

TYPE-IN SOFTWARE!

ANALOG

COMPUTING

T.M.

APRIL 1989
ISSUE 71

USA \$3.95
CANADA \$4.95

KRAZY MAZES
PIXEL AVERAGING
UNIVERT

FDC 50075

N77067TARXMI35D 904 8909
WILLIAM TART
13502 NORTHBOROUGH DR
#619
HOUSTON TX 77067
*
*

REVIEWS:
THE CONVERTER
CHEAT!
TALLADEGA





BREAK AWAY FROM THE PACK



The world of ATARI-ST continues to grow by leaps and bounds, and ST-LOG is there every step of the way! We stand apart from the competition by offering more color, comprehensive reviews and in-depth features. **SUBSCRIBE NOW!**

FILL OUT COUPON ON PAGE 53 TODAY!

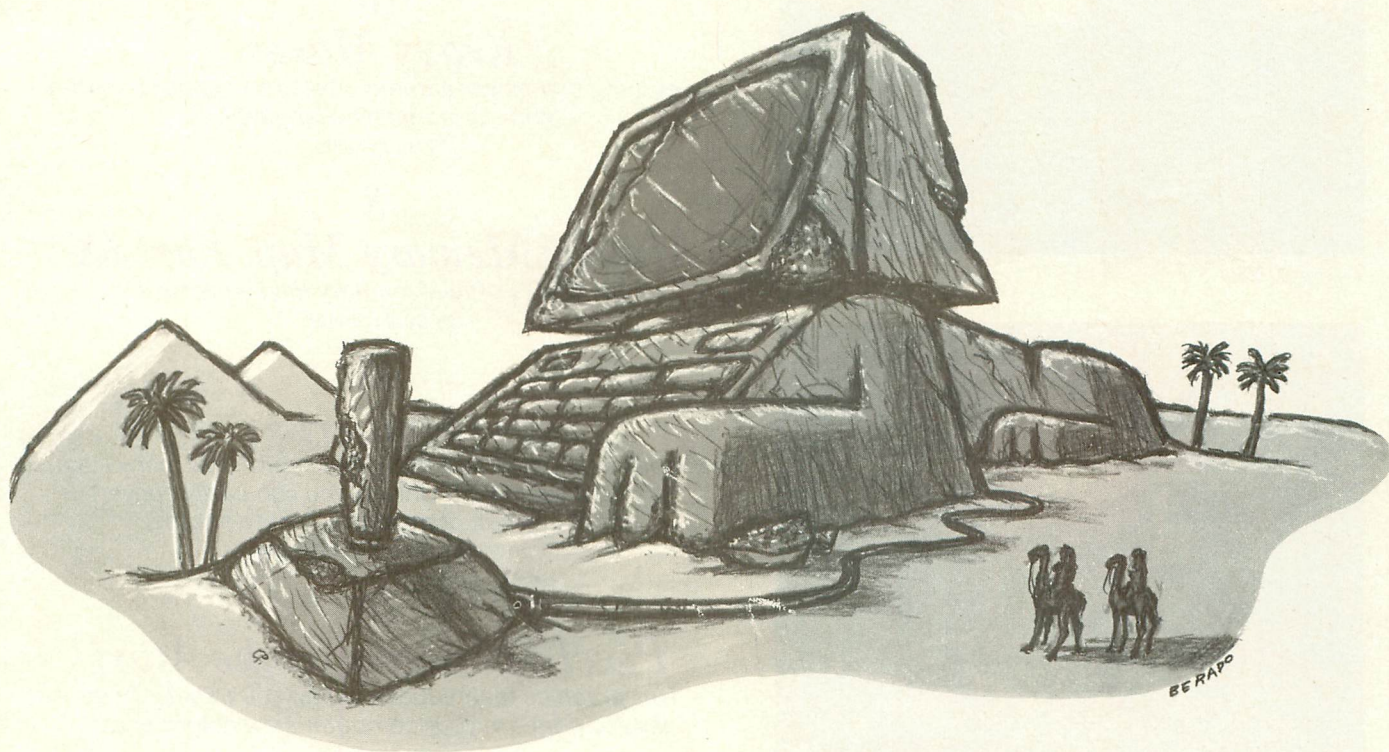
12 Issues \$28

\$19 OFF THE COVER PRICE

12 Issues with Disk \$79

NEW LOWER PRICE

Editorial



by Clayton Walnum

Every day, more and more 8-bit computer owners are trading up to STs. Needless to say, this has had many effects on the Atari 8-bit computer market. For one thing, with less users to support them, software publishers have not been willing to risk their development dollars on programs for the 8-bit line of computers. They are afraid that they won't be able to sell enough copies of their programs to make the effort profitable.

But there's one software publisher that is still putting out software regularly—in fact, one that publishes many programs a month. These programs include utilities, games and applications, just to mention a few types. That publisher is, of course, ANALOG Computing, whose commitment to the 8-bit Atari market is evidenced by the magazine you're

now holding in your hands.

But we'd be lying if we told you that things hadn't changed at ANALOG, as well. Just like everyone else, we've seen interest in the Atari 8-bit line of computers drop over the last few years. One of the ways that declining interest affects magazines like ANALOG is in the number of program submissions we receive. Today we receive far fewer submissions than was ordinary as little as a year ago. It seems that, just as users are moving on to the STs, so are the programmers.

What does this mean to you? It means that if you've been thinking about submitting an article but have been putting it off because you were afraid of the competition, this might be the best time. I'm not saying that there is *no* competition. But programmers have a far better chance of seeing their work in print now, than they have for many years.

So if you've been thinking about submitting something, please do. It's easy. Print out

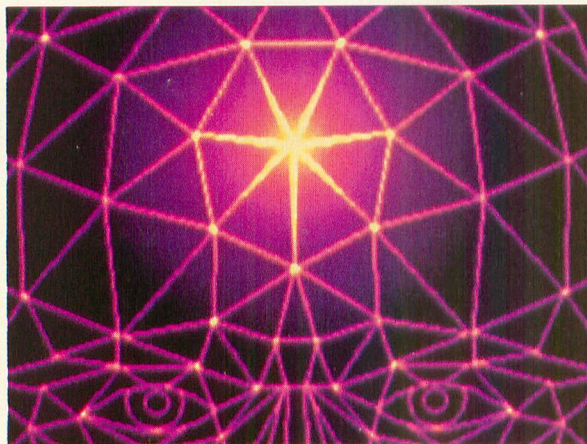
a copy of your program, then write up an article explaining it and print it out in double-spaced manuscript form; then place a copy of the article (assuming you've used a word processor) and your program on a disk containing DOS (only Atari DOS, please) and mail it to ANALOG Computing, P.O. Box 1413-M.O., Manchester, CT 06040-1413. That's all there is to it.

Usually, you will be notified of our decision within two weeks. And we pay well for articles that are accepted, with a check being mailed to you within 30 days of your returning the contract. What a great way to use your Atari!

In any case, ANALOG Computing plans to be around for a long time, and even though it's gotten a little tougher to come up with the quality programs you're used to seeing in these pages, it certainly hasn't become impossible. In the months to come, we've got some great things lined up. You just wait and see. **A**



on page 68



on page 8



on page 16

F E A T U R E S

16

Krazy Mazes

An exciting two-player game of mazes and chases written entirely in machine language.

by Barry Kolbe

8

Master Memory Map, Part IX

ANALOG's official memory map continues

by Robin Sherer

68

Univert

How many decimeters are there in a cubit? How many leagues in a light year? With Univert you can easily convert from any unit of measurement to another.

by William Frasz & Reid Brockway

76

Pixel Averaging on the Atari

This graphics technique will allow you to hide those jagged edges in your computer art masterpieces.

by Stephen Miller

ANALOG COMPUTING STAFF

Publisher
LEE H. PAPPAS

Executive Editor
CLAYTON WALNUM

Art Director
KRISTEL PECKHAM

Associate Editor
ANDY EDDY

Managing Editor
DEAN BRIERLY

East Coast Editor
ARTHUR LEYENBERGER

West Coast Editor
CHARLES F. JOHNSON

Contributing Editors
LEE S. BRILLIANT, M.D.;
MICHAEL BANKS; FRANK COHEN;
MAURICE MOLYNEAUX;
STEVE PANAK; CRAIG PATCHETT;
MATTHEW J. W. RATCLIFF;
ROBIN SHERER; KARL E. WIEGERS

Entertainment Editor
DAVID PLOTKIN

Cover Photography
GARRY BROD

Illustrations
JOHN BERADO
JOHN MCKINNEY

Copy Chief
KATRINA VEIT

Copy Editors
SARAH BELLUM
ANNE DENBOK
RANDOLPH HEARD
PAT ROMERO
KIM TURNER

Editorial Assistant
NORMA EDWARDS

Chief Typographer
KLARISSA CURTIS

Typographers
DAVID BUCHANAN
JUDY VILLANUEVA

Contributors
REID BROCKWAY
WILLIAM FRASZ
BARRY KOLBE
STEPHEN MILLER

Vice President, Production
DONNA HAHNER

Advertising Production Director
JANICE ROSENBLUM

National Advertising Director
JAY EISENBERG
(213) 467-2266

(For regional numbers, see right)

Corporate Ad Director
PAULA THORNTON

Subscriptions Director
IRENE GRADSTEIN

Vice President, Sales
JAMES GUSTAFSON

April 1989
Issue 71

R E V I E W S

10 **Panak Strikes**

This month Steve looks at Gauntlet (Atari) and Richard Petty's Talladega (Cosmi).

by Steve Panak

58 **The Converter**

(No Frills Software)

by Matthew J.W. Ratcliff

67 **Cheat!**

(Alpha Systems)

by Clayton Walnum

C O L U M N S

38 **Database DELPHI**

by Michael A. Banks

42 **Game Design Workshop**

by Craig Patchett

50 **ST Notes**

by Frank Cohen

62 **The End User**

by Arthur Leyenberger

D E P A R T M E N T S

3 **Editorial**

by Clayton Walnum

6 **Reader Comment**

14 **8-bit News**

55 **M/L Editor**

by Clayton Walnum

60 **BASIC Editor II**

by Clayton Walnum

Where to Write

All submissions should be sent to: **ANALOG Computing**, P.O. Box 1413-M.O., Manchester, CT 06040-1413. All other editorial material (letters, press release, etc.) should be sent to: Editor, **ANALOG Computing**, 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210.

Correspondence regarding subscriptions, including problems and changes of address, should be sent to: **ANALOG Computing**, P.O. Box 16927, North Hollywood, CA 91615, or call (818) 760-8983.

Correspondence concerning a regular column should be sent to our editorial address, with the name of the column included in the address.

We cannot reply to all letters in these pages, so if you would like an answer, please enclose a self-addressed, stamped envelope.

An incorrectly addressed letter can be delayed as long as two weeks before reaching the proper destination.

Advertising Sales

Address all advertising materials to:
Paula Thornton — Advertising Production
ANALOG Computing
9171 Wilshire Blvd., Suite 300
Beverly Hills, CA 90210.

Permissions

No portion of this magazine may be reproduced in any form without written permission from the publisher. Many programs are copyrighted and not public domain.

Due, however, to many requests from Atari club libraries and bulletin-board systems, our new policy allows club libraries or individually run BBSs to make certain programs from **ANALOG Computing** available during the month printed on that issue's cover. For example, software from the July issue can be made available July 1.

This does not apply to programs which specifically state that they are not public domain and, thus, are not for public distribution.

In addition, any programs used must state that they are taken from **ANALOG Computing Magazine**. For more information, contact **ANALOG Computing** at (213) 858-7100, ext. 163.

Subscriptions

ANALOG Computing, P.O. Box 16927, North Hollywood, CA 91615; (818) 760-8983. Payable in U.S. funds only. U.S.: \$28-one year, \$54-two years, \$76-three years. Foreign: Add \$7 per year. For disk subscriptions, see the cards at the back of this issue.

Authors

When submitting articles and programs, both program listings and text should be provided in printed and magnetic form, if possible. Typed or printed text copy is mandatory, and should be in upper- and lowercase with double spacing. If a submission is to be returned, please send a self-addressed, stamped envelope.

For further information, write to **ANALOG Computing**, P.O. Box 1413-MO, Manchester, CT 06040-1413.

JE Publishers Representative
6855 Santa Monica Blvd., Suite 200
Los Angeles, CA 90038

Los Angeles	— (213) 467-2266
San Francisco	— (415) 864-3252
Chicago	— (312) 445-2489
Denver	— (303) 595-4331
New York City	— (212) 724-7767

READER COMMENT

Can't Find Atari?

While I was going through the August issue, checking up on another outstanding BBK production, *B-CALC*, my eyes wandered over to the Steve Panak article and the words "Buy Atari..." jumped off the page at me. What Steve and Atari seem to forget is that there is a very large portion of this country that can't just wander down to a local Atari store and see what is available. We poor folks are totally limited to either driving long distances (in my case, in excess of 200 miles) to the nearest Atari dealer, or to mail-order. And now Atari is cutting out the mail-order houses. While we would be more than happy to support Atari, we first have to have the support from Atari. When that happens, Atari sales and software support will improve.

And, yes, I have expressed these same feelings to Atari.

—Frank Merson
San Angelo, TX

We're glad to hear that you've taken the time to contact Atari. That's the best way Atari users can tell the company what they like and don't like. Customer feedback is important to any company, and Atari is no exception. Anyone else who's interested in contacting Atari may write to them at Atari Corp., 1196 Borregas Avenue, Sunnyvale, CA 94086.

The Old Rehash Doubled

Please stop talking about favorite old games. I already heard Steve Panak talk about his favorites, so let's not do it again. Don't get me wrong, I always enjoy Arthur Leyenberger's *The End User*, but the subject of favorite oldies has been beaten to death.

Let's see some new stuff out there. You could review new hardware like *Easy Scan*. You could, for example, have some interviews with users' groups. There are also some decent programs out there like *AlfCrunch*, *SuperArc*, *Billboard* and many more. There's so much to talk about, so don't waste it on yet another look at the good old days.

After reading your *CES Video Game Report*, I have come to the conclusion that Sega, not Atari, will be the company to make headway into Nintendo territory. The reason: Atari titles lack any sort of creativity. They are nothing more than repackaged old games. I am sure that people will be less than thrilled to hear that they can play *Asteroids* and *One-on-One* yet again. Atari had better start releasing some new and creative titles, instead of trying to port all of their old titles to three different machines.

—Paul Siu
Darby, PA

Latest reports show that Nintendo has captured a phenomenal 80% of the video-game market, with Sega and Atari sharing the remaining portion. Sadly, you're absolutely correct in your conclusion that Sega has the greater chance of giving Nintendo a run for their money. Not only are the Sega games newer and more original, but the Sega machines have higher-quality graphics. It saddens us greatly to see the once video-gaming king now taking up a weak third-place position, but there is still hope for the future. Rumors abound that Atari is preparing a 68000-based game machine. The 68000 processor is the heart of the ST computer, and if Atari does it right, a 68000-based game machine could be the new state-of-the-art in video games. Of course, the other video game companies have new machines in the works, as well. It may be a tough battle for Atari.

Printscreen Update

My *Printscreen Utility* was published in the August 1988 issue of *ANALOG Computing*. I have a correction that should be made to this program.

Printscreen was designed to work with most programs, whether in BASIC or machine language. However, it doesn't work with some machine-language programs which access screen memory directly and do not use the CIO system. This is due to the way it is protected from dumping the screen when the printer is already in use. To make it work on these programs, the last two numbers in the DATA statement of Line 140 (15 and 144) must be changed to 6 and 176.

Thanks to Terry Johnson for bringing this problem to my attention. I hope this correction will help those who have had trouble using this utility.

—Justin E. Wilder
Goshen, IN

Moving Files

I received the September issue disk. I wanted to play *Snowplow*, but the information says to move *Snowplow* to another disk and rename it *AUTORUN.SYS*. However, no directions are given as to how to do this. Please advise.

—Robert Gunsberg
Medford, NY

First take a blank disk and format it using Option I of the DOS menu (type "DOS" to get to this menu), then write new DOS files to the disk by selecting Option H. Now insert your *ANALOG* disk into the drive and choose Option O, Duplicate File. Follow the on-screen prompts, switching the *ANALOG* disk (the source disk) and your newly formatted disk (the destination disk) as directed. When this process is completed, a new copy of *Snowplow* will have been placed on your disk. To rename the file, simply choose Option E of the DOS menu and type *SNOWPLOW.OBJ,AUTORUN.SYS*. The file will be renamed, and from that point on, to load *Snowplow*, just boot your computer with the new disk while holding down the Option key.

BOOT UP TO BIG SAVINGS!

1 YEAR FOR ONLY \$28

SAVE \$19 OFF THE COVER PRICE

1 YEAR WITH DISK ONLY \$105



**SAVE TIME AND MONEY
SUBSCRIBE TO ANALOG**

SAVE \$19 OFF THE
COVER PRICE WITH
THE CONVENIENCE
OF HAVING ANALOG
DELIVERED DIRECT-

LY TO YOUR DOOR
*BEFORE IT EVEN HITS
THE NEWSSTANDS.
GET THE MOST OUT
OF YOUR COMPUTER*

**SUBSCRIBE TO
ANALOG
TODAY**

1 YEAR @ \$28 — SAVE \$19
FOREIGN — ADD \$10 PER YEAR

1 YEAR WITH DISK @ \$105
FOREIGN — ADD \$15 PER YEAR

PAYMENT ENCLOSED BILL ME

CHARGE MY: VISA MC # _____

EXPIRATION DATE _____ SIGNATURE _____

MONEY BACK ON ALL UNUSED PORTIONS OF SUBSCRIPTIONS IF NOT SATISFIED.

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16927, N. Hollywood, CA 91615.
Your first issue will arrive in 6 TO 8 weeks.

Offer Expires June 28, 1989 WATCH FOR IT!

MCPYY

DCPYY

What's Coming Up

**Krazy Clown Jumper
Super Command Processor
What's New in Consumer Electronics**

Master Memory Map Part IX

by Robin Sherer

VCOUNT keeps track of what scan line is currently being drawn. Actually, it increases by 1 every two scan lines, so multiply the value by 2 to get the true number.

How to Read the Memory Map

Beginning users: Read the text that is printed in bold type only. These memory locations will be the easiest for you to use and usually don't involve assembly language.

Advanced users: Read everything! Many areas of memory are not of any practical use, but you can learn a lot about how a computer works by reading the boring parts.

PIA (6520)

PIA stands for Peripheral Interface Adapter and is also known as the 6520 chip. It takes care of the four controller jacks (two on some Atari models), which are the places that you plug your joysticks into. These controller jacks, or Atari ports as we will call them, have capabilities far greater than the reading

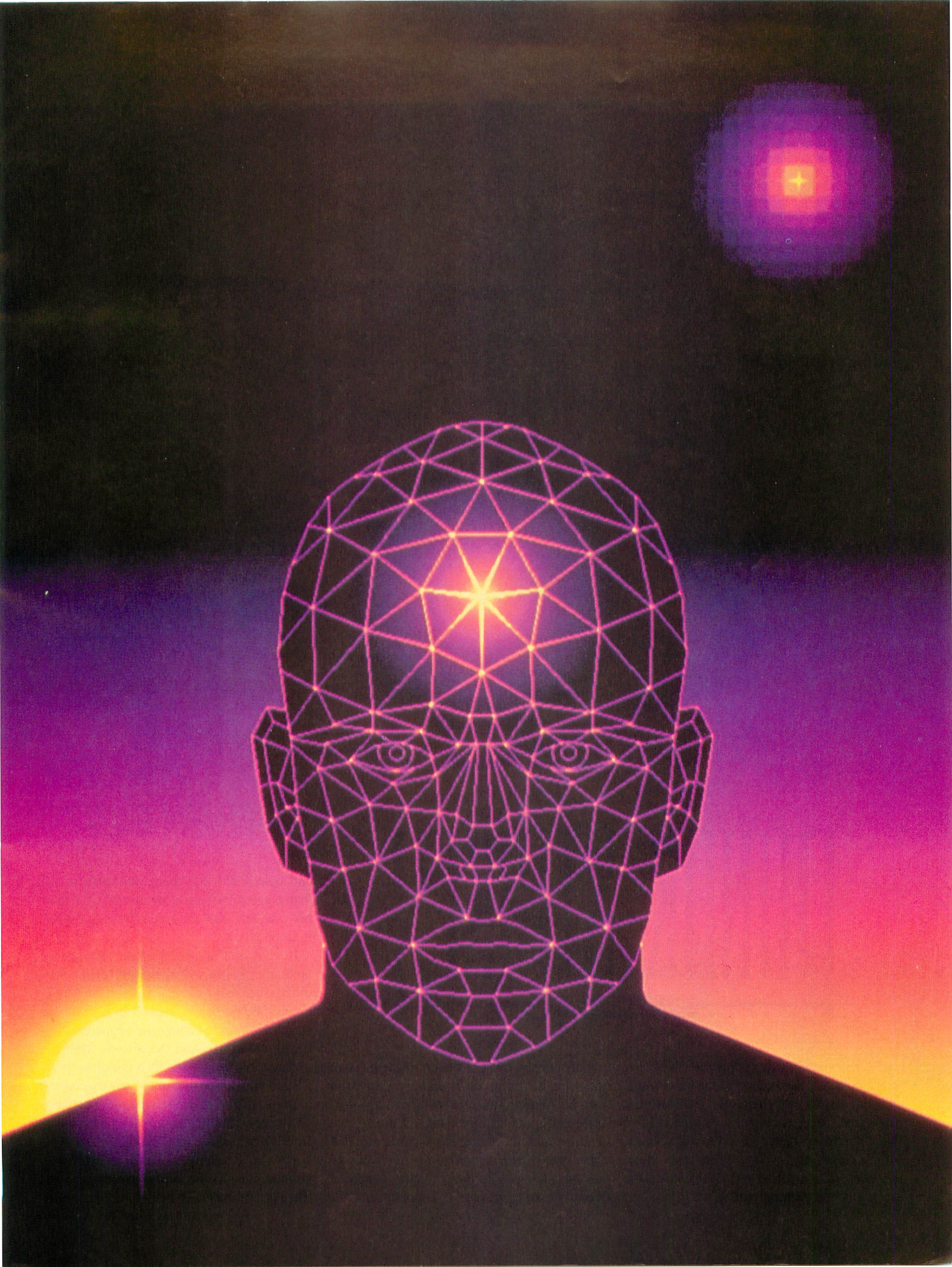
of joysticks, paddles and light pens. They can handle simultaneous input and output, which makes them perfect for use as an alternative to the 850 expansion interface. Some companies, in fact, already manufacture a cable that lets you run a printer from these ports instead of the 850.

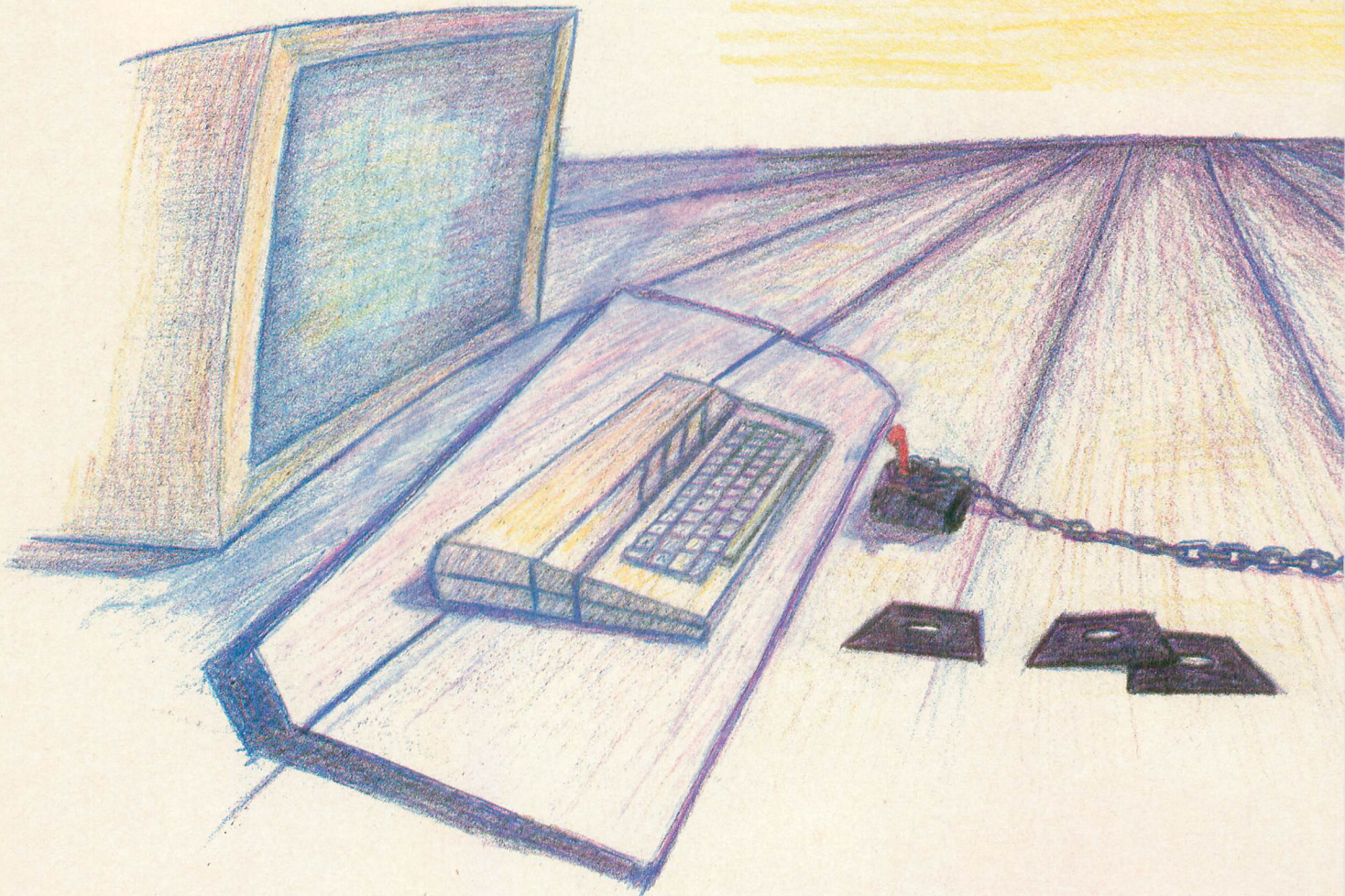
PORTA
54016

D300

PORTA actually has two functions, depending on whether Bit 2 of PACTL (following) is set. If it is set, then PORTA writes to or reads from the first two controller jacks. Depending on whether you're using joysticks or paddles, PORTA's bits will have the following meanings in Figure 1.

continued on page 32





PANAK STRIKES

by Steve Panak

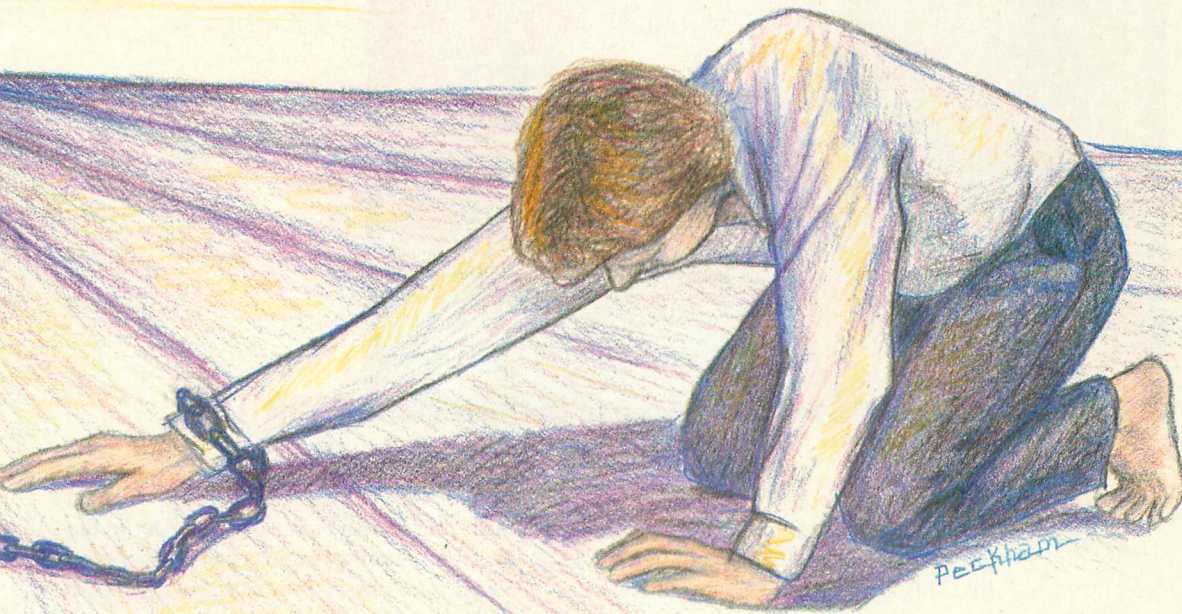
There exists in this country a disease. A contagious disease. A form of madness which renders its victims helpless, transforming them into crazed sociopaths, destined to live their remaining days chained to a keyboard, or a joystick, staring into a flickering phosphor screen, struggling to survive. This is the legacy of the gaming bug.

Fortunately, most of the populace has had the time to build up immunity. *Pac-Man* caught a nation weakened by *Space Invaders* off-guard, and millions fell. With all the defenses down, hundreds of games flooded the system, preying on those too weak to muscle their way to their favorite machine. However, in time, people became stronger

and discovered that there was more to life than chasing flashing blobs around on a phosphor screen. Just as suddenly as it started, it stopped.

But not for everyone. Because, although the epidemic has passed, there are still pockets of the disease remaining. Cursed clusters of barely-human beings locked in a desperate search for a never-ending and unattainable ultimate video fix. And the disease itself has mutated. No longer simply a mindless virus, it often transforms into mutations that appear to think, that offer deeper and more complex challenges, that infect many who escaped the first horrific onslaught.

For one to be cured, like the recovering alcoholic, it is first necessary to recognize that there is a problem. To help those of my read-



ers who may fear themselves infected (as well as those who deny their own susceptibility by allowing themselves to think “it can’t happen to me”), I offer the following multiple choice quiz. Honestly answer the questions; the person you save may be yourself. And no cribbing the answers from the end of the column.

1. What is your favorite, totally futile activity?
 - a) Searching for an honest politician.
 - b) Trying to copy a game diskette using Atari DOS.
 - c) Placing a Babel Fish in your ear.

2. When you pass an arcade game cabinet

in your local supermarket (or airport, shopping mall, etc.) do you:

- a) Unconsciously feel your pockets for change.
- b) Nervously look over your shoulder to see if anyone is watching as you probe the coin return for a forgotten token.
- c) Violently rattle the joystick when neither a) nor b) result in activating the game.

3. Do the palms of your hands:

- a) Sweat at the sight of a disk drive’s busy light?
- b) Contain callouses at the base of your thumbs?
- c) Have hair growing in them? (the second sign of impending insanity, the first being searching for the aforementioned hair).

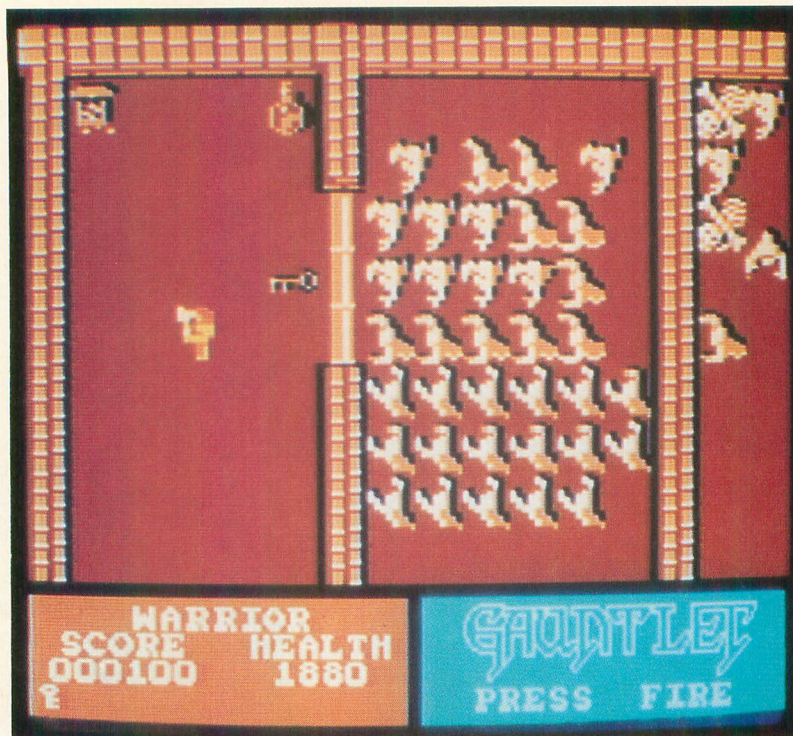
4. In *Boulder Dash*, the main character is:

- a) A lump of coal named Chunky.
- b) A soiled miner named Rockford.
- c) The poor soul holding the joystick, unable to eat, sleep, or excrete until he completes the current level.

5. What is your favorite diversion during your free time?

- a) Watching NBC’s Saturday evening lineup.
- b) Reading about computer games.
- c) Playing computer games.

Of course, there are hundreds of other signs, some of them obvious, others barely discernible. But rather than go into those, let’s take a little recess and pop into these two new games.



GAUNTLET

Gauntlet

by Atari

Mindscape
3444 Dundee Road
Northbrook, IL 60062
(312) 480-7667
\$34.95

Although I haven't been in the arcades for

a number of months, after the first boot it's easy to see how *Gauntlet* could easily be a hit. This is because the game adheres closely to the time-proven and weather-worn elements which can make an arcade action game so addictive.

One or two players may compete or cooperate. And, in a first, the program allows the second player to start at any time, simply by pressing the joystick button. Before starting the game, each player chooses a character through whom they will vicari-

ously run the *Gauntlet*. Go with Merlin the Wizard for strong magic powers or Thor the Warrior for physical strength. As is the case with all arcade hits, your goals are simple. Collect valuables. Destroy monsters. And, of course, survive.

Monsters are produced by monster generators located throughout the mazes. These heinous devices will continue to belch out dangerous creatures until they are destroyed. A number of species of vile beasts keep the action lively and fresh, as each has its own



RICHARD PETTY'S TALLADEGA

PANAK STRIKES

attack mode and vulnerability. Demons shoot fireballs while Lobbers toss stones, and each species comes in three levels of strength, which determine how many hits are necessary to kill it.

And to keep it interesting, various special objects are scattered throughout the dungeons. Potions to increase your hero's strength, food to increase his lifeline, armor to increase his protection; all of these provide you with a slight edge against seemingly incredible odds.

Although the graphics are not the best I've ever seen on an 8-bit machine, the action is nonetheless captivating. You always know the game is good when you find that suddenly an hour, then two, then three have passed you by. Probably the aspect of this game that is most addictive is the sheer number of adversaries you are required to wipe out. No matter what you do, they just keep on coming. And you keep on killing. What could be more relaxing?

Rounding out the package is a decent manual, written concisely and sporting numerous illustrations. Overall, *Gauntlet* is recommended for anyone looking for something new in arcade challenges.

Richard Petty's Talladega

by Robert T. Bonifacio

Cosmi
415 North Figueroa
Wilmington, CA 90744
(213) 835-9687
\$12.95

It's been a while since I've played a race game on the 8-bit, and unfortunately no one seems to have taken this time-interim to create a really good game. Still, *Richard Petty's Talladega* manages to provide a couple of surprises for would-be stock car drivers.

This game is more like a simulation, in that it puts you through many of the things that a real driver would experience, rather than being content with merely letting you speed around an oval track. For example, when you first start the game, you have to qualify for

This game is more like a simulation, in that it puts you through many of the things that a real driver would experience, rather than being content with merely letting you speed around an oval track.

starting position in a field of twenty opponents. Only then, and after choosing one of the three tracks, are you ready to start toward the checkered flag.

Just a perusal through the lengthy manual hints at the complexity of the simulation. Stops will be necessary in the pit stop for more fuel and new tires. A turbo boost, while eating up a lot of valuable fuel, may be just what the doctor ordered to get past the last couple of cars between you and the finish line. Wrecks will cause the action to slow while the yellow flag is out. Unfortunately, this complexity is not continued throughout every aspect of the game.

In particular I found the graphics to be surprisingly disappointing. For example, when you wreck your car, the monitor fills with a smoke screen of special characters, rather than the colorful conflagration one expects from the explosion of a race car full of alcohol. The display is reminiscent of that of the 2600. I know better is possible on this machine—I've seen it done. But while the graphics are a disappointment, the control of your vehicle is equally surprising, but this time the surprise is pleasant. For some reason, programmers in the past have had a lot of trouble translating the steering wheel's rotation to the joystick's lateral movements. This difficulty is not apparent here, as your car is very responsive to your every whim.

Overall, *Talladega* gives an uneven performance. It is too flawed to get an unqualified recommendation, but offers too much to be completely dismissed (especially considering its low price). Your best bet is to check it out at your local dealer. You might end up driving it home.

Getting back to the quiz, there really are no answers—it was all in fun. Probably the only ones surprised by this revelation are those of you who jumped back here looking for the answers. In the final analysis, no quiz or other diagnosis is necessary, as those afflicted (as I am) know who they are. And living with this disease is not all that bad, especially given all the great programs out there. So until next we meet, curl up with your favorite game and suffer. **F**

8 BIT NEWS

Certron Takes a Stand

Certron, a well-known computer supplies company, has recently made available a new series of universal computer - equipment stands.

The UTS-1 *Tilt 'N Swivel* monitor stand allows the computer owner to set his monitor at whatever angle is most convenient and comfortable. The stand is designed in such a way that the monitor can remain on the stand when angle adjustments need to be made. The unit is constructed of high-impact, non-magnetic plastic and retails for \$34.95.

Also available from Certron are the PS-80 and PS-132 double-tray printer stands, the former for 80-column printers and the latter for 132-column printers. Both stands contain input and output trays, and the vinyl-coated steel construction minimizes printer noise. The PS-80 is \$24.95, while the PS-132 is

\$27.95.

If you want to make sure that you have a printer stand that'll fit whatever printer comes your way, you might want to consider Certron's new UPS-1 *Universal Printer Stand*. Although it doesn't supply the input and output trays, it will accommodate any size printer. The UPS-1 can be purchased for \$12.95.

Finally, Certron has available locking diskette storage boxes. The 5.25-inch model will hold 50 disks and is made of sturdy plastic with smoke-tinted lids.

Certron Corporation
1651 S. State College Blvd.
Anaheim, CA 92806
(714) 634-4280

CIRCLE #156 ON READER SERVICE CARD.

SYSOPs Take Note!

Carina Software Systems has released *Carina II*, a new version of their popular BBS system for the 8-bit Atari computers. The Carina BBS system is set up using SIGs (special interest groups), and divides the bulletin board up into many smaller systems, each focused on a particular topic of interest. This makes it easier for users to locate items that are of interest to them, and at the same time, bypass those topics they don't care about. *Carina II* supports up to 26 of these areas.

Carina II also provides other BBS features such as download databases, electronic mail, online games, as well as some special features like the ability to create polls that al-

low users to vote on various questions.

Carina II requires ICD's *SpartaDOS*, and it is suggested that SYSOPs run the BBS with a RAMdisk of at least 192K and an ICD *Real-Time 8* cartridge. The complete system is available now and is priced at \$65. Owners of the original *Carina BBS* may be eligible for an upgrade discount.

Carina Software Systems
P.O. Box 6072
Rockford, IL 61125
(815) 874-1836

CIRCLE #155 ON READER SERVICE CARD.

German Club Offers Disk Magazine

The following is an excerpt from a letter recently received at ANALOG's offices:

Two years ago, the Atari Bit Byter User Club (ABBUC) was in contact with over 25 users' groups in the U.S. Today we hear from less than ten. Have the others died out or switched over to the STs? The remaining 8-bit users must work together to keep their machines from going the way of the VIC or the TI. To work together, first the users must communicate with each other. ABBUC, a group of over 500 users, would like to communicate with users' groups from the U.S. and the rest of the world as well. We are prepared to trade our tri-monthly disk magazine to all the clubs who send us their newsletter. We can be contacted at the following address:

ABBUC
c/o Wolfgang Burger
Wieschenbeck 45
D-4352 Herten
West Germany

We are looking forward to corresponding with our fellow Atarians throughout the world. Through the exchange of information between users' groups, the 8-bit Ataris will continue to grow and will one day be finally recognized as a machine that was truly ahead of its time.

So if your users' group would be interested in receiving ABBUC's disk-based magazine, drop them a line. I'm sure they'll be thrilled to hear from you.

CIRCLE #157 ON READER SERVICE CARD.

A 16-bit Atari 8-bit

DataQue Software has announced *Turbo-816*, a hardware upgrade for the Atari 8-bit computers that will both increase the speed of the computer and allow direct access to memory beyond 64K. The extra memory is not accessed using the awkward bank-select methods; it is a linear memory bank of up to 16 megabytes. Special internally mounted memory boards that, in most cases, require no hardware modifications, will be available to take advantage of the new extended addressing range.

The *Turbo-816* upgrade will be available in two forms: a replacement CPU board for the original 400/800 Computer system and a plug-in module for the XL/XE series. There are no hardware modifications required, except in the case of XL/XE systems which will require the removal of the existing CPU. DataQue claims that *Turbo-816* will be compat-

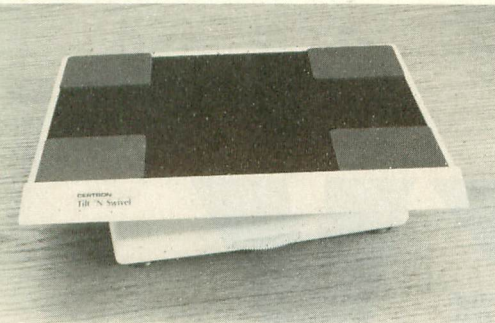
ible with most of the existing software.

New products planned for the *Turbo-816* system include a real-time multi-tasking operating system kernel, an assembler-editor-debugger, a new BASIC, a C development package, a *Turbo-GOS* operating system (graphics-based), and a developer's kit.

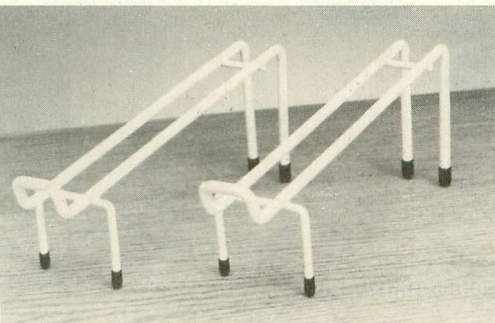
DataQue Software
Dept. T-800
P.O. Box 134
Ontario, OH 44862

New product announcements to be considered for use in 8-bit News should be sent to: 8-bit News, ANALOG Computing, P.O. Box 1413-M.O., Manchester, CT 06040-1413. Please include photos, screen shots and product samples whenever possible.

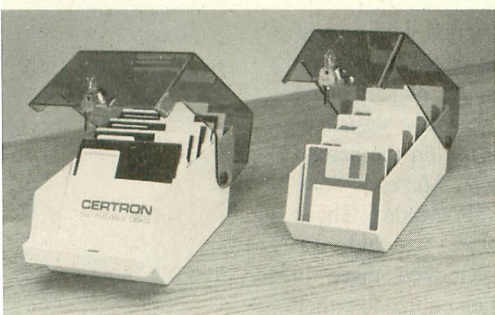
CIRCLE #158 ON READER SERVICE CARD.



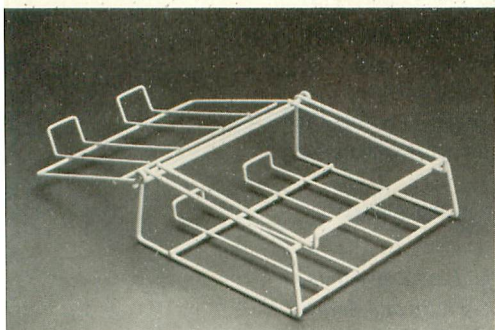
UTS-1 TILT 'N SWIVEL MONITOR STAND



80-COLUMN DOUBLE-TRAY PRINTER STAND



LOCKING DESKTOP STORAGE FILES



UPS-1 UNIVERSAL PRINTER STAND

Krazy Mazes

by Barry Kolbe

You and another adventurer are searching for the lost gold mines of Mazeaton. The mines are located in the midst of a large maze, well-protected by ghostly guardians. To reach the cache of gold, both of you must race through identical mazes while being hunted down by the guardians of the treasure. If you should happen to touch one of the guardians, you will be zapped back to the beginning of the maze. Whoever arrives at the end of the maze first receives 100 gold coins. This achievement also allows both of you to enter the middle maze where 277 gold pieces are kept—collect as much as you can. After this you must leave the maze city by racing through another maze like the first one. The victor (or Victoria) again receives 100 gold coins.

After the third maze your accumulated wealth is displayed and immediately phoned into the IRS. Pressing Start permits you to play again and add to your riches. Option resets your bank to zero. Unfortunately, this will not deceive the IRS in the least. While you are mazing, press Select to abort the game and return to this screen.

Typing It In

Krazy Mazes is an all-machine-language game and must be typed in using the M/L Editor found elsewhere in this issue. Load

and run the game by using the Binary Load option for your DOS.

