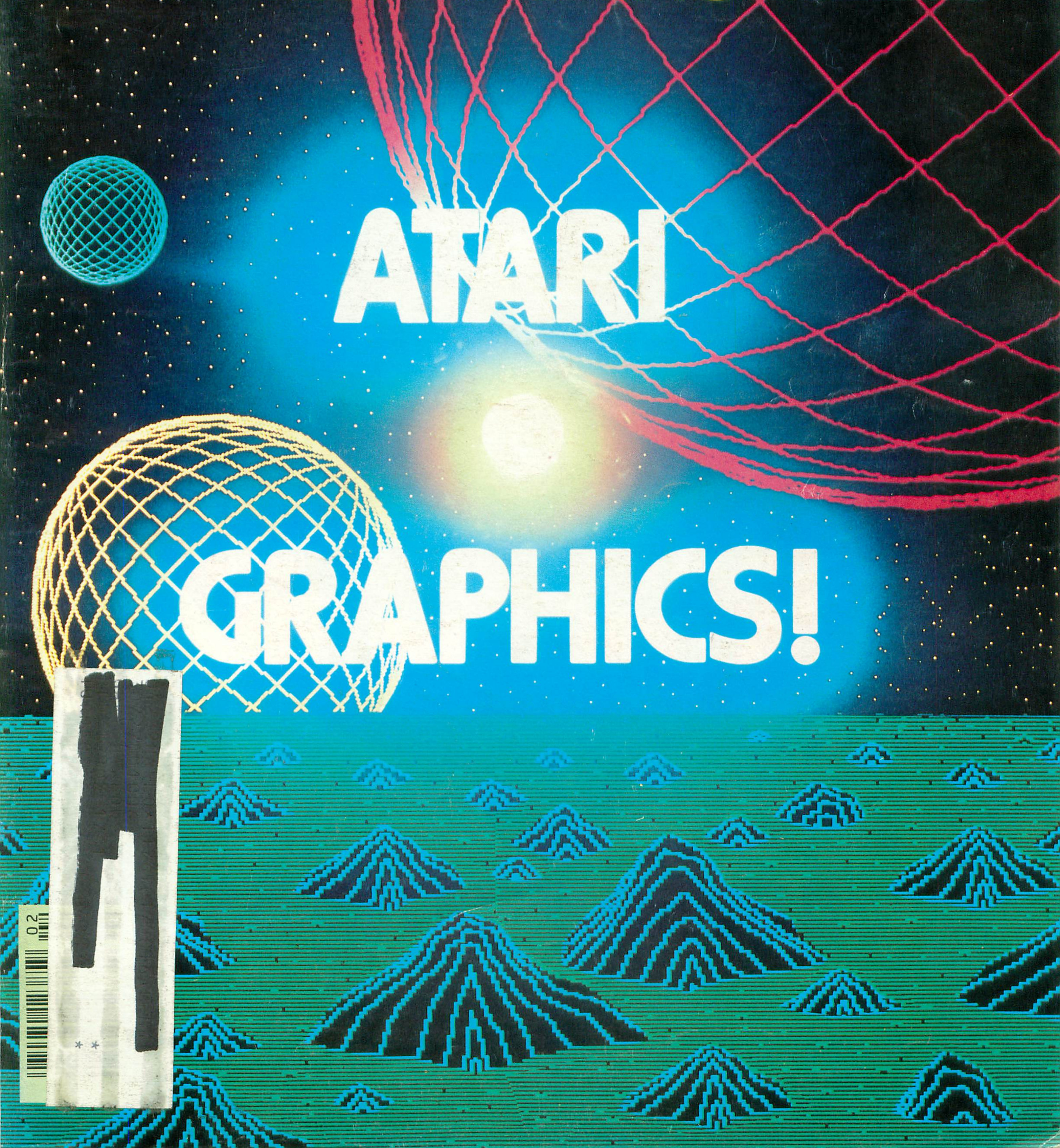
NO. 16 FEBRUARY 1984 $3.00

THE MAGAZINE FOR ATARI® COMPUTER OWNERS

A.N.A.L.O.G.™ COMPUTING



REMEMBER.

ELEPHANT™ NEVER FORGETS.

