

VIDEO GAMES SPECIAL!

ANALOG

COMPUTING

T.M.

MARCH 1989
ISSUE 70

P
LAD

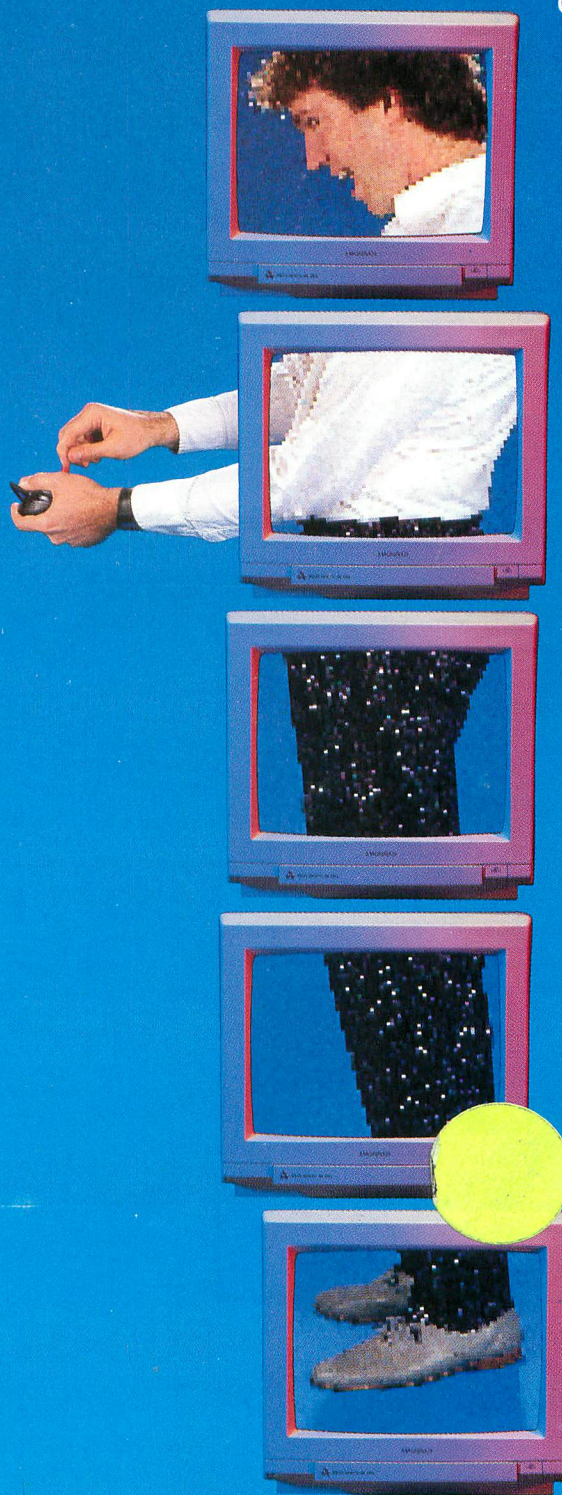
U.S.A. \$3.95
CANADA \$4.95

A VIDEODISC CONTROL SYSTEM FOR YOUR ATARI

TYPE-IN GAMES:
ELECTRA-BALL
PEBBLES

REVIEWS:
TURBOWORD
QUINTOPUS

ALSO:
MASTER MEMORY MAP
GAME DESIGN WORKSHOP



What's Coming Up

Krazy Mazes Pixel Averaging on the Atari Univert Master Memory Map, Part IX

BUY•SELL•TRADE

BUY•SELL•TRADE

BUY•SELL•TRADE

Computer Repeats

Atari 1040ST



\$449
NEW

with trade-in of 130XE, 1702 Mon., NP-10, 1200 modem
\$CALL for your system

Used

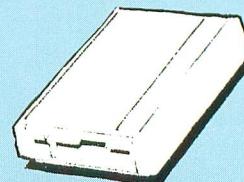
520ST Computer	\$335	1040ST Computer	\$505
Atari SF354 Drive	\$69	65XE Computer	\$85
Atari 800-48K	\$59	800XL Computer	\$69
Hayes Smartmodem	\$49	Atari 1030 Modem	\$35
1050 Drive	\$139	Atari 1025 Printer	\$79
Allen Group Voicebox	\$39	Okimate 10 w/PIP	\$99
ATR-8000 64K, Slaves	\$199	Slave Drives from	\$35
ICD PR:Connection	\$49	Olivetti PR2300 prn	\$119
C1802C Monitor	\$155	Software/Books from	\$1

New

1040ST Plus+ CPU	\$719	520ST FM CPU	\$499
Atari SF314 Drive	\$209	Indus GT-100 Drive	\$209
Atari 130XE Computer	\$149	SM124 Mono Mon	\$159
US Doubler w/DOS	\$39	XF-551 Drive	\$179
SC1224 Color Mon	\$325	Avatex 1200eHC	\$79
Atari SX-212 Modem	\$89	Modem cable	\$15
Avatex 2400HC	\$199	1802D Monitor	\$189
Star NX-2400 24 Pin	\$389	Microline 183 wide	\$315
Star NX-1000 144cps	\$189	Epyx 500XJ Joystick	\$19

\$Cash for your equipment
Thousands of software & book titles
Plus, MUCH, MUCH MORE!

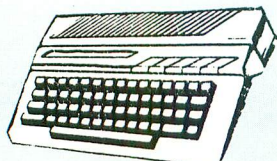
Atari XF551



\$49
NEW

with trade-in of 130XE, 1050 OR \$109 with 1050
\$CALL for your system

Atari 65XE



\$85
NEW

with trade-in of 800XL OR 800-48K
\$CALL for your system

Atari 520ST FM with Double Sided Drive



\$239
NEW

with trade-in of 130XE, 1050, 850, 1200 mdm., color mon.
\$CALL for your system

All references to trade-in assume equipment to be in good working condition. Shipping/handling will be added to all prices. No additional charge for credit cards or COD. Mail order prices shown.

WE CHECK FOR CREDIT CARD THEFT!

Technical Support/Questions: 1-303-939-8144

1-800-347-3457



Authorized Sales & Service for
COMMODORE/AMIGA
and **ATARI ST/XL/XE**
Computers and Accessories.



VISA

MASTERCARD

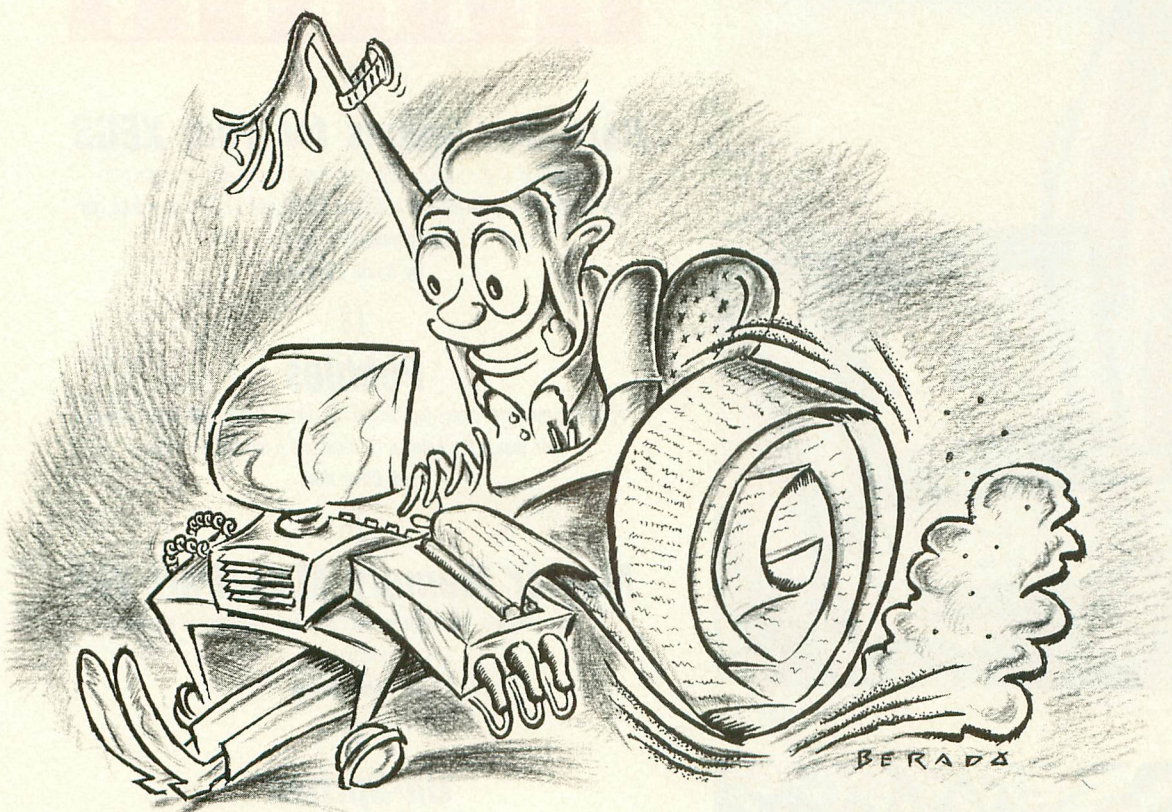
DISCOVER

AMERICAN EXPRESS

COD

CIRCLE #105 ON READER SERVICE CARD.

Editorial



by Clayton Walnum

The other day, when I was flipping through the local paper, I came upon an article that condemned video games, and by extension computer games, in a big way. The article focussed on young children, rather than teenagers or adults, and its author came to the conclusion that, because of their frustrating nature, video games were actually harmful to children, giving them more bad feelings than good.

The author offered as evidence certain behaviors manifested in children who played a lot of video games. These behaviors included crying, screaming and throwing the game controllers in fits of rage. The article made me think for a minute because, though I didn't agree with the author's assessment that these games were harmful, I did recognize that the behaviors he described were real. I've

watched my own kids throw similar tantrums when they were having difficulty with a game, but I never paid much attention—except to turn off the machine.

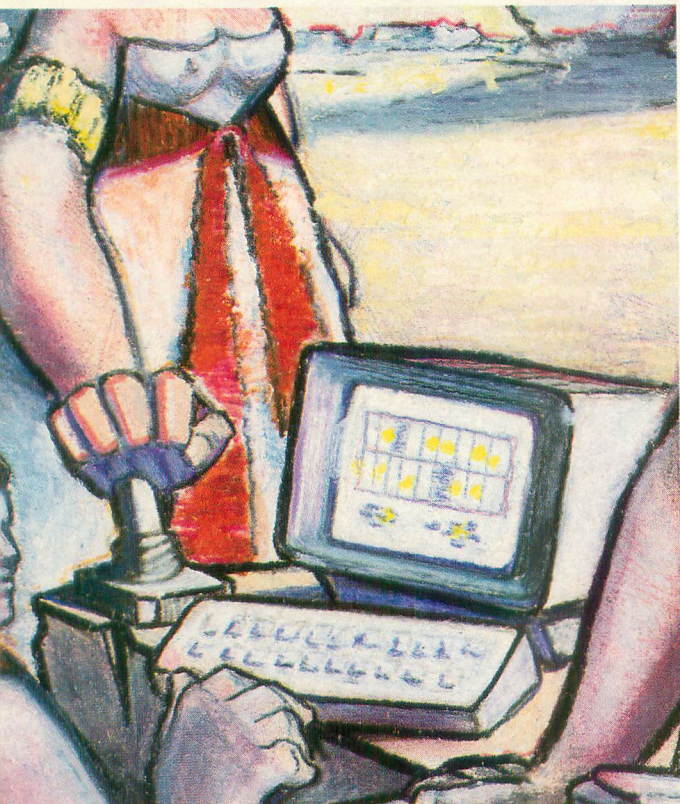
And then I realized that the only difference between me and my kids is that they haven't learned to control their emotions yet; I had to admit that I, too, am subject to "bad moods" when a computer game is being overly ornery. On the other hand, we all know (and clearly the author of the anti-video-game article doesn't) that computer games can also be a lot of fun. You have to take the good with the bad, you know?

But still I thought: What is it that makes these games so important to us that they can stir up such emotions? How can a stupid piece of silicon and plastic keep drawing us back for more abuse? Why, once we finally get on the high scoreboard, do we find that nothing will do except that we now beat *that* score—even if we have to stay up till the wee hours to do it?

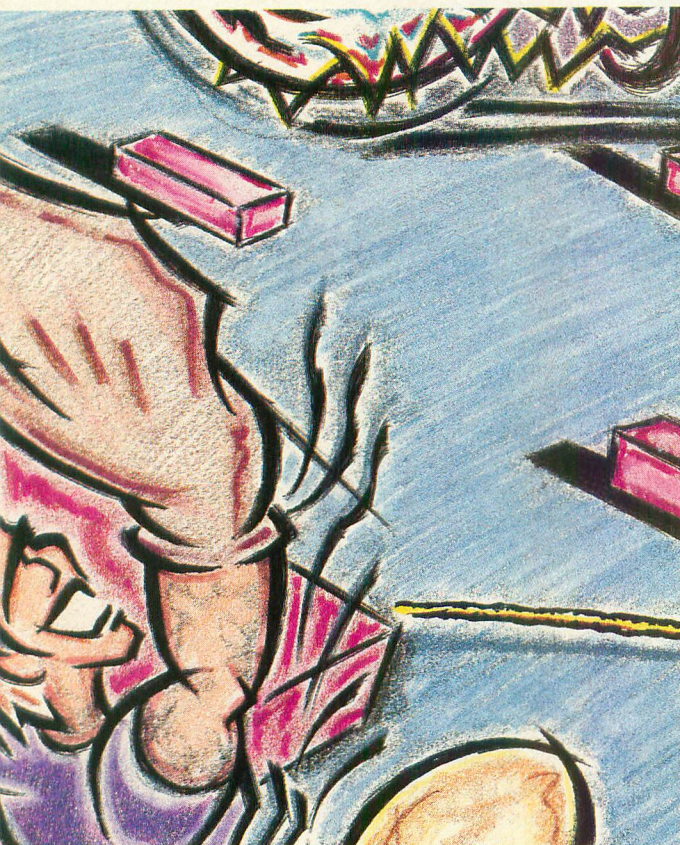
If you think I've got the answers, you're

wrong. I never came to any conclusions. Oh, sure, I've got some half-baked theories, some ideas that might explain why *I'm* drawn to computer gaming. But, just like anything else, I suspect the reasons people spend so much cash and time on these high-tech vices are as varied as the people who play them. Clearly, though, there is something about computer games that draws us in.

So, not being afraid to capitalize on a vice to get you to buy a magazine, we've dedicated this issue of ANALOG Computing to Atari video gamers everywhere. Herein you will find not only special reports on some of the exciting games available for owners of Atari computers and XE Game Systems, but also a couple of type-in games to keep your joysticks healthy and happy. Yep, we'll supply the fun—but just in case that video-game hater was right, you parents keep an eye on your kids, okay? ANALOG Computing doesn't want to be responsible for turning them into neurotics, underachievers or Disneyland tour guides. Deal?



page 18



page 74

FEATURES

9

Cartridge Games for Your XEGS

The addition of the XEGS to the Atari line has caused a resurgence of cartridge-based games—old and new—and 130XE and XEGS owners both can take advantage of the fun.

by David Plotkin

18

Pebbles

From ancient Egypt comes this deceptively simple desert game using nothing more than a few holes in the sand and a handful of stones.

by Clive King

26

Master Memory Map, Part VIII

The most complete Atari 8-bit memory map ever published in a magazine continues.

by Robin Sherer

36

Un-Sprites

Now you can have software-controlled sprites as well as Atari's player/missile graphics.

by Jason Leigh

44

Atari Videodisc System

The secrets of controlling a laser videodisc from your Atari computer.

by Bruce Frumker

48

DUPing BASIC

This handy patch to DOS 2.5 will automatically switch BASIC on and off as you enter and leave DOS.

by Bill Bodenstein

66

Disk Games for Your XEGS

Did you know that with the addition of a disk drive, all the disk-based games for the 130XE computer will also run on the XE Game System? Here's a quick overview of some of the exciting games available now.

by Matthew J.W. Ratcliff

74

Electra-Ball

A challenging, two-player game of strategy and reflexes written in Atari BASIC.

by Frank Martone

ANALOG COMPUTING STAFF

MARCH 1989
ISSUE 70

R E V I E W S

50 **Turboword**

(Micromiser Software)

reviewed by **Matthew J.W. Ratcliff**

73 **Quintopus**

(Computer Software Services)

reviewed by **Jim Patterson**

C O L U M N S

16 **Database DELPHI**

by **Michael A. Banks**

54 **The End User**

by **Arthur Leyenberger**

58 **Game Design Workshop**

by **Craig Patchett**

D E P A R T M E N T S

3 **Editorial**

by **Clayton Walnum**

6 **Reader Comment**

7 **8-Bit News**

53 **M/L Editor,**

by **Clayton Walnum**

70 **Basic Editor II**

by **Clayton Walnum**

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

Cover Photography
MARK CHEN

Illustrations
JOHN BERADO
STEVE STERLING

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
CLIFFORD LAWSON
JUDY VILLANUEVA

Contributors
BILL BODENSTEIN
BRUCE FRUMKER
CLIVE KING
JASON LEIGH
FRANK MARTONE
JIM PATTERSON

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

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

ANALOG Computing (ISSN 0744-9917) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. © 1989 L.F.P., Inc. Return postage must accompany all manuscripts, drawings, photos, disks, etc., if they are to be returned, and no responsibility can be assumed for unsolicited materials. All rights reserved on entire contents; nothing may be reproduced in whole or in part without written permission from the publisher. U.S. subscription: \$28 for one year (12 issues), \$52 for two years (24 issues), \$76 for three years (36 issues). Foreign subscription: Add \$7 per year. Single copy \$3.50 (add \$1 for postage). Change of address: Six weeks advance notice, and both old and new addresses are needed. POSTMASTER: Send change of address to **ANALOG Computing Magazine**, P.O. Box 16927, North Hollywood, CA 91615. Second-class postage paid at Beverly Hills, CA, and additional mailing offices.

READER COMMENT

Too much ST coverage?

As a long-time Atarian and an ex-reader of your magazine, there's a few issues I'd like to discuss with you. Atari computers have been a big part of my life, and I have enjoyed several years of service from my 8-bit computer. I have owned an Atari 800, 1200 and now a 130XE and have found that Atari computers are excellent machines.

My collection of Atari 8-bit software (all purchased!) consists of over 300 different titles. But it appears that we've reached the end of the line as far as software is concerned. I don't buy the argument that Atarians are more prone to copying software than buying it. But the lie worked, and software companies have apparently abandoned our machines.

So what do we do? Well, I haven't quit requesting that companies like MicroProse, Infocom or Electronic Arts continue supporting the 8-bit line. But it's very frustrating to sit by and watch as other computer magazines discontinue or reduce Atari coverage. In the past year, magazines like *Compute!*, *Family Computing* and even *Computer Shopper* have dropped or reduced the coverage of our machines.

Even more disturbing, however, is the fact that Atari-specific magazines like yours have become so ST-oriented. Over the past few years, we have had to thumb through page after page of ST hype, and yet we're asked to be supportive of Atari as a company and as a viable computer. I didn't plan to buy an ST, and probably never will, but I've read more than enough about these machines. Month after month of this has taken a toll on me, and since I've added a PC-compatible to my collection, I find that I am spending less and less time with my Atari. Because of this, I have dropped ANALOG Computing from

my standard reading each month.

I hope that Atari finally turns the corner, but with the problems that the company has shown in the past keeping promises, making deliveries and introducing new products (not just rehashing old titles I purchased four years ago), and in singing the praises of the ST (while ignoring the 8-bit users), it's looking very unlikely.

What will it take for Atari to come back? Well, in my opinion, they need to realize that everyone doesn't always want the latest model that comes down the pike. Also they need to realize that the best thing that any computer company has going for it is the good will of its users. It's very hard to tell someone to invest in Atari computers for their home when there's nothing new being introduced for the machine. In the past year, I've been asked to recommend home computers to four different people, and with a great deal of distaste, I've recommended Commodore 64s (ugh) on the basis that the machines are still actively supported and are excellent low-end machines (which they are, but not as good as Ataris). Also, if I wanted to be buried in ST news, I'd buy one of the ST-specific magazines.

Well, I've griped enough, but I wish you well in the future and hope that someday I will return to the role of avid reader of your magazine.

—Michael D. Winbush
Cincinnati, OH

Like many people who feel strongly about a subject, you are, to be blunt, allowing your emotions to overshadow your logic. First, it is certainly true that software piracy has been cited as a reason for the lack of new Atari-specific products. However, to say "the lie worked, and software companies have apparently abandoned our machines" is ridiculous. The word "lie" implies that someone is deliberately trying to destroy the Atari 8-bit software market. Who? Why? Are we talking about spies from Commodore user groups trying to sink the competition?

There is only one reason why software companies are making less and less software for the 8-bit Ataris: They believe it is not a profitable venture. Plain and simple. And software piracy is not, of course, helping increase the profit-making potential of software (any software). As much as you adore your

Atari computer, you have to face the fact that the computer was created with one objective in mind: to make money. And the software for that computer was created for that same reason. They call it capitalism—and it's a philosophy that our country holds dear.

What can you do? Show the software manufacturers that Atari 8-bit software is profitable. Write to them and tell them what you're interested in, and when they respond with a new product, buy it.

Secondly, you say that ANALOG Computing has become "so ST-oriented." Well, I just did a page count of ST-related material for the last year, and here's the scores:

8-bit material: 96.5%

ST material: 3.5%

Emotions can frequently warp our perception of reality. The above figures are a case in point.

Finally, Atari 8-bit owners have to realize that even though they are happy with their machines and have little desire to upgrade, the industry as a whole will not tolerate stagnation. The home computer market is still very much in its infancy and, as a result, is extremely volatile. If Atari does as some 8-bit users suggest—if they ignore new developments in order to concentrate on the old—they are dead. Stone, cold dead. Computer technology is advancing at a dizzying rate, and the rules of the game are keep up or drop out.

Atari does continue to support its 8-bit line, but now they have to divide their attention a number of ways. Always remember that any sale that Atari makes, whether it be a 130XE, an XE Game System or an ST, helps make the company as a whole successful. That success is to everyone's advantage.

Obviously, we at ANALOG Computing are as concerned with the continued success of the 8-bit computer line as you are. But we also know that if you are to tackle a problem, you have to understand it first.

Send your letters to:
ANALOG Computing
Reader Comment
P.O. Box 1413-M.O.
Manchester, CT 06040-1413

8-Bit

NEWS

An adventure game that listens

Covox Incorporated has announced the release of *Escape From Planet X*, the first computer text adventure that offers the option of game control by spoken commands when played with the Covox Voice Master or Voice Master Jr. The player teaches the program to recognize his voice, after which the game is played, without need of the keyboard, by speaking into the Voice Master microphone.

You begin the game strapped to a table in the Human Research Lab, peering into the noseless face of the mad professor Schism. In the words of the author, "Escape certain death by scientific experiments! Confront exotic ocelots in alien jungles! Explore the ruins of an ancient city! Dig for treasure in far-off islands! Culture! Art! Disease! Asteroids! Have fun exploring Planet X, and then have fun blowing it up!"

The game itself is priced at \$19.95. The game with Voice Master Jr. is \$49.95 and is \$89.95 with the original Voice Master.

Covox Incorporated
675 Conger Street
Eugene, OR 97402
(503) 342-1271

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 or product samples whenever possible.

CIRCLE #151 ON READER SERVICE CARD.

NEWS

Sophisticated database still available

Industrial Machine Products (IMP), a company that continues to support the Atari line of 8-bit computers, has available a combination database and report-generator program for use on an Atari computer with at least 64K RAM. *Super Database 1-2-3* consists of several software packages and associated manuals, including *Filewriter*, *Reportwriter*, *Menuwriter*, an on-disk tutorial and a 137-page manual.

Super Database allows the user to create complete data-entry applications, as well as report and menu applications, each tailored to the user's needs. These applications, once created, are full-fledged programs, compiled into BASIC code by the database program—

but the user need have no programming knowledge to take advantage of this feature.

Super Database 1-2-3 comes on three disks and is priced at \$59.95.

Also available from IMP are various home and business applications programs, priced at only \$12 each, as well as a series of text adventure games at only \$9 each. Call or write to IMP for more information.

Industrial Machine Products
Rt. 1 Box 362
Ozark, MO 65721
(417) 485-6398

CIRCLE #152 ON READER SERVICE CARD.

Cheaters take note!

Cheat! from AlphaSystems is a utility program that allows you to modify game programs so that you have unlimited lives. One of over 120 games included on *Cheat!*'s menu can be easily modified, including *Blue Thunder*, *Boulderdash*, *Caverns of Mars*, *Crossfire*, *Cyclod*, *Dig-dug*, *Frogger II*, *Gateway to Apshai*, *Jumpman*, *Karateka*, *Montezuma's Revenge*, *Pitfall*, *Shamus II*, *Zeppelin*, and many others. (Some magazine programs are included too, such as ANALOG Computing's own *Livewire*.)

Also included with *Cheat!* is a utility called *Uncheat!* which allows you to return the game to its original condition. *Cheat!* runs on any Atari 8-bit computer and is priced at \$24.95.

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

CIRCLE #153 ON READER SERVICE CARD.



SpartaDOS X released

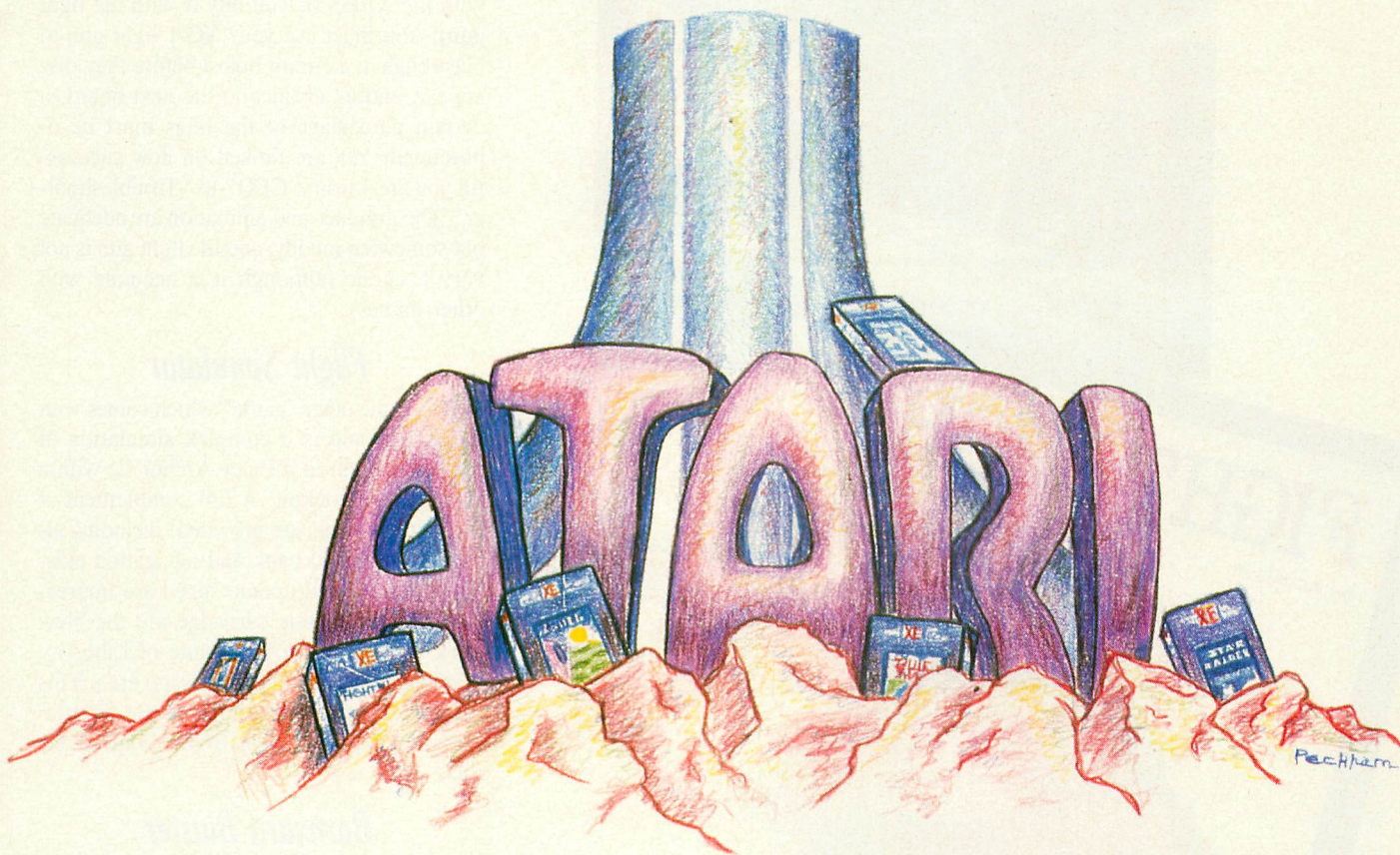
ICD has released *SpartaDOS X*, a disk-operating system that claims to revolutionize Atari 8-bit performance by providing an enhanced DOS that gives your machine more power than ever before. This special cartridge plugs into your computer's cartridge slot and provides a "piggyback" extension for other cartridges so that *SpartaDOS X* won't interfere with normal machine usage.

SpartaDOS X allows RAMdisks of up to one megabyte and provides high-speed operation for systems using the Indus GT or Atari XF551 disk drives. Also included in *SpartaDOS X* is a complete set of ARC utilities that allow you to compress and store files faster and better. *SpartaDOS X* sells for \$79.95.

ICD Corporation
1220 Rock Street
Rockford, IL 61101
(815) 968-2228

CIRCLE #154 ON READER SERVICE CARD.

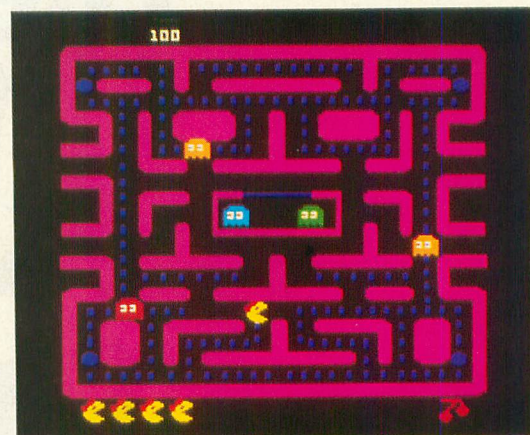
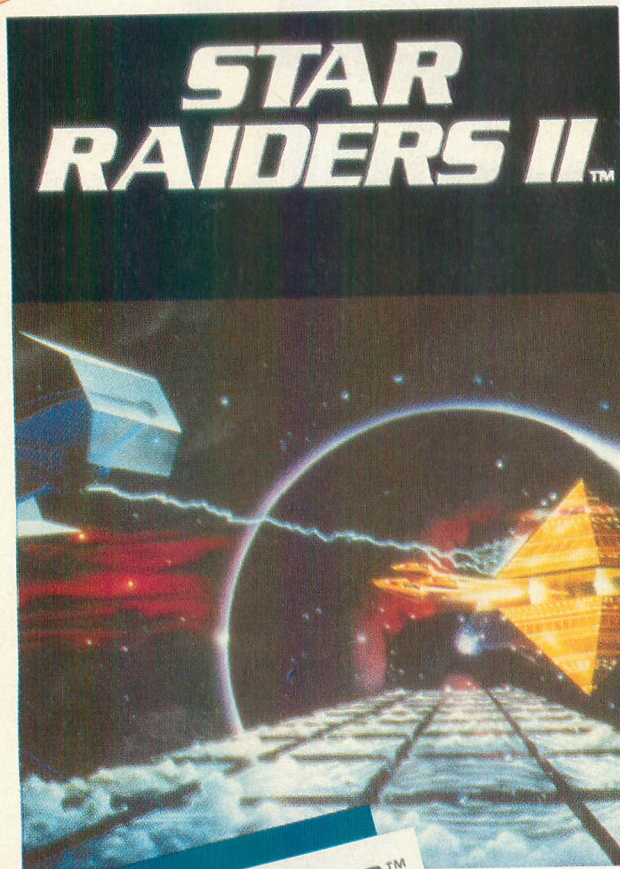
The XEGS Cartridge Summary



by David Plotkin

The supply of cartridges for the XE Game System (XEGS) has continued to grow as Atari releases some new titles and also some games seen in cartridge form for the first time. Increased availability of older cartridges has also

helped the new XEGS owner obtain games. The purpose of this article is to give a quick idea of the games available for XEGS as of this writing (late November, 1988). Long-time ANALOG readers will recognize many (most?) of the titles included here.



Ms. Pac-Man

Bughunt

This is one of the two games which comes with the XEGS (if you buy it with the light gun). You must use your XG-1 light gun to blast bugs in a circuit board before they disappear, and to advance to the next board, a certain percentage of the bugs must be obliterated. You are ranked on how successful you are—from “CEO” to “Troubleshooter.” The graphics and animation are adequate but somewhat muddy, and the light gun is not very accurate (although it *is* accurate with other games).

Flight Simulator

This is the other “game” which comes with the XEGS and is a complex simulation of real-time flight in a Piper Archer II, with a view out the cockpit. A full complement of flight instruments are provided, including altimeter, turn and bank and navigation aids. The graphics and update speed are impressive. Built into this cartridge are the New York area (look for the Statue of Liberty), Washington state and the Chicago area. This game is compatible with the Flight Simulator scenery disks (disk drive required for this).

Barnyard Blaster

Here you use the light gun (which is quite accurate in this game) to blast a variety of objects: tin cans on a fence, watermelons and other fruits/vegetables in a field, and small rodents and birds in the barn. There is even a bonus round where your “Gramps” tosses bottles into the air. Your ammunition is limited, and you must get a certain score to proceed to the next round. The graphics are good and the sound (blast of the gun, tinkle of glass, etc) is superb.

FIGHT NIGHT™ by Accolade



Archon

This is the classic battle of “dark against light,” played on a chess-like board with animated pieces. The two sides are evenly matched, although the pieces on each side are not identical. Each piece has its own traits, such as lifespan, weapons and speed. Some pieces have to get close to do damage, while others carry long-range weapons. There is even one piece on each side which can cast magic spells. When two pieces try to occupy the same square, they are transported to an obstacle-strewn battlefield to fight it out. The winner retains the square, while the loser is removed from the game. Pieces are more powerful on squares of their own color. Some of the squares even change color as the game progresses. Two can play, or the XEGS is a worthy opponent. The game is over when either all the “power points” are occupied by one side or all the pieces on one side are vanquished.

Fight Night

In this game your boxer can climb into the ring with any of five different opponents, either controlled by the XEGS or another person. Each opponent has a “specialty” punch. There are eight different punches, controlled with the joystick and button. You can build your own boxer from an assembly of body parts and save him to memory or disk (if you have a disk drive). You can get used to your custom boxer using the “Training” and “Sparring” options, or set up a tournament elimination. The Main Event lets you go up against the XEGS-controlled opponents, and if you win, you can go against the next toughest one—but if you lose, you must win a rematch before you can move on. The graphics are cartoon-like and comical.

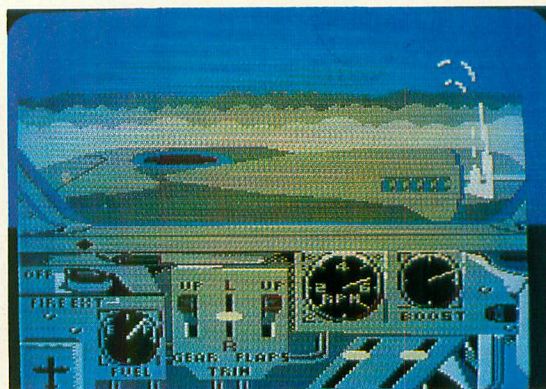
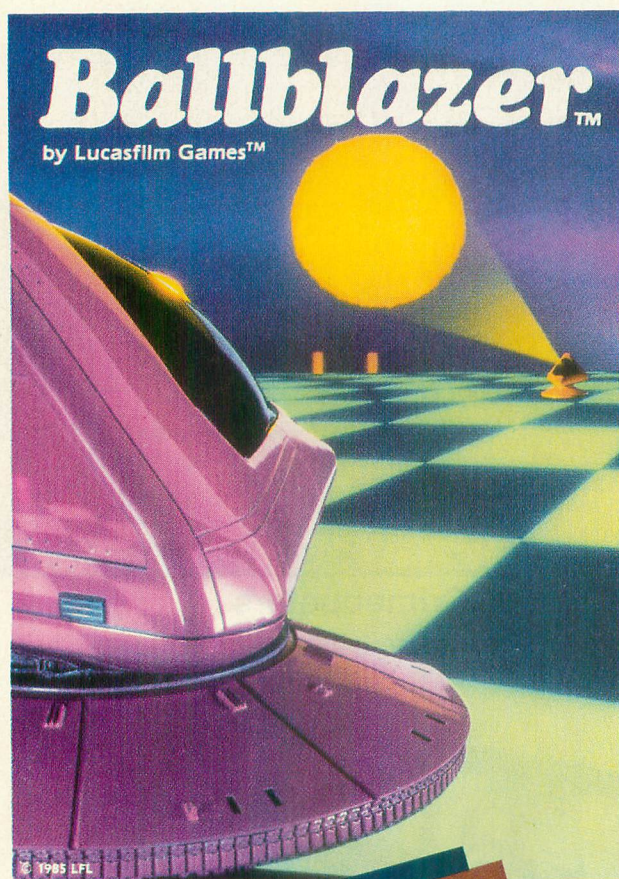
One on One

This is computer basketball at its finest. Julius Erving (Dr. J) and Larry Bird square off on the court. The XEGS can control either player, or two people can play. If the XEGS is playing, variable levels (from Park-and-Rec to Pro) are selectable. The graphics and animation are very good. Each player performs his trademark shots, and a particularly vicious slam-dunk can break the backboard.



Crimebusters





Ace of Aces

David's Midnight Magic

Pinball preceded computer-games, but this pinball simulation combines them effectively. It includes bumpers, rollover, drop targets and flippers. There is even a "tilt" feature if you bump the table too hard. The graphics are quite good and the sound is very realistic.

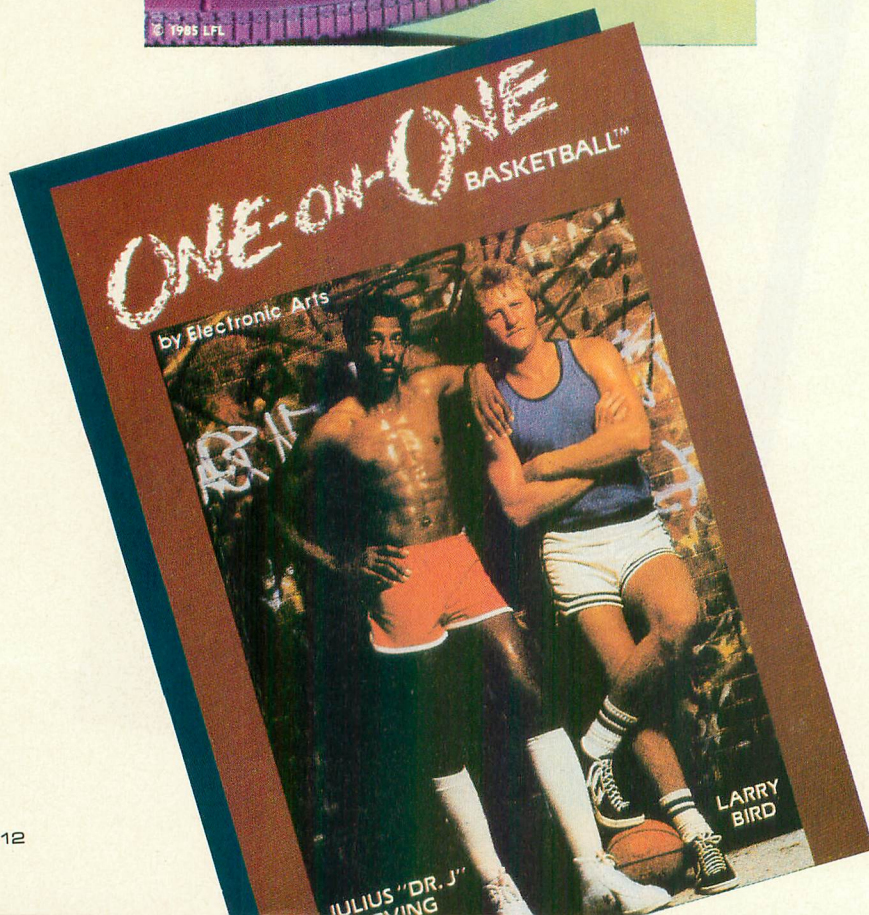
BallBlazer

This one is a futuristic sports contest, where your opponent is either the XEGS or another person. The object is to use your rotofoil (sort of a high-tech Hyundai) to move a floating sphere across a playing field and into your opponent's goal. The number of points you get for a goal depends on how far away you are when you shoot for the goal. Did I mention that the goalposts move? Also, your opponent can steal the ball if you are not careful. The graphics are outstanding: the screen is split horizontally, the top shows one view, the bottom shows the other, and the 3-D checkered playing field scrolls extremely smoothly.

Blue Max

In *Blue Max*, you pilot a WWI biplane on a mission to obliterate enemy headquarters. The view is 3/4 perspective and scrolls smoothly. The landscape is dotted with buildings, gun emplacements, tanks, boats, bridges and scenery. There are also enemy planes. Your plane has two weapons: a machine gun which is effective close to the ground, and bombs (great for buildings!). Everything is shooting back, but fortunately, friendly air-

MARCH A.N.A.L.O.G. Computing



fields show up periodically for refueling and repair. You get only one plane, though. You can win this game—it is possible to reach enemy headquarters and bomb it into a pile of rubble!

Star Raiders II

Although intended as the sequel for the original blockbuster *Star Raiders*, this game is completely different. It is still, however, full of furious action, has excellent graphics and an out-the-cockpit perspective. Your home system's four planets are being invaded by battle fleets from a neighboring system. Your single fighter is all that stands between your home and disaster. Your first job is to battle the oncoming fleets, which consist of fighters and heavy carriers. If any of the fleets reach a planet, the carriers begin destroying cities. You must descend to the planet and destroy the carriers before all cities are obliterated—ending the game. To refuel and rearm, you must warp to a space station; and to stop the seemingly inexhaustible supply of battle fleets, you must warp to the enemy system and destroy their cities. As you can imagine, it gets pretty hectic. But it's a lot of fun, too.