Author's Notes

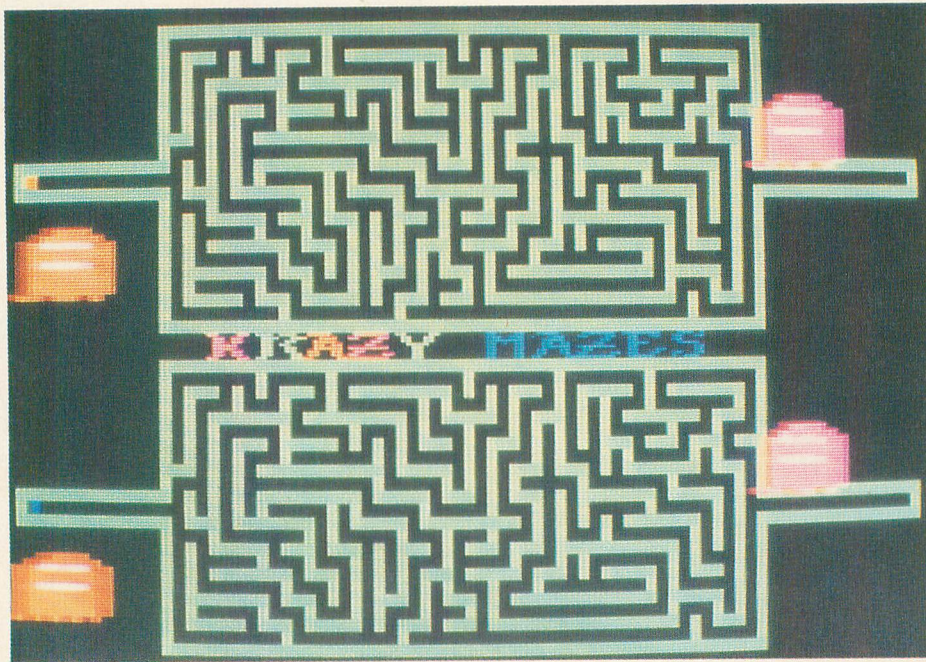
I have always been intrigued with mazes. I think it started with the Atari 2600 game called *Maze Craze* where two players raced through the same maze. The maze was small because of the large images on the screen (like GRAPHICS 2). *Krazy Mazes* is done in GRAPHICS 5 and ANTIC Mode 5. These were chosen for their colors, size and visibility. The randomly generated maze is formed in memory and then copied to either the GRAPHICS 5 or ANTIC 5 screen.

Most people who make maze games use one of two possible maze generators. They often start the maze generation in the upper left-hand corner and these mazes tend to be quite simple as there is usually only one path from beginning to end. I suggest starting the generation from the center as it makes for more interesting mazes.

Anyway, don't get lost in the mazes!

Barry Kolbe and his wife, Linda, live in the country with their children, Philip and Audra. The household uses time-sharing because the Atari 800 and the Nintendo Entertainment System are hooked up to the same TV.





LISTING 1: M/L EDITOR DATA

```

1000 DATA 255,255,0,48,51,60,32,85,59,
,169,0,133,179,169,16,162,4818
1010 DATA 2,157,222,58,157,242,58,202,
,16,247,169,80,141,244,2,32,7707
1020 DATA 174,53,169,7,162,55,160,27,3
32,92,228,169,0,133,180,32,5129
1030 DATA 200,54,32,51,54,169,0,133,15
56,169,128,133,157,169,2,133,6649
1040 DATA 181,169,1,133,179,32,39,59,3
32,92,48,32,88,58,169,0,537
1050 DATA 141,47,2,32,108,55,32,88,58,
,32,92,48,32,88,58,76,9381
1060 DATA 3,48,169,0,141,47,2,32,143,5
50,32,132,52,32,168,50,1090
1070 DATA 32,16,51,32,41,52,169,40,141
1,196,2,169,202,141,197,2,6301
1080 DATA 169,148,141,198,2,32,56,50,3
32,200,54,32,51,54,169,2,1006
1090 DATA 133,181,169,1,133,179,133,18
80,169,69,141,48,2,169,50,141,5942
1100 DATA 49,2,169,62,141,47,2,165,152
2,133,168,165,154,133,169,169,9858
1110 DATA 1,133,176,169,0,133,177,169,
,152,133,178,32,199,51,169,3,6894
1120 DATA 133,176,165,153,133,168,165,
,155,133,169,169,0,133,177,169,155,748
8
1130 DATA 133,178,32,199,51,169,0,133,
,156,169,128,133,157,173,120,2,7237
1140 DATA 201,15,208,3,76,119,49,201,7
7,208,25,165,152,201,79,240,9265
1150 DATA 103,32,169,52,176,98,32,85,4
49,230,152,32,112,49,32,132,3935
1160 DATA 50,76,82,49,201,11,208,25,16
65,152,201,1,240,74,32,178,7243
1170 DATA 52,176,69,32,85,49,198,152,3
32,112,49,32,132,50,76,82,2429

```

```

1180 DATA 49,201,14,208,25,165,154,201
1,1,240,45,32,186,52,176,40,6020
1190 DATA 32,85,49,198,154,32,112,49,3
32,132,50,76,82,49,201,13,2514
1200 DATA 208,22,165,154,201,21,240,16
6,32,195,52,176,11,32,85,49,3074
1210 DATA 230,154,32,112,49,32,132,50,
,76,119,49,169,0,133,176,165,5636
1220 DATA 152,133,168,165,154,133,169,
,169,0,133,177,169,152,133,178,141,974
4
1230 DATA 30,208,32,199,51,96,169,1,13
33,176,76,89,49,173,121,2,4357
1240 DATA 201,15,240,98,201,7,208,25,1
165,153,201,79,240,88,32,32,6804
1250 DATA 53,176,83,32,13,50,230,153,3
32,37,50,32,132,50,76,224,3963
1260 DATA 49,201,11,208,19,165,153,201
1,1,240,59,32,41,53,176,54,4561
1270 DATA 32,13,50,198,153,76,146,49,2
201,14,208,19,165,155,201,1,6716
1280 DATA 240,36,32,50,53,176,31,32,13
3,50,198,155,76,146,49,201,5320
1290 DATA 13,208,19,165,155,201,21,240
0,13,32,60,53,176,8,32,13,1305
1300 DATA 50,230,155,76,146,49,32,103,
,53,176,39,141,30,208,32,219,6295
1310 DATA 54,165,156,201,218,144,3,32,
,51,54,165,157,201,218,144,3,7768
1320 DATA 32,238,53,173,31,208,201,5,2
208,5,104,104,76,3,48,76,2809
1330 DATA 215,48,96,165,153,133,168,16
65,155,133,169,169,0,133,177,169,481
1340 DATA 155,133,178,169,0,133,176,32
2,199,51,96,165,153,133,168,165,9605
1350 DATA 155,133,169,169,3,133,176,14
41,30,208,32,199,51,96,169,1,5775
1360 DATA 133,152,133,153,169,11,133,1
154,133,155,96,112,112,112,74,0,5163
1370 DATA 152,10,10,10,10,10,10,10,10,
,10,10,10,10,10,10,2672
1380 DATA 10,10,10,10,10,10,10,70,168,
,58,74,0,155,10,10,10,7591
1390 DATA 10,10,10,10,10,10,10,10,10,1
10,10,10,10,10,10,2750
1400 DATA 10,10,65,69,50,169,254,20
08,2,169,255,133,20,165,20,7195

```

1410 DATA 208,252,133,77,96,169,0,133,
,144,169,144,133,145,162,7,160,8371
1420 DATA 0,169,0,145,144,136,208,251,
,230,145,202,16,242,96,169,0,297
1430 DATA 133,144,169,144,133,145,162,
,22,160,12,169,128,145,144,200,192,707
7
1440 DATA 65,208,249,165,144,24,105,80
0,133,144,165,145,105,0,133,145,7439
1450 DATA 202,16,229,169,0,24,105,32,1
133,144,169,144,105,3,133,145,6128
1460 DATA 162,2,160,0,169,128,145,144,
,200,192,79,208,249,165,144,24,1062
1470 DATA 105,80,133,144,165,145,105,0
0,133,145,202,16,229,96,169,125,9857
1480 DATA 133,144,169,145,133,145,162,
,15,189,183,51,149,186,202,16,248,1111
1
1490 DATA 169,6,133,175,208,22,169,133
3,133,144,169,147,133,145,162,15,8878
1500 DATA 189,167,51,149,186,202,16,24
48,169,128,133,175,169,5,160,0,8041
1510 DATA 145,144,173,10,210,41,3,133,
,170,133,171,165,170,10,170,165,9144
1520 DATA 144,24,117,186,133,148,165,1
145,117,187,133,149,177,148,197,175,29
977
1530 DATA 208,38,165,170,10,170,165,14
44,24,117,194,133,146,165,145,117,9737
7
1540 DATA 195,133,147,169,0,145,146,23
30,170,165,170,145,148,165,148,133,222
22
1550 DATA 144,165,149,133,145,76,44,51
1,230,170,165,170,41,3,133,170,7815
1560 DATA 197,171,208,183,160,0,177,14
44,133,170,169,0,145,144,165,170,498
1570 DATA 201,5,208,1,96,230,170,165,1
170,41,3,10,170,24,165,144,6197
1580 DATA 117,186,133,144,165,145,117,
,187,133,145,76,44,51,2,0,96,3292
1590 DATA 255,254,255,160,0,1,0,176,25
55,255,255,80,0,2,0,176,6626
1600 DATA 255,254,255,80,0,1,0,216,255
5,255,255,40,0,165,169,10,8317
1610 DATA 133,150,169,0,42,133,151,6,1
150,38,151,165,150,133,165,165,8961
1620 DATA 151,133,166,6,150,38,151,6,1
150,38,151,165,150,24,101,165,6454
1630 DATA 133,150,165,151,101,166,133,
,151,165,177,24,101,150,133,150,165,23
35
1640 DATA 178,101,151,133,151,165,168,
,41,3,170,165,168,74,74,24,101,5786
1650 DATA 150,133,144,165,151,105,0,13
33,145,164,176,189,165,52,57,157,8996
1660 DATA 52,133,165,189,161,52,160,0,
,49,144,5,165,145,144,96,169,7427
1670 DATA 0,133,174,133,168,169,2,133,
,176,169,0,133,177,169,152,133,9867
1680 DATA 178,169,0,133,148,169,144,13
33,149,160,0,177,148,240,9,132,9150
1690 DATA 168,165,174,133,169,32,199,5
51,230,168,164,168,192,80,208,235,4146
6
1700 DATA 230,174,165,174,201,23,240,1
16,165,148,24,105,80,133,148,165,8671
1710 DATA 149,105,0,133,149,76,67,52,1
160,0,185,0,152,153,0,155,4760
1720 DATA 185,0,153,153,0,156,200,208,
,241,96,169,152,133,145,169,0,82
1730 DATA 133,144,162,2,160,0,169,0,14
45,144,200,208,251,230,145,202,3959
1740 DATA 16,242,96,0,85,170,255,63,20
07,243,252,192,48,12,3,165,9108
1750 DATA 154,166,152,232,32,204,52,96
6,165,154,166,152,202,76,174,52,9943
1760 DATA 166,154,202,138,166,152,76,1
174,52,166,154,232,138,166,152,76,1278
8
1770 DATA 174,52,10,133,165,169,0,42,1
133,166,160,2,6,165,38,166,5040
1780 DATA 136,16,249,165,165,133,150,1
165,166,133,151,160,1,6,165,38,6933
1790 DATA 166,136,16,249,165,165,24,10
01,150,133,150,165,166,101,151,133,338
8
1800 DATA 151,169,0,24,101,150,133,146
6,169,144,101,151,133,147,138,24,8014
1810 DATA 101,146,133,146,165,147,105,
,0,133,147,160,0,177,146,201,128,9463
1820 DATA 208,2,56,96,24,96,165,155,16
66,153,232,32,204,52,96,165,9095
1830 DATA 155,166,153,202,32,204,52,96
6,166,155,202,138,166,153,32,204,1066
1840 DATA 52,96,166,155,232,138,166,15
53,32,204,52,96,112,112,112,68,7420
1850 DATA 0,144,4,4,4,4,4,4,4,4,4,4,
,4,4,4,2670
1860 DATA 4,4,4,4,4,4,4,70,189,58,65,7
70,53,165,152,201,4863
1870 DATA 77,240,8,165,153,201,77,240,
,10,24,96,238,222,58,32,21,6297
1880 DATA 60,56,96,238,242,58,32,26,60
0,56,96,32,143,50,169,0,2916
1890 DATA 133,144,169,144,133,145,162,
,20,160,0,169,6,145,144,200,192,9567
1900 DATA 39,208,249,165,144,24,105,40
0,133,144,165,145,105,0,133,145,7553
1910 DATA 202,16,229,96,169,62,141,47,
,2,169,1,141,111,2,169,102,4844
1920 DATA 141,192,2,169,38,141,193,2,1
169,14,141,194,2,141,195,2,6027
1930 DATA 169,0,141,0,208,141,1,208,14
41,2,208,169,1,141,8,208,7119
1940 DATA 141,9,208,141,10,208,141,11,
,208,169,3,141,29,208,169,136,8947
1950 DATA 141,7,212,96,32,144,54,173,1
10,210,41,31,24,105,71,133,3899
1960 DATA 164,162,0,168,189,167,54,153
3,0,141,153,100,141,200,153,0,7890
1970 DATA 141,153,100,141,200,232,224,
,12,208,234,164,164,169,20,153,4,157
1980 DATA 143,153,5,143,153,104,143,15
53,105,143,169,28,153,10,143,153,7922
1990 DATA 11,143,153,110,143,153,111,1
143,96,32,121,54,173,10,210,41,6098
2000 DATA 31,24,105,32,133,163,162,0,1
164,163,189,167,54,153,0,140,7572
2010 DATA 153,100,140,200,153,0,140,15
53,100,140,200,232,224,12,208,234,3780
0
2020 DATA 169,20,164,163,153,4,142,153
3,5,142,153,104,142,153,105,142,8611
2030 DATA 169,20,153,10,142,153,11,142
2,153,110,142,153,111,142,96,162,8933
2040 DATA 23,169,0,164,163,153,0,140,1
153,0,142,153,100,140,153,100,7840
2050 DATA 142,200,202,16,240,96,162,23
3,169,0,164,164,153,0,141,153,8201
2060 DATA 100,141,153,0,143,153,100,14
43,200,202,16,240,96,28,62,107,7536
2070 DATA 127,127,99,127,127,127,127,2
218,0,230,156,230,157,165,156,141,3009
9
2080 DATA 0,208,141,2,208,165,157,141,
,1,208,141,3,208,96,162,0,7338
2090 DATA 138,157,0,140,157,0,141,157,
,0,142,157,0,143,232,208,241,1360
2100 DATA 96,173,4,208,133,158,41,1,24
40,15,32,85,49,169,1,133,4122
2110 DATA 152,169,11,133,154,32,112,49
9,96,165,158,41,4,240,15,32,4196
2120 DATA 13,50,169,1,133,153,169,11,1
133,155,32,37,50,96,173,5,3810
2130 DATA 208,133,158,41,1,240,3,76,22
28,54,165,158,41,4,208,223,8896
2140 DATA 96,165,179,208,10,169,0,141,
,0,210,141,1,210,240,64,32,7352
2150 DATA 179,54,230,167,165,167,41,3,
,170,134,167,189,160,58,141,192,987
2160 DATA 2,189,164,58,141,193,2,165,1
180,240,3,32,219,54,198,181,367
2170 DATA 165,181,208,27,238,34,59,173
3,34,59,41,3,141,34,59,170,3917
2180 DATA 169,2,133,181,189,35,59,141,
,0,210,169,166,141,1,210,76,8336
2190 DATA 98,228,169,70,141,48,2,169,5

53,141,49,2,32,123,58,169,4052
2200 DATA 148,141,196,2,169,246,141,19
97,2,169,68,141,198,2,32,133,7468
2210 DATA 53,32,248,50,32,154,55,169,6
62,141,47,2,32,223,55,96,4500
2220 DATA 169,0,133,144,169,144,133,14
45,169,0,133,159,133,160,162,18,8743
2230 DATA 165,144,24,105,40,133,144,16
65,145,105,0,133,145,160,1,177,7424
2240 DATA 144,208,12,169,7,145,144,230
0,159,165,159,208,2,230,160,200,3437
2250 DATA 192,38,208,235,202,16,217,16
65,159,56,233,2,133,159,165,160,1595
2260 DATA 233,0,133,160,96,169,1,133,1
152,169,10,133,154,32,93,57,5618
2270 DATA 169,37,133,153,169,10,133,15
55,32,107,57,169,0,162,3,149,5310
2280 DATA 182,202,16,251,162,2,169,16,
,157,199,58,157,205,58,202,16,8739
2290 DATA 247,32,93,57,173,120,2,201,1
15,240,86,201,7,208,22,32,6053
2300 DATA 242,56,176,77,32,192,57,32,6
66,57,230,152,32,93,57,32,4060
2310 DATA 128,50,76,107,56,201,11,208,
,16,32,6,57,176,51,32,192,4189
2320 DATA 57,32,66,57,198,152,76,38,56
6,201,14,208,16,32,14,57,2547
2330 DATA 176,31,32,192,57,32,66,57,19
98,154,76,38,56,201,13,208,6506
2340 DATA 16,32,23,57,176,11,32,192,57
7,32,66,57,230,154,76,38,4560
2350 DATA 56,32,107,57,173,121,2,201,1
15,240,86,201,7,208,22,32,5970
2360 DATA 32,57,176,77,32,12,58,32,80,
,57,230,153,32,107,57,32,3173
2370 DATA 128,50,76,203,56,201,11,208,
,16,32,40,57,176,51,32,12,2127
2380 DATA 58,32,80,57,198,153,76,134,5
56,201,14,208,16,32,48,57,3934
2390 DATA 176,31,32,12,58,32,80,57,198
8,155,76,134,56,201,13,208,7111
2400 DATA 16,32,57,57,176,11,32,12,58,
,32,80,57,230,155,76,134,4995
2410 DATA 56,165,182,24,101,184,133,16
61,165,183,101,185,133,162,197,160,342
24
2420 DATA 208,7,165,161,197,159,208,1,
,96,173,31,208,201,5,208,5,8498
2430 DATA 104,104,76,3,48,76,11,56,165
5,154,166,152,232,32,121,57,7069
2440 DATA 160,0,177,146,201,6,240,2,24
4,96,56,96,165,154,166,152,8619
2450 DATA 202,76,247,56,164,154,136,15
52,166,152,76,247,56,164,154,200,3029
2460 DATA 152,166,152,76,247,56,165,15
55,166,153,232,76,247,56,165,155,3108
2470 DATA 166,153,202,76,247,56,164,15
55,136,152,166,153,76,247,56,164,2127
2480 DATA 155,200,152,166,153,76,247,5
56,165,154,166,152,32,121,57,160,9753
2490 DATA 0,169,0,145,146,96,165,155,1
166,153,32,121,57,160,0,152,7350
2500 DATA 145,146,96,165,154,166,152,3
32,121,57,160,0,169,8,145,146,7210
2510 DATA 96,165,155,166,153,32,121,57
7,160,0,169,9,145,146,96,10,5261
2520 DATA 133,165,169,0,42,133,166,6,1
165,38,166,6,165,38,166,165,7278
2530 DATA 165,133,150,165,166,133,151,
,6,165,38,166,6,165,38,166,165,8374
2540 DATA 165,24,101,150,133,150,165,1
166,101,151,133,151,169,0,24,101,7571
2550 DATA 150,133,146,169,144,101,151,
,133,147,138,24,101,146,133,146,165,29
96
2560 DATA 147,105,0,133,147,96,201,7,2
240,1,96,32,15,59,230,182,7216
2570 DATA 208,2,230,183,162,2,254,199,
,58,189,199,58,201,26,144,17,9102
2580 DATA 169,16,157,199,58,202,48,9,2
254,199,58,189,199,58,76,214,1103
2590 DATA 57,32,253,59,162,2,254,222,5
58,189,222,58,201,26,144,17,9041
2600 DATA 169,16,157,222,58,202,48,9,2

254,222,58,189,222,58,76,246,2256
2610 DATA 57,96,201,7,240,1,96,32,15,5
59,230,184,208,2,230,185,229
2620 DATA 162,2,254,205,58,189,205,58,
,201,26,144,17,169,16,157,205,9604
2630 DATA 58,202,48,9,254,205,58,189,2
205,58,76,34,58,32,31,60,3986
2640 DATA 162,2,254,242,58,189,242,58,
,201,26,144,17,169,16,157,242,623
2650 DATA 58,202,48,9,254,242,58,189,2
242,58,76,66,58,96,169,136,9127
2660 DATA 141,48,2,169,58,141,49,2,32,
,123,58,173,132,2,208,6,4266
2670 DATA 173,132,2,240,251,96,173,133
3,2,208,240,173,133,2,240,251,4366
2680 DATA 96,169,0,133,179,162,3,157,0
0,208,202,16,250,96,112,112,9350
2690 DATA 112,112,112,112,71,210,58,11
12,112,112,71,230,58,112,112,112,8190
2700 DATA 71,250,58,65,136,58,68,68,74
4,74,36,36,42,42,0,0,9121
2710 DATA 0,0,235,114,33,250,121,128,1
173,161,186,165,179,0,0,0,6927
2720 DATA 0,0,0,115,99,111,114,101,1
115,0,17,26,16,16,16,8899
2730 DATA 0,18,26,16,16,16,0,0,0,0,5
51,35,47,50,37,6151
2740 DATA 0,17,26,0,16,16,16,0,0,0,0
0,0,0,51,3956
2750 DATA 35,47,50,37,0,18,26,0,16,16,
,16,0,0,0,0,3947
2760 DATA 0,0,0,240,242,229,243,243,
,128,230,233,242,229,0,0,4104
2770 DATA 0,0,0,0,169,23,141,0,210,1
162,170,142,1,210,32,6517
2780 DATA 132,50,169,0,141,1,210,96,0,
,132,0,143,0,169,99,141,5611
2790 DATA 48,2,169,59,141,49,2,165,156
6,201,218,144,3,32,51,54,5574
2800 DATA 165,157,201,210,144,3,32,238
8,53,173,31,208,201,6,208,1,8497
2810 DATA 96,201,3,208,226,32,85,59,76
6,49,59,169,16,162,4,157,5437
2820 DATA 237,59,157,247,59,202,16,247
7,96,112,112,112,112,71,133,59,8178
2830 DATA 7,112,112,70,173,59,112,112,
,112,112,70,193,59,112,70,213,8583
2840 DATA 59,112,112,112,112,70,233,59
9,65,99,59,0,0,0,114,1038
2850 DATA 0,0,0,186,0,0,0,0,0,0,0,0,
,0,0,0,3594
2860 DATA 0,43,0,0,225,0,0,0,121,0,0
0,237,225,250,229,9151
2870 DATA 243,0,0,0,0,162,185,128,16
62,161,178,178,185,128,171,1966
2880 DATA 175,172,162,165,0,0,0,0,0,
,115,116,97,114,116,0,1799
2890 DATA 0,0,0,112,108,97,121,0,0,0,0
0,0,0,111,112,8764
2900 DATA 116,105,111,110,0,0,0,115,99
9,111,114,101,115,0,0,0,881
2910 DATA 17,26,0,16,16,16,16,0,0,1
18,26,0,16,16,16,4689
2920 DATA 16,16,0,162,4,254,237,59,189
9,237,59,201,26,144,11,169,9646
2930 DATA 16,157,237,59,202,16,238,189
9,237,59,96,162,2,76,255,59,73
2940 DATA 162,2,76,33,60,162,4,254,247
7,59,189,247,59,201,26,144,929
2950 DATA 8,169,16,157,247,59,202,16,2
238,96,0,80,255,81,0,0,5614
2960 DATA 0,0,0,0,56,56,56,56,56,0
0,56,0,102,102,9370
2970 DATA 102,0,0,0,0,102,255,102,
,102,255,102,0,24,62,4063
2980 DATA 96,60,6,124,24,0,0,102,108,2
24,48,102,70,0,223,253,5913
2990 DATA 223,253,223,253,223,253,0,40
0,170,190,190,170,40,0,0,40,7073
3000 DATA 0,105,40,40,65,65,0,20,0,150
0,20,20,130,130,0,102,1467
3010 DATA 60,255,60,102,0,0,0,24,24,12
26,24,24,0,0,0,0,6388
3020 DATA 0,0,0,24,24,48,0,0,0,126,0,0

```

0,0,0,0,0,4784
3030 DATA 0,0,0,24,24,0,0,6,12,24,48,9
96,64,0,124,206,1310
3040 DATA 198,198,198,230,124,0,56,56,
,24,24,24,24,24,0,124,230,3468
3050 DATA 12,24,48,96,254,0,126,12,24,
,12,6,102,60,0,12,28,8920
3060 DATA 60,108,204,254,12,0,126,96,1
124,6,6,102,60,0,124,198,4948
3070 DATA 192,252,206,230,124,0,126,6,
,12,24,48,48,48,0,124,206,4086
3080 DATA 230,124,206,230,124,0,124,20
06,198,230,126,12,24,48,0,56,5724
3090 DATA 56,0,0,56,56,0,0,0,24,24,0,2
24,24,48,6,12,5660
3100 DATA 24,48,24,12,6,0,0,0,126,0,0,
,126,0,0,96,48,8224
3110 DATA 24,12,24,48,96,0,60,102,102,
,12,24,0,24,0,0,60,7712
3120 DATA 102,110,110,96,62,0,120,156,
,60,54,62,102,102,195,238,115,9006
3130 DATA 99,99,110,99,99,222,60,102,2
204,192,192,192,230,124,238,115,5524
3140 DATA 99,99,99,99,99,222,254,102,9
96,120,96,99,102,124,254,102,1363
3150 DATA 96,120,96,96,96,96,60,102,19
98,192,222,198,102,60,198,198,3274
3160 DATA 198,206,254,230,198,198,48,2
24,24,24,24,24,12,30,12,288
3170 DATA 12,12,12,28,56,96,198,108,10
08,120,120,108,108,198,224,96,320
3180 DATA 96,96,96,102,126,120,198,238
8,254,214,198,198,198,198,198,9268
8
3190 DATA 230,246,222,206,198,198,124,
,206,198,198,198,198,230,124,124,102,6
6630
3200 DATA 102,102,108,96,96,192,124,23
30,198,198,198,198,206,127,238,115,673
36
3210 DATA 102,108,102,102,99,195,58,10
02,96,60,6,6,102,92,254,48,5923
3220 DATA 96,192,192,194,198,124,230,1
102,102,102,102,102,110,63,195,102,365
5
3230 DATA 102,102,102,102,60,24,195,19
95,195,211,203,223,119,98,195,195,5357
7
3240 DATA 102,60,60,102,195,195,195,10
02,102,60,24,24,24,24,126,198,6152
3250 DATA 12,24,254,96,195,254,0,30,24
4,24,24,24,30,0,0,64,9617
3260 DATA 96,48,24,12,6,0,0,120,24,24,
,24,24,120,0,0,8,7258
3270 DATA 20,54,99,0,0,0,0,0,0,0,0,2
255,0,226,2,440
3280 DATA 227,2,0,48,0,0,0,0,0,0,0,0,0
0,0,0,0,3703

```

```

0250 D2Y .DS 1
0260 GX .DS 1
0270 GX1 .DS 1
0280 COLS .DS 1
0290 GOLD .DS 2
0300 TOTL .DS 2
0310 P0YP .DS 1
0320 P1YP .DS 1
0330 LHD .DS 1
0340 HHD .DS 1
0350 GCNT .DS 1
0360 PX .DS 1
0370 PY .DS 1
0380 XHD .DS 1
0390 JHD .DS 1
0400 H51 .DS 1
0410 H52 .DS 1
0420 LINES .DS 1
0430 MZNUM .DS 1
0440 COLOR .DS 1
0450 WSCN .DS 2 ;gr5 a or b
0460 VFLG .DS 1
0470 TFLG .DS 1
0480 TIM .DS 1
0490 SCR1 .DS 2
0500 SCR2 .DS 2
0510 TB1 .DS 8
0520 TB2 .DS 8
0530 .OPT LIST
0540 .OPT NO LIST
0550 ;
0560 RANDOM = $D20A
0570 HPOSP0 = $D000
0580 HPOSP1 = $D001
0590 HITCLR = $D01E
0600 MYSET = $5000
0610 ;
0620 PMB = $8800
0630 PLR0 = $8C00
0640 PLR1 = $8D00
0650 PLR2 = $8E00
0660 PLR3 = $8F00
0670 MAZE0 = $9000
0680 GR5A = $9800
0690 GR5B = $9B00
0700 ;
0710 *= $3000
0720 BEGIN JSR ZH5C ;0 scores
0730 START LDA #0
0740 STA VFLG ;vbi flag
0750 LDA #510
0760 LDX #2 ;set final
0770 F5LP STA F5C1,X ;scores
0780 STA F5C2,X
0790 DEX
0800 BPL F5LP
0810 LDA # >MYSET ;chr set
0820 STA 756
0830 JSR SETPMG ;PMG setup
0840 LDA #7 ;install vbi
0850 LDX # >VBI
0860 LDY # <VBI
0870 JSR $E45C
0880 LDA #0
0890 STA TFLG ;title flag
0900 JSR CLPLRS ;clear plyrs
0910 JSR DEFP0 ;define p0
0920 LDA #0
0930 STA GX ;set ghost
0940 LDA #580 ;positions
0950 STA GX1
0960 LDA #2
0970 STA TIM ;timer
0980 LDA #1
0990 STA VFLG ;turn on vbi
1000 JSR INTRO ;what it says
1010 JSR GAME1 ;play 1st game
1020 JSR NTRM55 ;intermission
1030 LDA #0
1040 STA $022F ;screen off
1050 JSR GAME2 ;middle game
1060 JSR NTRM55
1070 JSR GAME1 ;1st again
1080 JSR NTRM55

```

LISTING 2: ASSEMBLY

```

0100 ;SAVE#D:RACE.M65
0110 ;5-28-88 9:00 P.M.
0120 ;
0130 .OPT NO LIST
0140 *= $90
0150 LO .DS 1
0160 HI .DS 1
0170 I .DS 2
0180 J .DS 2
0190 ML .DS 1
0200 MH .DS 1
0210 ;
0220 D1X .DS 1
0230 D2X .DS 1
0240 D1Y .DS 1

```

```

1090      JMP START      ;do again
1100 ;
1110 GAME1 LDA #0
1120      STA $022F
1130      JSR CLR0      ;clear maze
1140      JSR CLRGR5    ;clear gr5
1150      JSR PRPMZE    ;prepare maze
1160      JSR GENMZ5    ;generate it
1170      JSR COPYT05   ;to gr 5
1180      LDA #528      ;colors
1190      STA $02C4
1200      LDA #5CA
1210      STA $02C5
1220      LDA #594
1230      STA $02C6
1240      JSR SETUP     ;set up screen
1250      JSR CLPLR5    ;clear & define
1260      JSR DEFP0
1270      LDA #2
1280      STA TIM      ;ghost timer
1290      LDA #1
1300      STA VFLG     ;vbi flags
1310      STA TFLG
1320      LDA # <DL5   ;gr5 dlist
1330      STA $0230
1340      LDA # >DL5
1350      STA $0231
1360      LDA #62      ;screen on
1370      STA $022F
1380      LDA D1X      ;dot positions
1390      STA PX
1400      LDA D1Y
1410      STA PY
1420      LDA #1      ;plot color
1430      STA COLOR
1440      LDA # <GR5A  ;top half
1450      STA WSCN
1460      LDA # >GR5A
1470      STA WSCN+1
1480      JSR PLOT5
1490      LDA #3      ;plot color
1500      STA COLOR
1510      LDA D2X      ;dot pos
1520      STA PX
1530      LDA D2Y
1540      STA PY
1550      LDA # <GR5B ;bottom half
1560      STA WSCN
1570      LDA # >GR5B
1580      STA WSCN+1
1590      JSR PLOT5
1600      LDA #0      ;ghosties again
1610      STA GX
1620      LDA #580
1630      STA GX1
1640 ;
1650 ;move 1st dot, then 2nd
1660 ;
1670 STC1 LDA 632     ;stick 0
1680      CMP #50F
1690      BNE ST1
1700      JMP STC2
1710 ST1  CMP #7      ;right?
1720      BNE TRL
1730      LDA D1X
1740      CMP #79
1750      BEQ PJ1
1760      JSR LOKR     ;look right
1770      BCS PJ1
1780      JSR ER5D1    ;erase dot
1790      INC D1X
1800      JSR SHWD1    ;plot it in
1810      JSR WAIT     ;new position
1820      JMP PJ1
1830 TRL  CMP #11     ;left?
1840      BNE TRU
1850      LDA D1X
1860      CMP #1
1870      BEQ PJ1
1880      JSR LOKL
1890      BCS PJ1
1900      JSR ER5D1
1910      DEC D1X
1920      JSR SHWD1
1930      JSR WAIT
1940      JMP PJ1
1950 TRU  CMP #14
1960      BNE TRD
1970      LDA D1Y
1980      CMP #1
1990      BEQ PJ1
2000      JSR LOKU
2010      BCS PJ1
2020      JSR ER5D1
2030      DEC D1Y
2040      JSR SHWD1
2050      JSR WAIT
2060      JMP PJ1
2070 TRD  CMP #13
2080      BNE PJ1
2090      LDA D1Y
2100      CMP #21
2110      BEQ PJ1
2120      JSR LOKD
2130      BCS PJ1
2140      JSR ER5D1
2150      INC D1Y
2160      JSR SHWD1
2170      JSR WAIT
2180 PJ1  JMP STC2
2190 ;
2200 ;erase top dot
2210 ;
2220 ER5D1 LDA #0
2230      STA COLOR
2240 SED1  LDA D1X
2250      STA PX
2260      LDA D1Y
2270      STA PY
2280      LDA # <GR5A
2290      STA WSCN
2300      LDA # >GR5A
2310      STA WSCN+1
2320      STA HITCLR
2330      JSR PLOT5
2340      RTS
2350 ;
2360 ;plot top dot
2370 ;
2380 SHWD1 LDA #1
2390      STA COLOR
2400      JMP SED1
2410 ;
2420 ;move the 2nd dot
2430 ;
2440 STC2 LDA $0279  ;stick 1
2450      CMP #50F
2460      BEQ P5J
2470      CMP #7      ;left
2480      BNE T2L
2490      LDA D2X
2500      CMP #79
2510      BEQ P5J
2520      JSR L2R     ;look right
2530      BCS P5J
2540      JSR ER5D2  ;erase dot
2550      INC D2X
2560 L2L  JSR SHWD2  ;plot dot
2570      JSR WAIT
2580      JMP P5J
2590 T2L  CMP #11
2600      BNE T2U
2610      LDA D2X
2620      CMP #1
2630      BEQ P5J
2640      JSR L2F
2650      BCS P5J
2660      JSR ER5D2
2670      DEC D2X
2680      JMP L2L
2690 T2U  CMP #14
2700      BNE T2D
2710      LDA D2Y
2720      CMP #1
2730      BEQ P5J
2740      JSR L2U

```

```

2750     BCS P5J
2760     JSR ER5D2
2770     DEC D2Y
2780     JMP L2L
2790 T2D  CMP #13
2800     BNE P5J
2810     LDA D2Y
2820     CMP #21
2830     BEQ P5J
2840     JSR L2D
2850     BCS P5J
2860     JSR ER5D2
2870     INC D2Y
2880     JMP L2L
2890 ;
2900 ;is there a winner
2910 ;
2920 PSJ  JSR WINER?
2930     BCS NRGME
2940     STA HITCLR
2950     JSR CHKCOL
2960 ;
2970     LDA GX           ;reset ghosts
2980     CMP #5DA
2990     BCC FOK
3000     JSR DEFP0
3010 FOK  LDA GX1
3020     CMP #5DA
3030     BCC SOK
3040     JSR DEFP1
3050 SOK  LDA 53279      ;chk for
3060     CMP #5           ;select
3070     BNE SSK
3080     PLA
3090     PLA
3100     JMP START
3110 S5K  JMP STC1
3120 ;
3130 NRGME RTS           ;someone won
3140 ;
3150 ;erase bottom dot
3160 ;
3170 ER5D2 LDA D2X
3180     STA PX
3190     LDA D2Y
3200     STA PY
3210     LDA # <GR5B
3220     STA WSCN
3230     LDA # >GR5B
3240     STA WSCN+1
3250     LDA #0
3260     STA COLOR
3270     JSR PLOT5
3280     RTS
3290 ;
3300 ;plot bottom dot
3310 ;
3320 SHWD2 LDA D2X
3330     STA PX
3340     LDA D2Y
3350     STA PY
3360     LDA #3
3370     STA COLOR
3380     STA HITCLR
3390     JSR PLOT5
3400     RTS
3410 ;
3420 ;setup dots
3430 ;
3440 SETUP LDA #1
3450     STA D1X
3460     STA D2X
3470     LDA #11
3480     STA D1Y
3490     STA D2Y
3500     RTS
3510 ;
3520 ;display list for gr 5
3530 ;
3540 DL5  .BYTE $70,$70,$70,$4A
3550     .WORD GR5A
3560     .BYTE $0A,$0A,$0A,$0A,$0A,$0A
3570     .BYTE $0A,$0A,$0A,$0A,$0A,$0A

```

```

3580     .BYTE $0A,$0A,$0A,$0A,$0A
3590     .BYTE $0A,$0A,$0A,$0A,$0A
3600     .BYTE $0A,$0A,$46
3610     .WORD TITLE
3620     .BYTE $4A
3630     .WORD GR5B
3640     .BYTE $0A,$0A,$0A,$0A,$0A,$0A
3650     .BYTE $0A,$0A,$0A,$0A,$0A,$0A
3660     .BYTE $0A,$0A,$0A,$0A,$0A,$0A
3670     .BYTE $0A,$0A,$0A,$0A,$0A,$0A
3680     .BYTE $0A,$0A,$41
3690     .WORD DL5
3700 ;
3710 WAITL LDA #5FE
3720     BNE WT1
3730 WAIT  LDA #5FF
3740 WT1  STA $14
3750 WLP  LDA $14
3760     BNE WLP
3770     STA 77
3780     RTS
3790 ;
3800 ;clear maze
3810 ;
3820 CLR0  LDA # <MAZE0
3830     STA LO
3840     LDA # >MAZE0
3850     STA HI
3860 C03  LDX #7           ;7.5 pages
3870 C02  LDY #0
3880     LDA #0
3890 C01  STA (LO),Y
3900     DEY
3910     BNE C01
3920     INC HI
3930     DEX
3940     BPL C02
3950     RTS
3960 ;
3970 ;prepare maze for generation
3980 ;
3990 PRPMZE LDA # <MAZE0
4000     STA LO
4010     LDA # >MAZE0
4020     STA HI
4030     LDX #22           ;23 lines
4040 PR2  LDY #12
4050     LDA #580
4060 PR1  STA (LO),Y
4070     INY
4080     CPY #65
4090     BNE PR1
4100     LDA LO
4110     CLC
4120     ADC #80
4130     STA LO
4140     LDA HI
4150     ADC #0
4160     STA HI
4170     DEX
4180     BPL PR2
4190     LDA # <MAZE0
4200     CLC
4210     ADC # <800
4220     STA LO
4230     LDA # >MAZE0
4240     ADC # >800
4250     STA HI
4260     LDX #2           ;3 lines
4270 TUN  LDY #0
4280     LDA #580
4290 TUP  STA (LO),Y
4300     INY
4310     CPY #79
4320     BNE TUP
4330     LDA LO
4340     CLC
4350     ADC #80
4360     STA LO
4370     LDA HI
4380     ADC #0
4390     STA HI
4400     DEX

```

```

4410 BPL TUN
4420 RT5
4430 ;
4440 ;generate the middle maze
4450 ;in Antic Mode 4
4460 ;
4470 GENMZ4 LDA # <[MAZE0+381]
4480 STA LO
4490 LDA # >[MAZE0+381]
4500 STA HI
4510 LDX #15
4520 MVU LDA TB41,X
4530 STA TB1,X
4540 DEX
4550 BPL MVU
4560 LDA #6
4570 STA MZNUM
4580 BNE SMZ
4590 ;
4600 ;generate a maze for gr. 5
4610 ;
4620 GENMZ5 LDA # <[MAZE0+901]
4630 STA LO
4640 LDA # >[MAZE0+901]
4650 STA HI
4660 LDX #15
4670 MVT LDA TB51,X
4680 STA TB1,X
4690 DEX
4700 BPL MVT
4710 LDA #580
4720 STA MZNUM
4730 ;
4740 ;the maze generation is
4750 ;is done here
4760 ;
4770 SMZ LDA #5
4780 LDY #0
4790 STA (LO),Y ;start of maze
4800 GN1 LDA RANDOM ;get rand direct
4810 AND #3 ;0-3
4820 STA XHD ;save it
4830 STA JHD ;twice
4840 GN3 LDA XHD ;get direct
4850 ASL A ;x 2
4860 TAX ;0=rt, 1= up
4870 LDA LO ;2=1ft, 3=dn
4880 CLC ;find new pos
4890 ADC TB1,X ;2 away from
4900 STA J ;where we are
4910 LDA HI
4920 ADC TB1+1,X
4930 STA J+1
4940 LDA (J),Y ;what's there
4950 CMP MZNUM ;a wall?
4960 BNE GN2 ;no
4970 LDA XHD ;yes so erase
4980 ASL A ;it
4990 TAX
5000 LDA LO
5010 CLC
5020 ADC TB2,X ;get inbetween
5030 STA I ;location
5040 LDA HI
5050 ADC TB2+1,X
5060 STA I+1
5070 LDA #0 ;zero it
5080 STA (I),Y
5090 INC XHD ;store next
5100 LDA XHD ;direction
5110 STA (J),Y ;in new spot
5120 LDA J ;reset current
5130 STA LO ;location to
5140 LDA J+1 ;new pos
5150 STA HI
5160 JMP GN1 ;again
5170 GN2 INC XHD ;try next
5180 LDA XHD ;direction
5190 AND #3
5200 STA XHD ;if same as
5210 CMP JHD ;old-go back
5220 BNE GN3 ;try this direct
5230 LDY #0 ;going back

```

```

5240 LDA (LO),Y
5250 STA XHD
5260 LDA #0
5270 STA (LO),Y
5280 LDA XHD ;have we come
5290 CMP #5 ;back to begin?
5300 BNE GN4 ;no
5310 RT5 ;yes-done!
5320 GN4 INC XHD ;try next direct
5330 LDA XHD ;and see if it
5340 AND #3 ;will work
5350 ASL A
5360 TAX
5370 CLC
5380 LDA LO
5390 ADC TB1,X
5400 STA LO
5410 LDA HI
5420 ADC TB1+1,X
5430 STA HI
5440 JMP GN1
5450 ;
5460 ;tables to get rt,lf,up,dn
5470 ;for gr5 and antic 5
5480 ;
5490 TB51 .BYTE 2,0,96,255
5500 .BYTE 254,255,160,0
5510 TB52 .BYTE 1,0,176,255
5520 .BYTE 255,255,80,0
5530 TB41 .BYTE 2,0,176,255
5540 .BYTE 254,255,80,0
5550 TB42 .BYTE 1,0,216,255
5560 .BYTE 255,255,40,0
5570 ;
5580 ;plot routine for gr. 5
5590 ;
5600 PLOT5 LDA PY
5610 ASL A
5620 STA ML
5630 LDA #0
5640 ROL A
5650 STA MH
5660 ASL ML
5670 ROL MH ;X4
5680 LDA ML
5690 STA LHD
5700 LDA MH
5710 STA HHD
5720 ASL ML
5730 ROL MH ;X8
5740 ASL ML
5750 ROL MH ;X16
5760 LDA ML
5770 CLC
5780 ADC LHD
5790 STA ML
5800 LDA MH
5810 ADC HHD
5820 STA MH
5830 LDA WSCN
5840 CLC
5850 ADC ML
5860 STA ML
5870 LDA WSCN+1
5880 ADC MH
5890 STA MH
5900 LDA PX
5910 AND #3
5920 TAX
5930 LDA PX
5940 LSR A
5950 LSR A
5960 CLC
5970 ADC ML
5980 STA LO
5990 LDA MH
6000 ADC #0
6010 STA HI
6020 LDY COLOR
6030 LDA BMA5K2,X
6040 AND COLOR5,Y
6050 STA LHD
6060 LDA BMA5K1,X

```



```

6070     LDY #0
6080     AND (L0),Y
6090     ORA LHD
6100     STA (L0),Y
6110     RTS
6120 ;
6130 ;copy maze to gr5 screen
6140 ;
6150 COPYT05 LDA #0 ;24 lines
6160     STA LINES
6170     STA PX ;reset px
6180     LDA #2
6190     STA COLOR
6200     LDA # <GR5A
6210     STA WSCN
6220     LDA # >GR5A
6230     STA WSCN+1
6240     LDA # <MAZE0
6250     STA J
6260     LDA # >MAZE0
6270     STA J+1
6280 NLN LDY #0
6290 NXY LDA (J),Y
6300     BEQ NEXB
6310     STY PX
6320     LDA LINES
6330     STA PY
6340     JSR PLOT5
6350 NEXB INC PX
6360     LDY PX
6370     CPY #80
6380     BNE NXY
6390     INC LINES
6400     LDA LINES
6410     CMP #23
6420     BEQ CDOM
6430     LDA J
6440     CLC
6450     ADC #80
6460     STA J
6470     LDA J+1
6480     ADC #0
6490     STA J+1
6500     JMP NLN
6510 CDOM LDY #0
6520 MAB LDA GR5A,Y
6530     STA GR5B,Y
6540     LDA GR5A+$0100,Y
6550     STA GR5B+$0100,Y
6560     INY
6570     BNE MAB
6580     RTS
6590 ;
6600 ;clear out gr5 screen memry
6610 ;
6620 CLRGR5 LDA # >GR5A
6630     STA HI
6640     LDA # <GR5A
6650     STA L0
6660     LDX #2
6670 L5A LDY #0
6680     LDA #0
6690 L5 STA (L0),Y
6700     INY
6710     BNE L5
6720     INC HI
6730     DEX
6740     BPL L5A
6750     RTS
6760 ;
6770 COLORS .BYTE $00,$55,$AA,$FF
6780 BMASK1 .BYTE $3F,$CF,$F3,$FC
6790 BMASK2 .BYTE $C0,$30,$0C,$03
6800 ;
6810 ;look for walls
6820 ;to see if movement
6830 ;is ok
6840 ;
6850 LOKR LDA D1Y
6860     LDX D1X
6870     INX
6880 LLL JSR MUL80
6890     RTS

