONSTERS, DEMONS & TREASURES THAT IS COMPLETELY

DIFFERENT EVERYTIME YOU PLAY!

YOU WILL START AS A FIGHTER, THIEF, CLERIC OR MAGIC-USER. TRY TO GAIN EXPERIENCE & POWER TO ADVANCE TO HIGHER LEVELS OF **POWER.**

Discover GOLD, WEAPONS, ARMOR, and MAGIC SCROLLS among the TREASURES of those you defeat. But **BEWARE...**

Your power leads you to more deadly monsters who may have magic powers!

A wide choice of COMMANDS combined with SOUND EFFECTS makes for unparalleled computer melee!

OVER 35 completely different types of monsters

.... LABYRINTHS ....

CASSETTE - $24.95	DISK - $28.95
(32K required)	(48K required)

Progressive **C**omputer **A**pplications

is setting new standards of quality in microcomputer software.

In this tradition, we are also presenting another supreme simulation ...

COMPUTER EXCHANGE

An OUTSTANDING STOCK MARKET simulation

Solitaire, or up to 4 players trade stocks in a fictional stock market.

One joy stick required per player

.... COMPUTER EXCHANGE ....

CASSETTE - $21.95	DISK - $24.95
(16K required)	(32K required)

....

ORDER FROM :

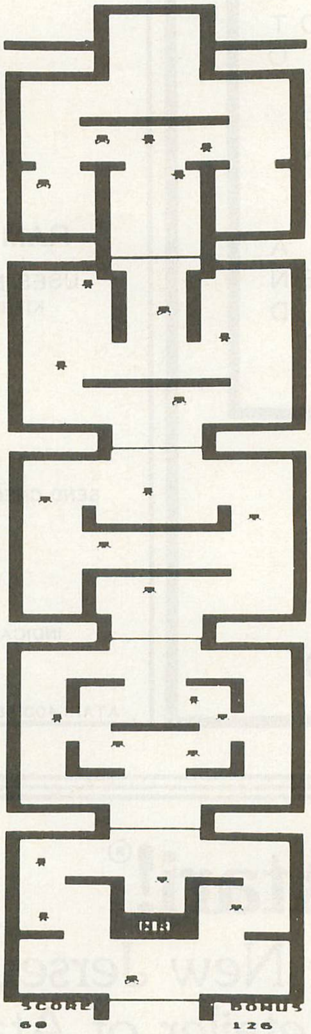
P.C.A.
P.O. BOX 46
BURTONSVILLE, MD. 20866

## Dealer inquires welcome ##

```
    A   S E T C O L O R       D
    B   A     L       E N T E R
  C S A V E   O       A   H   A
  L     E     S O U N D   E   W
D O S         F     E     N O T
  A   S             W         O
  D A T A     D I M     V A L
    I       S   N           E
      C L O G   P O P   R U N
  P O K E   N   U       E
  T     T     S T A T U S
F R E     B Y E     R   T     A
  I                 A   O P E N
  G R A P H I C S   P   R     D
    E     U   L         E N D
C O M   T O   R N D
```