Lode Runner

The hero of this arcade/strategy classic is the "Lode Runner," who must retrieve the gold from the 150 screens of the game. Not only are there guards whose touch is fatal, but each screen is a puzzle. Some can only be solved by moving through them in a certain sequence, and some even have pockets from which there is no way out. Picking up all the gold on a screen provides a ladder to the next screen. Each screen has bricks, ladders, poles and platforms for moving around. *Lode Runner* has a gun which can drill a hole in a brick. If a guard falls into the hole, he is temporarily immobilized. The bricks grow back after awhile, so if you time it right, you can get rid of guards this way. This game comes with a utility allowing you to build your own screens, which can be saved to disk. A single screen can be saved in memory, but is lost when the XEGS is shut off. The graphics are utilitarian, but more than adequate.

Battlezone

Hop into your tank, crank up the engine and move out onto the hostile landscape. The effective simulation of 3-D is done with line drawings (the original arcade hit used vector graphics). Your view is out of the cockpit, and the enemy is other tanks, saucers, missiles and supertanks. The graphics are not very sharp, and the obstacles on the landscape sometimes block your escape at a critical time. The result: a cracked windshield and the end of the game.

Hardball

You can play baseball on your XEGS, and the excellent graphics of this offering make it an enjoyable experience. You can play manager, shifting the infield or outfield or choosing a designated hitter. The main view is from over the pitcher's shoulder, with a view of the infield in a window in the lower right corner. If you are playing the field, you

placements and saucers, which you can blast with your lasers. Each pilot's downed craft sends out a beacon that you can home in on; and when you have cleared the enemy emplacements out of the way, you can land and pick up the stranded pilot. But beware of surprises even on the surface of the planet. The graphics are outstanding, with a realistic landscape and some remarkable effects.

Gato

Command a WWII submarine in *Gato*. You can choose from varying difficulty levels and day and night missions. Your sub is armed with torpedoes, mines and radar, and various missions are assigned, from rescuing airmen to sinking a convoy. The patrol area is divided into quadrants. In the lower levels of difficulty, your maps show the sub tender (for repairs) and enemy (for sinking). You also have a quadrant chart, which shows everything in your current quadrant in much

Remember those wonderful food fights in the cafeteria at school? Well, now you can have that same fun without getting dirty.

can choose the type of pitch to throw. If you are batting, you must select when to swing. If there are men on base, you can even try to steal bases. If you do manage to hit the ball, the view shifts to the outfield, where the defending player can move the fielders to get the ball and then choose where to throw. *Hardball* supports extra innings and changing the lineup during a game.

Rescue on Fractalus

Your mission: descend to the Planet Fractalus in your starfighter and rescue pilots that have been shot down. The enemy will try to prevent you from doing so with laser em-

greater detail. The joystick controls the sub's depth and direction, so *Gato* is easy to play. The main screen shows the bridge view when on the surface, a periscope view when shallowly submerged, and nothing when fully submerged. The graphics are limited and sound almost nonexistent. However, the tonnage sunk can be saved to disk.

Food Fight

Remember those wonderful food fights in the cafeteria at school? Well, now you can have that same fun without getting dirty. In this game you play "Charley," who loves ice-cream cones. You must maneuver him across

the screen to eat the ice-cream cone (which is rapidly melting). Four chefs appear to stop you by touching you or flinging food at you, but you can retaliate by gathering up food on the screen (watermelons, bananas, etc.) and flinging it back at them. Just good clean fun!

Necromancer

The three screens of this game are progressively more bizarre, but it is a lot of fun to play. In the first screen, your wizard must grow as many trees as possible by planting seeds with his "Wisp," a sparkling ring. You must also protect these trees from ogres who will try to knock them down (the Wisp blasts Ogres) and spiders who will poison them (the Wisp also blasts spiders). In the second screen, the trees will help you destroy spider larvae, and in the final screen you must defeat the evil Necromancer by pushing over tombstones while avoiding his fire.

Centipede

The original bug-blasting game still has fast action and great graphics. You guide your bug blaster in the lower quarter of the screen, firing upward at a centipede wiggling across the screen while avoiding bouncing spiders. Hitting the centipede breaks him into segments. Complicating matters, there are mushrooms on the screen that block your shots and fleas that dive on your blaster, leaving mushrooms in their wake.

Millipede

This is a more advanced version of *Centipede*, with much the same game mechanics, but with many more bugs to contend with and much better graphics. Swarms of multi-colored bugs show up periodically, and you can really run up the point total. There are also canisters of DDT in the garden, which can take out quite a few bugs at once.

Robotron

This game is just about the most hectic thing I've ever played. In it you must move from screen to screen, blasting enemy robots who are killing off the humans who wander

aimlessly around the screen. You can also rescue the humans by touching them, but before too long, you have to be concerned only with your survival. A variety of fiendish robots tax your shooting skills, and you can plug two joysticks into the XEGS so that you can move and shoot independently. This takes some getting used to, but your scores can increase significantly once you master the skill. The graphics are colorful but very jumpy.

Moon Patrol

In *Moon Patrol* you must pilot a moon buggy across a lunar landscape, which scrolls smoothly from right to left. The buggy can hop into the air and is armed with two guns which fire in different directions. Obstacles include rocks, holes, gun batteries and saucers. One of the nice things about this game is that you can always begin a new game where the old one left off, so you can see all the landscapes.

Jump over barrels and fireballs as you climb up the ladders to reach the top of the screen where Kong waits with your true love.

Space Invaders

The "original" space game returns! The aliens march across the screen, dropping bombs on your gun, which moves across the bottom of the screen, firing upwards at them. Things get desperate as the aliens move ever-lower on the screen, until your base is destroyed three times or they land—either way, the game is over. The graphics are adequate, but this game is very playable.

Asteroids

Chunky space debris flies across the screen and at your small fighter. To fly the fighter, you must move the stick left or right to ro-

tate the ship in the direction you want to go, then press the stick forward to apply thrust. You can fire up to four missiles at once to destroy the asteroids; each hit breaks up an asteroid into smaller pieces until it disappears. Alien ships also periodically appear, shooting blindly. Game variations include the availability of shields, hyperspace, and "bounce" borders.

Pole Position

With a view from above and behind your race car, you must maneuver your auto as it weaves through more and more opponents on a twisty race course. This graphically spectacular game requires you to complete each lap in the required time. If you run out of time, the game is over; if not, the extra time you have left gets you bonus points. Hitting another car results in a fiery crash, which costs precious time. Steering, acceleration and braking is all by joystick control, and

very easy to learn—but not to master.

Jungle Hunt

In this scrolling game, you must try to rescue your girlfriend from the cannibals. As you guide the hero through the vines, crocodile-infested water, bouncing volcanic rocks and the cannibal screens, you must have fast reflexes and a fine sense of timing.

Defender

You pilot a ship across the surface of a planet, shooting it out with aliens who are intent on kidnapping the people on the surface and turning them into dangerous mu-

tants. There are many different kinds of alien ships, some of which explode into deadly trackers when hit. In addition to your lasers, you have smart bombs (destroys everything on the screen) and hyperspace, which materializes your ship at a different place. A radar readout at the bottom of the screen gives some advance warning when aliens are approaching. The graphics on this arcade translation are not very clear, but the game itself is quite playable.

Joust

In this unusual game, you (and a friend, if you like) mount up on buzzards to do battle in an aerial version of knights jousting with lances. The object is to bump your opponents from a higher altitude. If you do, then the rider is killed, and the bird lays an egg, which can be recovered for more points. Failing to recover the egg causes it to hatch into another jouster which has to be defeated. Your altitude is controlled by pressing the fire button, which flaps the bird's wings. As levels progress, the ledges on which you can rest or run vary, and the opponents get faster and smarter. You also have to deal with a particularly tough opponent, the pterodactyl. The graphics are very good, and animation is smooth.

Donkey Kong

Mario, a construction worker, has to rescue his girlfriend from the clutches of the giant ape, Donkey Kong. On the first screen, Mario must jump over barrels and fireballs as he climbs up the ladders to reach the top of the screen where Kong waits with Mario's true love. Then Kong snatches her up, and we advance to the next screen, where a structure of girders supports Kong. Now there are smart fireballs to contend with, but as Mario jumps over the pins that hold up the girders, he pulls them out, and when all six pins have been pulled, Kong gets dumped on his head. But again he gets up, to go to the final screen, where Mario must leap across the moving elevators and climb up past bouncing springs that threaten his every step. Kong always

wins, but the graphics are great, and the game is a lot of fun.

Donkey Kong Jr.

In this scenario, Kong Sr. has been captured by Mario, and it is up to his son to rescue the big ape. There are four screens, but most people will never see beyond two, because this game is *hard*. In the first screen, Jr. must climb up vines, knocking down fruit to clobber the sharp-toothed snapjaws. Then he needs to move to the platforms and climb to the top of the screen. At the second screen, Jr. must push six keys into six locks, while avoiding a variety of birds and snapjaws. And I admit, I've never seen the other two screens. This game is recommended for master arcaders only, but once again, the graphics are superb.

Pilot a ship across the surface of a planet, shooting it out with aliens who are intent on kidnapping the people on the surface and turning them into dangerous mutants.

Pac-Man

The maze game that started it all, *Pac-Man* is a little round circle with a mouth, who moves through a maze, munching dots (but you know this, don't you?). He is chased by four ghosts, and he must outrun and outsmart all four. There are four power pills in the maze, which temporarily allow Pac-Man to turn the tables on the ghosts and munch them, sending them back to the center of the maze for many points. There are fruit bonuses which show up periodically, but the levels are similar, with the ghosts getting a little smarter on every screen.

Ms. Pac-Man

This sequel is very similar to *Pac-Man*, except the maze changes—which is a big change. Also, the location of the fruit bonus is variable.

Dig Dug

Dig through an underground world! Your little man digs tunnels, gaining points as he does so. There are already some tunnels, unfortunately, occupied by two kinds of bad-news creatures. One is a green reptile who breathes fire (it's frying time again) and another is just a small blob with eyes who tries to catch you. Touching either is fatal, but you are not defenseless. First of all, you have your pump-gun. This inflates the creature with air until it bursts (a great graphic effect). The other trick (worth a lot more

points) is to dig under a rock and cause it to fall on an enemy, squashing it. Altogether, this game is very playable and one of my favorites.

Qix

Imagine a blank rectangle that you have to fill. You move your cursor, which draws a line, and each time your line meets a previously drawn edge, the closed shape you have drawn is filled in. Each round is over when you have filled in a certain percentage of the rectangle. Attempting to stop you is the Qix, and a large bouncing *thing* that will clobber you if it touches you. You can't move back-

continued on page 72

by Michael A. Banks

Well, here we are facing Spring. (Okay, some of us are; those of you in Minnesota, Michigan's Upper Peninsula, and such like places have a while to wait yet, and for those in Southern California and much of the really southern South, Spring just means a transition from warm to hot!) You'll probably be spending a bit less time online as the days lengthen and the weather makes it possible to spend more time outside.

Thus, I'm sure you'll want to make the time you do spend on DELPHI more valuable. One way to do this was discussed last month,

wherever you are.

You've probably used DELPHI's Help system at one time or another, but it has features of which you may not be aware. So, I'll give you a quick tour in this column, *and* show you some help resources of which you may not have been aware.

First, let's take a quick look at the general Help system—which you'll find at most prompts.

Basic Help

DELPHI's basic Help system is accessed

DATABASE DELPHI

when we examined how to customize DELPHI to your needs and tastes. This month, we'll take a look at another way to streamline DELPHI usage by avoiding online fumbling. It's called "Help."

Help!

Help is where you find it, and on DELPHI, that's just about everywhere. As you may know, DELPHI features a context-sensitive help system that provides help

by typing HELP at most any prompt. (Note: You don't always have to type HELP. In some areas, typing ? is the same as typing HELP.)

The Help system is context-sensitive in most areas—meaning that it offers help with the area or menu where you type HELP. Type HELP at the Forum prompt, for instance, and you'll see this:

FORUM

Interactive messaging system for Special Interest Groups and Clubs, providing facilities for ongoing discussions on a variety of topics.

Additional information available:

ADD	READ	REPLY	FOLLOW	NEXT	DIRECTORY
FILE	MAIL	DELETE	EDIT	EXIT	QUIT
HIGH	TOPICS				

Enter desired subtopic or CTRL/Z to EXIT.

Note that there are a number of subtopics; if you type the first few characters of one of those, you'll see an explanation of the selected topic. And you may see a sub-subtopic to provide additional help. (Or enter Control-Z to

exit; if you've already selected a subtopic, you may have to enter Control-Z more than once to return to the original prompt. It all depends on how many levels into Help you are.)

Or, you can take a shortcut to any topic by typing it along with HELP (HELP ADD, for example).

Not all areas offer so many subtopics. At the Database topic prompt, for example, you'll see this when you type HELP:

DATABASES

This is the central place within the GROUP in which users may find public domain software submitted by other users. The Databases will house all the GROUP's Newsletters, Journals, Walkthru's, Reviews, Programs, and Documentation. All material(s) must be submitted for approval by the GROUP MANAGER before it is posted. All users are encouraged to share their PUBLIC DOMAIN software and other files with the GROUP.

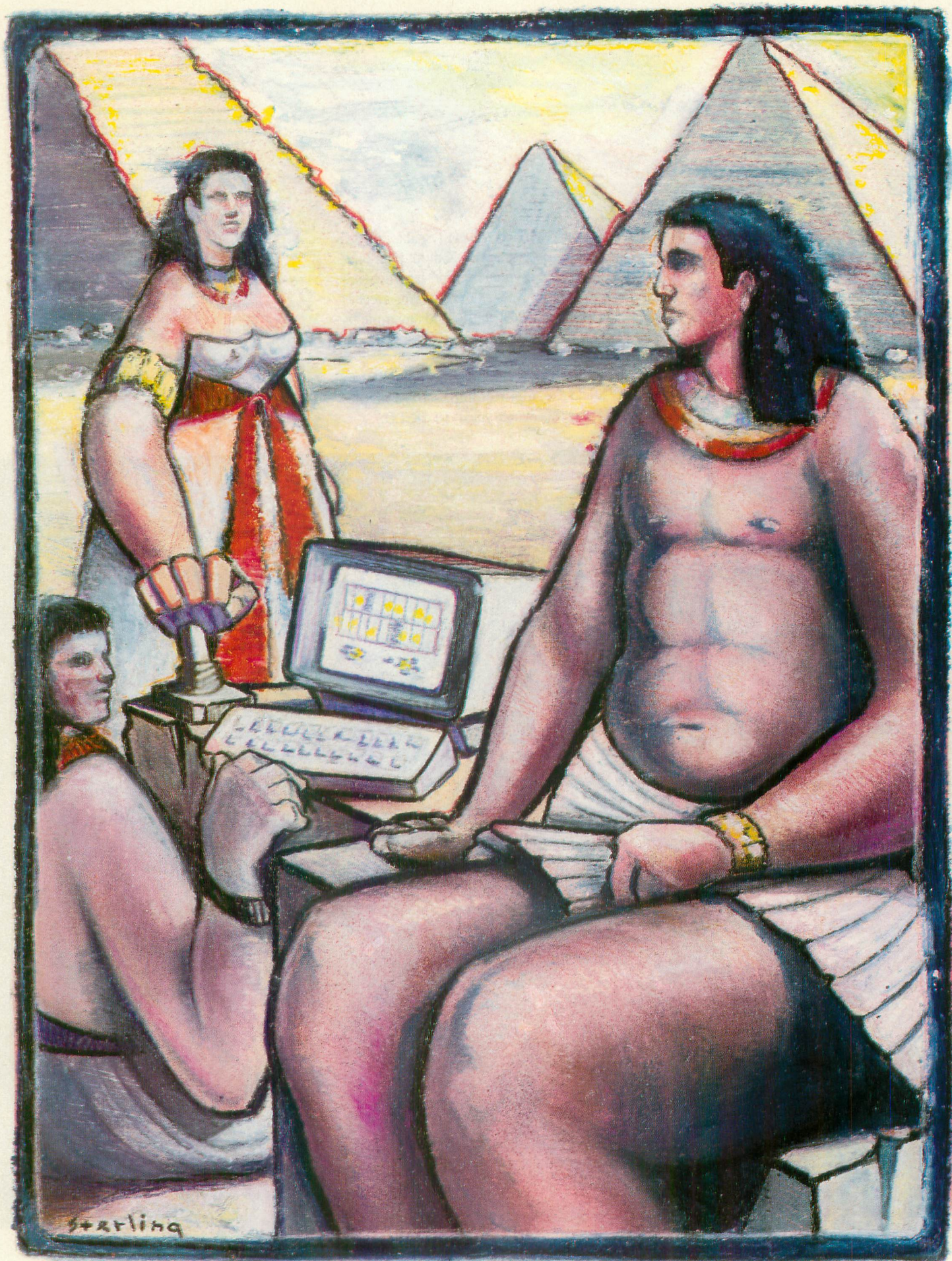
Additional information available:

DATABASE_NAME

Enter desired subtopic or CTRL/Z to EXIT.

GROUPS DATABASES Subtopic? *continued on page 23*





Pebbles

by Clive King

Thousands of years ago, long before playing cards, dice and video games, people entertained themselves using whatever items they could find laying around. Unfortunately, not all of the world's locales offered as much a variety of natural resources as others. For instance, in ancient Egypt, smack in the middle of the desert, they had to be a little more creative in coming up with gaming materials than, say, the American Indians, who lived in the lush forests of North America. Everywhere those poor Egyptians looked, it was sand and stone, sand and stone.

Not to be put off by such petty matters, the Egyptians came up with a game called "Oh-Wa-Ree" (the spelling varies wildly, depending on the source), which was played with nothing more than a bunch of pebbles and some pits dug in the sand. So popular did this game become that even today we remember it.

Pebbles is a version of this ancient Egyptian game for your Atari computer. No doubt, once you begin to appreciate the hidden complexities of this basically simple game, you'll be as fascinated with it as millions of people down through the ages have been.

Typing it in

Type in Listing 1, using the BASIC Editor II found elsewhere in this issue to check your typing. Then save the program to disk or tape.

Playing the game

When you first run the program, the title screen will appear, and you'll be asked if you'd like to see instructions. Type a "Y" followed by Return if you wish a brief recap of the rules, or type "N" followed by Return if you're ready to play.

After the title screen, the game board will be drawn. At the top of the screen, you will see the game options you may set. Use the Select key to set the difficulty (four one-player games against the computer only) from one to four (four is the hardest) and the Option key to set the number of players (one or two). When you have the options set the way you want them, press Start, and the game will begin.

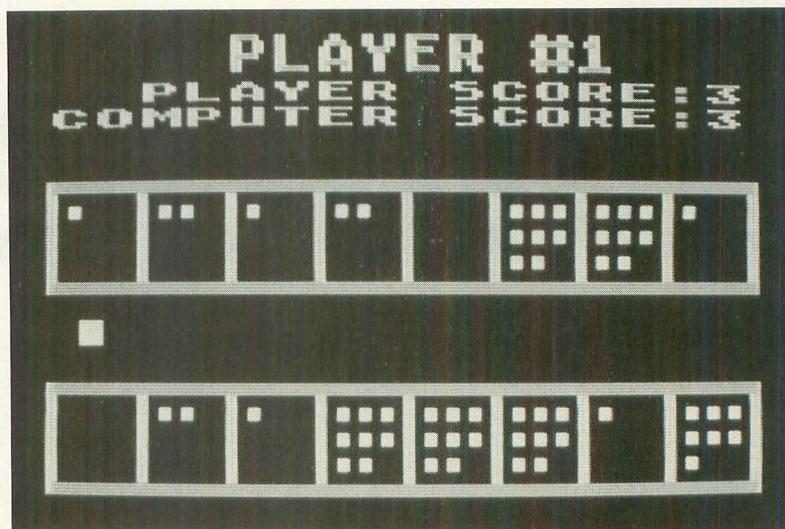
You choose which pebbles to move using the joystick to control the on-screen cursor. Press the joystick trigger when the cursor is beneath the pit of your choice.

The rules

The playing board consists of two rows of

In ancient Egypt, smack in the middle of the desert, they had to be a little more creative in coming up with gaming materials.

*No doubt, once you
begin to appreciate
the hidden
complexities of this
basically simple
game, you'll
become fascinated
with it.*



eight pits each, the top row belonging to Player 1 and the bottom row belonging to Player 2 (or the computer if you've chosen a one-player game). Each pit starts off with four pebbles. The object of the game is to maneuver the pebbles in such a way as to capture more of them than your opponent.

A move consists of "picking up" the pebbles from a pit in your row, and then "sowing" them, one by one, in each succeeding pit until they've all been played. If the pit in which you end was empty before you dropped your last pebble into it, your turn is over. If the pit contains exactly three pebbles (two plus the last one you sowed), you capture the pebbles in that pit, and they are added to your score. If the pit you finish on already contained other pebbles (unless, of course, the total is three), you must pick them all up and continue on, sowing the pebbles around the board until you run out again.

As play progresses, if the total number of pebbles in any pit is brought up to three, the pebbles in that pit are immediately captured by the player in whose row the pit resides (unless the play ended in the pit).

When you end in an empty pit, it is your opponent's turn. The play continues in this manner until there are less than six pebbles left in the pits, at which point the game is over, and the player with the highest score wins.

Note that, in the original version of Oh-Wa-Ree, the pebbles were sown in a circular fashion, clockwise. In this version, the pebbles are sown from left to right in each row. When a pebble is sown in the last pit of the bottom row, the play continues at the first pit of the top row.

Also, you should note that nine is the maximum number of pebbles that can be shown in any one pit. However, there may be many more than that actually there. There's a good reason for this: In the original Egyptian version, players were not allowed to touch the stones in any way unless they were picking them up for a move. Therefore, if a pit became very full, it was difficult to tell exactly how many pebbles it actually contained. In the Atari version, showing a maximum of nine pebbles simulates, to an extent, this extra complication.

Pebbles is truly an intriguing game. Don't be surprised if you find yourself all alone, late at night, endlessly sowing pebbles. It'll hook ya!

Clive King has owned his Atari computer for over eight years, and over that time, has written dozens of programs. An avid reader of science fiction, fantasy, horror and mainstream historical novels, he has long been fascinated by ancient Egyptian culture.

LISTING 1: BASIC

```

WN 0 REM *****
VY 1 REM *          PEBBLES          *
SF 2 REM *          BY CLIVE KING    *
ZC 3 REM *          *                *
BS 4 REM *          COPYRIGHT 1989   *
PL 5 REM *          BY ANALOG COMPUTING *
WT 6 REM *****
PG 10 GOTO 740
CK 30 FOR Z=N1 TO 20:NEXT Z:RETURN
ZK 40 FOR Z=N1 TO 500:NEXT Z:RETURN
VY 50 POKE 87,N5:PLOT C1,R1:PLOT C1+N1,R1
:PLOT C1,R1+N1:PLOT C1+N1,R1+N1:RETURN
KK 60 IF FL AND A=N0 THEN RETURN
AU 70 IF FL THEN NUM=A:GOTO 90
ZN 80 NUM=BOX(CH)
BK 90 IF CH<N9 THEN C=CH*N8+N2:R=N8
AP 100 IF CH>N8 THEN C=(CH-N8)*N8+N2:R=24
AX 110 POKE 87,N5
QR 120 IF FL THEN R=R+N8
DY 130 IF NUM>N9 THEN NUM=N9
JL 140 FOR X=N1 TO NUM:IF X=N4 OR X=N7 TH
EN R=R+N2:C=C-N6
QA 150 SOUND N0,100,10,N4:PLOT C,R:GOSUB
DLAY:SOUND N0,N0,N0,N0:C=C+N2:NEXT X:R
ETURN
XM 155 FOR Z=240 TO 0 STEP -20:SOUND 0,Z,
10,8:SOUND 1,Z+5,10,8:POKE 712,Z:NEXT
Z:SOUND 0,0,0,0:SOUND 1,0,0,0:RETURN
NA 160 IF STRIG(N0)=N0 AND BOX(CH)>N0 THE
N COLOR N4:GOSUB CUR:GOSUB MV:PT=N0:RE
TURN
XH 170 ST=STICK(N0):IF ST<>N7 AND ST<>11
THEN 160
KH 180 COLOR N4:GOSUB CUR:IF ST=N7 THEN C
H=CH+N1:C1=C1+N8
LA 190 IF ST=11 THEN CH=CH-N1:C1=C1-N8
AJ 200 IF P=N1 AND CH>N8 THEN CH=N8:C1=C1
-N8
HS 210 IF P=N1 AND CH<N1 THEN CH=N1:C1=C1
+N8
RZ 220 IF P=N2 AND CH>16 THEN CH=16:C1=C1
-N8
YP 230 IF P=N2 AND CH<N9 THEN CH=N9:C1=C1
+N8
NY 240 COLOR N2:GOSUB CUR:GOSUB DLAY:GOTO
160
PX 250 SOUND N0,100,10,N8:SOUND N1,135,10
,N8:GOSUB DLAY:SOUND N0,N0,N0,N0:SOUND
N1,N0,N0,N0:RETURN
UJ 260 IF BOX(CH)<N1 THEN 538
JQ 262 POKE 77,0
EI 265 A=BOX(CH):GOSUB PLT:FL=N1:COLOR N2
:GOSUB PLT:BOX(CH)=N0:GOSUB DLAY
PL 270 COLOR N4:GOSUB PLT:FL=N0
CO 280 CH=CH+N1:IF CH>16 THEN CH=N1
OY 290 A=A-N1:BOX(CH)=BOX(CH)+N1:COLOR N2
:GOSUB PLT:FL=N1:GOSUB PLT:GOSUB DLAY:
COLOR N4:GOSUB PLT:FL=N0
SW 300 IF A=N0 AND BOX(CH)=N1 THEN RETURN
NR 310 IF A<>N0 THEN 320
KG 315 A=BOX(CH):FL=1:COLOR N2:GOSUB PLT:
FL=N0:COLOR N4:GOSUB PLT:GOSUB DLAY:BO
X(CH)=N0:FL=1:COLOR 4:GOSUB PLT:FL=0
TZ 320 IF NP<>N1 OR BOX(CH)<>N3 OR CH<N8
THEN 330
WY 325 C5=C5+N3:GOSUB 503:POKE 87,N2:POSI
TION 5C,N2:? #N6;C5:POKE 87,N5:GOTO 38
0
JP 330 IF BOX(CH)<>N3 OR CH<N8 OR NP<>N2
THEN 340

```

```

QJ 335 P5(N2)=P5(N2)+N3:GOSUB 503:POKE 87
,N2:POSITION 5C,N2:? #N6;P5(N2):POKE 8
7,N5:GOTO 380
KN 340 IF BOX(CH)<>N3 OR CH=N9 THEN 350
KS 345 P5(N1)=P5(N1)+N3:GOSUB 503:POKE 87
,N2:POSITION 5C,N1:? #N6;P5(N1):POKE 8
7,N5:GOTO 380
IM 350 IF BOX(CH)=N0 AND A=N3 THEN FL=N1
GN 355 IF FL AND CT THEN C5=C5+N3:GOSUB 5
03:POKE 87,N2:POSITION 5C,N2:? #N6;C5:P
OKE 87,N5:GOSUB PLT:FL=N0:RETURN
ME 360 IF NOT FL OR NOT PT THEN 280
FL 365 P5(P)=P5(P)+N3:GOSUB 503:POKE 87,N
2:POSITION 5C,P:? #N6;P5(P):POKE 87,N5
:GOSUB PLT:FL=N0:RETURN
PQ 370 GOTO 280
KN 380 GOSUB PLT:BOX(CH)=N0:GOTO 280
EJ 390 SOUND N0,80,12,N8:GOSUB LDLAY:SOUN
D N0,N0,N0,N0:RETURN
MY 400 T=N0:FOR Z=N1 TO 16:T=T+BOX(Z):NEX
T Z:IF T>N6 THEN RETURN
ML 410 POKE 87,N2:POSITION N1,N0:? #N6;"
GAME OVER! ";GOSUB 502:GOSUB L
DLAY
HH 420 GRAPHICS 18:POSITION N7,N2:? #N6;"
SCORES":POSITION N7,N3:? #N6;"-----"
HB 430 POSITION N3,N5:? #N6;"PLAYER 01: "
:P5(N1):POSITION N3,N6:? #N6;"PLAYER 0
2: ";P5(N2)
KV 435 IF NP=N1 THEN POSITION N3,N6:? #N6
;" COMPUTER: ";C5
AT 440 IF PEEK(53279)<>N6 THEN DIF=1:GOTO
440
PJ 445 GOTO 810
XE 450 GOSUB CHK
QA 455 FOR P=N1 TO NP:POKE 87,N2:POSITION
N0,N0:? #N6;" PLAYER #";P;"
";C1=11
FT 460 IF P=N1 THEN R1=17:CH=N1:T=N0:FOR
Z=N1 TO N8:T=T+BOX(Z):NEXT Z:IF T=N0 T
HEN GOSUB DLAY:GOTO 490
WV 470 IF P=N2 THEN R1=33:CH=N9:FOR Z=N9
TO 16:T=T+BOX(Z):NEXT Z:IF T=N0 THEN G
OSUB DLAY:GOTO 490
NT 480 PT=N1:COLOR N2:GOSUB CUR:GOSUB STK
:GOTO 500
VU 490 POSITION N0,N0:? #N6;" NO MO
VE ";GOSUB 502:GOSUB LDLAY:NEXT
P
FA 500 IF NP=N2 THEN NEXT P
CG 510 IF NP=N2 THEN 450
IE 520 GOSUB CHK:POKE 87,N2:POSITION N2,N
0:? #N6;"COMPUTER THINKING":CT=N1:CH=N
9:H=-30
GR 530 T=N0:FOR Z=N9 TO 16:T=T+BOX(Z):NEX
T Z
HV 531 IF T=N0 THEN POSITION N0,N0:? #N6;"
NO MOVE ";GOSUB 502:GOSUB
LDLAY:GOTO 450
GZ 535 T=N0:FOR Z=N9 TO DIF:T=T+BOX(Z):NE
XT Z:IF T=N0 THEN 538
PR 537 GOTO 540
GV 538 FOR Z=N9 TO 16:IF BOX(Z)>N0 THEN C
H=Z:POP :GOTO 730
NQ 539 NEXT Z
EW 540 FOR X2=N9 TO DIF:TC5(X2)=N0:IF BOX
(X2)=N0 THEN NEXT X2:GOTO 720
TM 550 FOR X1=N1 TO 16:BX(X1)=BOX(X1):NEX
T X1
UB 560 A=BOX(X2):Y=X2:Y1=N0:BX(Y)=N0
HV 580 A=A-N1:FL1=N0:POKE 53279,N0
VY 590 Y=Y+N1:IF Y>16 THEN Y=N1
RV 600 IF Y=X THEN Y1=N1
MO 610 BX(Y)=BX(Y)+N1:IF A=N0 THEN FL1=N1
:IF BX(Y)=N1 THEN 710

```



```

TE 620 IF FL1 THEN A=BX(Y):BX(Y)=N0
GV 623 IF BX(Y)=N0 AND A=N3 AND Y<N9 THEN
    TC5(X2)=TC5(X2)+N5:GOTO 710
NC 626 IF BX(Y)=N0 AND A=N3 THEN TC5(X2)=
    TC5(X2)+N3:GOTO 710
TB 630 IF BX(Y)=N1 AND Y>N8 THEN TC5(X2)=
    TC5(X2)+N1:GOTO 580
VX 640 IF BX(Y)=N1 AND Y<N9 THEN TC5(X2)=
    TC5(X2)-N1:GOTO 580
WR 650 IF BX(Y)=N2 AND Y>N8 THEN TC5(X2)=
    TC5(X2)+N2:IF Y1 THEN TC5(X2)=TC5(X2)-
    N1:GOTO 580
KR 660 IF BX(Y)=N2 AND Y<N9 THEN TC5(X2)=
    TC5(X2)-N2:IF Y1 THEN TC5(X2)=TC5(X2)+
    N1:GOTO 580
SW 670 IF BX(Y)=N3 AND Y>N8 THEN TC5(X2)=
    TC5(X2)+N3:BX(Y)=N0:IF Y1 THEN TC5(X2)=
    TC5(X2)-N2:GOTO 580
QX 680 IF BX(Y)=N3 AND Y<N9 THEN TC5(X2)=
    TC5(X2)-N3:BX(Y)=N0:IF Y1 THEN TC5(X2)=
    TC5(X2)+N2
QK 700 GOTO 580
GX 710 NEXT X2
CP 720 FOR X=N9 TO DIF:IF TC5(X)>H THEN H
    =TC5(X):CH=X
MK 725 NEXT X
VI 730 POSITION N2,N0:?"#N6;"
    ":GOSUB MV:CT=N0:GOTO 450
UO 740 DIM A$(1):N0=0:N1=1:N2=2:N3=3:N4=4
    :N5=5:N6=6:N7=7:N8=8:N9=9:SC=17:GRAPHICS
    N0:SETCOLOR N2,N0,N0
VN 750 POKE 752,N1:DL=PEEK(560)+PEEK(561)
    *256:DL=DL+N4:POKE DL+N5,N7:POKE DL+N6
    ,N7:POKE DL+N8,N6:POKE DL+N9,N6
PF 760 POKE DL+28,65:POKE DL+29,PEEK(560)
    :POKE DL+30,PEEK(561):?"CHR$(125)
EJ 770 POSITION N6,N4:?"PEBBLES":POSITIO
    N N2,N6:?"an ancient game"
CT 780 POSITION 24,N6:?"of strategy":POS
    ITION 13,10:?"BY CLIVE KING"
KZ 782 POSITION N7,20:?"DO YOU NEED INST
    RUCTIONS?":INPUT A$:IF A$="Y" THEN 982
ZB 790 DLAY=30:PLT=60:SO=250:MV=260:SO2=3
    90:LDLAY=40:CHK=400:SDLAY=20:CUR=50:NP
    =N1:DIF=N1:STK=160:SO3=155
IQ 800 DIM BOX(16),BX(16),P5(N2),TC5(16)
RM 810 FOR X=N1 TO 16:BOX(X)=N4:NEXT X:P5
    (N1)=N0:P5(N2)=N0:CS=N0
QH 820 GOSUB DLAY
HD 830 GRAPHICS 5:DL=PEEK(560)+PEEK(561)*
    256:DL=DL+N4:POKE DL-N1,71:POKE DL+N2,
    N6:POKE DL+N3,N6:POKE DL+N4,N6
NM 840 POKE DL+41,65:POKE DL+42,PEEK(560)
    :POKE DL+43,PEEK(561)
VZ 850 COLOR N1:PLOT N8,N6:DRAWTO 72,N6:P
    LOT N8,14:DRAWTO 72,14:PLOT N8,22:DRAW
    TO 72,22:PLOT N8,30:DRAWTO 72,30
UX 860 FOR X=N8 TO 72 STEP N8:PLOT X,N6:D
    RAWTO X,14:PLOT X,22:DRAWTO X,30:NEXT
    X
MA 870 FOR CH=N1 TO 16:COLOR N2:GOSUB PLT
    :NEXT CH
WK 880 POKE 87,N2:POSITION N4,N1:?"#N6;"#
    OF PLAYERS":POSITION N6,N2:?"#N6;"DI
    FFICULTY:"
IC 890 POSITION N0,N0:?"#N6;"press start
    to begin":POSITION N1,N3:?"#6;"use opt
    ion & select"
IC 900 A=PEEK(53279):IF A=N6 THEN 932
VG 910 IF A=N3 THEN NP=NP+N1:IF NP>N2 THE
    N NP=N1

```

```

HT 920 IF A=5 THEN DIF=DIF+N1:IF DIF>N4 T
    HEN DIF=N1
RM 930 POSITION 17,N1:?"#N6;NP:POSITION 1
    7,N2:?"#N6;DIF:GOSUB DLAY:GOTO 900
FF 932 IF NP=N1 THEN 940
OF 933 POSITION N3,N1:?"#N6;" player #1:
    0 "
LE 934 POSITION N3,N2:?"#N6;" player #2:
    0 ":GOTO 960
EZ 940 POSITION N4,N1:?"#N6;"player score
    0"
VR 950 POSITION N2,N2:?"#N6;"computer sco
    re:0"
PJ 960 POSITION N1,N3:?"#N6;"
    "
CM 970 DIF=DIF*N2+N8:IF NP=N2 THEN SC=15:
    GOTO 450
PY 975 GOTO 450
YT 982 POSITION N2,10:?"The object is to
    capture the majority of the 'pebbles'
    . Use the joystick to"
HX 984 ? "place the cursor under the 'pit
    ' of your choice, then press the tri
    gger."
NT 986 ? "The contents of that pit will t
    hen":GOSUB 1100
BG 988 ? "be picked up and distributed on
    e by one into the other pits from le
    ft to "
AP 990 ? "right. When the last pebble is
    'sown' the contents of that pit are ta
    ken "
FE 992 ? "and sown in the same manner. Th
    is ":GOSUB 1100
TJ 994 ? "continues until the last pebble
    either lands in an empty pit or
    ends "
FU 996 ? "in a pit bringing the total to
    three. If the later, the pebbles are t
    hen "
WX 998 ? "'captured' and added to the pla
    yer's":GOSUB 1100
MY 1000 ? "score. During a turn if any pi
    t is brought up to three (without e
    nding "
MK 1002 ? "in that pit) the pebbles are c
    aptured by the player in whose territo
    ry they"
CL 1004 ? "reside. The top row is player
    #1's ":GOSUB 1100
SJ 1006 ? "territory. The bottom is playe
    r #2's or the computer's. A player al
    ways "
JR 1008 ? "starts his turn with a pit in
    his own territory. When the total
    # of "
QP 1010 ? "pebbles left is less than 6 th
    e game":GOSUB 1100
NP 1012 ? "is over, and the scores will b
    e dis- played. To play again press st
    art. "
WJ 1014 ? "By the way, each pit will show
    a max- imum of 9 pebbles, but thee ma
    y be"
MG 1015 ? "more!
    "
OE 1016 POSITION 14,19:?"GOOD LUCK!":GOS
    UB 1100:GOTO 790
QM 1100 POSITION 11,20:?"PRESS <RETURN>
    ";INPUT A$:POSITION N2,10:RETURN

```


continued from page 17

Help on Other Topics and "Help Help"

You can also get help on topics outside the current area at many prompts (such as the DELPHI Main Menu) by typing HELP and the subject of interest. (You can even get help with the Help system by typing HELP HELP at these prompts.)

Now, let's take a look some special help systems in the Atari SIG.

Helpful Hints

You've seen the HELP selection on the Atari SIG main menu, but many have figured it was a command, a reminder that help is available via the online Help system. Here, however, it's not. On the SIG menu, HELP is a selection that leads to "Helpful Hints," a collection of mini-articles on using the SIG.

Using "Helpful Hints"

When you type HELP at the Atari SIG menu, you don't see a menu, but this:

```
ANALOG>What do you want to do? HELP
19 hints available.
```

```
HINTS>(Scan, Read, "?" or Exit):
```

But don't panic; you need only type ? to see the menu:

Helpful Hints Menu:

SCAN	Table of Contents	NEXT	Item
READ	Item(s)	BACK	to Previous Item
HELP	with Commands	CLEAR	
EXIT	to Previous Menu	DOWNLOAD	Item(s)

```
HINTS>(Scan, Read, "?" or Exit):
```

SCAN lists the table of contents, a form of help in itself:

```
HINTS> SCAN
```

Contents

```
-----
1 >>> READ ME FIRST <<<
2 ANALOG SIG COMMANDS
3 DATABASE COMMANDS
4 DOWNLOADING
5 FORMAL CONFERENCE PROTOCOL
```

```
6 HOW TO USE CONFERENCE
7 L. USING THE FORUM
8 M. THE MAIL SYSTEM
9 N. CONFERENCE QUICK REFERENCE
10 O. FORUM QUICK REFERENCE
11 P. FORUM MESSAGE FORMATTING
12 Q. MAIL QUICK REFERENCE
13 R. /SLASH COMMAND SUMMARY
14 S. TEXT EDITING WITH "OLDIE"
15 SEARCHING THE DATABASES
16 TEXT UPLOADING
17 THE FILE DATABASES
18 USING YOUR WORKSPACE
19 XMODEM UPLOADING
```

Type READ alone to read the first article, or READ xx (where "xx" is any article's number) to read a specific article. After an article is displayed, press Return to read the next article (or, if you prefer, you can type NEXT; the result's the same).

The DOWNLOAD command operates the same as the READ command, and HELP, of course, will display helpful information on using this area.

Forum Help

The Atari SIG Forum is another area where special help is found in the form of a menu display. You don't normally see a menu when you're in the Forum, even if your prompt level is set at "Full" or "Menu." However, you can see the Forum menu by typing ? (just like in the HELP area), like this:

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

FORUM Menu:

ADD New Message (Thread)	FORWARD Message by Mail
REPLY To Current Message	DELETE Message
READ Message(s)	EDIT a Posted Message
FOLLOW Thread	NEXT Message
BACK to Previous Message	TOPICS (Set/Show)
DIRECTORY of Messages	HIGH Message (Set/Show)
MAIL	HELP
TAG Interesting Message	EXIT
FILE Message Into Workspace	

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

That's a fairly complicated menu, but you can get help with any item by typing HELP (as in the example at the beginning of this column).

**DATABASE
DELPHI**

DATABASE DELPHI

Slash Commands

Yet another special help feature has to do with "slash commands"—i.e., commands preceded by a slash (/). (These are more formally known as "Immediate Commands." They deal mainly with how DELPHI communicates with you and with accessing information that has nothing to do with the current menu or prompt.

Slash commands are, by the way, available at virtually all prompts, anywhere on DELPHI.