```

```

6900 ;
6910 LOKL LDA D1Y
6920     LDX D1X
6930     DEX
6940     JMP LLL
6950 LOKU LDX D1Y
6960     DEX
6970     TXA
6980     LDX D1X
6990     JMP LLL
7000 ;
7010 LOKD LDX D1Y
7020     INX
7030     TXA
7040     LDX D1X
7050     JMP LLL
7060 ;
7070 ;multiply by 80 for gr5 screen
7080 ;
7090 MUL80 ASL A
7100     STA LHD ;x2
7110     LDA #0
7120     ROL A
7130     STA HHD
7140     LDY #2
7150 M8A ASL LHD
7160     ROL HHD ;x16
7170     DEY
7180     BPL M8A
7190     LDA LHD
7200     STA ML
7210     LDA HHD
7220     STA MH
7230     LDY #1
7240 M8B ASL LHD
7250     ROL HHD ;x64
7260     DEY
7270     BPL M8B
7280     LDA LHD ;16+64=X80
7290     CLC
7300     ADC ML
7310     STA ML
7320     LDA HHD
7330     ADC MH
7340     STA MH
7350     LDA # <MAZE0
7360     CLC
7370     ADC ML
7380     STA I
7390     LDA # >MAZE0
7400     ADC MH
7410     STA I+1
7420     TXA
7430     CLC
7440     ADC I
7450     STA I
7460     LDA I+1
7470     ADC #0
7480     STA I+1
7490     LDY #0
7500     LDA (I),Y
7510     CMP #580
7520     BNE LOK
7530     SEC
7540     RTS
7550 LOK CLC
7560     RTS
7570 ;
7580 ;look routines for 2nd player
7590 ;
7600 L2R LDA D2Y
7610     LDX D2X
7620     INX
7630     JSR MUL80
7640     RTS
7650 ;
7660 L2F LDA D2Y
7670     LDX D2X
7680     DEX
7690     JSR MUL80
7700     RTS
7710 ;
7720 L2U LDX D2Y

```

```

7730     DEX
7740     TXA
7750     LDX D2X
7760     JSR MUL80
7770     RTS
7780 ;
7790 L2D LDX D2Y
7800     INX
7810     TXA
7820     LDX D2X
7830     JSR MUL80
7840     RTS
7850 ;
7860 ;display list for antic 4 scrn
7870 ;
7880 DL4 .BYTE $70,$70,$70,$44
7890     .WORD MAZE0
7900     .BYTE $04,$04,$04,$04,$04
7910     .BYTE $04,$04,$04,$04,$04
7920     .BYTE $04,$04,$04,$04,$04
7930     .BYTE $04,$04,$04,$04,$04
7940     .BYTE $04,$46
7950     .WORD SCRTXT
7960     .BYTE $41
7970     .WORD DL4
7980 ;
7990 ;see if there's a winner
8000 ;add 100 to winner's score
8010 ;
8020 WINER? LDA D1X
8030     CMP #77
8040     BEQ MZ1
8050     LDA D2X
8060     CMP #77
8070     BEQ MZ2
8080 ;
8090     CLC
8090     RTS
8100 MZ1 INC F5C1
8110     JSR AHUN1
8120     SEC
8130     RTS
8140 MZ2 INC F5C2
8150     JSR AHUN2
8160     SEC
8170     RTS
8180 ;
8190 ;set up for Middle maze
8200 ;
8210 PRPMID JSR CLR0
8220     LDA # <MAZE0
8230     STA LO
8240     LDA # >MAZE0
8250     STA HI
8260     LDX #20
8270 PMA LDY #0
8280     LDA #6
8290 PMC STA (LO),Y
8300     INY
8310     CPY #39
8320     BNE PMC
8330     LDA LO
8340     CLC
8350     ADC #40
8360     STA LO
8370     LDA HI
8380     ADC #0
8390     STA HI
8400     DEX
8410     BPL PMA
8420     RTS
8430 ;
8440 ;setup player graphics
8450 ;
8460 SETPMG LDA #62 ;single line res
8470     STA $022F
8480     LDA #1
8490     STA $026F
8500     LDA #566 ;colors
8510     STA $02C0
8520     LDA #526
8530     STA $02C1
8540     LDA #50E
8550     STA $02C2

8560     STA $02C3
8570     LDA #0
8580     STA HP05P0
8590     STA HP05P1
8600     STA HP05P0+2
8610     LDA #1 ;width
8620     STA $D008
8630     STA $D009
8640     STA $D00A
8650     STA $D00B
8660     LDA #3
8670     STA $D01D
8680     LDA # >PMB
8690     STA $D407
8700     RTS
8710 ;
8720 ;put player at random vertical
8730 ;position on screen within
8740 ;a certain range
8750 ;
8760 DEFP1 JSR CLRPI ;does bottom
8770     LDA RANDOM ;ghost
8780     AND #$1F
8790     CLC
8800     ADC #$47
8810     STA P1YP
8820     LDX #0
8830     TAY
8840 P1L LDA PDF1,X
8850     STA PLR1,Y
8860     STA PLR1+$64,Y
8870     INY
8880     STA PLR1,Y
8890     STA PLR1+$64,Y
8900     INY
8910     INX
8920     CPX #12
8930     BNE P1L
8940     LDY P1YP
8950     LDA #$14
8960     STA PLR3+4,Y
8970     STA PLR3+5,Y
8980     STA PLR3+$68,Y
8990     STA PLR3+$69,Y
9000     LDA #$1C
9010     STA PLR3+$0A,Y
9020     STA PLR3+$0B,Y
9030     STA PLR3+$6E,Y
9040     STA PLR3+$6F,Y
9050     RTS
9060 ;
9070 ;place top ghost on screen
9080 ;
9090 DEFP0 JSR CLRPO
9100     LDA RANDOM
9110     AND #$1F
9120     CLC
9130     ADC #$20
9140     STA P0YP
9150     LDX #0
9160     LDY P0YP
9170 P0L LDA PDF1,X
9180     STA PLR0,Y
9190     STA PLR0+$64,Y
9200     INY
9210     STA PLR0,Y
9220     STA PLR0+$64,Y
9230     INY
9240     INX
9250     CPX #12
9260     BNE P0L
9270     LDA #$14
9280     LDY P0YP
9290     STA PLR2+4,Y
9300     STA PLR2+5,Y
9310     STA PLR2+$68,Y
9320     STA PLR2+$69,Y
9330     LDA #$1C
9340     STA PLR2+$0A,Y
9350     STA PLR2+$0B,Y
9360     STA PLR2+$6E,Y
9370     STA PLR2+$6F,Y
9380     RTS

```

```

9390 ;
9400 ;erase players 0 & 2
9410 ;
9420 CLRPO LDX #23
9430 LDA #0
9440 LDY P0YP
9450 C0L STA PLR0,Y
9460 STA PLR2,Y
9470 STA PLR0+$64,Y
9480 STA PLR2+$64,Y
9490 INY
9500 DEX
9510 BPL C0L
9520 RTS
9530 ;
9540 ;erase players 1 & 3
9550 ;
9560 CLRPI LDX #23
9570 LDA #0
9580 LDY P1YP
9590 C1L STA PLR1,Y
9600 STA PLR1+$64,Y
9610 STA PLR3,Y
9620 STA PLR3+$64,Y
9630 INY
9640 DEX
9650 BPL C1L
9660 RTS
9670 ;
9680 ;the ghosties
9690 ;
9700 PDF1 .BYTE $1C,$3E,$6B,$7F
9710 .BYTE $7F,$63,$7F,$7F
9720 .BYTE $7F,$7F,$0A,$00
9730 ;
9740 ;move the ghosts
9750 ;
9760 MGHOST INC GX
9770 INC GX1
9780 LDA GX
9790 STA HPO5P0
9800 STA HPO5P0+2
9810 LDA GX1
9820 STA HPO5P1
9830 STA HPO5P1+2
9840 RTS
9850 ;
9860 ;clear out all plyrs
9870 ;
9880 CLPLRS LDX #0
9890 TXA
9900 CPI STA PLR0,X
9910 STA PLR1,X
9920 STA PLR2,X
9930 STA PLR3,X
9940 INX
9950 BNE CPI
9960 RTS
9970 ;
9980 ;chekc for collisions
9990 ;send anyone hit back to
010000 ;beginning of maze
010010 ;
010020 CHKCOL LDA $D004
010030 STA COL5
010040 AND #1
010050 BEQ KB
010060 FIR JSR ERSD1
010070 LDA #1
010080 STA D1X
010090 LDA #11
010100 STA D1Y
010110 JSR SHWD1
010120 RTS
010130 KB LDA COL5
010140 AND #4
010150 BEQ TRP1
010160 SCD JSR ERSD2
010170 LDA #1
010180 STA D2X
010190 LDA #11
010200 STA D2Y
010210 JSR SHWD2
010220 RTS
010230 ;
010240 TRP1 LDA $D005
010250 STA COL5
010260 AND #1
010270 BEQ KC
010280 JMP FIR
010290 KC LDA COL5
010300 AND #4
010310 BNE SCD
010320 RTS
010330 ;
010340 ;the vertical blank interrupt
010350 ;make some sounds,move the
010360 ;ghosts, & change their
010370 ;luminance
010380 ;
010390 VBI LDA VFLG ;off or
010400 BNE UVV ;on
010410 LDA #0 ;no sound
010420 STA $D200
010430 STA $D201
010440 BEQ NOV
010450 UVV JSR MGHOST
010460 INC GCNT
010470 LDA GCNT ;for luminance
010480 AND #3 ;changes in
010490 TAX ;color
010500 STX GCNT
010510 LDA UP1,X
010520 STA $02C0
010530 LDA UP2,X
010540 STA $02C1
010550 LDA TFLG ;title flag
010560 BEQ NOCHK ;don't chk for
010570 JSR CHKCOL ;collisions
010580 ;
010590 NOCHK DEC TIM ;on title
010600 LDA TIM ;screen
010610 BNE NOV
010620 INC NTX
010630 LDA NTX
010640 AND #3 ;play notes
010650 STA NTX
010660 TAX
010670 LDA #2
010680 STA TIM
010690 LDA NOTE,X
010700 STA $D200
010710 LDA #$A6
010720 STA $D201
010730 NOV JMP $E462 ;xit
010740 ;
010750 ;the middle maze
010760 ;
010770 GAME2 LDA # <DL4 ;new
010780 STA $0230 ;display list
010790 LDA # >DL4
010800 STA $0231
010810 JSR STOPAL ;no ghosts
010820 LDA #$94 ;screen colrs
010830 STA $02C4
010840 LDA #$F6
010850 STA $02C5
010860 LDA #$44
010870 STA $02C6
010880 JSR PRPMID ;prepare middle
010890 JSR GENMZ4 ;maze & generate
010900 JSR PUTGLD ;fill w/gold
010910 LDA #62
010920 STA $022F ;screen on
010930 JSR COLLECT ;go for gold!
010940 RTS
010950 ;
010960 ;fill the maze with gold
010970 ;
010980 PUTGLD LDA # <MAZE0
010990 STA LO
011000 LDA # >MAZE0
011010 STA HI
011020 LDA #0
011030 STA GOLD
011040 STA GOLD+1

```

```

011050 LDX #18
011060 GLL LDA LO
011070 CLC
011080 ADC #40
011090 STA LO
011100 LDA HI
011110 ADC #0
011120 STA HI
011130 LDY #1
011140 GLP LDA (LO),Y
011150 BNE GNY
011160 LDA #7 ;gold
011170 STA (LO),Y
011180 INC GOLD
011190 LDA GOLD
011200 BNE GNY
011210 INC GOLD+1
011220 GNY INY
011230 CPY #38
011240 BNE GLP
011250 DEX
011260 BPL GLL
011270 LDA GOLD
011280 SEC
011290 SBC #2
011300 STA GOLD
011310 LDA GOLD+1
011320 SBC #0
011330 STA GOLD+1
011340 RTS
011350 ;
011360 ;here's where we fight over
011370 ;the gold coins
011380 ;
011390 COLLECT LDA #1 ;staring pos.
011400 STA D1X
011410 LDA #10
011420 STA D1Y
011430 JSR MCH1 ;show man
011440 LDA #37 ;2nd man
011450 STA D2X
011460 LDA #10
011470 STA D2Y
011480 JSR MCH2 ;show him
011490 LDA #0 ;zero scores
011500 LDX #3
011510 SCC STA SCR1,X
011520 DEX
011530 BPL SCC
011540 LDX #2 ;screen scores
011550 LDA #510
011560 SCF STA SCRTXT+10,X
011570 STA SCRTXT+16,X
011580 DEX
011590 BPL SCF
011600 ;
011610 C51 JSR MCH1 ;show 1st man
011620 LDA $0278 ;stick 0
011630 CMP #50F
011640 BEQ C52
011650 CMP #7
011660 BNE CTL ;move right?
011670 JSR CLR ;chk 4 wall,etc
011680 BCS C52
011690 JSR CHGLD1 ;chk for gold
011700 JSR PCHR1
011710 INC D1X
011720 CGEN JSR MCH1 ;show move
011730 JSR WAITL ;delay
011740 JMP C52 ;plyr 2
011750 ;
011760 CTL CMP #11 ;left?
011770 BNE CTU
011780 JSR CLF
011790 BCS C52
011800 JSR CHGLD1
011810 JSR PCHR1
011820 DEC D1X
011830 JMP CGEN
011840 ;
011850 CTU CMP #14 ;up
011860 BNE CTD
011870 JSR CLU

```

```

011880 BCS C52
011890 JSR CHGLD1
011900 JSR PCHR1
011910 DEC D1Y
011920 JMP CGEN
011930 CTD CMP #13 ;down?
011940 BNE C52
011950 JSR CLD
011960 BCS C52
011970 JSR CHGLD1
011980 JSR PCHR1
011990 INC D1Y
012000 JMP CGEN
012010 ;
012020 C52 JSR MCH2 ;repeat for
012030 LDA $0279 ;2nd palyr
012040 CMP #50F
012050 BEQ MWIN
012060 CMP #7
012070 BNE ELF
012080 JSR C2R
012090 BCS MWIN
012100 JSR CHGLD2
012110 JSR PCH2
012120 INC D2X
012130 EGEN JSR MCH2
012140 JSR WAITL
012150 JMP MWIN
012160 ;
012170 ELF CMP #11
012180 BNE ELU
012190 JSR C2L
012200 BCS MWIN
012210 JSR CHGLD2
012220 JSR PCH2
012230 DEC D2X
012240 JMP EGEN
012250 ;
012260 ELU CMP #14
012270 BNE ELD
012280 JSR C2U
012290 BCS MWIN
012300 JSR CHGLD2
012310 JSR PCH2
012320 DEC D2Y
012330 JMP EGEN
012340 ;
012350 ELD CMP #13
012360 BNE MWIN
012370 JSR C2D
012380 BCS MWIN
012390 JSR CHGLD2
012400 JSR PCH2
012410 INC D2Y
012420 JMP EGEN
012430 ;
012440 ;has anyone won?
012450 ;
012460 MWIN LDA SCR1 ;get sum
012470 CLC
012480 ADC SCR2
012490 STA TOTL
012500 LDA SCR1+1
012510 ADC SCR2+1
012520 STA TOTL+1
012530 CMP GOLD+1 ;does sum =
012540 BNE FLIP ;total gold?
012550 LDA TOTL
012560 CMP GOLD
012570 BNE FLIP
012580 RTS ;somebody won
012590 ;
012600 FLIP LDA $D01F ;select?
012610 CMP #5
012620 BNE FFP
012630 PLA
012640 PLA
012650 JMP START ;yes,start ovr
012660 FFP JMP C51
012670 ;
012680 ;chk for walls
012690 ;
012700 CLR LDA D1Y

```

```

012710 LDX D1X
012720 INX
012730 CLLL JSR MUL40
012740 LDY #0
012750 LDA (I),Y
012760 CMP #6
012770 BEQ WALL
012780 GG CLC
012790 RTS
012800 WALL SEC
012810 RTS
012820 ;
012830 CLF LDA D1Y
012840 LDX D1X
012850 DEX
012860 JMP CLLL
012870 ;
012880 CLU LDY D1Y
012890 DEY
012900 TYA
012910 LDX D1X
012920 JMP CLLL
012930 ;
012940 CLD LDY D1Y
012950 INY
012960 TYA
012970 LDX D1X
012980 JMP CLLL
012990 ;
013000 ;chks for wall for man 2
013010 ;
013020 C2R LDA D2Y
013030 LDX D2X
013040 INX
013050 JMP CLLL
013060 ;
013070 C2L LDA D2Y
013080 LDX D2X
013090 DEX
013100 JMP CLLL
013110 ;
013120 C2U LDY D2Y
013130 DEY
013140 TYA
013150 LDX D2X
013160 JMP CLLL
013170 ;
013180 C2D LDY D2Y
013190 INY
013200 TYA
013210 LDX D2X
013220 JMP CLLL
013230 ;
013240 ;zero out top
013250 ;
013260 PCHR1 LDA D1Y
013270 LDX D1X
013280 JSR MUL40
013290 LDY #0
013300 LDA #0
013310 STA (I),Y
013320 RTS
013330 ;
013340 ;zero out bot man
013350 ;
013360 PCH2 LDA D2Y
013370 LDX D2X
013380 JSR MUL40
013390 LDY #0
013400 TYA
013410 STA (I),Y
013420 RTS
013430 ;
013440 ;show top man
013450 ;
013460 MCH1 LDA D1Y
013470 LDX D1X
013480 JSR MUL40
013490 LDY #0
013500 LDA #0
013510 STA (I),Y
013520 RTS
013530 ;

```

```

013540 ;show bot man
013550 ;
013560 MCH2 LDA D2Y
013570 LDX D2X
013580 JSR MUL40
013590 LDY #0
013600 LDA #9
013610 STA (I),Y
013620 RTS
013630 ;
013640 ;mult x 40 to get screen pos
013650 ;
013660 MUL40 ASL A
013670 STA LHD
013680 LDA #0
013690 ROL A
013700 STA HHD
013710 ASL LHD ;x4
013720 ROL HHD
013730 ASL LHD ;x8
013740 ROL HHD
013750 LDA LHD
013760 STA ML
013770 LDA HHD
013780 STA MH
013790 ASL LHD ;x16
013800 ROL HHD
013810 ASL LHD ;32
013820 ROL HHD
013830 LDA LHD
013840 CLC
013850 ADC ML
013860 STA ML
013870 LDA HHD
013880 ADC MH
013890 STA MH
013900 LDA # <MAZE0
013910 CLC
013920 ADC ML
013930 STA I
013940 LDA # >MAZE0
013950 ADC MH
013960 STA I+1
013970 TXA
013980 CLC
013990 ADC I
014000 STA I
014010 LDA I+1
014020 ADC #0
014030 STA I+1
014040 RTS
014050 ;
014060 ;is it gold we are moving
014070 ;onto- for top man
014080 ;
014090 CHGLD1 CMP #7 ;gold?
014100 BEQ GD1 ;yes
014110 RTS
014120 GD1 JSR BELL ;sounds
014130 INC SCR1 ;inc. score
014140 BNE DGA
014150 INC SCR1+1
014160 DGA LDX #2 ;show score
014170 INC SCRTXT+10,X
014180 LDA SCRTXT+10,X
014190 DGC CMP #51a
014200 BCC F5T
014210 LDA #510
014220 STA SCRTXT+10,X
014230 DEX
014240 BMI F5T
014250 INC SCRTXT+10,X
014260 LDA SCRTXT+10,X
014270 JMP DGC
014280 ;
014290 F5T JSR ADDH1 ;add 1 to hi
014300 LDX #2 ;scores
014310 INC F5C1,X
014320 LDA F5C1,X
014330 F5R CMP #51a
014340 BCC DGB
014350 LDA #510
014360 STA F5C1,X

```

014370 DEX
 014380 BMI DGB
 014390 INC F5C1,X
 014400 LDA F5C1,X
 014410 JMP F5R
 014420 DGB RTS
 014430 ;
 014440 ;do same for bottom man
 014450 ;
 014460 CHGLD2 CMP #7
 014470 BEQ EG1
 014480 RTS
 014490 EG1 JSR BELL
 014500 INC 5CR2
 014510 BNE EGA
 014520 INC 5CR2+1
 014530 EGA LDX #2
 014540 INC SCRTXT+16,X
 014550 LDA SCRTXT+16,X
 014560 EGC CMP #51A
 014570 BCC FSU
 014580 LDA #510
 014590 STA SCRTXT+16,X
 014600 DEX
 014610 BMI FSU
 014620 INC SCRTXT+16,X
 014630 LDA SCRTXT+16,X
 014640 JMP EGC
 014650 ;
 014660 FSU JSR ADDH2

014670 LDX #2
 014680 INC F5C2,X
 014690 LDA F5C2,X
 014700 F5V CMP #51A
 014710 BCC EGB
 014720 LDA #510
 014730 STA F5C2,X
 014740 DEX
 014750 BMI EGB
 014760 INC F5C2,X
 014770 LDA F5C2,X
 014780 JMP F5V
 014790 EGB RTS
 014800 ;
 014810 ;intermission time
 014820 ;show current scores
 014830 ;
 014840 NTRM55 LDA # <IL5T
 014850 STA \$0230
 014860 LDA # >IL5T
 014870 STA \$0231
 014880 JSR STOPAL
 014890 551 LDA \$0284
 014900 BNE 552
 014910 RR1 LDA \$0284
 014920 BEQ RR1
 014930 RTS
 014940 552 LDA \$0285
 014950 BNE 551
 014960 RR2 LDA \$0285

NEW HACK PACK Special OFFER

The Alpha Systems HACK PACK contains all our finest products for making Back-up copies, Analyzing, Understanding and Protecting your Atari programs. It comes complete with Atari

Protection Techniques (Book and Disk I), The Chipmunk, The Scanalyzer, The Impersonator and Disk Pack 1000. Worth over \$150. Get them all for the special price of **Just \$99.95**

Atari Software Protection Techniques Vol I & II

These Book and Disk packages detail the most advanced copy protection methods in use today. They guide you through the methods used to create the protection as well as the copying techniques to get around them. They include information on Phreaking • Hacking • On-line security • Black boxes • Self-destructing programs • Pirate bulletin board systems • Logic bombs • New piracy laws • Hardware data keys • Weak sectoring (Phantom, Fuzzy and unstable sectors) • Overflowed tracks • CRC errors • Bank Select cartridges and MUCH, MUCH MORE. The disks include automatic program protectors, Protection Scanners, directory hiding and more.

BOOK I and DISK I \$24.95
BOOK II (Advanced protection) and DISK II \$24.95
Special Offer, Order both sets for Only \$39.95

CHIPMUNK

Automatic Disk Back-Up System. Make perfectly running unprotected back-up copies of hundreds of the most popular Atari programs. Chipmunk's sophisticated programming Automatically finds and **REMOVES copy protection** from most Atari programs. Back-up even heavily protected programs with ease. Finally, a back-up system that needs no special hardware or skills.

(If you need a full list of what Chipmunk copies, call or write for our free catalog) **\$34.95**

Scanalyzer Automatically scan & analyze commercial programs. Unlock programming secrets and learn from the masters **\$29.95**

Impersonator Cartridge to Disk back up system. Create running back-up copies of any cartridge (up to 16K) **\$29.95**

NEW CHEAT

Get more from your games with CHEAT Tired of spending days trying to beat a game? Tired of getting stuck just when you need another life? Cheat is an innovative new product that gives you the chance you need to beat your favorite games. Cheat works with hundreds of Atari games to give you unlimited lives or power. End the frustration and get hours more enjoyment from your games. (Call or write Alpha Systems for our free catalog with a full list of the programs that work with Cheat) **ONLY \$24.95**

BASIC TURBOCHARGER

NOW for the first time a BASIC programmer can get the power, flexibility and incredible speed of machine language. BASIC TURBOCHARGER is a **book and disk package** that contains over 150 ready to use machine language routines. Complete instructions show how to add them to your own BASIC programs to get these features and more: • Smooth Scrolling • Player/Missile control • Load & Save Picture files • Sorting and Searching • Special Effects Graphics • Incredible Speed • Much, Much More • Over 150 programs. You've heard of the power of Assembler, now harness it for your own needs. **\$24.95**



24 HOUR HOTLINE **216-374-7469**

VISA & MASTERCARD, ORDER BY PHONE, OR SEND MONEY ORDER TO:

ATARI 8-BIT POWER

ALPHA SYSTEMS is constantly innovating to provide more power for your 8-bit Ataris

NEW PARROT II

An All New Parrot sound digitizer for your Atari. Parrot II is a sophisticated new hardware device that plugs into your joystick port. Parrot II has two inputs. One for a microphone and one for a powered source such as a tape player, radio or Compact Disk.

The Powerful Parrot II software lets you record sounds into your computer and play them back on any Atari. Parrot II turns your computer's keyboard into a musical instrument with nine different sounds covering three octaves each. The sounds can be anything, a dogs bark, a piano, a complete drum set, a symphony or your own voice.

Parrot II lets you modify the sounds on a graphic display to create brand new sounds and special effects. Best of all, the sounds and voices can be put into your own programs that can be used on any standard Atari. Explore the world of digital sound and music. **ONLY \$59.95**

Pre-Recorded Sound Disk More pre-recorded sounds for Parrot **\$4.95**
PARROT II Demo Disk (Does not require Parrot to run) **\$5.00**

NEW POP-N-ROCKER

a fast paced, multi-player trivia game that mixes questions with real songs (digitized) with Parrot. Be the first to identify the songs and answer the music trivia questions. *Pop-N-Rocker* comes with three data disks and lets you add new questions so it will never get old. You can use a Parrot Sound digitizer to add new songs too! Use any kind of music from Rock to Classical to Nursery Rhymes. A new concept in entertainment and a perfect add-on for Parrot. **\$24.95**

COMPUTEREYES & MAGNIPRINT II +

Turn your computer into a digital portrait studio. This complete package lets you **capture, save & print** digital images from your **Video Camera, VCR or TV**. **COMPUTEREYES** hardware plugs directly into your joystick ports for easy use. Print your picture on a 6 foot poster. **\$119.95**

ComputerEyes camera system

Comes complete with everything above, plus a black and white video camera and connecting cable. **\$329.95**

Graphics 9 Software - Add a new dimension to your **COMPUTEREYES** pictures - captures images in 16 shades of grey. **\$12.00**

Magniprint II +

Easily the most powerful print program available today. Print graphics from almost any format in hundreds of shapes, sizes, and shades. Supports **color printing** and lets you create **giant posters**. Magniprint II+ lets you stretch and squeeze, invert, add text, adjust shading and much more. Works with EPSON, NEC, Citoh, Panasonic, Gemini, Star, XMM801, and compatible printers. (850 interface or equivalent required). **\$24.95**

Graphics Transformer

Now you can combine the most powerful features of all your graphics programs. Create print shop icons from a Koala pad picture, from a photo digitized with ComputerEyes, or any picture file. Graphics Transformer lets you **Shrink, Enlarge and Merge** pictures for unequaled flexibility. **\$22.95**

YOUR ATARI COMES ALIVE

SAVE MONEY! Finally an alternative to buying expensive computer add-ons. Your Atari Comes Alive shows you how to **build them yourself**. This "How-To" **book and disk package** gives you complete step by step instructions and programs needed to build and control these exciting devices and MORE: • Light Pen • Light & Motor Controllers • Alarm Systems • Voice Recognition • Environmental Sensors • Data Decoders • More than 150 pages. **Your Atari Comes Alive** **\$24.95**

ALPHA SYSTEMS 1012 SKYLAND DRIVE MACEDONIA, OH 44056 FREE BONUS: DELUXE SPACE GAMES (3 games on a disk) Free with any order of 3 or more items. Include \$3.00 ship & hldg (US Canada) Ohio res. add 5 1/2% sales tax. Foreign orders add \$8.00 ship & hldg. Call or write for free catalog. Customer Service Line (216) 467-5665 M-F 9-3.



GIANT WALL SIZED POSTERS.

Attention Programmers!

ANALOG Computing is interested in programs, articles, and software review submissions dealing with the Atari 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.

All submissions for publication, both program listings and text, should be provided in printed and magnetic form. Typed or printed copy of text is mandatory and should be in upper and lower case with double spacing. By submitting articles to **ANALOG Computing**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ANALOG Computing**. 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 your programs and articles to:
ANALOG Computing
P.O. Box 1413-M.O.
Manchester, CT 06040-1413

continued from page 9

JOYSTICKS	
-----0	means that joystick zero is moved up.
-----0-	means that joystick zero is moved down.
-----0--	means that joystick zero is moved left.
-----0---	means that joystick zero is moved right.
---0----	means that joystick one is moved up.
--0-----	means that joystick one is moved down.
-0-----	means that joystick one is moved left.
0-----	means that joystick one is moved right.

PADDLES	
----0--	means that paddle zero's button is pressed.
---0---	means that paddle ones' button is pressed.
-0-----	means that paddle two's button is pressed.
0-----	means that paddle three's button is pressed.

Figure 1. PORTA (paddles/joystick) bit chart

Substitute "is not" for "is" in the preceding descriptions if the bit is set to 1.

The shadow registers for PORTA in this sense are STICK0 and STICK1 (632 and 633) and PTRIG0 through PTRIG3 (636 through 639).

If Bit 2 of PACTL is set, then PORTA writes to the direction-control register. A direction-control register, as the name implies, is used to specify the direction that information (data) is traveling on the various port pins. What are port pins? Take a close look at the controller ports and you can see them. They are numbered 1 through 5 on the top row (left to right) and 6 through 9 on the bottom. PORTA only deals with 1 through 4; Bits 0 through 3 represent Pins 1 through 4 on Jack 1, while Bits 4 through 7 represent Pins 1 through 4 on Jack 2. When Bit 2 of PACTL is set, a bit set to 1 in PORTA means that the corresponding pin will be used for output. Similarly, a bit set to 0 means that pin will be used for input.

You may be wondering what the other port pins are used for. Pin 5 is used for the right-paddle value (POT1/3/5/7), Pin 6 for the joystick button (TRIG0/1/2/3), Pin 7 supplies five volts to the paddles (this pin isn't connected in the joysticks), Pin 8 is the ground (for both joysticks and paddle), and Pin 9 is the left-paddle value (POTO/2/4/6).

PORTB
54017 D301

PORTB is the same as PORTA, except it's used with Controller Jacks 3 and 4 rather than 2 and 3. Also, its function is determined by PBCTL, not PACTL.

The shadow registers for PORTB and STICK2 (634 and 635) and PTRIG4 through PTRIG7 (640 through 643).

PACTL
54018 D302

If you have a cassette player hooked up to your computer, try putting a music cassette in it, pressing PLAY and then entering the following statement:

POKE 54018, 52

This turns on the cassette motor and lets you play music or voice through the TV speaker. To turn off the motor, use the following:

POKE 54018, 60

Other uses of PACTL, which is also called the "Port B Controller," are shown in Figure 2.

A few words on the preceding values before we move on. PACTL is initialized to 60, which means that the cassette motor is turned off (Bit 3) and PORTA will read and write to the first two controller jacks (Bit 2). This piece of information should hopefully clear up a question you might have had concerning turning the cassette motor on and off. The reason the POKEs I gave you earlier are different than the bit values above is that Bit 2 should be on in order for the joysticks and paddles to work properly. That's why we use 52 and 60 in the POKEs instead of 48 and 56.

PBCTL
54019 D303

This is the Port B Controller and has the same functions as PACTL with the following differences:

1. Bits 0 and 7 deal with the "peripheral interrupt available" interrupts.

2. Bit 2 deals with PORTB.

3. Bit 3 no longer controls the cassette motor but is instead used for peripheral command identification. It is not clear anywhere as to exactly what this means (although most sources also label it as the "serial bus command line"). It is initialized to 1.

POKEing 54019 with a 56 tells the computer to take the next POKE to 54017 as a data-direction code. This is a binary code with each bit corresponding to a pin on the jacks. If the corresponding bit is 1, the pin is defined as output, and if it is 0, then the pin is input. Once you have completed that section of the code, you may POKE to 554017 whatever you may want to send out. If you POKE there and then PEEK the same location, you will get back the code you sent, as if it were a RAM location. This means that if you sent the lower four bits as output and the upper four bits as input, you can send a code out, then read the input combined with the code you sent. This makes scanning the controllers simple to set up in your software. The value you read is what you sent plus 16 times the value that your device sends back.

Here is an example setting up Port B as output:

100 POKE 54019, 56
110 POKE 54017, 255
120 POKE 54019, 60

-011--00	(48)	disables peripheral A interrupts.
-011--01	(49)	enables "peripheral proceed line available" interrupts (IRQ, vectored through VINTER at locations 514 and 515).
-011-00-	(48)	means that PORTA above will write to the direction control register.
-011-10-	(52)	means that PORTA will read and write to the first two controller jacks.
-0110-0-	(48)	turns the cassette motor on (also called peripheral motor).
-0111-0-	(56)	turns the cassette motor off.
1011--0-	(176)	means that a "peripheral proceed line available" interrupt has occurred. You cannot write to this bit, but rather clear it by PEEKing PORTA.

Figure 2. PACTL "Port B Controller" bit chart

The 56 tells the ATARI that the next POKE to 54017 will be in a direction-control code. It is in binary, so the 255 sets up all eight pins for output. The 60 is then sent out.

Noname
54020-54271 D304-D3FF

Here we are at the end of the useful PIA locations, once again faced with lots of *unused* locations, as in all of these.

ANTIC

We found out previously that the G/CTIA chip converts information about the screen into a form that the television set can understand. It gets most of this information from ANTIC, which in turn gets it from you.

ANTIC is like a computer within a computer. It has its own special program, called the display list, which has its own special commands. These commands tell ANTIC such things as how the screen is supposed to look and where to find the data that is to appear on it. But we already know this from our discussion at SDLSTL (560,561). ANTIC also takes care of the Non-Maskable Interrupts (NMIs), fine scrolling and various pointers, all of which will affect the way the screen will appear. Let's take a look.

DMACTL (POKE only)
54272 D400

DMACTL controls DMA (Direct Memory Access). Since there is a wonderful description of it at its shadow register, SDMCTL (559), I won't repeat myself here. There are, however, two more things to add. First of all, DMACTL must be used along with GRACTL (53277) when turning on players and missiles. Secondly, both DMACTL and GRACTL are initialized to 34.

CHACTL (POKE only)
54273 D401

CHACTL makes various changes to the way inverse characters appear and also allows you to make all characters appear upside down (what fun!). See its shadow register CHACT at Location 755 for a complete description.

DLISTL, DLISTH (POKE only)
54274, 54275 D402, D403

DLISTL/H specifies the address of the beginning of the display list. See its shadow register, SDLSTL, at Locations 560 and 561.

HSCROL (POKE only)
54276 D404

Fine scrolling is by far one of the most impressive features the Atari has to offer. We've all been impressed by games that have smoothly scrolling playfields and wondered, no doubt, how we could do it ourselves. HSCROL and VSCROL are the way, but unfortunately, machine language is required to get the kind of effects you've seen. Although fine scrolling can be done from BASIC, it is not nearly as smooth as machine language and is darn near impossible when you're scrolling more than one or two lines. So, although I'll cover the basics here, if you really want to learn to do great fine scrolling from machine language, check out the excellent section on scrolling in *De Re Atari*.

HSCROL allows you to fine scroll horizontally, one color clock at a time (a color clock is the size of a graphics mode 7 pixel). It will affect every mode line that has Bit 4 set in the corresponding display-list instructions (see SDLSTL at Locations 560 and 561). For example:

```
100 GRAPHICS 0
110 DLIST=PEEK(560)+PEEK(561)*256
120 POKE DLIST+7,10
130 LIST
140 FOR COLCLK=0 TO 15
150 POKE 54276,COLCLK
160 FOR DELAY=1 TO 50
170 NEXT DELAY
180 NEXT COLCLK
190 GOTO 140
```

As you can see, there are a few things acting screwy here. First of all, why are the lines below the one being scrolled messed up? ANTIC expects to see 48 bytes per horizontally scrolling line instead of the regular 40. In our example, that causes the lower lines to get shifted over. The solution, and the reason that fine scrolling from BASIC is so difficult, is to give each horizontally scrolling line an LMS instruction (again, see SDLSTL). This brings us to our second problem. Why aren't the characters in our example being scrolled all the way across the screen? HSCROL can only handle values between 0 and 15. If you want to scroll more than 15 color

clocks, what you have to do is set HSCROL back to 0 and change the LMS addresses of the lines you're scrolling. In graphics mode 0, for example, you would subtract 4 from each LMS address. Why 4? Each character in graphics mode 0 is four color clocks wide, so the equivalent of setting HSCROL to 16 would be moving the characters four to the right, which is the same as subtracting 4 from the LMS addresses. As I said before, this can get messy from BASIC.

Let's look at a checklist of what you need to do to have fine horizontal scrolling:

1. LMS addresses for all the lines you are going to scroll.
2. Bit 4 set on all the display-list instructions for the lines you are going to scroll.
3. Screen memory set up properly to ac-

VSCROL (POKE only)
54277 D405

VSCROL is like HSCROL, except it takes care of fine vertical scrolling. Bit 5 in the display list instructions is responsible for turning the scrolling on or off for each line (1 equals on), and the value you POKE into VSCROL is the number of scan lines you want to scroll the line upwards. Try the following:

```
100 GRAPHICS 0
110 DLIST=PEEK(560)+PEEK(5
61)*256
120 POKE DLIST+7,34
130 LIST
140 FOR SCNLIN=0 TO 7
150 POKE 54277,SCNLIN
160 FOR DELAY=1 TO 50
170 NEXT DELAY
180 NEXT SCNLIN
190 GOTO 140
```

you should take care of fine scrolling during VBLANK, which is another reason why machine language is necessary.

One nice thing about vertical scrolling is that you only need to have an LMS instruction on the first line to be scrolled. For example, try the following:

```
100 GRAPHICS 0
110 DLIST=PEEK(560)+PEEK(5
61)*256
120 POKE DLIST+3,98
130 FOR INSTR=6 TO 27
140 POKE DLIST+INSTR,34
150 NEXT INSTR
160 LIST
170 LMSLO=PEEK(DLIST+4)
180 LMSHI=PEEK(DLIST+5)
190 FOR SCNLIN=0 TO 7
200 POKE 54277,SCNLIN
210 FOR DELAY=1 TO 50
220 NEXT DELAY
230 NEXT SCNLIN
240 LMSLO=LMSLO+40
```

*The OS initializes NMIEN to 64,
thereby enabling vertical blank
interrupts. It also sets NMIEN to 64
during the SETVBV routine
mentioned at VVBLKD (548,549).*

So what?

count for the longer lines.

And to do the actual scrolling:

1. Set HSCROL to 0 (15 if you're scrolling from right to left).
2. Add one to HSCROL (subtract 1). Remember that HSCROL is POKE only, so you'll have to keep track of its current value in a separate variable.
3. If HSCROL equals 16 (minus 1), set it to 0 (15) and subtract (add) four to each LMS address. If you're using graphics modes 1 or 2, add or subtract two instead of four, since each character in these modes is twice as wide as in graphics 0.
4. Go to Stop 1.

When you change the LMS addresses, you should also check to make sure that you haven't scrolled too far to the left or right.

There are a few things to notice here. First of all, there's no problem with the lines below the one scrolling. Vertical scrolling does not expect extra bytes per line, so there is no need to worry about that. Secondly, try BREAKing the program and then POKE 54277,0.

Where's line 120? In fine vertical scrolling, the line after the last line to be scrolled acts as a "buffer." The buffer provides data to scroll into the last scrolling line. To see this, get rid of line 190 and RUN the program again. You should always make sure that there is one nonscrolling line to act as the buffer.

Our final thing to notice here is that every now and then the screen "jumps" a little while the program is running. This is because ANTIC does not like you changing VSCROL (or HSCROL) while it's trying to draw the screen. That means that

```
250 IF LMSLO>255 THEN LMSL
0=LMSLO-256:LMSHI=LMSHI+1:
POKE DLIST+5,LMSHI
260 POKE DLIST+4,LMSLO
270 GOTO 190
```

This program makes all the mode lines (except the last) scrollable vertically and then proceeds to scroll them. It does not check how far it's scrolled so far, so it will eventually start showing garbage on the screen. See SAVMSC at Locations 88 and 89 for an explanation of why. Press SYSTEM RESET when you've had enough.

One thing you'll also see when you run this program is the main problem with fine scrolling from BASIC: You can't change the LMS address and VSCROL at exactly the same time, so the whole screen appears to jump down a line every so often. Although there is no way to get rid of this, you can use SDMCTL at Location 559 to

turn off the screen while you change the LMS. This generates a brief "flash" instead of the jump. See for yourself; add the following lines to the preceding program:

```
205 POKE 559,34
255 POKE 559,0
```

This is about the best you can do from BASIC.

In Modes 0 and 1, where the characters are eight scan lines high, VSCROL can vary from 0 to 7. In Mode 2, where the characters are 16 scan lines high, it can vary from 0 to 15.

Noname
54278 D406

This location is not used.

PMBASE (POKE only)
54279 D407

Back in G/CTIA, we were discussing player/missile graphics. We discovered that we could either keep supplying G/CTIA with the player/missile data ourselves or have ANTIC do it for us. If ANTIC is doing it, then PMBASE is used to tell ANTIC where the data is stored. It is the high byte of the address, so the address itself is equal to the value you POKE into PMBASE times 256. Because of some esoteric requirements of ANTIC, PMBASE must be on a 2K boundary if you are using regular-height players, and a 1K boundary if you are using double-height players. How can you tell? If the value you are going to POKE into PMBASE is a multiple of 4, then it's a 1K boundary. It has to be a multiple of 8 to be a 2K boundary.

Noname
54280 D408

Another location that isn't used.

CHBASE (POKE only)
54281 D409

Another location with a shadow register that explains everything. See CHBAS at Location 756 for a description of the character-set address.

WSYNC (POKE only)
54282 D40A

Storing any value in WSYNC will cause the 6502 to stop everything until the end of the current scan line (HBLANK). This is very useful if you want to synchronize something with the screen display. For an example and more information, see VDSLST at Locations 512 and 513. Note that VDSLST is *not* a shadow register for WSYNC.

VCOUNT (PEEK only)
54283 D40B

VCOUNT keeps track of what scan line is currently being drawn. Actually, it increases by 1 every two scan lines, so multiply the value by 2 to get the true number.

If you have more than one DLI, VCOUNT is a good way for your DLI routine to check which one is being processed. It can also be used to stimulate DLIs. For example, you might write a loop that waits for VCOUNT to reach a certain value before going on. This allows you to spend more time in a certain routine than DLIs allow, but it also wastes a lot of time waiting.

There are a total of 262 scan lines on a screen (312 in Europe), so VCOUNT can range from 0 to 130 (155 in Europe).

PENH (PEEK only)
54284 D40C

This tells you the horizontal position of the light pen. See its shadow register, LPENH, at Location 564 for more information.

PENV (PEEK only)
54285 D40D

Same as the preceding, except it's the vertical position, and you should see LPENV at Location 565.

NMIEN (POKE only)
54286 D40E

The last two bits of NMIEN are used to enable or disable the NMIs. They are used as shown in Figure 3.

The OS initializes NMIEN to 64, thereby enabling vertical blank interrupts. It also sets NMIEN to 64 during the SETVBV routine mentioned at VVBLKD (548,549). So what? If you are writing a program where you will be using your own VBLANK routine and display-list interrupts, make sure that you enable the display-list interrupts after you use SETVBV. There have been a couple of times when I couldn't figure out why my DLIs weren't working, only to discover that I had enabled them before I set up my VBLANK routine.

A few of you out there may be thinking *What about SYSTEM RESET? Isn't that an NMI as well?* Yes it is, but the computer does not allow you to disable it. Pressing SYSTEM RESET will always cause a warmstart to occur. You can, however, store an address in DOSINI (12,13), since the OS jumps through DOSINI after it is done with the warmstart. Most machine-language programmers have DOSINI point to their program's initialization routine. That way, the program will start over again if someone presses SYSTEM RESET (normally the OS would go to BASIC or reboot the system).

NMIREs (POKE) and NMIST (PEEK)
54287 D40F

(POKE): POKEing any value here clears NMIST.

(PEEK): The last three bits of NMIST are

--1-----	(32)	enables the SYSTEM RESET interrupt.
-1-----	(64)	enables the vertical blank interrupt.
1-----	(128)	enables the display list interrupt.

Figure 3. NMIEN bit chart

used to identify what kind of interrupt has occurred, as you can see in Figure 4.