A full line of top-quality floppies, in virtually every 5 1/4" and 8" model, for compatibility with virtually every computer on the market. Guaranteed to meet or exceed every industry standard, certified 100% error-free and problem-free, and to maintain its quality for at least 12 million passes (or over a lifetime of heavy-duty use).

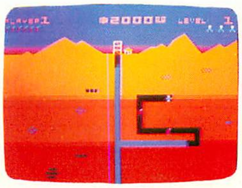
Contact Dennison Computer Supplies, Inc., 55 Providence Highway, Norwood, MA 02062 or call toll-free 1-800-343-8413. In Massachusetts, call collect (617) 769-8150. Telex 951-624.

Dennison

TURN YOUR HOME INTO THE HOTTEST ARCADE IN TOWN

O'RILEY'S MINE™

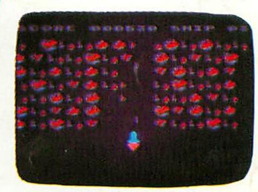
You're a mad Irishman digging your way through an abandoned mine filled with oil, coal, gold, rubies, diamonds—and hungry creatures. You'll need the luck of the Irish to survive, but with so much at stake, it's a chance you're willing to take.



O'RILEY'S MINE™ available for Atari®, Apple® and Commodore 64®

MOON SHUTTLE™

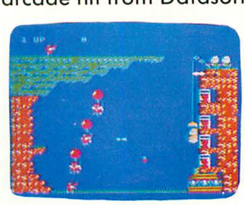
Your screen explodes with life-threatening action as you pilot your Moon Shuttle through outer space in this home version of the Nichibutsu® favorite. Awaiting you is The Prince of Darkness and his forces, which mysteriously multiply!



MOONSHUTTLE™ available for Atari®, Radio Shack Color®, Commodore 64®, coming soon for Apple®

POOYAN™

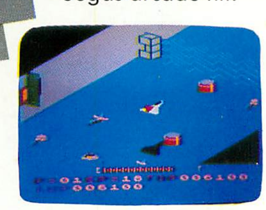
One of the biggest arcade game hits from Konami is a game of fast action in life-and-death encounters. You battle a pack of hungry wolves eager to catch your defenseless piglets. You'll need quick reflexes and a good arm in the new arcade hit from Datasoft.



POOYAN™ available for Atari®, Radio Shack Color®, Apple®, coming soon for Commodore 64®

ZAXXON™

Zaxxon has all the unique color graphics, super sound and unprecedented 3-dimensional effects that made it the star of the arcades. Maneuver your ship through the state-of-the-art defenses of the floating fortress to come face-to-face with the deadly Zaxxon Robot. This is the official home version of the Segas arcade hit.



ZAXXON™ available for Atari®, Apple®, Radio Shack Color®

Our reputation as one of the premier game manufacturers assures you of the highest quality attainable in action, strategy and graphics. For the hottest titles in entertainment, keep your eye on the leader—DATASOFT.

You'll Find The Best In Home Computer Software from Datasoft®

Pooyan™ is a trademark of Konami Industries Company, Ltd.

O'Riley's Mine® and Datasoft® are registered trademarks of Datasoft Inc.

Moon Shuttle® is a registered trademark of Nichibutsu, USA.