Appropriately enough, you type /HELP for help with slash commands, like this:

```
- Immediate "/" Commands -
/EXIT - same as CTRL/Z
/BUSY - disables conference messages
/LENGTH 24 - sets page size to 24
/NOBUSY - re-enables conference messages
/PROMPT - sets prompt mode to MENU, VERBOSE, or BRIEF
/TIME - show time
/WIDTH 80 - sets terminal width to 80
/WHOIS username
- Use /HELP FULL to get a complete list.
```

But that's just a partial list. If you want to see the full list, do what the sign says and use /HELP FULL to get a complete list, which goes like this:

```
- Immediate "/" Commands -
/EXIT - same as CTRL/Z
/PAD - set/show network params
/TIME - show time
/WHOIS username
/SEND - send message to someone now
/WHO - list people in this area
/MAIL - read or send E-mail message
/ENTRY - display entry log
/DATE - display day of week and whether it is a holiday
/[NO]AUTOPAD - Whether PADTYPE is picked each login
/[NO]BUSY - shut off conference pages
/DOWNLOAD_EOL - change text download line terminators
/ECHO - controls the echoing of characters
/EDITOR EDIT or OLDIE
/FX_METHOD - preferred file transfer method
/[NO]HIGH_BIT - high bit handling during XMODEM or Kermit text downloads
/[NO]KER_7BIT - Tells Kermit to only use SEVEN bits of each byte.
/KER_ERR - selects the Kermit error checking mode
/KER_RETRY 15 - sets the Kermit retry maximum to 15
/KER_TIMEOUT 10 - sets your Kermit timeout to 10 seconds
/LENGTH 24 - sets screen length to 24 lines
/[NO]OLDIE - Initialization for OLDIE text editor
/[NO]PADSETUP - /PAD command to use upon login
/PADTYPE - Defined at Login unless AUTOPAD is FALSE (0)
/PROMPT - sets mode BRIEF, VERBOSE, or MENU
/[NO]TERM - set or show terminal parameters
/TERM_TYPE - VT100, VT52, or UNKNOWN
/TIMEOUT - number of minutes we wait for input before logging you off
/WIDTH 80 - sets your terminal width to 80 columns
/[NO]XMODEM - Use windows for XMODEM uploads
/[NO]XM_CRC - selects CRC or Checksum XMODEM uploading
/XM_KEELOG - Does system keep error logs? NORMAL, NEVER, ALWAYS.
/XM_LASTBLOCK - NORMAL, NO-Control-Z, or ATARI-Mode
/XM_RETRY 20 - sets the XMODEM retry maximum to 20
/XM_TIMEOUT 10 - sets the XMODEM timeout to 10 seconds
/SAVE x - save current settings in profile x (0-9)
/RESTORE x - restore settings from profile x (0-9)
```

Explaining all of *that* is beyond the scope of this column, but most slash commands are self-explanatory. For more information, look

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

ST owners! Ask about Ultrabase ST (B/W 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

CIRCLE #102 ON READER SERVICE CARD.

Make the DELPHI Connection!

As a reader of *ANALOG Computing*, you are entitled to take advantage of a special DELPHI membership offer. For only \$19.95 plus postage and handling (\$30 off the standard membership price!), you will receive a lifetime subscription to DELPHI, a copy of the 500-page *DELPHI: The Official Guide* by Michael A. Banks and a credit equal to one free evening hour at standard connect rates. Almost anyone worldwide can access DELPHI (using Tymnet, Telenet or other networking services) via a local phone call. Make the DELPHI connection by signing up today!

To join DELPHI:

1. Dial 617-576-0862 with any terminal or PC and modem (at 2400 bps, dial 576-2981).
2. At the Username prompt, type JOINDELPHI.
3. At the Password prompt enter ANALOG.

For more information, call DELPHI Member Services at 1-800-544-4005, or at 617-491-3393 from within Massachusetts or from outside the U.S.

DELPHI is a service of General Videotex Corporation of Cambridge, Massachusetts.

up slash commands in *DELPHI: The Official Guide*, or check the article on slash commands in "Helpful Hints." (Note: Most of these commands you'll never use. The most frequently-used slash commands are: /DATE, /ECHO, /EXIT, /BUSY, /MAIL, /PROMPT, /SEND /TIME, /WHO and /WHOIS. (Note that any slash command that "toggles" a feature from one state to another—like /BUSY—can be toggled back to its original state by preceding the command with NO (thus, /BUSY can be turned "off" by typing /NOBUSY).

Conference Help

The Help system in Conference has to be special, since most anything you type while in a conference group is displayed. Conference conveniently uses slash commands for HELP (and commands). Type /HELP while in a conference group, and you'll see this:

```
/HELP
Immediate Commands:
/JOIN <groupname or #> /EXIT /BYE
/NAME <name> /RNAME <name> /SHOWRN
/SEND <name> <message> /BUSY /SQUELCH
/WHO /MAIL /WHOIS <name> /ENTRY <name>
/PAGE <name> /ANSWER /CANCEL /REJECT
/GNAME <groupname> /TALK <gname or #>
/GPRIVATE /GPASS /GLOCK /GROLL /GQUIET
/PASS /PROLL /ROLL /<#> <message>
/ECHO /REPEAT /WIDTH /TIME /KEYSUB
/PROMPT /PAD /PORT /TERM /QUEUE
/LOG /DIRECTORY /DISPLAY /WAIT /HELP
/XMOD /XGLISTEN /XLISTEN /XPUB /XGOWN
for more help, example:
/HELP /SQUELCH
```

Some commands are reversed by including
"NO" in the command, e.g., /NOECHO
For other commands, type /HELP MORE

As noted in the display, you can type /HELP followed by the name of a command for detailed information on the command, and /HELP MORE for additional commands.

(And, hey—speaking of Conference, don't forget the Tuesday night real-time conference in ANALOG's Atari SIG! Log on at 10 P.M., EST, type CO, then WHO and JOIN followed by the name of the conference. The Atari Users' Group conference is open to everyone. Come on by, meet other Atari users, and practice what you've been learning from this column.)

Other Sources of Help

Menus: The most basic form of help is viewing menus when you're in unfamiliar areas. To do this, make sure your prompt level is set to the appropriate level (if you need to see menus, type /PROMPT = MENU at any prompt). Or, if you're running with the prompt set at BRIEF or VERBOSE, remember that you can type MENU to see a menu at any time.

People: Don't forget DELPHI's people resources! You can get help from other users by leaving a query in the Forum. E-mail to SYSOP, SERVICE or me can be used to get help with specific problems.

Articles: You'll find a number of mini-articles about using DELPHI in the Atari SIG's GENERAL database. Type DA GEN, then type READ ATARI SIG TUTORIAL. These contain all sorts of information about using the wealth of features the SIG has to offer.

DELPHI: The Official Guide: And, finally, you might want to get a copy of DELPHI's offline help system, *DELPHI: The Official Guide*, if you don't already have one.

Here's a quick reminder to those of you who haven't yet joined DELPHI and are missing such benefits as keeping in touch with other Atari users and downloading the ANALOG Computing program listings (a major convenience, when you consider the time you spend keyboarding programs—and correcting errors). You can sign up right now: see the accompanying sidebar for online sign-up information.

That's it for now. Next month: Forum in depth. Until then, see you online!

In addition to having published science fiction novels and books on rocketry, Michael A. Banks is the author of DELPHI: The Official Guide and The Modem Reference—both from Brady Books/Simon & Schuster. Look for his general articles on telecommunications and tips on using DELPHI in the Atari Users' Group databases. You can contact Banks to exchange weather reports and other information on DELPHI by sending E-mail to membername KZIN.

Master Memory Map, Part VIII

by Robin Sherer

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.

Special chips and ROM

Okay, we're now done with all the RAM locations. That leaves the ROM and the special chips that cover the GTIA (or CTIA), POKEY, PIA and ANTIC chips along with the OS ROM. We'll learn what each of the chips does as we come across it.

ROM, of course, stands for Read Only Memory, which means that you can't store values in it. You can't store values in the chips either, but as you read through the following locations, you'll notice that I tell you to POKE to them. What's going on here? When you POKE a value in a chip location, the chip will see the value and act accordingly, but won't store it anywhere. That means that if you POKE some location with a value and then PEEK that location, you won't necessarily get the value you POKEd. It's okay though, because the chip knows what you were trying to do and acts as though the value is in the location. Confused? Just remember that POKE works, but PEEK won't always.

Because the value you POKE is different from the value you PEEK, a lot of the

ROM locations have different meanings depending on whether you're PEEKing or POKEing. In such a case, I'll give you an explanation.

Now that you think you understand, here's one more thing to try and trip you up. You can't POKE to the OS ROM at all. Well, you can if you want to, but nothing will happen.

Let's do a little memory mathematics. First of all, let's start with a 48K Atari. It's actually a 64K Atari. Where's the other 16K? The OS ROM takes up 10K, so that leaves 6K unaccounted for. The chips mentioned take up 1.25K, so we're down to 4.75K. Would you believe that 4.75K of memory is unused? Well, you should, because it is. Unfortunately, it's all in ROM, and therefore, you can't use it either. Exactly 4K of it is right here at Locations 49152 through 53247 (\$C000 through \$CFFF). The other .75K, which, by the way, is equal to three pages, since a page is .25K, is found amongst the chips at Locations 53504 through 53759 (\$D100 through \$D1FF) and 54784 through 55295 (\$D600 through \$D7ff).

Anyway, that means we have 48K to program with, right? Nope. BASIC takes up 8K, as we saw; so we're down to 40K. We already saw that the first 1792 bytes (or 1.75K) are used by BASIC, the OS and the floating point package, and then on top of that there's screen memory and the memory that the FMS and DOS use if you're using a disk drive. That leaves anywhere from about 37K down to 29K or

less! Oh, well, I guess even computers can lose their memories. (Okay, okay, I'm sorry.)

GTIA OR CTIA?

Time to answer a question that may have been nagging at a lot of you. What's the difference between GTIA and CTIA? And for that matter, what is a CTIA/GTIA? Let's start with the second question first. Way back when the Atari computers were still being designed, they had nicknames. The 400 was called Candy, and the 800, Colleen. So much for trivia. Both Colleen and Candy needed some way of talking to the television set, so a television interface adapter chip was designed. That's where you get the name CTIA from (Colleen/Candy Television Interface Adapter). The CTIA chip gets information from ANTIC and uses it to tell the television set what to do. So much for that.

When the CTIA was originally designed, it was supposed to support 12 graphics modes (0 to 11). Unfortunately, the last three modes couldn't be implemented in time to make the production deadline. Rather than hold off releasing the computers, Atari decided to put them out without those three modes and add the modes later. So that's why the early computers don't support modes 9, 10 and 11. After the first batch of CTIAs ran out, Atari started putting in chips that had the extra three modes; for some reason calling them GTIAs, with the "G" standing for George. Maybe George was the guy

who finally figured out how to get the last modes to work. I don't know.

In any case, that's the difference between CTIA and GTIA. Oh, GTIA also corrects another problem that CTIA had. It seems that you couldn't line up players and playfields exactly (they were off by half a color clock). To correct this problem, GTIA shifts the playfield by half a color clock. Unfortunately, this means that colors obtained by artifacting (see COLOR1 at Location 709 [02C5]) will be the opposite of what they're supposed to be.

The G/CTIA takes up Locations 53248 through 53505 (\$D000 through \$D0FF).

You're now about to enter player/missile territory. That's right, G/CTIA also takes care of player/missile graphics.

HPOSPO (POKE) and MOPF (PEEK)
53248 D000

(POKE): HPOSPO specifies the horizontal location of Player 0, and can range from 0 to 227. Note that some of these positions will be off the edge of the screen, so experiment to see what values can be seen on your screen. For most television sets, anything between 48 and 208 will be visible.

Because this location is in one of the chips, you will not be able to PEEK here to find out the current position of the player. That means that you should keep track of the position in a separate variable.

(PEEK): MOPF is a collision register. Collision registers are used to tell who's collided with whom on the screen (in other words, who's sharing the same pixels with whom). They can be very useful in games and other applications. Note that the collision registers do not work properly in GTIA modes 9 through 11. You should also check HITCLR at Location 53278.

MOPF tells you what part of the playfield Missile 0 has collided with (the background is not considered part of the playfield here). Its bits have the meanings in Figure 1.

-----1	(1)	means a collision with playfield zero.
-----1-	(2)	means a collision with playfield one.
-----1--	(4)	means a collision with playfield two.
-----1---	(8)	means a collision with playfield three.

Figure 1. MOPF bit chart

HPOSPI (POKE) and M1PF (PEEK)
53249 D001

(POKE): HOPSP1 specifies the horizontal position of Player 1.

(PEEK): M1PF is the collision register for Missile 1/playfield collisions.

HPOSP2 (POKE) and M2PF (PEEK)
53250 D002

(POKE): HOPSP2 specifies the horizontal position of Player 2.

(PEEK): M2PF is the collision register for Missile 2/playfield collisions.

HPOSP3 (POKE) and M3PF (PEEK)
53251 D003

(POKE): HOPSP3 specifies the horizontal position of Player 3.

(PEEK): M3PF is the collision register for Missile 3/playfield collisions.

HPOSMO (POKE) and POPF (PEEK)
53252 D004

(POKE): HPOSMO specifies the horizontal position of Missile 0 (missiles move just like players).

(PEEK): POPF is the collision register for Player 0/playfield collisions. It works just like the missile/playfield collision registers.

HPOSMI (POKE) and P1PF (PEEK)
53253 D005

(POKE): HPOSM1 specifies the horizontal position of Missile 1.

(PEEK): P1PF is the collision register for Player 1/playfield collisions.

HPOSM2 (POKE) and P2PF (PEEK)
53254 D006

(POKE): HPOSM2 specifies the horizontal position of Missile 2.

(PEEK): P2PF is the collision register for Player 2/playfield collisions.

HPOSM3 (POKE) and P3PF (PEEK)
53255 D007

(POKE): HPOSM3 specifies the horizontal position of Missile 3.

(PEEK): P3PF is the collision register for Player 3/playfield collisions.

SIZEPO (POKE) and MOPL (PEEK)
53256 D008

(POKE): You can set the size of each player (in terms of width) to one of three possibilities, each of which is twice as wide as the one before it. A value of zero or two in this or the next three locations specifies normal width, which is eight color clocks wide (the same width as a graphics mode 2 character). Similarly, a one specifies double width and three, quadruple width.

A player is normally eight bits wide. Changing the width does not affect this but rather slows each bit two or four times in a row. See Figure 2 for an example of this.

Normal Width (zero)

For Player 0: POKE 53256,0

```
10011001
10111101
11111111
10111101
10011001
```

```
# ## #
# #### #
#####
# #### #
# ## #
```

Figure 2A. Changing player widths.

Two things to note here. First, the height of the player is not changed; and second, each player can be set separately.

Double Width (one)

POKE 53256,1

```
11000011111000011
1100111111110011
1111111111111111
1100111111110011
11000011111000011
```

```
## ##### ##
## ##### ##
#####
## ##### ##
## ##### ##
```

Figure 2B.

Quadruple Width (three)

POKE 53256,3

```
11110000000011111111000000001111
111100001111111111111100001111
11111111111111111111111111111111
111100001111111111111100001111
11110000000011111111000000001111
```

```
####          #####          ####
####          #####          ####
#####
####          #####          ####
####          #####          ####
```

Figure 2C.

(PEEK): MOPL is the collision register for Missile 0/player collisions. Its bits have the meanings in Figure 3.

-----1	(1)	means a collision with player zero.
-----1-	(2)	means a collision with player one.
-----1--	(4)	means a collision with player two.
-----1---	(8)	means a collision with player three.

Figure 3. MOPL bit chart.

Because of the way player/missile graphics are designed, there is no way to detect a collision between two missiles.

SIZEP1 (POKE) and MIPL (PEEK) 53257 D009

(POKE): SIZEP1 specifies the width of Player 1.

(PEEK): MIPL is the collision register for Missile 1/player collisions.

SIZEP2 (POKE) and M2PL (PEEK) 53258 D00A

(POKE): SIZEP2 specifies the width of Player 2.

(PEEK): M2PL is the collision register for Missile 2/player collisions.

SIZEP3 (POKE) and M3PL (PEEK) 53259 D00B

(POKE): SIZEP3 specifies the width of Player 3.

(PEEK): M3PL is the collision register for Missile 3/player collisions.

SIZEM (POKE) and P0PL (PEEK) 53260 D00C

(POKE): The missiles only have one location that is used to specify their widths, and SIZEM is it. Figure 4 shows how the bits are used.

(PEEK): Now it's time for player-to-player collision registers. POPL has exactly the same meaning as MOPL, except it detects Player 0/player collisions rather than Missile 0/player collisions. Note that a player will never collide with itself.

GRAFP0 (POKE) and P1PL (PEEK) 53261 D00D

(POKE): When G/CTIA is drawing the screen, it relies on the five GRAF registers (this one and the four following) to tell it what the players and missiles will look like (GRAF stands for "GRAFics register"). Everytime G/CTIA comes to a horizontal

position on the screen where a player/missile is supposed to be, it looks at the corresponding register to see what to put on the screen.

Now, it's probably occurred to you that the GRAF registers are only one byte long. That means that unless someone puts new values in them every time a new line on the screen gets drawn, all the players and missiles will just look like a bunch of vertical lines. So who's going to do the changing?

If you're adventurous (and good with machine language), you can do it yourself, but there's a much easier way. If we tell GRCTL (53277) and DMACTL (54272) to turn on DMA (direct memory access) for players and missiles, then ANTIC will very thoughtfully keep changing the GRAF registers for us. All we have to do is put a description in memory of how we want the players and missiles to look. See PMBASE (54279) for information on how to do that.

If you want a quick way to create some kind of vertical border, use the GRAF registers without DMA. By poking a value of 255 into GRAFO, for example, you can get a vertical band the height of the screen. Note, however, that you have to turn off DMA for all players.

(PEEK): P1PL is the collision register for Player 1/player collisions.

GRAFP1 (POKE) and P2PL (PEEK) 53262 D00E

-----00 or -----10 -----01 -----11	(0) (2) (1) (3)	 for normal width missile zero. for double width missile zero. for quadruple width missile zero.
----00-- or ----10-- ----01-- ----11--	(0) (8) (4) (12)	 for normal width missile one. for double width missile one. for quadruple width missile one.
--00---- or --10---- --01---- --11----	(0) (32) (16) (48)	 for normal width missile two. for double width missile two. for quadruple width missile two.
00----- or 10----- 01----- 11-----	(0) (128) (64) (192)	 for normal width missile three. for double width missile three. for quadruple width missile three.

Figure 4. GRAFPO bit chart

(POKE): GRAFP1 is the graphics register for Player 1.

(PEEK): P2PL is the collision register for Player 2/player collisions.

GRAFP2 (POKE) and P3PL (PEEK)
53263 D00F

(POKE): GRAFP2 is the graphics register for Player 2.

(PEEK): P3PL is the collision register for Player 3/player collisions.

GRAFP3 (POKE) and TRIG0 (PEEK)
53264 D010

(POKE): GRAFP3 is the graphics register for Player 3.

(PEEK): Remember the STRIGs back at Locations 644 through 647? Well, they were the shadow registers for the TRIGs, which work the same way. A zero means the joystick button is pressed (Joystick 0 in this case), a one means it isn't. With the TRIGs, you can make it so that if a button is pressed, TRIG will stay set to zero until you reset it. See GRCTL (53277) to find out how. Otherwise, TRIG will return to one as soon as the button is released.

For those of you who are hardware

BIT	7	6	5	4	3	2	1	0
USE	M3		M2		M1		M0	

Figure 5. GRAFM bit chart

minded, the TRIGs read the value of Pin 6 of the controller jacks.

GRAFM (POKE) and TRIG1 (PEEK)
53265 D011

(POKE): GRAFM is the graphics register for the missiles. Since missiles are only two bits wide, all four can fit into one register. The bits are assigned as in Figure 5.

If all the missiles are in the same byte, how do you change one without affecting the others? It's not easy from BASIC. You have to somehow break the byte up into four parts, change the part you want and then put it back together. In machine language, all you have to do is use AND to look at the bits you want and to clear them, and ORA to reset them. Don't forget that you can't PEEK GRAFM, so you'll have to keep track of its value in a separate vari-

able (which is taken care of for you if you're using DMA).

(PEEK): TRIG1 is the value of Joystick button 1. It has a shadow register STRIG1 at Location 285.

COLPMO (POKE) and TRIG2 (PEEK)
53266 D012

(POKE): This specifies the color and brightness of Player 0 and Missile 0. Sound familiar? That's because it has a shadow register called PCOLRO at Location 704. Look there for more information.

(PEEK): TRIG2 is the value of Joystick button 2. It has a shadow register STRIG2 at Location 286.

COLPM1 (POKE) and TRIG3 (PEEK)
53267 D013

(POKE): TRIG3 is the value of Joystick button 3. It has a shadow register STRIG3 at Location 287.

COLPM2 (POKE) and PAL (PEEK)
53268 D014

(POKE): COLPM2 is the color and brightness of Player 2 and Missile 2. It has a shadow register PCOLR2 at Location 706.

(PEEK): It would be too easy to make fun of this location, so I'll spare you. What a pal, huh? There are three types of television standards around the world: PAL, NTSC and SECAM (a television standard just specifies how the television should work). PAL is used in Europe, NTSC here in North America and SECAM in France, Russia and parts of Africa. Atari has two different versions of the computer, one for PAL and one for NTSC (so we'll ignore SECAM).

What's the difference? The PAL Ataris run about 25% faster than the NTSC Ataris. Also, PAL televisions have more scan lines than NTSC ones, which means that VBLANK occurs approximately every 1/50 of a second rather than every 1/60.

"So what?" you're saying (I can hear you). Actually, unless you're programming for a PAL Atari, nothing I said will be of any concern to you. If you are programming for a PAL, however, you should be

aware of the differences.

I almost forgot. PAL (the memory location) is used to determine what kind of Atari you have. The bits have the following meaning:

----000: (0) means that the computer is set up for PAL.

----111: (14) means that it's set up for NTSC.

Make sure you only check Bits 2, 3 and 4 (i.e., on an NTSC computer, PEEK(53268) won't necessarily give you 14; it might give you 15).

COLPM3 (POKE only)
53269 D015

COLPM3 is the color and brightness of Player 3 and Missile 3. It has a shadow register PCOLR3 at Location 707.

COLPF0 (POKE only)
53270 D016

Okay, we're out of the player/missile color registers and into the playfield color register. See the shadows at Locations 708 through 712 for more information.

COLPF0 is the color and brightness of Playfield 0 and has a shadow register COLOR0 at Location 708.

COLPF1 (POKE only)
53271 D017

COLPF1 is the color and brightness of Playfield 1 and has a shadow register at Location 709.

COLPF2 (POKE only)
53272 D018

COLPF1 is the color and brightness of Playfield 2 and has a shadow register at Location 710.

COLPF3 (POKE only)
53273 D019

COLPF1 is the color and brightness of Playfield 3 and has a shadow register at Location 711.

COLBK (POKE only)
53274 D01A

COLBK is the color and brightness of the background and has a shadow register at Location 712.

PRIOR (POKE only)
53275 D01B

PRIOR is used to select the priorities of the various objects on the screen. What? Check its shadow register GPRIOR at Lo-

When the computer is first turned on, it is not set up to accept player/missile data from ANTIC (player/missile DMA is turned off). Before you can get G/CTIA and ANTIC to talk to each other about

of the screen. That way, you won't have to look at any garbage that may appear in them before you get a chance to change the GRAFs.

GRCTL is also used to make a slight change to the joystick and paddle buttons. Normally, these buttons will give a value of zero when they are pressed, and a value of one when they're not. Sometimes, however, they may get pressed and released before you get a chance to check them. GRCTL helps you solve this problem as in Figure 8.

Whoa! Wanna translate that to English? Well, "trigger input" is just a fancy term for button value. "Latch" means that once you press a button, its value will stay zero until you clear the trigger inputs, even if the button is released before then. Ta da!

Note that you cannot latch or clear individual button values.

The joystick button values are stored in the preceding TRIG locations, and the paddle button values are in PORTA at Location 54016.

HITCLR (POKE only)
53278 D01E

After you've looked at the collision registers and seen who has run into whom, it's a good idea to set them to zero. Why? If you don't, then the old collision values will still be there the next time you check, and you won't know whether anything new has happened or not. HITCLR to the rescue. POKE any value into HITCLR and all the collision registers will be cleared (set to zero).

CONSOL
53279 D01F

(PEEK): We're going to do this one backward (PEEK before POKE), and you'll see why in a second. PEEKing CONSOL will tell you which of the buttons on the keyboard (OPTION, SELECT and START) are pressed. The bits are used as in Figure 9.

-----1	(1)	means move missile zero down one.
-----1-	(2)	means move missile one.
-----1--	(4)	means move missile two.
-----1---	(8)	means move missile three.
---1----	(16)	means move player zero.
--1-----	(32)	means move player one.
-1-----	(64)	means move player two.
1-----	(128)	means move player three.

Figure 6. VDELAY bit chart

cation 623 for an explanation.

VDELAY (POKE only)
53276 D01C

As you'll find out later, at Location DMACTL (54272), you can choose one- or two-line resolution for your players and missiles. If you choose two-line, however, you have to move things vertically two lines at a time rather than one. VDELAY can

players and missiles, you have to tell G/CTIA to accept player/missile data and ANTIC to send it. GRCTL is used to do the former (DMACTL [54272] is used for the latter). Figure 7 shows how the bits are used.

What about turning players and missiles off? Can you just POKE GRCTL with a zero? Not always; POKEing GRCTL will stop new data from coming in but will not get rid of whatever's in the GRAF

-----0	(0)	tells G/CTIA not to accept missile data.
-----1	(1)	tells it to accept missile data.
-----0-	(0)	tells it not to accept player data.
-----1-	(2)	tells it to accept player data.

Figure 7. GRCTL bit chart

-----1--	(4)	will "latch the trigger inputs."
-----0--	(0)	will clear the trigger inputs.

Figure 8. GRCTL/joystick bit chart

be used to move things down by one line. Its bits specify the object to be moved, as show in Figure 6.

To continuously move an object down, just repeat the following loop:

1. Set the appropriate bit to one.
2. Move the shape data forward one byte in memory and set the bit back to zero.
3. Go to Step 1.

To continuously move it up:

1. Move the data back one byte and set the appropriate bit to one.
2. Set the bit back to zero.
3. Go to Step 1.

GRCTL (POKE only)
53277 D01D

REGISTERS (see the preceding). That means you have to turn off GRCTL (and DMACTL) and then POKE all the GRAF registers with zeroes. If you're doing this from BASIC, you may want to POKE all the horizontal position registers with a zero first, thus moving everybody off the edge

-----0	(0)	means that START is pressed.
-----1	(1)	means it isn't.
-----0-	(0)	means that SELECT is pressed.
-----1-	(2)	means it isn't.
-----0--	(0)	means that OPTION is pressed.
-----1--	(4)	means it isn't.

Figure 9. CONSOL bit chart

(POKE): If you're looking at CONSOL from machine language, you have to first POKE it with an eight. This tells it to clear out the old values and bring in the new. Don't worry about it in BASIC.

If you POKE CONSOL with a value less than eight, the speaker inside your Atari will make a clicking sound. If you keep POKEing it, you'll get a buzz. Try this:

```
100 POKE 53279,0:GOTO 100
```

Why do you have to keep POKEing the zero over and over? Because the OS automatically stores a value of eight in CONSOL every Stage 2 VBLANK, which turns off the speaker.

You can make changes to the preceding line to get slightly different buzzes. For example:

```
100 POKE 53279,0
110 POKE 53279,8:POKE 53279,8
120 GOTO 100
```

That's all there is to the G/CTIA chip. Even though it takes up another 224 bytes (through Location 53503), they are not used at the moment. Could that mean incredible additions in the future that will allow you to do three-dimensional movement with hundreds of different colors and incredibly realistic detail? Nah!

POKEY

POKEY handles a whole bunch of stuff, including sound, the paddles, the keyboard, IRQ interrupts and serial I/O, so let's get right into it.

SOUND

BASIC, with its simple SOUND statements, doesn't even hint at the complex options available from your Atari for sound generation. That's a shame, since sound can be as important a part of your program as graphics. If you're interested in learning more about how the sound capabilities of your computer work, everything you need to know appears next. If you really want to apply these capabilities to complicated sound effects, I urge you to get your hands on a copy of *De Re Atari*. It has the most in-depth look at sound and how to program it that I've seen.

TIMERS

Hold on, what happened to sound? Don't

worry, I haven't forgotten it; the system timers mentioned back at Locations 528 through 533 share a few of the sound locations. I thought we'd get them out of the way first. Because these timers are only going to be used by those of you programming in machine language, I'll just give you a checklist of what you need to do to use them. For more information, consult good old *De Re Atari*.

1. Use AUDCTL to pick the particular clock frequency you want to use (the system timers count clock pulses).

2. Depending on which timer you're using, use AUDC1, AUDC2 or AUDC4 to set the volume for the associated audio channel to zero.

3. Similarly, set AUDF1, AUDF2 or AUDF4 to the number of pulses you want to count.

4. Make sure your interrupt routine is set up and ready to go.

5. Change VTIMR1, VTIMR2 or VTIMR4 (these start at Location 528) to point to your interrupt routine. Note that VTIMR4 will not work in the original version of the OS.

6. Use IRQEN and POKMSK (53774 and 16) to enable the timer interrupts.

7. Lastly, get the timers going by writing any value to STIMER (53769).

You should be aware that DMA, DLIs and vertical blank processing can affect the timers' performance, so don't rely on complete accuracy.

AUDF1 (POKE) and POT0 (PEEK)
53760 D200

(POKE): The BASIC SOUND statement has the following format:

SOUND VOICE, PITCH,
DISTORTION, VOLUME

VOICE is a value from zero to three. Think of the Atari as having four separate sound-effects performers inside, called Voiced 0, Voice 1, Voice 2 and Voice 3. Actually, while BASIC numbers them zero through three, POKEY prefers the more normal one through four. We'll go with POKEY, which also calls them "audio channels" rather than voices. We'll go with it on that, too.

AUDF1 has nothing to do with audio channel numbers other than the fact that it specifies the pitch value for audio channel one, so we'll start discussing pitch. Pitch is another word for frequency or note or tone, whichever you prefer. We'll use frequency. Basically, frequency describes

how "high" or "low" a sound is (you may prefer the terms "treble" and "bass"). Before we look at how this got the name frequency, we need to understand what makes a sound.

Sound of any type is nothing more than air moving in "waves." What's an airwave (uh,oh, those two words together sound familiar)? It's essentially the same as an ocean wave in that it consists of a large movement of air followed by almost no movement. A whole bunch of waves, one after the other, make up a sound, and the length of the waves, or how far apart in time they are, determines the frequency of the sound. Do you see now why it's called frequency? Because the frequency of the sound depends on how frequent the waves are.

How does the computer create sound waves? Actually, the speaker creates the waves; the computer just controls the speaker. A speaker makes the waves by moving in and out at the desired frequency (each move in and out creates one wave). To make the speaker move in and out, the computer sends it a "pulse" of electricity. A pulse is just a short burst, so this is equivalent to quickly flicking a switch to the speaker on and then off again. This moves the speaker out and then in, thereby creating the air movement we need. All this work just for one little airwave!

Our next step is to look at where the pulses come from. The frequency of the pulses will determine the frequency of the sound, so it's safe to assume that the AUDF registers will have something to do with it. Try the following statements:

```
SOUND 0,10,10,8
SOUND 0,200,10,8
```

Uh, oh, the higher frequency value gave us a lower frequency sound. What gives? It turns out that the AUDF registers are used to determine the frequency rather than used as the frequency. In technical terms, they specify "N" in divide-by-N circuits.

What does that mean in nontechnical terms? The Atari computers have several "clocks" on board. These clocks don't tell the time, but rather send out a stream of pulses with a specific frequency. The clock that is usually used by the AUDF registers has a frequency of 64 MHz (Megahertz, meaning 1,000 pulses per second). A divide-by-N circuit takes this stream as input, and for every N pulses coming in it sends one out, in this case to the speaker.

----0000	(0)	means no volume (no sound)
----0001	(1)	means the lowest volume
•		
•		
•		
----1111	(15)	means the highest volume

Figure 10. AUDC1 bit chart

For example, if N were equal to two, then one pulse would go out for every two coming in, resulting in an output frequency of 32 KHz. If N were equal to 128, the output would have a frequency of .5 MHz, or 500 Hz (hertz, meaning one pulse per second). Now you should be able to see where the divide-by-N name comes from.

There is one important detail that I failed to mention. In the case of Atari sound, before the pulses from the divide-by-N circuits get to the speaker, they automatically go through a divide-by-two circuit, regardless of what the value of N was. This means that our two examples would result in actual sound frequencies of 16 KHz and 250 Hz.

You should also note that POKEY adds one to the value you put in AUDF before it gives it to the divide-by-N circuit. That results in a possible frequency range of 125 Hz (64 KHz/256/2) to 32 KHz (64 KHz/1/2). Most human ears can't hear sounds higher than 20 KHz, so there's more than enough in the high-frequency range. In the low range, you can hear down as far as 20 Hz, so you can see that the low end is lacking. We'll see ways a little later on that allow you to get around that.

(PEEK): POT0 is the value of Paddle 0. Since it has a shadow register PADDL0 back at Location 624, you should go back there for a description.

Machine language programmers should also consult ALLPOT at Location 53768 and POTGO at Location 53771.

AUDC1 (POKE) and POT1 (PEEK) 53761 D201

(POKE): The four AUDC registers are used to specify the DISTORTION and VOLUME parts of the SOUND statement (see AUDF1). Their official name is the "audio control registers." Bits 0 through 3 are used for volume, and Bits 4 through 7 for distortion. Figure 10 shows how the volume bits work.

How does volume tie in to our discussion on sound waves? The higher the volume, the more electricity there is in each pulse

to the speaker. The more electricity there is in each pulse, the greater the distance the speaker moves in and out. The greater the distance the speaker moves, the larger the sound wave. Finally, the larger the wave, the louder the sound. (Note that "large" refers to height, not length.)

Distortion, unfortunately, is not quite as easy to explain as volume. Let's start by explaining the easiest distortion bit, number four. When Bit 4 is set to 1 (which adds 16 to the previous value), you control the speaker directly. In other words, AUDF is ignored, and the value in the volume bits sent directly to the speaker. Try the following:

POKE 53761,24

This sends a volume value of eight to the speaker, and you hear a "pop" as a result. Now try it again. Don't bother turning your volume up; there wasn't a pop this time. Why? As long as Bit 4 is set, the volume will always be sent; the speaker will receive a constant stream of electricity rather than a pulse. This moves it out but not back in again. Turn the volume off to move it back in:

POKE 53761,16

You should hear another pop as it moves back. That means that a pulse is actually two pops, but since they happen so close

together, the result is one loud pop. Try this:

POKE 53761,24;POKE 53761,16

Now you hear the result of a complete pulse. In any case, this is how you can define your own pulses. BASIC is too slow to really do anything with this technique (called "volume only" sound), but with machine language you define your own frequencies, wave shapes and sound envelopes, creating anything from the sound of a piano to the sound of a human voice. If this sounds interesting, please see *De Re Atari* for more details on each of these topics.

We're now left with the other three distortion bits. They involve something called "poly-counters." Without going into explicit detail, a poly-counter takes a stream of pulses and "randomly" removes some of them. This results in a frequency that, although close to the original, is constantly changing by small amounts. This in turn results in a messy sound, often called "noise."

The word "random" is used loosely in the preceding description, because the so-called random pattern will eventually repeat itself. I'm not going to explain why, because the innermost workings of a poly-counter are not important to us here. See *De Re Atari* if you're curious. For now, suffice it to say that there are three different types of poly-counters in the Atari; a 17-bit one, a five-bit one and a four-bit one. The greater the number of bits the poly-counter uses, the longer it will take for the pattern to repeat. Let's jump a little ahead of ourselves by listening to the result of the 17-bit poly-counter:

SOUND 0,100,8,8

--0-----	(0)	means that either the four bit or seventeen bit counter will be applied (depending on the value of bit six).
--1-----	(32)	means that neither will be applied regardless of what bit six is set to.
-0-----	(0)	means that the seventeen bit counter will be applied if bit five is not set.
-1-----	(64)	means that the four bit counter will be applied if bit five is not set.
0-----	(0)	means that the five bit counter will be applied.
1-----	(128)	means that the five bit counter will not be applied.

Figure 11. Poly-counters bit chart

And now the five-bit:

SOUND 0,100,2,0

And finally the four-bit:

SOUND 0,100,12,0

Notice how the four- and five-bit poly-counters create more of a repetitious sound, while the 17-bit sound is much more random (it seems to be just noise).

By this time it should be apparent that the last three bits are used to select which poly-counters are applied to our unsuspecting frequency. Figure 11 shows their exact uses.

You can see that the five-bit counter can be combined with either or neither of the other two. Also note that the divide-by-two circuit that was mentioned under AUDF1 is applied after the poly-counters. In case all of this has you confused, Figure 12 shows all the possible bit combinations and

the corresponding order of things.

Remember to divide these values by 16 to get the DISTORTION value for BASIC SOUND command.

After all of this, do you know what I'm going to tell you now? Don't worry about any of it. Everything I just told you is for the sole purpose of letting you understand what's going on behind the sounds you are creating. As long as it sounds good, don't worry what poly-counters are being used. It just doesn't matter.

(PEEK): POT1 is the value of Paddle 1. It has a shadow register PADDL1 at Location 625.

AUDF2 (POKE) and POT2 (PEEK)
53762 D202

(POKE): AUDF2 specifies the frequency for Audio channel 2.

(PEEK): POT2 is the value of Paddle 2. It has a shadow register PADDL2 at Location 626.

AUDC2 (POKE) AND POT3 (PEEK)
53763 D203

(POKE): AUDC2 specifies distortion and volume for Audio channel 2.

(PEEK): POT3 is the value of Paddle 3. It has a shadow register PADDL3 at Location 627.

AUDF3 (POKE) and POT4 (PEEK)
53764 D204

(POKE): AUDF3 specifies the frequency for Audio channel 3.

(PEEK): POT4 is the value of Paddle 4. It has a shadow register PADDL4 at Location 628.

AUDC3 (POKE) and POT5 (PEEK)
53765 D205

(POKE): AUDC3 specifies distortion and volume for Audio channel 3.

(PEEK): POT5 is the value of Paddle 5. It has a shadow register PADDL5 at Location 629.

AUDF4 (POKE) and POT6 (PEEK)
53766 D206

(POKE): AUDF4 specifies the frequency for Audio channel 4.

(PEEK): POT6 is the value of Paddle 6. It has a shadow register PADDL6 at

000-----	(0)	1. divide-by-N circuit 2. five bit poly-counter 3. seventeen bit poly-counter 4. divide-by-two circuit
0-1-----	(32)	1. divide-by-N circuit 2. five bit poly-counter 3. divide-by-two circuit
010-----	(64)	1. divide-by-N circuit 2. five bit poly-counter 3. four bit poly-counter 4. divide-by-two counter
100-----	(128)	1. divide-by-N circuit 2. seventeen bit poly-counter 3. divide-by-two circuit
1-1-----	(160)	1. divide-by-N circuit 2. divide-by-two circuit
110-----	(192)	1. divide-by-N circuit 2. four bit poly-counter 3. divide-by-two circuit

Figure 12. Bit combinations for AUDC1

-----0		means that the 64 KHz clock is the main source of pulses for all channels (unless otherwise specified).
-----1	(1)	means that the 15 KHz clock is used instead.
-----1-	(2)	inserts a high-pass filter into channel two and clocks it with channel four.
-----1--	(4)	inserts a high-pass filter into channel one and clocks it with channel three.
----1---	(8)	joins channel four to channel three ("N" becomes sixteen bits long).
---1----	(16)	joins channel two to channel one.
--1-----	(32)	means use the 1.79 MHz clock for channel three (MHz stands for Mega-Hertz, which is one million pulses per second).
-1-----	(64)	means use the 1.79 MHz clock for channel one.
1-----	(128)	changes the seventeen bit poly-counter into a nine bit poly-counter.

Figure 13. AUDCTL/ALLPOT bit chart

Location 630.

AUDC4 (POKE) and POT7 (PEEK)
53767 D207

(POKE): AUDC4 specifies distortion and volume for Audio channel 4.

(PEEK): POT7 is the value of Paddle 7. It has a shadow register PADDL3 at Location 631.

AUDCTL (POKE) and ALLPOT (PEEK)
53768 D208

(POKE): AUDCTL is the audio control register. This means that you can use it to make changes to the basic sound setup. So, without any further ado, let's take a look at Figure 13 to see how its bits are used (unless otherwise noted, a bit set to zero simply cancels the effect of it being set to one).

Some of these are probably giving you problems, so let's take a closer look. First of all, we just discussed poly-counters, so there should be no problem there. You should also understand the effect that using different types of clocks will have. The higher the frequency of the clock, the higher frequency of the sound. For example, try the following statement:

SOUND 0,100,10,8

Now use Bit 0 of AUDCTL to change the main clock from 64 KHz to 15 KHz:

POKE 53768,1

Notice how the sound got lower? Now change the clock for Channel 1 to 1.79 MHz:

POKE 53768,64

By using this ability to change the clocks, you can extend the range of frequencies that the Atari can produce. When you use the 1.79 MHz clock, however, all the sound will be up in the high range, since even with AUDF set to 255, the resulting frequency will still be 3.5 KHz.

How do we get down into the lower frequencies? This is where being able to join two channels together comes in handy. By having sixteen bits available for "N" in our divide-by-N circuit rather than only eight, we can fully utilize the 1.79 MHz clock. Now we can go all the way down to 14 Hz.

Try the following program to get the idea. It joins Channel 2 with Channel 1 (Channel 2 is the high byte, Channel 1 the

low), switches Channel 1's clock to 1.79 MHz, and then lets you use Joystick 0 to change the values of Channel 1 (move the joystick up or down) and Channel 2 (move the joystick left and right):

```
100 SOUND 0,0,0,0
110 POKE 53768,80
120 POKE 53761,160
130 POKE 53763,168
140 CH1=0:CH2=0
150 POKE 53760,CH1
160 POKE 53762,CH2
170 S0=STICK(0)
180 CH1=CH1-(S0=14)+(S0=13)
190 CH1=CH1-CH1*(CH1=256)+256*(CH1=-1)
200 CH2=CH2-(S0=7)+(S0=11)
210 CH2=CH2-CH2*(CH2=256)+256*(CH2=-1)
220 GOTO 150
```

Note that Line 120 sets Channel 1's volume to 0, since Channel 2 is where the sound will come from. Line 130 sets Channel 2 to Distortion 10 (pure tone) and Volume 8.

The last thing we need to explain is probably what was confusing you most: high-pass filters. Actually, high-pass filters are a relatively simple concept. All they do is stop a frequency from getting through to the speaker unless it's higher than some other specified frequency. For example, let us suppose that we set Bit 2 of AUDCTL. This, we are told, means that a high-pass filter will be inserted in Channel 1, and Channel 3 will be used to clock it. In English, Channel 1 will only be heard if its frequency is greater than that of Channel 3. Let's look at a picture of this in Figure 14.

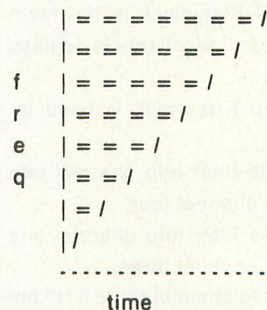


Figure 14. Frequency vs. time

If the diagonal line represents the sound from Channel 3, then the shaded area above it represents the frequencies that Channel 1 can play at any given time during that sound. Unfortunately, this isn't quite the way things actually work.