Unfortunately, since the OS has already taken care of NMIST and NMIREs by the time you can get to them, they don't really do you much good (you don't have to reset

NMIST during your DLI routine).

Noname
54288-54783 D410-D5FF

These locations, the rest of ANTIC, are currently unused.

Noname
54784-55295 D600-D7FF

So are these.

Floating-Point Package

Locations 55296 through 57343 hold the floating-point package, a series of routines

to wonder. Suppose you had the following line in BASIC:

```
250 X=3.14159*37.5
```

--1-----	(32)	means that the SYSTEM RESET key has been pressed.
-1-----	(64)	means that a vertical blank interrupt has occurred.
1-----	(128)	means that a display list interrupt has occurred.

Figure 4. NMIST bit chart

The Operating System

Finally, way back at the end of memory, we come to the Operating System itself, stored in a 10K ROM cartridge (inside your computer). This OS ROM includes not only the program for the Operating System (yes, the OS is just another pro-

gram), but also the floating-point package, the data for the Atari character set, the device handlers and various vectors. As I've mentioned, there are two versions of the OS as of this writing. Version B includes some changes to get rid of a few of the bugs that appeared in Version A. These changes come mainly in S10 and the interrupt handler routines. The addresses I'll be giving below will be for Version A, since it is the best documented. You should see Appendix 5 on OS changes to determine which locations will not be the same in Version B. How do you know which version you have? If PEEK (58383) equals 0 then you have Version B.

There have been a couple of times when I couldn't figure out why my DLIs weren't working, only to discover that I had enabled them before I set up my VBLANK routine.

gram), but also the floating-point package, the data for the Atari character set, the device handlers and various vectors.

As I've mentioned, there are two versions of the OS as of this writing. Version B includes some changes to get rid of a few of the bugs that appeared in Version A. These changes come mainly in S10 and the interrupt handler routines. The addresses I'll be giving below will be for Version A, since it is the best documented. You should see Appendix 5 on OS changes to determine which locations will not be the same in Version B. How do you know which version you have? If PEEK (58383) equals 0 then you have Version B.

If you need more specific information on the locations and routines described next, I suggest you study the appropriate part of the OS Listing. It is well commented and relatively easy to understand.

All locations in the OS are PEEK only.

gram), but also the floating-point package, the data for the Atari character set, the device handlers and various vectors.

As I've mentioned, there are two versions of the OS as of this writing. Version B includes some changes to get rid of a few of the bugs that appeared in Version A. These changes come mainly in S10 and the interrupt handler routines. The addresses I'll be giving below will be for Version A, since it is the best documented. You should see Appendix 5 on OS changes to determine which locations will not be the same in Version B. How do you know which version you have? If PEEK (58383) equals 0 then you have Version B.

If you need more specific information on the locations and routines described next, I suggest you study the appropriate part of the OS Listing. It is well commented and relatively easy to understand.

AFP
55296 D800

This routine takes an ATASCII representation of a number (e.g., "12345") and converts it to floating point (with carry). INBUFF points to the ATASCII number; floating-point register 0 (FRO) will hold the result.

You may be wondering why such a routine would be needed, and that's a very good thing

When you type in such a line, BASIC sees the numbers as nothing more than a bunch of ATASCII characters. Before it can do the math, it must convert those characters into numbers it can understand. That's what AFP is for. BASIC will use AFP on both numbers

(moving one of them to FRI), do the multiplication and then store the result in X. AFP is also needed for BASIC's STR \$ function.

FASC
55526 D8E6

FASC does just the opposite of AFP. It takes a floating-point number from FRO and stores the ATASCII representation in LBUFF. This is necessary when a number needs to be printed on the screen and also for BASIC's VAL function.

IFP
55722 D9AA

IFP is used to convert integers to floating point. It expects to see the integer in the first two bytes of FRO (Locations 212 and 213) and will store the result in FRO.

FPI
55762 D9D2

This does the exact opposite of IFP (with carry).

ZFRO
55876 DA44

ZFRO sets all the bytes in FRO to 0.

AFI
55878 DA46

Sets FR_x to 0, where x is the value in the X register.

FSUB
55904 DA60

FSUB subtracts FR₁ from FR₀ (with carry) and stores the result in FR₀.

FADD
55910 DA66

Adds FR₁ to FR₀ (with carry) and stores the result in FR₀ (notice that FADD is actually a part of FSUB).

FMUL
56027 DADB

Multiplies FR₀ by FR₁ (with carry) and stores the result in FR₀.

FDIV
56104 DB28

Divides FR₀ by FR₁ (with carry) and stores the result in FR₀.

PLYEVL
56640 DD40

This one is a little complicated, so bear with me. PLYEVL evaluates a polynomial, such as $5Z^4 + 10Z^2 + 2Z + 1$ (read "five Z to the fourth plus ten Z squared plus two Z plus one"). For the sake of this routine, we'll write such a polynomial as:

SUM (I=N to 0)(A[I]Z^I)

So, in the preceding example, N=4, A(0)=1, A(1)=2, A(2)=10, A(3)=0 (since there is no Z cubed) and A(4)=5.

Why are we doing all of this? When you call PLYEVL, it expects you to provide the following information:

Somewhere in memory: a list of the A() values, in floating-point format (BCD), starting with A(0).

X register: low byte of the starting address of the preceding list.

Y register: high byte of the starting address of the preceding list.

Accumulator:N+1

FR₀:Z

PLYEVL will take all of this and use it to evaluate the polynomial (with carry). The result will be stored in FR₀.

FLDOR
56713 DD89

FLDOR will load FR₀ with the floating-point number pointed to by the X and Y registers. X should hold the low byte of the address of this number, Y the high.

FLDOP
56717 DD8D

FLDOP will load FR₀ with the floating-point number pointed to by FLPTR (252).

FLDIR
56728 DD98

FLDIR will load FR₁ with the floating-point number pointed to by the X and Y registers. X should hold the low byte of the address of this number, Y the high.

FLDIP
56732 DD9C

FLDIP will load FR₁ with the floating-point number pointed to by FLPTR (252).

FSTOR
56743 DDA7

FSTOR will store FR₀ in memory, starting at the address pointed to by the X and Y registers. X should hold the low byte of this address, Y the high.

FSTOP
56747 DDAB

FSTOP will store FR₀ in memory, starting at the address pointed to by FLPTR (252).

FMOVE
56758 DDB6

FMOVE moves the floating-point number in FR₀ to FR₁.

EXP
56768 DDCO

EXP raises "e" to the FR₀ power and stores the result in FR₀ (FR₀=e^{FR₀}).

EXP10
56780 DDCC

EXP10, as you may have guessed, raises 10 to the FR₀ power and stores the result in FR₀ (FR₀=10^{FR₀}). Notice that it is actually part of the EXP routine.

LOG
57037 DECD

LOG figures out the natural logarithm (Base E) of FR₀ and stores it back in FR₀.

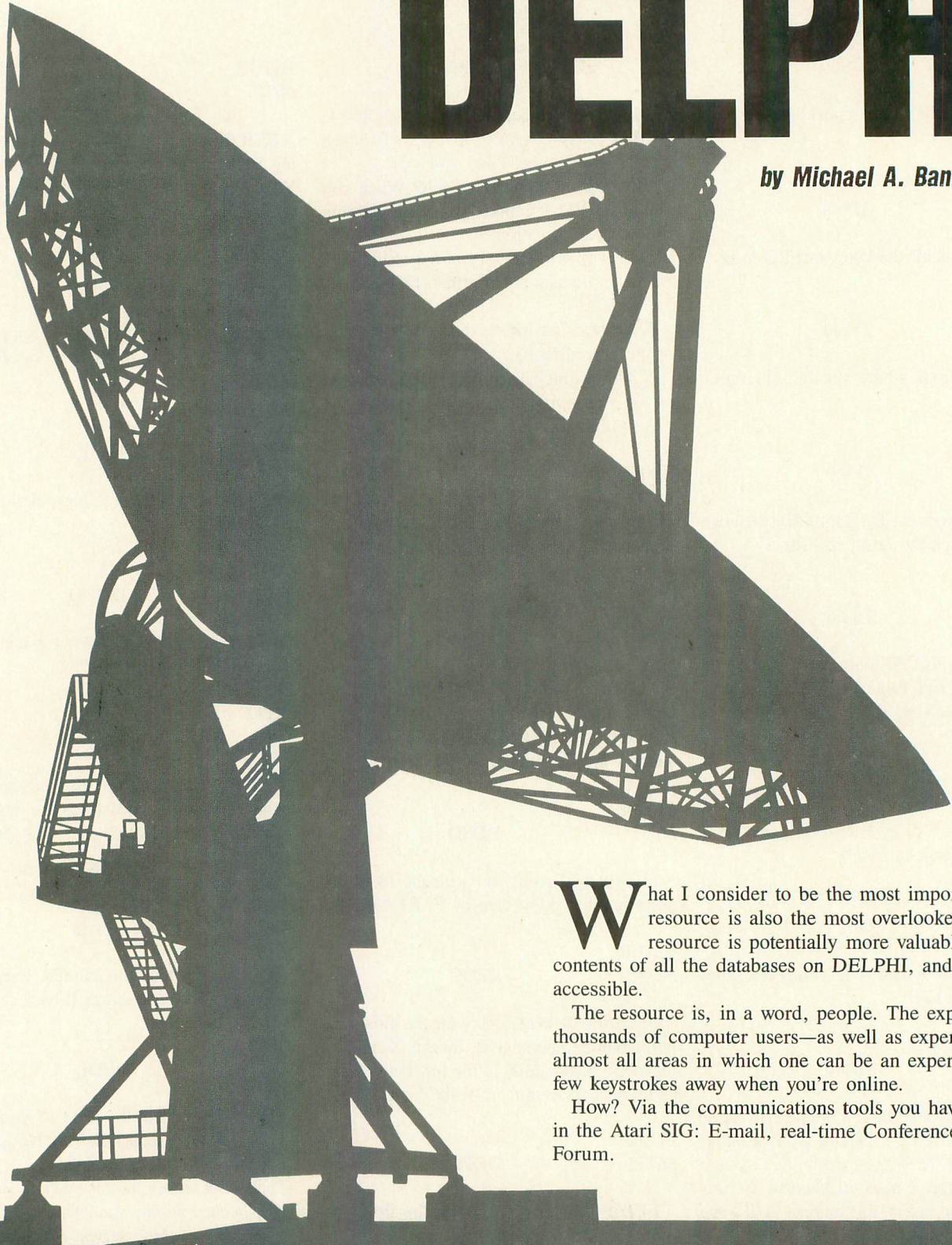
LOG10
57041 DEDI

LOG10 figures out the Base 10 logarithm of FR₀ and stores it back in FR₀. Notice that it is part of the LOG routine.

That about closes out this session. Be sure to stop by next month when we conclude the *Master Memory Map* series. See you then!

Database DELPHI

by Michael A. Banks (KZIN)



What I consider to be the most important online resource is also the most overlooked. This resource is potentially more valuable than the contents of all the databases on DELPHI, and easily as accessible.

The resource is, in a word, people. The expertise of thousands of computer users—as well as experts in almost all areas in which one can be an expert—is but a few keystrokes away when you're online.

How? Via the communications tools you have available in the Atari SIG: E-mail, real-time Conference and the Forum.

I've discussed each of these at one time or another in this column, but the Forum only in brief. So, as promised last month, we'll take a very detailed look at what the Forum has to offer.

If you haven't checked out the Atari SIG Forum yet, you're missing a lot of useful, interesting and just plain fascinating conversation. And if you do use the Forum, are you aware of all the features it has to offer? Either way, read on to see what you've been missing!

"threads." (A thread consists of an original message and all the replies to that message—including replies to the replies.)

Getting There

To get to the Forum, type FORUM at the Atari SIG menu. The Forum banner and prompt will scroll by: See Figure 1

```
THE FORUM
```

The Voice of the Atari User

```
Welcome to the ATARI Forum.  
Forum contains messages 37798 through 38448.  
Highest message you've read is 38425.
```

```
You have 2 new Forum Messages  
Press RETURN to READ WAITING messages
```

```
FORUM>Reply, Add, Read, "?" or Exit> ?
```

Figure 1

What is the Forum?

The Atari SIG Forum is what online old-timers might call a "computer conferencing system." Since this description is easily confused with real-time conferencing (ala Atari SIG's Conference), it's easier to refer to the Forum as a public message system.

A distinctive feature of the Forum is the organization of related messages into

Commands

The most important and most frequently used Forum commands are listed in the prompt. However, the Forum offers a plethora of commands for the sophisticated user, which I'll discuss in detail in a few paragraphs.

Finally, the system tells me that two of the unread messages are addressed to me.

Now let's take a look at what you can do with that information.

The Forum Menu and Commands

The Forum menu lists available options and commands. Type ? to see it: See Figure 2

FORUM Menu:

```
ADD New Message (Thread)  
REPLY To Current Message  
READ Message(s)  
FOLLOW Thread  
BACK to Previous Message  
DIRECTORY of Messages  
MAIL  
TAG Interesting Message  
FILE Message into Workspace
```

```
FORWARD Message by Mail  
DELETE Message  
EDIT a Posted Message  
NEXT Message  
TOPICS (Set/Show)  
HIGH Message (Set/Show)  
HELP  
EXIT
```

```
FORUM>Reply, Add, Read, "?" or Exit>
```

Figure 2

(If you're using a 40-column display, the menu will be displayed in one column rather than two.)

Forum commands can be categorized by the kinds of functions they perform:

★ *Entering and editing messages (this includes replying to a message).*

★ *Deleting messages.*

★ *Finding messages (using the Forum Directory to search for messages on single*

The Forum's sophisticated directory system can be used to locate messages by several criteria including date, subject, addressee or message number.

or combined criteria).

★ *Reading messages (by number, by search criteria, or by thread; and creating special groups of messages for later reference).*

★ *Filing messages (copying messages to files in your Workspace).*

★ *Accessing Mail (sending copies of messages to others by E-mail, and replying to messages by E-mail).*

★ *Getting help.*

Entering and Editing Messages

There are two ways you can add a new message to the Forum: you can add an original message, or you can reply to an existing message.

Adding a Message

ADD initiates a new message and begins a new message thread. When you type ADD, you are prompted for an addressee for the message (type ALL if the message is to everyone), a topic for the message (one of the existing topics, which match the SIG's database topics), and the message's subject (whatever you want it to be). After you enter this header information, simply type in your message, and enter Control-Z when finished; the message will be automatically posted. If

you address the message to a specific SIG member, he or she will be notified when he enters the SIG (and again on entering the Forum) that a Forum message is waiting.

(Note: The command FORUM can be used in place of ADD.)

Replying to a Message

To reply to a message, type REPLY at the FORUM prompt after reading the message.

You will be prompted to enter the addressee of the message; press Return, and the message will be addressed to the person to whom you are replying, and the Topic and Subject headers will be filled in automatically. (Otherwise, you can type the name of a different addressee; the message will be retained in the Forum.)

After you type your message, press Control-Z to post it.

Note that when you reply to a message, your message becomes a part of the message thread of which the message to which you replied was a part. (Or, if the message to which you replied was a one-shot, a thread is created.) Thus, the commands that move through a thread (FOLLOW and BACK) will "find" your new message as well as others in the thread. If, however, you create a new message with the same subject using ADD, it will not be a part of the thread. This is important if you want to make sure your comments are read by anyone reading that particular thread.

Editing Messages

You can edit messages you have posted by typing EDIT followed by the number of the message, or by reading the message and then typing EDIT at the FORUM prompt.

When you type EDIT, the message header

is displayed, along with this prompt line: EDIT Text, Subject, Topic, "?" or Exit

To see the Forum Edit menu, type ? at this prompt. The menu looks like this:

FORUM EDIT Menu:

```
TEXT of Current Message
SUBJECT of Current Message
TOPIC of Current Message
SHOW Message Header
HELP
EXIT
```

As you can see, you can edit the text, subject or topic of a message. Commands available here also redisplay the message header and get help.

When you select SUBJECT or TOPIC to edit, all you have to do is type the new subject or topic and press Return. If you change your mind, enter Control-C or Control-Z to cancel the change.

When you select TEXT to edit, you will be using your default editor, OLDIE or EDT. To get help, type HELP or /HELP, depending on which editor you're using. To save any changes, type EXIT or /EXIT. To cancel any changes type QUIT or /QUIT.

Editing a message will not remove it from the thread.

Deleting Messages

The only Forum messages you can delete are those you've posted, and those addressed to you.

To delete a message, type DEL followed by the message number, or read the message and type DEL at the FORUM prompt.

Finding Messages in the Forum Directory

The Forum's sophisticated directory system can be used to locate messages by several criteria including date, subject, addressee or message number. You can also locate messages posted by a specific DELPHI member. These criteria can be used individually, or combined in any fashion you wish, in the form of qualifiers used with the DIRECTORY (DIR) command.

Here's an example of how this works. If you were interested only in reading Forum mes-

sages on the subject of photos, you would enter the Forum and type *DIR SUBJ Photo*. The system would display a list of all messages on that subject. If you wanted to get more precise, you could type *DIR SUBJ Photo FROM KZIN* to see a directory of all messages from membername KZIN on the subject of photos.

The Directory is especially handy if you haven't been in the Forum for a while and several hundred messages have piled up dur-

Again, you can combine criteria in various ways to specify just the messages you want to see. (Note that when you use SUBJECT with DIR, the system looks for the specified character string anywhere in the subject header. Thus, if you type *DIR SUBJ Test*, you will see a directory of all messages that have the string "test" anywhere in the subject header. This could include such headers as "Testing Disks," "One Protester" or "Disk Test.") An extreme example might be:

sages when you enter the Forum area. DELPHI keeps track of whether you've read waiting messages and displays them first if you press Return when you first enter Forum. If you read other messages before reading those addressed to you (by number or by qualifier), you'll have to type READ WAITING to display unread messages addressed to you.

The TAG command allows you to create a small, temporary subset of Forum messages that you can manipulate later.

ing your absence. You can browse the directory for messages on a certain topic or exclude messages lower than a certain number by typing *DIR xxxxx* (which would show all messages from that number on).

The full range of options available for use with DIR is:

NEW: messages above your HIGH counter.
CURRENT: the message you just read.
TO: messages addressed to.
FROM: messages posted by.
WAITING: messages waiting for you.
NONSTOP: nonstop mode.
NS: short for NONSTOP.
PAUSE: pause on message for you.
nnn:mmm: range of messages by number.
+n or -n: skip forward or backward.
TAG: messages you previously tagged.
THREAD: messages related by REPLY.
INITIAL: only the first message of a thread.
FOLLOW: automatic FOLLOW thread.
FT: short for FOLLOW.
SUBJECT: messages by subject.
WILDCARD: wildcard subject (* and % are special).
TOPIC: restricted to one topic.
ALL: for DIR NEW ALL, show messages you have read.

DIR SUBJ Atari FROM KZIN TO ANALOG2 35000.

This would display all messages with the word "Atari" in the subject header, addressed by KZIN to ANALOG2, with numbers higher than 35000.

Reading Messages

READ is the operative command in reading messages, but unless you intend to add a specification to the command, pressing Return will have the same effect as typing READ alone.

You can use the same specifications with READ as with DIR. You can also read messages by typing the number of a message (handy after you've used DIR). If you're not looking for specific messages, simply press Return to read the next unread (NEW) message.

(Note: If you read a message by typing its number, pressing Return will display the next message in numerical sequence, rather than the next unread message. To "reset" the system to display only new messages, you must type READ NEW.)

As mentioned earlier, if Forum messages addressed to you are waiting, you are notified automatically when you enter the Atari SIG; you are notified again of waiting mes-

Following Threads

You can follow a message thread in forward or reverse order with the special commands FOLLOW and BACK.

FOLLOW displays all messages that are replies to the current message and all messages that are replies to any of the replies.

BACK does the same thing, but in reverse numerical order. This allows you to trace a thread back to its original message.

Reading messages with FOLLOW or BACK will display messages you've previously read, as well as unread messages. When you reach the end or beginning of a thread, the system notifies you that there are no more messages in the thread.

Tagging Messages

The TAG command allows you to create a small, temporary subset of Forum messages that you can manipulate later with DIR or READ. This is useful if you see a message or messages that you want to refer to after you read other new Forum messages or which you'd like to download for future reference.

To tag a message, simply type TAG after you read it. Later, you can type DIR TAG or READ TAG to see each message you've tagged.

The group of tagged messages will cease to exist when you leave the Forum. (If you have a tagged message or messages waiting, DELPHI notifies you when you attempt to exit the Forum, at which time you can read the tagged message(s) or not, as you prefer.)

Accessing Mail From the Forum

While the Forum is a public messaging system, it isn't entirely public. There are commands that allow you to reply to a member who posted a particular message by private E-mail and to send copies of messages to

continued on page 66

GAME DESIGN Workshop

by Craig Patchett

Game Logic

Let's start off this section in a terribly cliché way. The dictionary defines *logic* as being a "way of reasoning." This in turn can be translated to a "way of analyzing," and that's exactly what this section is about. An arcade game does a lot more than look pretty. It has to analyze what's going on and come up with an appropriate response. But there are a lot of things that have to be analyzed. For example, *what* should be moved *where*? Have we reached the end of the game? The end of the level? Should the game become more difficult now? And so on. In a general sense, game logic refers to all decisions that the computer must make during the course of the game. We'll now take a look at the types of decisions and how to deal with them.

Some Examples

Let's start off by looking at some sample games and seeing what kind of logic is used in them. How about *Pong*? Is there any logic used in this game? Of course, although it's somewhat simple. First of all, the computer must decide where to move the player's pad-

dles. Even though this is just a matter of checking the game controllers and moving the paddles accordingly, the computer has to check to make sure that the player isn't trying to move the paddles off the screen.

Next, the computer moves the ball and checks to see if it has hit a paddle. If it has, then the computer has to figure out what angle to bounce it off. If it hasn't, then the computer has to check to see if the ball is off the screen, in which case one of the players has scored.

Finally, if one of the players has scored, is it the end of the game? If not, the computer has to serve the next ball. Believe it or not, this is just a simplified version of what's going on. Pretty complicated for what you thought was just a simple game, right? Let's look at something a little (lot) more complicated.

Pac-Man seems at first to be a relatively simple game. All it consists of is a maze, some dots, four ghosts and the player. For now, let's forget about the ghosts and look at what logic is involved in simply moving the player around the maze. First of all, the computer has to look at the joystick and see what

direction to move the player. Then it has to see if that's a valid direction. After all, you're not allowed to move through walls. Assuming you moved in a valid direction, the computer then has to check to see if you ran over a dot, in which case it gives you points and erases the dot. It then checks to see whether it was a power pill, in which case you get stronger for a while. Finally, it checks to see whether or not that was the last dot. If it was, then it gets ready for the next level. The next level, in case you hadn't noticed, is more difficult. You move faster. In any level you move slower while eating dots. Okay, that's what it takes just to move around the maze. Now let's bring in the ghosts.

The ghosts make things considerably more complicated, because they are reasonably "intelligent." Although at first they just appear to be wandering around the maze, eventually some of them start to follow you. Stop and think about this for a minute. How would you go about programming the ghost to follow the player? Do you remember the route that the player is taking and instruct the ghost to follow it? This might work, but it's too easy for the player to avoid the ghosts since he will

just be leading them around. Play the game and you'll see that the ghosts try to cut the player off. How can you make them that intelligent? Although I'm not sure exactly how it's done in the original game, I'll explain one way that it could be done.

Artificial Intelligence

What we're about to see is a very rudimentary example of something you've probably heard a lot about lately: artificial intelligence. Even though you probably associate artificial intelligence with robots, it applies to any kind of computer program that appears to have a mind of its own. In our case, the ghosts appear to be intelligent, and the computer program that controls them is thereby an example of artificial intelligence. No big deal, really. In any case, the question still remains as to how we would program *Pac-Man* to give the ghosts this "intelligence."

Let's start by considering our goal. We want a ghost to intercept the player as quickly as possible, which means it has to take the shortest route between where it is now and where the player is. So far, so good. Let's make it so that the only place a ghost can change direction is at an intersection. The *Pac-Man* playfield has a total of 34 intersections (corners don't count).

Now we set up an array that tells us in what direction we must turn at each intersection to get to any of the other intersections as quickly as possible. This makes an array that is 34 by 34 (34 choices for each of the 34 intersections), for a total of 1,156 bytes. Actually, since there are at most four possible directions at each intersection, we could cram four choices into each byte and reduce the number of bytes down to around 300, if we wanted to conserve memory. In any case, we now have everything we need. When a ghost that's following the player gets to an intersection, we figure what intersection the player will get to next and look up the direction to take to get to that intersection. We do this at each intersection the ghost reaches until it succeeds in capturing the player. Ta-da!

What you've just seen is an example of the use of tables (or arrays). This is probably one of the easiest and most common ways of controlling game logic. It does, however, require

that you have a finite number of possible choices and that this number is small enough for a table to fit in memory. How might you approach a game that does not comply with these restrictions? To begin with, there aren't too many games that don't. Most games fit into one of two categories. Either they consist of a predictable routine repeated over and over again (like *Space Invaders*, *Centipede*, etc.), or they are limited in their choices (like *Pac-Man*, *Donkey Kong*, *Popeye*, etc.).

Breaking Away From the Ordinary

Some games combine the two categories (*Missile Command*), but very few break away from the mold. But what if you wanted to? How would you go about programming intelligence into a game in which there was an infinite number of choices? About the only way is to make these choices predictable so that you can use an algorithm to come up with the correct choice. For example, you might give the algorithm the current player position, the current computer position and a few other factors, and the algorithm would figure out what to do based on this information and then go ahead and do it. The only problem here, as I stated before, is that whatever you're doing must be predictable. I know I'm being awfully vague about all of this, but there are so many possibilities that there isn't just one answer that I can give you. You have to evaluate your own game idea and try and come up with something that will work for you.

General Logic-Learning Curves

Let's assume right now that you've got all the basic logic worked out and now have a playable game in front of you. Are you done as far as the logic is concerned? No. Even though the logic for each of the game elements is in place, you still have to work out the general logic, or how the game comes across as a whole. There are two things that have to be considered as far as this step is concerned. First of all, there is something fancy called the "learning curve." The learning curve tells how easy or hard it is to learn the game and become an expert. Figure 1

The ghosts make things considerably more complicated, because they are reasonably "intelligent."

shows a few possible learning curves.

Let's take a look at them in order. Curve 1 is probably the ideal. It represents a game in which your score continues to increase the more you play. Unfortunately, there aren't too many of these games around.

Curve 2 represents a more difficult game, in which it takes a long time to get the hang of things, after which your skill improves rapidly. This is not a good curve to have, since it tends to frustrate the beginner and make him not want to play again.

Curve 3 is the exact opposite. This is a game that is easy to learn and presents very little challenge once you do. Great for beginners, but a game has to present a challenge for it to be appealing.

Finally, we have Curve 4. This is a game which has a trick to it. Once you learn that trick, your score improves dramatically and there may or may not be any more challenge. The problem with this kind of game is that players who haven't discovered the trick get frustrated when they see people getting scores they thought were impossible. Often, however, such a trick works its way into a game without the designer knowing it. *Asteroids* is an example of this, as is *Pac-Man*. The designers of these games probably never guessed that players would find a safe hiding place or discover a pattern. But they did.

These four learning curves cover most, if not all, of the games currently out on the market. The question is, how do you design a game to follow Curve 1, which we decided was the best? Ideally, what you want to do is keep up with the player's ability. If he gets a little bit better, then make the game a little bit harder. But don't make it too hard too quickly or not hard enough soon enough. Sound difficult to do? It is, and about the only way to even come close is to let a bunch of your friends play it and watch how they do. If they think it's too easy, make it a little harder, and vice versa. They'll let you know when they think it's great. Why can't you just play it yourself? Because you'll always think it's too easy, since you know all its secrets.

Wait a minute. How do you make a game harder? Well, there are a lot of ways to do it, but here are a few suggestions:

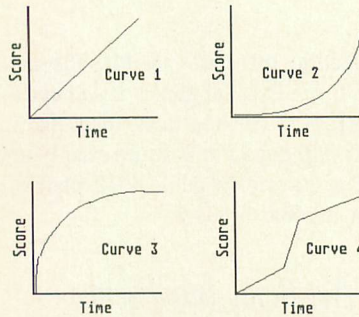


FIGURE 1: Learning Curves

1. Make it faster.
2. Make the opponents a little more intelligent.
3. Put more opponents on the screen.
4. Make it more complicated.

In other words, give the computer more of an advantage over the player. To see these ideas at work, play some of your favorite games. See what happens as they progress. Look at how the different levels vary from each other. Analyze.

Game Appeal

The second thing that we have to consider in looking at the overall logic of a game is its appeal. Of course, if we knew how to design a game that was appealing, every game to hit the arcades would be a smash hit. Obviously it isn't that easy. There are, however, a few things that every game must have if it is to be successful. First of all, the game must be something that a person wants to play. It must have something that the player wants and can't get anywhere else. In other words, something that will attract the player to the game. Next, it must create the illusion of being winnable. Everybody likes to win, and they're certainly not going to pump money into a game that will make them feel like a loser. So, even though the player will not be able to beat the game, he must feel as though he accomplished something.

One way of doing this is to reward him every now and then. *Pac-Man* does this with its "intermissions," during which little cartoon shows take place. Also, when a player does fail in the game, and he always will, he must see that failure as being his own fault, not the game's, and as being a result of something he can correct. Otherwise, there is no motivation for him to play the game again.

While we're on the topic of motivation, it is important to maintain motivation for both the beginning player and the advanced. Just because a player gets good doesn't mean he will keep playing. Actually, it's a good reason for him to stop, if he feels that he's mastered the game and there is nothing left for him to reach for. Make sure that there is as much for the expert as there is for the beginner.

That about covers logic. You're probably disappointed that I didn't give you a clear-cut method of designing a winning game. If I, or anybody else for that matter, knew of such a method, I'd be writing games, not articles! The purpose of this section is to get you thinking and give you the basics that are necessary in designing a game. Take what's written here, add a healthy amount of your own ideas and start designing. Even though the result may not be a million-seller, if you can come up with something that even one other person likes, you've done well.

A Change in Direction (and Color)

Our first step in developing the game logic for BASIC invaders has to do with changing the direction of the invaders. What we want to do is have the invaders change direction when they reach each side of the screen. In other words, they should change direction when the right-most invader reaches the right edge of the screen, and when the left-most invader reaches the left edge. Up until last month, the left-most and right-most invaders haven't changed during the course of the game, so we've been able to change direction based on how far we've scrolled the invaders (see lines 1270 and 1280 of our program so far). Now that we can shoot the invaders down, however, we have to do something a little more elaborate.

What Happens When the Invaders Get Shot?

Let's pretend we're the program and take a look at exactly what must happen now when an invader gets shot. First of all, we have to

check to see whether or not the invader was in the left-most or right-most column and, if it was, we then have to check whether or not it was the last invader in that column. Let's suppose that it was the last invader, which would mean that we have to find the new left-most or right-most column. Can we just assume it's the column adjacent to the one that was shot out? No, because that adjacent column may have been shot out already as well. What we have to do is somehow find the closest column that still has at least one invader in it. Once we do, we make it our new left-most or right-most column and adjust our scrolling limits accordingly. And that's all there is to it. The only problem is, how do we translate this into something the computer can understand?

Arrays and Matrices

As I mentioned in the previous section, game logic often involves a lot of arrays and matrices. The most obvious solution to this particular problem would be to use an array that keeps track of how many invaders are left in each column. We would also have a pair of variables that kept track of which columns were the current left-most and right-most, say LEFTCOL and RIGHTCOL. When an invader was shot, we would subtract 1 from the appropriate array element and then check to see if that element was now equal to 0. If it was, then that would mean that we just shot the last invader in that particular column, so we would check the column number to see if it was equal to LEFTCOL or RIGHTCOL. If it was, then we would simply check our array to see where the nearest column was that didn't have an array value of 0, and set LEFTCOL and RIGHTCOL to this new value. A simple change to the variable that kept track of the scrolling limits would follow, and we'd be done.

What's the problem with the above technique? Nothing except that it tends to slow things down because of the calculations that have to be done every time an invader is shot. If we were programming in machine language, this time period would be negligible, and the same holds true if we were going to

compile our BASIC program. Of course, we've already decided that there's no way we'll be able to do a fast game in pure BASIC anyway, so does it really matter? Well, I started off by programming the direction-change logic exactly as I just described, and it really wasn't that bad as far as the speed was concerned. As I was polishing it up, however, a thought suddenly occurred to me. What I was trying to do was have the invaders change direction whenever they hit either side of the screen, right?

Another PMG Trick

Well, another word for "hit" is "collided," and that automatically made me think of PMG. I'd only used three of the players so far, which left one unused. Why couldn't I just make that player a solid vertical strip and position it at the side of the screen that the invaders were heading towards? That way, all I had to do was check for a collision between the player and the playfield, without worrying about which invaders were where. And, by making the player the same color as the background, it wouldn't be visible on the screen. (Actually, since I just needed a narrow strip, the missile would work just as well as the player, and it ended up being what I used.)

This seemed like a great idea, so I tried it and it worked, and this is what we'll be adding to our program in just a minute. The reason I even bothered to explain the array technique is because often you won't have a chance to do something sneaky like this, depending on exactly how your program is set up. On the other hand, you should also look for alternatives instead of going immediately for the obvious. Often you'll find that there are several ways of doing something, and the obvious way isn't always the best. In any case, here are the changes to make to put my solution to work:

```
1270 IF PEEK(53251)=0 THEN
1330
1280 CHANGE=-CHANGE:POKE 17
91,129-PEEK(1791):POKE 5325
5,127+77*CHANGE
```

```
3310 FOR BYTE=39 TO 205:POKE
PA+768+BYTE,192:NEXT BYTE
5440 POKE 704,15:POKE 705,4
0:POKE 706,40:POKE 707,15
5450 POKE 53248,128:POKE 53
249,0:POKE 53250,5:POKE 532
55,204
```

1270: Instead of checking to see how far we've scrolled, we're checking M3PF to see whether or not the invaders collided with Missile 3.

1280: This line remains the same except that we reposition Missile 3 to the opposite side of the screen in preparation for the next collision. I should probably point out at this point the reason for using Missile 3 instead of Missiles 1 or 2, neither of which have been used as yet. Remember that we need whichever missile we use to be the same color as the background, in order for it to remain invisible. Also remember that the missiles take on the same color as the corresponding players and that Players 1 and 2 already make up the alien saucer and therefore have a color different than that of the background. As a result, we are left with Missile 3 as our only choice.

3310: Here we simply initialize the missile so that it is a solid vertical strip, extending from just below the alien saucer to just above the lowest point the invaders can go.

5440: For now we're going to set the missile color to a bright white, just so that you can get a better idea of exactly what's going on. We'll change it to the background color later.

5450: We've changed this line so that the missile is initially positioned on the right-hand side of the screen.

Once you've made these changes and got-

ten the program running, try shooting out the left-most or right-most columns and see for yourself that everything is working as it should. Do you notice any problems? Probably not, but there is one hidden problem that I should bring up. Because of the timing problems with collision detection from BASIC, it is possible for the computer to detect the same collision with Missile 3 twice, thereby changing direction twice in a row, something which we don't want. In order to get around this problem, which showed up once or twice while I was writing BASIC invaders, we're going to make sure that the program waits awhile after changing direction before attempting to change it again. Here's the code to do this:

```
1260 IF PEEK(20)<30 AND PEEK(19)=0 THEN 1330
1280 CHANGE=-CHANGE:POKE 17
91,129-PEEK(1791):POKE 19,0
:POKE 20,0:POKE 53255,127+7
7*CHANGE
5430 POKE 53278,0:POKE 19,0
:POKE 20,0
```

1260: We now make sure that at least half a second has passed since we made the last direction change.

1280: When we make the direction change, we now reset the system timer as well (Locations 19 and 20.)

Now we're all set and ready to move on. Before we do, let's set the color of Missile 3 to that of the background. While we're at it, let's also change the playfield colors so that they're more aesthetically pleasing. The following changes should do the trick:

```
4010 SETCOLOR 0,4,14:SETCOL
OR 1,4,6:SETCOLOR 2,15,14:5
ETCOLOR 3,4,10:SETCOLOR 4,7
,0
5440 POKE 704,15:POKE 705,4
0:POKE 706,40:POKE 707,112
```

Onwards and Downwards

If you've played around with the game so far, then you've probably noticed some problems. First of all, things don't quite work

right when the invaders reach the barriers. Second, nothing happens when all the invaders are shot down. We're going to address both these problems, and it will turn out that the two solutions are somewhat dependent on each other.

Moving the Invaders Down

Let's start by moving the invaders down. There are two options available to us here. We can coarse-scroll the screen, which is probably the most obvious solution, or we can simply change the way we move the string data onto the screen.

What if we were to coarse-scroll the screen? SCROLL allows us to do this quite easily, but let's look at what happens when we do. Each time we coarse-scroll the screen down by one line, we subtract 24 from the graphics mode 1 LMS address. By the time the invaders get to the bottom of the screen, we will have subtracted at least 240 from this address. This means that screen memory will begin at least 240 bytes before its original position, which in turn means that we'll need another 240 or more bytes for screen memory. Keeping this figure in mind, let's now take a look at our other option and see whether or not it's a better alternative.

Just as we can move the invaders down a line by subtracting 24 from the LMS address, we can also move them down a line by starting them on the screen 24 characters past their old position. This doesn't take up any extra memory and is quite simple to do in our case, so we're going to go ahead and do it. Just make the following changes to our program:

```
1000 X=USR(ADR(MOUMEM$),ADR
(INV$)+5B, MEM1+LINE*24,287)
1280 CHANGE=-CHANGE:POKE 17
91,129-PEEK(1791):POKE 19,0
:POKE 20,0:POKE 53255,127+7
7*CHANGE:LINE=LINE+1
1320 GOTO 1000
5280 LINE=0
```

Not as much work as you thought it would be, was it? Here's the explanation:

1000: We've changed this so that the invaders are now moved onto the

screen starting at MEM1+LINE 24 instead of just MEM1. LINE is a variable that keeps track of how many lines the invaders have scrolled down.

1280: Now when we reach the edge of the screen, we want to increase LINE as well as change direction.

1320: We're also going to consider our move down as the equivalent of a move across, so we skip the scrolling section after we've moved down.

5280: Finally, we want LINE to be 0 initially.

Run the program with these changes, and you'll see that the invaders now move down. Unfortunately, you'll also see that they leave a trail of "dead" invaders behind them. Why? Well, when we move down and start drawing the invaders a line below where we were drawing them before, we don't get rid of the old first line, and that's what you're seeing on the screen.

So how do we get rid of it? We could add a line that moves a string of spaces into that old line, thereby erasing it. There's nothing wrong with this, except that there's another solution that I tend to favor. What if we were to shift the data in INV\$ so that the lines alternated between blank and invaders instead of invaders and blank? This means that the first line in INV\$ would be blank and would automatically erase the old invaders. Of course, it would also mean that the invaders would begin one line lower on the screen, but we can easily make up for that.

Why do I favor this solution over the other? I guess it's because it means not having to add another line to the program; it's just a matter of making slight changes to existing lines. This is, of course, just my own personal preference, so please don't feel that the other solution is wrong. Anyway, here are the necessary changes for my solution:

```
330 INV$(R+C+27,R+C+28)="█
":INV$(R+C+315,R+C+316)="█
```

```

":EF=2
1030 INV$(R+C+27,R+C+28)="♥
♥":INV$(R+C+315,R+C+316)="♥
♥"
1320 GOTO 1000
5280 LINE=-1
5390 FOR LP=0 TO 4 STEP 2:R
EAD DAT$:INV$(LP*48+27,LP*4
8+42)=DAT$:INV$(LP*48+363,L
P*48+378)=DAT$
5400 READ DAT$:INV$(LP*48+7
5,LP*48+90)=DAT$:INV$(LP*48
+315,LP*48+330)=DAT$:NEXT L
P

```

330-1030: We've just added 24 to all the numbers here to shift everything down a line.

5280: By initializing LINE to -1 instead of 0, we make sure that the invaders start on the same row on the screen. You should be aware that we will now be moving INV\$ into screen memory starting at MEM1-24 instead of MEM1. You'll remember that the screen memory area before MEM1 is used for the score, so won't this change affect the score? No, because when we reserved screen memory for the score, we reserved more than was necessary, which means that the score data ends before MEM1-24 (I cheated a little in doing this, because I knew we'd be running into this problem!)

5390-5400: Again, we've just added 24 to the numbers here.

It's Still a Mess

Okay, so now we have the invaders moving down without the mess. We still have problems however. For example, try shooting the invaders after they've moved down a line or two, and you'll notice that the explosions are in the wrong row. The reason for this is as simple as the solution; we're not taking into account the downward shift when we calculate the row that the hit invader is in. So here's our solution.

```

320 R=48*INT((PEEK(1696)-38
-8*LINE)/16):C=2*INT((PEEK(
1692)-SCROLL-COARSE*8-47)/1
6):R=R*(R)=0)

```

All we've done here is use LINE to make the necessary adjustment to R.

Erasing the Barriers

Our next problem occurs when the invaders reach the barriers. At the moment they get lost for a while and then eventually turn the barriers into a mess (let the program run for a while and you'll see what I mean). What we'd like to happen is for the barriers to be gradually erased by the invaders as the invaders move down into them.

How do we go about doing this? Well, the first thing we need to do is have some way of knowing when the invaders have reached the barriers. To be more specific, we need to know when the bottom row of invaders has reached the tops of the barriers. And, since the bottom row can change as the player shoots down the invaders, we're going to have to have some way of keeping track of the bottom row.

So let's begin with a system to keep track of which row is at the bottom. We'll use the array method that I described in the "Change in Direction" section. Our array will keep track of how many invaders are left in each row, and we'll also have a variable that keeps track of the current bottom row. When the last invader is shot in the bottom row, we'll search the array to find the nearest row that still has invaders in it, and it will become our new bottom row. As an added bonus, if we can't find a new bottom row, we'll know that all the invaders have been destroyed and that the current level is complete. Keeping all this in mind, let's take a look at the program changes that we need:

```

1040 TMP=R/48:ROW(TMP)=ROW(
TMP)-1:IF ROW(TMP)<0 OR TM
P<>BOTROW THEN 1080
1050 FOR LP=BOTROW TO 0 STE
P -1:IF ROW(LP)=0 THEN NEXT
LP:GOTO 2000
1060 POP:BOTROW=LP
2000 STOP
3030 DIM MLANG$(90),INV$(57
8),DAT$(16),ROW(5)
5270 SCROLL=0:CHANGE=1:SB=0
:EF=0:BOTROW=5:COARSE=0:SAU
CER=0
5420 FOR R=0 TO 5:ROW(R)=8:
NEXT R

```

In case you're having trouble relating this to the initial description, here's the line-by-line explanation:

1040: Remember that R is the row of the invader that was hit times 48. We divide it by 48 to get back the row number (from 0 to 5) and then subtract 1 from the array element for that row. If the array element isn't equal to 0, which means that there are still invaders left in the row, or if it wasn't the bottom row, then we skip over the next two lines.

1050: There are no more invaders in the bottom row, so we have to find the new bottom row. What we do is go backwards through the array and check each element to see if it's equal to 0. If it's not, then we skip to the next line. If it is, then we keep checking. If we get all the way through the array without finding a nonzero element, then that means there are no more invaders left on the screen, so we skip ahead to line 2000.

SAVE MONEY ON ATARI 800/XL/XE SOFTWARE

- * Atari Public Domain & Shareware Software
- * Over 250 Theme Disks! Every disk is Guaranteed!
- * Games! Graphics! Educational! Music! Utilities! Home & Business!
- * Fast dependable world-wide service!

Send for your FREE descriptive Catalog.

BELLCOM
P.O.Box 1043-G
Peterborough, Ontario
Canada K9J 7A5

CIRCLE #109 ON READER SERVICE CARD.

1060: To get to this line, we must have found a row with invaders still in it and left the loop. Whenever you leave a loop in the middle, like we did here, you should always use the POP command so that the computer doesn't think it's still in the middle of a loop. We also set BOTROW to its new value.

2000: We get to this line when all the invaders have been shot. Eventually it will be the beginning of a routine to start a new level, but for now we'll just stop the program.

3030: Here we've just added our array to be DIMensioned.

5270: BOTROW is initialized to 5.

5420: We also want to initialize ROW() so that we know each row begins with eight invaders in it.

Erasing the Barriers Properly

Before we go ahead and change the program so that the barriers will erase properly, let's make a few more adjustments to what we have so far. When the bottom row is destroyed, there's no point in drawing it on the screen anymore. As a matter of fact, drawing it may even cause some problems as the invaders move into the barriers, since the invader data may eventually get into the barrier screen memory. Luckily, all it takes is a simple change to line 1000 to adjust for this:

```
1000 X=USR(ADR(MOVMEM$),ADR
(INV$)+5B, MEM1+LINE*24, 287-
48*(5-BOTROW))
```

Now, each time BOTROW changes, less of INV\$ will be put on the screen. Unfortunately (there are a lot of *unfortunatelines* in the world of game design), we've just created another problem in that the last invader explosion in the bottom row is not erased from the screen. The following line takes care of this for us:

```
1070 X=USR(ADR(MOVMEM$),ADR
("#####"), MEM1+LINE*24+TMP*48+24, 23
)
```

Now, finally, we're ready to take care of the barriers.