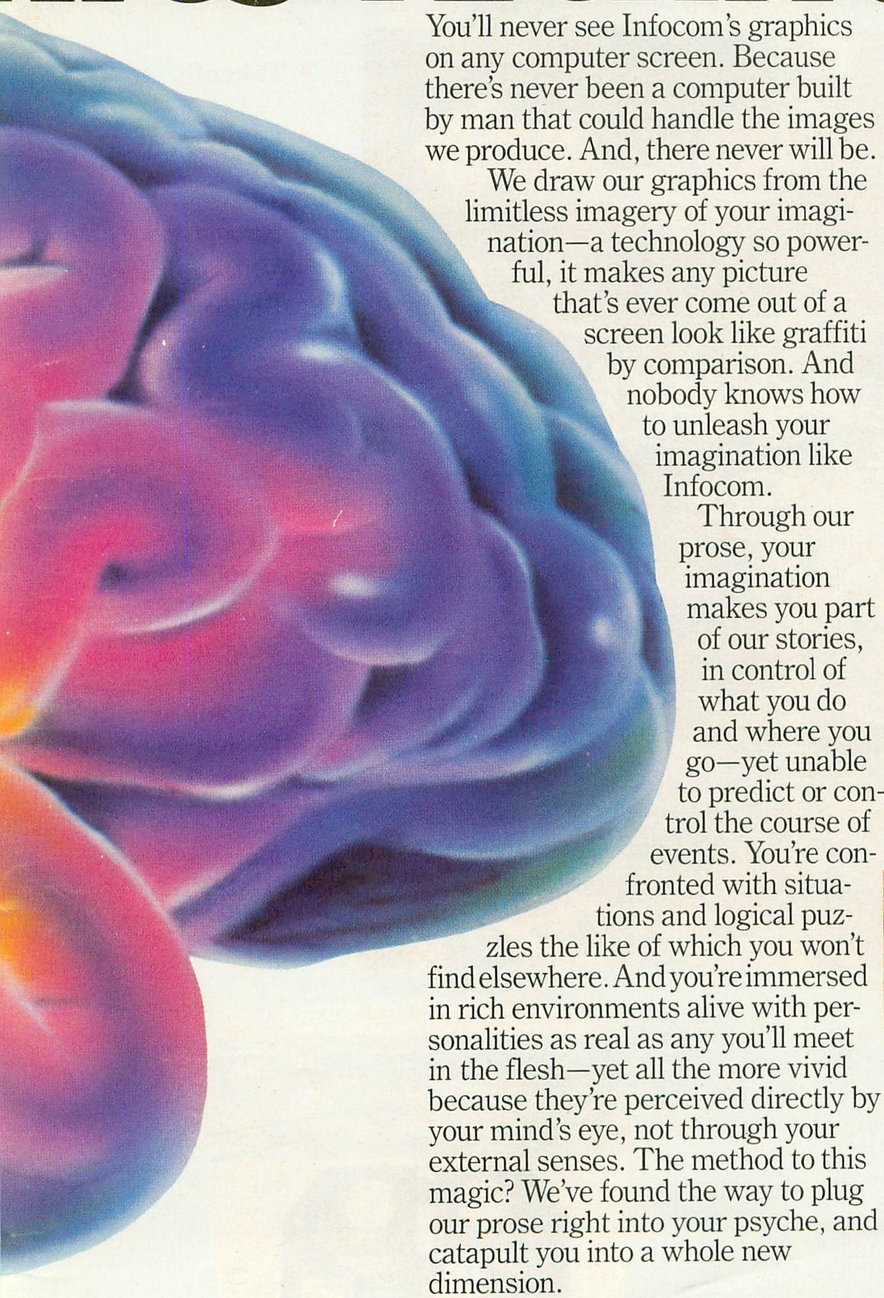
ZAXXON™ and Sega® are trademarks of Sega Enterprises, Inc.

9421 Winnetka Avenue, Chatsworth, CA 91311 (213) 701-5161

CIRCLE 150 ON READER SERVICE CARD

WE UNLEASH TH
POWERFUL GRAP

E WORLD'S MOST HICS TECHNOLOGY.



You'll never see Infocom's graphics on any computer screen. Because there's never been a computer built by man that could handle the images we produce. And, there never will be.

We draw our graphics from the limitless imagery of your imagination—a technology so powerful, it makes any picture that's ever come out of a screen look like graffiti by comparison. And nobody knows how to unleash your imagination like Infocom.

Through our prose, your imagination makes you part of our stories, in control of what you do and where you go—yet unable to predict or control the course of events. You're confronted with situations and logical puzzles the like of which you won't find elsewhere. And you're immersed in rich environments alive with personalities as real as any you'll meet in the flesh—yet all the more vivid because they're perceived directly by your mind's eye, not through your external senses. The method to this magic? We've found the way to plug our prose right into your psyche, and catapult you into a whole new dimension.

Take some tough critics' words about our words. SOFTALK, for example, called ZORK® III's prose "far more graphic than any depiction yet achieved by an adventure with graphics." And the NEW YORK TIMES saw fit to print that our DEADLINE™ is "an amazing feat of programming." Even a journal as video-oriented as ELECTRONIC GAMES found Infocom prose to be such an eye-opener, they named one of our games their Best Adventure of 1983.