On the Atari, a high-pass filter apparently works by looking at the pulses from each channel. If two pulses coincide, then

the high-pass filter will only allow one through. I'm not exactly sure whether one is given priority over the other (it would make sense to give the pulse from the higher frequency priority), but the result is that Channel 1 is not completely cut off when it should be, and is partially cut off when it shouldn't be. Try the following to see what I mean:

```
100 SOUND 0,0,0,0
110 POKE 53768,4
120 POKE 53761,168
130 POKE 53765,168
140 POKE 53760,200
150 POKE 53764,100
160 GOTO 160
```

What you're hearing is a bizarre combination of the pulses coming from Channels 1 and 3. According to the definition of a high-pass filter, though, you shouldn't be hearing Channel 1 at all, since its frequency is lower than that of Channel 3. Oh well, at least it allows for some neat sound effects.

(PEEK): ALLPOT is used to determine whether or not the POT value for a particular paddle is valid (see the previous eight locations). If bit n of ALLPOT is set, then POTn contains a valid value for paddle n. This should not concern you unless you're programming in machine language.

STIMER (POKE) and KBCODE (PEEK)
53769 D209

(POKE): POKEing any value in STIMER will get the system timers (POKEY timers) going. See the TIMER section for more information.

(PEEK): When a key is pressed, this location is the first to know about it. From here it goes into the shadow register CH at Location 764, which is where you should look for more details.

SKRES (POKE) and RANDOM (PEEK)
53770 D20A

(POKE): POKEing any value here sets bits five through seven of the serial port status register SKSTAT (53775) to zero.

(PEEK): If you do any machine language programming, you've probably wondered at one time or another how to get random numbers. Wonder no more; RANDOM holds the highest eight bits of the 17-bit poly-counter mentioned under AUDC1. In other words, it will give you a "random" number between zero and 255. The quotes come from the fact that the values in RANDOM will eventually start repeating them-

selves (i.e., you'll get the same series of numbers all over again). For all practical purposes, however, you can consider RANDOM to be random.

POTGO (POKE only)
53771 D20B

POKEing any value here starts the POT scan sequence. The POT scan sequence is simply the routine that figures out what values should be in the POT registers. The Stage 2 VBLANK routine automatically takes care of POTGO; and you should generally let.

Noname
53772 D20C

This location is not used. Don't forget that it's in the middle of a chip, so don't try to use it yourself.

SEROUT (POKE) and SERIN (PEEK)
53773 D20D

(POKE): SEROUT is used when the serial port needs another byte to send. It's called the eight-bit parallel holding register; a long name that simple means that it holds the byte until the serial output shift register needs it. The serial output shift register then sends out the byte one bit at a time.

SEROUT is usually written to in response to a "serial output data needed" interrupt, which is generated when the serial output shift register needs another byte. See IRQEN.

(PEEK): SERIN is also the parallel holding register, but is used when reading rather than writing.

You usually read the parallel holding register when a "serial input data ready" interrupt occurs. This happens when all eight bits of the incoming byte have been received and transferred to the parallel holding register. See IRQEN.

IRQEN (POKE) and IRQST (PEEK)
53774 D20E

(POKE): IRQEN is used to enable or disable IRQ interrupts. See its shadow register POKMSK at Location 16 for a complete description. For more information on interrupts in general, see the section right before Location 512.

(PEEK): IRQST is the IRQ interrupt status register. Its bits correspond to the same interrupts as those in IRQEN and are set to zero when the corresponding interrupt occurs. In order to reset it after an interrupt does occur, you must clear the interrupt bit in IRQEN (you can then reset it if you want the

interrupt to be able to happen again).

There are two IRQ interrupts that are enabled and have status registers elsewhere (in PIA). See PACTL and PBCTL at Locations 54018 and 54019.

SKCTL (POKE) and SKSTAT (PEEK)
53775 D20F

(POKE): At the shadow register for SKCTL (SSKCTL, 562), I shied away from giving a description. I won't do that here, but be forewarned that most of it will be for the expert.

SKCTL controls the configuration of the serial port, determines the type of pot scan to be used and enables the keyboard circuits. Its bits have the meanings in Figure 15 (unless noted otherwise, a bit set to zero has the opposite effect of when it's set to one).


(PEEK): SKSTAT is a status register for the serial port and the keyboard. Instead of trying to explain, let me just give you the bit meanings (bits are normally set and have the following meanings when they aren't).

If any of the last three bits get cleared, they

should be reset to one using SKRES at 53770.

SKSTAT is also helpful if you want to add a voice track to your program. You probably already know that you can play a tape through the TV speaker while you are running a program (see PACTL at Location 54018 if not). Unfortunately, if you want what's playing on the tape to coincide with what's going on on the screen, it's difficult to get the timing right. You can, however, put a digital track alongside the voice/music that will tell the computer when to do things. SKSTAT is used in such cases to look at the digital track. Since this technique has limited applications, I won't go into it here. If you're interested, however, you should take a look at the section on cassette in *De Re Atari*. It has an excellent explanation, along with the programs needed to both read and write the digital track.

Noname
53776-54015 D210-D2FF

These locations are unused at this time, even though they are a part of POKEY. 

-----1	(1)	enables the keyboard debounce circuit.
-----1-	(2)	enables the keyboard debounce circuit.
-----0--		means that POKEY will take 228 scan lines (one frame) to determine the POT values.
-----1--	(4)	means that POKEY will only take two scan lines to determine the POT values, but they won't be as accurate.
-----0---		means that serial output will be transmitted as a logic true/false signal.
-----1---	(8)	means that serial output will be transmitted as a two-tone signal (used for cassette data).
-001----	(16)	
•		
•		
•		
-111----	(112)	these three bits determine how to transmit and receive data (with respect to clock rates). See page 11.27 of the Hardware Manual for a complete description.
1-----	(128)	forces the serial output to zero (forces a break).

Figure 15. SKCTL (POKE) bit chart

-----1	this bit is not used (and is always set to one).
-----0-	means that the serial input shift register is busy.
-----0--	means that the last key pressed is still pressed.
-----0---	means that the shift key is pressed.
---0----	means that you can ignore the shift register and read data straight from the serial input port.
--0-----	means that a keyboard over-run has occurred. To tell you the honest truth, I have no idea what that means.
-0-----	means that a serial data input over-run has occurred. Ditto.
0-----	means that a serial data input frame error has occurred.

Figure 16. SKSTAT (PEEK) bit chart

UN-SPRITES: COPY & PASTE GRAPHICS

Sprites (Player/Missile Graphics) has always been the easiest method of animating screen objects on Atari computers. But what about computers like the Apple IIs and IBM PCs that do not have *Sprites*? They achieve animation by actually moving parts of the graphics screen. Although this is a less efficient method, it does have capabilities that *Sprites* does not; that is, they are not limited to the 8/255 resolution that *Sprites* has, and they can have as many colors as a graphics mode will allow. *Un-sprites* is a machine-language program that will allow you to paste or copy windows to or from any of the 16 Atari graphics modes.

Setting Un-sprites Up

Type in Listing 1 using BASIC Editor II found elsewhere in this issue, and save it to disk or cassette. Executing this program will cause it to POKE the machine-language program directly into memory. For cassette-recorder users, append Listing 1 to the end of your program that will use *Un-sprites* and call a GOSUB to load in the data. Disk users should save the data at address \$6000 to \$6300 using the "save binary" command in DOS.

Two demonstration programs have been included (Listing 2 and 3) to give you an idea

BY JASON LEIGH

of how *Un-sprites* can be used for animation. Listing 4 is the BASIC equivalent of *Un-sprites*; the BASIC program should make it easier to understand the assembly listing.

Using Un-sprites in BASIC

Copying a window

The command in BASIC to copy a window is:

```
COPY=USR(24842, Screen Start, StartX,  
StartY, EndX, EndY, Buffer Address, Graph-  
ics Mode)
```

Screen Start is the address of the start of the screen memory. This can easily be computed by entering the desired graphics mode and reading memory locations \$58 (88) and \$59 (89). Here's an example:

```
10 GRAPHICS 7  
20 SCRNSTART=PEEK(88)+PEEK  
(89)*256
```

You are not limited to just that screen area; you may also specify a totally different memory address and copy a window from that by assigning the appropriate values for *Screen Start*.

StartX, StartY & EndX, EndY

These specify the boundaries of the copy window. Although there is no limit to the size of the window you can copy, it is advisable to keep these coordinates within the limits of the declared graphics mode.

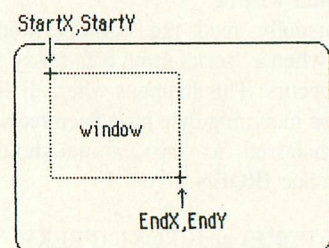


FIGURE 1

Buffer Address

The window you copy must be stored somewhere in memory, and so a storage buffer is needed. The advantage of having a user-definable buffer is that several windows can be copied, saved, pasted and re-pasted as needed by specifying the same buffer address. A buffer space can be declared at any address where nothing else occupies the memory. This can safely be done by dimensioning a relatively large string and using that as a buffer pool for many windows or just

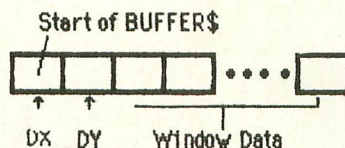


FIGURE 3

one window. The alternative is to dimension many little strings and store only one window in each of them.

The short program segment below dimensions a string 8K in size and assigns zeros to it. BUFFADDRS holds the starting address of the string buffer.

```
10 DIM BUFFER$(8192)
20 BUFFADDRS=ADR(BUFFER$)
30 BUFFER$(1)=" ":BUFFER
  $(8192)=" ":BUFFER$(2)=B
  UFFER$(1)
```

Graphics Mode

These correspond exactly to those of the XL and XE Ataris. Graphics modes 0 to 11 are the same for all 8-bit machines, but graphics modes 12 to 15 are defined as follows:

Graphics Mode 12: ANTIC Mode 4 (four-color text mode 40/24)

Graphics Mode 13: ANTIC Mode 5 (four-color text mode 40/12)

Graphics Mode 14: ANTIC Mode 12 (two-color graphics mode 160/192)

Graphics Mode 15: ANTIC Mode 14 (four-color graphics mode 160/192)

Pasting a Window

The command in BASIC to paste a window is:

PASTE=USR(25077, Screen Start, StartX, StartY, Buffer Address)

Screen Start is the address of the screen memory in which you wish to paste the window.

StartX, StartY specify the position to place the top-left corner of the window.

Buffer Address is the address of the buffer that holds data from a previously copied window.

It is assumed that you will paste a window in the same graphics mode it was cut from. You may, however, choose to paste it in a

different graphics mode; the effects can be quite interesting (or disastrous).

Warning: If you accidentally omit a parameter or put them in incorrect order, you could experience a serious lockup that will result in the loss of your program.

That's basically all you need to know to use *Un-sprites*; for those of you interested in how *Un-sprites* works, read on.

Explanation

Listing 4 is the BASIC equivalent of *Un-sprites*. It would probably be a good idea to

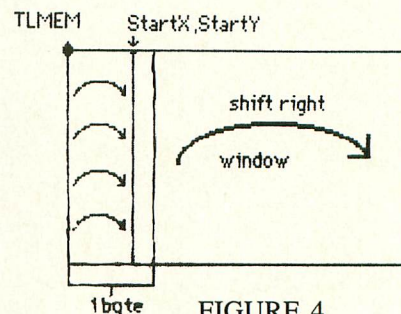


FIGURE 4

The number of bytes used per window can therefore be calculated as: $(DX * DY)$. After DX and DY have been calculated, the top-left memory (TLMEM) location of the win-

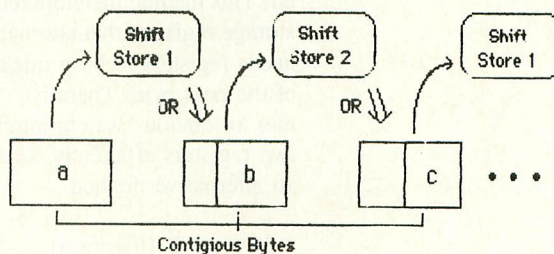


FIGURE 4

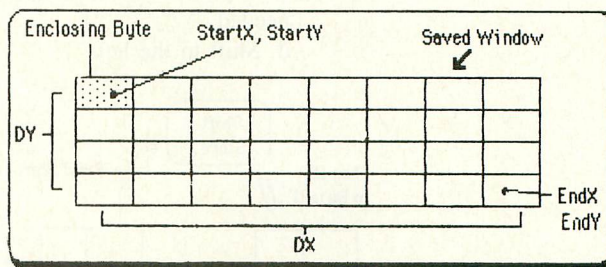


FIGURE 2

flip between the BASIC and assembly listings while reading the following explanation.

Copying

When start and end coordinates are specified, *Un-sprites* will find the closest byte that encloses those coordinates. (See Figure 2.)

DX holds the width of the window and DY, the height. Before the window is saved, these are used as headers in the buffer, as shown in Figure 3.

down is computed. When all this is done, DX and DY will be saved to the buffer followed by the data in the window.

Pasting

Whereas Copy saved the entire byte that enclosed StartX and StartY, Paste allows exact pixel alignment. *Un-sprites* will automatically shift the required number of bits to place the window at the exact StartX and StartY position.

There are two methods of performing these shifts:

Method 1.

- Calculate TLMEM.
- Find from shift table, the number of shifts needed.
- Shift to the right.

Byte *a* is shifted the required number of bits to the right. Shift Store 1 acts as a bit bucket that collects all the bits that have shifted out of *a*. The data in the shift store is then ORed with shifted byte *b*. But in order to shift *b*, a separate shift store (Shift Store 2) is needed. This method therefore requires two shift-storage registers that alternate in use, plus it needs registers to keep track of the address of the next byte. There isn't (at least not to me) an obvious way to alternate the use of two registers efficiently, so I came up with an alternative method.

Method 2. (Figure 5)

- Calculate TLMEM.
- Add 1 to TLMEM (TLMEM+1).
- Find from shift table the number of shifts needed.
- Shift to the left.

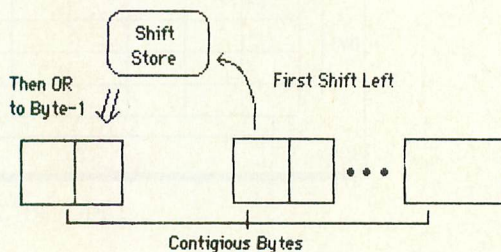


FIGURE 5

Instead of shifting to the right, we move to TLMEM+1 and shift the required number of bits to the left. This is much easier, as only one shift-storage register is needed; there is no intriguing alternation between two shift registers as in Method 1.

Calculating the Shift

The code needed to do this is:

Shift_pointer=STX-(INT(STX/Pixels__

Per__Byte)*Pixels__Per__Byte)

With the shift pointer computed, the number of shifts is then found from SHFTABLE.

An example of calculating the shift is shown below:

Let Pixels__Per__Byte (PPB)=4 and STX=9;

$STX/PPB = 00001001_2 / 100_2 = 10_2 = 2_{10}$;
then $2*4=8$ and $(STX-8)=(9-8)=1$.

Look up 1 in SHFTABLE4 and we get 6. That means we need to do six left-shifts from byte to byte 1. (Figure 6)

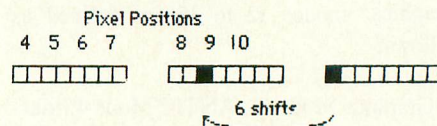


FIGURE 6

Note: Two PPB graphics modes (GTIA Modes 9,10,11) need four shifts per pixel, four PPB modes (Graphics Modes 3,5,7,15) need two shifts per pixel, eight PPB modes (Graphics Modes 4,6,8,14) need one shift and one PPB modes (the text modes) need no shifts.

That should be enough information for you to tailor the *Un-sprites* assembly listing to your own needs. If \$6000 is not a convenient location for this program, you could always move the start of the program to another location. The .OPT LIST command has been included so that it will always display the new location of COPY and PASTE routines.

Note that for larger windows the program could get rather slow, but if you are page flipping, the window can be made to appear instantaneously by pasting on one screen while displaying another.

There are many applications for windowing: animating frames, editing graphics screens and manipulating GEM-like windows are just a few. *Sprites* and *Un-sprites* should be the complete pair that will allow you to perform the graphics manipulation you've always wanted on the Atari. Enjoy.

LISTING 1: BASIC

```

QM 1 REM Program Listing 1:
YN 2 REM Un-sprite Loader
HI 3 REM By Jason Leigh
MJ 4 REM
QO 10 GOSUB 32100: ? "UN-SPRITES LOADED":E
ND
IN 32000 DATA 201,10,208,3,76,43,96,201,2
0,208
AY 32001 DATA 3,76,91,96,201,2,208,3,76,9
4
HL 32002 DATA 96,201,40,208,3,76,103,96,2
01,4
ZJ 32003 DATA 208,3,76,109,96,201,8,208,3
,76
LR 32004 DATA 115,96,96,160,0,173,1,6,133
,208
MM 32005 DATA 173,2,6,133,209,24,46,1,6,4
6
QL 32006 DATA 2,6,200,192,3,208,244,24,38
,208
DT 32007 DATA 38,209,24,165,208,109,1,6,1
41,1
FH 32008 DATA 6,165,209,109,2,6,141,2,6,2
4
AF 32009 DATA 96,32,43,96,24,46,1,6,46,2
TX 32010 DATA 6,24,96,32,91,96,76,94,96,3
2
IC 32011 DATA 94,96,76,94,96,32,109,96,76
,94
KL 32012 DATA 96,201,2,208,3,76,143,96,20
1,4
XO 32013 DATA 208,3,76,152,96,201,8,208,3
,76
YQ 32014 DATA 158,96,96,24,110,2,6,110,1,
6
LK 32015 DATA 24,96,32,143,96,76,143,96,3
2,152
GK 32016 DATA 96,76,143,96,174,11,6,189,2
08,98
OJ 32017 DATA 141,16,6,189,224,98,141,17,
6,96
FE 32018 DATA 173,7,6,141,1,6,169,0,141,2
MY 32019 DATA 6,173,17,6,32,0,96,173,1,6
BS 32020 DATA 141,14,6,173,2,6,141,15,6,2
4
HJ 32021 DATA 173,3,6,109,14,6,141,14,6,1
73
OM 32022 DATA 4,6,109,15,6,141,15,6,173,5
UB 32023 DATA 6,141,1,6,173,6,6,141,2,6
IW 32024 DATA 173,16,6,32,121,96,24,173,1
,6
GU 32025 DATA 109,14,6,141,14,6,173,2,6,1
09
AL 32026 DATA 15,6,141,15,6,96,104,104,14
1,4
TO 32027 DATA 6,104,141,3,6,104,141,6,6,1
04
UT 32028 DATA 141,5,6,104,104,141,7,6,104
,141

```

```

PI 32029 DATA 9,6,104,141,8,6,104,104,141
,10
LL 32030 DATA 6,104,133,207,104,133,206,1
04,104,141
NF 32031 DATA 11,6,169,0,141,20,6,32,164,
96
CZ 32032 DATA 32,180,96,56,173,8,6,237,5,
6
SG 32033 DATA 141,1,6,173,9,6,237,6,6,141
LO 32034 DATA 2,6,173,16,6,32,121,96,173,
1
WZ 32035 DATA 6,24,105,1,141,12,6,56,173,
10
EG 32036 DATA 6,237,7,6,24,105,1,141,13,6
CB 32037 DATA 162,0,173,12,6,129,206,230,
206,165
BA 32038 DATA 206,201,0,208,2,230,207,173
,13,6
SK 32039 DATA 129,206,230,206,165,206,201
,0,208,2
LA 32040 DATA 230,207,169,0,141,19,6,141,
18,6
IJ 32041 DATA 162,0,24,173,14,6,109,18,6,
133
KO 32042 DATA 208,173,15,6,105,0,133,209,
173,20
XI 32043 DATA 6,201,1,240,7,161,208,129,2
06,76
HG 32044 DATA 189,97,32,145,98,230,206,16
5,206,201
UF 32045 DATA 0,208,2,230,207,238,18,6,17
3,18
FP 32046 DATA 6,205,12,6,144,202,169,0,14
1,18
HF 32047 DATA 6,24,173,14,6,109,17,6,141,
14
GX 32048 DATA 6,173,15,6,105,0,141,15,6,2
38
XN 32049 DATA 19,6,173,19,6,205,13,6,144,
168
SN 32050 DATA 96,104,104,141,4,6,104,141,
3,6
RE 32051 DATA 104,141,6,6,104,141,5,6,104
,104
NX 32052 DATA 141,7,6,104,133,207,104,133
,206,169
CG 32053 DATA 1,141,20,6,32,180,96,162,0,
161
EI 32054 DATA 206,141,12,6,230,206,165,20
6,201,0
KE 32055 DATA 208,2,230,207,161,206,141,1
3,6,230
ZN 32056 DATA 206,165,206,201,0,208,2,230
,207,173
BU 32057 DATA 16,6,201,1,208,3,76,142,98,
238
DL 32058 DATA 14,6,173,14,6,201,0,208,3,2
38
UJ 32059 DATA 15,6,173,5,6,141,1,6,173,6
NP 32060 DATA 6,141,2,6,173,16,6,32,121,9
6
VD 32061 DATA 24,173,16,6,32,0,96,56,173,
5
JL 32062 DATA 6,237,1,6,170,173,16,6,201,
8
UD 32063 DATA 208,6,189,240,98,76,139,98,
201,4
SS 32064 DATA 208,6,189,249,98,76,139,98,
189,253
HF 32065 DATA 98,141,21,6,76,146,97,173,1
6,6
DD 32066 DATA 201,1,208,5,161,206,129,208
,96,169
HF 32067 DATA 0,141,22,6,165,208,133,204,
165,209

```



```

IS 32068 DATA 133,205,198,204,165,204,201
,255,208,2
MC 32069 DATA 198,205,160,0,161,206,204,2
1,6,240
FD 32070 DATA 9,24,42,46,22,6,200,76,184,
98
JX 32071 DATA 129,208,161,204,13,22,6,129
,204,96
CN 32072 DATA 1,1,1,4,8,4,8,4,8,2
TT 32073 DATA 2,2,1,1,8,4,40,20,20,10
AL 32074 DATA 10,20,20,40,40,40,40,40,40,
40
LE 32075 DATA 20,40,8,7,6,5,4,3,2,1
JR 32076 DATA 0,8,6,4,2,8,4
AW 32077 REM * 767 BYTES
XG 32100 FOR T=0 TO 766:READ A:POKE 24576
+T,A:NEXT T:RETURN

```

LISTING 2: BASIC

```

RJ 1 REM Program Listing 2:
MJ 2 REM Un-sprite demonstration 1
HI 3 REM By Jason Leigh
TO 4 REM (Copy, paste & scroll)
NK 5 REM
EG 20 GRMODE=7:WDCPY=24842:WDPSTE=25077:D
IM BUF$(8192):BUF$(1)="":BUF$(8192)="
":BUF$(2)=BUF$(1):BUFFER=ADR(BUF$)
BA 30 REM
FH 40 REM PART 1
BC 50 REM
AO 60 GRAPHICS GRMODE:COLOR 1:MEM=PEEK(88
)+PEEK(89)*256
ZR 70 ? " Draw random shape"
LR 80 FOR T=1 TO GRMODE+2:X=RND(1)*GRMODE
*3+6:COLOR RND(1)*15
MF 90 FOR R=T TO T+2+(GRMODE*3):PLOT X,R:
DRAWTO X+15,R:NEXT R:NEXT T
AL 100 COPY=USR(WDCPY, MEM, 0, 0, 20, 20, BUFFE
R, GRMODE)
RL 110 ? " Paste shape"
MW 120 FOR T=1 TO 40:X=RND(1)*160:Y=RND(1
)*50:PASTE=USR(WDPSTE, MEM, X, Y, BUFFER):
NEXT T
QS 130 REM
PK 140 REM PART 2
QW 150 REM
MU 160 GRAPHICS GRMODE:X=0:Y=15
BS 170 ? " Horizontal scroll"
MC 180 FOR T=1 TO 128:X=X+1:PASTE=USR(WDP
STE, MEM, X, Y, BUFFER):NEXT T
RE 190 REM
PR 200 REM PART 3
QP 210 REM
MC 220 GRAPHICS GRMODE:X=40:Y=0
HW 230 ? " Vertical scroll"
IB 240 FOR T=1 TO 40:Y=Y+1:PASTE=USR(WDP
STE, MEM, X, Y, BUFFER):NEXT T
QX 250 REM
QR 260 REM PART 4
RB 270 REM
GW 280 GRAPHICS GRMODE:X=0:Y=0
SC 290 ? " Diagonal scroll"
JN 300 FOR T=1 TO 50:X=X+1:Y=Y+1:PASTE=US
R(WDPSTE, MEM, X, Y, BUFFER):NEXT T
SK 310 GRAPHICS 0

```

LISTING 3: BASIC

```

SG 1 REM Program Listing 3:
SQ 2 REM Un-sprite Demonstration 2
HI 3 REM By Jason Leigh
YG 4 REM (2.5D rotating crystal matrix)
NK 5 REM
EQ 10 GRMODE=8:WDCPY=24842:WDPSTE=25077:D
IM BUF$(8192):BUF$(1)="":BUF$(8192)="
":BUF$(2)=BUF$(1):BUFFER=ADR(BUF$)
DE 20 TRAP 20:GRAPHICS 0:?"Number of poi
nts in crystal (3-10)":INPUT PTS
BH 30 IF PTS<3 OR PTS>10 THEN 20

```

```

BG 40 GRAPHICS GRMODE:COLOR 1:MEM=PEEK(88
)+PEEK(89)*256:POKE 752,1:POKE 82,0:?"
:?"
BC 50 REM
FS 60 REM GENERATE & COPY FRAMES
BE 70 REM
UF 80 DIM A(PTS):DEG :BX=1:BY=1
DR 90 ANG=0:FOR T=1 TO PTS:A(T)=ANG:ANG=A
NG+(360/PTS):NEXT T
DH 100 FOR R=0 TO (360/PTS)-5 STEP 5:COUN
T=0
QX 110 ? "GENERATE & COPY EACH ROTATION O
F CRYSTAL"
ZK 120 FOR T=1 TO PTS:COUNT=COUNT+1
JU 130 X=SIN(A(T))*20+160:Y=COS(A(T))*6+8
0:IF T=1 THEN X1=X:Y1=Y
TW 140 PLOT X,Y:IF COUNT<>1 THEN DRAWTO X
0,Y0
CX 150 PLOT 160,63:DRAWTO X,Y:PLOT 160,95
:DRAWTO X,Y
HA 160 X0=X:Y0=Y:A(T)=A(T)+5:NEXT T
AH 170 PLOT X,Y:DRAWTO X1,Y1
PB 180 COPY=USR(WDCPY, MEM, 131, 61, 189, 97, B
UFFER, GRMODE):BUFFER=BUFFER+300:?" #6;"
K":NEXT R
QN 200 REM
WD 210 REM ANIMATE FRAMES
QR 220 REM
KY 230 GRAPHICS 0:POKE 752,1:?" ? " PRE
55 A KEY TO BEGIN ANIMATION"
DA 240 IF PEEK(764)=255 THEN 240
GZ 250 POKE 764,255:X=100:Y=60:GRAPHICS 8
+16:SETCOLOR 2,0,0:SETCOLOR 1,0,15:SET
COLOR 4,7,5
BI 260 FOR T=ADR(BUF$) TO BUFFER-300 STEP
300
VE 270 PASTE=USR(WDPSTE, MEM, X, Y, T)
SZ 280 X=X+BX:Y=Y+BY
BX 290 IF Y<1 OR Y>150 THEN BY=-BY
GX 300 IF X>267 OR X<1 THEN BX=-BX
KN 310 NEXT T:GOTO 260

```

LISTING 4: BASIC

```

TD 1 REM Program Listing 4:
IR 2 REM BASIC Un-sprite equivalent
HI 3 REM By Jason Leigh
NJ 4 REM
DJ 10 DIM BUF$(8192)
RB 20 GRAPHICS 0:?"GRAPHICS MODE (0-8)":
:INPUT GRMODE
XV 30 FOR T=0 TO GRMODE:READ PPB:NEXT T
KO 40 RESTORE 10030:FOR T=0 TO GRMODE:REA
D BPL:NEXT T
YB 50 GRAPHICS GRMODE:COLOR 1
BD 60 REM
AP 70 REM Generate random shape
BF 80 REM
LW 90 FOR T=1 TO GRMODE+2:X=RND(1)*GRMODE
*3:COLOR RND(1)*4
TN 100 FOR R=T TO T+2+(GRMODE*3):PLOT X,R
:DRAWTO X+15,R:NEXT R:NEXT T
QN 200 REM
JC 210 REM Copy window
QR 220 REM
LV 230 ? "START VECTOR->":INPUT STX,STY
MI 240 ? "END VECTOR->":INPUT EX,EY
HU 250 SCRN=PEEK(88)+PEEK(89)*256
TD 260 TLMEM=SCRN+STY*BPL+INT(STX/PPB):DX
=INT((EX-STX)/PPB)
UK 270 DY=EY-STY
OE 280 MEM=TLMEM
FP 290 FOR DOWN=0 TO DY
SS 300 FOR ACROSS=0 TO DX:COUNT=COUNT+1:B
UF$(COUNT)=CHR$(PEEK(MEM+ACROSS)):POKE
(MEM+ACROSS),255:NEXT ACROSS
ZY 310 MEM=MEM+BPL:NEXT DOWN
QR 500 REM
OC 510 REM Paste Window
QU 520 REM
UZ 530 ? "PUT BLOCK AT->":INPUT STX,STY
GV 540 TLMEM=SCRN+STY*BPL+INT(STX/PPB):ME
M=TLMEM

```



```

FK 550 FOR DOWN=0 TO BY
TA 560 FOR ACROSS=0 TO DX:ADD=ADD+1:POKE
(MEM+ACROSS),ASC(BUF$(ADD,ADD)):NEXT A
CROSS
AM 570 MEM=MEM+BPL:NEXT DOWN
RG 580 REM
RI 590 REM
LK 10000 REM Pixels Per Byte
YF 10010 DATA 1,1,1,4,8,4,8,4,8
IV 10020 REM Bytes Per Line
IM 10030 DATA 40,20,20,10,10,20,20,40,40

```

LISTING 5: ASSEMBLY

```

10 ; *****
20 ; * UN - SPRITES *
30 ; *****
40 ; Assembler used : MAC65
50 ;
60 ; .OPT NO LIST
70 ; Program to transfer blocks of
80 ; screen data in a window.
90 ;
0100 ; BY JASON LEIGH 8/28/86
0110 ; ANALOG COMPUTING 1986
0120 ;
0130 ; To use routine in BASIC
0140 ;
0150 ; COPY=USR(WDWCOPY,Screen Start,Sta
rtX,StartY,EndX,EndY,Buffer Address,Gr
aphics Mode)
0160 ; ('Graphics Mode' corresponds to
those on the XL & XES)
0170 ;
0180 ; PASTE=USR(WDWPSTE,Screen Start,S
tartX,StartY,Buffer Address)
0190 ;
0200 ;
0210 ; *** EQUATES ***
0220 ;
0230 ; *= $0600
0240 LOVAL = *+1 ; Lo Byte Param.
0250 HIVAL = *+2 ; For X and /.
0260 SCRML0 = *+3 ; Screen RAM.
0270 SCRMLH = *+4 ; Start.
0280 STXL0 = *+5 ; StartX.
0290 STXHL = *+6
0300 STY = *+7 ; StartY.
0310 EXL0 = *+8 ; EndX.
0320 EXH = *+9
0330 EY = *+10 ; EndY.
0340 GRMODE = *+11 ; Graphics Mode.
0350 DX = *+12
0360 DY = *+13
0370 TLMEMLO = *+14 ; Top Left corner
0380 TLMEMHI = *+15 ; RAM location.
0390 PPB = *+16 ; Pixels Per Byte
0400 BPL = *+17 ; Bytes Per Line
0410 ACROSS = *+18
0420 DOWN = *+19
0430 DIRECT = *+20 ; 1=PASTE & 0=COPY
WMDW
0440 SHIFTS = *+21 ; Shift align
0450 TEMPSHFT = *+22
0460 TEMPL0 = $D0 ; Temporary.
0470 TEMPHI = $D1
0480 BUFLO = $CE ; Buffer Address.
0490 BUHH = $CF
0500 TMPLOCOPY = $CC ; TEMPL0-1
0510 TMPHICPY = $CD
0520 ;
0530 ; *** SUBROUTINES ***
0540 ;
0550 ; MULTIPLICATION CONTROLLER
0560 ;
0570 ; *= $6000
0580 MULT CMP #10 ; Do multiplicatio
n
0590 BNE M1 ; of LOVAL,HIVAL t
o
0600 JMP X10 ; accumulator.
0610 M1 CMP #20
0620 BNE M2
0630 JMP X20

```

```

0640 M2 CMP #2
0650 BNE M3
0660 JMP X21
0670 M3 CMP #40
0680 BNE M4
0690 JMP X40
0700 M4 CMP #4
0710 BNE M5
0720 JMP X4
0730 M5 CMP #8
0740 BNE M6
0750 JMP X8
0760 M6 RTS
0770 ;
0780 ; MULTIPLY BY 10 ALGORITHM
0790 ;
0800 X10 LDY #0 ; SHIFT LEFT 3
0810 LDA LOVAL ; TIMES TO X8,
0820 STA TEMPL0 ; THEN ADD X2.
0830 LDA HIVAL
0840 STA TEMPHI
0850 X11 CLC ; 3 left shifts.
0860 ROL LOVAL
0870 ROL HIVAL
0880 INY
0890 CPY #3
0900 BNE X11
0910 CLC
0920 ROL TEMPL0 ; Add to 1 left sh
ift.
0930 ROL TEMPHI
0940 CLC
0950 LDA TEMPL0
0960 ADC LOVAL
0970 STA LOVAL
0980 LDA TEMPHI
0990 ADC HIVAL
1000 STA HIVAL
1010 CLC
1020 RTS
1030 ;
1040 ; MULTIPLY BY 20
1050 ;
1060 X20 JSR X10 ; Multiply by 10.
1070 X21 CLC ; X2 algorithm.
1080 ROL LOVAL
1090 ROL HIVAL
1100 CLC
1110 RTS
1120 ;
1130 ; MULTIPLY BY 40
1140 ;
1150 X40 JSR X20 ; Multiply by 20.
1160 JMP X21 ; Double it.
1170 ;
1180 ; MULTIPLY BY 4
1190 ;
1200 X4 JSR X21
1210 JMP X21
1220 ;
1230 ; MULTIPLY BY 8
1240 ;
1250 X8 JSR X4
1260 JMP X21
1270 ;
1280 ; DIVISION CONTROLLER
1290 ;
1300 DIV5 CMP #2 ; Divide LOVAL/HIV
AL by
1310 BNE DV1 ; accumulator.
1320 JMP DIV2
1330 DV1 CMP #4
1340 BNE DV2
1350 JMP DIV4
1360 DV2 CMP #8
1370 BNE DV3
1380 JMP DIV8
1390 DV3 RTS
1400 ;
1410 ; DIVISION BY 2 ALGORITHM
1420 ;
1430 DIV2 CLC
1440 ROR HIVAL
1450 ROR LOVAL
1460 CLC
1470 RTS
1480 ;

```



```

1490 ;DIVISION BY 4 ALGORITHM
1500 ;
1510 DIV4 JSR DIV2
1520 JMP DIV2
1530 ;
1540 ;DIVISION BY 8 ALGORITHM
1550 ;
1560 DIV8 JSR DIV4
1570 JMP DIV2
1580 ;
1590 ;Get PPB and BPL from table.
1600 ;
1610 COPYDATA LDX GRMODE
1620 LDA PIXPBYTE,X
1630 STA PPB
1640 LDA BYTEPLN,X
1650 STA BPL
1660 RTS
1670 ;
1680 ;Calculate Top Left Corner Address
1690 ;
1700 ;Equivalent BASIC statement:-
1710 ;TLMEM=SCRN+STY*BPL+INT(STX/PPB)
1720 ;
1730 TLCLC LDA STY ;STY*BPL
1740 STA LOVAL
1750 LDA #0
1760 STA HIVAL
1770 LDA BPL
1780 JSR MULT
1790 LDA LOVAL
1800 STA TLMEMLO
1810 LDA HIVAL
1820 STA TLMEMHI
1830 CLC
1840 LDA SCRNL0 ;Add SCRNL
1850 ADC TLMEMLO
1860 STA TLMEMLO
1870 LDA SCRNH1
1880 ADC TLMEMHI
1890 STA TLMEMHI
1900 LDA STXL0 ;STX/PPB
1910 STA LOVAL
1920 LDA STXH1
1930 STA HIVAL
1940 LDA PPB
1950 JSR DIV5
1960 CLC
1970 LDA LOVAL ;Add to TLMEM
1980 ADC TLMEMLO
1990 STA TLMEMLO
2000 LDA HIVAL
2010 ADC TLMEMHI
2020 STA TLMEMHI
2030 RTS
2040 ;
2050 ;**** END OF GENERAL SUBROUTINES ****
2060 ;
2070 ;
2080 ;**** MAIN COPY BLOCK ROUTINE ****
2090 ;
2100 ;Get parameters from USR command
2110 ;
2120 .OPT LIST
2130 MDMCPY PLA ;Remove TOS
2140 .OPT NO LIST
2150 PLA ; Screen Start Address
2160 STA SCRNH1
2170 PLA
2180 STA SCRNL0
2190 PLA ; StartX of Top Left Corner
2200 STA STXH1
2210 PLA
2220 STA STXL0
2230 PLA
2240 PLA ; StartY
2250 STA STY
2260 PLA ; EndX of Bottom Right Crnr
2270 STA EXH1
2280 PLA
2290 STA EXL0
2300 PLA ; EndY

2310 PLA
2320 STA EY
2330 PLA ; Storage buffer address
2340 STA BUFH1
2350 PLA
2360 STA BUFLO
2370 PLA ; Graphics mode
2380 PLA
2390 STA GRMODE
2400 LDA #0
2410 STA DIRECT ;0=COPY 1=PASTE
2420 ;
2430 ; Get PPB and BPL for GRMODE
2440 ;
2450 JSR COPYDATA
2460 ;
2470 ; Get TLMEM (Top Left Memory)
2480 ;
2490 JSR TLCLC
2500 ;
2510 ; Calculate DX & DY
2520 ; DX=INT((EX-STX)/PPB)
2530 ;
2540 SEC
2550 LDA EXL0 ;EX-STX
2560 SBC STXL0
2570 STA LOVAL
2580 LDA EXH1
2590 SBC STXH1
2600 STA HIVAL
2610 LDA PPB ;Divide by PPB
2620 JSR DIV5
2630 LDA LOVAL
2640 CLC
2650 ADC #1
2660 STA DX
2670 ;
2680 ;DY=EY-STY
2690 ;
2700 SEC
2710 LDA EY
2720 SBC STY
2730 CLC
2740 ADC #1
2750 STA DY
2760 ;
2770 ;Main Loop To get block
2780 ;
2790 ;Store DX & DY first
2800 ;as headers of the buffer.
2810 ;
2820 LDX #0
2830 LDA DX
2840 STA (BUFLO,X)
2850 INC BUFLO
2860 LDA BUFLO
2870 CMP #0
2880 BNE G1
2890 INC BUFH1
2900 G1 LDA DY
2910 STA (BUFLO,X)
2920 INC BUFLO
2930 LDA BUFLO
2940 CMP #0
2950 BNE LOOP
2960 INC BUFH1
2970 ;
2980 ;Now do loops
2990 ;See equivalent BASIC code.
3000 ;
3010 LOOP LDA #0 ;FOR DOWN=0 To DY
3020 STA DOWN ;FOR ACROSS=0 To DX
3030 STA ACROSS
3040 LDX #0
3050 REP CLC
3060 LDA TLMEMLO
3070 ADC ACROSS
3080 STA TEMPLO
3090 LDA TLMEMHI
3100 ADC #0
3110 STA TEMPHI
3120 LDA DIRECT
3130 CMP #1 ; 1 Means PASTE,
3140 BEQ PASTE

```



```

3150 LDA (TEMPLO,X) ;Get Byte
3160 STA (BUFLO,X)
3170 JMP COPY
3180 PASTE JSR PASTEBYTE
3190 COPY INC BUFLO
3200 LDA BUFLO
3210 CMP #0
3220 BNE F1
3230 INC BUFHI
3240 F1 INC ACROSS5
3250 LDA ACROSS5
3260 CMP DX
3270 BCC REP
3280 LDA #0
3290 STA ACROSS5
3300 CLC
3310 LDA TLMEMLO
3320 ADC BPL
3330 STA TLMEMLO
3340 LDA TLMEMHI
3350 ADC #0
3360 STA TLMEMHI
3370 INC DOWN
3380 LDA DOWN
3390 CMP DY
3400 BCC REP
3410 RTS ;Done, RETURN TO
BASIC.
3420 ;
3430 ;
3440 ; *** MAIN PASTE BLOCK ROUTINE ***
*
3450 ;
3460 .OPT LIST
3470 MDWPSTE PLA ;Get parameters.
3480 .OPT NO LIST
3490 PLA ;Get Start of
3500 STA SCRNIHI ;screen RAM.
3510 PLA
3520 STA SCRNL0
3530 PLA
3540 STA STXHI ;StartX
3550 PLA
3560 STA STXLO
3570 PLA
3580 PLA
3590 STA STY ;StartY
3600 PLA
3610 STA BUFHI ;Buffer Address
3620 PLA
3630 STA BUFLO
3640 LDA #1
3650 STA DIRECT ;1=PASTE
3660 ;
3670 ; Calculate TLMEM
3680 ;
3690 JSR TLALC
3700 LDY #0 ;Get DX, DY
3710 LDA (BUFLO,X)
3720 STA DX
3730 INC BUFLO
3740 LDA BUFLO
3750 CMP #0
3760 BNE H1
3770 INC BUFHI
3780 H1 LDA (BUFLO,X)
3790 STA DY
3800 INC BUFLO
3810 LDA BUFLO
3820 CMP #0
3830 BNE H2
3840 INC BUFHI
3850 H2 LDA PPB ;IF 1 pixel/Byte
3860 CMP #1 ;then no shift.
3870 BNE CALCSHIFT ;else calculate
shifts.
3880 JMP ESC
3890 CALCSHIFT INC TLMEMLO
3900 LDA TLMEMLO ;Increment TLMEM
3910 CMP #0
3920 BNE CONT1
3930 INC TLMEMHI
3940 ;
3950 ;Calculate Shift pntnr
3960 ; = STX - ((STX/PPB)*PPB)
3970 ;
3980 CONT1 LDA STXLO ;(STX/PPB)
3990 STA LOVAL
4000 LDA STXHI
4010 STA HIVAL
4020 LDA PPB
4030 JSR DIV5
4040 CLC
4050 LDA PPB ;(STX/PPB)*PPB
4060 JSR MULT
4070 SEC ; STX-((STX/PPB)*
PPB)
4080 LDA STXLO
4090 SBC LOVAL ; Only 10 byte
4100 TAX ; relevant.
4110 LDA PPB ;Get # of shifts
4120 CMP #8 ;from shift table
S.
4130 BNE CCCN1
4140 LDA SHFTABLE8,X
4150 JMP CCCE
4160 CCCN1 CMP #4
4170 BNE CCCN3
4180 LDA SHFTABLE4,X
4190 JMP CCCE
4200 CCCN3 LDA SHFTABLE2,X
4210 CCCE STA SHIFTS ;Store shifts.
4220 ESC JMP LOOP ; Put to screen.
4230 ;
4240 ;Subroutine to put bytes in
4250 ;correct position with correct
4260 ;number of shifts.
4270 ;
4280 PASTEBYTE LDA PPB
4290 CMP #1 ; If not 1 Pixel
4300 BNE PROCEED ; per byte then
4310 LDA (BUFLO,X) ;shift, else
4320 STA (TEMPLO,X) ;just put.
4330 RTS
4340 PROCEED LDA #0
4350 STA TEMP5HFT ;Clear Temp
4360 LDA TEMPLO
4370 STA TMPLOCY
4380 LDA TEMPHI
4390 STA TMPHICPY ;Copy TEMPLO/HI
&
4400 DEC TMPLOCY ;decrement by 1.
4410 LDA TMPLOCY
4420 CMP #5FF
4430 BNE KONT
4440 DEC TMPHICPY
4450 KONT LDY #0 ;Do required
4460 LDA (BUFLO,X) ;# of shifts.
4470 KONT1 CPY SHIFTS
4480 BEQ EXT
4490 CLC
4500 ROL A
4510 ROL TEMP5HFT
4520 INY
4530 JMP KONT1
4540 EXT STA (TEMPLO,X) ;Put on screen
4550 LDA (TMPLOCY,X)
4560 ORA TEMP5HFT ;OR byte-1
4570 STA (TMPLOCY,X)
4580 RTS
4590 ;
4600 ; Pixels per Byte
4610 ;
4620 PIXPBYTE .BYTE 1,1,1,4,8,4,8,4,8,
2,2,2,1,1,8,4
4630 ;
4640 ;Bytes per Line
4650 ;
4660 BYTEPLN .BYTE 40,20,20,10,10,20,2
0,40,40,40,40,40,40,20,40
4670 ;
4680 ;Table of shifts for 8 PPB modes
4690 ;
4700 SHFTABLE8 .BYTE 8,7,6,5,4,3,2,1,0
4710 ;
4720 ;Table of shifts for 4 PPB modes
4730 ;
4740 SHFTABLE4 .BYTE 8,6,4,2
4750 ;
4760 ;Table of shifts for 2 PPB modes
4770 ;
4780 SHFTABLE2 .BYTE 8,4
4790 ;
4800 ;767 Bytes.