Taking Care of the Barriers Once and for All

If you remember from the column on display lists, we set up the display list so that the barrier area was divided into three LMS sections. That way, we could remove one section without messing up the rest. Well, that's all we're going to do now. When the invaders reach the tops of the barriers, we're simply going to replace one section of the barriers with a scrolling-graphics mode-1 line. Since a scrolling-graphics mode-1 line replaces eight graphics mode-7.5 lines, we'll also have to move the rest of the display list up a little. Figure 2 shows what I mean.

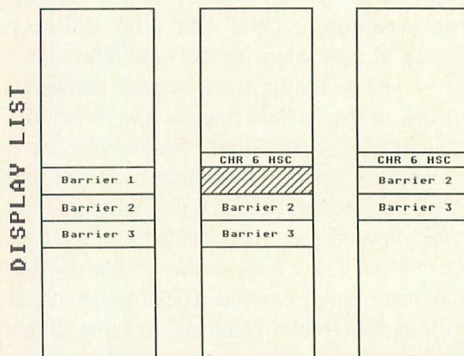


FIGURE 2: Removing the Barriers

And here's the code to do it:

```
1290 IF CHANGE=1 THEN 1400
1320 GOTO 1400
1400 POKE 53278,0
1430 IF LINE+BOTROW*2<>15+B
ARLIM OR BARLIM=3 THEN 1000
1435 POKE 559,0:POKE 20,0
1440 IF PEEK(20)=0 THEN 144
0
1450 POKE DLIST+21+BARLIM,2
2:X=USR(ADR(MOVMEM$),DLIST+
31+BARLIM,ADR(DL$),29-10*BA
RLIM)
1460 X=USR(ADR(MOVMEM$),ADR
```

```
(DL$),DLIST+22+BARLIM,29-10
*BARLIM)
1470 POKE 559,62:BARLIM=BAR
LIM+1:GOTO 1000
3030 DIM MLANG$(90),INV$(57
8),DAT$(16),ROW$(5),DL$(30)
5150 BARLIM=0
```

1290-1320: After we do everything that's necessary for a direction change, we now want to skip over to line 1400, for the routine that will erase the barriers if it's necessary.

1400: We're just clearing the collision registers again here, for no reason in particular other than it seems like a good thing to do.

1430: Here we simply check to see whether or not the bottom row of invaders is at the tops of the barriers. BARLIM, which I forgot to mention earlier, is used to keep track of how many sections of the barriers have been removed so far, which we use to determine where the tops of the barriers are. If BARLIM is equal to 3, then the barriers have been completely removed from the screen, and we don't have to worry about them anymore.

1435 We want the screen to be off while we change the display list, so we turn it off here and set the system clock to 0.

1440: Now we keep checking the system clock until one jiffy (one-sixtieth of a second) has passed. The reason we have to do this is because it takes up to one jiffy for the screen to turn completely off. If we don't wait, then the screen may appear to jump while we're changing the display list. Try leaving this line out and you'll see what I mean.

1450-1460: Now we actually change the display list. First we put the CHR 6 HSC into place, and then we

“scrunch” the display list (as I described above) by moving the end of the display list into a temporary string and then moving it back into place right after the CHR 6 HSC. You’re probably wondering why we can’t just move the end of the display list directly into place, without having to use the string. We could, except for the fact that when MOV-MEM is used to move things backwards in memory, as it is here, the area it’s moving in cannot overlap with the area it’s moving to. In this case the two areas do overlap, so we have to use DL\$ as a kind of mid-leman.

1470: Now we turn the screen back on, update BARLIM and get back to the main part of the program.

3030: We’ve added DL\$ to this line.

5150: And here we initialize BARLIM.

Ending the Game — A Beginning

Now how do things look? Pretty good until the invaders reach the bottom of the screen. At this point, the game should end, but right now it doesn’t. So let’s make it:

```
1400 POKE 53278,0:IF LINE+B
OTROW*2<>20 THEN 1430
1410 X=USR(ADR(MOVMEM$),ADR
(INV$)+5B,MEM1+LINE*24,287-
48*(5-BOTROW))
1420 GOTO 2020
2020 STOP
```

1400: We check to see if the bottom row of invaders has reached the bottom of the screen (the level of the player’s base). If it hasn’t, then we skip ahead to line 1430.

1410: The invaders have reached the bottom of the screen, so we want to make sure they’re drawn in their final position before we end the game. This line takes care of it.

1420: Now we go off to the end-of-game routine.

2020: For now, our end-of-game routine simply stops the program. Later on we’ll expand it into something a little more appropriate.

We’re just about set now except for two more details. If the invaders reach the bottom of the screen after they’ve hit the right-hand edge, then the alien saucer will still fly merrily across the top of the screen, regardless of the fact that the game is over (change line 2020 to 2020 GOTO 2020 to see what I mean). One simple change (for this column at least) will take care of this problem:

```
1290 IF CHANGE=1 OR LINE+BO
TROW*2=20 THEN 1400
```

Lastly, we run into the problem that the player can shoot the invader explosions as the invaders near the bottom of the screen. This didn’t happen before because the explosion was cleared before a new missile could reach it. The simplest solution to this problem is not to let the player shoot until the explosion is cleared:

```
1120 IF STRIG(0)=1 OR PEEK(
1700)<>0 OR PEEK(1720)<>0 O
R EF>1 THEN 1250
```

Next month we’ll finish BASIC Invaders, Don’t miss it!

80 COL. SCREEN!—NEW PRICES!

Turbobase™

Winner ANTIC awards '88

"IBM power without the price... I really can't think of any feature associated with running a business that has been left out—except for the **huge** prices charged for comparable software on MS-DOS computers."
—ANTIC, Dec. '87

"... the most time consuming review I have ever done, due to the number of features... Turbobase finally gives what 8-bit owners have been clamoring for years; true, powerful business software... set up a fully capable business system for less than \$1,000... customer support is superb... Practicality-excellent. Documentation-excellent."
—COMPUTER SHOPPER, Aug. '87

"... one of the most powerful and versatile database programs available..."
—COMPUTER SHOPPER, Aug. '88

COMPARE TO IBM CLONES:

• Capability	• Complete Documentation	• Speed among thousands of records
• Capacity	• \$20-\$50 Customizations	• Ease of learning (per feature)
• Remote Terminals	• One package/all modules	• Number of Existing error messages
• Exhaustive Support	• All Hardware Upgrades	• Adaptability to Existing Application
• No Disk Switching	• Brand Name Hardware	• Hardware/DOS easier than Clone MS DOS™
• Tiny Footprint	• True Integration	• Faster Back-up to inexpensive floppy
• Not Copy Protected	• Free Application Set-up	• Complete Invoice/Payments Error Checking

Turbobase takes \$20,000 video store sale from IBM... S.V. Plainfield, NJ
Turbobase takes \$20,000 IBM sale for waterbed store... A.J. Phoenix, AZ
Turbobase replaces \$37,000 air conditioning application... A.B. Alton, NH
Until you have Turbobase you don't have a database!... Acorn Users Group

Micromiser is looking for resellers. If you have 2 DD drives, or an MIO™, or hard disk. **You qualify** for free training, dealer prices, marketing/direct mail help, and myriad customer references who express extreme satisfaction with Turbobase. Compare the Turbobase™/MIO™ configuration at \$830 (all hardware & software except printer) with the IBM AT™. Immediate **RAM access** to 6,000 invoices, or 15,000 inventory items, or 50,000 G/L records, or 20,000 payroll records, or any combination of above! With a hard drive (add only \$100) the figures go up! 4,000 addresses too! An unbeatable selling point: replace any component for the cost of a typical IBM™/Apple™ repair bill! The small business market is yours! **Just ask, "Is IBM™ compatibility worth \$20,000 to you?"**

TURBOBASE™ — the all in one database/business system: 3 databases + word processor includes file manager/spread sheet/relational features/accounting/report generator, G/L, P/S, AR, AP, open invoicing/statements, inventory, payroll, mailing, utilities, all truly integrated in one program/manual so simplified that we can present complete *detailed* instructions in only 700+ pages of superb documentation (third re-write) includes separate Quick Course and Cookbook + 8 disk sides. Runs on any 48K 8-bit Atari. **only 1 drive req. Call today!**

Turbobase \$159—Turbo Jr \$99
For XEP—80 col. screen:
Turbobase 80 \$179—Jr 80 \$119
w/80 col word processor add \$24

STowners! Ask about Ultrabase ST (BAW monitor only) all Turbobase features + much more + Ultimate SIMPLICITY and speed

80 COL WORD PROCESSOR \$49 **(407) 857-6014**

MICROMISER SOFTWARE, 1635-A HOLDEN AVE., ORLANDO, FL 32809

ST NOTES

by Frank Cohen

The First Atari Convention of Canada turned out several thousand Atari enthusiasts and some surprise products for both the 8-bit and ST.

Amidst cold and rainy weather, several thousand Atari enthusiasts converged on the Toronto Airport Hilton last November for the First Canadian Atari Convention. The show turned out to be very successful; outside it rained and hailed, while inside, people spent their money buying new and used software for the ST and 8-bit Atari home computers. Everyone seemed to leave with a smile.

Atarifest History

Created by Sandy Austin, Atari Corporation's former Users' Group Coordinator, the tradition of the Atari Festival was to take Atari's grass roots marketing effort directly to the public. Early on, Atari management decided not to promote the ST computer with

substantial radio, print or television ads; instead, they opted for a city-by-city tour of the major computer markets in the United States. Using the resources of the local Atari users' groups, Atari had a motivated and enthusiastic audience for their new line of home computers.

The first show, held in Glendale, drew more than 4,500 attendees. Many found the Glendale Atarifest to be their first chance to look at the ST computer and some of the software offered by the many participating manufacturers. Austin was told to schedule more shows. A trailer truck was leased to transport ST computers, booths and other equipment from city to city. Atarifests were more or less successful in Worcester, Salt Lake City, Detroit, Pittsburgh, Washington,

Seattle and several other cities.

In fact, the Atarifest formula worked *too* well. Soon, Austin found more than one show had to be scheduled each month. At best, an Atarifest took six months to plan correctly. Tensions inside Atari mounted, and eventually Austin tendered her resignation.

The turning point occurred at the Dallas Atarifest. Held at the Infomart, a mall completely devoted to computers and technology, Austin arranged for three users' groups to share the responsibility of arranging the show. The differences between the groups caused an all-out war. The Dallas Atari Computer Enthusiasts (DALACE), and the North Texas Users' Group were particularly caustic towards each other. During the show, one users' group member began shouting,

"DALACE is a pirate club."

Atari was not blameless. Neil Harris, Atari's Director of Corporate Communications, had gone to the previous Atarifests giving well-attended speeches on Atari's plans for the future of the ST and 8-bit computers. But this time, at the last moment, Harris backed out of the show because his wife was nearing the end of her pregnancy. The local users' groups, unaware of the situation, were outraged, venting their frustration on the only Atari employee at the show, Sandy Austin. At a dinner for the exhibitors, Austin was attacked for Atari's wishy-washy marketing efforts and vaporware products.

Under pressure from the users' groups, Harris made a hurried trip to Dallas to give one speech, then departed for Sunnyvale. The show left Austin tired and burned-out. Later, Austin confided to ANALOG that she was looking for another job.

Several Atarifests have been scheduled for '89. However, without the support of a central Atari representative, like Austin, coordinating and supporting the project, local users' groups are finding it very difficult to hold a show. The costs of advertising, printing and securing the show location usually amount to \$10,000. Without Atari's support, users' groups do not have the resources to pull it off.

Atari Canada

The First Canadian Atari Computer Convention was supported by Atari Canada, who displayed all of Atari's products in a trade show-sized booth. Since Atari Canada's offices are located in the Toronto area, most of Atari's sales and support employees were at the show promoting products and answering questions.

Atari Canada is a subsidiary of Atari Corporation. "Atari Canada has spent the past three years building a good reputation among Canadians," said Martin Herzog, Product Manager for Atari Canada. Headed by Ian Kennedy, Atari Canada is the antithesis of

Atari Corp. USA. Atari Canada has created a series of advertising campaigns that make the ST a well-known and well-respected Canadian personal computer.

Atari Canada has built a good relationship with the approximately 150 Canadian ST dealers. In August, 1987, Atari Canada hosted all Canadian dealers at a one-day dealer trade show in Toronto to roll-out the new Mega ST. Between hour-long seminars on sales strategies, dealers were invited into a special hall where the latest hardware and software ST products were on display. Third-party manufacturers were invited to the dealer show to make contact with the local salesmen. Atari Canada supplied meals, drinks, and even gave each dealer five shares of Atari public stock as a sign of the partnership Atari Canada intended to build with the Canadian dealers. Atari Canada's message was expressed best by Kennedy, "We want [dealers] to be partners in our success."

Toronto Show

Arriving in Toronto International Airport, a visitor might think this city is in the United States. It is not! Canada is a foreign country, complete with different driving rules, customs, vocabulary and money. This is particularly true of Canadians from Ontario, the province where Toronto is located.

Toronto has been called the New York of Canada. The CN Tower is just as touristy as the Empire State Building; both structures give you a breathtaking view of a huge metropolis.

The Toronto show turned out to be much larger than most Atarifests. Several local dealers were on hand, with what appeared to be their entire inventory of hardware and software products. The balance of exhibitors was equally matched between business software vendors and MIDI (Musical Instrument Digital Interface) music-related developers. A number of the region's Atari users' clubs displayed newsletters and public-domain software libraries.

The Toronto show turned out to be much larger than most Atarifests.

Atari Canada displayed the *Atari PCI*, Atari's ill-fortuned IBM PC clone. Three hundred PCIs were shipped to Canada in 1988. An Atari Canada employee told ANALOG that because of the high price and lack of salability, they were trying to unload them at any cost.

Of news to ST users, the *Atari PCI* comes with the new *PCF-554* 5.25-inch double-sided floppy disk drive which is compatible with ST and Mega computers. Atari Canada has been marketing the *PCF-554* since last summer to Canadian ST users, while Atari USA has still not released the drive to the U.S. market.

Rumors persisted that Atari Canada was entering the software market with several bundled software/hardware combinations. High on the list of possibilities was a 1040 ST computer bundled with *WordPerfect*, *Regent Base 2* and *LDW Power*. Atari Canada is looking for ways to make the ST attractive to businessmen working with IBM PCs at work.

Atari Canada hosted a benefit *MIDI Maze* competition. Twelve 1040 STs were connected to allow players to compete against one another for software prizes.

A new company on the ST scene, CodeHead Software, brought Charles Johnson and John Eidsvoog to the show demonstrating their new utility products for the ST. *G+Plus* (\$34.95 List) is a replacement for Atari's terrible GDOS system. GDOS gives an Atari ST computer the ability to display and print graphics and fonts with a number of popular printers. Atari's release of GDOS is filled with major bugs that, among other things, noticeably slow down the GEM operating system, crash many ST programs and eat up memory like a hungry child. Johnson and Eidsvoog wrote their own software to replace GDOS; the result: *G+Plus*.

MultiDesk (\$39.95 List) is CodeHead's second product offering. The stock GEM system allows only six desk accessories. (Desk accessories are smaller programs that may be

used while a larger GEM program is running.) *MultiDesk* removes the limit, allowing up to 32 desk accessories to be used.

ANALOG expected a database showdown at the Toronto show. The makers of *dbMAN*, *Superbase* and *Regent Base* have made significant strides into the Canadian ST database market. Only *Regent* made it to the show. On display, *Regent Base 2* (\$150 List) is a very powerful relational database that uses the Structured Query Language—the emerging standard microcomputer database language. *GFA BASIC* users have a new accessory product that enhances *GFA BASIC*'s instruction set. The *SQL Database Add-On* for *GFA BASIC* (Versions 2 & 3) is a command SQL interpreter that uses *GFA BASIC* variables to pass SQL commands and retrieved data. The *SQL Add-On* (\$59.95 List) assumes very little knowledge of databases, but is a powerful database manager.

Michtron was on hand, showing what it had left to sell. It seems the demand for *GFA BASIC 3.0* has outpaced the supply of manuals. Michtron has also recently announced it will be marketing yet another BASIC, *Hi-Soft BASIC*, developed by a British company.

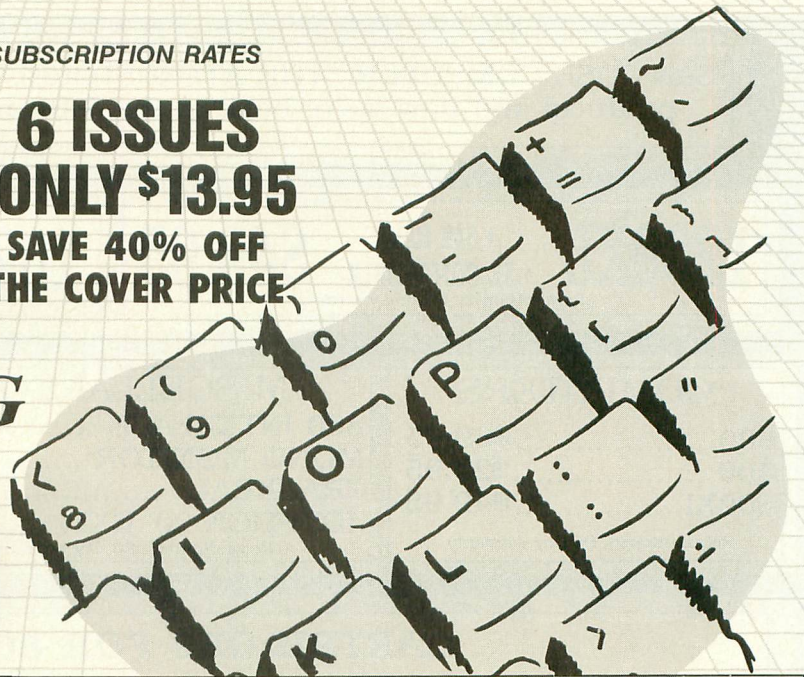
All in all, the First Atari Convention of Canada turned out to be successful for its organizers, the dedicated Atari enthusiasts that attended, and the manufacturers showing their new products. It seems a safe bet that there will be a second Atari Convention of Canada.

Frank Cohen has been developing Atari programs since his first commercial product, Clowns & Balloons. He later developed Regent Base 2, an SQL 4GL database, and is currently involved with several other ST-related products and small business software packages. If you have questions or comments he may be contacted directly on Compuserve (76004,1573) and Genie (FCOHEN), or directly at P.O. Box 14628, Long Beach, CA 90803-1208

Rumors persisted that Atari Canada was entering the software market with several bundled software/hardware combinations.

SPECIAL CHARTER SUBSCRIPTION RATES

**6 ISSUES
ONLY \$13.95**
SAVE 40% OFF
THE COVER PRICE



INTRODUCING

PC Laptop

COMPUTERS MAGAZINE

SIGN ME UP FOR 6 ISSUES OF LAPTOP MAGAZINE FOR ONLY \$13.95!
 PAYMENT ENCLOSED BILL ME CHARGE MY VISA MC
 No. _____ EXP _____ SIGNATURE _____
 NAME _____
 ADDRESS _____
 CITY _____ STATE _____ ZIP _____
 Money back on unused subscriptions if not completely satisfied. Foreign add \$7. Make checks payable to LFP, INC., Your first issue will arrive in 6 to 8 weeks. Watch for it.
 CJQWV _____ Offer expires June 28, 1989

The world of ATARI-ST continues to grow by leaps and bounds, and ST-LOG is there every step of the way! We stand apart from the competition by offering more color, comprehensive reviews and in-depth features. SUBSCRIBE NOW!

12 Issues \$28

MCPYW



12 Issues with Disk \$79

DCPYW

PAYMENT ENCLOSED BILL ME
 CHARGE MY: VISA MASTERCARD

CARD # _____ EXP _____ SIGNATURE _____
 NAME _____
 ADDRESS _____
 CITY _____ STATE _____ ZIP _____

MAKE CHECK PAYABLE TO L.F.P., INC., P.O. Box 16928, N. Hollywood, CA 91615. Offer expires June 28, 1989

WE'RE BACK!

SAN JOSE COMPUTER, 'THE ATARI STORE', is the largest seller of ATARI products and now we're back in the 8-Bit market to serve you better.

1200XL • 64K RAM • XE COMPATIBLE **\$49.95**
Reconditioned, 90 day warranty

1025 PRINTER **\$89.95**
Reconditioned, 90 day warranty

COMPUTERS

800 \$69.95
400 \$29.95
600XL \$49.95

Reconditioned, 90 day warranty

MISCELLANEOUS

850 INTERFACE \$89.95
MONO MONITOR \$39.95
TRACKBALL \$9.95
JOYSTICK 20' EXT. \$.99

Reconditioned, 90 day warranty

PRINTERS

825 \$79.95
1029 \$129.95
820 \$39.95

Reconditioned, 90 day warranty

CARTRIDGES FOR 800, XL, XE

TURMOIL \$4.95	TENNIS \$19.98	NECROMANCER \$19.98
PAC-MAN (no box) \$4.95	FINAL LEGACY \$19.98	RESCUE ON FRACTALUS \$19.98
DONKEY KONG (no box) \$4.95	MARIO BROS. \$19.98	BALLBLAZER \$19.98
GORF (400,800) \$4.95	DONKEY KONG JR. \$19.98	BLUE MAX \$19.98
DEMON ATTACK (400,800) \$4.95	JUNGLE HUNT \$19.98	STAR RAIDERS II \$19.98
DELUXE INVADERS \$4.95	MOON PATROL \$19.98	DAVID'S MIDNIGHT MAGIC \$19.98
JOURNEY TO THE PLANETS \$4.95	BATTLEZONE \$19.98	ARCHON \$19.98
MATH ENCOUNTER \$7.98	FOOD FIGHT \$19.98	GATO \$24.98
SKY WRITER \$14.95	HARDBALL \$19.98	ACE OF ACES \$24.98
FOOTBALL \$14.95	FIGHT NIGHT \$19.98	LODE RUNNER \$24.98
DEFENDER \$14.95	ONE ON ONE BASKETBALL \$19.98	BARNYARD BLASTER (XE gun) \$24.98
ROBOTRON \$19.98	DESERT FALCON \$19.98	ATARI LAB LIGHT MODULE \$29.98
		ATARI LAB STARTER KIT \$39.98

DISK SOFTWARE FOR 800, XL, XE

DAVID'S MIDNIGHT MAGIC \$4.98	CROSSCHECK \$4.98	SPIDERMAN \$4.98
ZORRO \$4.98	MOLECULE MAN \$4.98	HULK \$4.98
BANDITS (48K 400,800) \$4.98	CRYSTAL RAIDER \$4.98	VISICALC \$24.98
PROTECTOR II \$4.98	DESPATCH RIDER \$4.98	BOOKKEEPER W/ numeric keypad... \$29.98
CLAIM JUMPER \$4.98	MISSION ASTEROID \$4.98	GET RICH \$39.98

810
DISK
DRIVES
\$129.

8-BIT SERIAL I/O CABLE ... \$5.95

SAN JOSE COMPUTER

T H E A T A R I S T O R E

Sunrise Plaza 640 Blossom Hill Rd. San Jose, CA 95123
(408) 224-8575 • BBS (408) 224-0952

5.25" DISKS
20 CENTS EA.*

QTY.	PRICE
10	\$4.00
100	\$29.95
*1000	\$200

MAJORITY ARE UNNOTCHED
CONTAINING OLD SOFTWARE

SHIPPING: ADD \$5.00 TO ALL ORDERS. AIR AND INTERNATIONAL SHIPPING EXTRA. THAT'S IT.
WARRANTY: 90 DAY WARRANTY ON ALL ITEMS. **TAX:** CALIFORNIA RESIDENTS ADD 7% SALES TAX.
PREPAYMENT: USE VISA, MASTERCARD, MONEY ORDER, CASHIER'S CHECK OR PERSONAL CHECK.
PERSONAL CHECK MUST CLEAR PRIOR TO SHIPMENT. **C.O.D.:** CASH, CASHIER'S CHECK OR M.O. ONLY.
Prices subject to change without notice. Brand and/or product names are trademarks or registered trademarks of their respective holders.
Ad produced on an ATARI ST.

U T I L I T Y M/L EDITOR

For use in machine-language entry.

by Clayton Walnum

M/L Editor provides an easy method to enter our machine-language listings. It won't allow you to skip lines or enter bad data. For convenience, you may enter listings in multiple sittings. When you're through typing a listing with M/L Editor, you'll have a complete, runnable object file on your disk.

There is one hitch: It's for disk users only. My apologies to those with cassette systems.

Listing 1 is M/L Editor's BASIC listing. Type it in and, when it's free of typos, save a copy to disk, then run it.

On a first run, you'll be asked if you're starting a new listing or continuing from a previously saved point. Press S to start, or C to continue.

You'll then be asked for a filename. If you're starting a new listing, type in the filename you want to save the program under, then press RETURN. If there's already a file by that name on the disk, you'll be asked if you wish to delete it. Press Y to delete the file, or N to enter a new filename.

If you're continuing a file, type in the name you gave the file when you started it. If the program can't find the file, you'll get an error message and be prompted for another filename. Otherwise, M/L Editor will calculate where you left off, then go on to the data entry screen.

Each machine-language program in ANALOG Computing is represented by a list of BASIC data statements. Every line contains 16 bytes, plus a checksum. Only the numbers following the word DATA need to be considered.

M/L Editor will display, at the top of the screen, the number of the line you're currently working on. As you go through the line, you'll be prompted for each entry. Simply

type the number and press Return. If you press Return without a number, the default is the last value entered.

This feature provides a quick way to type in lines with repetitions of the same number. As an added convenience, the editor will not respond to the letter keys (except Q for "quit"). You must either enter a number or press Return.

When you finish a line, M/L Editor will compare the entries' checksums with the magazine's checksum. If they match, the screen will clear, and you may go on to the next line.

If the checksums *don't* match, you'll hear a buzzing sound. The screen will turn red, and the cursor will be placed back at the first byte of data. Compare the magazine listing byte by byte with your entries. If a number is correct, press RETURN.

If you find an error, make the correction. When all data is valid, the screen will return to gray, and you'll be allowed to begin the next line.

Make sure you leave your disk in the drive while typing. The data is saved continuously.

You may stop at any time (except when you have a red screen) by entering the letter Q for byte 1. The file will be closed, and the program will return you to BASIC. When you've completed a file, exit M/L Editor in the same way.

When you've finished typing a program, the file you've created will be ready to run. In most cases, it should be loaded from DOS via the L option. Some programs may have special loading instructions; be sure to check the program's article.

If you want the program to run automatically when you boot the disk, simply name the file AUTORUN.SYS (make sure you have DOS on the disk.).

The two-letter checksum code preceding the line numbers here is not a part of the BASIC program. For more information, see the "BASIC Editor II" elsewhere in this issue.

LISTING 1: BASIC LISTING

```
AZ 10 DIM BF(16),M$(4),A$(1),B$(1),F$(15)
LF 11 DIM MOD$(4)
BN 20 LINE=1000:RETRN=155:BACKSP=126:CHK5
UM=0:EDIT=0
GO 30 GOSUB 450:POSITION 10,6:?"Start or
Continue? ";GOSUB 500:?"CHR$(A)
ZG 40 POSITION 10,8:?"FILENAME":INPUT F
$:POKE 752,1:?" "
FE 50 IF LEN(F$)<3 THEN POSITION 20,10:?"
":GOTO 40
MF 60 IF F$(1,2)<>"D:" THEN F1$="D":F1$(
3)=F$:GOTO 80
KL 70 F1$=F$
TN 80 IF CHR$(A)="5" THEN 120
FD 90 TRAP 430:OPEN M2,4,0,F1$:TRAP 110
HQ 100 FOR N=1 TO 16:GET M2,A:NEXT N:LINE
=LINE+10:GOTO 100
HM 110 CLOSE M2:OPEN M2,9,0,F1$:GOTO 170
VT 120 TRAP 160:OPEN M2,4,0,F1$:GOSUB 440
:POSITION 10,10:?"FILE ALREADY EXISTS
!":POKE 752,0
ZU 130 POSITION 10,12:?"ERASE IT? ";GOS
UB 500:POKE 752,1:?"CHR$(A)
UH 140 IF CHR$(A)="n" OR CHR$(A)="N" THEN
CLOSE M2:GOTO 30
QG 150 IF CHR$(A)<>"Y" AND CHR$(A)<>"y" T
HEN 130
BH 160 CLOSE M2:OPEN M2,8,0,F1$
IE 170 GOSUB 450:POSITION 10,1:?"NOW ON
[RE]:";LINE:CHKSUM=0
GH 180 L1=3:FOR N=1 TO 16:POSITION 13*(X
10)+12*(N)>9,N+2:POKE 752,0:?"BYTE H"
:":X":?"":GOSUB 310
KH 190 IF EDIT AND L=0 THEN BYTE=BF(X):GO
TO 210
FY 200 BYTE=VAL(M$)
OZ 210 MOD$=M$
BU 210 POSITION 22,X+2:?" BYTE:" "
YZ 220 BF(X)=BYTE:CHKSUM=CHKSUM+BYTE*M:IF
CHKSUM>9999 THEN CHKSUM=CHKSUM-10000
MS 230 NEXT X:CHKSUM=CHKSUM+LINE:IF CHKSUM
M>9999 THEN CHKSUM=CHKSUM-10000
IG 240 POSITION 12,X+2:POKE 752,0:?"CHEC
KSUM: ";L1=4:GOSUB 310
EM 250 IF EDIT AND L=0 THEN 270
DM 260 C=VAL(M$)
SY 270 POSITION 22,X+2:?" C:" "
IL 280 IF C=CHKSUM THEN 300
DI 290 GOSUB 440:EDIT=1:CHKSUM=0:GOTO 180
LM 300 FOR X=1 TO 16:PUT M2,BF(X):NEXT X:
LINE=LINE+10:EDIT=0:GOTO 170
FU 310 L=0
KZ 320 GOSUB 500:IF (A=ASC("Q") OR A=ASC(
"q")) AND X=1 AND NOT EDIT THEN 420
PD 330 IF A<>RETRN AND A<>BACKSP AND (A<4
0 OR A>57) THEN 320
DX 331 IF A=RETRN AND M$="" THEN M$=MOD$
TD 335 IF A=RETRN AND L=0 AND X>1 THEN 35
0
JR 340 IF ((A=RETRN AND NOT EDIT) OR A=B
ACKSP) AND L=0 THEN 320
DW 350 IF A=RETRN THEN POKE 752,1:?" ":R
ETURN
GG 360 IF A<>BACKSP THEN 400
SA 370 IF L>1 THEN M$=M$(1,L-1):GOTO 390
A5 380 M$=""
? CHR$(BACKSP):L=L-1:GOTO 320
BB 400 L=L+1:IF L>1 THEN A=RETRN:GOTO 35
0
WX 410 M$(L)=CHR$(A):?"CHR$(A):":GOTO 320
KM 420 GRAPHICS 0:END
YT 430 GOSUB 440:POSITION 10,10:?"NO SUC
FILE!":FOR N=1 TO 100:NEXT N:CLOSE
M2:GOTO 30
FD 440 POKE 710,48:SOUND 0,100,12,8:FOR X
=1 TO 50:NEXT X:SOUND 0,0,0,0:RETURN
MY 450 GRAPHICS 23:POKE 16,112:POKE 53774
,112:POKE 559,0:POKE 710,4
NR 460 DL=PEEK(560)+256*PEEK(561)+4:POKE
DL-1,70:POKE DL+2,6
HM 470 FOR X=3 TO 39 STEP 2:POKE DL+X,2:N
EXT X:FOR X=4 TO 40 STEP 2:POKE DL+X,0
:NEXT X
ZH 480 POKE DL+41,65:POKE DL+42,PEEK(560)
:POKE DL+43,PEEK(561):POKE 87,0
AC 490 POSITION 2,0:?"analog M1 editor":
POKE 559,34:RETURN
MZ 500 OPEN M1,4,0,"K":GET M1,A:CLOSE M1
:RETURN
```

Attention Programmers!

ANALOG Computing is interested in programs, articles, and software review submissions dealing with the Atari 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.

All submissions for publication, both program listings and text, should be provided in printed and magnetic form. Typed or printed copy of text is mandatory and should be in upper and lower case with double spacing. By submitting articles to **ANALOG Computing**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ANALOG Computing**. 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 your programs and articles to:
ANALOG Computing
P.O. Box 1413-M.O.
Manchester, CT 06040-1413

I N T R O D U C I N G

Video Games & Computer Entertainment™

**12 ALL COLOR ISSUES
ONLY \$19.95**

**Save over \$15
off the cover price!**



- GAME REVIEWS
- ARCADE ACTION
- STRATEGY GUIDES
- TECHNICAL REPORTS
- COMPUTER SOFTWARE

Video Games & Computer Entertainment

P.O. BOX 16927, N. HOLLYWOOD, CA 91615

Yes!

Sign me up for 12 issues for only \$19.95—I'll save over \$15!

Payment Enclosed — Charge My VISA MC NAME _____

_____ EXP _____ ADDRESS _____

SIGNATURE _____ CITY _____ STATE _____ ZIP _____

Money back on unused subscriptions if not satisfied!
Foreign—add \$10 for postage.

Your first issue will arrive in 6 to 8 weeks.

WATCH FOR IT!!

Offer expires June 28, 1989

CJQYA

No Frills Software
800 East 23rd Street
Kearney, NB 68847
(308) 234-6250
\$19.95

The Converter, Version 1.2

Reviewed by Matthew J. W. Ratcliff

Over the years, *Print Shop* has been the most popular graphics program for the 8-bit Ataris. More recently *Newsroom* has been ported to the Atari, making it very close to a complete desktop-publisher. While support for the 8-bit Ataris has slackened, Hi-Tech Expressions has come on strong with many impressive graphics printing programs at remarkably low prices. Among these, the most popular are *Print Power*, *AwardWare* and *Sesame Street Print Kit*. If you have more than one of these programs, then you need *The Converter* from No Frills Software.

The Converter can translate graphics from one format to another with ease. It can provide a directory of *Print Shop* disks, allow you to load and edit an icon and save it to a *Print Power* disk, for example. Once loaded and run, *The Converter* first allows you to specify Drive 1 or 2 as the current drive. At the left of the display, occupying the upper three-quarters of the screen, is the graphics work area. Here all your graphics will be loaded and displayed. With a joystick plugged into Port 1, you can edit your favorite graphics and icons here.

On the right side of the display is a text window, where directories are listed. It is also used as a clip area, for use when icons must be trimmed before being saved to a new format. At the bottom left is a small text-command window. Using the arrow keys on the keyboard you may highlight LOAD, SAVE, EDIT, AW-PrP (convert *AwardWare* to *Print Power* format), or SPEC (change drive specification) and simply press Return. The bottom center of the screen is used to display text prompts, and show the current operating status of *The Converter*. Above this is one line of text where you will specify drive and filenames.

Once the drive is logged, you are ready to select an icon to edit. *The Converter* is very smart in that it detects just what sort of disk format you are working with. If a standard DOS format disk is detected, you are prompted to load a *Print Power*, *AwardWare* graphic or an *AwardWare* seal. *Sesame Street Print Kit* is the same format as *Print Power*. If a *Print Shop* disk is detected, which is a non-standard format, you are simply prompted for an icon name.

You may press the Return key at any filename prompt for a list of the current files,

of the current graphics type. When the directory is listed to the 16-line text window, you may keep pressing Return to see more files or enter the name of the graphic to load it. *Print Power* graphics names may have both upper and lower case, and their names must be typed exactly as they are seen on the display. *The Converter* is smart enough to switch the keyboard input to lower case, when necessary, and back to upper case when appropriate. The user interface of *The Converter* is well-thought-out. I am impressed with its user friendliness and built-in "intelligence" which helps prevent goof-ups.

Once the name has been entered, you may immediately select SAVE to convert it to a new format. The first time SAVE is selected, you are prompted to select Clip Art, *AwardWare* Graphic or *AwardWare* Seal formats. Since there are so many *Print Shop* icon editors and converters available in the public domain, the authors of *The Converter* do not support saving to *Print Shop* disks, only loading from them.

The Clip Art files are for *Newsroom* users. *The Converter* will format a clip-art disk for you, if necessary. Why pay big money for *Newsroom* graphics disks, when you can convert tons of *Print Shop* icons in the public domain?

Saving to an *AwardWare* graphic or seal will add more graphics to the nifty ribbons, awards and certificates you can make with this program. *Print Power* and *Sesame Street Print Kit* use a different directory than does *AwardWare*. When you select the "AW-PrP" menu option, *The Converter* will read all of the *AwardWare* graphics on the selected drive and build the correct directory file, allowing the same disk to be used with any of these Hi-Tech programs. Any time you add another *AwardWare* graphic to an existing disk, you must run the "AW-PrP" option once again, to rebuild this directory file.

The Converter documentation warns you not to place more than 64 graphics files on a disk, due to directory size-limitations of Atari DOS. SpartaDOS users can have twice that number on a disk. And SpartaDOS X users can have over 1400 graphics on a single disk. Yes, that is correct, *The Converter* is completely SpartaDOS compatible. It is not protected, allowing you to make a backup copy, run it under a different DOS, run it

from a large RAMdisk, or even install it on a hard drive! I switched to Hi-Tech products because they are unprotected, allowing me to use them on my ICD FA-ST hard drive. With *The Converter* I have been able to move all of my *Print Shop* icons, from their incompatible format disks, onto my hard drive, for use with my favorite Hi-Tech Expressions software.

When a graphic is loaded, you may select EDIT to modify the picture. This is a powerful feature lacking in Hi-Tech's products. The arrow keys, or a joystick in Port 1, moves the editing cursor (a flashing pixel). While editing, a complete menu of all the controls is displayed in the text window at the right.

To shift the picture you can simply press the Control key, and then the arrow keys. This will rapidly slide the graphic in any direction you choose. Box and line commands are supported. The graphic may be inverted with the negative command. Before you get ready to make a drastic change, simply press the B key to buffer the picture, then, if you do not like the change, press U to undo it.

This editor is much faster than the one found in *Newsroom*. If you want to personalize some of the *AwardWare* graphics, *The Converter* will certainly help you get the job done.

The conversions do not come full circle, but seem quite adequate. To summarize, *The Converter* can load *Print Shop* icons, *AwardWare* graphics, *AwardWare* seals, *Print Power* graphics and *Sesame Street Print Kit* graphics. *The Converter* can save *AwardWare* graphics, *AwardWare* seals (and make *AwardWare* disks compatible with *Print Power*) and *Newsroom* clip art.

The Converter is one of the most useful, powerful and flexible utilities I have encountered in ages. It does just about everything and does it well. If you have any of the Hi-Tech products mentioned, you will want *The Converter* for its graphics-editing capabilities. If you have two or more of these products from different vendors, *The Converter* will make short order of pooling all your favorite graphics. No Frills Software also sells a myriad of public-domain *Print Shop* icon disks, collected from users' groups and bulletin boards across the country. *The Converter* is a superbly crafted product, at a down-to-earth price.



800/LX/XE SOFTWARE

ALL TITLES ON DISK

ENTERTAINMENT

ALIANIS	26.95
ALT. REALITY CITY	26.95
ALT. REALITY DUNGEON	26.95
BEYOND CASTLE WOLF	14.95
BISMARCK	26.95
BRUCE LEE	17.95
CASTLE WOLFENSTEIN	14.95
DALLAS QUEST	7.95
D-BUG	7.95
GAUNTLET (64K)	31.50
DEEPER DUNGEONS	22.50
GUNSLINGER	26.95
HARD HAT MAC	7.95
KARATEKA	13.50
KORONIS RIFT	13.50
LAST V-8	8.95
MONTEZUMA'S REVENGE	14.95
MOUSETRAP	17.95
NINJA	8.95
O'RILEY'S MINE	9.95
RESCUE ON FRACTALAS	13.50
SPEEDKING	8.95
SPIDERMAN	7.95
SPIFFIRE 40	31.50
STOCKMARKET	22.50
STRIP POKER	26.95
THE HULK	7.95
TOMAHAWK (64K)	26.95
TOP GUNNER	17.95
ZAXXON	13.50

SPORTS

FIGHT NIGHT	17.95
LEADERBOARD	13.50
MICROLEAGUE BASEBALL	35.95

MUSIC

MUSIC CONSTRUCTION SET	13.50
VIRTUOSO	29.95

PROGRAMMING

BASIC XL	53.95
BASIC XL TOOLKIT	26.95
BASIC XE	71.95
KYAN PASCAL	62.95
LIGHTSPEED C	35.95
LOGO	19.95
MAC/65	71.95
MAC/65 TOOLKIT	26.95
PILOT	19.95

PRODUCTIVITY

ATARIWRITER+	39.95
BANK STREET WRITER	14.95
CELEBRITY COOKBOOK	26.95

COMPUTE YOUR ROOTS	35.95
COMPUTER GOURMET	26.95
FAMILY FINANCE	6.95
HOME ACCOUNTANT	19.95
HOME FILING MANAGER	6.95
HOMEPAK	24.95
LETTER WIZARD	29.95
NEWSROOM (1050 - 64K)	44.95
NEWS STATION	26.95
NEWS STA. COMPANION	26.95
PRINT POWER (1050)	13.50
PROOF READER	17.95
PUBLISHING PRO	35.95
SYNCALC	31.50
SYSALC (130XE)	44.95
SYNTREND (130XE)	35.95
THE LOTTO PROGRAM	17.95
TIMWISE	6.95
TURBOWORD/80 COLUMN	
REQUIRES XEP80	44.95
VIDEO TITLES (64K)	26.95
GRAPHICS COMPANION	17.95
VISICALC	24.95

ART

ANIMATION STATION	89.95
BLAZING PADDLES	31.50

EDUCATION

BUZZWORD	35.95
HEY DIDDLE (AGE 3-10)	9.95
MASTER TYPE	14.95
PLANATARIUM	22.50
TOUCH TYPING	9.95
CBS (AGE 3-6):	
ASTROGROVER	8.95
BIG BIRD SPEC DELIVE	8.95
ERNIE'S MAGIC SHAPE	8.95

DESIGNWARE:

MATHMAZE (6-11)	35.95
MISSION ALGEBRA (13+)	35.95
SPELLICOPTER (6-11)	35.95

TINK TONK (AGE 4-6):

ABC'S	8.95
COUNT AND ADD	8.95
SMART THINKER	8.95
SPELLING	8.95
SUBTRACTION	8.95
THINKING SKILLS	8.95
ALL 6 TINK TONKS	39.95

WEEKLY READER (PRE-SCHOOL):

STICKY BEAR SHAPES	26.95
STICKY BEAR NUMBERS	26.95
STICKY BEAR ABC'S	26.96
STICKY BEAR OPPOSITE	26.95

800/LX/XE SOFTWARE

ALL TITLES ON CARTRIDGE

ENTERTAINMENT

ALIEN AMBUSH	9.95
ACE OF ACES (XL/XE)	24.95
ARCHON	19.95
ASTEROIDS	15.95
ATARI TENNIS	9.95
BALL BLAZER	19.95
BARNYARD BLASTER	
(REQ. LIGHT GUN)	24.95
BATTLEZONE	19.95
BLUE MAX	19.95
CAVERNS OF MARS	14.95
CENTIPEDE	14.95
CHICKEN	9.95
CHOPLIFTER	14.95
CLAIM JUMPER (400/800)	9.95
CLOUDBURST	9.95
DAVIDS MIDNIGHT MAGIC	19.95
DEFENDER	14.95
DESERT FALCON	19.95
DIG DUG	19.95
DONKEY KONG	5.00
DONKEY KONG JR.	19.95
EASTERN FRONT (1941)	19.95
E.T. PHONE HOME	9.95
FINAL LEGACY	19.95
FOOD FIGHT (XL/XE)	19.95
FROGGER	14.95
GALAXIAN	19.95
GATO	24.95
GORF (400/800)	5.00
GYRUSS	14.95
JOURNEY TO PLANETS	9.95
JOUST	19.95
JUNGLE HUNT	19.95
KABOOM!	14.95
LODE RUNNER	24.95
MARIO BROS.	19.95
MILLIPEDE	19.95
MISSILE COMMAND	5.00
MOON PATROL	19.95
MR. COOL	9.95
MS. PAC MAN	19.95
NECROMANCER	19.95
PAC MAN	5.00
PENGO	19.95
POLE POSITION	19.95
POPEYE	14.95
Q-BERT	14.95
QIX	14.95
RESCUE ON FRACTALAS	19.95
RETURN OF THE JEDI	14.95
ROBOTRON:2084	19.95

SKY WRITER	14.95
SLIME (400/800)	9.95
SPACE INVADERS	14.95
STAR RAIDERS	5.00
STAR RAIDERS II	19.95
SUPER BREAKOUT	9.95
TRACK & FIELD	24.95
WIZARD OF WOR	5.00

SPORTS

FIGHT NIGHT	19.95
FOOTBALL	14.95
HARDBALL	19.95
ONE ON ONE (XL/XE)	19.95
TRACK & FIELD	24.95

PRODUCTIVITY

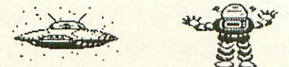
ATARIWRITER	19.95
MICROFILERS	22.50

EDUCATION

MATH ENCOUNTERS	9.95
FISHER PRICE (PRE SCHOOL):	
DANCE FANTASY	8.95
LINKING LOGIC	8.95
LOGIC LEVELS	8.95
MEMORY MANOR	8.95

SPINNAKER (AGE 3-10):

ALF IN COLOR CAVES	9.95
ALPHABET ZOO	9.95
DELTA DRAWING	9.95
KIDS ON KEYS	9.95
KINDERCOMP	9.95
(AGE 7 - ADULT):	
ADVENTURE CREATOR	9.95
FRACTION FEVER	9.95



ATARI XE GAME MACHINE \$139.95

Includes Missile Command, Flight Simulator II, Bug Hunt, light gun, Joystick, BASIC programming language, and 64K of memory with detachable keyboard. Add a diskdrive and printer for complete home computer system!!