Better still, bring an Infocom game home with you. Discover firsthand why thousands upon thousands of discriminating game players keep turning everything we write into instantaneous bestsellers.

Step up to Infocom. All words. No graffiti. The secret reaches of your mind are beckoning. A whole new dimension is in there waiting for you.

(For more information on Infocom games contact: Infocom, Inc., P.O. Box 855, Garden City, NY 11530.)

INFOCOM™

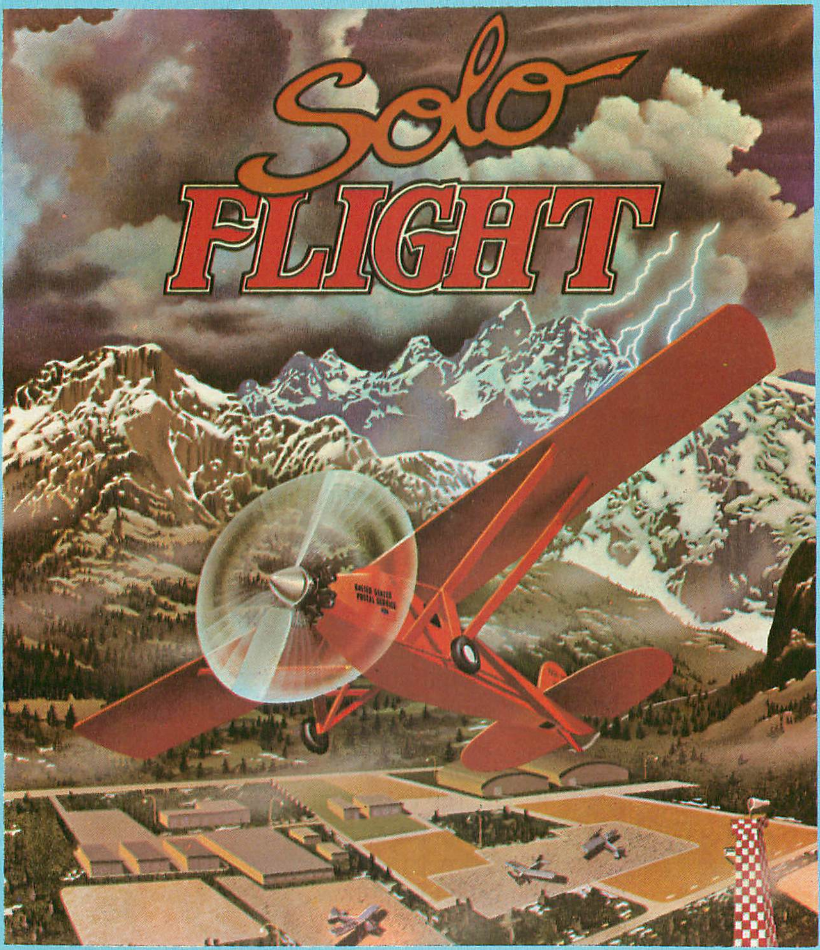
The next dimension.

For your: Apple II, Atari, Commodore 64, CP/M 8," DEC Rainbow, DEC RT-11, IBM, MS-DOS 2.0, NEC APC, NEC PC-8000, Osborne, TI Professional, TI 99/4A, TRS-80 Model I, TRS-80 Model III.

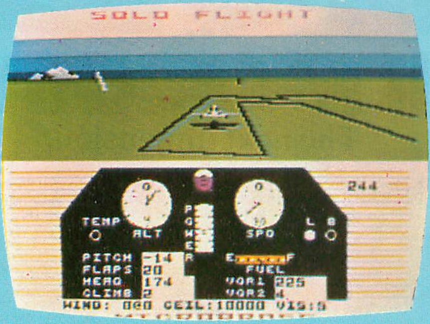
AT LAST!!! A REAL FLIGHT SIMULATOR!!!
— From The Creators of HELLCAT ACE!