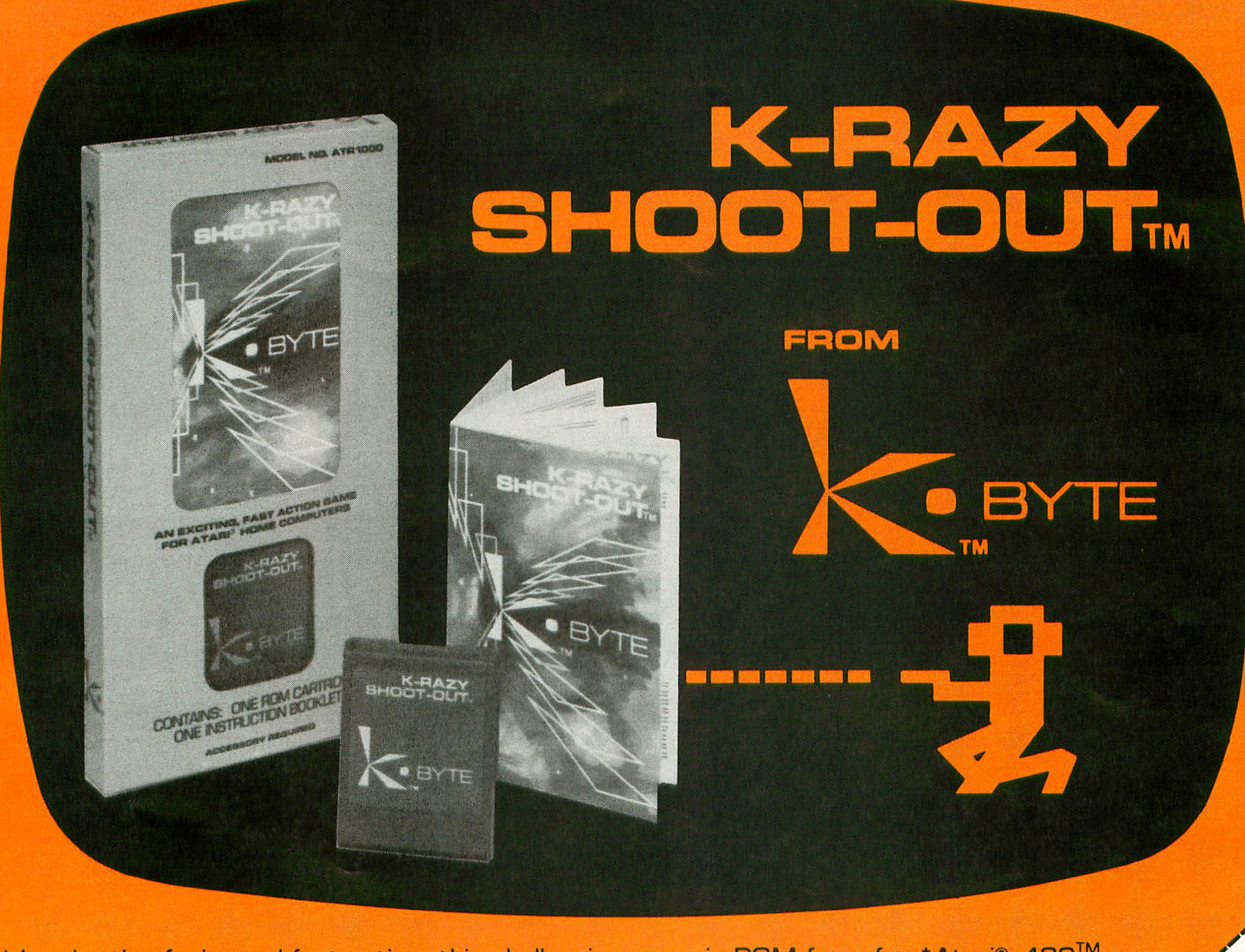
BASIC CROSSWORD PUZZLE ANSWERS

(A.N.A.L.O.G. Computing No. 7)



INDEX TO ADVERTISERS

READY FOR A
REAL CHALLENGE?



It's a battle of wits and fast action, this challenging game in ROM form for *Atari® 400™ or Atari® 800™ Personal Computer Systems. You're trapped in an Alien Fortress. Your mission is to clear your sector of the Alien Droids and progress to the next sector. The first sector seems easy enough... BUT–WATCH-OUT... as you progress from sector to sector, the Droids become more numerous and aggressive. A great challenge and fun too!

K-RAZY SHOOT-OUT is destined to be a favorite of Atari enthusiasts across the country with features like:

- Solid-state ROM pack
- Seven progressive levels of play
- Every game different
- Bonuses and penalties
- Full-color T.V. graphics and space-age sound
- Full-color instruction booklet
- Millions of variations
- A constant challenge

[WARNING: Playing the game could be HABIT forming]

Now at your Atari Dealer!

Coming soon:

- K-DOS™
- K-STAR PATROL™
- K-RAZY KRITTERS™

*Atari is the Registered Trademark of Atari, Inc.

FOLD & TEAR

If not available in your area, send a check or money order directly to:
K-BYTE • P.O. Box 456 • 1705 Austin Street • Troy, Michigan 48099
Toll Free Phone Orders: **1-800-521-0560** • In Michigan: **(313) 524-9878**
□ $49.95, plus $1.50 shipping and handling, Michigan residents add 4% sales tax.
□ $1.00 for instruction book only.

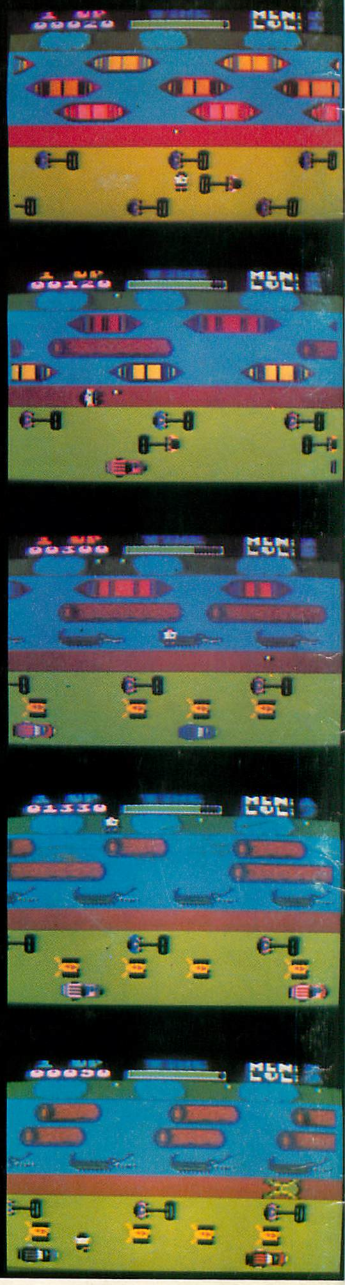
NAME ______________________
ADDRESS ______________________
CITY ______________ **STATE** ______________ **ZIP** ______________

Please allow 3 weeks for delivery.

MasterCard VISA □ **MasterCard** □ **VISA**

CHARGE CARD NO. ______________________
EXPIRATION DATE ______________________

by Russ Wetmore



Good Golly! What A GAME!

The original arcade game **PREPPIE!** will give you hours of toe-tickling fun. **You** take the controls, moving your young prepster about on the golf course. And what a golf course! It's unlike any you have ever played on before. There are dangers **everywhere,** and only you can keep the little Ivy League from a fate worse then Penn State!

PREPPIE! is written in state-of-the-art machine language, is joystick compatible and fully utilizes the Atari's sound and graphics capabilities. Quite frankly, it will give you the most fun you can have on an Atari microcomputer.

PREPPIE! is available at fine computer, book, and hobby stores everywhere.

ATARI 16K TAPE — $29.95
ATARI 32K DISK — $29.95

An exclusive game from

ai
Adventure
INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC.

To order, see your local dealer. If he does not have the program you want, then call 1-800-327-7172 (orders only) or write for our free catalog.

Published by ADVENTURE INTERNATIONAL
a division of Scott Adams, Inc.
BOX 3435 • LONGWOOD, FL 32750 • (305) 862-6917

"My Atari never did things like this before!"
—Holister Townsend Wolfe

"I had so much fun I almost blew my doughnuts."
—Theodore Boston III

"I haven't had this much fun since Buffy and I went to Princeton for the weekend."
—Martha Vineyard