SUPER SPECIALS

RECONDITIONED ATARI MERCHANDISE - 30 DAY WARRANTY

BOOKS ONLY DE RE ATARI 10.00 ATARIWRITER 10.00 DOS 2,5 12.95 BASIC REF. 5.00 LOGO 10.00 BOOKKEEPER 10.00	ATARI SPACE AGE JOYSTICK \$5.00 GREAT GIFTS!	800 (48K) COMPUTER \$79.95 INCLUDES BASIC	ATARI 820 PRINTER 40 COLUMNS DIRECT CONNECT NO INTERFACE REQUIRED \$39.95	1010 PROGRAM RECORDER \$29.95 410 PROGRAM RECORDER - NO WARRANTY \$9.95	1030 MODEM WITH EXPRESS! \$24.95 GET ONLINE TODAY!
ATARI TRACKBALL \$9.95 SPICE UP THE ACTION IN YOUR ARCADE GAMES!!	400 (16K) COMPUTER \$29.95 48K UPGRADE KIT \$25.00	1020 COLOR PRINTER/PLOTTER \$29.95 40 COLUMNS WIDE INCLUDES PAPER AND COLOR PEN SET	ATARI NUMERIC KEYPAD \$7.95 INCLUDES HANDLER DISK -	ATARI BOOKKEEPER \$14.95 - NO BOX (19.95 WITH KEYPAD) \$24.95 - IN BOX (29.95 WITH KEYPAD)	DISKETTES AS LOW AS 20 CENTS 10 FOR \$4.00 100 FOR \$29.95 1000 FOR \$200 MOST ARE UNNOTCHED WITH OLD SOFTWARE

SHIPPING INFORMATION - Prices do not include shipping and handling. Add \$5.00 for small items (\$8.00 Min. for Canada). Add \$8.00 for disk drive. Calif. res. include 7% sales tax. Mastercard and Visa accepted if your telephone is listed in your local phone directory. Orders may be pre-paid with money order, cashier check, or personal check. Personal checks are held for three weeks before order is processed. C.O.D orders are shipped via UPS and must be paid with cash, cashier check or money order. International and APO orders must be pre-paid with cashier check or money order. \$20.00 minimum on all orders. All sales are final - no refunds - prices are subject to change. Phone orders accepted TUESDAY THROUGH FRIDAY from 10:00 am to 6:00 pm PST.

We carry a complete line of ATARI products and have a large public domain library. Write or call for free catalogue. (408) 749-1003 TUE - FRI 10AM - 6 PM

PRICES SUBJECT TO CHANGE WITHOUT NOTICE - ALL SALES ARE FINAL

BASIC Editor II

by Clayton Walnum

BASIC Editor II is a utility to help you enter BASIC program listings published in ANALOG Computing. To simplify the identification of errors, each program line is evaluated immediately after it's typed, eliminating the need for cumbersome checksum listings. When you've finished entering a program using BASIC Editor II, you can be certain it contains no typos.

An option is provided for those who wish to use standard BASIC abbreviations. Also, the program retains all Atari editing features. Finally, for those who prefer to type programs the conventional way, using the built-in editor, a post-processing mode is available. It allows you to check typing after the entire listing has been entered.

Typing in the Editor

To create your copy of BASIC Editor II, follow the instructions below— exactly.

Disk version:

- (1) Type in Listing 1, then verify your work with Unicheck (see Issue 39).
- (2) Save the program to disk with the command *SAVE "D:EDITORLI.BAS"*.
- (3) Clear the computer's memory with the command *NEW*.
- (4) Type in Listing 2, then verify your work with Unicheck.
- (5) Run the program (after saving a backup copy) and follow all the on-screen prompts. A data file will be written to your disk.
- (6) Load Listing 1 with the command *LOAD "EDITORLI.BAS"*.
- (7) Merge the file created by Listing 2 with the command *ENTER "D:ML.DAT"*.

- (8) Save the resultant program with the command *LIST "D:EDITORII.LST"*.

Cassette version:

- (1) Type in Listing 1 and verify your work with Unicheck.
- (2) Save the program to cassette with the command *CSAVE*. (Do not rewind the cassette.)
- (3) Clear the computer's memory with the command *NEW*.
- (4) Type in Listing 2 and verify your work with Unicheck.
- (5) Run the program and follow the on-screen prompts. A data file will be written to your cassette.
- (6) Rewind the cassette.
- (7) Load Listing 1 with the command *CLOAD*.
- (8) Merge the file created by Listing 2 with the command *ENTER "C:"*.
- (9) On a new cassette, save the resultant program with the command *LIST "C:"*.

Using the Editor

Take a look at one of the BASIC program listings in this issue. Notice that each program line is preceded by a two-letter code. This code is the checksum for that line; it's not a part of the program.

To enter a program listing from the magazine, load BASIC Editor II with the *ENTER* command, and run it. You'll be asked if you wish to allow abbreviations (see your BASIC manual). If you do, type *Y* and press *RETURN*. Otherwise, type *N*.

Note: If you set BASIC Editor II to allow abbreviations, the program will run slightly slower.

Your screen will now be divided into two "windows." The upper window will display each line after it's processed, as well as the

checksum generated for that line. The lower window is where program lines are typed and edited.

When the program's waiting for input, the cursor will appear at the left margin of the typing window. Type a program line and press *RETURN*. The line will be evaluated and reprinted in the message window, along with the checksum generated.

If the checksum matches the one in the magazine, then go on to the next program line. Otherwise, enter the command *E* (edit) and press *RETURN*. The line you just typed will appear in the typing window, where you may edit it. When you think the line has been corrected, press *RETURN*, and it'll be reevaluated.

Note: You may call up any line previously typed, with the command *E* followed by the number of the line you wish to edit. For example, *E230* will print Line 230 in the typing window. *Do not attempt to edit any program lines numbered 32600 and higher.* These lines fall within the BASIC Editor II program.

If you're using BASIC abbreviations, the two versions of the command *E* work slightly differently. The command *E*, without a line number, will call up the line exactly as you typed it. When you append the line number, the line will be printed in its expanded (unabbreviated) form.

Leaving the Editor

You may leave BASIC Editor II at any time, by entering either *B* (BASIC) or *Q* (quit). If you type *B*, the Editor will return you to BASIC. Enter *LIST* to review your work, if you wish. Note that lines 32600 and above are the Editor program. Your work will appear before these lines. To return to the Editor, type *GOTO 32600*.

Type *Q*, and you'll be asked if you really want to quit. If you type *Y*, the Editor program will be erased from memory, and you may then save your work in any manner you like. If you type *N*, the *Q* command will be aborted.

Large listings

If the program you're entering is particularly long, you may need to take a break. When you want to stop, type *Q* and press *RETURN*, then save your work to disk or cassette. When you're ready to start again, load the program you were working on, then load BASIC Editor II with the *ENTER* command. Type *GOTO 32600*, and you're back in business.

The post-processor

Many people may not want to use BASIC Editor II when entering a program listing, preferring, instead, the Atari's built-in editor. For that reason, BASIC Editor II will allow you to check and edit your programs after they've been typed.

To take advantage of this option, type any magazine program in the conventional manner, then save a copy to disk or cassette (just in case). With your typed-in program still in memory, load BASIC Editor II with the *ENTER* command, then type *GOTO 32600*.

Respond with *N* to the "abbreviations" prompt. When the Editor appears on your screen, enter the command *P* (post-process), and the first program line will appear in the typing window. Press *RETURN* to enter it into the Editor.

The line will be processed, and the checksum, along with the program line, will be printed in the upper window. If the checksum matches the one in the magazine, press *RETURN* twice, and the next line will be processed.

If you find you must edit a line, enter the command *E*, and the line will be moved back to the typing window for editing.

When the entire listing has been checked, you'll be asked if you wish to quit. Type *Y* and press *RETURN*. The Editor program will be removed from memory, and you may then save the edited program in any manner you wish.

Murphy's Law

Anyone who's been associated with computing knows this is the industry Murphy had in mind. You may find that, after typing a program with BASIC Editor II, it still won't run properly. There are two likely causes for this.

First, it may be that you're not following the program's instructions properly. Always read the article accompanying a program *before* attempting to run it. Failure to do so may present you with upsetting results.

Finally, though you can trust BASIC Editor II to catch your typos, it can't tell you if you've skipped some lines entirely. If your program won't run, make sure you've typed all of it. Missing program lines are guaranteed trouble.

One last word: Some people find it an unnecessary and nasty chore to type REM lines. I don't condone the omission of these lines, since they may be referenced within the program (a bad practice, but not unheard of). If you want to take chances, BASIC Editor II is willing to comply.

When you've finished entering a program using BASIC Editor II, you can be certain it contains no typos.

Listing 1.
BASIC listing.

```
32600 IF FL THEN 32616
32602 DIM L$(115),S$(115),C2$(2),B$(1
15),M$(119),S$(98),E$(69),A$(1):FL=1:5
TMTAB=PEEK(136)+PEEK(137)*256
32604 GRAPHICS 0:POKE 710,0:P=0:ABR=0:
? "ALLOW ABBREVIATIONS":INPUT A$:IF A
S="Y" OR A$=" " THEN ABR=1
32606 B$(1)="":B$(15)="":B$(2)=B$
32616 OPEN #17,4,0,"E":L$="" :GOSUB 3
2662:START=0
32618 POKE 766,1:POKE 83,39:POSITION 1
,3:IF LEN(L$)<39 THEN ? L$:GOTO 32624
32620 IF LEN(L$)<77 THEN ? L$(1,38):?
L$(39,LEN(L$)):GOTO 32624
32622 ? L$(1,38) ? L$(39,76) ? L$(77,L
EN(L$))
32624 POKE 752,0:POKE 766,0:POKE 559,3
4:POKE 82,1:POKE 83,38:POSITION 0,10:
" " :INPUT #17:L$:POKE 766,1
32626 IF (L$="p" OR L$="P") AND START=
0 THEN P=1:L$=""
32628 IF L$="e" OR L$="E" THEN E=1:POS
ITION 1,10: ? S$:GOTO 32624
32630 IF L$="q" OR L$="Q" THEN 32690
32632 IF L$="" AND P=1 THEN 32686
32634 IF L$="" THEN 32624
32636 IF L$="b" OR L$="B" THEN GRAPHIC
S 8: ? "TYPE 'GOTO 32600' TO CONTINUE":
END
32638 IF L$(1,1)="E" OR L$(1,1)="e" TH
EN E=1:TRAP 32624:EL=VAL(L$(2)):POSITI
ON 1,9:LIST EL:GOTO 32624
32640 S$(5)=L$:TRAP 32624:X=VAL(L$)
32642 START=1:IF P AND NOT E THEN 326
52
32644 GOSUB 32674:IF NOT ABR OR P THE
S 32652
32646 POKE 766,0: ? CHR$(125):POSITION
0,3:L=VAL(L$):LIST L: ? ? ? "CONT":L$
=B$
32648 POSITION 0,0:POKE 842,13:STOP
32650 POKE 842,12:A=USR(ADR(S$),ADR(L$
)) :L$=L$(L$,1)
32652 CHKSUM=USR(ADR(M$),ADR(L$),LEN(L
$)):CHKSUM=CHKSUM+PEEK(1542)*65536
32654 CHK=CHKSUM-(INT(CHKSUM/676)*676)
:HI=INT(CHK/26):LO=CHK-(HI*26):C2$(1)=
CHR$(HI+65):C2$(2)=CHR$(LO+65)
32656 IF NOT P OR E THEN E=0:GOSUB 32
662:IF NOT P THEN 32660
32658 POKE 83,39:POKE 752,1:FOR X=3 TO
5:POSITION 1,X: ? B$(1,38):POSITION 1
,X+7: ? B$(1,38):NEXT X:POKE 83,38
32660 POKE 766,1:POKE 83,38:POSITION 6
,7: ? C2$(1):POKE 752,0:GOTO 32618
32662 GOSUB 32702:POKE 766,0:POKE 752,
1: ? "M":POKE 82,1:DL=PEEK(560)+256*PEE
K(561)+4
32664 POKE DL-1,70:POKE DL+2,6:POKE DL
+3,112:POKE DL+4,112:POKE DL+5,112:POK
E DL+13,112:POKE DL+14,112
32666 POKE DL+22,112:POKE DL+23,112:PO
KE DL+24,65:POKE DL+25,PEEK(560):POKE
DL+26,PEEK(561):POKE 83,39
32668 POSITION 0,0: ? "Basic editor
M",M$
END M$
32670 POSITION 0,1: ? " " M$9
AGE WINDOW "":POSITION 1,7
? "CODES":
32672 POKE 559,34:RETURN
32674 GRAPHICS 0:POKE 559,0:POKE 766,1
:POKE 82,0:POKE 83,39:POSITION 0,31: L
$: ? ? ? ? "CONT":POSITION 0,0
32676 POKE 842,13:STOP
32678 POKE 842,12:TRAP 32682:A=USR(ADR
(E$),VAL(L$)):IF A=4 THEN POP I:GOTO 32
682
32680 RETURN
32682 GOSUB 32662:SOUND 0,75,10,8:FOR
X=1 TO 20:NEXT X:SOUND 0,0,0,0:POSITIO
N 1,31: ? "SYNTAX ERROR!":POKE 766,1
32684 POKE 83,38:POSITION 1,10: ? S$:G
OTO 32624
32686 LINE=PEEK(STMTAB)+PEEK(STMTAB+1
)*256:IF LINE=32599 THEN 32690
32688 OFS=PEEK(STMTAB+2):STMTAB=STMTAB
+OFS:POSITION 1,9:LIST LINE:GOTO 32624
32690 POKE 766,0:POSITION 1,10: ? "READ
Y TO QUIT":INPUT A$:IF A$<"Y" THEN P
OSITION 1,10: ? B$(1,38):GOTO 32624
32692 GRAPHICS 0: ? ? ? ? :FOR X=32600
TO 32636 STEP 2: ? NEXT X: ? "CONT":PO
SITION 0,0:POKE 842,13:STOP
32694 POKE 842,12:GRAPHICS 0: ? ? ? ? :
FOR X=32638 TO 32674 STEP 2: ? NEXT X
? ? ? "CONT":POSITION 0,0
32696 POKE 842,13:STOP
32698 POKE 842,12:GRAPHICS 0: ? ? ? ? :
FOR X=32676 TO 32702 STEP 2: ? NEXT X
? ? ? "POKE 842,12":POSITION 0,0
```

```
32700 POKE 842,13:STOP
32702 POKE 16,112:POKE 53774,112:RETUR
N
```

CHECKSUM DATA.
(see issue 39's *Unicheck*)

```
32600 DATA 6,665,923,757,809,171,225,8
98,532,499,910,267,912,144,735,8453
32638 DATA 97,358,230,693,706,878,317,
127,36,597,238,258,182,430,168,5315
32668 DATA 864,953,472,385,887,724,72,
687,908,736,625,612,672,184,891,9672
32698 DATA 8,856,85,949
```

Listing 2.
BASIC listing.

```
10 DIM L$(120),M$(119),A$(1)
20 GRAPHICS 0:POKE 710,0:"DISK OR TA
PSETTE":INPUT A$:IF A$<"C" AND A$<"
D" THEN 20
30 IF A$="C" THEN 50
40 ? "PLACE FORMATTED DISK IN DRIVE":
? "THEN PRESS RETURN":INPUT L$:OPEN #1
,0,"D":M$.DA":GOTO 60
50 ? ? "READY CASSETTE, PRESS RETURN"
:INPUT L$:OPEN #1,0,"C":
60 L$="32608 M$(1)":"L$(13)=CHR$(34)
70 M=119:GOSUB 130:L$(14)=M$(1,50):L$
=LEN(L$)+1:CHR$(34): ? M1:L$
80 L$(1)="32610 M$(59)":"L$(14)=CHR$(3
4):L$(15)=M$(59):L$(LEN(L$)+1)=CHR$(3
4): ? M1:L$
90 L$(1)="32612 S$":"L$(10)=CHR$(34)
100 M$="" :M=98:GOSUB 130:L$(11)=M$:L
$(LEN(L$)+1)=CHR$(34): ? M1:L$
110 L$(1)="32614 E$":"L$(10)=CHR$(34)
120 M$="" :M=69:GOSUB 130:L$(11)=M$:L
$(LEN(L$)+1)=CHR$(34): ? M1:L$:END
130 FOR X=1 TO M:READ A:M$(X)=CHR$(A)
:NEXT X:RETURN
140 DATA 104,104,133,204,104,133,203,1
04,104,133,205,169,0,141,3,6,141,2,6,1
41,4,6,141,5,6
150 DATA 141,6,6,238,3,6,32,60,218,172
2,6,177,203,133,212,32,170,217,32,182
,221,32,60,218
160 DATA 173,3,6,133,212,32,170,217,32
,219,218,32,210,217,165,212,141,0,6,16
5,213,141,1,6,24
170 DATA 173,0,6,109,4,6,141,4,6,173,1
,6,109,5,6,141,5,6,144,3,6,238,6,6,238,2
180 DATA 6,172,2,6,196,205,208,176,173
,4,6,133,212,173,5,6,133,213,96
190 DATA 104,104,133,204,104,133,203,1
04,104,141,255,6,169,0,133,213,216,165
,88,133,205,165,89,133,206
200 DATA 174,255,6,24,165,205,105,40,1
33,205,144,2,230,206,202,208,242,160,0
,177,205,201,64,144,18
210 DATA 201,96,144,19,201,128,144,18,
201,92,144,6,201,224,144,7,176,0,24,1
05,32,144,3,56,233
220 DATA 64,145,203,200,192,114,240,2
,208,215,177,203,201,32,208,3,136,208,2
47,200,132,212,96
230 DATA 104,104,141,254,6,104,141,253
,6,169,0,133,213,216,165,136,133,205,1
69,204,104,105,160,0,177
240 DATA 205,205,253,6,208,8,200,177,2
05,205,254,6,240,15,160,2,177,205,24,1
01,205,133,205,144,228
250 DATA 230,206,176,224,160,4,177,205
,201,55,240,4,160,0,240,0,132,212,96
```

CHECKSUM DATA.
(see issue 39's *Unicheck*)

```
10 DATA 203,265,465,844,294,973,652,27
0,978,797,278,275,835,209,301,7639
50 DATA 355,94,254,420,935,840,580,41
974,564,5435
```

End User

by Arthur Leyenberger

If you ever plan to go to Las Vegas for COMDEX, be sure to book your room early. I didn't follow this advice and as a result had to spend the first few hours in "glitter city" searching for a room. Fortunately, the gamble paid off—I got a room—but the motel had to be the worst I've ever stayed in. (Except for the night I missed my connecting flight in Houston and was put up (and out) in a real fleabag, courtesy of Continental Airlines. But that's another story.)

For months before, Atari billed Winter COMDEX 1988 as the show that would demonstrate how serious Atari was about making a showing in the U.S. computer market. There had been a lot of anticipation among users, retailers and the few still-non-jaded members of the press. It all happened in the "Gold Room"—a large, separate room, slightly off the main drag of the principal exhibit hall. The Atari room was packed from morning to night. Unfortunately, it was packed with people rather than new Atari products.

Although Atari had a big booth, they failed to announce any significant new products, claiming that they didn't want to introduce

new products before they were ready. However, some three- to four-dozen exhibitors were pleased with the crowds and the response to their products (more on that later). The highlight event of the week was a live mini-concert by Fleetwood Mac, right in the Gold Room. Of course, the band was using plenty of MIDI-equipped instruments all controlled by a Mega ST.

Stealth Laptop

It could have been called that—it was there and yet it wasn't. We got a look at it back at the hotel where it was being shown somewhat secretly. Two versions were observed: an actual-size Styrofoam mock-up (the lightest laptop I've yet seen) and a (family-size) working prototype. The prototype used a Mega motherboard and it contained one double-sided disk drive. It had a supertwist LCD screen (a backlit version is said to be available as an option), a built-in trackball controller to replace the mouse, MIDI ports and contained one megabyte of memory. Exact details were sketchy, but the laptop could conceivably contain a built-in hard disk (prob-

bly 20 megabytes) by the time it reaches production early in 1989.

Atari's target price for the laptop is \$1,500. Perhaps Atari will be the first with a "Mac Laptop," beating Apple to the finish line.

The High End

Atari's long-awaited and much-rumored 68030 machine (the "TT") was apparently not ready in time to be shown. But Sam Tramiel briefly described it at a gathering of developers. It is said to be TOS-compatible, uses Unix 5.31 and will run at 16 MHz. Insiders also say it will include a math coprocessor and hard disk. According to Atari it will be available in the second quarter of 1989.

The Abaq, now called the Atari Transputer Work Station (ATW), was seen in its latest version. It features the T800-20 transputer processor chip, which can perform 10 million instructions per second (MIPS) and includes 4K of RAM, a floating-point processor and four asynchronous communications links. Up to four additional transputer chips can be combined on an expansion board for

parallel-processing applications, and the ATW can hold four expansion boards, for a maximum total of 17 transputer processors.

In addition to the transputer chip(s), the ATW has a 68000 processor for input/output functions and the Blossom graphics processor for memory refresh and graphics acceleration. The floor-standing machine runs X-Windows, the Unix windowing system and is expected to be used for applications requiring accelerated high-resolution graphics, lots of memory and fast storage, such as CAD, animation, desktop publishing and 3-D modeling. Atari said that manufacturing will begin during the first quarter of 1989, with the first units currently scheduled to ship to Europe.

Cloning Around

Well, Atari did show a couple of new products. For one, there was an AT clone. Billed as the PC-3 (or was it the PC-4?), it used an 80286 microprocessor, had a handful of expansion slots and incorporated EGA graphics. Specifications differed depending upon which Atari spokesperson you talked to. It wasn't clear how much RAM was standard nor what capacity hard disk was available, but most AT clones typically have at least a 20-megabyte hard disk and one megabyte of RAM. Pricing and availability were equally elusive, but most accounts suggested that it would be shipped sometime around the middle of the year.

The other mystery PC was an 80386-based computer dubbed the PC-5. It too had some sort of hard disk and expansion slots. Pricing information was unavailable and delivery was expected to be "mid-year." Apparently, both the 286 and 386 PCs will first be introduced to the overseas market. That is probably a good idea, since from what little pricing information I heard about these machines, they appeared not to be competitive with the dozen or so clones that are already available.

More Atari News

Atari was also showing their new Mega File 30, a 30-megabyte hard disk for the ST line of computers. The hard disk sells for \$900, fits nicely under a monitor and plugs into the DMA port on an ST or Mega com-

puter. It has an average access time of 65 milliseconds and up to four hard disks can be daisy-chained together.

The CD-ROM player was not shown at the booth or anywhere else for that matter. Sam Tramiel told me that its release is dependent upon software becoming available for it. Atari is currently pursuing CD-ROM software development with several vendors and hopes to be able to release the product sometime in 1989.

Atari also announced that they have just concluded signing a contract with a major memory chip supplier. Atari expects that with the increased availability of DRAM chips, they will be able to solve their production problems. They also hope that with more production, they can begin to supply the U.S. market with more STs and thus regain some of that lost market. However, it will take more than just additional memory chips to strengthen the U.S. market. What is needed is a serious commitment by Atari, complete with a major advertising campaign.

Atari is also continuing its search for a U.S. manufacturing facility. With a better supply of memory chips, Atari should be able to move ahead with their plans for increased production capability. A U.S. plant seems to fit well into these plans.

In an effort to strengthen relations with developers and the entire ST-user base, Sam Tramiel has announced that it will be easy for anyone to send electronic mail to him or Sig Hartmann. There will be a direct Easyplex line via CompuServe in place by the time you read this. He also stated his desire to hold more on-line conferences with users. Definitely a good move for Atari.

Software—Desktop Publishing

Atari was showing *Ultrascript*, a Postscript emulator for the Atari Desktop-Publishing System. It's a page-description language that is used with Atari's SLM804 laser printer and Mega ST2 or Mega ST4 systems. *Ultrascript* provides a variety of fonts, gray scales and halftones. Type is treated as a raster image and can be rotated and scaled. Included fonts are the Lucida family (six styles), Helvetica, Courier, Times Roman, Palatino, Avant Garde, Bookman, New Century Schoolbook, Souvenir, Garamond, Zapf Chancery, Zapf Dingbats and Symbol.

Another software announcement from Atari was the *DeskSet II* desktop-typesetting package. This professional program uses the GEM interface, contains a text editor, an object-oriented draw package and can accept ASCII files and graphics from ST paint programs. It is designed to work with the Atari SLM804 laser printer and Mega computers. Genuine Compugraphic scalable outline fonts are used for both printing and on the screen. Features include automatic and user-definable kerning, reverse video, condensed and expanded type, linguistic-based hyphenation and text flow into arbitrarily shaped graphics regions. Suggested price is \$300.

ISD Marketing was showing still another desktop-publishing program in the Gold Room called *Calamus*. This long-awaited desktop-publishing program provides text and document processing, design tools and page-layout capabilities. It uses a full GEM interface with mouse control and drop-down menus or can be used with keyboard commands. It uses vector fonts rather than Postscript-type fonts and provides loads of features such as outline fonts, printer preview, automatic kerning and stretching. It can output in PDL, DDL, interpress and linotype formats. It sells for \$300.

SoftLogik was showing their recently completed program *PageStream*, formerly *Publishing Partner Professional*. It incorporates typographical features, word processing, page layout and graphics. It also offers capabilities such as color separation of pictures within documents and color printing. Word-processing features include spell checking, search-and-replace by attributes, automatic kerning, scalable fonts and text flow around irregularly shaped graphics. Text rotation can be performed in one-degree increments and text point-sizes range from 1/50th to 1310 points. *PageStream* sells for \$200.

Other Software Highlights

QMI was showing all of their latest wares. The GEM-based *ST-Talk Professional* offers several types of file-transfer protocols, an auto-dialer and an auto-dialer database, script language for automated sign-on and other tasks, ANSI and Vidtex graphics and a host of disk utilities. It also includes a built-in word processor, capture-while-editing, type-ahead with three lines, activity-logging and

unattended auto-answer capabilities. *ST-Talk Professional* sells for \$40.

QMI was also showing *Disk Librarian*, a program that helps you keep track of your software and data files. The program will read the contents of a selected disk and allow you to sort, search and add comments to its database. Then the program can print disk labels from the information you provided it. *Disk Librarian* sells for \$30.

David Small was there demonstrating his *Spectre 128* Macintosh emulator cartridge and software. His new company, Gadgets by Small, continues to provide Macintosh compatibility on the ST with this new \$180 product. *Spectre 128* has a faster screen redraw and file-access time compared to the *Magic Sac*, Dave's previous creation. *Spectre* can use either the Mac 128K or 64K ROMS, runs all Mac system and Finder versions and runs the major Mac software such as *HyperCard*, *Adobe Illustrator* and *Pagemaker*.

Michtron was in the Atari Gold Room announcing that GFA has dropped Michtron as a distributor for the *GFA BASIC* series of software. In its place, Michtron is now selling *Hi-Soft BASIC*, which is a complete and fast version of BASIC that is Microsoft BASIC compatible. It allows direct porting of BASIC programs to/from the Amiga and MS-DOS machines. *Hi-Soft BASIC* sells for \$160.

Regent Software was showing their database program *Regent Base II* with 4GL SQL, which uses the U.S. standard database language called Structured Query Language. The language uses English phrases to perform data insertion, modification and retrieval of information stored in the database tables. A utilities program allows you to visually create, modify or delete your database tables and you can also import or export *dBase III* files. In addition, another utility allows you to move your database onto a RAMdisk for faster access. *Regent Base II* sells for \$150.

Intersect Software Corp. is committed to the development of high-quality software for the ST computer. This was evidenced by their new program called *Revolver*. It is primarily a switcher program but also combines an assortment of utilities. *Revolver* lets you configure the ST memory into as many as eight partitions, each of a different size and total-

ly independent of each other. Even if one of the partitions crashes, it will not affect the others. When in one partition, you can "roll-out" the entire partition, including the memory contents, values stored in the hardware registers and any active programs, desk accessories and RAMdisks. When the memory image is "rolled" back in, the computer, programs and data will be restored to their previous state. *Revolver* sells for \$50 and requires one megabyte of memory.

Intersect Software was also showing two other programs. *Interlink* is their telecommunications package that features call-logging, background downloading, PC-ANSI graphics, ATASCII graphics and a built-in mini-BBS. The program also includes a type-ahead buffer, auto-dialer, built-in disk commands, user-definable function keys and more. *Masterlink* contains all of the features of *Interlink* but also has an advanced script language, context-sensitive help and remote control operation. *Interlink* sells for \$40, while *Masterlink* retails for \$60.

Several companies were showing MIDI software for the ST. Dr. T's Music Software has *Sample Maker*, a digital laboratory program for sample creation and editing. It uses the standard GEM interface, provides graphics sample editing and supports such samplers as the S-900, Emax, EPS, FZ-1, Mirage and Prophet 2000.

Another program from Dr. T's, *Phantom*, is a combination MPE software module and hardware that enables you to both read and write industry-standard SMPTE time-code formats (24, 25, 30 and 30df), song-pointer-encoded FSK and standard pulse sync. The hardware plugs into the ST serial port and has a sync-in, sync-out, MIDI-in and Aux-MIDI-out connections. Suggested price for *Phantom* is \$249.

Dr. T's other MIDI-based software is *The Copyist*. It is a publishing-quality score-editing, transcription and printing program that supports both mouse and keyboard input. Notes, text symbols and lines can be placed wherever you desire in a page and can be separately moved or deleted. It will transcribe treble, bass, alto and percussion clefs and will convert any of the supported formats to any other. *The Copyist* sells for \$399.

Hybrid Arts was demonstrating *ADAP*, a general purpose, professional, digital-audio workstation. The program is designed to provide complete control of stereo audio while retaining digital fidelity throughout the production process. *ADAP* has a variety of uses. For musicians, it can be a high-quality MIDI performance sampler; for radio stations, it can be used to help produce commercial spots; for film and video postproduction work, it can be used for effects editing, background looping, dialog fill and electronic Foley.

Hybrid Arts also has a product called *SMPTE Track*. This hardware/software package allows you to synchronize any type of audio (or video) recorder to your ST computer. The device is specifically designed to read and write SMPTE time code. Since the SMPTE time is read directly into the RS-232 port without converting to MIDI clock, it provides an extremely accurate lock to tape.

Legend Software introduced *The Final Cut*, a MIDI sequencer for the ST. It uses animated tape reels and standard tape machine controls to provide a powerful multi-track recording environment. The program features a wide variety of editing functions such as cut, paste, append, repeat, merge, transpose, etc. *The Final Cut* also provides an event editor for editing individual notes and their parameters, two time-correction methods, full MIDI synchronization and special effects options for creating echoes and fade-ins. Suggested retail price is \$90.

Midisoft was displaying their *Midisoft Studio* music software. *The Standard Edition* offers an easy-to-use method for real-time recording, playback and overdubbing as well as incorporating up to 32 polyphonic independently controlled tracks. Tape recorder-style controls are used with either a mouse or the keyboard and 70,000 notes per song can be recorded. *The Advanced Edition* has all the features of *The Standard Edition* plus 64 independent tracks, MIDI event editing, velocity scaling, automatic quantization, transpose and instrument changes for each track, MIDI thru controls and support for programmable tempo changes. *The Standard Edition* costs \$99, and *The Advanced Edition* costs \$149.

FOR OUR DISK SUBSCRIBERS

The following programs from this issue are on disk:

THE A.N.A.L.O.G. #71 DISKETTE CONTAINS 18 MAGAZINE FILES. THEY ARE LISTED BELOW.

SIDE 1:

FILENAME.EXT	LANG.	LOAD	COMMENTS
KRAZYMZE.OBJ	ML	(#3)	KRAZY MAZES
KRAZYMZE.M65	MAC/65	LOAD	KRAZY MAZES SOURCE
PIXELAVG.BAS	BASIC	LOAD	PIXEL AVERAGING
PIXELAVG.M65	MAC/65	LOAD	PIXEL AVER. SOURCE
UNIVERT .BAS	BASIC	LOAD	UNIVERT
UNIVERT .M65	MAC/65	LOAD	UNIVERT SOURCE
GDW1 .LST	BASIC	ENTER	GAME DESIGN
GDW2 .LST	BASIC	ENTER	GAME DESIGN
GDW3 .LST	BASIC	ENTER	GAME DESIGN
GDW4 .LST	BASIC	ENTER	GAME DESIGN
GDW5 .LST	BASIC	ENTER	GAME DESIGN
GDW6 .LST	BASIC	ENTER	GAME DESIGN
GDW7 .LST	BASIC	ENTER	GAME DESIGN
GDW8 .LST	BASIC	ENTER	GAME DESIGN
GDW9 .LST	BASIC	ENTER	GAME DESIGN
GDW10 .LST	BASIC	ENTER	GAME DESIGN
MEDITOR.BAS	BASIC	M/L	EDITOR
EDITORII.BAS	BASIC	BASIC	EDITOR II

TO LOAD YOUR A.N.A.L.O.G. DISK

- 1) INSERT BASIC CARTRIDGE (NOT REQUIRED FOR XL OR XE COMPUTERS)
- 2) TURN ON DISK DRIVE AND MONITOR
- 3) INSERT DISK IN DRIVE
- 4) TURN ON COMPUTER (XL AND XE OWNERS DO NOT HOLD DOWN OPTION KEY!)

WARNING: BEFORE YOU RUN A PROGRAM, READ THE APPROPRIATE ARTICLE IN THE MAGAZINE.

NOTE: ONLY PROGRAMS WITH THE ".BAS" OR ".OBJ" EXTENSION MAY BE RUN FROM THE MENU. OTHER PROGRAMS SHOULD BE LOADED AS INSTRUCTED IN THE LOADING NOTES AND MAY REQUIRE ADDITIONAL SOFTWARE AS LISTED BELOW. HOWEVER, YOU SHOULD NOT ASSUME THAT EVERY FILE WITH THE PROPER FILE EXTENSION WILL RUN FROM THE MENU. YOU MAY HAVE TO MOVE CERTAIN PROGRAMS TO A DIFFERENT DISK TO OBTAIN CORRECT RESULTS.

EXT	DESCRIPTION
.M65	REQUIRES THE OSS MAC/65 ASSEMBLER
.AMA	REQUIRES THE ATARI MACRO ASSEMBLER
.ASM	REQUIRES THE ATARI ASSEMBLER/EDITOR
.ACT	REQUIRES THE OSS ACTION! CARTRIDGE
.LGO	REQUIRES THE ATARI LOGO CARTRIDGE
.SYN	REQUIRES THE SYNAPSE SYN ASSEMBLER
.STB	REQUIRES ST BASIC

LOADING NOTES

LOAD BASIC PROGRAM:	LOAD "D:FILENAME.EXT"
ENTER BASIC PROGRAM:	ENTER "D:FILENAME.EXT"
LOAD MAC/65 PROGRAM:	LOAD #D:FILENAME.EXT
ENTER ASM/ED PROGRAM:	ENTER #D:FILENAME.EXT
LOAD LOGO PROGRAM:	LOAD "D:FILENAME.EXT"
LOAD SYN/AS PROGRAM:	LOAD "D:FILENAME.EXT"

- #1: SEE ACTION! MANUAL.
- #2: SEE ATARI MACRO ASSEMBLER MANUAL.
- #3: MAY ALSO BE LOADED FROM DOS USING THE "L" OPTION OF THE DOS MENU.
- #4: THIS FILE SHOULD BE TRANSFERRED TO ANOTHER DISK AND RENAMED "AUTORUN.SYS".
- #5: SEE ST BASIC MANUAL.

Passport Design's *Master Tracks Pro* is a sophisticated MIDI recording and editing package. Its five main modules provide 64 tracks of real-time and step-time input, graphical song editing, graphical step editing, a system-exclusive librarian and keyboard-control mapper. It also reads and writes standard MIDI files and sells for \$350. *Master Tracks Jr.* is a low-cost personal MIDI recording studio. Used with MIDI instruments, it provides a composing environment for recording your own songs. *Master Tracks Jr.* contains tools for composing, recording and editing music with a graphical interface. The program is fully compatible with *Master Tracks Pro* files and industry-standard MIDI files and sells for \$130.

Masterscore from Russ Jones Software is a scoring program for the ST that converts PRO-24 files to notation. It provides the capa-

bility for up to 48 staves, transposing of single sections or entire voices, naming staves, ritardandos/accelerandos, repeats, first and second endings, crescendos/decrescendos and editing. The program supports MIDI song files, can print on dot matrix or laser printers and sells for \$350.

Pro-24 III is a MIDI sequencer with a note capacity of 150,000 events or 999 measures. It can contain either 24 polyphonic or 32 monophonic tracks and use linear or song-mode sequencing. It also features a drum-machine mode and real-time graphic editing via note-grid, music score or rhythm editor. *Pro-24 III* sells for \$295.

By the Way

COMDEX is a huge show, no question about it. Here are some facts and figures that describe the event, compiled by The Inter-

face Group, the show's promoters.

One thousand workers were hired to set up the exhibits, which took six days or 65,000 staff-hours. It took about four full days to tear it down. Some 1,600 trucks traveled to Las Vegas carrying six million pounds of exhibits, which were spread over eight exhibit areas.

Attendees numbered about 100,000, used some 60,000 hotel rooms (I don't know where the rest of them slept) and required the services of 650 taxis over the course of the four-day event. There were over 1,000 temporary workers hired to help the exhibitors, about 1,000 police and security people on duty and more than 1,300 telephone lines installed.

These facts are incredible. So is the estimated revenue to Las Vegas—\$138 million. COMDEX is something else. So is Las Vegas. I wonder how I survived it?

Catch you next time.

continued from page 41
other DELPHI members by E-mail.

Replying by E-mail

If you wish to reply to a Forum message in private, you can send an E-mail reply from the FORUM prompt by typing REP MAIL. An E-mail message is automatically addressed for you, along with a subject header pertaining to the Forum message. All you have to do is type your reply, and then enter Control-Z to send the message. It's all done without leaving the Forum.

Sending Copies of Forum Messages

You may sometimes wish to share Forum messages with others on DELPHI (or send a copy to yourself for later reference). You can do this by typing FOR followed by the membername(s) to which you wish the copy sent at the FORUM prompt.

If you wished to send a copy of a message to KZIN, you would read the message in question, then type FOR KZIN. You would then be prompted for a subject for the E-mail message; enter it and press Return, and the message is on its way—again without your having to leave the Forum.

Going to Mail

If you find that you need to go to Mail to read messages or whatever, simply type MAIL at the FORUM prompt, and you'll move to DELPHI's E-mail system. After you take care of your business in Mail, type EXIT or enter Control-Z, and you'll be returned to the FORUM prompt. The last message you read will remain the current message, and any commands you've implemented (such as FOLLOW) will remain in effect.

Filing Messages

If you'd like a copy of a message for later download, editing, sending to others as E-

mail messages, or for whatever reason, you can place one in your personal Workspace. Type FILE followed by a filename after you read a message, and the system will create a new file in your Workspace that contains the Forum message.

Getting Help

I've explained the more important Forum commands and features in detail here, but you may still need help while in the Forum, on occasion. When you need help, first type ? to see the Forum menu; this command list will most likely answer your question. For more detailed help with any command, type HELP followed by the name of the command (Example: HELP DIRECTORY).

You can also type HELP or /HELP during any operation (such as editing) for more detailed information.

Utilities

Two utilities help you manage message access: High Message and Set Topics.

High Message

Your high message is the highest number message (also the most recent) you have read in the Forum. High Message is a simple utility that displays the highest message you've read and allows you to reset the high message number.

You may wish to check this number while you're in the Forum to see just where you're at. You may wish to reset it so that no messages before that number are presented as "new" messages. For either purpose, type HIGH at the FORUM prompt. You'll see this:

```
High Message on Entry: 38401
Current High Message : 38401
New Value (or RETURN):
```

If you press Return, the message number will be unchanged. If you type in a new number, the system will consider any message with a number higher than the number you

entered as unread, whether you've read it or not.

(Note: You can set your high message number to the highest message in the Forum by typing 99999 at the FORUM prompt. This is useful, too, if you don't feel like reading all new Forum messages and simply wish to "clear" unread messages.)

Set Topics

In addition to using DIR and READ with qualifiers to select specific messages, you can further limit the messages you'll see, by topic. For example, if you aren't a game player, you won't want to see messages under the "Games & Entertainment" topic.

You can render messages in that topic (or any other topic or topics) effectively invisible to you with the Forum's SET TOPICS command. (Topic access can be reinstated if you change your mind.)

To select or de-select the appropriate topics, type TOPICS and follow the prompts. (Topics available in the Atari SIG are General Interests, Games & Entertainment, Telecommunications, Utilities, Sight & Sound, Education, Recent Arrivals, Reviews & News, Koala Pictures, Current Issue, and Home Use.)

Weekly Conferences

The weekly Atari SIG conferences are still rolling. Drop in and meet your fellow Atari users and SIG managers any Tuesday at 10 P.M., EST. You'll find the conferences an excellent venue for sharing information about Atari computers, getting answers to questions and participating in friendly discussions of all types.

In addition to science fiction novels and books on model rocketry and other topics, Michael A. Banks is the author of DELPHI: The Official Guide and The Modem Reference, both from Brady Books. You can write to him via E-mail on DELPHI to member-name KZIN.

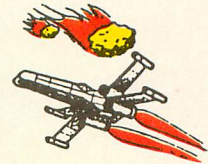
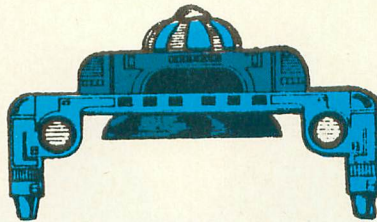
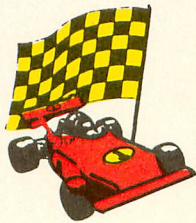


Cheat!



Alpha Systems
1012 Skyland Drive
Macedonia, OH 44056
(216) 467-5665
\$24.95

Reviewed by Clayton Walnum



Have you ever seen the higher levels of *Karateka* or made it through every screen of *Jumpman*? Have you ever attained a score of 999,999 in *Shamus: Case II*? Unless you're a dedicated computer gamer who spends every free moment of his time twisting a joystick, the answer to these questions—and many other similar ones—is probably “no.” Now, thanks to *Cheat!* from Alpha Systems, you can be a video-game hotshot every time you boot up.

Cheat! allows you to modify over one hundred games so that you have unlimited lives. What's more, the process is painless and simple. Just boot the *Cheat!* program disk, select the Cheat option, choose a game from the menu and then insert the game disk. That's all there is to it. *Cheat!* will modify the disk with no further input.

The program works by scanning each sector of the game disk, looking for the section of code that handles the life-count portion of the game. When it locates that code, it writes a new code to the disk that stops the program

from decrementing the number of remaining lives whenever you get “killed.” Although the process is simple, the disk scanning can take several minutes.

Also included on the *Cheat!* disk is a program called *Uncheat!* The more astute among you will have already guessed that this program takes a disk that's been modified by *Cheat!* and returns it to its original condition.



The program's manual is short, containing only three pages of instructions; but, because *Cheat!* is so simple to run, a larger manual is unnecessary. It would have been nice, however, if the manual included a list of the games that *Cheat!* can modify. You must boot the program to get this list.

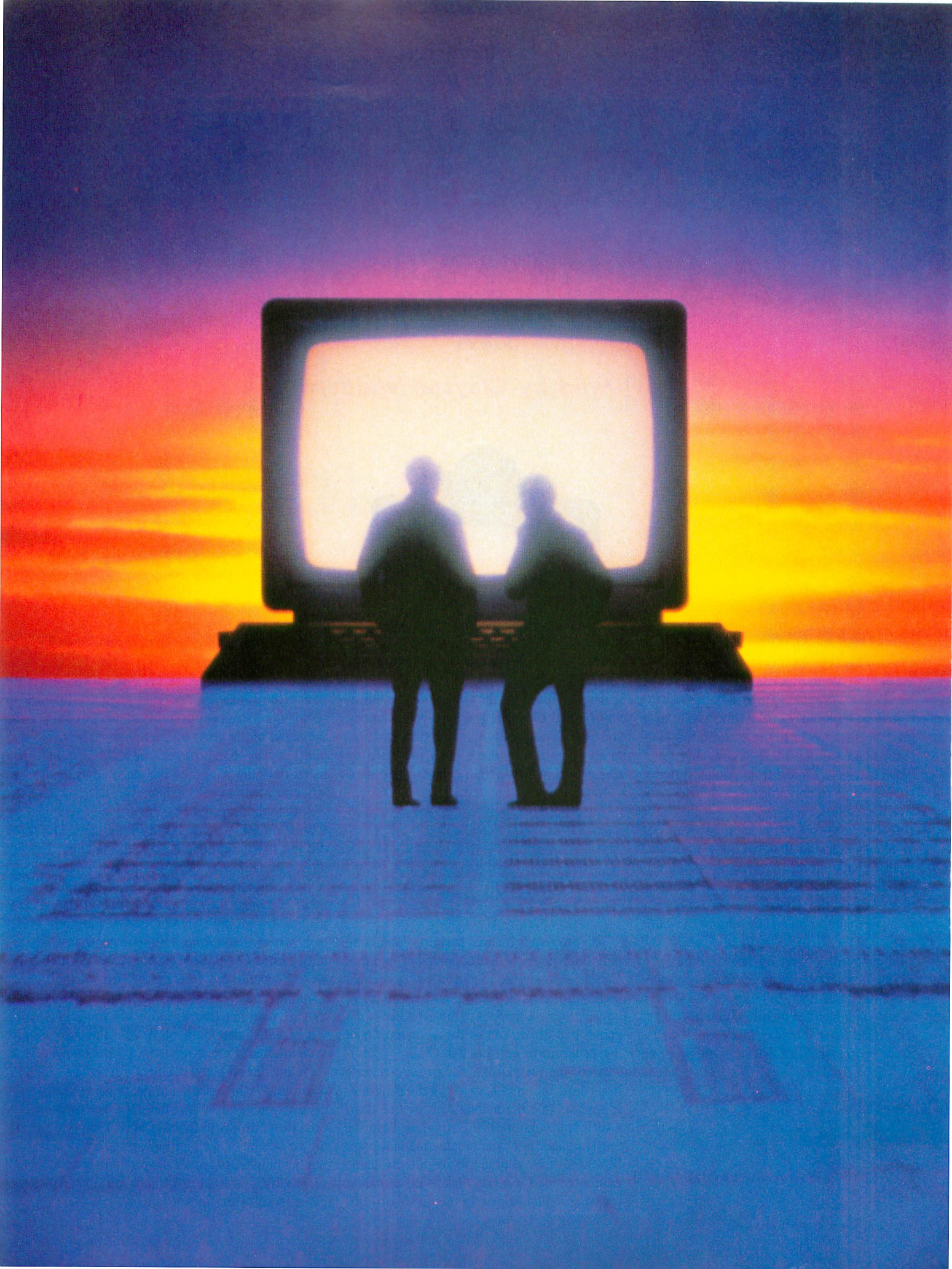
Also, the manual is not entirely accurate in its description of how to run the program. It appears that the manual was intended for a different version of the program than the one supplied. For example, the manual states “Press ‘Z’ for a directory of all the game names on the *Cheat!* disk.” In actuality, pressing Z just brings up the disk's directo-

ry, not a listing of the games (which is already shown on the screen, anyway). Also according to the manual, the program offers you the option of entering the disk sector on which the “lose-life” portion begins. This sector number is supposedly shown next to the name of the program on the menu but it didn't appear on my copy of *Cheat!* The menu listed only the names of the games and the letter to press to select them.