```


An Atari-based Interactive Laser Videodisc System

by Bruce Frumker

Interactive laser videodiscs have not yet reached their full potential use, especially in education. The primary reason is expense—an interactive laser videodisc system can easily cost \$10,000 or more. What I am going to describe in this article is a fully functioning, interactive system for about one-fourth of that cost. And the key to this dramatic price difference is the use of an Atari 8-bit computer as the laser disc controller.

A system like this (with an Atari 130XE) is installed in an interactive exhibit at the Cleveland Museum of Natural History, in its newly opened Sears Hall of Human Ecolo-

gy. This interactive videodisc game challenges the museum visitor to place a randomly selected animal (seen in a short video segment from the videodisc) into the appropriate biome. The visitor has available one-and-a-half-minute videodisc segments about each biome, either as a reward for a correct answer or for assistance if desired. The entire exhibit is joystick driven, with no keyboard in sight. This interactive game has proven very popular with museum visitors of all ages.

Because "The Biome Game" program is very long, I am using a short demonstration

program in this article instead—one that presents a simple menu with keyboard selectable choices. Although this demonstration program is very limited in interaction, it does present the techniques necessary for you to be able to design your own interactive program.

Now, just what *is* an interactive laser videodisc system? A videodisc looks like a large CD and has on it video and sound information like videotape, but is digitally encoded like a CD and read by a laser. A videodisc is different than tape, in that each "frame" of video (there are 30 frames per second) has

an individual address number encoded on the disc. Since the videodisc player can be told to find any of these specific frame address numbers very quickly, and, when found, do some specific action, the videodisc can be used very differently than videotape. Videotape is linear—that is, you start it somewhere, and play it until somewhere later. The videodisc is not restricted that way. You can play any sequence of frames, in any order, at any speed, in any direction. You can even have it hold on a still frame for a desired duration.

Because of this flexibility, the laser disc can become truly interactive; the viewer can select among a variety of choices, paths or branches, each with its own subchoices or subpaths. It can almost immediately jump to any frame anywhere on the videodisc. The educational uses are immediately apparent in such a system, but there have also been interactive video games, museum exhibits, and now in malls, interactive kiosks where you can make purchases. Before describing the Cleveland Museum of Natural History system and requisite programming, let me mention that we are talking only of CAV laser videodiscs. Only CAV (constant angular velocity) type laser videodiscs provide the needed frame-by-frame addressability.

The Equipment

The videodisc player chosen was the Pioneer LD-V6000 (about \$1,500). It was chosen because it could easily communicate with a computer using an RS-232 interface, and do so with a minimum of required handshaking programming. The Atari 850 interface “speaks” RS-232, as does the P:R: Connection (and perhaps other interfaces). So the controlling Atari 130XE (with 1050 disk drive) was connected to Port 1 of an 850 interface, which, in turn, was connected to the videodisc player.

Since it was very desirable to use one monitor for both the computer selection screens and the video from the videodisc player, a means of switching between them had to be provided. If you just throw a switch to change the monitor input from the computer to the videodisc, you get a very unpleasant vertical roll on the screen when you switch. This is because the computer and video signals are not synchronized together. Although proper video switchers for non-synchronized sources

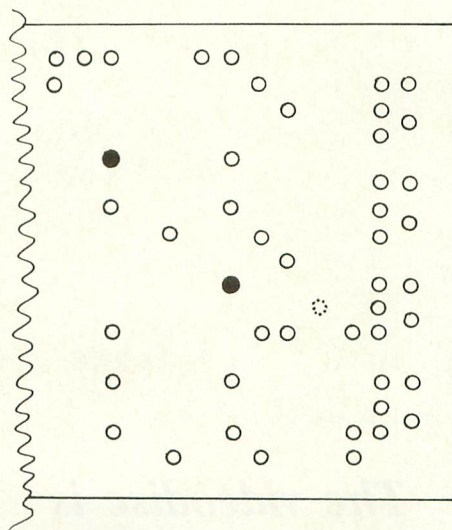


Figure 1 shows the right side of the printed circuit board in back of the front panel switches of the Radio Shack Video Special Effects Switcher. To install the relay, first unplug the switcher from the AC power. Remove the tan cover by removing six screws (two on the back at the top, four on the bottom). Pop the cover loose and slide it off toward the back. Looking at the back (inside) of the front panel, you will see the arrangement of solder pads on the printed circuit board as shown. The two filled-in pads in Figure 1 are the pads to which you tack solder the relay contact wires. Do not attempt this if you are not comfortable doing electrical work.

are very expensive, Radio Shack does have available an inexpensive (\$89.95) switcher (Archer Special Effects Switcher 15-1274).

A means of the computer “pushing the fade button” on the Radio Shack switcher had to be provided, to allow the system to function automatically. A good way to achieve this is to connect a relay (Radio Shack 275-232) coil to joystick Port 2, and the relay contacts to the “FADE” button on the video switcher. (Note: this modification probably voids your Radio Shack warranty.) When the program tells the Atari to put out voltage to the relay, the relay contacts close across the fade-button contacts, just as if someone has pushed the button. See Figure 1 for a drawing of how to connect the relay to the switcher.

Line 20 in the demonstration program sets up the Atari to be able to output voltage to the relay, and Lines 4000-4060 (the PUSH

subroutine) actually send the voltage. The techniques for connecting that kind of relay to your Atari, and controlling it through software, were best described in the July, 1986 issue of ANALOG magazine (“Bits and Pieces”, by Lee S. Brilliant, pp. 41 to 43).

Custom Cables

In order for the 850 interface to talk to the videodisc player, you will have to construct your own cable. The male end of a Radio Shack Joystick Extension Cable (270-1705) is excellent for the 850 end, and all nine wires are included and connected. Just cut the cable in half (you will use the female end for connecting the relay between the computer joystick port and the switcher). There is no consistent color coding among individual joystick extension cables from Radio Shack. You must test each individual cable for its own pin-out color coding! For the videodisc player end, you can use a Radio Shack DB-25 connector. The proper pin connections are:

Atari	RS-232
850	DB-25
=====	
1	20
2	8
3	2
4	3
5	7
6	6
7	4
8	5
9	no connection

The female cable end, for the relay connected to joystick Port 2, will have Pins 1 and 7 connected to the relay coil, as described in the ANALOG article.

The Software

In the demonstration program included with this article in Listing 1, the computer puts control programs of desired videodisc actions into the memory of the videodisc player (see next paragraph). Then it puts a menu on the screen for the user's choice. When a choice is made, the computer tells the videodisc player which program in its memory to play. The computer “pushes” the fade button on the switcher so that the videodisc is shown on the monitor. Then the computer keeps asking the videodisc player if it

is through playing the program. When it is, the computer recognizes the report from the videodisc that it is through and pushes the fade button again, so that the monitor shows the computer screen with the menu again.

This program flow requires several programming steps. Although they are REMarked in the program listing, I will briefly describe some of the steps here.

First, the RS-232 XIO commands are given (Lines 25-50) so that the Atari and the Pioneer videodisc player can communicate with each other. Be sure to use an AUTO-RUN.SYS that initializes the "R:" device when you boot the disc (such as the one created by the DOS 2.5 utility, SETUP.COM). In this program, IOCB #5 is set up as the communications channel between the computer and the videodisc player, enabling you to PRINT or INPUT strings of control information through IOCB #5. For more information on RS-232 programming on an Atari, consult the Atari 850 Interface Technical Manual.

In order to make sense of the rest of the program, it is necessary to understand how the Pioneer videodisc player works. It has its own RAM and CPU, so it is actually a little computer on its own. You can send programs of desired videodisc actions to the memory of the videodisc player. Each of these individual programs resides at a specific memory location in the videodisc player RAM. These programs of desired videodisc actions can then be called by your menu choices. Sending the control programs to the videodisc player is done during program initialization (Lines 2000-2600) because these control strings are relatively long, and the programming information shows on the screen during input.

Once these control strings are in the memory of the videodisc player, it is a simple matter to tell the player to run the program at its own memory location 0200 (or whatever) by printing that command to IOCB #5. These run commands are found in Lines 1000-1502, in the main part of the program. The advantage of this method is that you have much shorter commands to send to the videodisc player during the main program, and such program run commands don't show on

*The videodisc is
not restricted. You
can play any
sequence of
frames, in any
order, at any speed,
in any direction.
You can even have
it hold on a still
frame for a desired
duration.*

the screen when they are sent.

If the Pioneer commands in Lines 2000-2600 look odd, they are. The Pioneer requires hexadecimal pair commands. Each hexadecimal pair has a specific meaning. \$CF means run, \$D4 means transmit status, etc. These hexadecimal commands are expected by the Pioneer in ASCII—that is, an ASCII "C" followed by an ASCII "F" becomes a hex CF, the command for run. The numbers are even more peculiar, as they are sent digit by digit (coded by hex pairs), in ASCII characters. The hex codes for the numbers,

so that you can interpret the codes in the program, are:

1=0F	2=8F	3=4F
4=2F	5=AF	6=6F
7=1F	8=9F	9=5F
0=3F		

Therefore, in order to tell the Pioneer videodisc player that you want Frame 125, you would have to send (in ASCII) \$0F, \$8F and \$AF! The rest of the demonstration program is more obvious and heavily remarked, so that you can follow what is going on. Notice that menu choice "4" goes back and forth between computer text screens and videodisc still frames.

With this basic system, you can control any CAV laser videodisc in any programmed way that you want. Just select which frames you want and how you want them shown. Then design your own computer program to control the flow and accessibility of each video segment or still frame as you desire. *You* have complete control over what is shown and when.

There are excellent commercial videodiscs available. You might write to Ztek Co. (P.O. Box 54790, Lexington, KY 40555) for their catalog listing releases from many different companies. The Instant Replay (479 Winter St., Waltham, MA 02154-1216) is another good catalog source. Optical Data Corp. (66 Hanover Rd., Box 97, Florham Park, NJ 07932) has an excellent space and science series. Remember, though, that your videodisc *must* be CAV (not CLV) in order to work properly under computer control. Although the Pioneer will play CLV discs, they don't have the frame-by-frame addressability necessary for computer control.

Bruce Frumker is a past president of the Atari Computer Enthusiasts of Cleveland. He authored Memory Match for the now-defunct Atari Program Exchange and has had programs published in Compute! and Creative Computing. He has also done reviews for InfoWorld Books and ANALOG Computing. Professionally, he has been the head of photographic and audio/visual services at the Cleveland Museum of Natural History for 20 years.


```

XQ 1 REM For use with Space Archive disk
Vol. 3 Mars and Beyond
XP 2 REM With full duplex I/O and the com
puter "pushing the fade button" on the
video switcher # 15-1274
MX 3 REM Atari 850 interface - to Pioneer
LD-V6000 laser video disk player
QI 4 REM you must use an AUTORUN that ini
tializes the "R:" device
IT 10 DIM A$(500),B$(32)
TV 20 PUSH=4000:P=PEEK(54018):POKE 54018,
P-4:POKE 54016,16:POKE 54018,P:POKE 54
016,16:REM See LINE 4040 REMarks
ZP 25 REM The following XIO commands are
for RS-232 communications through the
850 interface PORT 1
WD 30 XIO 36,#5,10,0,"R":REM Aux1 10 = 1
0 (1200 BAUD) + 0 (8 BIT WORD SIZE) +
0 (1 STOP BIT)
IF 35 XIO 34,#5,240,0,"R:"
GD 40 REM the XIO 34 is required before p
rinting to #5 - Aux1 240 = 192 (TURN O
FF ON) + 48 (TURN RTS ON)
SM 50 OPEN #5,12,0,"R:"
WB 55 GOSUB 2000
IE 60 GRAPHICS 0:POKE 752,1: ? : ? : ? : ? "J
UPITER, SATURN AND MARS": ? : ? " PRESS
A NUMBER FOR YOUR SELECTION"
VJ 70 ? : ? : ? "1 - Jupiter's red spot in
motion"
FL 80 ? : ? "2 - Jupiter's red spot in col
or still photographs"
JN 90 ? : ? "3 - Computer generated approa
ch and flyover of Saturn's ring
plane"
SD 100 ? : ? "4 - Volcanic eruptions on Ju
piter's": ? " moon Io (still photogr
aphs)"
YY 120 ? : ? "5 - The first photo from the
surface": ? " of Mars comes in - Ju
ly 20, 1976"
WI 200 POKE 764,255
WB 210 IF PEEK(764)=255 THEN 210:REM has
a key been pressed?
WX 220 IF PEEK(764)=31 THEN SLO=1:GOTO 10
00
YA 230 IF PEEK(764)=30 THEN SLO=0:GOTO 12
00
DG 240 IF PEEK(764)=26 THEN SLO=0:GOTO 13
00
QY 250 IF PEEK(764)=24 THEN SLO=0:POKE 53
279,0:FOR DELAY=1 TO 50:NEXT DELAY:GOT
O 1400
JA 255 IF PEEK(764)=29 THEN SLO=0:GOTO 15
00
MD 260 GOTO 200
MZ 500 CLOSE #5:OPEN #5,13,0,"R:"
LT 501 REM The Aux1 13 in OPEN #5 is an O
PEN for INPUT and OUTPUT (12) plus 1 t
o enable Concurrent I/O
IL 505 XIO 40,#5,0,0,"R":REM XIO 40 star
ts Concurrent I/O
WZ 506 REM XIO 40 starts Concurrent I/O
SR 510 PRINT #5,A$
AY 511 GOSUB PUSH:REM this switches to th
e videodisk picture and sound
OY 520 PRINT #5,"D4EF"
HI 525 REM TRANSMIT STATUS (D4) COMMAND -
tells the video disk to transmit its
status
VD 530 INPUT #5,B$
XO 550 IF B$="65" THEN GOSUB PUSH:IF SLO
THEN 600:REM this switches back to th
e computer screen
IT 560 IF B$="65" THEN 60
NL 565 REM 65 is line feed (LF) plus free
ze frame status (65) which means the p
rogram Run is complete
OI 570 GOTO 520
CJ 600 GRAPHICS 0:POKE 752,1:SETCOLOR 2,1
2,4: ? : ? : ? : ? : ? " Would you l
ike to see that again"
AN 610 ? : ? " in SLOW MOTION ?": ? : ?
: ? " Type Y (Yes) if you do."
ZA 615 ? : ? " (Press any other key if you
don't)"

```

```

WQ 620 POKE 764,255
GE 630 IF PEEK(764)=255 THEN 630
ZR 640 IF PEEK(764)=43 THEN SLO=0:GOTO 11
00
RP 650 GOTO 60
JF 1000 A$="03F3F3F3FCF1BEF":GOTO 500
AM 1002 REM 0000 RUN (CF) VIDEO ON (1B) E
ND PROGRAMMING (EF)
HU 1100 A$="03F0F3F3FCF1BEF":GOTO 500
EO 1102 REM 0100 VIDEO ON RUN
LW 1200 A$="03F8F3F3FCF1BEF":GOTO 500
FB 1202 REM 0200 VIDEO ON RUN
JY 1300 A$="03F4F3F3FCF1BEF":GOTO 500
FO 1302 REM 0300 VIDEO ON RUN
VV 1400 GRAPHICS 2+16: ? #6: ? #6: ? #6: ? #6
: ? " JUPITER'S MOON IO"
GB 1402 REM 0400 VIDEO ON RUN
ZO 1405 ? #6: ? #6: ? #6: ? " volcano pele is
in": ? #6: ? " the lower center"
EK 1410 A$="03F2F3F3FCF1BEF"
LT 1415 FOR DELAY=1 TO 800:NEXT DELAY
CY 1420 CLOSE #5:OPEN #5,13,0,"R":XIO 40
,#5,0,0,"R":PRINT #5,A$:GOSUB PUSH
ZS 1425 PRINT #5,"D4EF":INPUT #5,B$
WV 1430 IF B$="65" THEN 1440
TM 1435 GOTO 1425
KQ 1440 GOSUB PUSH
KQ 1445 GRAPHICS 2+16:SETCOLOR 4,4,2
TJ 1450 ? #6: ? #6: ? #6: ? #6: ? #6: ? " NOTICE
THE VOLCANO": ? #6: ? " ERUPTING ON THE"
: ? #6: ? " LEFT EDGE OF IO"
CS 1455 FOR DELAY=1 TO 1000:NEXT DELAY
RU 1460 A$="03F2FAF3FCF1BEF":GOTO 500
HF 1465 REM 0400 VIDEO ON RUN
QP 1500 A$="03FAF3F3FCF1BEF":GOTO 500
GO 1502 REM 0500 VIDEO ON RUN
BE 2000 REM This subroutine loads the mem
ory of the Pioneer video disk player-
GV 2001 REM First, the Pioneer is told wh
ere in its nonvolatile memory to progr
am the commands
CG 2002 REM Then the commands to control
the video disk player are programmed a
t that memory location
VP 2005 A$="6F11":GOSUB 3000
DK 2006 REM 6 (6F) PAGE (11) MEMORY SIZE
command for the Pioneer videodisk play
er
MO 2010 A$="3F3F3F3FDF4F4F3F3F3F74F4F8F0
F8FF31CBFEF":GOSUB 3000
HC 2012 REM 0000 PROGRAM (DF) 33000 SEARC
H (F7) 33212 AUTOSTOP (F3) VIDEO OFF (
1C) HALT (BF) END PROGRAMMING (EF)
LW 2100 A$="3F0F3F3FDF4F4F3F4F5FF72FED4F4
F0F6F0FF21CBFEF":GOSUB 3000
DR 2102 REM 0100 PROGRAM 33039 SEARCH 1/4
SPEED (2D EF) 33161 MULTISPEED FORWAR
D (F2) VIDEO OFF HALT END
WT 2104 REM THE PLACEMENT OF THE 1/4 SPEE
D AFTER A SEARCH (RATHER THAN BEFORE)
IS IMPORTANT
KM 2200 A$="3F8F3F3FDF4F4F5F5F4F1FF72F3FFB4
F8F5F4F9FF72F3FFB4F8F5F4F5FF76F3FFB1CB
FEF":GOSUB 3000
OH 2202 REM 0200 PROGRAM 32937 SEARCH 4.0
SEC WAIT (2F3F FB) 32938 SEARCH 4.0 S
EC WAIT etc.....
ID 2300 A$="3F4F3F3FDF4F4F9F9F3FF74F2FAF5
F3FF31CBFEF":GOSUB 3000
RA 2400 A$="3F2F3F3FDF4F4F9F4F2FF76F3FFB1
CBFEF":GOSUB 3000
GB 2410 A$="3F2FAF3FDF4F4F9F4F9FF7AF3FFBF
62F3FFBF62F3FFB1CBFEF":GOSUB 3000
MF 2500 A$="3FAF3F3FDF0F0F9FAF1F3FF70F5F2F1
F3FF31CBFEF":GOSUB 3000
AM 2600 RETURN
BJ 3000 PRINT #5,A$:RETURN
YF 4000 POKE 54016,0
ML 4010 FOR DELAY=1 TO 30:NEXT DELAY
KM 4020 POKE 54016,16
AL 4030 RETURN
KU 4040 REM The above subroutine is expla
ined in ANALOG Computing magazine - Ju
ly, 1986, Pg. 41-43
CU 4050 REM As programmed, this sets joys
tick 2 to output, and outputs voltage
from pins 1 and 7 (com)
ZH 4060 REM This operates the # 275-232 r
elay that is connected to the video sw
itcher FADE button

```


DUPing BASIC

by Bill Bodenstein

There have been several simple machine-language programs written to toggle BASIC ROM in an XL/XE machine on and off, but no one has tried modifying the DUP.SYS file to do this automatically. At least, not until now.

Since BASIC is invisible to the DOS 2.5 Disk Utility Package (DUP), why not disable it each time you go to DOS? And whether or not the BASIC cartridge was originally on, why not enable it each time you use the RUN CARTRIDGE function (option B)?

Typing it in

Type in Listing 1 using the BASIC Editor II found elsewhere in this issue, and save it. Now insert a diskette with DUP.SYS (DOS 2.5 version only) and run the BASIC program. A few short modifications will be appended to your DUP.SYS file. Once modified, use the DUPLICATE FILE function instead of WRITE DOS FILES to put copies onto your other disks. The reason: the initialization routine that disables BASIC is stored in memory outside the DUP code, and thus, is not saved when using the WRITE DOS FILES menu

option. Look over Listing 2, the MAC/65 source code, to better understand the changes made.

Using your new DUP

Just type "DOS" from BASIC and hit Return as usual. If, for some reason, you would like the cartridge to remain enabled, hold down Select while DOS is being loaded. This may be necessary if you intend to binary load a file that checks for BASIC first or makes an illegal exit from DOS.

With the added memory "underneath" BASIC now at your disposal, copying files will frequently require fewer passes. And as a reminder, the word "mod" has been placed above the DOS menu in the title line, so you'll always know you're using a modified DOS.

Exit DOS with the RUN CARTRIDGE function, and BASIC will be enabled again—or for the first time if you booted with Option held down. If you should happen to use this modified DUP program on a non-XL/XE machine (400/800 model), fear not: the changes only affect Atari XL/XE systems.

LISTING 1: BASIC

```

HY 10 REM *****
GY 11 REM *      BASIC DUPER      *
EV 12 REM *      By Bill Bodenstein  *
ZW 13 REM *      *
BL 14 REM *      COPYRIGHT 1988      *
SS 15 REM *      BY ANALOG COMPUTING  *
IK 16 REM *****
BM 17 REM
RV 50 DIM A$(1)
ZU 60 ? :? "This program modifies a DOS 2
.5";? "DUP.SYS file to disable BASIC w
hen"
QJ 70 ? "loaded, and enable BASIC when ex
ited";? "with the RUN CARTRIDGE functi
on.";?
V5 100 CLOSE #1:? "Insert a diskette with
DUP.SYS";? "and hit <RETURN>":INPUT A
$
JA 110 TRAP 150:OPEN #1,4,0,"D:DUP.SYS"
Y5 120 FOR X=1 TO 47:GET #1,N:NEXT X
EZ 130 IF N=ASC("S") THEN 200
ON 150 ? "CAN'T FIND DOS 2.5 DUP.SYS FILE
":GOTO 100
UG 200 RESTORE :TRAP 300:CLOSE #1:OPEN #1
,9,0,"D:DUP.SYS";? "Modifying..."
NL 210 READ N:IF N>-1 THEN PUT #1,N:GOTO
210
MV 220 CLOSE #1:? "DUP.SYS file has been
modified.";? "Type 'DOS' to load.":END
MN 300 ? "ERROR - ":PEEK(195):STOP
UG 500 DATA 128,5,157,5,173,31,208,201,5,
240,22,173,1,211,9,2
CX 510 DATA 205,1,211,240,12,141,1,211,16
9,1,141,248,3,169,192,133
TJ 520 DATA 106,96,226,2,227,2,128,5,80,3
9,134,39,169,0,141,248
FS 530 DATA 3,141,0,212,173,1,211,41,253,
141,1,211,172,253,191,206
DZ 540 DATA 253,191,204,253,191,240,13,14
0,253,191,169,159,162,39,32,176
WR 550 DATA 49,76,15,33,169,160,133,106,1
73,252,191,208,237,173,253,191
MW 560 DATA 240,232,234,41,31,53,31,86,69
,02,83,46,32,178,174,181
LB 570 DATA 32,109,111,100,-1

```

LISTING 2: ASSEMBLY

```

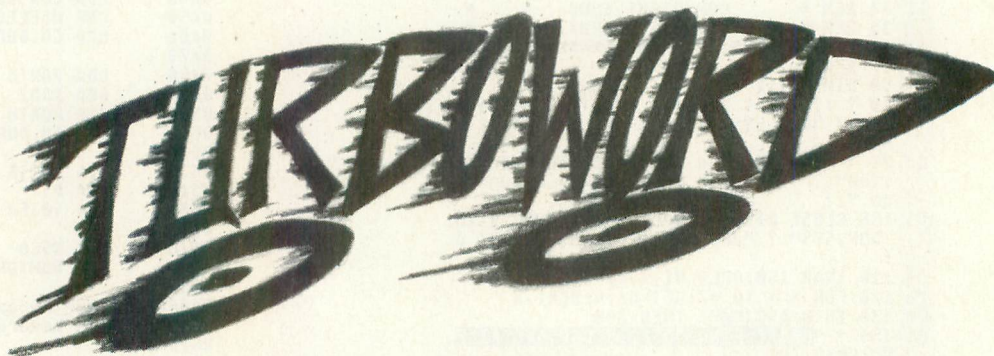
10 *****
20 *      BASIC DUPER      *
30 *      By Bill Bodenstein  *
40 *      *
50 *      COPYRIGHT 1988      *
60 *      BY ANALOG COMPUTING  *
70 *****
80 ;
90 ;This program modifies DUP.SYS
0100 ;(DOS 2.5 version only) to make
0110 ;it disable BASIC when loaded,
0120 ;and enable the BASIC cartridge
0130 ;when exiting with the RUN
0140 ;CARTRIDGE function.
0150 ;Append this code to your
0160 ;DUP.SYS file.
0170 ;
0180 *** EQUATES ***
0190 ;
0200 CONSOL = $D01F ;Button pressed?
0210 SELECT = 5 ;This button
0220 ;
0230 PORTB = $D301 ;ROM or RAM?
0240 ;
0250 DMACTL = $D400 ;Screen on/off?
0260 ;
0270 RAMTOP = $6A ;Num RAM pages
0280 ;
0290 INITADR = $02E2 ;Bin ld vector
0300 ;
0310 ;
0320 *-----*
0330 ; Disable BASIC cartridge.
0340 *-----*
0350 ;Before DUP takes over, turn the
0360 ;cart off unless ISELECT

```

```

0370 ;pressed. This routine is only
0380 ;used once, so store it in the
0390 ;text buffer.
0400 ;
0410 *= $0580
0420 ;
0430 CART.OFF
0440 LDA CONSOL ;[Select] being
0450 CMP #SELECT ;pressed?
0460 BEQ GO.DUP ;Exit if yep
0470 ;
0480 LDA PORTB ;ROM/RAM status
0490 ORA #502 ;Bit 1 tells us
0500 CMP PORTB ; if cart on/off
0510 BEQ GO.DUP ;Exit if off
0520 ;
0530 STA PORTB ;Else turn off
0540 LDA #1 ;Tell o.s. to
0550 STA $03F8 ; keep cart off
0560 ;
0570 LDA #5C0 ;Give us more
0580 STA RAMTOP ; free memory
0590 ;
0600 ;Let DUP re-open editor to move
0610 ;screen memory up in RAM.
0620 ;
0630 GO.DUP
0640 RTS ;Finish bin load
0650 ;
0660 ;Remember: DUP.SYS is loaded
0670 ;like any binary file. So we can
0680 ;use the initialize jump vector.
0690 ;
0700 *= INITADR
0710 .WORD CART.OFF
0720 ;
0730 ;
0740 *-----*
0750 ; Enable BASIC cartridge.
0760 *-----*
0770 ;We'll stick this routine right
0780 ;in the RUN CART function code.
0790 ;We'll then need to re-write the
0800 ;original DUP code, condensing
0810 ;it to now fit in the same area.
0820 ;
0830 *= $2750
0840 ;
0850 CART.ON
0860 LDA #0 ;Cart on in case
0870 STA $03F8 ; of IRESET
0880 STA DMACTL ;Scr temp off
0890 ;
0900 LDA PORTB ;Memory status
0910 AND #5FF-$02 ;Clr bit 1 to
0920 STA PORTB ; turn cart on
0930 ;
0940 ;
0950 ;Now, we re-write part of the
0960 ;original code for DUP's RUN
0970 ;CARTRIDGE.
0980 ;
0990 LDY $BFFD ;See if this
1000 DEC $BFFD ; byte ROM/RAM
1010 CPY $BFFD
1020 BEQ YES.CART
1030 STY $BFFD
1040 NO.CART
1050 LDA #$9F ;Print "no cart"
1060 LDY #$27 ; msg and exit
1070 JSR $3180
1080 JMP $210F
1090 YES.CART
1100 LDA #$A0 ;Reset # free
1110 STA RAMTOP ; RAM pages
1120 LDA $BFFC ;Check if cart
1130 BNE NO.CART ; can be
1140 LDA $BFFD ; re-entered
1150 BEQ NO.CART
1160 NOP ;Re-align code
1170 ;
1180 ;Rest of code remains the same.
1190 ;
1200 ;
1210 *-----*
1220 ; New menu header.
1230 *-----*
1240 ;Replace the original DOS info
1250 ;with our text.
1260 ;
1270 *= $1F29
1280 ;
1290 .BYTE "VER5. 2.5 mod"

```

Reviewed by Matt Ratcliff

Micromiser has brought to market the first full-featured 80-column word processor for the 8-bit Atari computers. If you have been waiting for a good excuse to get an XEP80, *Turboword* is it!

Turboword comes on an Atari DOS 2.5 disk, single density. You are first instructed to make a backup copy, since it is not copy protected. This lack of copy protection is a nice feature, allowing *Turboword* to run from any DOS you choose, including *SpartaDOS*. (I was also able to install *Turboword* on my hard drive.) However, *Turboword*, although *SpartaDOS* compatible, will not run from a subdirectory. This isn't good for hard drive users, who must organize everything in subdirectories because of filename limitations. But I have not had any problems storing *Turboword* in a subdirectory, and then copying everything to an MIO RAMdisk and running it from there.

Turboword comes with an AUTORUN.SYS file that loads the XEP80 handler and starts up the main menu, which requires BASIC. The XEP80 handler for *Turboword* is built into the AUTORUN files provided, one for

Atari DOS and another *SpartaDOS*. The biggest problem with this setup is that the XEP80 handler version supplied is for driving the printer through the XEP80 interface. There are no provisions for using a printer on an MIO, 850 or P/R: Connection interface. There is, however, a utility on the Atari XEP80 boot disk which allows you to modify the original handler for use with other printer interfaces. It took a bit of hacking to get the AUTORUN file provided by Micromiser modified to allow the use of my MIO printer interface.

Once all the setup work was completed for *SpartaDOS* use, I had *Turboword* up and running with no other problems. As mentioned earlier, *Turboword* requires BASIC to run the main menu. The word processor itself is an assembly-language routine, as is the spelling checker. This approach allowed the author Steve Bolduc to modularize everything, using a simple BASIC program as the main control center. It works well, although it costs you 8K of buffer space under Atari BASIC.

Turboword first presents an 80-column menu, the control center for file management

and printing. The first step is to specify a file you wish to edit with the Name option. The Editor option then will turn control over to the editing screen. If the file already exists, it will be loaded and displayed, with the text formatted between a pair of vertical bars, set at your predefined margins. These margins may be reprogrammed from the main menu or while editing.

Editing commands are control-key combinations, such as Control-I to toggle between insert and overwrite mode. If you are stumped at any time, simply press the Help key for a menu of editing commands, a nice touch. Normally you will edit in the replace mode. While in insert mode, text is pushed down the screen if necessary. Inserting text is a very slow process and can really frustrate a good typist. Normally insert mode will be employed only for small changes.

When you need to insert several lines of text, simply press Shift-Help. This places the cursor on a blank screen, allowing you to type your insertion at top speed. When completed, Help will take your block and place it at your original cursor position, reformatting

TURBOWORD

MICROMISER

SOFTWARE

1635-A Holden Ave.
Orlando, FL 32809
(407) 857-6014
Requires: XEP80 80-column
board, 80-column monitor,
one disk drive, Atari XL or XE
\$49.95

and redrawing the display as quickly as possible. This is a very nice touch, providing the best of true 80-column word processors (what-you-see-is-what-you-get and continuous page formatting), and the best performance possible given the limitations of the driver software for the XEP80 itself.

There is a minor, but annoying problem with the parser. It is proper to enter two spaces after the end of a sentence, before starting the next. *Turboword* should recognize this when it word-wraps, as all other word processors I've used do. However, if a word-wrap occurs just at the end of a sentence, the following line of text may be indented by one space instead of being left justified. In such instances you must go back and manually edit out one space.

Pressing the Escape key redraws the display. If Control-R is pressed first, all spaces and Returns are displayed as underlines and diamonds, respectively. Other control-key command features include Control-U for cursor blink toggle, Control-F to find a string of text, Control-L to show a ruler and so on. There is no command for search and replace operations; you must make all changes manually.

Turboword provides about 25K of text buffer space, due to its modular design, since code for the main menu is not in memory while you are editing. Pressing Control-P displays a "gas gauge" at the top of the screen, showing your current position in memory and indicating total buffer space remaining.

Turboword uses inverse video characters for all special text formatting and markers. From the main menu, you may assign printer codes. The most common ones are already defined, such as italics, bold, underline, superscript and subscript and so on. To print a word in italics for example, you would precede the word with an uppercase inverse video "I" followed by a lowercase inverse video "i" to turn it off. When *Turboword* comes across these command characters

while printing, they are replaced with the appropriate command strings from the previous assignments.

A pair of inverse video asterisks mark off a block of text. Block operations include Control-E to erase the markers, Control-D to delete, Control-C to copy and Control-M to move. The block delete function always sends the cursor to the end of the current file, instead of retaining its current position. This can be annoying. Multiple editing buffers or separate windows are not supported, as in *PaperClip*.

Pressing Option returns control to the main menu, after saving the file you are currently editing, *always*. If you make irrevocable mistakes in your document, you must press Reset to prevent *Turboword* from saving the file. An abort menu option really should have been included so that this problem could have been avoided.

Pressing "J" from the main menu sends control to the spell checker; your document is then checked against a series of word files. The spell checker lists the entire line on which a word in question appears, and you are then given the option to skip, fix or add this word to the current dictionary. This is a good checker that is convenient to use.

The spelling checker does have its limitations, however. The dictionary files are not exactly overflowing with words, and I've found myself adding a lot of fairly simple words. The spell checker isn't smart enough to ignore words with numbers in them, such as XEP80. Nor does it provide the option to skip proper names. Apparently plurals, past tense and "ing" words must be added to the dictionary files separately, since the spell checker is not smart enough to figure these out on its own. Once you add a word to the dictionary, the spell check still prompts you to repair this misspelled word. Added words do not take effect until the next time you spell check a document. This can be annoying.


The "P" option of the main menu will print

your file. Other printer controls from the main menu include margin setting, lines printed per page, overall lines per page and hold between pages to allow for single-sheet feed paper.

The assign-printer-codes option allows you to view and modify the current print-control features. You can redefine the commands provided, or add your own. You will need a good printer manual handy for reference. I had no problems defining the commands for font and color selections on my NX1000 Rainbow printer. The printer codes are shown as a string of decimal characters, following the letter command. It would have been nice to also attach a short text description to each, similar to that shown at the back of the *Turboword* manual. I ended up creating a separate text file to use as a reference guide for these commands.

Turboword provides advanced address-handling options, designed to interface nicely with their *Turbobase*, database package. User-defined macros may be created. Form letters are easily generated with your documents and address files.

Turboword is not the most advanced word processor I have ever used. There is no provision for double spacing documents. The manual is only 28 pages, including only minimal documentation for all the commands. Sample menu displays, tutorials and an index would be welcome improvements. The spell checker is limited, but user-expandable. However, *Turboword* does a good job of printing and seems bug-free (no crashes, or unexpected losses of text). Further, it is extremely flexible and, with the user-definable macros and printer codes, simple to customize.

This program is not perfect and lacks many features. If the program sells well, I am certain enhancements will be made in future updates. If you have an XEP80, or have been wanting to get an XEP80 for true 80-column word processing, *Turboword* is a good choice. 