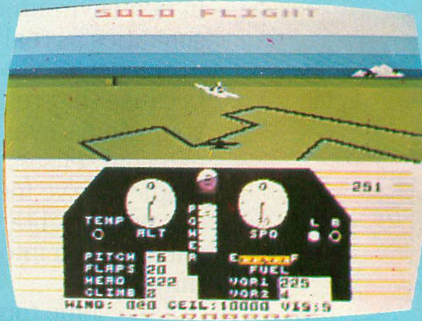
A real flight simulator allows you the pilot to accomplish Takeoffs, Landings, Cross-Country Navigation, and Emergency Procedure practice under Clear (VFR), Cloudy (IFR), and Cross-Wind conditions. SOLO FLIGHT is such an advanced simulation that uses realistic three dimensional terrain graphics, actual configuration instrument panel, multiple air navigation maps, and accurate aircraft performance characteristics to provide the challenge, thrill, excitement, and joy of the flight experience!!!

- Three-Dimensional Terrain Graphics
- Multiple Airports and Runways
- Dual Radio Navigation Instruments
- Full Feature, Actual Configuration Instrument Panel
- Multiple Air Navigation Charts
- Instrument Approaches



- Takeoffs
- *Landings*
- Cross-Country Navigation — Day, VFR and IFR Flying
- Emergency Procedures Practice
- Air Mail Delivery Game for 1-4 Players
- Smooth Landing & Good Navigation Scoring
- For All Atari and Commodore 64 computers, disk or cassette . . . $34.95
- Coming Soon for IBM-PC





Look for MicroProse's two new exciting real time combat simulations — MIG ALLEY ACE & NATO COMMANDER. Write or call for our Free Catalog.

If you cannot find our games at your local store, you can order by MasterCard or VISA, Money Order, COD or Check. Add $2.50 for Postage and Handling. Maryland Residents add 5% Sales Tax.

MicroProse Software

DEALER INQUIRIES INVITED.

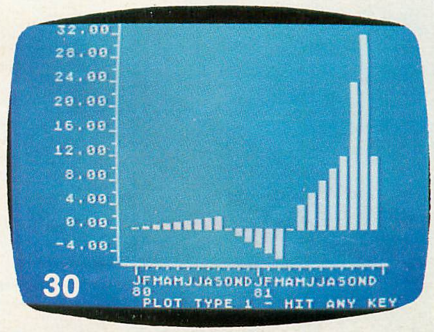
10616 Beaver Dam Road, Hunt Valley, MD 21030
(301) 667-1151

NO. 16 FEBRUARY 1984

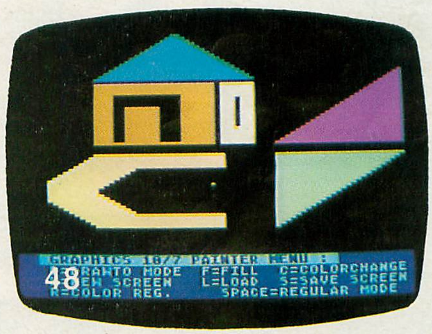
THE MAGAZINE FOR ATARI® COMPUTER OWNERS

A.N.A.L.O.G. COMPUTING™

FEATURE ARTICLES

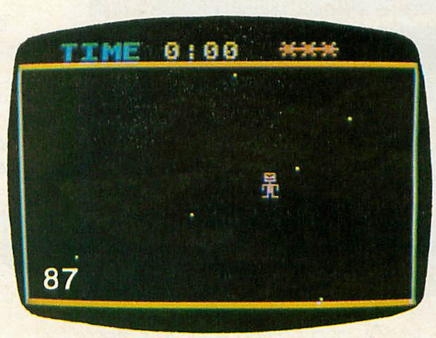


PRODUCT REVIEWS

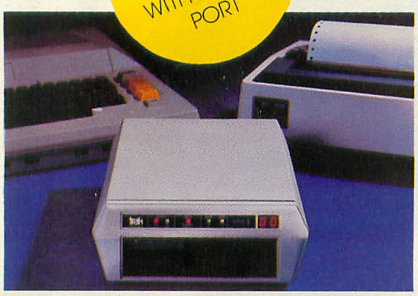


PROGRAM LISTINGS

COLUMNS



AVAILABLE NOW! WITH PRINTER PORT



TRAK'S floppy and hard disk drive systems for the Apple and IBM-PC are available at many Computerland and other fine stores worldwide.

. . .WITH IT'S FLEET OF THE MOST ADVANCED SINGLE AND DOUBLE DENSITY DRIVE SYSTEMS IN THE UNIVERSE - TOTALLY COMPATIBLE WITH ATARI* COMPUTER MODELS 400, 800, 600XL, 1200XL, 1400XL, AND 1450XL.