Although it was not possible for me to test *Cheat!* on every game listed in the menu, I did test it with quite a few, including *Jumpman*, *Zeppelin*, *Repton*, *Cyclod*, *Caverns of Mars*, *Ghost Encounters*, *Chicken*, *Protector II*, *Shamus: Case II*, *Boulderdash*, *Gateway to Apshei* and ANALOG Computing's own *Livewire* and *Roto*. Of those tested, the only games *Cheat!* didn't seem to work with were *Zeppelin* and *Caverns of Mars*. After scanning these disks, *Cheat!* informed me that it “couldn't find the lose-life routine.” It's possible that both of these games are available in different versions. I don't know.

At any rate, if you'd like to attain some extraordinarily high scores or get to some levels that you've never seen before, *Cheat!* may be a program worth looking into. The price tag's a wee bit on the high side, but, considering the number of games the program is able to modify, it's a worthwhile addition to a gamer's software library.





Let's say you're mixing some garden spray and it calls for so many tablespoons per gallon, but you have a measuring device that's marked in fluid ounces. Or let's say you have an old seafaring friend who thinks in leagues, and you're trying to tell him how far a light year is. These two examples are rather different, but both involve conversions between systems of units. *Univert* can handle both, and many more.

There have been a number of conversion programs published for the Atari, including one which appeared way back in ANALOG 14. Typically, these convert between English and metric units and have a limited selection of possible conversions. Try, for example, to convert meters to furlongs, or, to take an extreme, cubic microns to cubic miles. Furthermore, these programs usually require the user to make choices from menus of conversion options printed on the screen.

Univert is different. It can convert between any units of length, area, volume, weight (mass), angle or time. If you can come up with some new units *Univert* hasn't heard of, they can be easily added. And *Univert* is smart. You merely type in the number you want to convert followed by the units, either spelled out or abbreviated, then the units you want to convert to, and *Univert* gives you the answer. If you misspell the units or give *Univert* something it doesn't recognize, it tells

you. And if, like us, you're often not sure about spelling, there's a Help option that lists the units available and their standard abbreviations.

Running the Program

Running *Univert* is straightforward. Once you have typed in Listing 1 and saved it (and don't forget to check your typing with *BASIC Editor II*), simply type RUN. *Univert* takes about ten seconds to load the tables it uses plus a machine-language subroutine, then prompts you for the next value to be converted. Type in the number of "whatevers" you want to convert from, followed by the units and a return. Put a space between the number and the units, and between each word of the units if more than one word. For example, to convert 27.5 square feet type "27.5 SQUARE FEET". You can use standard abbreviations, too, (no periods, please) so you could also type "27.5 SQ FT".

The number can be any valid BASIC numeric, so for very large or small numbers you can use scientific notation, such as 3.274E-6.

After the value to be converted from is entered, *Univert* asks you for the units to be converted to. The same rules for units apply. Then *Univert* gives you the answer and is ready for the next conversion. Use System Reset or Break to exit the program.

Univert

The Universal Conversion Program

by William Frasz and Reid Brockway

If at any time you are not certain of the units or abbreviations to use, type HELP instead of a "to" or "from" entry. *Univert* will ask you for the first letter of the units you want and will list all the ones it knows that begin with that letter.

How It Works

There are two principal features of *Univert* that give it its speed and versatility. The speed comes from a machine-language search routine, and the versatility comes from a pair of tables containing units character strings and conversion factors.

The search routine is represented by the DATA statements at lines 1000-1030. (An assembly listing is included in Listing 2.) It is loaded into memory page six and called via the BASIC USR statements at lines 590 and 660. When *Univert* initializes, it builds a long string variable, TABLE\$, by reading the units listed starting at line 5000 and inserting a comma between each one to define substrings. The search routine performs a random-record-length search of TABLE\$ using the commas as delimiters and returns an index to the substring that matches the units typed.

The index returned by the search routine is used by BASIC to fetch two values: a conversion factor and a "type" code from another table, FACTOR. This table is a two-dimensional array formed by reading the DATA statements at line 6000 and following. The type code tells *Univert* which of the six types of conversions mentioned above is to be performed. The codes are defined as follows:

- 1 = lineal
- 2 = area
- 3 = volume
- 4 = weight (mass)
- 5 = time
- 6 = angle

When valid "from" units are entered by the user, the type code from the FACTOR table is saved, and the number you entered is multiplied by the conversion factor to convert it to a value, FRNUM, in a standard intermediate set of units. This is a key feature of *Univert's* design. The intermediate units are (in order by type) meters, square meters, cubic meters, grams, seconds and degrees. Then, when the "to" units are entered, FACTOR is again accessed, and the type code is checked against the saved type code. If it does not match, you are told you have requested a conversion between incompatible units and are asked again for the "to" units. Otherwise FRNUM is divided by the conversion factor

associated with the "to" units to obtain the result.

Neat, huh? This means *Univert* can handle any new units you may want to add so long as they are one of the six types. You just add the units string as a DATA statement at the next available line in the 5000's and the type code and conversion factor (to give meters, grams, etc.) as a DATA statement at the corresponding line in the 6000's. Also, be sure to change the value of COUNT at line 100 to equal the right three digits of both line numbers.

A final note on accuracy: The conversion factors in many cases are not exact. To keep the typing task manageable, we limited the number of significant digits to seven. This should be sufficient for most purposes, but you can add more digits if you feel like it. (Atari BASIC supports nine or ten significant digits.) This non-exactness, combined with the mechanism of converting through an intermediate value, does sometimes result in an

answer that may seem strange. For example, the conversion from 1 POUND to OUNCES (via grams) gives an answer which is very nearly, but not quite, the exact answer, which is 16. This is the small price you pay for the universality of *Univert*.

Now put away your book of look-up tables, your hand calculator and even your dictionary. The next time someone asks you how many decimeters there are in a cubit, just fire up your Atari and ask *Univert*.

William Frasz is a regional service manager for Fujitsu, Inc., installing and repairing telefax machines and other electronic equipment. William uses a 130XE to maintain service records of the equipment he services. Reid Brockway is a software and systems engineer for Intermetrics, Inc. Reid designs real-time software for aircraft and space applications, as well as doing general design of computer-based systems.

LISTING 1: BASIC

```

HY 10 REM *****
II 11 REM *          UNIVERT          *
DY 12 REM * by W. Frasz and R. Brockway*
ZN 13 REM *
VT 14 REM *          COPYRIGHT 1989    *
VD 15 REM *          BY ANALOG COMPUTING *
IK 16 REM *****
OE 50 GRAPHICS 0:POKE 709,12:POKE 710,48:
DL=PEEK(560)+256*PEEK(561)
DJ 60 POKE DL+3,71:POKE DL+6,6:POKE DL+27
,6:POKE DL+28,65:POKE DL+29,PEEK(560):
POKE DL+30,PEEK(561)
CA 70 POSITION 6,0:? "univert":POSITION 3
,5:? "THE UNIVERSAL CONVERSION PROGRA
M"
FS 80 POSITION 5,13:? "brought to you by"
:POSITION 15,15:? "Biff & The Bitfreak
"
XF 90 POSITION 2,21:? "(LOADING TABLES)"
OF 100 COUNT=144:PARSE=400:WORK=500:HELP=
200:WAIT=800
US 110 DIM BLANK$(40):BLANK$=""
":REM (39
SPACES)
FF 120 DIM TABLE$(2000),TEMP$(30),FRUNITS
$(20),TOUNITS$(20),FACTOR(COUNT,1),A$(
1)
BR 130 RESTORE 1000:FOR I=0 TO 104:READ X
:POKE 1536+I,X:NEXT I
WR 140 RESTORE 5000
JN 150 FOR I=0 TO COUNT:READ TEMP$
GY 160 LET TABLE$(LEN(TABLE$)+1)=TEMP$:LE
T TABLE$(LEN(TABLE$)+1)=", "
GF 170 NEXT I
HB 180 RESTORE 6000:FOR I=0 TO COUNT:READ
A,B:FACTOR(I,0)=A:FACTOR(I,1)=B:NEXT
I
JX 190 GOTO WORK:REM (Don't forget your 1
unch.)
BH 200 ? "K":POSITION 6,0:? "univert":POS
ITION 23,0:? "*** HELP ***":?
GQ 210 POSITION 3,1:? "Enter first letter
of units: ";
ZD 220 INPUT #16,A$
RL 230 POSITION 4,21:? "(Searching)"
PR 240 I=0:J=0:R=2:C=1
QL 250 I=I+1
XZ 260 IF TABLE$(I,I)=A$ THEN 310
QP 270 I=I+1

```

```

CK 280 IF TABLE$(I,I)<>"," THEN 270
VK 290 J=J+1:IF J=COUNT THEN POSITION 4,2
1:? BLANK$:RETURN
NV 300 GOTO 250
DQ 310 TEMP$=""
EI 320 TEMP$(LEN(TEMP$)+1)=TABLE$(I,I):I=
I+1
XG 330 IF TABLE$(I,I)<>"," THEN 320
PU 340 POSITION C,R:? TEMP$
GS 350 IF C=1 THEN C=20:GOTO 290
MK 360 C=1:R=R+1:GOTO 290
KT 400 REM *** "FROM" STRING PARSE SUBROU
TINE ***
FL 410 FLAG=0:I=1:L=LEN(TEMP$)
AC 420 IF L=0 THEN 480
JI 430 IF I=L THEN 470
MK 440 IF TEMP$(I,I)<>" " THEN I=I+1:GOTO
430
SI 450 TRAP 480:FRNUM=VAL(TEMP$(I,I))
GL 460 FRUNITS$=TEMP$(I+1,L):RETURN
WN 470 IF TEMP$="HELP" THEN FLAG=2:RETURN
DS 480 FLAG=1:RETURN
KR 500 REM ** START OF WORK SCREEN ***
HC 510 ? "K":POSITION 6,0:? "univert":POS
ITION 1,1:? "Instructions:"
FL 520 ? "Values to be converted should b
e":? "entered in the form":POSITION 6
,5:? "number followed by units"
BG 530 ? :? "For example, 27.5 square fee
t":?
WE 540 ? "If uncertain of units, type HEL
P":? :?
HT 550 POSITION 1,12:POKE 702,64:? "Enter
next value to be converted":? ;
NP 560 INPUT #16,TEMP$:GOSUB PARSE
MV 570 POSITION 4,19:IF FLAG=1 THEN ? "Nu
mber SPACE then units, please.":GOSUB
WAIT:GOTO WORK
MQ 580 IF FLAG=2 THEN GOSUB HELP:GOSUB WA
IT:GOTO WORK
VI 590 INDEX=USR(1536,COUNT+1,ADR(TABLE$)
,ADR(FRUNITS$),LEN(FRUNITS$))
KN 600 IF INDEX<>0 THEN 620
BO 610 POSITION 2,19:? "I don't recognize
";FRUNITS$:GOSUB WAIT:GOTO WORK
LO 620 TYPE=FACTOR(INDEX-1,0)
AD 630 TEMP=FRNUM*FACTOR(INDEX-1,1)
ND 640 POSITION 1,15:? "Enter units desir
ed":? ;:INPUT #16,TOUNITS$
UH 650 IF TOUNITS$="HELP" THEN GOSUB HELP
:GOSUB WAIT:? "K":POSITION 6,0:? "univ
ert":GOTO 640
MB 660 INDEX=USR(1536,COUNT+1,ADR(TABLE$)
,ADR(TOUNITS$),LEN(TOUNITS$))
KD 670 IF INDEX<>0 THEN 700
HC 680 POSITION 2,19:? "I don't recognize
";TOUNITS$:GOSUB WAIT
QG 690 FOR R=15 TO 21:POSITION 0,R:? BLAN
K$:NEXT R:GOTO 640
GR 700 IF FACTOR(INDEX-1,0)=TYPE THEN 740
WU 710 POSITION 2,19:? "You can't convert
";FRUNITS$;
RR 720 IF LEN(FRUNITS$)+LEN(TOUNITS$)>16
THEN ?
RL 730 ? " to ";TOUNITS$:GOSUB WAIT:GOTO
WORK
OH 740 RESULT=TEMP/FACTOR(INDEX-1,1)
LL 750 ? :? FRNUM;" ";FRUNITS$;
LL 760 IF LEN(STR$(FRNUM))+LEN(FRUNITS$)+
LEN(STR$(RESULT))+LEN(TOUNITS$)>33 THE
N ?
VQ 770 ? " = ";RESULT;" ";TOUNITS$:? :?
XE 780 GOSUB WAIT:GOTO WORK
CF 800 REM *** WAIT FOR KEYSTROKE ***
XE 810 POSITION 3,21:? "PRESS ANY KEY"
HE 820 IF PEEK(764)=255 THEN 820
XO 830 POKE 764,255:RETURN
GC 1000 DATA 104,104,133,213,104,133,212,
104,133,204,104,133,203,104,133,206,10
4,133,205,104,104,133,208,169,1
IY 1010 DATA 133,207,160,0,177,205,209,20
3,208,11,200,177,203,201,44,240,38,196
,208,48,239,165,207,197,212,208,5
XB 1020 DATA 169,0,133,212,96,200,177,203
,201,44,208,249,200,152,24,101,203,133
,203,144,2,230,204,230,207,24,144
IE 1030 DATA 203,196,208,240,16,200,196,2
08,208,7,136,177,205,201,83,240,4,160,
0,240,202,165,207,133,212,96

```

```

VG 5000 DATA ACRE
YK 5001 DATA ANGSTROM
OZ 5002 DATA BUSHEL
PY 5003 DATA CARAT
WO 5004 DATA CC
KO 5005 DATA CENTIGRAM
AH 5006 DATA CENTILITER
YL 5007 DATA CENTIMETER
BI 5008 DATA CL
BY 5009 DATA CM
BF 5010 DATA CORD
XA 5011 DATA CU CM
CV 5012 DATA CU FT
BD 5013 DATA CU IN
BU 5014 DATA CU KM
KP 5015 DATA CU M
AW 5016 DATA CU MI
DI 5017 DATA CU MM
ET 5018 DATA CU YD
GQ 5019 DATA CUBIC CENTIMETER
NT 5020 DATA CUBIC DECAMETER
UB 5021 DATA CUBIC DECIMETER
GA 5022 DATA CUBIC FEET
UK 5023 DATA CUBIC FOOT
DV 5024 DATA CUBIC INCH
ZY 5025 DATA CUBIC INCHES
NJ 5026 DATA CUBIC KILOMETER
ZA 5027 DATA CUBIC METER
LC 5028 DATA CUBIC MICRON
IA 5029 DATA CUBIC MILE
HE 5030 DATA CUBIC MILLIMETER
MV 5031 DATA CUBIC YARD
UR 5032 DATA CUBIT
TB 5033 DATA CUP
OX 5034 DATA DAY
UD 5035 DATA DECAGRAM
GL 5036 DATA DECALITER
ES 5037 DATA DECAMETER
YK 5038 DATA DECIGRAM
LF 5039 DATA DECLITER
IB 5040 DATA DECIMETER
DV 5041 DATA DEGREE
OU 5042 DATA FATHOM
AU 5043 DATA FEET
IX 5044 DATA FL OZ
CC 5045 DATA FLUID OUNCE
KW 5046 DATA FOOT
AC 5047 DATA FORTNIGHT
GT 5048 DATA FT
UH 5049 DATA FURLONG
TH 5050 DATA G
JL 5051 DATA GAL
OC 5052 DATA GALLON
FP 5053 DATA GR
BS 5054 DATA GRAM
ED 5055 DATA HECTARE
LJ 5056 DATA HECTOGRAM
BC 5057 DATA HECTOLITER
ZG 5058 DATA HECTOMETER
OF 5059 DATA HOUR
FR 5060 DATA HR
EL 5061 DATA HUNDREDWEIGHT
YH 5062 DATA HWT
CS 5063 DATA IMPERIAL GALLON
EI 5064 DATA IMPERIAL QUART
FA 5065 DATA IN
ZL 5066 DATA INCH
MM 5067 DATA INCHES
DC 5068 DATA KG
LY 5069 DATA KILOGRAM
WU 5070 DATA KILOLITER
VC 5071 DATA KILOMETER
FB 5072 DATA KM
WC 5073 DATA L
AS 5074 DATA LB
IM 5075 DATA LEAGUE
TU 5076 DATA LIGHTYEAR
AW 5077 DATA LITER
XH 5078 DATA M
ZT 5079 DATA METER
DW 5080 DATA MI
NV 5081 DATA MICROGRAM
DO 5082 DATA MICROLITER
TO 5083 DATA MICRON
AY 5084 DATA MICROSECOND
BU 5085 DATA MILE
MK 5086 DATA MILLIGRAM
CD 5087 DATA MILLILITER
AH 5088 DATA MILLIMETER

```

ZN 5089 DATA MILLISECOND
 QZ 5090 DATA MIN
 XU 5091 DATA MINUTE
 FR 5092 DATA ML
 GH 5093 DATA MM
 RA 5094 DATA NANOSECOND
 OP 5095 DATA NAUTICAL MILE
 WE 5096 DATA OUNCE
 NT 5097 DATA OZ
 AN 5098 DATA PECK
 TL 5099 DATA PENNYWEIGHT
 OM 5100 DATA PICOSECOND
 KR 5101 DATA PINT
 BC 5102 DATA POUND
 JV 5103 DATA PT
 BJ 5104 DATA PWT
 KO 5105 DATA QT
 GH 5106 DATA QUART
 FI 5107 DATA RADIAN
 RD 5108 DATA ROD
 MP 5109 DATA SEC
 KS 5110 DATA SECOND
 QL 5111 DATA SECTION
 CE 5112 DATA SQ CM
 HZ 5113 DATA SQ FT
 GH 5114 DATA SQ IN
 GY 5115 DATA SQ KM
 PT 5116 DATA SQ M
 GA 5117 DATA SQ MI
 IM 5118 DATA SQ MM
 JX 5119 DATA SQ YD
 SU 5120 DATA SQUARE CENTIMETER
 XL 5121 DATA SQUARE DECAMETER
 EB 5122 DATA SQUARE DECIMETER
 BY 5123 DATA SQUARE FEET
 RC 5124 DATA SQUARE FOOT
 ZR 5125 DATA SQUARE INCH
 BQ 5126 DATA SQUARE INCHES
 YJ 5127 DATA SQUARE KILOMETER
 YJ 5128 DATA SQUARE METER
 NI 5129 DATA SQUARE MICRON
 CQ 5130 DATA SQUARE MILE
 UY 5131 DATA SQUARE MILLIMETER
 JF 5132 DATA SQUARE YARD
 VJ 5133 DATA STATUTE MILE
 CX 5134 DATA STONE
 CY 5135 DATA TABLESPOON
 CW 5136 DATA TBLSP
 KT 5137 DATA TBSP
 XI 5138 DATA TEASPOON
 XM 5139 DATA TON
 EA 5140 DATA TOWNSHIP
 ZF 5141 DATA TSP
 EU 5142 DATA YARD
 GS 5143 DATA YD
 FZ 5144 DATA YEAR
 GX 6000 DATA 2,40468564E3
 DY 6001 DATA 1,1.E-10
 TQ 6002 DATA 3,3.52383E-2
 JF 6003 DATA 4,200E-3
 BH 6004 DATA 3,1.E-6
 EX 6005 DATA 4, .01
 AY 6006 DATA 3,1.E-5
 DY 6007 DATA 1, .01
 EY 6008 DATA 3, .01
 EG 6009 DATA 1, .01
 JI 6010 DATA 3,3.624576
 AY 6011 DATA 3,1.E-6
 CA 6012 DATA 3,2.8317E-2
 PK 6013 DATA 3,1.6387064E-5
 ZM 6014 DATA 3,1.E9
 DZ 6015 DATA 3,1
 HU 6016 DATA 3,4.168182E9
 DV 6017 DATA 3,1.E-9
 LT 6018 DATA 3, .76456
 CE 6019 DATA 3,1.E-6
 VH 6020 DATA 3,1.E3
 ZC 6021 DATA 3,1.E-3
 QT 6022 DATA 3, .028317
 QX 6023 DATA 3, .028317
 TG 6024 DATA 3,1.638706E-5
 TK 6025 DATA 3,1.638706E-5
 ZX 6026 DATA 3,1.E9
 EK 6027 DATA 3,1
 LQ 6028 DATA 3,1.E-18
 IJ 6029 DATA 3,4.168182E9
 CZ 6030 DATA 3,1.E-9
 KX 6031 DATA 3, .76456
 WN 6032 DATA 1, .4572

GH 6033 DATA 3,2.3658E-4
 RJ 6034 DATA 5,8.6400E4
 EM 6035 DATA 4,10
 EZ 6036 DATA 3, .01
 DN 6037 DATA 1,10
 DZ 6038 DATA 4, .1
 BC 6039 DATA 3,1.E-4
 BP 6040 DATA 1, .1
 EZ 6041 DATA 6,1
 KG 6042 DATA 1,1.8288
 UL 6043 DATA 1, .3048
 WI 6044 DATA 3,2.957373E-5
 WM 6045 DATA 3,2.957373E-5
 VX 6046 DATA 1, .3048
 VG 6047 DATA 5,1.2096E6
 WF 6048 DATA 1, .3048
 PJ 6049 DATA 1,201.168
 EC 6050 DATA 4,1
 GR 6051 DATA 3,3.7854E-3
 GV 6052 DATA 3,3.7854E-3
 EO 6053 DATA 4,1
 ES 6054 DATA 4,1
 WP 6055 DATA 2,1.E4
 GO 6056 DATA 4,100
 DQ 6057 DATA 3, .1
 FP 6058 DATA 1,100
 PF 6059 DATA 5,3600
 NY 6060 DATA 5,3600
 EZ 6061 DATA 4,4.535924E4
 FD 6062 DATA 4,4.535924E4
 OH 6063 DATA 3,4.546E-3
 BT 6064 DATA 3,1.1366E-3
 TN 6065 DATA 1, .0254
 TR 6066 DATA 1, .0254
 TV 6067 DATA 1, .0254
 KN 6068 DATA 4,1000
 KR 6069 DATA 4,1000
 DX 6070 DATA 3,1
 IH 6071 DATA 1,1000
 IL 6072 DATA 1,1000
 ZZ 6073 DATA 3,1.E-3
 BA 6074 DATA 4,453.59237
 RP 6075 DATA 1,5.55977E3
 PQ 6076 DATA 1,9.4597E15
 AP 6077 DATA 3,1.E-3
 EH 6078 DATA 1,1
 EL 6079 DATA 1,1
 AC 6080 DATA 1,1.609344E3
 CE 6081 DATA 4,1.E-6
 DW 6082 DATA 3,1.E-9
 BF 6083 DATA 1,1.E-6
 DB 6084 DATA 5,1.E-6
 AW 6085 DATA 1,1.609344E3
 AZ 6086 DATA 4,1.E-3
 CR 6087 DATA 3,1.E-6
 AA 6088 DATA 1,1.E-3
 BW 6089 DATA 5,1.E-3
 HI 6090 DATA 5,60
 HM 6091 DATA 5,60
 CA 6092 DATA 3,1.E-6
 ZJ 6093 DATA 1,1.E-3
 FD 6094 DATA 5,1.E-9
 SB 6095 DATA 1,1.8533E3
 YG 6096 DATA 4,28.3495
 YK 6097 DATA 4,28.3495
 SQ 6098 DATA 3,8.809E-3
 YO 6099 DATA 4,1.555
 GY 6100 DATA 5,1.E-12
 MB 6101 DATA 3,4.732E-4
 ZZ 6102 DATA 4,453.59237
 MJ 6103 DATA 3,4.732E-4
 WV 6104 DATA 4,1.555
 YU 6105 DATA 3,9.46326E-4
 YY 6106 DATA 3,9.46326E-4
 AP 6107 DATA 6,57.2957
 ER 6108 DATA 1,5.0292
 FK 6109 DATA 5,1
 ED 6110 DATA 5,1
 EO 6111 DATA 2,2.590E6
 ZL 6112 DATA 2,1.E-4
 RC 6113 DATA 2, .092903
 DV 6114 DATA 2,6.4516E-4
 XL 6115 DATA 2,1.E6
 DU 6116 DATA 2,1
 XI 6117 DATA 2,2.58999E6
 BR 6118 DATA 2,1.E-6
 UK 6119 DATA 2, .836127
 ZG 6120 DATA 2,1.E-4
 ER 6121 DATA 2,100

UNIVERT

Instructions:
 Values to be converted should be entered in the form:

number followed by units

For example, 27.5 square feet

If uncertain of units, type HELP

Enter next value to be converted:
 23 FEET

Enter units desired:
 CUBITS

23 FEET = 15.3333333 CUBITS

PRESS ANY KEY

```

DX 6122 DATA 2, .01
RF 6123 DATA 2, .092903
RJ 6124 DATA 2, .092903
EC 6125 DATA 2, 6.4516E-4
EG 6126 DATA 2, 6.4516E-4
KW 6127 DATA 2, 1.E6
EF 6128 DATA 2, 1
HH 6129 DATA 2, 1.E-12
WM 6130 DATA 2, 2.58999E6
AV 6131 DATA 2, 1.E-6
TO 6132 DATA 2, .836127
AB 6133 DATA 1, 1.609344E3
QQ 6134 DATA 4, 6350
RP 6135 DATA 3, 1.479E-5
RT 6136 DATA 3, 1.479E-5
RX 6137 DATA 3, 1.479E-5
TM 6138 DATA 3, 4.928E-6
PB 6139 DATA 4, 9.07185E5
PY 6140 DATA 2, 9.32396E7
SN 6141 DATA 3, 4.928E-6
WO 6142 DATA 1, .9144
WS 6143 DATA 1, .9144
VF 6144 DATA 5, 3.1536E7
  
```

```

0290 PLA
0300 STA UNITSH
0310 PLA
0320 STA UNITSL
0330 PLA ; (IGNORE HI BYTE)
)
0340 PLA
0350 STA NCHARS
0360 LDA #1
0370 STA RECNUM
0380 LOOP1 LDY #0 ; USE Y AS INDEX
0390 LOOP2 LDA (UNITSL),Y
0400 CMP (TBLL),Y ; COMPARE CHARS
0410 BNE SE1 ; NO MATCH, DONE?
0420 INY ; CHARACTERS MATCH
H
0430 LDA (TBLL),Y
0440 CMP #', ' ; END OF RECORD?
0450 BEQ SE2 ; YES, COMP LEN
0460 CPY NCHARS ; NO, ALL TRIED?
0470 BMI LOOP2 ; NO
0480 SE1 LDA RECNUM
0490 CMP COUNTL ; LAST RECORD?
0500 BNE LOOP3 ; NO, NEXT RECORD
D
0510 LDA #0 ; YES, COUNT=0
0520 STA COUNTL ; NO MATCH
0530 RTS ; RETURN TO BASIC
0540 LOOP3 INY ; INC CHARACTER IN
NDEX
0550 LDA (TBLL),Y
0560 CMP #', ' ; FIND NEXT COMMA
A
0570 BNE LOOP3
0575 INY
0580 TYA
0590 CLC
0600 ADC TBLL ; ADVANCE POINTER
0610 STA TBLL ; TO NEXT RECORD
0630 BCC #+4 ; IF PAGE CROSSED
0640 INC TBLL ; INCREMENT HI
0650 INC RECNUM
0660 CLC
0670 BCC LOOP1
0680 SE2 CPY NCHARS ; LENGTHS EQUAL?
0690 BEQ EXIT ; YES.
0700 INY ; LONGER BY 1?
0710 CPY NCHARS
0720 BNE SE3 ; NO
0730 DEY ; YES
0740 LDA (UNITSL),Y
0750 CMP #'5 ; LAST CHAR AN 5?
?
0760 BEQ EXIT ; YES.
0770 SE3 LDY #0 ; RESET INDEX
0780 BEQ SE1 ; GO SEE IF DONE
0790 EXIT LDA RECNUM ; MATCH FOUND
0800 STA COUNTL ; PASS REC COUNT
0810 RTS ; AS RETURN ARG
  
```

LISTING 2: ASSEMBLY

```

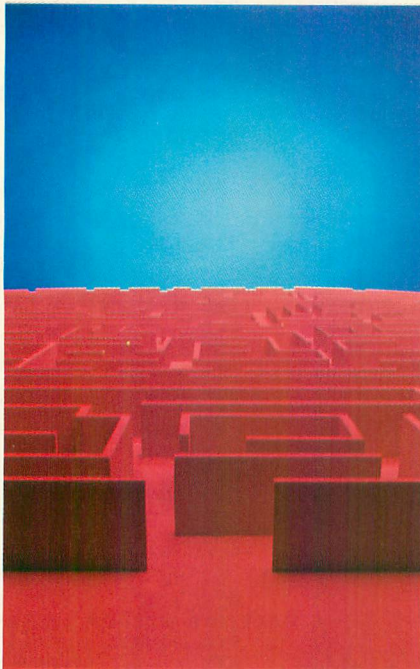
10 .TITLE "SEARCH SUBROUTINE"
20 ;* CONVERSION SEARCH SUBROUTINE *
30 ;A BASIC USR ROUTINE
40 ;INDEX=USR(ADR(SUBR),COUNT,ADR(TABL
LE),ADR(UNITS),LEN(UNIT$))
50 ;
60 ;Variable length records
70 ;Records terminated with comma
80 ;Checks for trailing "5"
90 ;
0100 *= $0600
0110 TBLL = $CB ; TABLE ADDRESS LO
0
0120 TBLH = $CC ; HI BYTE
0130 UNITSL = $CD ; UNITS STRING LO
0140 UNITSH = $CE ; HI BYTE
0150 RECNUM = $CF ; RECORD NUM INDEX
X
0160 NCHARS = $D0 ; CHARS IN STRING
0170 COUNTL = $D4 ; RECORD COUNT LO
0180 COUNTH = $D5 ; HI BYTE
0190 ;
0200 PLA
0210 PLA
0220 STA COUNTH ; FETCH ARGUMENTS
0230 PLA ; FROM STACK
0240 STA COUNTL
0250 PLA
0260 STA TBLH
0270 PLA
0280 STA TBLL
  
```



```

014970 BEQ RR2
014980 RTS
014990 ;
015000 ;stop vbi & remove players
015010 ;
015020 STOPAL LDA #0
015030 STA VFLG
015040 LDX #3
015050 STP STA HPO5P0,X
015060 DEX
015070 BPL STP
015080 RTS
015090 ;
015100 ;intermission display list
015110 ;
015120 ILST .BYTE $70,$70,$70,$70
015130 .BYTE $70,$70,$47
015140 .WORD PS1
015150 .BYTE $70,$70,$70,$47
015160 .WORD PS2
015170 .BYTE $70,$70,$70,$47
015180 .WORD FIRMS
015190 .BYTE $41
015200 .WORD ILST
015210 ;
015220 ;misc. data & tables
015230 ;
015240 VP1 .BYTE $44,$44,$4A,$4A
015250 VP2 .BYTE $24,$24,$2A,$2A
015260 TITLE .SBYTE " RAZY MAZES "
"
015270 SCRTXT .SBYTE " scores 1:000 2:
:000 "
015280 PS1 .SBYTE " SCORE 1: "
015290 F5C1 .SBYTE "000 "
015300 PS2 .SBYTE " SCORE 2: "
015310 F5C2 .SBYTE "000 "
015320 FIRMS .SBYTE " press fire "
"
015330 ;
015340 ;bell sound for middle maze
015350 ;
015360 BELL LDA #23
015370 STA $D200
015380 LDX #5AA
015390 STX $D201

```



```

015400 JSR WAIT
015410 LDA #0
015420 STA $D201
015430 RTS
015440 ;
015450 ;frequencies for bell
015460 ;
015470 NTX .BYTE 0
015480 NOTE .BYTE 132,0,143,0
015490 ;
015500 ;introduction screen w/ghosts
015510 ;
015520 INTRO LDA # <INTL
015530 STA $0230
015540 LDA # >INTL
015550 STA $0231
015560 CCC LDA GX
015570 CMP #SDA
015580 BCC IIA
015590 JSR DEFP0
015600 IIA LDA GX1
015610 CMP #SDA
015620 BCC IIB
015630 JSR DEFP1
015640 IIB LDA $D01F
015650 CMP #6
015660 BNE IIC
015670 RTS
015680 IIC CMP #3
015690 BNE CCC
015700 JSR ZHSC
015710 JMP CCC
015720 ;
015730 ;zero out high scores
015740 ;
015750 ZHSC LDA #10
015760 LDX #4
015770 IIL STA H5C1,X
015780 STA H5C2,X
015790 DEX
015800 BPL IIL
015810 RTS
015820 ;
015830 ;introduction display list
015840 ;
015850 INTL .BYTE $70,$70,$70
015860 .BYTE $70,$47
015870 .WORD INTIL
015880 .BYTE $07
015890 .BYTE $70,$70,$46
015900 .WORD CREDIT
015910 .BYTE $70,$70,$70,$70,$46
015920 .WORD PST
015930 .BYTE $70,$46
015940 .WORD PRSRES
015950 .BYTE $70,$70,$70,$70,$46
015960 .WORD HISC
015970 .BYTE $41
015980 .WORD INTL
015990 ;
016000 ;screen data
016010 ;
016020 INTIL .SBYTE " r Z
"
016030 .SBYTE " K a y mazes "
"
016040 CREDIT .SBYTE " BY BARRY KOLB
BE "
016050 PST .SBYTE " start play
"
016060 PRSRES .SBYTE " option scor
res "
016070 HISC .SBYTE " 1: "
016080 H5C1 .SBYTE "00000 2: "
016090 H5C2 .SBYTE "00000 "
016100 ;
016110 ;add 1 to high score top man
016120 ;
016130 ADDH1 LDX #4

```

```

016140 AA1 INC H5C1,X
016150 LDA H5C1,X
016160 CMP #51A
016170 BCC AA1
016180 LDA #510
016190 STA H5C1,X
016200 DEX
016210 BPL AA1
016220 LDA H5C1,X
016230 AA1 RTS
016240 ;
016250 ;add 100 to top man
016260 ;
016270 AHUN1 LDX #2
016280 JMP AA1
016290 ;
016300 ;add 100 to bottom man
016310 ;
016320 AHUN2 LDX #2
016330 JMP BB1
016340 ;
016350 ;add 1 to high score bot man
016360 ;
016370 ADDH2 LDX #4
016380 BB1 INC H5C2,X
016390 LDA H5C2,X
016400 CMP #51A
016410 BCC BBE
016420 LDA #510
016430 STA H5C2,X
016440 DEX
016450 BPL BB1
016460 BBE RTS
016470 ;
016480 ;the characer set
016490 ;
016500 *= MYSET
016510 .BYTE $00,$00,$00,$00
016520 .BYTE $00,$00,$00,$00
016530 .BYTE $38,$38,$38,$38
016540 .BYTE $38,$00,$38,$00
016550 .BYTE $66,$66,$66,$66
016560 .BYTE $00,$00,$00,$00
016570 .BYTE $00,$66,$FF,$66
016580 .BYTE $66,$FF,$66,$00
016590 .BYTE $18,$3E,$60,$3C
016600 .BYTE $06,$7C,$18,$00
016610 .BYTE $00,$66,$6C,$18
016620 .BYTE $30,$66,$46,$00
016630 .BYTE $DF,$FD,$DF,$FD
016640 .BYTE $DF,$FD,$DF,$FD
016650 .BYTE $00,$28,$AA,$BE
016660 .BYTE $BE,$AA,$28,$00
016670 .BYTE $00,$28,$00,$69
016680 .BYTE $28,$28,$41,$41
016690 .BYTE $00,$14,$00,$96
016700 .BYTE $14,$14,$82,$82
016710 .BYTE $00,$66,$3C,$FF
016720 .BYTE $3C,$66,$00,$00
016730 .BYTE $00,$18,$18,$7E
016740 .BYTE $18,$18,$00,$00
016750 .BYTE $00,$00,$00,$00
016760 .BYTE $00,$18,$18,$30
016770 .BYTE $00,$00,$00,$7E
016780 .BYTE $00,$00,$00,$00
016790 .BYTE $00,$00,$00,$00
016800 .BYTE $00,$18,$18,$00
016810 .BYTE $00,$06,$0C,$18
016820 .BYTE $30,$60,$40,$00
016830 .BYTE $7C,$CE,$C6,$C6
016840 .BYTE $C6,$E6,$7C,$00
016850 .BYTE $38,$38,$18,$18
016860 .BYTE $18,$18,$18,$00
016870 .BYTE $7C,$E6,$0C,$18
016880 .BYTE $30,$60,$FE,$00
016890 .BYTE $7E,$0C,$18,$0C
016900 .BYTE $06,$66,$3C,$00
016910 .BYTE $0C,$1C,$3C,$6C
016920 .BYTE $CC,$FE,$0C,$00
016930 .BYTE $7E,$60,$7C,$06
016940 .BYTE $06,$66,$3C,$00
016950 .BYTE $7C,$C6,$0C,$FC
016960 .BYTE $CE,$E6,$7C,$00
016970 .BYTE $7E,$06,$0C,$18

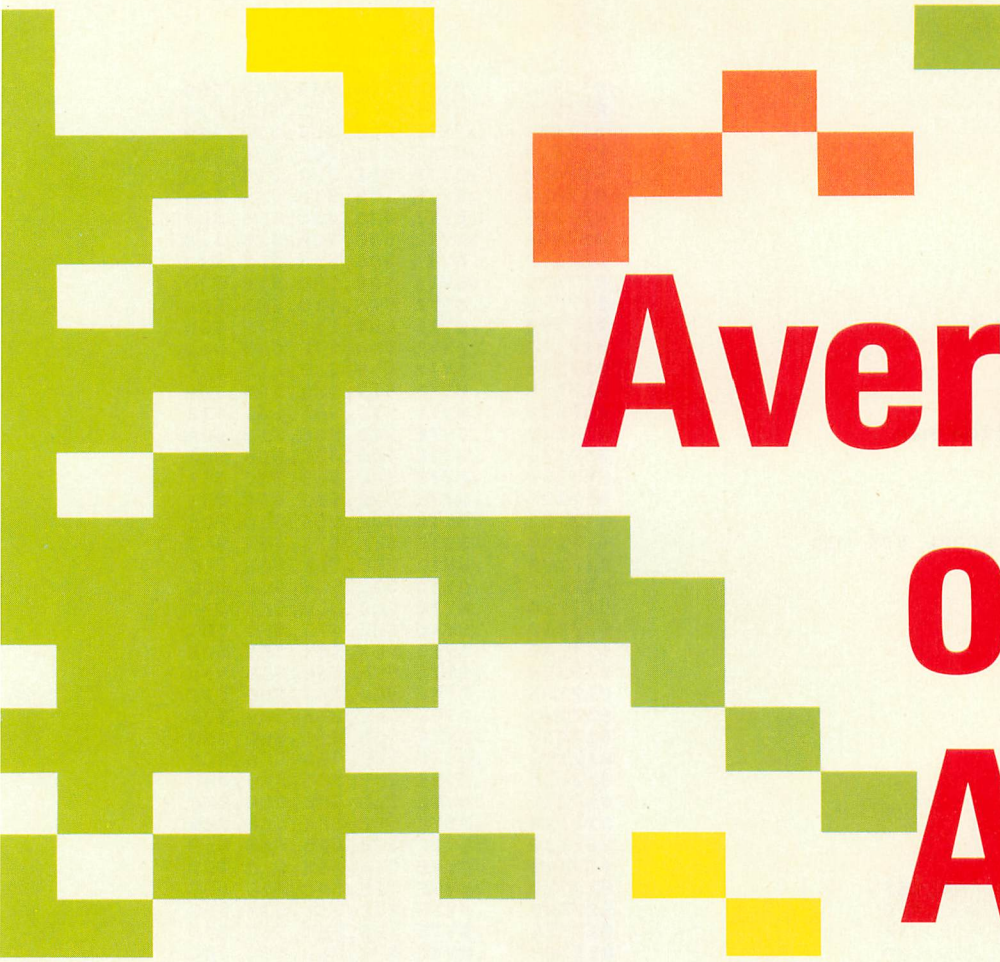
```

```

016980 .BYTE $30,$30,$30,$00
016990 .BYTE $7C,$CE,$E6,$7C
017000 .BYTE $CE,$E6,$7C,$00
017010 .BYTE $7C,$CE,$C6,$E6
017020 .BYTE $7E,$0C,$18,$30
017030 .BYTE $00,$38,$38,$00
017040 .BYTE $00,$38,$38,$00
017050 .BYTE $00,$00,$18,$18
017060 .BYTE $00,$18,$18,$30
017070 .BYTE $06,$0C,$18,$30
017080 .BYTE $18,$0C,$06,$00
017090 .BYTE $00,$00,$7E,$00
017100 .BYTE $00,$7E,$00,$00
017110 .BYTE $60,$30,$18,$0C
017120 .BYTE $18,$30,$60,$00
017130 .BYTE $3C,$66,$66,$0C
017140 .BYTE $18,$00,$18,$00
017150 .BYTE $00,$3C,$66,$6E
017160 .BYTE $6E,$60,$3E,$00
017170 .BYTE $78,$9C,$3C,$36
017180 .BYTE $3E,$66,$66,$C3
017190 .BYTE $EE,$73,$63,$63
017200 .BYTE $6E,$63,$63,$DE
017210 .BYTE $3C,$66,$CC,$C0
017220 .BYTE $C0,$C0,$E6,$7C
017230 .BYTE $EE,$73,$63,$63
017240 .BYTE $63,$63,$63,$DE
017250 .BYTE $FE,$66,$60,$78
017260 .BYTE $60,$63,$66,$7C
017270 .BYTE $FE,$66,$60,$78
017280 .BYTE $60,$60,$60,$60
017290 .BYTE $3C,$66,$C6,$C0
017300 .BYTE $DE,$C6,$66,$3C
017310 .BYTE $C6,$C6,$C6,$CE
017320 .BYTE $FE,$E6,$C6,$C6
017330 .BYTE $30,$18,$18,$18
017340 .BYTE $18,$18,$18,$0C
017350 .BYTE $1E,$0C,$0C,$0C
017360 .BYTE $0C,$1C,$38,$60
017370 .BYTE $C6,$6C,$6C,$78
017380 .BYTE $78,$6C,$6C,$C6
017390 .BYTE $E0,$60,$60,$60
017400 .BYTE $60,$66,$7E,$78
017410 .BYTE $C6,$EE,$FE,$D6
017420 .BYTE $C6,$C6,$C6,$C6
017430 .BYTE $C6,$C6,$E6,$F6
017440 .BYTE $DE,$CE,$C6,$C6
017450 .BYTE $7C,$CE,$C6,$C6
017460 .BYTE $C6,$C6,$E6,$7C
017470 .BYTE $7C,$66,$66,$66
017480 .BYTE $6C,$60,$60,$C0
017490 .BYTE $7C,$E6,$C6,$C6
017500 .BYTE $C6,$C6,$CE,$7F
017510 .BYTE $EE,$73,$66,$6C
017520 .BYTE $66,$66,$63,$C3
017530 .BYTE $3A,$66,$60,$3C
017540 .BYTE $06,$06,$66,$5C
017550 .BYTE $FE,$30,$60,$C0
017560 .BYTE $C0,$C2,$C6,$7C
017570 .BYTE $E6,$66,$66,$66
017580 .BYTE $66,$66,$6E,$3F
017590 .BYTE $C3,$66,$66,$66
017600 .BYTE $66,$66,$3C,$18
017610 .BYTE $C3,$C3,$C3,$D3
017620 .BYTE $CB,$DF,$77,$62
017630 .BYTE $C3,$C3,$66,$3C
017640 .BYTE $3C,$66,$C3,$C3
017650 .BYTE $C3,$66,$66,$3C
017660 .BYTE $18,$18,$18,$18
017670 .BYTE $7E,$C6,$0C,$18
017680 .BYTE $FE,$60,$C3,$FE
017690 .BYTE $00,$1E,$18,$18
017700 .BYTE $18,$18,$1E,$00
017710 .BYTE $00,$40,$60,$30
017720 .BYTE $18,$0C,$06,$00
017730 .BYTE $00,$78,$18,$18
017740 .BYTE $18,$18,$78,$00
017750 .BYTE $00,$08,$1C,$36
017760 .BYTE $63,$00,$00,$00
017770 .BYTE $00,$00,$00,$00
017780 .BYTE $00,$00,$FF,$00
017790 ;
017800 *= $02E0
017810 .WORD BEGIN

```





Pixel Averaging on the ATARI

by Stephen Miller

If you've ever looked at a picture created with one of the ANTIC Mode E graphics programs (i.e. *MicroPainter*, *Atari Artist*, *KoalaPad*, etc.—those using screens made up of ANTIC Mode E instructions in the display list, also known as GRAPHICS 15 or GRAPHICS 7+), you probably noticed that, despite having the highest resolution possible while still having controllable colors, the curves and diagonal lines remain awfully jagged-looking. This is caused by an effect known as “stairstepping” or “aliasing” (the individual “steps” are called “jaggies”). With PIXAV 1.0, the program presented here, this problem can be helped.