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

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),H$(4),A$(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)
ZC 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$(A)="5" THEN 120
FD 90 TRAP 430:OPEN #2,4,0,F1$:TRAP 110
HO 100 FOR K=1 TO 16:GET #2,A:NEXT K:LINE
=LINE+10:GOTO 100
HM 110 CLOSE #2:OPEN #2,9,0,F1$:GOTO 170
UT 120 TRAP 160:OPEN #2,4,0,F1$: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$(A)
VH 140 IF CHR$(A)="N" OR CHR$(A)="n" THEN
CLOSE #2:GOTO 30
OG 150 IF CHR$(A)<>"Y" AND CHR$(A)<>"y" T
HEN 130
BH 160 CLOSE #2:OPEN #2,8,0,F1$
XE 170 GOSUB 450:POSITION 10,1:?"NO MORE
DATA":LINE=CHKSUM=0
GN 180 L1=3:FOR K=1 TO 16:POSITION 13*(K
10)+12*(K)9),K+2:POKE 752,0:?" BYTE #
":K:":GOSUB 310
KH 190 IF EDIT AND L=0 THEN BYTE=BF(K):GO
TO 210
FY 200 BYTE=VAL(N$)
OZ 201 MOD$=N$
BU 210 POSITION 22,K+2:?" BYTE:" "
YZ 220 BF(K)=BYTE:CHKSUM=CHKSUM+BYTE*(K:IF
CHKSUM>9999 THEN CHKSUM=CHKSUM-10000
M3 230 NEXT K:CHKSUM=CHKSUM+LINE:IF CHK5U
M3 230 THEN CHKSUM=CHKSUM-10000
IG 240 POSITION 12,K+2:POKE 752,0:?"CHEC
KSUM: ";L1=4:GOSUB 310
EW 250 IF EDIT AND L=0 THEN 270
QM 260 C=VAL(N$)
SY 270 POSITION 22,K+2:?" C:" "
280 IF C=CHKSUM THEN 300
DI 290 GOSUB 440:EDIT=1:CHKSUM=0:GOTO 180
LM 300 FOR K=1 TO 16:PUT #2,BF(K):NEXT K:
LINE=LINE+10:EDIT=0:GOTO 170
FU 310 L=0
KZ 320 GOSUB 500:IF (A=ASC("Q")) OR A=ASC(
"q") AND K=1 AND NOT EDIT THEN 420
PO 330 IF A=RETRN AND A<>BACKSP AND (A<4
0 OR A>57) THEN 320
DX 331 IF A=RETRN AND N$="" THEN N$=MOD$
TD 335 IF A=RETRN AND L=0 AND K>1 THEN 35
0
JR 340 IF ((A=RETRN AND NOT EDIT) OR A=B
ACKSP) AND L=0 THEN 320
DM 350 IF A=RETRN THEN POKE 752,1:?" ":R
ETURN
GG 360 IF A<>BACKSP THEN 400
SA 370 IF L>1 THEN N$=N$(1,L-1):GOTO 390
AS 380 N$=""
RE 390 ? CHR$(BACKSP):L=L-1:GOTO 320
BB 400 L=L+1:IF L>1 THEN A=RETRN:GOTO 35
0
HX 410 N$(L)=CHR$(A):?" CHR$(A):GOTO 320
KN 420 GRAPHICS 0:END
YT 430 GOSUB 440:POSITION 10,10:?"NO SUC
H FILE!":FOR K=1 TO 1000:NEXT K:CLOSE
#2:GOTO 30
FD 440 POKE 710,48:SOUND 0,100,12,8:FOR K
-1 TO 50:NEXT K:SOUND 0,0,0,0:RETURN
MY 450 GRAPHICS 23:POKE 16,112:POKE 53774
,112:POKE 559,0:POKE 710,4
XR 460 DL=PEEK(560)+256*PEEK(561)+4:POKE
DL-1,70:POKE DL+2,6
HM 470 FOR K=3 TO 39 STEP 2:POKE DL+K,2:N
EXT K:FOR K=4 TO 40 STEP 2:POKE DL+K,0
:NEXT K
ZW 480 POKE DL+41,65:POKE DL+42,PEEK(560)
:POKE DL+43,PEEK(561):POKE 87,0
AC 490 POSITION 2,0:?"analog ml editor":
POKE 559,34:RETURN
MZ 500 OPEN #1,4,0,"K:":GET #1,A:CLOSE #1
:RETURN
```


End End User

by Arthur Leyenberger

It's hard to believe but it's true. This month's column marks the fourth anniversary of *The End User*. That's right, for 48 months—give or take a couple—I have been sitting right here transmitting my thoughts into the computer via these stubby little fingers, and then ultimately to these pages every month. Looking at it another way, some 120,000 words have found their way into this column over that period of time. The mind reels.

Since I started writing this column, I have seen many companies, products and phenomena come and go. I have witnessed the growth and death of the "old Atari," the birth of the "new Atari," the transformation of the "new Atari" into the "same old Atari," and what appears now to be the "new, old, new Atari." Confused? Me too!

End User Highlights

In any case, I thought it would be interesting to journey back in time to 1985 and look at what has concerned us throughout the last four years as dedicated and sometimes not-so-dedicated Atari users. We and Atari all have come a long way since then. In the early 1980s, many of us felt like Rodney Dan-

MARCH A.N.A.L.O.G. Computing

*Some 120,000 words
have found their
way into this
column over 48
months. The
mind reels.*

gerfield. Our Atari 8-bit computers got no respect from most people, especially the Apple II users who had a more popular computer even though our Atari 800s were in many respects more powerful and easier to use.

Our story begins in April 1985 with the first installment of this column. Atari was indeed a fresh, new Atari, having just been taken over the previous summer by Jack Tramiel and sons. Their debutante ball was the January Consumer Electronics Show where the world was first introduced to the Atari ST and "Power Without the Price."

The ST blew everybody away. It had graphics equal to or better than anything else available at the time, a mouse-controlled, friendly user interface (the GEM Desktop) and a fast 68000 processor. It seemed clear at the time that the ST was truly the computer for the rest of "the rest of us."

The first *End User* contained an interview with Rob LaTulipe of Digital Research. Digital Research was the designer and developer of GEM, and at the time they had a close, working relationship with Atari. The primary thrust of the interview was to better understand the new Graphics Environment Manager software for the ST.

One of the exciting aspects of GEM that Rob discussed was that it is not hardware dependent. Rather, it is portable and any software application designed to work under GEM can run on any machine. This meant, for example, that *GEM Draw* which has been written for MS-DOS computers could fairly easily be ported to run on the Atari ST. In fact many developers actually used IBM PCs or clones in the early days to develop GEM software for the ST.

Unfortunately the business relationship between Atari and Digital Research deteriorated within a year or so. In addition, the relatively small ST software market prevented the GEM software designed for the PC—*GEM Draw Plus*, *GEM Wordchart*, *GEM Graph*—to ever become available for the ST.

The next couple of *End Users* were more lighthearted. An interview with Atari-founder Nolan Bushnell and his robotic pets (remember the Catster?) was interesting since Bush-

nell always seemed to start flamboyant companies. On the recently introduced Atari ST he commented, "The Macintosh and the GEM system are attempts to make computers what it [sic] should be. It is a step in the right direction, but I feel that they are not really breakthrough products."

Rereading the interview four years later, I get a strong sense that Mr. Bushnell was engaging in a little "techno-babble." When asked what a breakthrough product is and what do computers have to become in order to be what they *should* be, he refused to comment saying that this is a competitive industry and he didn't want to talk too much about it. When you recall Chuck E. Cheese Pizza Parlors, Androbot, Axlön and who knows what other "now-you-see-'em, now-you-don't" companies that Bushnell started, his credibility wanes.

In October 1985 I got to talk about my experiences with the MIDI (Musical Instrument Digital Interface) ports on the ST. One of the strengths of the ST then and now is the built-in MIDI ports that allow any MIDI-compatible device to be controlled by the computer. Playing with MIDI and the Casio CZ-101 keyboard were fun. MIDI programs have come far since those days and the prices of MIDI-equipped instruments has remained constant or decreased.

By the end of the year I had finally gotten my hands on an ST and was able to get some intimate experience with it. Almost immediately I discovered the "loose chips" problem. It seemed that many of the early STs had some of their chips work loose during shipping. The result was that many STs were dead on arrival or soon exhibited erratic and intermittent behavior. Fortunately, Atari acknowledged the problem and soon instituted a fix. Loose ST chips have not been a problem for a couple of years.

Ironically, I lamented the lack of software for the infant ST. Further, I discussed that many companies were not developing software for the ST wanting to take a "wait and see" attitude. They wanted to see what kind of market developed for the ST before they would commit their resources to support the machine.

*Now, many
software companies
refuse to develop
software for the ST,
wanting to take a
“wait and see”
attitude on the
future of Atari and
the ST.*

Funny, at this point in the Atari adventure, the U.S. is the secondary (if not tertiary) market—the main ST market is in Europe. Now, many software companies refuse to develop software for the ST, wanting to take a “wait and see” attitude on the future of Atari and the ST. Atari is at a real turning point which may well have been decided by the time you read this.

Another one of my first reactions to the ST was an immediate dislike for the keyboard. To me, the diagonal shape of the ten function keys across the top of the keyboard almost guaranteed that I would press two function keys when trying to press just one. The problem disappeared after I became more familiar with the keyboard. A more serious problem for me was the main keyboard.

Initially, and since that time, I have had problems typing on the ST. Specifically, I make a lot of typing mistakes. The individual keys are about 20% larger than the PC clone and PC laptop keyboards I use during my day job. Also, the ST keyboard key spacing is about twice that of my other keyboards. Perhaps my problem lies in using several different keyboards. If I used only an ST, I would probably not have a problem.

Overall, my early reaction to the ST was extremely positive, which has not changed over the years. The monochrome monitor is still as crystal-clear as it was the first day I saw it, the color RGB monitor still is one of the best color monitors I have seen on any computer, and the GEM Desktop is still an easy way for both novices and expert users to control their machines.

The new year, 1986, brought Apple bullying Digital Research into changing the “look and feel” of the GEM Desktop. Little did we know that Apple Computer was starting a trend of “look and feel” litigation. Fortunately, it didn’t affect the ST, but it *did* affect GEM on the PC. I use GEM software on a PC a lot, and the new version of the GEM Desktop is a brain-damaged concoction. For many, the new version of the Desktop defeats the purpose of the GEM interface by making it more difficult to copy files and move throughout the Desktop.

For the first six or eight months that the ST became available, the popular press as well as the computer press gave it little notice. Instead, the Commodore Amiga got the news coverage and glory even though it was

released later than the ST. At the same time, the big computer magazines were writing about preproduction Amigas when they could have walked into their local Atari dealer and picked up a *production* ST computer. It still burns me.

In February 1986, I was able to finally articulate my views of Atari and their products from a user’s point of view. I said, “Simply put, I like the Atari product, but I don’t always like the way Atari Corp. does business.” This was the first of many times when I stated that my goals and Atari’s were the same: I want them to succeed. As a user, I really *do*. But they continually do things that appear to alienate users, dealers and developers and make it more difficult for the ST to be “the computer for the masses not the classes.”

The year of 1986 was also the year (specifically January) that Atari first showed the elusive CD-ROM player for the ST. The ST and CD-ROM combination was fantastic. I’d say everyone who saw it wanted one, myself included. But Captain Jack was playing the “stealth product” game: briefly show a new product, get some reaction to it, and then somehow it mysteriously never appears again. Atari claimed they were waiting for CD-ROM hardware prices to drop. We’ve been waiting for more than three years. Will 1989 be the year of the Atari CD-ROM? Beats me!

When the ST was first released, TOS (operating system) and GEM were disk-based. You had to boot the ST from the floppy in order to load the operating system. Within the year, TOS became available in ROM, a set of six chips. Atari claimed they wanted to get the bugs out of the operating system before they released it on ROM. This was the first of many examples of Atari using the user base as beta testers for their products. Three years later, we are still waiting for a newer, faster, bug-free version of TOS. When will it arrive? You got me!

Within a year after the introduction of the 520ST, Atari announced and delivered the 1040ST. Again, Atari came through with a breakthrough product: one megabyte of RAM, built-in disk drive, all for under \$1,000. However, for the 520ST owners, there was no upgrade path from Atari. If you wanted a megabyte of memory, you either bought the new machine or had an aftermarket memory upgrade installed in your 520. This lack of upgrades for users was to become a hallmark of the new Atari Corp.

Captain Jack was playing the "stealth product" game: briefly show a new product, get some reaction to it, and then somehow, it mysteriously never appears again.

In 1987 we saw the rebirth of video games. The new (old) Atari 7800 was reintroduced. Sales of 2600 video games, surprisingly, went through the roof. Sega and Nintendo successfully introduced new game machines. Atari was doing well in video games and computers too. Things looked pretty rosy. Atari, in a real "Marketing 101" move, repackaged the 65XE into the XE Game System, calling it a new video-game machine that could also perform real computer functions.

Also, that year saw Atari introduce some additional products, such as their PC clones. When first announced, the Atari PC was a bargain, once again offering low-price computing to the masses. But by the time the machine finally arrived on the street (yet to see an American street), it was yet another clone that was no longer price competitive and, worst of all, lacked slots for adding additional cards.

The year 1987 also brought rumors about a new high-end Atari computer. Called the TT or ST enhanced, it was to use a 68020 processor and perhaps run with the Unix operating system. As you know, it has yet to be seen outside the laboratory.

I would be remiss if I failed to mention one of the main events of 1987/1988. ANALOG Computing was purchased by LFP, Inc. As a result of this deal, ANALOG has become even better. More reviews, more Atari information and a much more professional appearance. ANALOG is now a world-class magazine.

And that, my friends, just about brings us up to date. In the last year or so, Atari has become a public company, making them stronger financially. More recently, Atari has acquired the Federated Group electronics stores. That, for the time being, has made them weaker financially. Mail-order dealers have been given the boot and the dealer network, although continuing to shrink, has become stronger due to better support from Atari.

The End

There you have it. Four years of the *End User*. Four years of the new Atari Corp. Four years of the Atari adventure.

The last four years have been fun. We have shared the exciting saga of Atari in good times and bad, and there are few things I enjoy more than meeting with you here once a month.

In this column, I have tried to mention most of the significant events that occurred during the past four years. But, of course, there were other important events—some regrettable, such as the loss of such companies as Synapse, Batteries Included and OSS from both the 8-bit and ST software markets. These and other companies had supported the Atari user with quality products for many years. There are now only a handful of top notch software firms producing Atari software. You know who they are.

The time went by awfully fast, and the computer market has changed drastically. In 1985 the ST was clearly the best deal in town for high-powered, economical computing. Today—well—the typical PC clone with EGA (Enhanced Graphics Adapter) or VGA (Video Graphics Array) graphics can equal or better the ST's graphics.

Run-of-the-mill AT clones with 80286 processors out-perform the ST's 68000 processor. Where is the next generation Atari machine? The Mega is simply a repackaged ST with more memory. Maybe Atari will introduce their new high-end computer this year. I hope so.

As mentioned above, the U.S. ST market is a poor second to the European and Asian marketplaces. Atari claims the DRAM (Dynamic Random Access Memory) chip shortage is responsible for the scarcity of computers in the land of the free. They further claim that it makes no sense to advertise when there is so little product to ship.

At the end of the 1980s, Atari appears to lack focus. Their revolving-door school of management only worsens the problem. As ST users, we have gone from loose chips to no chips. For the future to be bright, Atari must make some significant changes. There is no question they can do it if they want to.

To quote Bob Hope, thanks for the memories. Regardless of the Atari future, I will always cherish these last four years. I am thankful for the friends I have made within the Atari community, the enormous amount I have learned about computing, and the forum and opportunity I have been given by ANALOG. I am proud that I have always called 'em like I see 'em, and I am fortunate that ANALOG has had the courage to allow me to do just that.

Whatever happens, we will always be End Users.

Game Design Workshop

by Craig Patchett

This month we'll finish our study of Player/Missile (P/M) graphics and, at the same time, finally get back to work on BASIC Invaders, that game program we started way back when. Those of you who need to catch up on where we are so far with BASIC Invaders should see Listings 1 and 2. To create the program, type Listing 1 and save it to disk. Then type Listing 2 and run it (after saving a copy). The program will construct some strange-looking BASIC lines filled with machine-language code in character format. When the program is finished, type LIST "D:MACHINE",29000,32510 to store the new program lines to disk. Now load Listing 1 back into memory, and type ENTER "D:MACHINE" to merge the lines created by Listing 2 with the lines from Listing 1.

Moving Onward

We will be using P/M graphics to create the "base" in BASIC Invaders. The first step in doing this is to clear the memory we need and to place the data for the base into that memory. We'll be using Player 0 for our base. Add the following code to the BASIC Invaders program.

```
3070 DIM MEMCLR$(36):GOSUB
30500:MEMCLR$=MLANG$
3150 PB=PEEK(740)-8:CB=PB-
4:POKE 106,CB-4:CA=CB*256:
PA=PB*256
3250 X=USR(ADR(MEMCLR$),PA
+768,1280)
3270 FOR BYTE=201 TO 208:R
EAD DAT:POKE PA+1024+BYTE,
DAT:NEXT BYTE
3280 DATA 16,16,56,56,124,
124,198,198
5200 POKE 54279,PB:POKE 55
9,62:POKE 704,10
5480 POKE 53277,3
```

Here is an explanation of the above code segment:

3070: Set up the machine-language routine MEMCLR, which is used to clear memory.

3150: Set aside memory for PMG and the character set.

3250: Clear the PMG area and set a few memory locations to get the computer ready for turning PMG on.

3270-3280: Set up the player's base in Player 0.

5200: Turn the screen back on.

5480: Turn on PMG.

Now we need to add the PMOVE routine into the program:

```
3120 FOR BYTE=1 TO 40:READ
DAT:POKE 1737+BYTE,DAT:NE
XT BYTE
3130 DATA 252,243,207,63,0
,128,0,128,128,2,2,3,3,1,0
,0,0,0,0,4,5,6,7,3,76,128
3140 DATA 64,76,80,64,76,1
77,64,76,5,65,76,88,65,0
3180 MEM=PA
3190 FOR SEC=0 TO 7:GOSUB
32000+10*SEC:X=USR(ADR(MOV
MEM$),ADR(MLANG$),MEM,LEN(
MLANG$)-1)
3200 MEM=MEM+LEN(MLANG$):N
EXT SEC
5470 X=USR(PA,PB,PB)
```

Here's what we're doing:

3120-3140: Set up some data for PMOVE, the PMG machine language routine.

3180-3200: Set up PMOVE.

5470: Turn on PMOVE.

Now let's make use of PMOVE by attaching the joystick, to our Player 0:


```

5300 POKE 1664,0:POKE 1665
,255:POKE 1666,255:POKE 16
67,255
5310 POKE 1668,50:POKE 167
2,200
5320 POKE 1676,201:POKE 16
80,201
5330 POKE 1684,128:POKE 16
88,201

```

Here's the explanation:

5300: We're attaching Player 0 to Joystick 0 and not using the other players.

5310: The left limit for Player 0 is 50, the right is 200.

5320: The upper and lower limits are the same, which means that Player 0 is not allowed to move vertically.

5330: The initial horizontal position of Player 0 is 128, and the initial vertical position is 201.

Now try moving the base from side to side with Joystick 0. As you can see, PMOVE really does the trick.

Moving a Player Without a Joystick

I mentioned that PMOVE can move a player without it having to be attached to a joystick so let's look at an example of this. In our BASIC Invaders, we want to have an alien spaceship fly across the top of the screen from time to time. We'll make it so that it flies each time the invaders reach the right-hand side of the screen. Here's the code to do it:

```

1270 IF COARSE<>4 AND COAR
SE<>-2 THEN 1330
1280 CHANGE=-CHANGE:POKE 1
791,129-PEEK(1791)
1290 IF CHANGE=1 THEN 1330
1300 POKE 1685,235:POKE 16
86,240:SAUCER=1
1310 POKE 675,11
1370 IF SAUCER=1 AND PEEK(
1686)<40 THEN SAUCER=0:POK
E 675,15
3290 FOR BYTE=30 TO 38:REA

```

```

D DAT:POKE PA+1280+BYTE,DA
T:READ DAT:POKE PA+1536+BY
TE,DAT:NEXT BYTE
3300 DATA 15,16,31,24,63,2
8,106,22,106,22,255,31,255
,31,56,28,16,8
5300 POKE 1664,0:POKE 1665
,43:POKE 1666,43:POKE 1667
,255
5310 POKE 1668,50:POKE 166
9,0:POKE 1670,5:POKE 1672,
200:POKE 1673,235:POKE 167
4,240
5320 POKE 1676,201:POKE 16
77,30:POKE 1678,30:POKE 16
80,201:POKE 1681,30:POKE 1
682,30
5330 POKE 1684,128:POKE 16
85,0:POKE 1686,5:POKE 1688
,201:POKE 1689,30:POKE 169
0,30
5365 POKE 675,15
5440 POKE 704,15:POKE 705,
40:POKE 706,40
5450 POKE 53248,128:POKE 5
3249,0:POKE 53250,5

```

And the explanation:

1270-1280: All we've done here is take the old Line 1280 and break it into two parts so that we skip ahead to Line 1330 if we're not changing direction.

1290: If we haven't just hit the right-hand side, we also want to skip ahead.

1300-1310: If we have hit the right-hand side, then we want to get the saucer going. We tell PMOVE that it's off the right-hand side of the screen, set a flag (SAUCER) to indicate that it's on, and then set up the imaginary joystick so that it's moving left.

1370: If the saucer is on, and it's moved off the left-hand side of the screen, then we clear the flag and set the imaginary joystick back to the center.

3290-3300: This is the initialization section. These two lines set up the data for the saucer in the players. You've probably noticed by now that the saucer is made up of two players positioned side by side. This is a good way to get more dots per object if you have players to spare.

5300: We've changed this line so that Players 1 and 2 are attached to an imaginary joystick, as was described above.

5310-5330: Again, these lines have been changed so that PMOVE is prepared for the new players. If you're still unsure about what's going on here, check the description of PMOVE again, as well as the descriptions for these lines when we first added them to the program.

5365: This initializes the imaginary joystick so that it's centered (no movement).

5440: Give the saucer a color.

5450: And position it off the left-hand side of the screen (where it will stay until we're ready for it).

(Don't worry that the spaceship runs into the invaders; it won't when we're done.)

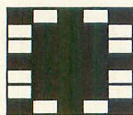
Vertical Player Movement

Before we go on, a few words about moving a player (or missile) vertically. There are no memory locations that are used to position a player vertically. Vertical movement is accomplished by moving the data within the player. For example, the data for our base starts at the byte #201 in Player 0. If we wanted to move the base up one, we would have to shift the data so that it began at byte #200. Similarly, to move the base down we would have to shift the data so that it began at byte #202.

PMOVE includes some routines that do this shifting around in machine language, and since there are times when you may need to get at these routines directly, I'm going to explain how you can use them to move a player or missile vertically, one dot at a time.

Basically, there are just two steps. First of all, PMOVE should be put in memory and turned on, as was described previously. Next, use one of the following commands, depending on which player or missile you want to move, and in which direction:

FIGURE 1



```
10 REM To move a player up
: X=USR(ADR("hhh$ 0/0"))
,n)
20 REM To move a player do
wn: X=USR(ADR("hhh$ 0/0"))
,n)
30 REM To move a missile u
p: X=USR(ADR("hhh$ 0/0"))
,n)
40 REM To move a missile d
own: X=USR(ADR("hhh$ 0/0"))
,n)
```

(n is the number of the player or missile to be moved.)

And that's all there is to it. Now back to PMOVE.

Firing a Missile

We're now going to give our base the ability to fire a missile, thereby demonstrating how PMOVE handles missiles. Add these lines to our program:

```
1120 IF STRIG(0)=1 OR PEEK
(1700)<>0 THEN 1270
1140 X=USR(ADR(MISCLR$),PA+
+768,255,252):POKE 1692,PE
EK(1684)+2:POKE 53252,PEEK
(1692):POKE 1696,199:POKE
PA+967,1
1150 POKE PA+968,1:POKE 17
00,1
3060 DIM MISCLR$(26):GOSUB
30000:MISCLR$=MLANG$
5340 POKE 1700,0
5350 POKE 1704,0:POKE 1708
,129
5430 POKE 53278,0
```

1120: If the joystick button isn't pressed or if a missile is already in the air, then skip the next two lines.

1140-1150: A missile is to be fired, so clear it out (more on this later), position it at the horizontal position of the base plus two (which puts it in the middle of the base), tell the computer this position, tell PMOVE the vertical position (at the top of the base), set up the missile data and tell PMOVE to start it moving.

3060: This sets up MISCLR, a machine-language routine used to clear the missiles. The reason that we can't use MEMCLR for this is because MEMCLR clears whole bytes,

and a missile only takes up part of a byte. We have to be able to clear this part without affecting the rest. MISCLR to the rescue. Here's how it's used:

X=USR(ADR(MISCLR\$),MISBASE,RES,MASK)

MISBASE is the address of the beginning of the missile area. It's equal to PMBASE+768 for single-line resolution, and PMBASE+384 for double-line resolution. RES tells what kind of resolution is being used and is equal to 255 for single-line and 127 for double. Finally, MASK is used to specify which missile you want to clear and has one of the following values:

Missile 0: 252
Missile 1: 243
Missile 2: 207
Missile 3: 63

See Line 1140 for an example of how this all fits together.

5340-5350: Here we initialize some of the PMOVE Locations. In order, we make sure Missile 0 is turned off, set its horizontal direction to zero (no horizontal direction) and set its vertical direction to 129 (up).

5430: We want to make sure the collision registers are clear at the start of the game.

30000: This is the data for MISCLR.

Keep in mind that because of the way we arranged our program, Lines 3060 through 5350 will be executed before Lines 1120 through 1150.

Giving the Missiles a Purpose

Okay, we've now got a base that moves from side to side and fires missiles. You've probably noticed that these missiles aren't really paying much attention to what's on the screen; they just pass through everything. Let's change that so a missile will stop when it hits something. PMOVE was designed to make this easy to do, so let's jump right into the program changes:

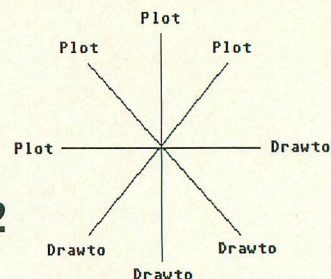


FIGURE 2

```
180 X=USR(ADR(MISCLR$),PA+
768,255,252):POKE 1720,0:R
ETURN
1090 IF PEEK(1720)<>0 THEN
GOSUB 180
1110 POKE 53278,0
1120 IF STRIG(0)=1 OR PEEK
(1700)<>0 OR PEEK(1720)<>0
THEN 1270
1150 POKE PA+968,1:POKE 53
278,0:POKE 1712,15:POKE 17
16,6:POKE 1720,0:POKE 1700
,1
5340 POKE 1700,0:POKE 1720
,0
```

180: This is actually a little subroutine that gets executed when a collision has occurred. All it does is clear the missile and clear the collision flag.

1090: This checks the collision flag to see if a collision has occurred. If it has, then we go off to the previous subroutine.

1110: Since we've taken care of any collisions at this point, we want to clear all the collision registers so that we won't detect the same collision twice.

1120: We've added one more condition to this line, in that we don't want to fire a missile until any collisions have been taken care of (i.e., the collision flag is clear).

1150: We've changed this line so that the collision registers are cleared, PMOVE is set up to look out for collisions between Missile 0 and any type of playfield and between Missile 0 and Players 1 and 2, and the collision flag is cleared; all before we turn the missile on.

5340: Finally, when we initialize PMOVE we make sure that the collision flag is cleared.

More on Collisions

It may seem here like the collision registers are being cleared too often. One of the biggest problems with using PMG from BASIC, even with PMOVE, is in collision detection. The problem comes from the fact that the collision registers keep track of all collisions that

have occurred since the last time they were cleared. This means that if you're not careful, you can detect the same collision more than once, something I ran into quite a bit while I was programming BASIC Invaders.

You should therefore try to make absolutely sure that the registers are cleared properly each time you process a collision and each time you fire a new missile. Also keep in mind that you should not only erase a missile that's collided before you clear the registers, but you should also try to do something between erasing the missile and clearing the registers. This is because it can take up to 1/60th of a second to get the missile off the screen, and during that time the collision might be detected again.

Explosions

Our player's missiles are now able to collide with the other objects on the screen, so the next logical step is to do something when they collide. In other words, we want to generate some explosions. There are three types of explosions that we have to deal with, since there are three types of objects that the missiles can collide with. In the order that we're going to approach them, the explosions are the alien saucer, the barriers and the invaders.

The Saucer

We're starting with the saucer because it's the easiest to deal with; all we're going to do is stop it and then fade it out. Here's the program additions:

```
180 X=USR(ADR(MISCLR$),PA+
768,255,252):POKE 1720,0:I
F PEEK(1716)<>255 THEN RET
URN
190 POKE 675,15
200 FOR X=4 TO 0 STEP -1:P
OKE 705,112+X*3:POKE 706,1
12+X*3:FOR L=1 TO 10:NEXT
L:NEXT X
210 POKE 1685,0:POKE 1686,
5
220 POKE 53249,0:POKE 5325
0,0:RETURN
1300 POKE 1685,235:POKE 16
86,240:POKE 705,40:POKE 70
6,40:SAUCER=1
```

The explanation:

180: You'll recall that this is the beginning of a collision routine. We've changed it so that it checks PMOVE to see whether a collision with one of the players occurred, and returns if not.

190: If a collision with one of the players has occurred, then that means the saucer was hit. This line moves the imaginary joystick back to center, thereby stopping the saucer.

200: This loop simply fades out the color of the saucer until it's the same color as the background.

210: We now tell PMOVE that the saucer is off the left-hand side of the screen.

220: This line tells the computer that the saucer is off the screen and then returns to the main program.

1300: Because we may have faded out the saucer, we change this line so that the saucer's color is restored before we turn it on.

If you'd prefer something a little more elaborate than this, you might try modifying the program so that the saucer appears to disintegrate. One way to do this would be to design several versions of the saucer, each a little more disintegrated than the other. Store the data for these versions in a string and then use MOVMEM to move them into the players, one after the other, in Line 200 above (instead of changing the colors). This will give the effect of the saucer disintegrating.

The Barriers

Our next explosion, the barriers, is a little more complicated. Basically, what we're going to do is erase a part of the barrier in the shape of an explosion. Figure 1 is the shape we'll be erasing.

The easiest way to erase part of a bit-mapped object is to PLOT and DRAWTO with the background color. It just so happens that the above explosion is made up of four lines, as shown in Figure 2.

We're now left with the question of where to draw the explosion. In other words, how

do we know where the missile hit the barriers? You should recall that PMOVE includes locations that keep track of the vertical and horizontal positions of the missiles. PMOVE stops the missile as soon as a collision is detected, so all we have to do is look at the vertical and horizontal positions in PMOVE and translate them into X and Y values for the barriers. Enough talk though, let's see all this in action:

```
180 X=USR(ADR(MISCLR$),PA+
768,255,252):POKE 1720,0:I
F PEEK(1716)<>255 THEN 230
230 TMP=PEEK(53248):IF TMP
<4 OR TMP=8 THEN RETURN
250 X=PEEK(1692)-47+2*(RND
(0)>0.5)-1:Y=PEEK(1696)-15
8
260 POKE 89,INT(MEM7/256):
POKE 88,MEM7-256*PEEK(89):
POKE 87,7:COLOR 0
270 YT=Y-3:YT=YT*(YT)=0}
280 PLOT X-2,YT:DRAWTO X+2
,YT+6:PLOT X,YT:DRAWTO X,Y
+6:PLOT X+2,YT:DRAWTO X-2
,YT+6
290 PLOT X-2,Y*(Y)=0}:DRAW
TO X+2,Y*(Y)=0}
300 RETURN
```

Here's what's going on:

180: We've changed this again so that we skip over the saucer routine if we haven't hit it.

230: Here we check MOPF to see whether or not we've collided with the barriers (Playfield 2). If not, we return to the main program.

250: Now we figure out the X and Y values. It turns out that the X value is equal to the horizontal missile position minus 47, and the Y value is equal to the vertical missile position -158. How do we know this? Actually, I figured it out on a trial-and-error basis, by changing the values until things clicked.

You may also be wondering where the $2*(RND(0) > 0.5) - 1$ comes from. Before we look at where it comes from, let's look at what it does. The formula gives us -1 half the time and +1 the other half. This has the effect of shifting the explosion left one or right one. Why do we do this? Try taking it

out of the line and see what happens. If we don't randomly shift the explosion, then it tends to be a little too "neat" in the way it destroys the barriers.

260: You'll remember this line from our bit-mapping discussion. It gets the computer ready to make changes to the graphics mode 7.5 section of the screen. We also use the COLOR 0 command, which means that we'll be drawing in the background color.

270: The X and Y positions that we just figured out are for the middle of the explosion. We need to know where the top of the explosion is (since that's where we'll start most of our lines), so we set up YT. We also make sure that we don't try and start the explosion too far up (in case Y was at the top of the barrier).

280-290: Finally, we draw the explosion.

300: And then get back to the main program.

So far, so good. The last explosion that we have to concern ourselves with, for now at least, is the invaders. I've left it for last because it's the trickiest, and also because it will present a problem that we'll solve later. Right now, though, let's take a look at what we have to do for the explosion.

You should recall that we included an explosion shape in our redefined character set. What we're going to do is replace the hit invader with this explosion shape and then with an empty space. This presents a few

problems, however. First of all, we have to be able to figure out where the hit invader is in INV\$. But this won't be that much different than figuring out where the barriers were hit. The main problem (a small one at that) comes from the way we're printing the invaders on the screen.

Suppose that we've located the invader in INV\$ and replaced it with the explosion. Do we now wait a second and then replace it with a space? No, because replacing it in INV\$ doesn't replace it on the screen until INV\$ is printed again. What we have to do is replace the invader with the explosion, wait until INV\$ has been printed on the screen again, and then replace the explosion with the space. This means that we need to maintain a variable that keeps track of whether or not we're in the middle of an explosion so that we know whether or not an explosion needs to be changed into a space. Keeping all this in mind, here are the program changes necessary:

```
230 TMP=PEEK(53248):IF TMP
<4 OR TMP=8 THEN 310
310 IF TMP=8 THEN 250
320 R=48*INT((PEEK(1696)-3
0)/16):C=2*INT((PEEK(1692)
-5*ROLL-COARSE*8-47)/16):R
=R*(R)=0)
330 INV$(R+C+3,R+C+4)="■"
:INV$(R+C+291,R+C+292)="■"
":EF=2
390 RETURN
1020 IF EF=0 THEN 1080
1030 INV$(R+C+3,R+C+4)="■"
":INV$(R+C+291,R+C+292)="■"
"
1120 IF STRIG(0)=1 OR PEEK
(1700)<>0 OR PEEK(1720)<>0
THEN 1250
1250 EF=EF-(EF<>0)
5270 SCROLL=0:CHANGE=1:5B=
0:EF=0:COARSE=0
```

230: We've just changed this so that if a barrier wasn't hit, it skips ahead to Line 310 instead of returning like it did before.

310: Because of the problems with dealing with the collision registers from BASIC, sometimes we'll get to this line with the collision register reading zero. For some reason, this seems to only happen when the barriers have been hit, so this line skips back to the barrier explosion if we encounter this situation.

Incidentally, this is not a problem that you should know about. I programmed this section without Line 319 and found that invaders would occasionally explode after I had hit a barrier. By looking at the value of TMP after this happened, I was able to see that somehow it was always equal to zero and thereby come up with this solution to the problem.

To this day I have no idea of why this should happen. The whole purpose of telling you this is to make you aware that there is more than one way to deal with a bug in your program. You can either get rid of it or go around it, and there's nothing wrong with going around it if you can't figure out how to destroy it.

320: Now that we've made sure the missile has collided with an invader, we figure out which one. This line is essentially the equivalent of Line 250 in the barrier explosion routine except that we're figuring out R (row) and C (column) instead of X and Y. There are, of course, a few differences. For example, after we subtract the necessary amount (again determined from trial and error) from the horizontal and vertical positions, we have to divide by 16 to account for the fact that each invader is 16 positions wide and 16 high (including the space between the invader and the one above it). We then multiply the resulting row by 48 so that R is the starting position in INV\$ for the invader's row. We multiply C by two, since each invader is made up of two characters. Oh, I almost forgot, when figuring out the value of C, we also adjust for how far the invaders have been scrolled from their original position.

330: Now that we know where the invader is, we replace it with the explosion shape (in case you were wondering, the funny characters in this line are inverse CTRL-M and inverse CTRL-N) and initialize our explosion flag (EF).

390: Back to the main program.

1020: This line comes after we've printed the invaders, and it checks the explosion flag to see whether we're in the middle of an explosion or not. If we aren't, then it skips over the next line.

1030: If we are in the middle of an explosion, then we replace the explosion shape with blank space.

1250: This just updates the explosion flag, if necessary.

5270: Our last change, we've simply added the explosion flag to this initialization line.

With all our explosions now in place, our game is actually starting to look and act like a game. But it's far from finished. What's left

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 #103 ON READER SERVICE CARD.

is largely something called "game logic," so next month we'll see exactly what this logic stuff is.

LISTING 1: BASIC

```
JF 90 GOTO 3010
JO 1000 X=USR(ADR(MOVMEM$),ADR(INV$)+5B,M
    EMI,263)
ZH 1080 SB=288*(5B=0)
TJ 1280 IF COARSE=4 OR COARSE=-2 THEN CHA
    NGE=-CHANGE:POKE 1791,129-PEEK(1791)
TJ 1330 SCROLL=SCROLL+CHANGE
CK 1340 IF SCROLL>15 THEN SCROLL=SCROLL-1
    6:COARSE=COARSE+2:POKE 1790,2:GOTO 136
    0
JC 1350 IF SCROLL<0 THEN SCROLL=SCROLL+16
    :COARSE=COARSE-2:POKE 1790,2
LG 1360 POKE 1788,SCROLL:POKE 1787,1
DP 1380 IF PEEK(1790)<>0 THEN 1380
NU 1390 GOTO 1000
QQ 3010 POKE 559,0
UQ 3030 DIM MLANG$(90),INV$(576),DAT$(16)
PG 3040 DIM VBL0FF$(20):GOSUB 29000:VBL0F
    F$=MLANG$
GE 3050 DIM MOVMEM$(41):GOSUB 29500:MOVME
    M$=MLANG$
VT 3100 FOR BYTE=0 TO 10:READ DAT:POKE 15
    36+BYTE,DAT:NEXT BYTE
KM 3110 DATA 72,169,212,141,10,212,141,26
    ,208,104,64
ZB 3150 CB=PEEK(740)-4:POKE 106,CB-4:CA=C
    B*256
```

```
WX 3160 FOR SEC=0 TO 1:GOSUB 31000+10*5EC
VV 3170 X=USR(ADR(MOVMEM$),ADR(MLANG$),CA
    -256+90*5EC,LEN(MLANG$)-1):NEXT SEC
RQ 3210 X=USR(ADR(MOVMEM$),57344,CA,1023)
NE 3220 MEM=CA+512:FOR SEC=0 TO 1:GOSUB 3
    2500+10*5EC:X=USR(ADR(MOVMEM$),ADR(MLA
    NG$),MEM,LEN(MLANG$)-1)
YM 3230 MEM=MEM+LEN(MLANG$):NEXT SEC
NC 3240 X=USR(ADR(MOVMEM$),CA+128,CA+640,
    335)
LE 4000 GRAPHICS 22:POKE 559,0:POKE 756,C
    B+2
PL 5010 DLIST=PEEK(560)+PEEK(561)*256
WH 5020 POKE DLIST+3,86
MJ 5030 L=PEEK(DLIST+4)+44:POKE DLIST+5,P
    EEK(DLIST+5)+(L*255):POKE DLIST+4,L-25
    6*(L*255)
TB 5040 FOR X=6 TO 20:POKE DLIST+X,22:NEX
    T X:FOR X=24 TO 50:POKE DLIST+X,14:NEX
    T X
EF 5050 MEM7=PEEK(88)+PEEK(89)*256+600
UJ 5060 POKE DLIST+21,78:POKE DLIST+23,IN
    T(MEM7/256):POKE DLIST+22,MEM7-INT(MEM
    7/256)*256
CR 5070 POKE DLIST+31,78:POKE DLIST+33,IN
    T(MEM7+320)/256:POKE DLIST+32,MEM7+3
    20-PEEK(DLIST+33)*256
IR 5080 POKE DLIST+41,78:POKE DLIST+43,IN
    T(MEM7+640)/256:POKE DLIST+42,MEM7+6
    40-PEEK(DLIST+43)*256
QL 5090 POKE DLIST+51,22:POKE DLIST+52,22
    513,6:POKE 54286,192
RW 5100 POKE DLIST+53,150:POKE 512,0:POKE
    513,6:POKE 54286,192
QB 5110 POKE DLIST+54,6:POKE DLIST+55,70:
    POKE DLIST+56,PEEK(88):POKE DLIST+57,P
    EEK(89)
FV 5120 POKE DLIST+58,65:POKE DLIST+59,PE
    EK(560):POKE DLIST+60,PEEK(561)
MH 5180 MEM1=PEEK(DLIST+4)+PEEK(DLIST+5)*
    256:SCRL=PEEK(DLIST+56):SCRH=PEEK(DLIST
    +57)
```

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), Advanced Protection Techniques (Book and Disk II), The Chipmunk, The Scannerizer, 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) • Overfilled 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**

Scannerizer 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:

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. Includes \$3.00 ship & hndg (US Canada) Ohio res. add 5 1/2% sales tax. Foreign orders add \$8.00 ship & hndg. Call or write for free catalog. Customer Service Line (216) 467-5665 M-F 9-3.

ATARI 8-BIT POWER

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

NEW PARROT II

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 dog's 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 unequalled 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**



GIANT WALL SIZED POSTERS.

CIRCLE #104 ON READER SERVICE CARD.


```
CU 1020 ? CHR$(34);":RETURN":? "CONT":POS
ITION 0,0:POKE 842,13:STOP
UF 1030 POKE 842,12:RETURN
UG 20000 DATA 104,162,228,160,95,169,6,32
,92,228,162,228,160,98,169,7,32,92,228
,96,2548
QY 21000 DATA 104,104,133,207,104,133,206
,104,133,209,104,133,208,104,170,160,2
55,138,208,2,104,168,177,206,145,3719
EW 21010 DATA 208,136,192,255,208,247,230
,207,230,209,202,224,255,208,233,96,33
40
TH 22000 DATA 104,104,133,207,104,133,206
,104,104,168,104,104,133,208,177,206,5
7,208,145,206,136,192,255,208,245,3931
YN 22010 DATA 96,96
JJ 23000 DATA 104,104,133,204,104,133,203
,104,170,169,0,160,255,224,0,208,4,104
,168,169,0,145,203,136,192,3396
PM 23010 DATA 255,208,249,230,204,202,224
,255,208,234,96,2365
FT 24000 DATA 173,251,6,240,104,173,252,6
,141,4,212,173,253,6,141,5,212,173,254
,6,240,79,173,48,2,3327
JY 24010 DATA 133,204,173,49,2,133,205,16
0,3,177,204,201,65,240,61,201,1,240,52
,41,112,201,64,144,48,3114
IX 24020 DATA 201,80,144,42,200,173,255,6
,48,18,177,204,24,216,109,254,6,145,20
4,200,177,204,105,0,145,3337
SL 24030 DATA 204,144,20,177,204,56,216,2
37,254,6,145,204,200,177,204,233,0,145
,204,144,2,200,200,200,208,3984
MY 24040 DATA 189,169,0,141,254,6,141,251
,6,76,95,228,1556
IE 25000 DATA 104,104,170,104,168,169,6,3
2,92,228,96,1273
HM 26000 DATA 104,104,104,141,188,6,104,1
04,141,228,6,141,231,6,141,234,6,141,2
37,6,238,237,6,141,240,3235
WO 26010 DATA 6,238,240,6,169,127,141,199
,6,162,9,160,4,173,47,2,41,16,240,9,16
9,255,141,199,6,2765
FA 26020 DATA 162,19,160,8,140,200,6,160,
9,189,206,6,153,189,6,202,136,16,246,1
69,7,174,240,6,160,2969
OH 26030 DATA 108,32,92,228,96,32,238,6,1
89,152,6,24,109,200,6,168,205,199,6,14
4,3,172,199,6,189,2809
BK 26040 DATA 152,6,56,237,200,6,141,201,
6,136,177,204,200,145,204,136,240,5,20
4,201,6,176,242,169,0,3450
BE 26050 DATA 145,204,96,32,238,6,189,152
,6,56,237,200,6,168,176,2,160,0,189,15
2,6,24,109,200,6,2759
MY 26060 DATA 141,201,6,200,177,204,136,1
45,204,200,204,199,6,240,7,204,201,6,1
44,239,240,237,169,0,145,3855
TM 26070 DATA 204,96,138,72,162,4,32,238,
6,104,170,189,160,6,56,237,200,6,168,1
76,2,160,0,189,160,2935
HO 26080 DATA 6,24,109,200,6,141,201,6,13
6,177,204,61,202,6,145,204,200,200,189
,202,6,73,255,49,204,3206
OF 26090 DATA 136,136,17,204,145,204,200,
200,204,199,6,176,7,204,201,6,144,221,
240,219,189,202,6,49,204,3719
UU 26100 DATA 145,204,136,189,202,6,49,20
4,145,204,96,138,72,162,4,32,238,6,104
,170,189,160,6,24,109,2994
IH 26110 DATA 200,6,168,205,199,6,144,3,1
72,199,6,189,160,6,56,237,200,6,141,20
1,6,200,177,204,61,3152
BK 26120 DATA 202,6,145,204,136,136,189,2
02,6,73,255,49,204,200,200,17,204,145,
204,136,136,240,5,204,201,3699
CO 26130 DATA 6,176,224,189,202,6,49,204,
145,204,200,189,202,6,49,204,145,204,9
6,189,189,6,133,204,24,3445
```

LISTING 2: BASIC

```

VN 100 GRAPHICS 0:?"Make sure you have s
aved a copy of it"? "this program before
  RUNNING it":FOR X=1 TO 1050:NEXT X
SQ 110 ? :?
RO 120 DIM LN(8):FOR X=1 TO 8:READ DAT:LN
(X)=DAT:NEXT X
PE 130 DATA 20,41,26,36,112,11,657,128
OJ 140 FOR X=1 TO 8:TOT=0:N=0:GOSUB 1000
NH 150 FOR N=1 TO LN(X):READ DAT:TOT=TOT+
DAT
HP 160 IF N/25<INT(N/25) THEN 190
QP 170 T=TOT:TOT=0:READ DAT:IF DAT<>T THE
N ? "...ERROR":STOP
QY 180 GOSUB 1000
JW 190 NEXT N:READ DAT:IF DAT<>TOT THEN ?
"...ERROR":STOP
LM 200 NEXT X
AJ 210 RESTORE 20000
OV 220 FOR X=1 TO 8:L=28500+500*X:GOSUB 1
010
BP 230 FOR N=1 TO LN(X):READ DAT:?" CHR$(2
7):CHR$(DAT);
TJ 240 IF N/25=INT(N/25) THEN READ DAT
WF 250 IF N/90=INT(N/90) THEN GOSUB 1020:
L=L+10:GOSUB 1010
RO 260 NEXT N:READ DAT:GOSUB 1020
MA 270 NEXT X
OH 280 END
LM 1000 ? :? "CHECKING LINE ";19000+1000*
X+10*INT(N/25):RETURN
DJ 1010 GRAPHICS 0:POSITION 2,4:?" PLA
NGS=":CHR$(34):RETURN

```



```

      8,106,185,120,2,41,2707
M5 26150 DATA 8,208,23,189,148,6,221,136,
    6,240,43,169,0,133,77,254,148,6,189,14
    8,6,157,0,208,208,2931
WH 26160 DATA 28,185,120,2,41,4,208,21,16
    9,0,133,77,189,148,6,221,132,6,240,9,2
    22,148,6,189,148,2652
WY 26170 DATA 6,157,0,208,188,128,6,185,1
    20,2,41,2,208,17,189,152,6,221,144,6,2
    40,30,254,152,6,2668
UX 26180 DATA 32,229,6,138,16,21,185,120,
    2,41,1,208,14,189,152,6,221,140,6,240,
    6,222,152,6,32,2385
EX 26190 DATA 226,6,232,224,4,208,140,162
    ,0,189,164,6,240,83,189,168,6,240,50,1
    6,23,222,156,6,222,3182
NF 26200 DATA 156,6,189,156,6,157,4,208,2
    01,47,176,32,169,0,157,164,6,240,53,25
    4,156,6,254,156,6,2959
EL 26210 DATA 189,156,6,157,4,208,201,208
    ,144,9,169,0,157,164,6,240,106,208,196
    ,189,172,6,240,57,16,3208
XO 26220 DATA 23,222,160,6,222,160,6,32,2
    32,6,189,160,6,201,16,176,39,169,0,157
    ,164,6,240,74,254,2920

```

```

AI 26230 DATA 160,6,254,160,6,32,235,6,18
    9,160,6,24,216,105,16,205,199,6,176,4,
    41,240,208,7,169,2830
AX 26240 DATA 0,157,164,6,240,42,189,176,
    6,61,0,208,240,13,169,255,157,176,6,15
    7,184,6,169,0,157,2938
NV 26250 DATA 164,6,189,180,6,61,8,208,24
    0,13,169,255,157,180,6,157,184,6,169,0
    ,157,164,6,232,224,3141
KA 26260 DATA 4,208,145,76,98,228,0,759
NF 27000 DATA 0,0,0,0,0,0,0,1,3,7,13,15
    ,2,5,10,128,192,224,176,240,64,160,80,
    1,1321
MD 27010 DATA 3,7,13,15,5,8,4,128,192,224
    ,176,240,160,16,32,8,4,15,29,31,23,20,
    2,16,32,1403
PT 27020 DATA 240,184,248,232,40,64,2,20,
    23,29,31,15,4,8,64,40,232,184,248,240,
    32,16,3,15,31,2245
CW 27030 DATA 25,31,6,9,48,192,240,248,15
    2,248,96,144,12,3,15,31,25,31,13,24,12
    ,192,240,248,152,2437
XA 27040 DATA 248,176,24,48,0,9,5,0,12,0,
    5,9,0,32,64,0,96,0,64,32,16,16,56,56,1
    24,1892
UU 27050 DATA 124,198,198,520

```



FOR OUR DISK SUBSCRIBERS

The following programs from this issue are on disk:

THE A.N.A.L.O.G. #70 DISKETTE CONTAINS 23 MAGAZINE FILES. THEY ARE LISTED BELOW.

SIDE 1:

FILENAME.EXT	LANG.	LOAD	COMMENTS
DUPBASIC.BAS	BASIC	LOAD	DUPING BASIC, L1
DUPBASIC.M65	MAC/65	LOAD	DUPING BASIC, L2
ELECTRA.BAS	BASIC	LOAD	ELECTRA-BALL
VIDEODSC.BAS	BASIC	LOAD	VIDEODISC SYSTEM
PEBBLES.BAS	BASIC	LOAD	PEBBLES
UNSPRIT1.BAS	BASIC	LOAD	UN-SPRITES, L1
UNSPRIT2.BAS	BASIC	LOAD	UN-SPRITES, L2
UNSPRIT3.BAS	BASIC	LOAD	UN-SPRITES, L3
UNSPRIT4.BAS	BASIC	LOAD	UN-SPRITES, L4
UNSPRIT5.M65	MAC/65	LOAD	UN-SPRITES, L5
GDW1.BAS	BASIC	LOAD	GAME DESIGN, L1
GDW2.BAS	BASIC	LOAD	GAME DESIGN, L2
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
GDW11.LST	BASIC	ENTER	GAME DESIGN
MLEDITOR.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.

DISK GAMES FOR THE ATARI XEGS

The Atari XEGS isn't just another pretty game system; it's a computer too. With a disk drive connected, the XEGS can run just about anything its big brother, the 130XE, can. Here are some of the best disk games currently available for the Atari computers, which are compatible with the XEGS.

"All you have to do is shoot the disrupters, take the crystals and run." It sounds simple enough, but *Solar Star* is a seriously difficult game to play.

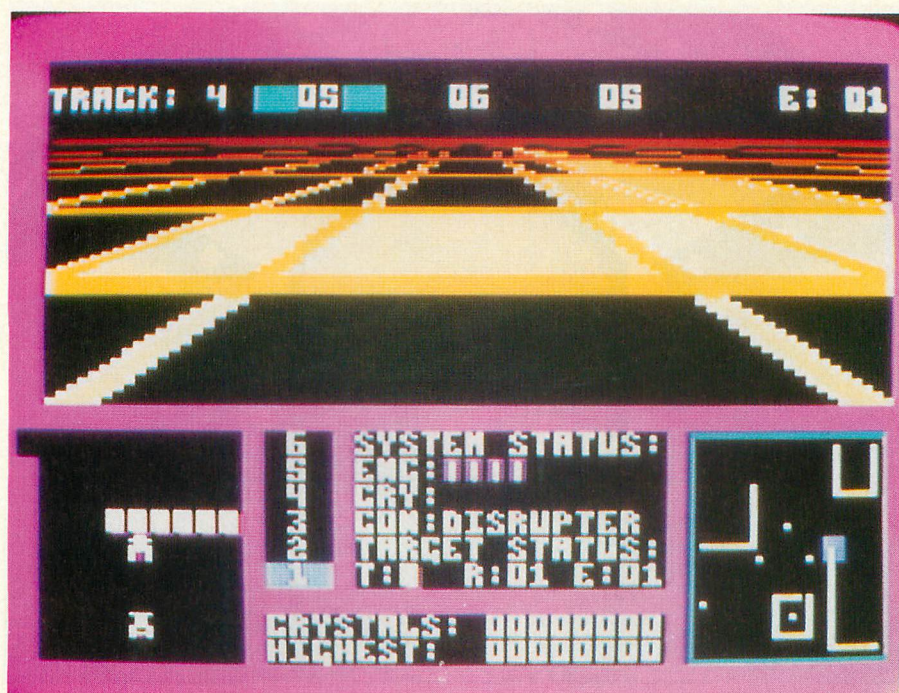
The first thing you will notice about Microdaft's games is the title screen. It's impressive—unless you have played Lucasfilm games, such as *Ballblazer*, before. The title screen for Microdaft's games is a virtual clone of the Lucasfilm logo.

It is the 21st century, and the X Xarion Corporation has built solar energy grids around the sun so they can use the sun's energy to develop crystals for spaceship fuels—synthetic dylithium crystals, like those found in the starship Enterprise, possibly.

by Matthew J.W. Ratcliff

Solar Star

Microdaft
19 Harbor Drive
Lake Hopatcong, NJ 07849
(201) 663-0202
\$19.95



Solar Star

The computer in the solar energy grid has gone haywire, similar to HAL in the movie *2001, A Space Odyssey*. The computer designed to protect the energy grid from fuel pirates has taken complete control, and will not allow friendlies to access the grid either.

At the start of the game your "hyper drive" engines transport your *Solar Star* craft to the first of 16 energy grids. The energy grid looks like a sheet of graph paper tilting away from you, in a 3-D effect. You must drive your spaceship along the open areas of the grid (black squares)

in search of the disrupters (flashing white squares). When you shoot a disrupter, it releases an energy crystal (it turns green), at which time you can run over it and collect the crystal. However, when a disrupter is zapped, the resulting crystal seems to jump diagonally a grid or two, which adds a great deal of difficulty to the game. Your goal is to collect as many crystals as possible.

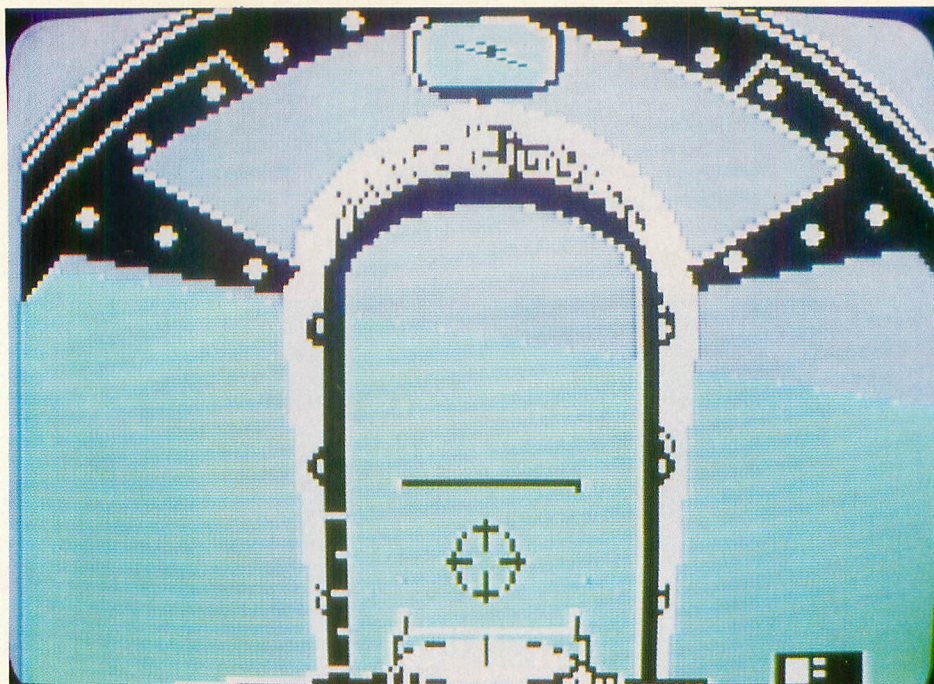
The top half of the screen is the play grid. The bottom left of the display is your "short range" scan, which shows about nine squares of the grid, while the bottom

right displays your long range scan of the entire grid. On the grid you will encounter blue and light-green walls which make up a maze of sorts. If you zap a disrupter, you may find your crystals escaping through walls that you must go around. The bottom center of the display shows the game status, items such as energy, score, crystals collected and so on.

Each grid has a different maze, with a "warp cell" protected by a revolving wall. Once you have collected ten crystals on one grid, you must shoot the warp cell to move on to the next grid.



Dropzone



Spitfire 40

Solar Star is a game which requires lightning-fast reflexes to master—if it can be mastered at all. The graphics are fair, the sound effects are good, the speed of this game is incredible, but at the expense of playability. I think the earlier grids should have started out with a much slower play speed, making the game easier to master in increments. If you demand a serious challenge in your video games, *Solar Star* provides it. You won't find much of a plot here, however; it's kill or be killed—until you are dead.

Dropzone

Microdaft
19 Harbor Drive

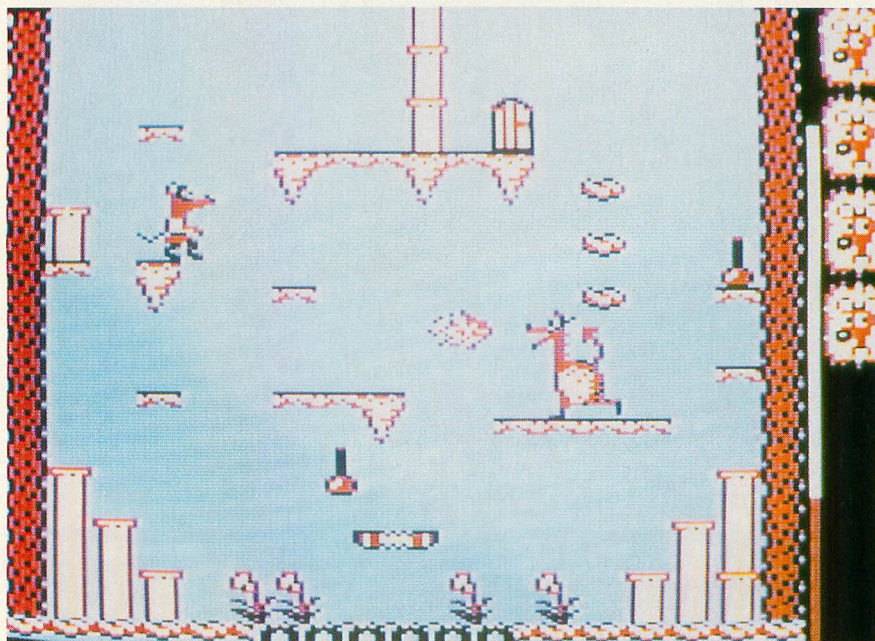
Lake Hopatcong, NJ 07849
(201) 663-0202
\$19.95

Dropzone is another incredibly fast video game from Microdaft, and it too suffers poor playability because of this "feature." It also lacks originality, cloning the popular arcade game *Defender*. Atari holds the license for the official version of *Defender*, which is currently available in cartridge form, and although *Dropzone* has better graphics and sound effects than Atari's version, it is also less playable.

Your man is in a spacesuit, with a jet powered back-pack, and he always remains in the center of the screen as the scenery

scrolls left and right, under your controls. He can fly up and down, and shoot left or right. To help in the battle, the space bar fires a bomb that nukes all the nasties currently on the screen. Your long range scanner is at the bottom of the display. Pressing the escape key toggles between pause and play modes, while any other key will toggle the "cloaking device." When cloaked, your player turns gray, and the aliens can't see him. This only lasts for a short period of time, but can help when the screen is full of meanies, and you are out of bombs.

Unlike *Defender*, you must pick up *all* the people stranded on the planet's surface and personally deliver them to the Dropzone, before proceeding to the next level.



Mousetrap

*Dropzone is
incredibly
challenging,
obviously intended
for master gamers
who are immune to
joystick wrist
tendonitis.*

The planet is drawn in fine detail, which occupies valuable screen space. This makes the flying battle area smaller than in *Defender*. The less space you have to work in, the more often you get clobbered by the baddies. Also, your hero is always forced to stay in the center of the display, giving you much less time to line up your shots. In *Defender*, when moving left to right, your spaceship resides in the left quarter of the display. This allows you three-fourths of the display to line up shots against the bad guys. The same holds true when flying in the opposite direction.

Someone needs to tell Microdaft that the XEGS is not a coin-operated video-game machine. The games do not have to be designed with high frustration factors and fast game play from the outset, just to gobble up more quarters from the neophyte arcade addicts. *Dropzone* is incredibly challenging, obviously intended for master gamers who are immune to joystick wrist tendonitis. It will frustrate a novice gamer.

Spitfire 40

Avalon Hill Game Company
4517 Harford Road
Baltimore, MD 21214
(301) 254-5300
\$35.00

Avalon Hill Game Company, well-known for its war-game simulations, delivers another excellent game in *Spitfire 40*. You take the controls of a World War II Spitfire, flying against German Me-109 fighters or STUKA dive bombers.

Your view is out the front canopy, with a rearview mirror at the top center. The horizon is depicted by a blue sky over green earth. Landmarks are in blue.

Spitfire allows you to select either full simulation or a game mode. In the game mode you are placed in the middle of a dogfight at 10,000 feet. You have three lives to practice with. In the simulation mode, you are presented with your orders, must take off, intercept, shoot down the enemy, and return safely in one flight to win.

Spitfire increases in difficulty along with your experience. At the beginning of play, you may start a new simulation or select a previous pilot's log (including your own). At the end of each mission, you log your flights to a separate formatted disk, with the "flight hours" stored in real time. For

example, if your simulation lasted 20 minutes, that is how much time is recorded to disk. As you accumulate flight hours, all controls become more realistically difficult. Your opponents get tougher and smarter. This is a superb feature. Furthermore, if you send a copy of your pilot's log disk with 60 flight hours or more to Avalon Hill, along with a coupon from the program package, you will receive a "certificate of merit for outstanding service."

The space bar toggles between the canopy view and the cockpit instrument display. Unfortunately, the XEGS just does not have the resolution to display both the canopy and cockpit views on a single screen. You'll have to practice jumping between the displays. There are tiny indicators on the canopy, showing throttle setting, altitude and direction. As you acquire more experience with *Spitfire*, you will be able to make good approximations of actual gauge settings just by glancing at these indicators.

Some controls, such as brakes, gears, flaps and others, are managed from the keyboard, while the basic controls necessary for a dogfight are handled from the joystick. The manual provides plenty of instructions and some interesting historical information. *Spitfire 40* is just one of the many excellent war-game simulations currently available from Avalon Hill. They are graphic adventures that also provide useful historical lessons.

Mousetrap

Micro Value
TDC Distributors, INC.
3331 Bartlett Blvd.
Orlando, FL 32811
(305) 423-1987
\$19.95

Marvin the mouse is in love with Meryl, but she thinks he will never amount to anything but "an artful cheese scavenger." Well, Marvin has decided to take matters into his own paws, and, with the assistance of you and your trusty joystick, win her love by showing that scavenging is an honorable profession.

You must help Marvin run, hop and jump from screen to screen, past ever more challenging dragons, ants and other nasty obstacles to amass his fortune of cheese and win the heart of Meryl. The game play is

continued on page 80

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 Unichck (see Issue 39).
- (2) Save the program to disk with the command `SAVE "D:EDITORII.BAS"`.
- (3) Clear the computer's memory with the command `NEW`.
- (4) Type in Listing 2, then verify your work with Unichck.
- (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 "EDITORII.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 Unichck.
- (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 Unichck.
- (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),C$(2),B$(1
15),M$(119),S$(198),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
$="Y" OR A$="U" THEN ABR=1
32606 B$(1)="":B$(115)="":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$:GOTO 32624
32622 ? L$(1,38):? L$(39,76):? L$(77,L
EN(L$))
32624 POKE 752,0:POKE 766,0:POKE 559,3
4:POKE 82,1:POKE 83,38:POSITION 0,10:
? "":INPUT M17,L$:POKE 766,1
32626 IF L$="P" OR L$="P" AND START=
0 THEN P=1:L$=""
32628 IF L$="E" OR L$="E" THEN E=1:PO
SITION 1,10:POKE 752,0:POKE 559,3
32630 IF L$="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$="B" THEN GRAPHIC
S 0: ? "TYPE 'GOTO 32600' TO CONTINUE":
END
32638 IF L$(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$(1)=L$:TRAP 32624:M=VAL(L$)
32642 START=1:IF P AND NOT E THEN 326
52
32644 GOSUB 32674:IF NOT ABR OR P THE
N 32652
32646 POKE 766,0:POKE 842,13:STOP
3,1:VAL(L$):LIST L$: ? "CONT":L$=
B$
32648 POSITION 0,0:POKE 842,13:STOP
32650 POKE 842,12:A=USR(ADR(M$),ADR(L$
),4):L$=L$(1,A)
32652 CHKSUM=USR(ADR(M$),ADR(L$),LEN(L
$)):CHKSUM=CHKSUM+PEEK(1542)*65536
32654 CHK=CHKSUM-(INT(CHKSUM/676)*676)
:HI=INT(CHK/26):LO=CHK-(HI*26):C$(1)=
CHR$(HI*65):C$(2)=CHR$(LO*65)
32656 IF NOT P OR THEN E=0:GOSUB 32
662:IF NOT P THEN 32668
32658 POKE 83,39:POKE 752,1:FOR M=3 TO
5:POSITION 1,M:POKE 83,39:POSITION 1,
M+7:POKE 752,1:POKE 83,39:POSITION 6
7:POKE 83,39:POSITION 67,1:POKE 752,1
32662 GOSUB 32702:POKE 766,0:POKE 752,
1: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+24,55:POKE DL+25,PEEK(560):POKE
DL+26,PEEK(561):POKE 83,39
32668 POSITION 20,0: ? "basic editor
1: ? "POSITION 0,7: ? " TYP
ING WINDOW " "
32670 POSITION 0,1: ? " MESS
AGE WINDOW " "POSITION 1,7
: ? "MODE":
32672 POKE 559,34:RETURN
32674 GRAPHICS 0:POKE 559,0:POKE 766,1
:POKE 82,0:POKE 83,39:POSITION 0,3: ? L
$: ? " ? " ? "CONT":POSITION 0,0
32676 POKE 842,13:STOP
32678 POKE 842,12:TRAP 32682:A=USR(ADR
(E$),VAL(L$)):IF A=4 THEN POP:GOTO 32
682
32680 RETURN
32682 GOSUB 32662:SOUND 0,75,10,8:FOR
M=1 TO 20:NEXT M:SOUND 0,0,0:POSITIO
N 1,3: ? "SYNTAX ERROR!":POKE 766,1
32684 POKE 83,38:POSITION 1,10: ? S$:G
OTO 32624
32686 LINE=PEEK(STMTAB)+PEEK(STMTAB+1)
*256:IF LINE32599 THEN 32698
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:POKE 83,38:GOTO 32624
32692 GRAPHICS 0: ? "FOR N=32600
TO 32636 STEP 2: ? N:NEXT N: ? "CONT":PO
SITION 0,0:POKE 842,13:STOP
32694 POKE 842,12:GRAPHICS 0: ? " ? " ?
: ? "CONT":POSITION 0,0
32696 POKE 842,13:STOP
32698 POKE 842,12:GRAPHICS 0: ? " ? " ?
: ? "CONT":POSITION 0,0
32699 POKE 842,12:GRAPHICS 0: ? " ? " ?
: ? "CONT":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 6,665,923,757,889,171,225,8
98,532,499,910,267,912,144,735,8453
32638 DATA 97,358,230,693,706,878,317,
127,36,597,238,258,182,438,168,5315
32668 DATA 864,953,472,365,687,724,72,
687,98,736,625,612,672,184,891,9672
32698 DATA 8,856,85,945
```

Listing 2. BASIC listing.

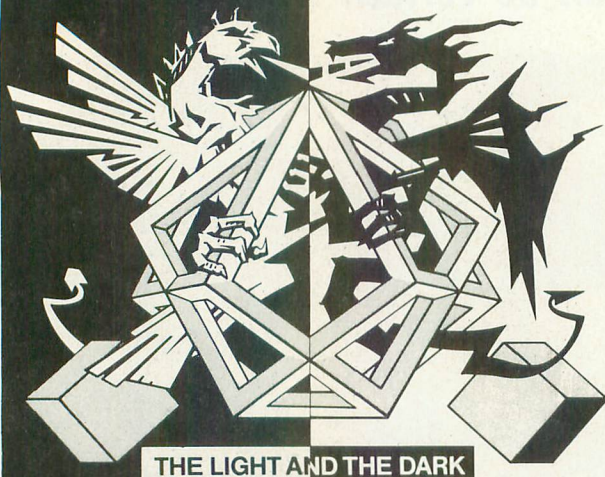
```
10 DIM L$(120),M$(119),A$(1)
20 GRAPHICS 0:POKE 710,0: ? "DISK OR CA
SSETTE":INPUT A$:IF A$<"C" AND A$<"D"
THEN 20
30 IF A$="C" THEN 50
40 ? "PLACE FORMATTED DISK IN DRIVE":
? "THEN PRESS RETURN":INPUT L$:OPEN M1,
8,0,"D:ML.DA":GOTO 60
50 ? "READY CASSETTE, PRESS RETURN"
:INPUT L$:OPEN M1,8,0,"C":
60 L$="32608 M$(1)=":L$(13)=CHR$(34)
70 M=119:GOSUB 130:L$(14)=M$(1,58):L$
=LEN(L$)+1:CHR$(34): ? M1:L$
80 L$(11)="32618 M$(59)=":L$(14)=CHR$(3
4):L$(15)=M$(59):L$(LEN(L$)+1)=CHR$(3
4): ? M1:L$
90 L$(11)="32612 S$=":L$(10)=CHR$(34)
100 M$=""N=8:GOSUB 130:L$(11)=M$(1,5)
:LEN(L$)+1:CHR$(34): ? M1:L$
110 L$(11)="32614 S$=":L$(10)=CHR$(34)
120 M$=""N=8:GOSUB 130:L$(11)=M$(1,5)
:LEN(L$)+1:CHR$(34): ? M1:L$END
130 FOR M=1 TO N:READ A:M$(M)=CHR$(A)
:NEXT M: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 141,6,6,238,3,6,32,68,218,172
,2,6,177,203,133,212,32,170,217,32,162
,221,32,68,218
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
6,109,5,6,172,5,6,14,3,238,6,6,238,2
180 DATA 173,172,2,6,196,205,208,176,173
,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
,201,192,144,6,201,224,144,7,176,8,24,1
05,32,144,3,56,233
220 DATA 64,145,203,208,192,114,240,2
,208,215,177,203,201,32,208,3,136,208,2
47,209,132,212,96
230 DATA 104,104,141,254,6,104,141,253
,6,169,0,133,213,216,165,136,133,205,1
65,137,133,206,160,0,177
240 DATA 205,205,253,6,208,0,280,177,2
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
,201,55,248,4,160,0,240,0,132,212,96
```

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

```
10 DATA 203,265,465,844,294,973,652,27
0,970,797,270,275,835,289,301,7639
50 DATA 355,94,254,420,935,840,580,41
,974,564,5435
```


ARCHON™

by Free Fall Associates
and Electronic Arts



THE LIGHT AND THE DARK

continued from page 15

wards over your drawn line, and if you stand still, a fuse lights on your line and follows the line, trying to catch you. There are also sparks and—well you get the idea. This game is rather slow, especially the fill routine, and not graphically exciting, but it is strategically intriguing.

Super Breakout

The ultimate bouncing-ball game, grandson of *Pong*. There are four variations of the basic game, which involves moving your paddle across the bottom of the screen to bounce the ball back up and knock out bricks. These variations include advancing walls and freeing balls from brick prisons to help you knock out other bricks.

Final Legacy

This complex game combines elements of strategy and arcade. You command a super battleship, roaming the seas of the far future. Your mission is to protect the remaining cities from the automatic war machinery left over from mankind's last effort to destroy himself. As you move from port to port, you may be intercepted by enemy vessels who will shoot missiles at you while you try to sink them. Another goal is to blast the missile silos before they launch their missiles; unfortunately, they launch when attacked, so you must then defend the cities from the airborne

missile attacks. Each of these obstacles is played on a different screen, with the main controls from the ship screen. The graphics are superb, and the action is fast and furious.

Choplifter

Pilot your helicopter from its base into enemy territory to try to rescue the 64 hostages held in camps. Your chopper can only lift 16 at a time, so it must make multiple trips back to the base. Each trip subjects you to more hazards: tanks, homing mines and speedy jet planes. To rescue hostages, you must blast open their camp (careful not to kill any with your fire), then land and wait while they board. You have to make sure not to wait too long, because tanks and other nasties will try to blast you while you are on the ground. The tanks also kill lots of hostages with their fire if you let them. Once the hostages have boarded, you must head back to base, being extra careful not to get shot down—which loses one of your lives and kills all the hostages aboard. This is a challenging game, with well-animated graphics: the hostages wave their thanks as they disembark.

Crystal Castles

Each screen of this maze-like 3-D game contains many little pills. The object is to guide your little bear as he walks around, picking up the pills. He will be threatened

by trees, tornadoes, and other hazards. The simulated 3-D graphics work well, and your bear can go behind obstacles and through doors.

Mario Brothers

Mario (and his brother Luigi in the two-player version) is back. The pipe-pests are giving the Mario brothers grief, getting in their way as they try to fix the plumbing. The object in this multilevel arcade hit is to jump up, bumping the level above under the enemies, which include Shellcreepers, Sidesteppers, Fighterflies and Fireballs. This will flip them over (although some require more than one bump). Mario can then kick them off the screen before they can right themselves. There are various tokens for extra points, a bonus coin round, and the animation is cute and effective.

Into the Eagle's Nest

It's time to venture into the famed Nazi fortress to rescue Allied prisoners. The fortress is packed with enemy soldiers, who will try to touch (?) you, stealing your strength. You can shoot them, provided you make sure to keep a good supply of ammunition, which doesn't seem to be in short supply in the Eagle's Nest. You can also shoot open chests to get grenades and grab elevator passes to get to the next floor. Other useful items include first aid kits. But make sure one of your bullets doesn't hit a chest full of dynamite, or the game is over!

Ace of Aces

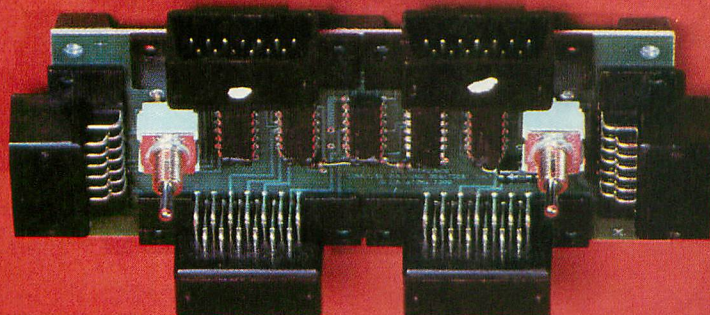
Pilot your Mosquito Fighter/Bomber on missions against Nazi-occupied Europe. You have many tasks to take care of, including equipping your plane, dog fighting with enemy fighters, and bombing your targets. The graphics are quite good and the action is exciting.

Airball

You are an airball (guess that's better than an airhead), roaming the corridors of an ancient mansion in search of a book of spells. You have a slow leak, so must periodically find an air pump to fill up (but don't overfill). The rooms of the mansion are chock-full of hazards, including needles, spikes, and such. The simulated 3-D perspective is effective, although the joystick controls take a little getting used to.

continued on page 79

QUINTOPUS COMPUTER SOFTWARE SERVICES



P.O. Box 17660
Rochester, New York
14617
(716) 467-9326
\$39.95 (\$59.95 with
switchable ports)

Reviewed by Jim Patterson

Well, those folks at Computer Software Services (CSS) have done it again. They see a need and rush to their drawing boards. This time they have come up with a jim-dandy of a device for the 8-bit Atari line.

If you caught my review in the October '88 issue of ANALOG Computing, concerning the use of a second Atari 8-bit as either a printer buffer or a high-speed disk emulator, you may be aware of the problem that I'm going to be referring to in the next paragraph and the one CSS has addressed with their new product.

The problem is the lack of daisy-chaining ability for far too many peripherals, such as the Atari XM301 modem, the Atari 410 cassette recorder, and the two devices mentioned above. Many of these devices are designed to be the last in the daisy-chain line, presenting quite an inconvenience to the user, not to mention a hardship on the hardware. You are required to pull plugs every time you want to use one of these devices.

As stated earlier, the innovators at CSS have seen the need, and two fine products are now available: the regular *Quintopus* and the deluxe *Quintopus* with two switchable ports.

According to the manufacturer, the *Quintopus* should be the first peripheral in the daisy-chain line plugged directly into the computer with all other peripherals plugged into the other five remaining ports. If you own more than five peripherals you may then proceed to daisy-chain the extra devices into the others already plugged into the *Quintopus*.

In my tests I found the *Quintopus* to work fine no matter where it was located in the daisy-chain line.

One good reason for the *Quintopus* being first in line, as stated in the operating instructions, is for use with modified disk drives capable of Warp Speed or Ultra Speed data-transfer rates. The literature states that it greatly increases the reliability factor of these increased transfer rates by shortening the distance the signals have to travel through the cable before reaching the computer, thereby reducing cable capacitance.

In my tests, the only difficulty I encountered was when connecting the Atari XM301 modem and the *Disk Emulator Routine* (DER) from B. L. Enterprises. This problem was quickly resolved by connecting the modem to one of the two switchable ports available on the Deluxe version. In discussing this problem with Ron Bryant of CSS, he stated that it was probably due to the XM301 not having an external power supply, so this problem would, I imagine, not be present with other modems such as the Atari SX212. Even with the XM301 connected to one of the unswitched ports and with the *Quintopus* both first and last in the daisy-chain line, only the DER was affected, all other devices functioned as before.

It's nice when a product does all that is promised and more. The two switchable ports allow the user to switch between any two devices. This is great if you own more than one computer and have only one printer or want to access the same disk drives. Of

course only one port can be activated at a time. In the case of two computers, they can both be on and working at the same time, but only one should be accessing the other devices at any given time.

Bryant assured me that they are doing everything they can to go along with Atari's "Power without the price!" by cutting costs to the consumer without sacrificing the quality of their products. They conducted a survey of their customers, and the results indicated that they agreed with CSS that a \$7 to \$10 price increase to the dealer wasn't in the consumer's best interest. Thus the *Quintopus* doesn't come with a "pretty case," but I think it more than meets our needs.

I would just like to state here what a pleasure it is to deal with a company willing to help with problems encountered with their products. This added bit of concern goes a long way toward influencing a next purchase. And it is a must when dealing with mail order where all you have to go on is an ad or perhaps a product review. I just hope other dealers take the hint.

Jim Patterson is a former Product Support Representative (PSR) for Texas Instruments. He holds a degree in electro-mechanical engineering and is currently working towards a degree in computer science. He is a member of PHI THETA KAPPA and the Data Processing Management Association (DPMA), and has been an avid Atari user since 1981.



ELECTRA-BALL:



A Futuristic Sports Game

by Frank Martone

Remember when video games used to be fun? I can recall many late nights playing games on my Atari 2600. The graphics were not great, but it really didn't matter. Back then it was the concept of the game that made playing the game fun. Now all anyone really cares about is the graphics that a game has. One of the reasons I think 2600 video games went down was the fact that the programmers used too much memory on making the games graphics look better rather than using the memory on playing value. So I decided to program a fun game, *Electra-Ball*, okay on graphics and loaded with play value.

Electra-Ball is a two-player futuristic sports game that can be played on any Atari 8-bit with two joysticks. There are two teams, the Red team and the White team. Each team has a goal; Red's goal has a red background, and White's goal has a blue background (don't ask). There are Red and White deflector panels scattered throughout the playfield. The object of the game is to deflect the electra-ball into the opponents goal. The first player to reach a player-selected score will win.

To select the winning score, simply move joystick #1 up or down. When the desired score appears on the title screen, press the joystick button, and the game will begin. The preset winning score is five goals. You can select from one to 25 goals as the winning score.

The deflector panels have three possible deflection positions: up, down and straight ahead. If you push the joystick up, all the panels on your team will face up, and if the ball hits the panel in this position the ball will be deflected up. Moving your joystick up or down makes your deflector panels face up or down, and moving your joystick left or right returns your deflector panels to the straight position. If the ball hits your panel in the straight position, the ball will simply be deflected straight ahead. After the ball collides with any panel on the playfield both teams panels are reset to the straight position.

To move your goalie, hold in the joystick button and push the joystick up or down. When the ball collides with a goalie, it is deflected away in a random direction. The electra-ball then becomes unpredictable, moving in an erratic course forcing both teams to be on their guard. Good Luck!

Frank Martone's interest in computers started in junior high school where he learned BASIC programming. He got his first computer, an Atari 400, when he was 12, and within two years, his first game was published. Frank is now 20.