Typing It In

Type in Listing 1. Check your work with *BASIC Editor II* and be careful to save a copy before it's run. If you have made any typing mistakes in the DATA statements, the program will tell you where.

The Jaggie Blender

As yet, there is no way to get around some degree of jagginess in a picture created by a

computer, but its effects can be minimized by a long-used technique called “pixel averaging” (“pixel” is a short form of “picture element,” meaning an individual dot of a picture), which blends the jaggies together and smooths out their differences.

This technique involves taking the average brightness (in computerese, “luminance”) of the pixels in a small area of a picture and replacing the center pixel's luminance value with this value. This process is repeated for every pixel in the picture, the area each time centering on the pixel in consideration. For instance, consider a picture with four different possible luminances, including background (sound like any kind of display you know of, with the right SETCOLOR statements?), containing a “neighborhood” of pixels with the following luminance values:

```
3 1 3
2 0 3
1 2 2
```

To find the new value for the center pixel, we add all the values in the neighborhood:
 $3+1+3+2+0+3+1+2+2=17$
and divide by the number of pixels in the neighborhood:

$$17/9 = 1.888$$

or, rounding to the nearest integer, we get 2. We would now change the center pixel's luminance to 2. These steps are repeated for every pixel in the picture. This process is called pixel averaging, or, for short, PIXAVing. However, because PIXAVing a picture on top of itself this way would create an error which would slowly accumulate because previous pixel changes would throw off new ones, we should PIXAV onto another, more expendable picture. This generally creates a cleaner, more polished and consistent output. However, PIXAVing a picture on top of itself does have its uses, as we'll see later.

Variations on a Theme

Also, replacing the center pixel's value is not all we could do. We could also add the luminance of the center pixel to the average luminance, letting the sum “roll over” from three back down to zero, like this:

```
0 + 1 -> 1
1 + 1 -> 2
2 + 1 -> 3
3 + 1 -> 0
```

We could subtract the center from the aver-

age, letting the difference roll under from zero to three, much like in the previous addition; we could AND the average with the center (both in binary form, of course); we could OR the two; and finally, we could EOR (Exclusive OR) the two.

The effects of each of these approaches, multiplied by the two modes of PIXAVing (on top of itself or to an alternate picture) give a wide range of results.

Replace-PIXAVing to an alternate picture tends to blend colors evenly from light to dark. In other words, dark areas encroach on light areas equally as much as light areas encroach on dark areas. This usually works best on pictures of medium overall brightness. Replace-PIXAVing a picture onto itself has much the same effect, except that the picture generally seems to have a light shining on it from an angle. This also usually works best with medium-brightness pictures.

Add-PIXAVing to an alternate picture tends to make light areas dark and dark areas light. The difference between it and simply reversing the colors is that the order of the colors is kept the same. Like replace-PIXAVing a picture on top of itself, add-PIXAVing a picture on top of itself tends to make the picture look like a light is shining on it from an angle.

Subtract-PIXAVing to an alternate picture gives the effect of edge detection. The edges of the objects in the picture become light, while the rest of the picture becomes dark. Subtract-PIXAVing a picture on top of itself tends to make the texture of the picture (usually digitized) stand out and give it more relief. If the picture is drawn and not digitized, it heightens the existing texture and creates some texture of its own.

AND-PIXAVing to an alternate picture works much like replace-PIXAVing, except that the light areas are prevented from encroaching on the dark areas. This usually works best on pictures which have a high overall brightness or many thin, black lines or dots which would normally be swamped by surrounding white areas. Like other forms of PIXAVing a picture on top of itself, AND-PIXAVing has much the same effect of its alternate-picture counterpart, but makes it seem like there is light hitting from the side.

OR-PIXAVing to an alternate picture works like replace-PIXAVing as well, but the dark

areas don't encroach on the light ones. In other words, the areas AND would deaden, OR heightens. OR-PIXAVing a picture on top of itself has much the same effect, but tends to make the picture look washed-out. Both of the OR-PIXAVs work best on pictures of low overall brightness or which have thin, white lines or dots which would also otherwise be swamped by surrounding areas of black.

OR-PIXAVing to an alternate picture, at first glance, has the same effect as subtract-PIXAVing to an alternate page. Not so! Closer examination reveals that the subtracted pictures seem to lack shading, whereas the EORed ones do not. As a matter of fact, what you see when you look at an EOR-PIXAVed picture is a map of the absolute differences between the original picture and a replace-PIXAVed version of it; that is, each pixel in an EOR-PIXAVed picture reflects the absolute difference between that pixel's original value and its new value (if the picture had been replace-PIXAVed). Hard to believe, but true. Strangely enough, EOR-PIXAVing a picture on top of itself looks nothing like a picture EOR-PIXAVed to an alternate picture. EORing a picture on top of itself introduces weird patterns to the picture.

BASICally Slow

Now that we know how to actually PIXAV a picture, it should be easy to code, right? Just slap something together in BASIC, right? Wrong. I tried this and had to interrupt the program 15 minutes after I had typed RUN, because only one-quarter of the screen had been finished by then. This would have meant one hour for every picture processed. I certainly wasn't going to sit through this time and time again, and I knew that, deep down inside, underneath all that BASIC, my Atari wasn't *that* lethargic. I set forth with all the determination I could muster (and all the time I could muster—it was the middle of the school year) to write the main loop in machine language.

Slowing Down to Speed Up

When I started writing, I realized that it was going to be uphill all the way. The only assembler I had was the good old Atari

ASMED (ASsembler/Editor), and I didn't have the money to buy a better one. There was no solution except to keep plugging away. Suddenly, several months later. . . .

The Perfectionist's Curse

I finally had the program written, debugged, and working beautifully. However, even in machine language, with ANTIC's DMA turned off (read: with the screen turned off so the program would run faster) it still took four minutes to PIXAV a picture. Being the perfectionist I am, I wanted it to work even quicker. So I said to myself, "Self, how can you make it any faster?" The only answer I could think of was. . . .

The PLOT Thickens

. . . to write my own PLOT/LOCATE routine. I had been using CIO's PLOT/LOCATE routine up until now, which, I realized, had things to deal with which it didn't need to (in my program, anyway) like split screens, graphics mode checking, etc. So, I plodded along once more, still bearing the weight of the problem of the ASMED. About a month later, *voila!* It worked! Now the PIXAV routine took only 1½ minutes to do its job on a picture. You may not think that this is very fast, but consider that in this minute and a half, the computer has to do 30,720 PLOTS, average groups of numbers of various sizes 30,720 times and do a whopping 276,480 LOCATES (whew!).

After the main part had been perfected, some final polishing made it complete: brightness correction, SAVE routine, etc. Then I realized that I had written my program on a 1200 XL and that GRAPHICS 63 (15 for Mode 7 + 16 for no text window + 32 for no screen clear = 63) wouldn't cut it on an 800 or 400. So, with a quick modification, I had the program working with *all* Ataris. This routine is interesting because it makes use of Location 65527 to identify the ROM version it is operating on (it just so happens that the I.D. numbers for all XLs and XEs are less than 34, while all 800s and 400s are equal to or greater than 34).

You may also wonder why I went to the trouble to write a memory-clear routine to clear the screen and have the GRAPHICS

LISTING 1: BASIC

commands in the program not clear the screen. The reason is that to make the program and both screens fit in memory, they both have to be put so close together that, due to a small bug in the screen-clear routine, the first ten bytes of the Screen-2 display list, which immediately follows Screen 1, would be wiped out by a screen clear on Screen 1 (see *Mapping the Atari*, page 19).

Using the Program

If you accidentally select one of these following options, don't worry. You will be given a chance to abort the operation before it is actually performed.

Load. This option allows you to load only files which are 62 sectors in length. This is the size of an "uncompacted" picture (*MicroPainter* saves its files this way automatically, but to do this with *KoalaPad* or *Atari Artist*, you must press the greater-than key (>) to save the picture as an uncompacted file with the filename D:PICTURE. To load the uncompacted file back into either of these two programs, press the less-than key (<). After you have selected a picture, PIXAV will load it. There is a ten- to 20-second blank screen after this, during which time the program adjusts the pixels so that their color numbers match up with their brightnesses (determined by the color values saved by the graphics program you used).

View. Just what it says. Press Option to return to the menu after the picture is on the screen.


PIXAV. This is the heart of the program, the part that does the actual PIXAVing. Simply follow on-screen instructions.

Save. Fairly self-explanatory.

Quit. If you wish to leave the program, it is imperative that you either use this option or turn the computer off. Otherwise, any subsequent programs run will be working with 16K of memory less than normal.

The End

Well, I hope you enjoy using PIXAV, and get good results from it. Take care, and Happy PIXAVing!

Stephen Miller is a self-taught programmer and a student, caught between high school and college, who started tinkering with Ataris in the fifth grade (1980). This is his first article, and he enjoys programming in BASIC, assembly language and Pascal. His hobbies are reading and burning the midnight oil. 

```

WN 0 REM *****
EB 1 REM *          PIXAV 1.0          *
OS 2 REM *          by Stephen W. Miller      *
ZC 3 REM *
BS 4 REM *          COPYRIGHT 1989          *
PL 5 REM *          BY ANALOG COMPUTING      *
WT 6 REM *****
MY 40 DIM PIXAV$(186),CLEAR$(42),CSWITCH$(
CF 50 DEL$=CHR$(156):DEL$(24)=DEL$:DEL$(2
UP 60 POKE 559,0:GOSUB 960:POKE 559,34:IF
KN 70 GOSUB 1940:DL2=PEEK(560)+PEEK(561)*
WK 80 GOSUB 1940:DL1=PEEK(560)+PEEK(561)*
NG 90 A=USR(ADR(CLEAR$),SCR1,SCR1+7680):A
SW 100 OPEN #2,12,0,"E:":POKE 752,1:POSIT
BU 110 ? :? "Load":? "View":? "PIXAV":? "
TB 120 OLD OPT1=OPT:ON OPT GOSUB 140,350,4
QX 130 CLOSE #1:CLOSE #2:GOTO 100
TA 140 GRAPHICS 0:POKE 752,1:OPEN #1,6,0,
GJ 150 INPUT #1,NTRY$:IF NTRY$(2,2)<)" "
DQ 160 TRAP 180:IF NTRY$(15,17)="062" THE
NX 170 GOTO 150
UC 180 TRAP 40000:CLOSE #1:? "          Re
EI 190 CLOSE #1:OPEN #1,6,0,"D:*.":FOR C
TE 200 INPUT #1,NTRY$:IF NTRY$(15,17)<)"0
AK 210 NEXT C:CLOSE #1:TRAP 40000:Q=10:F0
AB 220 IF NTRY$(C,C)=" " THEN Q=C-1:C=10
HS 230 NEXT C:NAME$(1,2)="D":NAME$(3)=NT
JO 240 FUNC$="load":GOSUB 890
OM 250 IF Q=1 THEN SCRL=SCR1L:SCRH=SCR1H
RH 260 IF Q=2 THEN SCRL=SCR2L:SCRH=SCR2H
VQ 270 IF Q=3 THEN RETURN
DK 280 OPEN #1,4,0,NAME$
JS 290 POKE 849,1:POKE 850,7:POKE 852,SCR
YT 300 A=USR(ADR(CIO$)):FOR X=0 TO 3:GET
ST 310 FOR Y=0 TO 3:LUM=16:FOR X=0 TO 3:I
JB 320 NEXT X:LUM(L0)=16:LUM2(L0)=Y:NEXT
TK 330 IF LUM2(0)=0 AND LUM2(1)=1 AND LUM
VE 340 POKE 88,SCRL:POKE 89,SCRH:POKE 559
YK 350 FUNC$="view":GOSUB 890
PR 360 IF Q=1 THEN POKE 561,INT(DL1/256):
TH 370 IF Q=2 THEN POKE 561,INT(DL2/256):
UT 380 IF Q=3 THEN RETURN

```

```

NA 390 POKE 708,4:POKE 709,8:POKE 710,12:
IF PEEK(53279)<>3 THEN 390
ZA 400 RETURN
ML 410 POKE 82,10:? :? "Replace":? "Add":
? "Subtract":? "AND":? "OR":? "EOR":TO
P=10:MAXOPT=6:OLDOPT=OLDOPT2
PB 420 GOSUB 780:MODE=OPT-1:OLDOPT2=OLDOPT
T:FUNC$="PIXAV":GOSUB 890
AA 430 IF Q=1 THEN 51L=SCR1L:51H=SCR1H:51
=5CR1
EJ 440 IF Q=2 THEN 51L=SCR2L:51H=SCR2H:51
=5CR2
VO 450 IF Q=3 THEN RETURN
XE 460 FUNC$="PIXAV onto":GOSUB 890
YS 470 IF Q=1 THEN 52=5CR1
AB 480 IF Q=2 THEN 52=5CR2
VW 490 IF Q=3 THEN RETURN
AA 500 ? "I'll turn off the screen to mak
e it quicker, but it'll still take a
bout one and a half minutes."
HB 510 ? "Press START to begin."
XN 520 IF PEEK(53279)<>6 THEN 520
NK 530 PIXAV$(119,119)=CHR$(234):IF MODE=
1 THEN PIXAV$(119,119)=CHR$(24)
NW 540 IF MODE=2 THEN PIXAV$(119,119)=CHR
$(56)
JB 550 PIXAV$(120,120)=CHR$(234):IF MODE=
1 THEN PIXAV$(120,120)=CHR$(101)
BN 560 IF MODE=2 THEN PIXAV$(120,120)=CHR
$(229)
TX 570 IF MODE=3 THEN PIXAV$(120,120)=CHR
$(37)
LF 580 IF MODE=4 THEN PIXAV$(120,120)=CHR
$(5)
DA 590 IF MODE=5 THEN PIXAV$(120,120)=CHR
$(69)
ML 600 PIXAV$(121,121)=CHR$(234):IF MODE
THEN PIXAV$(121,121)=CHR$(204)
KO 610 POKE 88,51L:POKE 89,51H:POKE 559,0
:A=USR(ADR(PIXAV$),51<>52,51,52):POKE
559,34:RETURN
KC 620 TRAP 620:CLOSE #1:POKE 752,0:? "Ty
pe the filename: ";:INPUT #2,FILE$:PO
KE 752,1:TRAP 40000
CC 630 IF LEN(FILE$)<2 THEN FILE$(LEN(FIL
E$)+1)=" "
ZW 640 IF FILE$(1,2)<>"D:" THEN NAME$(1,2
)="D:"
WK 650 NAME$(LEN(NAME$)+1)=FILE$
RZ 660 FUNC$="save":GOSUB 890
OU 670 IF Q=1 THEN 5CR1=SCR1L:5CRH=SCR1H
RP 680 IF Q=2 THEN 5CR1=SCR2L:5CRH=SCR2H
VY 690 IF Q=3 THEN RETURN
EZ 700 OPEN #1,0,0,NAME$
RU 710 POKE 849,1:POKE 850,11:POKE 852,5C
RL:POKE 853,5CRH:POKE 856,0:POKE 857,3
0
DH 720 A=USR(ADR(CIO$)):PUT #1,0:PUT #1,4
:PUT #1,8:PUT #1,12:CLOSE #1
ZJ 730 RETURN
OZ 740 ? "Really quit? (Y/N)":OPEN #1,4,0
,"K:"
QH 750 GET #1,A:IF A<>ASC("Y") AND A<>ASC
("N") THEN 750
HO 760 CLOSE #1:IF A=ASC("N") THEN RETURN
WM 770 POKE 106,255:A=USR(58484)
AM 780 OPT=OLDOPT:POKE 82,2:? :POKE 82,9:
? "Press: OPTION to move up":? " STAR
T to move down":? " SELECT to choose"
CG 790 Y=PEEK(84):POKE 766,1:POKE 82,2
VM 800 POSITION 9,TOP+OPT:PUT #2,31
OZ 810 Q=PEEK(53279)
UZ 820 IF Q=6 THEN POSITION 9,TOP+OPT:PUT
#2,32:OPT=OPT+1*(OPT<MAXOPT)-(MAXOPT-
1)*(OPT=MAXOPT)
WP 830 IF Q=5 THEN POSITION 2,Y:POKE 766,
0:? CHR$(28);CHR$(28);CHR$(28);DEL$(1,
3);:RETURN
MQ 840 IF Q=3 THEN POSITION 9,TOP+OPT:PUT
#2,32:OPT=OPT-1*(OPT>1)+(MAXOPT-1)*(O
PT=1)

```

```

VW 850 POSITION 9,TOP+OPT:PUT #2,31
TZ 860 C=20
OF 870 IF PEEK(53279)=Q AND C THEN C=C-1:
GOTO 870
PG 880 GOTO 810
RH 890 POKE 82,9:? "Press: START to ";FU
NC$;" screen 1":? " SELECT to ";FUNC$;
" screen 2":? " OPTION for main menu"
QY 900 Q=PEEK(53279)
EC 910 IF Q=3 THEN 950
SE 920 IF Q=5 THEN Q=2:GOTO 950
RX 930 IF Q=6 THEN Q=1:GOTO 950
OY 940 GOTO 900
IJ 950 POKE 82,2:? CHR$(28);CHR$(28);CHR$
(28);DEL$;:RETURN
GT 960 FOR F=0 TO 4:IF F=0 THEN Z=181
QD 970 IF F=1 THEN Z=41
VD 980 IF F=2 THEN Z=65
IS 990 IF F=3 THEN Z=164
YP 1000 IF F=4 THEN Z=6
YX 1010 FOR X=1 TO Z STEP 6:5=0:FOR Y=X T
O X+5:READ A
TS 1020 IF F=0 THEN PIXAV$(Y,Y)=CHR$(A)
VL 1030 IF F=1 THEN CLEAR$(Y,Y)=CHR$(A)
CP 1040 IF F=2 THEN C$SWITCH$(Y,Y)=CHR$(A)
KP 1050 IF F=3 THEN POKE Y+1535,A
AX 1060 IF F=4 THEN CIO$(Y,Y)=CHR$(A)
FN 1070 5=5+A*Y:NEXT Y:READ C:5=5-10000*I
NT(5/10000):IF C<>5 THEN L=PEEK(183)+P
EEK(184)*256:GOTO 1100
AQ 1080 NEXT X:NEXT F
BA 1090 RETURN
MP 1100 GRAPHICS 0:? :? "Error in line ";
L;""":END
SF 1110 REM PIXAV
GJ 1120 DATA 104,104,104,133,212,104,2840
GU 1130 DATA 133,207,133,89,104,133,7414
NM 1140 DATA 206,133,88,104,133,209,3547
OP 1150 DATA 104,133,208,169,0,133,5914
DZ 1160 DATA 84,169,0,133,85,169,7753
WK 1170 DATA 1,133,218,169,0,133,2015
GA 1180 DATA 203,133,205,32,0,6,2092
QR 1190 DATA 133,204,230,85,32,0,459
BQ 1200 DATA 6,198,84,32,0,6,6466
ED 1210 DATA 198,85,32,0,6,198,9708
DE 1220 DATA 85,32,0,6,230,84,8047
DU 1230 DATA 32,0,6,230,84,32,6926
RK 1240 DATA 0,6,230,85,32,0,6618
FI 1250 DATA 6,230,85,32,0,6,8887
WG 1260 DATA 198,85,198,84,162,0,3176
LD 1270 DATA 165,203,56,229,205,144,3724
OE 1280 DATA 3,232,208,249,24,101,1245
UW 1290 DATA 205,70,205,144,2,230,238
YD 1300 DATA 205,197,205,144,1,232,9459
XH 1310 DATA 134,203,165,203,234,234,8143
HE 1320 DATA 234,41,3,133,203,169,6846
QY 1330 DATA 0,208,156,208,150,165,5218
XW 1340 DATA 212,240,8,165,208,133,726
5F 1350 DATA 88,165,209,133,89,169,750
BC 1360 DATA 0,133,218,165,203,32,931
CJ 1370 DATA 15,6,165,212,240,8,9518
GJ 1380 DATA 165,206,133,88,165,207,3779
RD 1390 DATA 133,89,230,85,165,85,170
DX 1400 DATA 201,160,208,211,230,84,7435
NQ 1410 DATA 165,84,201,192,208,205,7544
ZX 1420 DATA 96,0,0,0,0,0,7376
EA 1430 REM CLEAR
RG 1440 DATA 104,104,133,204,104,133,2845
NC 1450 DATA 203,104,133,206,104,133,8250
CB 1460 DATA 205,162,0,169,0,129,9959
GC 1470 DATA 203,165,203,197,205,208,5461
XZ 1480 DATA 7,165,204,197,206,208,7703
UO 1490 DATA 1,96,230,203,208,235,3335
RJ 1500 DATA 230,204,184,80,230,0,6068
ZE 1510 REM C$SWITCH
TT 1520 DATA 104,104,104,133,203,104,2795
QR 1530 DATA 104,133,204,104,104,133,7408
AN 1540 DATA 205,104,104,133,206,169,4353
ZL 1550 DATA 0,133,85,133,84,169,3359
YW 1560 DATA 1,133,218,32,15,6,880
PR 1570 DATA 170,213,203,240,9,169,3344

```

```

ZX 1580 DATA 0,133,218,181,203,32,463
AL 1590 DATA 15,6,230,85,165,85,7004
WB 1600 DATA 201,160,144,227,169,0,5954
KD 1610 DATA 133,85,230,84,165,84,4832
VX 1620 DATA 201,192,144,215,96,0,3237
LR 1630 REM PAGE 6
JL 1640 DATA 32,15,6,192,0,208,2096
TJ 1650 DATA 7,24,101,203,133,203,7079
GA 1660 DATA 230,205,96,41,3,133,401
SA 1670 DATA 215,165,85,201,160,176,1496
UE 1680 DATA 112,165,84,201,192,176,5834
DP 1690 DATA 106,160,0,162,8,169,278
TE 1700 DATA 40,133,214,169,0,133,7226
DJ 1710 DATA 213,6,213,38,214,144,7726
NI 1720 DATA 13,24,165,213,101,84,1217
GH 1730 DATA 133,213,165,214,105,0,7255
EH 1740 DATA 133,214,202,208,234,165,3519
UC 1750 DATA 85,74,74,24,101,213,20
AZ 1760 DATA 133,213,169,0,101,214,2615
MC 1770 DATA 133,214,24,165,88,101,8889
MW 1780 DATA 213,133,213,165,89,101,9605
RF 1790 DATA 214,133,214,165,85,41,9133
LL 1800 DATA 3,133,216,56,169,3,7684
QI 1810 DATA 133,217,229,216,133,216,767
GV 1820 DATA 6,216,166,216,240,5,4722
UD 1830 DATA 6,217,202,208,251,165,3709
TG 1840 DATA 218,240,18,165,217,49,1631
FS 1850 DATA 213,166,216,240,32,74,1323
UL 1860 DATA 202,208,252,240,26,160,7040
EG 1870 DATA 1,208,22,165,215,166,440
WR 1880 DATA 216,240,4,10,202,208,9726
UO 1890 DATA 252,133,215,165,217,73,1596
PY 1900 DATA 255,49,213,5,215,145,549
UE 1910 DATA 213,96,0,0,0,0,463
IZ 1920 REM CIO
SY 1930 DATA 104,162,16,76,86,228,2578
YR 1940 IF PEEK(65527)<34 THEN GRAPHICS 6
3:RETURN
OX 1950 GRAPHICS 56:POKE 559,0:X=PEEK(560
)+PEEK(561)*256
BI 1960 IF PEEK(X)=79 THEN POKE X,78:X=X+
3:GOTO 1960
OG 1970 IF PEEK(X)=15 THEN POKE X,14
DM 1980 IF PEEK(X)<>65 THEN X=X+1:GOTO 19
60
BS 1990 RETURN

```

```

0400 SCRNI1L = $CE ;SCReen 1 addrLo
0410 SCRNI1H = $CF ;SCReen 1 addrHi
0420 SCRNI2L = $D0 ;SCReen 2 addrLo
0430 SCRNI2H = $D1 ;SCReen 2 addrHi
0440 MODE = $D4 ;1 or 2 scr MODE
0450 ;
0460 ; PLTLOC variables
0470 ;
0480 ADRL = $D5 ;scr data AdDRLo
0490 ADRH = $D6 ;scr data AdDRHi
0500 COLR = $D7 ;pixel COLOR
0510 COUNT = $D8 ;shifts to COUNT
0520 MASK = $D9 ;scrn data MASK
0530 CMD = $DA ;CoMmand
0540 ;
0550 ; PLTLOC constants
0560 ;
0570 LOC = 1 ;LOCate
0580 PLOT = 0 ;PLOT
0590 ;
0600 ; CLEAR variables
0610 ;
0620 BEGL = $CB ;BEGin addrLo
0630 BEGH = $CC ;BEGin addrHi
0640 FINL = $CD ;FINish addrLo
0650 FINH = $CE ;FINish addrHi
0660 ;
0670 ; CSWITCH variables
0680 ;
0690 NEW0 = $CB ;New val for 0s
0700 NEW1 = $CC ;New val for 1s
0710 NEW2 = $CD ;New val for 2s
0720 NEW3 = $CE ;Nes val for 3s
0730 ;
0740 ; *****
0750 ; PIXAV
0760 ; *****
0770 ;
0780 ; This routine, called from
0790 ; BASIC, takes the average of
0800 ; the values of each available
0784 ; pixel in a 3 x 3 square
0810 ; and modifies the center
0820 ; pixel's value with that
0830 ; average. This process is
0840 ; repeated on every pixel in the
0850 ; 160 x 192 ANTIC mode E
0860 ; display.
0870 ;
0880 *=55800
0890 ;
0900 ; Get parameters
0910 ;
0920 PLA ;Satisfy BASIC
0930 PLA
0940 PLA
0950 STA MODE
0960 PLA
0970 STA SCRNI1H
0980 STA SAUM5C+1
0990 PLA
1000 STA SCRNI1L
1010 STA SAUM5C
1020 PLA
1030 STA SCRNI2H
1040 PLA
1050 STA SCRNI2L
1060 ;
1070 ; Main Loop
1080 ;
1090 LDA #0 ;Init loop
1100 STA ROWCR5 ;Top edge
1110 YLOOP LDA #0 ;Init loop
1120 STA COLCR5 ;Left edge
1130 XLOOP LDA #LOC ;Set up for
1140 STA CMD ;9 LOCates
1150 LDA #0 ;Get ready
1160 STA AVG ;to average
1170 STA NUMPIX ;and sum
1180 ;
1190 ; LOCates

```

LISTING 2: ASSEMBLY

```

0100 .OPT NOLIST
0110 ;
0120 ;
0130 ;
0140 ;
0150 ;
0160 ;
0170 ;
0180 ;
0190 ;
0200 ;
0210 ; *****
0220 ; System equates
0230 ; *****
0240 ;
0250 ; Screen
0260 ;
0270 ROWCR5 = $54
0280 COLCR5 = $55
0290 SAUM5C = $58
0300 ;
0310 ; *****
0320 ; Program variables
0330 ; *****
0340 ;
0350 ; PIXAV variables
0360 ;
0370 AVG = $CB ;AVG. of pixels
0380 CENTER = $CC ;value of CENTER
0390 NUMPIX = $CD ;NUM. of PIXels

```

PIXAV Utilities
 by
 Stephen W. Miller
 July, 1987
 for
 ANALOG Computing Magazine


```

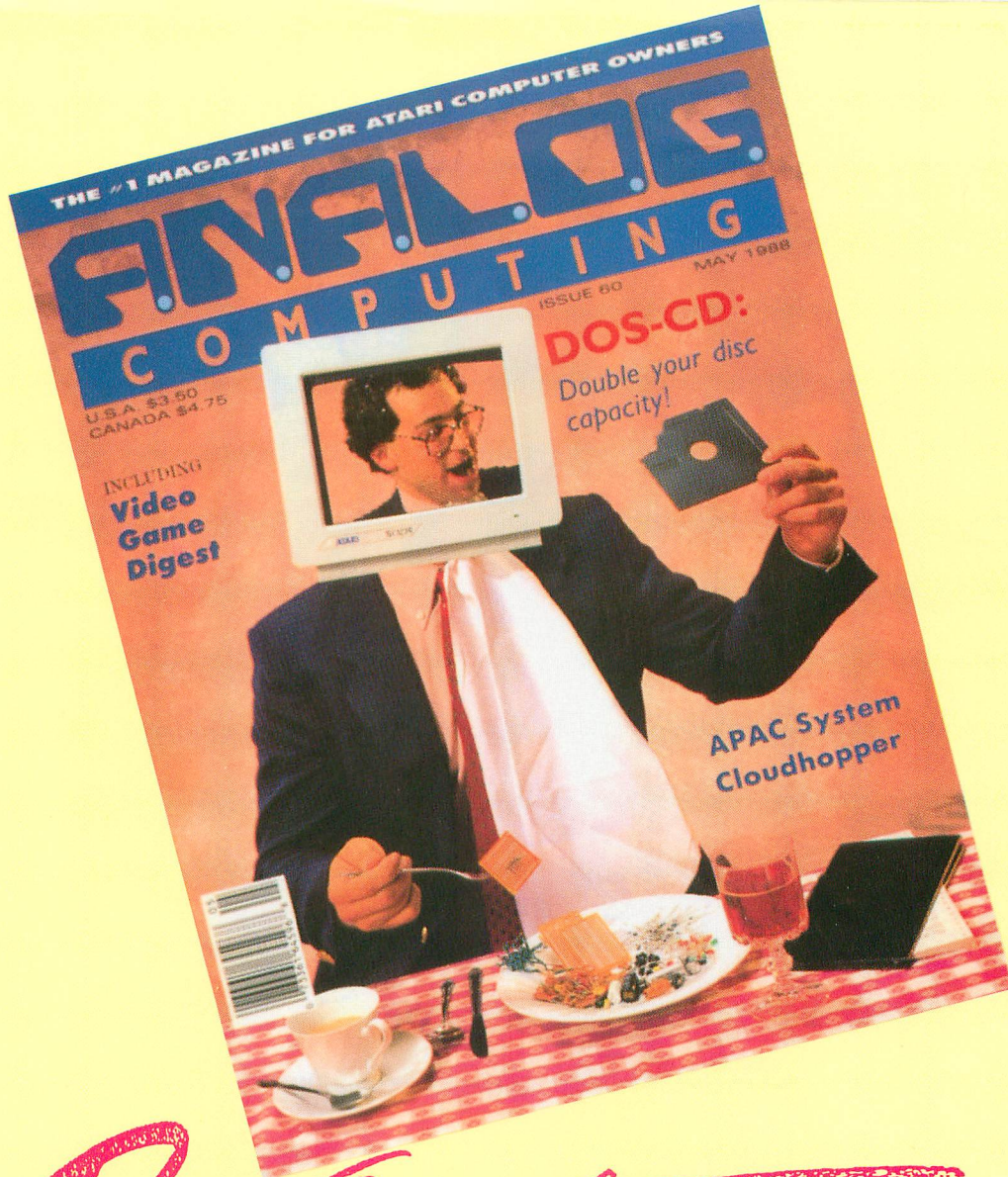
1200 ;
1210 JSR LOCADD ;Center center
1220 STA CENTER
1230 INC COLCR5 ;Center right
1240 JSR LOCADD
1250 DEC ROWCR5 ;Upper right
1260 JSR LOCADD
1270 DEC COLCR5 ;Upper center
1280 JSR LOCADD
1290 DEC COLCR5 ;Upper left
1300 JSR LOCADD
1310 INC ROWCR5 ;Center left
1320 JSR LOCADD
1330 INC ROWCR5 ;Lower left
1340 JSR LOCADD
1350 INC COLCR5 ;Lower center
1360 JSR LOCADD
1370 INC COLCR5 ;Lower right
1380 JSR LOCADD
1390 DEC COLCR5 ;Back to
1400 DEC ROWCR5 ;the center
1410 ;
1420 ; AVG=INT(AVG/NUMPIX+0.5)
1430 ;
1440 LDX #0 ;0 to quotient
1450 LDA AVG ;Get sum
1460 SEC ;Subtract prep
1470 NXTSUB SBC NUMPIX ;Subt. divisor
1480 BCC ROUND ;Round if done
1490 INX ;else count
1500 BNE NXTSUB ;and keep going
1510 ROUND CLC ;Add prep
1520 ADC NUMPIX ;Fix rollunder
1530 LSR NUMPIX ;Divide by 2
1540 BCC COMP ;Round down
1550 INC NUMPIX ;or up
1560 COMP CMP NUMPIX ;remainder-sum/2
1570 BCC FINDQU ;Down if < 0
1580 INX ;or up if > 0
1590 FINDQU STX AVG ;Store result
1600 ;
1610 ; Do operation on AVG
1620 ;
1630 LDA AVG ;get AVG
1640 NOP ;Do
1650 NOP ;the
1660 NOP ;operation
1670 AND #3 ;Cut off all but
1680 STA AVG ;2 LSBs
1690 LDA #0 ;Fail BBX & BBY
1700 BBX BNE XLOOP ;Needed to be
1710 BBY BNE YLOOP ;relocateable
1720 ;
1730 ; PLOT results
1740 ;
1750 LDA MODE ;See if we need
1760 BEQ NOFLP1 ;to flip screens
1770 LDA SCRNL1 ;If yes,
1780 STA SAVMSC ;then
1790 LDA SCRNL2 ;do
1800 STA SAVMSC+1 ;it
1810 NOFLP1 LDA #PLOT ;Setup to PLOT
1820 STA CMD
1830 LDA AVG ;Color to PLOT
1840 JSR PLTLOC ;Do it
1850 LDA MODE ;See if we need
1860 BEQ NOFLP2 ;to flip back
1870 LDA SCRNL1 ;If yes,
1880 STA SAVMSC ;then
1890 LDA SCRNL2 ;do
1900 STA SAVMSC+1 ;it
1910 ;
1920 ; Loop management
1930 ;
1940 NOFLP2 INC COLCR5 ;Move right
1950 LDA COLCR5 ;See if we are
1960 CMP #160 ;at right edge
1970 BNE BBX ;If no, continue
1980 INC ROWCR5 ;If yes, go down
1990 LDA ROWCR5 ;See if we are
2000 CMP #192 ;at bottom edge
2010 BNE BBY ;If no, continue
2020 RTS ;If yes, done!
2030 ;
2040 ; *****
2050 ; CLEAR
2060 ; *****
2070 ;
2080 ; This routine, called from
2090 ; BASIC, sets all memory
2100 ; locations in the specified
2110 ; boundaries, inclusive, to 0.
2120 ;
2130 *=$5700
2140 ;
2150 ; Get parameters
2160 ;
2170 PLA ;Satisfy BASIC
2180 PLA
2190 STA BEGH
2200 PLA
2210 STA BEGL
2220 PLA
2230 STA FINH
2240 PLA
2250 STA FINL
2260 LDX #0 ;Indir nonindex
2270 ;
2280 ; Main loop
2290 ;
2300 CLOOP LDA #0 ;Erase
2310 STA (BEGL,X) ;location
2320 LDA BEGL ;See if we're
2330 CMP FINL ;at the end
2340 BNE CCONT ;If not, cont
2350 LDA BEGH ;See if we're
2360 CMP FINH ;really done
2370 BNE CCONT ;If not, cont
2380 RTS ;Done!
2390 CCONT INC BEGL ;Move on
2400 BNE CLOOP ;Repeat if not
2410 ; ;on page bound
2420 INC BEGH ;If so, INC page
2430 CLV ;Unconditional
2440 BVC CLOOP ;branch
2450 ;
2460 ; *****
2470 ; CSWITCH
2480 ; *****
2490 ;
2500 ; This routine, called from
2510 ; BASIC, replaces each pixel
2520 ; value in the 160 x 192 ANTIC
2530 ; mode E display with a
2540 ; specified pixel value,
2550 ; depending on the original
2560 ; value of the pixel.
2570 ;
2580 *=$5780
2590 ;
2600 ; Get parameters
2610 ;
2620 PLA ;Satisfy BASIC
2630 PLA
2640 PLA
2650 STA NEW0 ;New val for 0s
2660 PLA
2670 PLA
2680 STA NEW1 ;New val for 1s
2690 PLA
2700 PLA
2710 STA NEW2 ;New val for 2s
2720 PLA
2730 PLA
2740 STA NEW3 ;New val for 3s
2750 LDA #0 ;Start at
2760 STA COLCR5 ;upper
2770 STA ROWCR5 ;right
2780 ;
2790 ; Main loop
2800 ;
2810 CSLOOP LDA #LOC ;LOCate

```

```

2820 STA CMD ;Do
2830 JSR PLTLLOC ;it
2840 TAX ;Use result as
2850 CMP NEW0,X ;index to table
2860 BEQ NOPLOT ;If same, skip
2870 LDA #PLOT ;else PLOT
2880 STA CMD
2890 LDA NEW0,X ;color to use
2900 JSR PLTLLOC ;Do it
2910 NOPLOT INC COLCR5 ;Move right
2920 LDA COLCR5 ;See if we are
2930 CMP #160 ;off right edge
2940 BCC CSLOOP ;If no, repeat
2950 LDA #0 ;else return to
2960 STA COLCR5 ;left edge
2970 INC ROWCR5 ;and move down
2980 LDA ROWCR5 ;See if we are
2990 CMP #192 ;off bottom edge
3000 BCC CSLOOP ;If no, repeat
3010 RTS ;else done!
3020 ;
3030 ; *****
3040 ; LOCADD
3050 ; *****
3060 ;
3070 ; This routine, called from
3080 ; PIXAV, adds the value of a
3090 ; specified pixel to AVG and
3100 ; increments NUMPIX, but only if
3110 ; the pixel is within the
3120 ; boundaries.
3130 ;
3140 *= $600
3150 ;
3160 ; Do job
3170 ;
3180 LOCADD
3190 JSR PLTLLOC ;LOCate
3200 CPY #0 ;See if legal
3210 BNE LOCADQ ;If no, then RTS
3220 CLC ;If yes, then
3230 ADC AVG ;update AVG
3240 STA AVG
3250 INC NUMPIX ;and NUMPIX
3260 LOCADQ RTS ;Go back
3270 ;
3280 ; *****
3290 ; PLTLLOC
3300 ; *****
3310 ;
3320 ; This routine, called from
3330 ; PIXAV and C5WITCH, either
3340 ; returns the value of a
3350 ; specified pixel through the A
3360 ; register or changes the value
3370 ; of a specified pixel to the
3380 ; value in the A register,
3390 ; depending on the value in CMD.
3400 ;
3410 ; Prelim fixing and checking
3420 ;
3430 PLTLLOC
3440 AND #3 ;Zero top 6 bits
3450 STA COLR ;Keep it
3460 LDA COLCR5 ;Get Column
3470 CMP #160 ;Out of bounds?
3480 BCS ABORT ;If yes, abort
3490 LDA ROWCR5 ;else get row
3500 CMP #192 ;Out of bounds?
3510 BCS ABORT ;If yes, abort
3520 LDY #0 ;Indir, nonindex
3530 ;
3540 ; ADR=40*row+col/4+5AVM5C
3550 ;
3560 LDX #8 ;8 bit factor
3570 LDA #40 ;Bytes
3580 STA ADRH ;per line
3590 LDA #0 ;Clear
3600 STA ADRL ;product
3610 MULT ASL ADRL ;Move ADR pair
3620 ROL ADRH ;as one big byte
3630 BCC NOADD ;Cont if bit clr
3640 CLC ;Neaten up
3650 LDA ADRL ;Add
3660 ADC ROWCR5 ;other
3670 STA ADRL ;factor
3680 LDA ADRH ;to
3690 ADC #0 ;the
3700 STA ADRH ;product
3710 NOADD DEX ;Count bits
3720 BNE MULT ;Cont if more
3730 LDA COLCR5 ;Divide
3740 LSR A ;column
3750 LSR A ;by four
3760 CLC ;Add
3770 ADC ADRL ;it
3780 STA ADRL ;to
3790 LDA #0 ;the
3800 ADC ADRH ;screen
3810 STA ADRH ;address
3820 CLC ;Add
3830 LDA SAVM5C ;SAVM5C
3840 ADC ADRL ;to
3850 STA ADRL ;the
3860 LDA SAVM5C+1 ;screen
3870 ADC ADRH ;address
3880 STA ADRH ;also
3890 ;
3900 ; Get # of bits to shift things
3910 ; (COUNT=(col MOD 4)*2)
3920 ;
3930 LDA COLCR5 ;Get column
3940 AND #3 ;Clear top bits
3950 STA COUNT ;Keep it
3960 SEC ;Subtract prep
3970 LDA #3 ;Max shift/2
3980 STA MASK ;Also bot 2 bits
3990 SBC COUNT ;Subtract
4000 STA COUNT ;it
4010 ASL COUNT ;Times 2
4020 ;
4030 ; Adjust MASK
4040 ;
4050 LDX COUNT ;# of shifts
4060 BEQ MASKLQ ;Skip it if none
4070 MASKL ASL MASK ;Move left
4080 DEX ;Count shifts
4090 BNE MASKL ;Cont if more
4100 MASKLQ LDA CMD ;PLOT or LOCate?
4110 BEQ PLT ;If 0, then PLOT
4120 ;
4130 ; Do LOCate
4140 ;
4150 LDA MASK ;Keep only
4160 AND (ADRL),Y ;needed bits
4170 LDX COUNT ;Now
4180 BEQ PLTLQ ;move
4190 LOCR LSR A ;it
4200 DEX ;back
4210 BNE LOCR ;over
4220 BEQ PLTLQ ;Done!
4230 ;
4240 ; Abort
4250 ;
4260 ABORT LDY #1 ;Flag error
4270 BNE PLTLQ ;Done!
4280 ;
4290 ; Do PLOT
4300 ;
4310 PLT LDA COLR ;Get pixel color
4320 LDX COUNT ;Move
4330 BEQ COLRLQ ;it
4340 COLRL ASL A ;into
4350 DEX ;correct
4360 BNE COLRL ;position
4370 COLRLQ STA COLR ;Keep it
4380 LDA MASK ;Get mask
4390 EOR #$FF ;A=NOT(MASK)
4400 AND (ADRL),Y ;Clear 2 bits
4410 ORA COLR ;Fill 2 bits
4420 STA (ADRL),Y ;Put it back
4430 PLTLQ RTS ;Done!

```



BOOT UP BIG SAVINGS!

You can save time, and save a lot of money by subscribing to A.N.A.L.O.G. Computing Magazine. Save \$19 off the cover price with the convenience of having A.N.A.L.O.G. delivered directly to your door before it even hits the newsstands. To order use the handy postage-paid order card located in the back of this magazine!

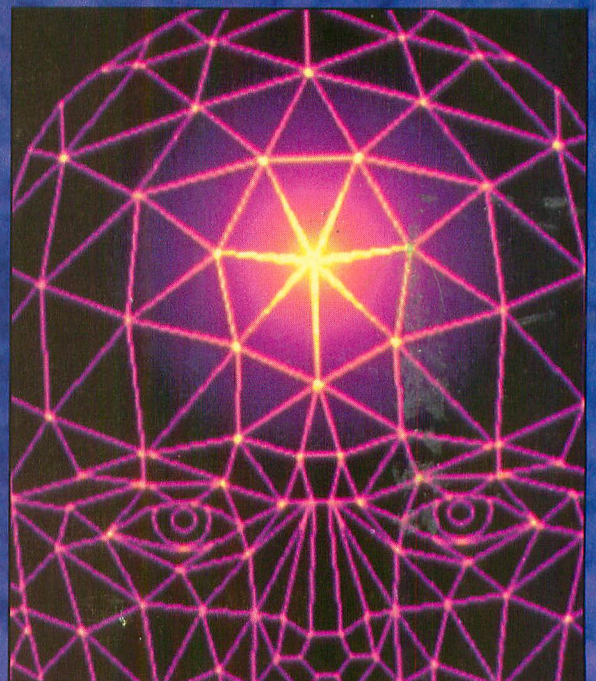
1 YEAR FOR ONLY \$28

SAVE \$19 OFF THE COVER PRICE

1 YEAR WITH DISK ONLY \$79

new
low
price

INSIDE THIS ISSUE:



MORE
MASTER MEMORY MAP
PLUS
**END USER
DATABASE DELPHI**