LISTING 1: BASIC

```

WN 0 REM *****
SV 1 REM *      ELECTRA-BALL      *
WK 2 REM *      BY FRANK MARTONE  *
WQ 3 REM *****
UV 4 GOSUB 750:SC=0:SC1=0:GOTO 3999
XM 5 GRAPHICS 17:POKE 559,0:POKE 708,91:P
   OKE 710,0:POKE 623,1:GY=11:GP=11
RQ 6 POKE 756,CH/256:POKE 709,135
ZS 8 POSITION 4,0:? #6;"ELECTRA-BALL":POK
   E 710,14:RPV=38:WPV=166:GOSUB 100:GOSU
   B 200:GOSUB 600
CR 9 GOSUB 1500:GOTO 400:POKE 623,1
WX 10 SOUND 0,0,0,0:ST=STICK(0):ST1=STICK
   (1):IF ST=7 OR ST=11 THEN RPV=38:GOSUB
   100
DA 11 POSITION 6,22:? #6;SC:POSITION 17,2
   2:? #6;SC1:POKE 77,0
TQ 14 IF ST=13 THEN RPV=37:GOSUB 100
VX 15 IF ST=14 THEN RPV=39:GOSUB 100
BM 16 POSITION P,Q:? #6;"":POKE 623,4:PO
   KE 53250,134:POKE 704,65:POKE 705,133
WS 20 IF ST1=7 OR ST1=11 THEN WPV=166:GOS
   UB 200
TK 21 IF ST1=13 THEN WPV=167:GOSUB 200
SA 22 IF ST1=14 THEN WPV=165:GOSUB 200
WG 23 IF W=35 THEN GOSUB 2000
IE 24 IF W=163 THEN GOSUB 2050
BB 25 POSITION 1,GY:? #6;"+"
SE 31 IF STRIG(0)=0 AND ST=14 THEN POSITI
   ON 1,GY:? #6;"":GY=GY-1
MO 32 IF STRIG(0)=0 AND ST=13 THEN POSITI
   ON 1,GY:? #6;"":GY=GY+1
HK 33 IF GY>12 THEN POSITION 1,GY:? #6;"
   ":GY=GY-1
CK 34 IF GY<10 THEN POSITION 1,GY:? #6;"
   ":GY=GY+1
PS 35 POSITION 1,GY:? #6;"+" :POSITION 17,
   GP:? #6;" "
AB 41 IF STRIG(1)=0 AND ST1=14 THEN POSIT
   ION 17,GP:? #6;"":GP=GP-1
UG 42 IF STRIG(1)=0 AND ST1=13 THEN POSIT
   ION 17,GP:? #6;"":GP=GP+1
PF 43 IF GP>12 THEN POSITION 18,GP:? #6;"
   ":GP=GP-1
KD 44 IF GP<10 THEN POSITION 18,GP:? #6;"
   ":GP=GP+1
QH 45 POSITION 17,GP:? #6;" " :RETURN
HN 100 POSITION 9,3:? #6;CHR$(RPV):POSITI
   ON 2,5:? #6;CHR$(RPV)

```

```

BK 101 POSITION 12,5:? #6;CHR$(RPV):POSIT
   ION 10,8:? #6;CHR$(RPV)
EQ 102 POSITION 15,8:? #6;CHR$(RPV):POSIT
   ION 7,11:? #6;CHR$(RPV)
TW 103 POSITION 12,11:? #6;CHR$(RPV):POSIT
   ION 10,14:? #6;CHR$(RPV)
MD 104 POSITION 15,14:? #6;CHR$(RPV):POSIT
   ION 2,17:? #6;CHR$(RPV)
FZ 105 POSITION 12,17:? #6;CHR$(RPV):RETU
   RN
TR 200 POSITION 6,5:? #6;CHR$(WPV):POSITI
   ON 16,5:? #6;CHR$(WPV):POSITION 3,8:?
   #6;CHR$(WPV)
PE 201 POSITION 8,8:? #6;CHR$(WPV):POSITI
   ON 6,11:? #6;CHR$(WPV)
UY 202 POSITION 11,11:? #6;CHR$(WPV):POSIT
   ION 3,14:? #6;CHR$(WPV)
PA 203 POSITION 8,14:? #6;CHR$(WPV):POSIT
   ION 6,17:? #6;CHR$(WPV):POSITION 16,17
   :? #6;CHR$(WPV)
FT 204 POSITION 9,19:? #6;CHR$(WPV):RETUR
   N
EN 300 RPV=38:WPV=166
SN 301 POKE 712,233:SOUND 0,25,10,2:SOUND
   1,45,10,2:GOSUB 100:GOSUB 200
MM 302 SOUND 0,0,0,0:SOUND 1,0,0,0:POKE 7
   12,0:P=P+0:Q=Q+K:RETURN
HZ 400 P=9:Q=11:D=1:K=0
YY 410 POSITION P,Q:? #6;" "
EN 425 IF W=172 THEN D=-1:K=K+INT(RND(0)*
   1):K=K-RND(0)*1:GOSUB 3050:GOSUB 300:G
   OSUB 10
PE 426 IF W=43 THEN D=1:K=K+INT(RND(0)*1)
   :K=K-RND(0)*1:GOSUB 3000:GOSUB 300:GOS
   UB 10
HL 427 IF W=169 THEN D=-1:K=0:GOSUB 300:G
   OSUB 10
VJ 428 IF W=38 THEN D=1:K=0:GOSUB 300:GOS
   UB 10
VZ 429 IF W=37 THEN D=1:K=1:GOSUB 300:GOS
   UB 10
HD 430 IF W=39 THEN D=1:K=-1:GOSUB 300:GOS
   UB 10
US 431 IF W=165 THEN D=-1:K=-1:GOSUB 300:
   GOSUB 10
FO 432 IF W=166 THEN D=-1:K=0:GOSUB 300:G
   OSUB 10
HG 434 IF W=167 THEN D=-1:K=1:GOSUB 300:G
   OSUB 10
MU 435 IF P<1 AND K=0 THEN D=1:SOUND 0,10
   ,10,10
JY 436 IF P<1 AND K>0.1 THEN D=1:K=-1:SOU
   ND 0,20,10,10

```



```

XQ 437 IF P<1 AND K<-1 THEN D=1:K=1:SOUN
D 0,10,10,10
AZ 440 IF Q>19 THEN K=-1:SOUND 0,10,10,10
SF 441 IF P>17 AND K=0 THEN D=-1:SOUND 0,
20,10,10
PY 442 IF P>17 AND K=1 THEN D=-1:K=1:SOUN
D 0,10,10,10
K5 443 IF P>17 AND K=-1 THEN D=-1:K=-1:50
UND 0,20,10,10
IO 445 IF Q<4 THEN K=1:SOUND 0,15,10,10
CO 446 POSITION P,Q:?"#6;"":P=P+D:Q=Q+K:
LOCATE P,Q,W:GOSUB 10
GQ 447 SOUND 0,0,0,0:SOUND 1,0,0,0:GOTO 4
10
EC 499 REM P/M GRAPHICS
GZ 500 I=PEEK(106)-16:POKE 54279,I:POKE 5
3277,3:P0=I*256+1024:Y=110:Q=110:S=32:
K=120:POKE 623,1:POKE 559,0
MW 501 RESTORE 520:POKE 623,4:POKE 704,17
3:POKE 705,173
HX 502 P1=I*256+1280:P2=I*256+1536:POKE 7
06,223:P3=I*256+1792
LK 510 FOR L=P0+Y TO P0+28+Y:READ D:POKE
L,D:NEXT L
TR 512 FOR L=P1+Q TO P1+28+Q:READ D:POKE
L,D:NEXT L:RESTORE 520
BB 513 FOR L=P2+5 TO P2+7+5:READ D:POKE L
,D:NEXT L
AP 514 FOR L=P3+K TO P3+7+K:READ P:POKE L
,P:NEXT L:POKE 623,1:POKE 704,4:POKE 7
05,4:POKE 53259,0
VW 515 POKE 53248,32:POKE 53249,184:RETUR
N
JF 520 DATA 255,255,255,255,255,255,255,2
55
JI 521 DATA 255,255,255,255,255,255,255,2
55
JL 522 DATA 255,255,255,255,255,255,255,2
55
JO 523 DATA 255,255,255,255,255,255,255,2
55
JR 524 DATA 255,255,255,255,255,255,255,2
55
JU 525 DATA 255,255,255,255,255,255,255,2
55
JX 526 DATA 255,255,255,255,255,255,255,2
55
KA 527 DATA 255,255,255,255,255,255,255,2
55
YM 528 DATA 6,12,24,30,4,8,16,32
FD 530 DATA 144,18,64,0,214,0,16,146
KR 600 POSITION 0,1:?"#6;"":GOSUB 500
GOSUB 500:POKE 711,133:GOSUB 500
MO 601 POSITION 0,21:?"#6;"":GOSUB 500
GOSUB 500
VQ 605 FOR J=10 TO 12:POSITION 0,J:?"#6;"
":NEXT J
IQ 606 FOR J=10 TO 12:POSITION 18,J:?"#6;"
":NEXT J:POKE 559,62:POKE 53256,3:P0
KE 53257,1
ZE 610 RETURN
SR 750 GRAPHICS 1:GOSUB 500:POKE 709,14:P
OKE 710,0:POKE 752,1:DL=PEEK(560)+256*
PEEK(561):POKE 623,4
EN 751 POKE 559,46:?"#6:?"#6:POKE 752,1:?"
FROM ANALOG COMPUTING"
YA 752 ?"#6:?"#6:?"#6:POKE 711,79
RA 753 ?"#6;" ELECTRA-BALL BY FRANK
MARTONE"
DQ 754 POKE 710,131:POKE DL+10,3:POKE 712
,131:POKE 708,255:POKE 709,14:POKE 705
,84:POKE 704,82:POKE 711,88
ND 756 POSITION 0,13:?"#6;"":POSITION 3
,12:?"#6;" reading data "
BQ 757 POSITION 0,9:?"#6;"":

```

```

RW 758 GOSUB 1000
ZP 760 RETURN
SQ 770 T=20:R=220
KQ 771 POSITION 6,22:?"#6:5C:POSITION 17,
22:?"#6:5C1:FOR Z=1 TO 10:NEXT Z
QU 775 T=T+8:R=R-8:SOUND 0,T,6,4:SOUND 1,
R,10,4
WX 780 POKE 53248,T:POKE 53249,R
KG 781 IF T=220 THEN POKE 53248,96:POKE 5
3249,128:SOUND 0,0,0,0:SOUND 1,0,0,0:R
ETURN
TY 790 GOTO 775
PL 1000 REM CHARACTER SET
EH 1010 CH=(PEEK(106)-8)*256
BS 1015 FOR I=0 TO 512:POKE CH+I,PEEK(573
44+I):NEXT I
HM 1020 RESTORE 1100
SX 1030 READ A:IF A=0 THEN RETURN
TY 1040 FOR J=0 TO 7:READ B:POKE CH+A*8+J
,B:NEXT J
OQ 1050 GOTO 1030
JB 1100 DATA 42,16,24,92,222,0,255,254,12
4
RT 1105 DATA 1,0,0,192,225,115,51,51,51
SX 1106 DATA 3,0,0,3,7,15,31,31,3
PN 1110 DATA 12,62,93,107,127,107,93,62,0
KV 1115 DATA 11,0,0,62,127,223,255,127,0
KY 1117 DATA 14,0,0,131,230,252,124,6,3
PH 1125 DATA 5,3,1,0,24,28,12,1,0
EY 1130 DATA 6,254,248,24,0,0,128,0,64
QY 1135 DATA 7,0,12,36,60,24,4,40,128
RZ 1140 DATA 2,255,255,255,0,255,0,255,25
5
NT 1149 DATA 56,0,0,40,16,4,1,2,0
WK 1175 DATA 26,0,0,0,0,0,0,0,3
AZ 1176 DATA 27,0,198,41,128,16,16,0,16
LJ 1177 DATA 28,0,0,0,0,0,0,128,192
CJ 1178 DATA 29,2,6,14,60,254,227,128,0
MP 1179 DATA 30,0,56,124,124,124,56,0,0
WC 1180 DATA 31,24,24,24,24,24,24,24,0
HU 1181 DATA 32,0,0,68,40,68,40,66,0
KZ 1182 DATA 61,0,0,68,238,124,56,24,28
YU 1183 DATA 62,224,192,224,24,12,24,32,0
BA 1184 DATA 49,0,0,126,106,86,126,0,0
TP 1192 DATA 59,0,0,34,119,62,28,24,56
CE 1193 DATA 13,0,62,127,223,255,126,124,
12
MZ 1194 DATA 5,1,2,4,0,0,32,64,0
EU 1195 DATA 6,8,8,0,0,0,8,8,8
NA 1196 DATA 7,64,32,16,0,0,2,1,0
DY 1197 DATA 8,1,2,4,8,16,32,64,0
SV 1198 DATA 9,8,8,8,8,8,8,8,8
AV 1199 DATA 10,64,32,16,8,4,2,1,0
ZE 1200 DATA 11,1,242,250,46,250,242,1,0
BX 1201 DATA 12,0,128,158,183,230,183,158
,128
TR 1202 DATA 49,170,255,255,255,153,255,2
55,85
EV 1203 DATA 56,255,255,255,255,255,0,255
,255
AF 1210 DATA 16,60,66,66,66,0,66,66,60
KX 1211 DATA 17,8,8,8,8,0,8,8,8
PR 1212 DATA 18,62,1,1,1,30,32,32,31
AY 1213 DATA 19,62,1,1,1,62,1,1,62
TQ 1214 DATA 20,34,34,34,34,60,2,2,2
HG 1215 DATA 21,30,32,32,32,28,2,2,60
WL 1216 DATA 22,30,32,32,32,60,34,34,28
XH 1217 DATA 23,28,34,34,2,0,2,2,2
WC 1218 DATA 24,60,66,66,66,60,66,66,60
SQ 1219 DATA 25,28,34,34,34,28,2,2,60
YX 1230 DATA 3,84,170,84,170,84,170,84,17
0
EX 1250 DATA -1
RQ 1500 SOUND 0,0,0,0:SOUND 1,0,0,0:POSIT
ION 1,22:?"#6:"RED ":"0:"5C;" WRITE
":"0:"5C1:RPV=38:MPV=166
ES 1501 GOSUB 100:GOSUB 200:POSITION 9,11

```



```

: ? #6;"":FOR J=1 TO 7:POKE 623,4:POSI
TION 1,GY: ? #6;" ":POSITION 17,GP
PM 1502 ? #6;" ":POSITION 5,10: ? #6;"GET
READY":FOR C=1 TO 20:NEXT C:SOUND 0,10
0,10,10
OY 1503 POSITION 5,10: ? #6;" ":FO
R C=1 TO 20:NEXT C:SOUND 0,0,0,0:NEXT
J:POKE 704,65:POKE 705,133
RI 1504 SOUND 0,50,10,10:FOR C=1 TO 20:NE
XT C:W=32
DJ 1505 FOR J=10 TO 12:POSITION 0,J: ? #6;
" ":NEXT J
BE 1506 FOR J=10 TO 12:POSITION 18,J: ? #6;
" ":NEXT J:POKE 559,62:POKE 53256,3:P
OKE 53257,1
XQ 1507 GY=11:GP=11
UX 1508 POKE 53251,120:POKE 707,255:FOR Z
=15 TO 0 STEP -0.6:SOUND 0,100,0,Z:POK
E 707,Z:NEXT Z:POKE 53251,0
AM 1510 RETURN
MO 1999 REM GOAL!
JE 2000 FOR R=10 TO 15:POSITION 0,R: ? #6;
" ":NEXT R
PM 2001 FOR T=15 TO 0 STEP -1:SOUND 0,T*1
8,10,T:SOUND 1,T*5,8,T:POKE 712,T:NEXT
T
GF 2003 FOR T=1 TO 4:POSITION 1,GY: ? #6;"
+":FOR G=1 TO 5:NEXT G:POSITION 1,GY: ?
#6;" ":FOR G=1 TO 5:NEXT G:NEXT T
AS 2007 FOR R=1 TO 5:FOR T=-15 TO 15 STEP
1:SOUND 0,ABS(T)+25,10,10:POKE 704,AB
S(T+5)
BJ 2008 NEXT T:NEXT R:POKE 704,65:POKE 70
5,133:GY=11
JQ 2009 SOUND 0,0,0,0:SC1=5C1+1:IF 5C1=5C
G THEN 8000
QW 2010 POKE 53278,255:P=9:Q=11:D=-1:K=0:
REM LOSERS OUT
CN 2011 FOR J=10 TO 12:POSITION 0,J: ? #6;
" ":NEXT J
IW 2012 FOR T=15 TO 0 STEP -1:SOUND 0,T/2
,12,T:SOUND 1,T/2,12,T:NEXT T
EF 2013 GOSUB 1500:GOTO 10
SZ 2050 FOR R=10 TO 15:POSITION 17,R: ? #6;
" ":NEXT R
QB 2051 FOR T=15 TO 0 STEP -1:SOUND 0,T*1
8,10,T:SOUND 1,T*5,8,T:POKE 712,T:NEXT
T
QY 2053 FOR T=1 TO 4:POSITION 17,GP: ? #6;
" ":FOR G=1 TO 5:NEXT G:POSITION 17,GP
: ? #6;" ":FOR G=1 TO 5:NEXT G:NEXT T
EC 2057 FOR R=1 TO 5:FOR T=-15 TO 15 STEP
1:SOUND 0,ABS(T)+25,10,10:POKE 705,AB
S(T+5)
IE 2058 NEXT T:NEXT R
SR 2059 SOUND 0,0,0,0:POKE 705,133:5C=5C+
1:POKE 704,65:IF 5C=5CG THEN 8000
NS 2060 POKE 53278,255:P=9:Q=11:D=1:K=0:R
EM LOSERS OUT
ZN 2061 FOR J=10 TO 12:POSITION 18,J: ? #6;
" ":NEXT J
JL 2062 FOR T=15 TO 0 STEP -1:SOUND 0,T/2
,12,T:SOUND 1,T/2,12,T:NEXT T
EU 2063 GOSUB 1500:GOTO 10
OC 3000 POSITION 1,GY: ? #6;" "+
IT 3001 FOR R=7 TO 0 STEP -1:SOUND 0,7,4,
R:POKE 712,R+50:NEXT R
ZY 3002 POSITION 1,GY+1: ? #6;" ":POSITION
1,GY-1: ? #6;" ":RETURN
RP 3050 POSITION 17,GP: ? #6;" "
OC 3051 FOR R=7 TO 0 STEP -1:SOUND 0,6,4,
R:POKE 712,R+53:NEXT R
SO 3052 POSITION 17,GP+1: ? #6;" ":POSITIO
N 17,GP-1: ? #6;" ":RETURN
FK 3999 GRAPHICS 2:POKE 559,0:GOSUB 500:P
OKE 53248,0:POKE 53249,0:5C=0:5C1=0
LU 4000 POKE 756,CH/256:POKE 708,135:POKE
710,0:POKE 755,0: ? " A N A L O G
C O M P U T I N G"
OM 4001 POKE 559,62:POSITION 4,0: ? #6;"EL
ECTRA>BALL":POKE 623,1:POKE 709,216
XT 4002 DL=PEEK(560)+256*PEEK(561):POKE D
L+6,6

```

```

AR 4003 ? #6;" SPORT OF THE FUTURE":POKE
DL+9,6
MB 4004 POSITION 3,4: ? #6;"by frank marto
n":5CG=5:POKE 711,14
CH 4050 FOR T=20 TO 130:POKE 53250,T:NEXT
T
UF 4051 FOR R=1 TO 180 STEP 3:POKE 53250,
RND(0)*4+134:POKE 712,14:POKE 712,0:P0
KE 708,133+RND(0)*8
ML 4055 SOUND 0,R*9,2,4:SOUND 1,R*5,8,10
IG 4060 NEXT R:SOUND 0,0,0,0:SOUND 1,0,0,
0
SH 4061 FOR R=15 TO 0 STEP -0.3:SOUND 0,R
*9,0,R:POKE 708,R+130:NEXT R
JZ 4062 POKE 708,155:U=85:F=3:POKE 53259,
3:POKE 623,1
UY 4065 POSITION 4,6: ? #6;"enter score ";
"0":5CG;" ":IF STRIG(0)=0 THEN 4078
ED 4067 IF STICK(0)=13 THEN 5CG=5CG-1:FOR
D=15 TO 0 STEP -3:SOUND 0,80,10,D:NEX
T D
KK 4068 IF STICK(0)=14 THEN 5CG=5CG+1:FOR
D=15 TO 0 STEP -3:SOUND 0,60,10,D:NEX
T D
IK 4069 SOUND 0,0,0,0:IF 5CG<1 THEN 5CG=1
:SOUND 0,255,10,10
DC 4070 IF 5CG>25 THEN 5CG=1:SOUND 0,255,
10,10
UP 4071 GOTO 4065
IL 4078 RESTORE 7100:GOSUB 7000:GOTO 5
KT 7000 RESTORE 7100
YH 7005 READ MUSIC
JS 7006 IF MUSIC=255 THEN SOUND 1,0,0,0:5
OUND 0,0,0,0:RETURN
VZ 7010 FOR G=15 TO 0 STEP -6: ? :SOUND 0,
MUSIC+10,10,G:SOUND 1,MUSIC+14,10,3:NE
XT G:POKE 708,MUSIC
SG 7020 GOTO 7005
OK 7050 RESTORE 7150
VA 7055 READ MUSIC,H
LN 7056 IF MUSIC=255 THEN SOUND 1,0,0,0:5
OUND 0,0,0,0:POKE 704,0:POKE 705,0:RET
URN
GD 7057 FOR G=15 TO 0 STEP -6:SOUND 0,MUS
IC,10,7:SOUND 1,H,10,G:POKE 704,G+55:P
OKE 705,G+55:NEXT G
WI 7058 GOTO 7055
SZ 7100 DATA 100,50,100,50,100,50,100
YC 7152 DATA 100,60,100,80,100,60,100
YG 7153 DATA 100,60,100,80,100,60,100
YK 7154 DATA 100,60,100,80,100,60,100
VL 7155 DATA 100,50,100,50,100,60,100
XA 7156 DATA 100,60,100,60,100,60,100
IZ 7157 DATA 50,80,50,50,80,80,50,80
QR 7158 DATA 40,30,40,50,40,40,30,40
YT 7159 DATA 255,255,255,255,255
MZ 7200 FOR W=1 TO 50:SETCOLOR 0,PEEK(537
70),PEEK(53770):NEXT W
DL 8000 GOSUB 770:GRAPHICS 18:POKE 756,CH
/256:POKE 53248,0:POKE 53249,0:POKE 55
9,62:POKE 710,14:POKE 708,91
OR 8001 POKE 623,1:POKE 53250,127
GZ 8010 POSITION 6,0: ? #6;"game>over"
SB 8020 POSITION 5,3: ? #6;"final score"
CS 8030 POSITION 5,5: ? #6;"RED ";5C
AI 8031 POSITION 5,7: ? #6;"WHITE ";5C1
PA 8032 POSITION 0,9: ? #6;"cccccccccccccccc
cccccc"
KC 8034 POSITION 0,1: ? #6;"cccccccccccccccc
cccccc"
JE 8035 FOR R=15 TO 0 STEP -0.3:SOUND 0,R
*55,16,R:POKE 712,R*5:NEXT R
EP 8037 RESTORE 7150:GOSUB 7050
ZJ 8041 FOR R=1 TO 270 STEP 3:POKE 53250,
RND(0)*4+124:POKE 712,14:POKE 712,0:P0
KE 711,133+RND(0)*8
FS 8042 SOUND 0,R*9,8,4:SOUND 1,R*5,0,10
IQ 8043 NEXT R:SOUND 0,0,0,0:SOUND 1,0,0,
0
HH 8048 POKE 53250,0:FOR R=0 TO 16:PLOT R
,0:DRAWTO R,9:FOR D=1 TO 2:NEXT D:NEXT
R:GOTO 3999
CC 19999 END

```


continued from page 72

Summer Games

This one's a real joystick-buster, but the fluid animation is hard to beat. Compete in swimming, skeet shooting, pole vaulting, diving and gymnastics.

Desert Falcon

Pilot a falcon over the desert, shooting enemy creatures such as flying fish, vultures and scarabs with your arrows. The falcon can land and pick up treasures for points, or grab three hieroglyphics to gain superpowers such as invincibility and a "smart bomb" that destroys all enemies. At the end of each section, battle with the giant sphinx. The 3-D, three-quarter-perspective graphics and animation are very smooth, although controlling the falcon with your joystick takes some getting used to.

Thunderfox

Easily the hardest game of all the games reviewed here (typical games lasted just a minute or two), the object of this graphically superior game is to pilot your craft over a hostile spaceship, defending yourself with lasers and dropping bombs on various parts of the ship to blow open a hole. Once the hole has been blasted open, you must pilot the ship inside, where lasers will try to shoot you down, until you reach a final confrontation, the "Crystal Guardian." You must switch weapons by pressing the Option key (or space bar), and the manual doesn't tell you that parts of the enemy ship stick up high enough to crash into!

Dark Chambers

There are 26 levels in this game, with each level (A through Z) being a multiscreen scrolling maze. The object is to guide your well-detailed little man through the maze, shooting the hordes of enemies who chase you. These enemies originate from "spawners" which can be destroyed with your gun. The enemies (also drawn well) chase you with a mindlessness that can be turned to your advantage. Most enemies take multiple shots to destroy, mutating into less powerful forms each time they are shot. Strength can be replenished by eating the food lying around,

and better weapons, shields and treasure can be picked up. Two people can play, and one can rejuvenate the other if one dies. But watch out for the poison and traps! This game has three difficulty levels: The lowest will allow even a poor arcader to see most of the levels, while the most advanced is rugged indeed.

Realsports Tennis

In this one- or two-player game, you play tennis, plain and simple. The view of the court is from the behind one of the players, high up in the stands. As a match progresses, you take turns serving, and switch sides of the court, so that sometimes you control the player closest to you, and other times you control the player across the court. The "ball" throws a shadow on the ground so that you can judge its height, and this works pretty well. Using your joystick, you guide the player close to the ball; the racket is swung automatically. Both backhand and forehand swings are possible, and you can guide where you want the ball to go by pressing the fire button and pushing the joystick in the desired direction before hitting the ball. This is tricky, since your player stops running when the button is pressed. Scoring is kept just as in real tennis, and there are two levels of difficulty when playing against the computer.

Realsports Football

This cartridge is one of the more complex, but the control scheme is easy to use. The object is (surprise) to outscore the opposing team using touchdowns, field goals and safeties. The game does not support an "extra point" feature; instead, each touchdown is automatically worth seven points. This game is best played against a human opponent, although there is a practice mode to familiarize yourself with the available plays. To call plays, you must specify a formation (when requested by the scoreboard), and a play using your joystick. There are 15 offensive plays, and five defensive plays. A play card is included to remind you what plays are available. Once the play begins, each participant controls a single on-screen player. You can change who you are controlling by pressing the fire button, but only once during each scrimmage. Passing is a little tricky, but it is a skill that can be learned, and the

various players run at certain speeds, so plays need to be called with this in mind. Interceptions, quarterback sacks, and other events of real football can occur, but no penalties or injuries.

Crimebuster

This target-shooting game is for the light gun, and unfortunately suffers from a lack of accuracy (shots fired at the screen tend to hit to the left of the aiming point). You are presented with a map of the city. Firing at the map reveals rectangular areas where gang activities are suspected. Firing at the rectangle a second time then starts the action in that sector of the city. If the section of the city selected is not adjacent to the sector you just cleared, then you must drive your police car to your destination. When you arrive at your destination, you see various buildings, and mobsters shoot at you from the windows, doorways, manhole covers and other hiding places; you need to get them before they get you. Innocent bystanders also come on the scene. Shooting them costs you points and gets you a warning. If you run out of bullets before the enemy runs out of targets, you lose a life. The game is over either when you run out of lives or clear the city of crime activity; you earn a final rating either way. The graphics are very sharp.

Crossbow

In *Crossbow*, you must defend your friends as they travel through a dangerous world. First, you must select one of two or three paths. Since the paths don't show on the screen, you need to familiarize yourself with these paths so that you don't keep visiting the same place over and over again. You start with two friends, who wander through a city, forest, past an erupting volcano, through caverns, jungles, etc. In each place, there are all kinds of things trying to kill them. For example, in the city there are ghosts, a rock throwing demon and lightning bolts. The idea is for you to shoot the hazards out of the way before your friends are clobbered by them. If you are good, you can also blast a variety of other things for extra points. As you get better, more friends show up, and keeping more people alive is even harder. The graphics are satisfactory, but once again, the inaccuracy of the light gun is a problem. **A**

*Electronic Arts,
Activision,
Baudville,
Mindscape and
others have
recognized the
recent surge in the
entertainment-
software market*

continued from page 69

very similar to *Donkey Kong Jr.*, where you must jump from platform to platform while avoiding all sorts of perils.

I spent a lot of time killing off poor Marvin on the very first screen, until I found that he can only jump so far. In some instances he must just walk off a ledge, to fall a short distance down to the next level. Try to jump there, and Marvin is a goner.

This is a graphic arcade-style adventure, that is challenging from the beginning. It is frustrating in that each time Marvin gets clobbered, he must replay the entire screen, instead of picking up where he left off. Another problem is that, when Marvin runs out of lives, he must start at Screen 1 the next time. It would have been nice if *Mousetrap* had a continue feature, the way *Moon Patrol* does. These two "character flaws" make a charming little game more frustrating and tedious to play than it should be.

Gauntlet

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

Gauntlet is a popular arcade game that has been ported to the XEGS in a big way. This is a *Dungeons and Dragons* (TM) style, graphic adventure where you may take on the identity of one of four heroes including: Thor the Warrior, Thyra the Valkyrie (for lady gamers), Merlin the Wizard or Questor the Elf. Each has his (or her) own special strengths, rated in terms of armor, shot power, hand-to-hand combat and magic power. After your fighter is chosen, you are off to the dungeons, some 200 of them!

In the dungeons, you will face hordes of ghosts. The best strategy is to destroy the skull and crossbones that generate them, and then shoot all the ghosts in sight. If a potion is handy, you may be able to detonate all the creatures on the screen. Merlin is especially good at wielding magic.

Other villains you must face include

grunts, demons, lobbers, sorcerers and the horrid Mister Death. Death you cannot kill (makes sense, he's already dead), but he can sap 200 points from your score and then kill *you*. Death is best avoided at all costs.

Each dungeon is a maze, a puzzle to solve, that you navigate from a top-down view. Since not all of the maze is on the screen at once, the exit to the next level is not always obvious. Some dungeons have more than one exit, allowing you to skip several dungeons, say from four to eight, with a comparable increase in game play difficulty. Along the way you can collect extra armor, magic powder, more shot speed and power, fight power, and pick-up power (allowing you to carry more things). There are keys and treasures to acquire, and poisons and traps to avoid—as if all those nasty creatures weren't enough.

Gauntlet is a superb graphic adventure for the XEGS—and very challenging. You will not master it any time soon. If you do, however, *Gauntlet: The Deeper Dungeons* is available for \$24.95. This disk provides more than 500 extra dungeons to challenge your wits and reflexes.

Video Vegas

Baudville
1001 Medical Park Drive SE
Grand Rapids, MI 49506
(616) 957-3036
\$29.95

When you feel the need for some serious gambling, *Video Vegas* won't send you to the poorhouse. This package provides realistic gambling entertainment with the Lucky 7 Slot Machine, Draw Poker, Keno and Blackjack. They are all accurate simulations of what you are up against when you hit the glamor and glitz of Las Vegas.

You begin *Video Vegas* with a \$1,000 stake. In the glittering streets of a Las Vegas, neon-filled night, up pops a menu with the four game selections. The graphics of this game are stunning, and playability in every aspect of each simulation is superb. The games were done in BASIC, but obviously with plenty of assembly-language support. The documentation even provides



Video Vegas

“hacker notes,” encouraging you to study the code and learn some of the secrets of an accurate gambling simulation. (This does *not* mean that you can make copies of this game, the original or modified, for your friends, however.)

On any game screen you may enter a question mark to see your odds of winning the current simulation. The Lucky 7 Slot Machine has some excellent effects: dropping the coins in the slot, pulling the handle, and spinning the wheels. You can play this one for a long time before you get a payoff, but Escape always quits the current game and returns to the main menu. You will notice that as you go from game to game, your stake remains current. You don't start over with another \$1,000 with each new game.

In *Draw Poker* you are dealt five cards

and are allowed to make a bet of one to five dollars. Draw one time for a pair of jacks or better. Blackjack is played according to house rules. You can even split a matched pair of cards, double down or buy insurance against a dealer's ace. In Blackjack the number of decks used, from one to four, are user-selectable, allowing you to practice different “systems” against the computer—at no financial risk.

Keno is similar to bingo. From one to 15 numbers are selected from a board of 80. The computer then selects 20 of the 80 at random. You can win anywhere from a dollar to 20,000 or more. But the odds against you are much higher than would first appear.

Video Vegas is a remarkably well-done gambling simulation. It also provides an interesting study of odds. And the ability to

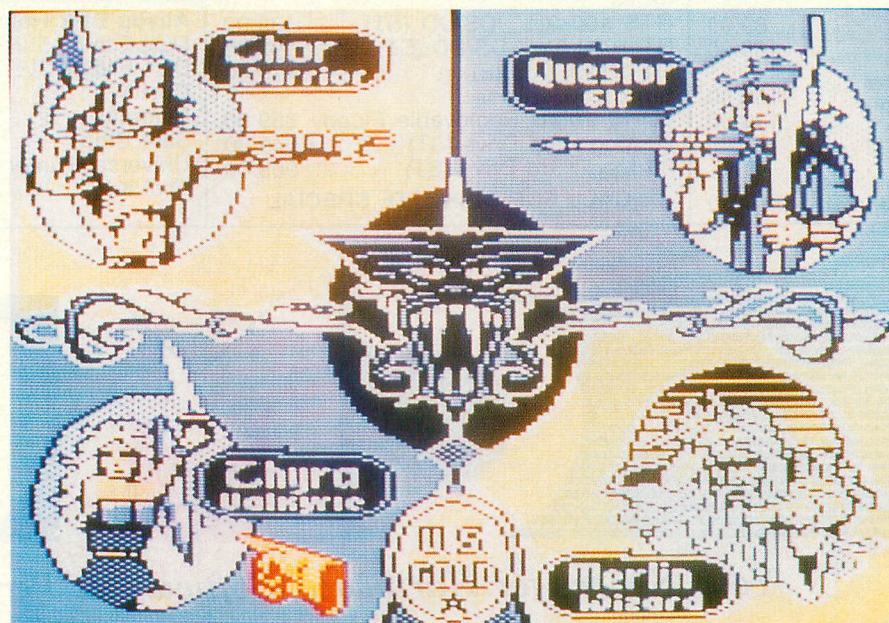
look at the BASIC source code provides the opportunity for you to learn more about it.

In Conclusion

Electronic Arts, Activision, Baudville, Mindscape and others have recognized the recent surge in the entertainment-software market due to the popularity of the XEGS and other game machines. But unlike many other game systems, the XEGS has the advantage of being a “real computer,” allowing you to take advantage of new software releases on disk.

The author would like to thank Randall's Home Computers of St. Louis for their valuable assistance in the development of this article.

A



Gauntlet

When you want to talk Atari

XL/XE HARDWARE



CMO SPECIAL

Atari 800XL \$69.99

65XE	109.00
130XE	149.00

INTERFACES

ICD	
P:R Connection	58.99
Printer Connection	34.99

Supra

1150	39.99
1151 (1200 XL)	40.99

Xetec

Graphix Interface	38.99
-------------------------	-------

Atari

850 Interface	109.00
---------------------	--------

XL/XE ENHANCEMENTS

Axlon 32K Mem. Board (400/800) ..	19.99
Atari 80 Column Card	74.99

ICD

BBS Express (ST)	52.99
Sparta DOS Construction Set	27.99
US Doubler/Sparta DOS	45.99
Real Time Clock	48.99
Rambo XL	26.99
US Doubler	27.99

MODEMS

Atari

SX212 300/1200 (ST)	78.99
XMM301	44.99

Anchor

VM520 300/1200 ST Dir. Con.	109.00
----------------------------------	--------

Avatex

1200 HC	89.99
2400	169.00

Supra

2400 Baud XL/XE or ST	159.00
2400 Baud (no software)	139.00

MONITORS

Magnavox

CM 8505 14" Composite/RGB/TTL ...	189.00
-----------------------------------	--------

ST HARDWARE

Call For Current Information
On The Entire ST Line!



ATARI SM1224 RGB/Color Monitor \$329

520ST FM RGB/Color System	829.00
SM124 Monochrome Monitor	179.00



CMO PACKAGE EXCLUSIVE

Atari 800XL & XF551 Drive

w/5 Undocumented ROMS Asteroids, Defender,
Missile Command, QIX, Star Raiders

\$259

DRIVES

Atari

ST 314 DS/DD	219.00
XF551 Drive (XL/XE)	179.00

I.B.

5 1/4" 40 Track (ST)	219.00
5 1/4" 80 Track (ST)	279.00

I.C.D.

FA•ST 20 Meg	629.00
FA•ST 30 Meg	849.00

FA•ST Dual Hard Drives

Indus

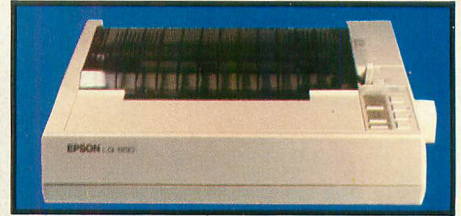
GTS 100 3 1/2" DS/DD (ST)	189.00
GT 1000 5 1/4" DS/DD (ST)	199.00
GT Drive (XL/XE)	179.00

Supra

FD-10 10MB Removable Floppy	859.00
30 Meg Hard Drive (ST)	649.00
60 Meg Hard Drive (ST)	989.00

CALL FOR DISKETTE SPECIAL

PRINTERS



EPSON LQ-850 24-Wire, 330 Cps \$499

Atari

1020 Printer, 40 Col. Color	19.99
1027 Letter Quality XL/XE	69.99
XDM121 LQ (XL/XE)	189.00
XM-M801 XL/XE Dot Matrix	189.00

Brother

M-1109 100 cps Dot Matrix	159.00
HR-20 22 cps Daisywheel	339.00

Citizen

120D 120 cps Dot Matrix	159.00
-------------------------------	--------

Epson

LX-800 150 cps, 80 col	189.00
FX-850 264 cps, 80 col	Call
LQ-500 180 cps, 24-wire	Call
LQ-1050 330 cps, 132 col	Call

NEC

P2200 pinwriter 24-wire	349.00
-------------------------------	--------

Okidata

Okimate 20 color printer	139.00
ML-182 + 120 cps, 80 column	229.00
ML-390 + 270 cps, 24-Wire	539.00

Panasonic

KX-P1080i 144 cps, 80 col	169.00
KX-P1091i 194 cps, 80 col	189.00
KX-P1124 24-Wire, 192 cps	319.00

Star Micronics

NX-1000 140 cps, 80 column	179.00
NX-1000 RainBow Color	239.00

Toshiba

P321-SL 216 cps, 24-wire	499.00
--------------------------------	--------

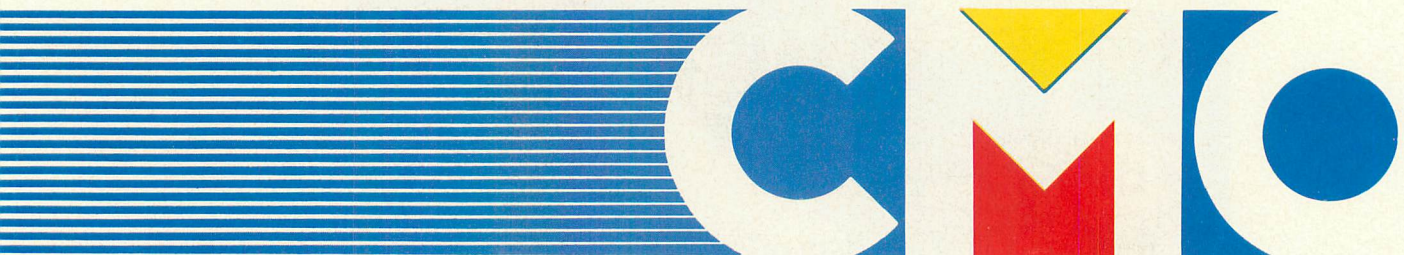
ACCESSORIES

Allsop Disk Holders

Disk File 60-5 1/4"	9.99
Disk File 30-3 1/2"	9.99

Curtis

Emerald	39.99
Safe Strip	19.99
Universal Printer Stand	14.99
Tool Kit	22.99



Your Source for Hardware, Software & Peripherals

.....you want to talk to us.

SPECIALS XL/XE

#AAB822P 822 Printer Paper.....	\$2.49
#AA14746 T.V. Switch Box.....	2.49
#AA4010 Tic-Tac-Toe.....	4.99
#AA4011 Star Raiders.....	3.99
#AA4012 Missile Command.....	3.99
#AA4013 Asteroids.....	4.99
#AA4022 Pac Man.....	4.99
#AA4025 Defender.....	4.99
#AA4027 QIX.....	4.99
#AA4102 Kingdom (Cass.).....	1.99
#AA4112 States & Capitals (Cass.).....	1.99
#AA4121 Energy Czar.....	1.99
#AA4123 Scram (Cass.).....	1.99
#AA4126 Speed Reading.....	2.99
#AA4129 Juggle's Rainbow.....	1.99
#AA415 File Manager.....	8.99
#AA4204 1020 Color Pens.....	1.99
#AA5047 Timewise (D).....	3.99
#AA5049 Visicalc (D).....	24.99
#AA5081 Mickey Outdoors.....	5.99
#AA5081 Music Painter (D).....	9.99
#AA6006 Counseling Procedure.....	1.99
#AA7102 Arcade Champ (No J. Stk).....	6.99
#AA8030 E.T. Phone Home.....	3.99
#AA8048 Millipede.....	4.99

CLOSEOUTS XL/XE

ROM CARTS (XL/XE) \$2⁹⁹ ea or 5 for \$13⁹⁹



Loose/Undocumented

Choose from: Space Invaders, Star Raiders, Missile Command, Asteroids, Pac Man, Galaxian, Defender, QIX, Super Breakout, E.T., Eastern Front, Robotron.

Rocklyn

Gorf.....	2.99
Anti-Sub (Disk).....	2.99
Journey to Planet.....	2.99

Atari Program Exchange

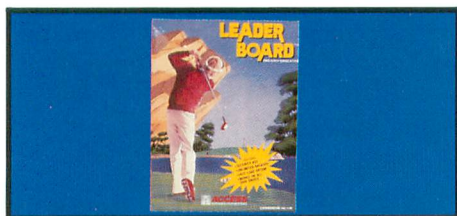
10 Different Cassettes For.....	\$11.99
---------------------------------	---------

SPECIALS XL/XE

Access	
Leaderboard Golf.....	13.99
Accolade	
Fight Night.....	19.99
Hardball.....	19.99
Atari	
Atariwriter Plus.....	35.99
Broderbund	
Graphics Library.....	18.99
Printshop.....	26.99
Datasoft	
Alternate Reality (City).....	23.99
221 Baker St.....	20.99
Electronic Arts	
Pinball Construction.....	15.99
Microprose	
Silent Service.....	23.99
Top Gunner.....	14.99
F-15 Eagle Strike.....	22.99
Origin Systems	
Ultima 4.....	36.99
Strategic Simulations	
Battalion Commander.....	25.99
Gemstone Warrior.....	11.99
Sublogic	
Flight Simulator.....	34.99

ST SOFTWARE

Accolade	
Hardball.....	21.99
Activision	
Hacker II/Music Studio (ea.).....	28.99
Antic	
Flash.....	21.99
Avant Garde	
PC Ditto.....	59.99
Batteries Included	
Degas Elite.....	37.99
Cygnus	
Starfleet.....	32.99



Access

Leaderboard Golf \$2⁹⁹

ST SOFTWARE

Data East	
Speed Buggy.....	25.99
Electronic Arts	
Auto Duel.....	31.99
Isgur's Portfolio.....	129.00
Firebird	
Silicon Dreams.....	13.99
The Sentry/Tracker (ea.).....	12.99
FTL	
Dungeonmaster.....	29.99
Metacomco	
ISO Pascal.....	59.99
Michtron	
Leatherneck.....	29.99
Microprose	
Gunship.....	28.99
F-15 Strike/Silent Service (ea.).....	24.99
Miles Software	
ST Wars.....	24.99
Mindscape	
High Roller.....	31.99
Mark Williams	
Debugger.....	45.99
Paradox	
Wanderer (3D).....	24.99

ST SOFTWARE

Progressive Computer	
Graphic Artist 1.5.....	109.00
Psygnosis	
Barbarian.....	25.99
Soft Logik Corp.	
Page Stream.....	129.00
Strategic Simulation	
Questron II.....	35.99
Timeworks	
Swiftcalc..... (ea.)	46.99
Desktop Publisher.....	79.99
Unison World	
Printmaster Plus.....	25.99



TimeWorks
WordWriter

\$44⁹⁹

In U.S.A.

Call: 1-800-233-8950

In Canada call: 1-800-233-8949 All Other Areas call: 717-327-9575 Fax: 717-327-1217
Educational, Governmental and Corporate Organizations call toll-free 1-800-221-4283
CMO, 101 Reighard Ave., Dept. B7, Williamsport, PA 17701

MEMBER
MMC
MICROCOMPUTER
MARKETING COUNCIL
of the Direct Marketing Association, Inc.

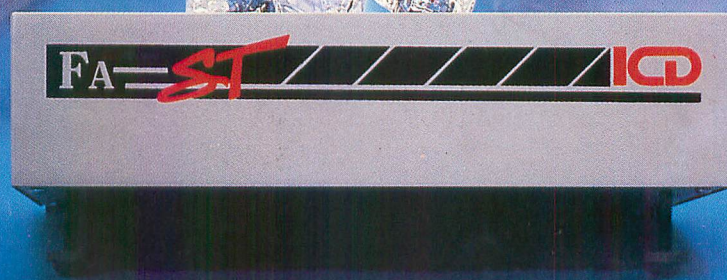
OVER 350,000 SATISFIED CUSTOMERS • ALL MAJOR CREDIT CARDS ACCEPTED • CREDIT CARDS ARE NOT CHARGED UNTIL WE SHIP

POLICY: Add 3% (minimum \$7.00) shipping and handling. Larger shipments may require additional charges. Personal and company checks require 3 weeks to clear. For faster delivery, use your credit card or send cashier's check or bank money order. Credit cards are not charged until we ship. Pennsylvania and Maryland residents add appropriate sales tax. All prices are U.S.A. prices and are subject to change, and all items are subject to availability. Defective software will be replaced with the same item only. Hardware will be replaced or repaired at our discretion within the terms and limits of the manufacturer's warranty. We cannot guarantee compatibility. All sales are final and returned shipments are subject to a restocking fee. We are not responsible for typographic or photographic errors.

CIRCLE #121 ON READER SERVICE CARD.

B703

Some Call It A Refreshing Change



We named our drive after its swift and aggressive behavior. But it's really not fair to limit this incredible peripheral to just one name.

Call it cool. Cool, calm and collected with its whisper-quiet fan to prevent heated situations.

Call it high-class. With refined style, its sleek design complements your Atari computer system.

Quite simply, functional elegance under your monitor that's designed to adjust to your system and lift your sights for easy viewing.

Call it friendly. Our FA-ST Hard Drive welcomes a host of features like dual DMA ports which invite new devices. Our SCSI expansion is ready when you are. And inside, our drive can handle a partner like no others

We Call It The FA-ST Hard Drive

can. Have the time? The FA-ST drive does . . . the right time, everytime.

Call it durable. Unwavering dependability from a winning

design. Only the best components are found inside our FA-ST Hard Drive. A full one year warranty and ICD's uncompromised reputation for quality should say it all.

Now, don't let the abundance of features scare you . . . FA-ST Hard Drives are available in all sizes *and* at prices you can afford.

So, to be quite honest, we really don't care what you call our hard drive –

as long as you call for it today. And get ready for the best thing that ever happened to your Atari ST.

Call or write for our free catalog today.

1220 Rock Street • Rockford, Illinois 61101 • (815) 968-2228 • MODEM: (815) 968-2229 • FAX: (815) 968-6888

CIRCLE #101 ON READER SERVICE CARD.

ICD

FA-ST is a trademark of ICD Corporation.

Atari ST is a trademark of Atari Corporation.