DOCK YOUR FAVORITE PRINTER DIRECTLY TO OUR DRIVE'S BUILT-IN PARALLEL PRINTER INTERFACE. THERE'S NO NEED TO BUY AN EXPENSIVE ATARI 850 MODULE.

THE COMMAND CONTROL PANEL FEATURES READ, WRITE AND DENSITY INDICATORS, A TRACK COUNTER TO LOCATE EVERY BIT OF YOUR DATA AND A TOUCH-SENSITIVE SWITCH TO PROTECT IT.

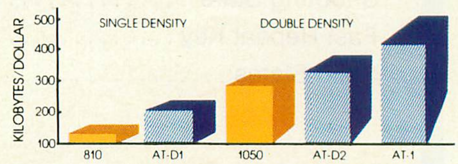
WARP SPEED OF UP TO 18 TIMES FASTER THAN **ANY** OTHER ATARI DRIVE IS ACHIEVED WITH OUR OPTIONAL TURBO-CHARGED SOFTWARE.

AWESOME POWER — DESTROY A PLANET OR LEARN HOW ONE IS CREATED. BALANCE YOUR CHECKBOOK OR WRITE A BESTSELLER. LEARN A NEW LANGUAGE OR CREATE ONE.

IT'S MAGNIFICENTLY SIMPLE! — JUST PLUG IT IN AND LIFT-OFF!

*Atari is a registered trademark of Atari, Inc.

TRAK GIVES YOU THE MOST STORAGE FOR YOUR DOLLAR!



OUR FLAGSHIP, AT-D2, WITH TWICE THE STORAGE CAPACITY OF THE ATARI 810 DRIVE PLUS A PRINTER INTERFACE AND COMMAND CONTROL PANEL, HAS A VALUE OF OVER $1000 YET COSTS LESS THAN $500!

SEE YOUR DEALER NOW FOR A TEST FLIGHT, OR

call toll free: **1-800-323-4853**
in Illinois, call collect: 1-312-968-1716
TWX 910-222-1848

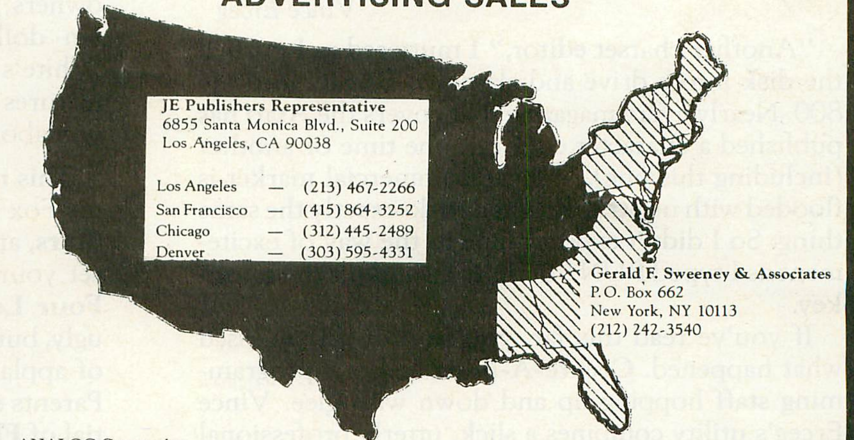
trak™ A commitment to quality

microcomputer corporation 1511 Ogden Ave., Downers Grove, IL 60515

ABOUT THE COVER

No, it's not a real Atari graphics display, but a combination of computer printouts and special photographic effects. The landscape and mountains were drawn using Datasoft's **Micropainter** program. The planets and moons were generated from a sphere-drawing demo originally written by Tom Hudson. Screen dumps of the landscape, mountains and spheres were made on a C.Itoh 8510-P printer. These were cut apart and composited on white posterboard. A large negative was then shot of the composition. As with the covers of Issues 7 and 12, acetate was used to add color to the various geometric shapes. The negative was then placed on a sheet of glass and back-lit. Multiple exposures were used to "burn in" various light effects within the shot. The final result is a cover that could fool the eye of an Atari programmer: what graphics mode did they use to get that? □

ADVERTISING SALES



ANALOG Computing
Home Office
Michael DesChenes
National Advertising
(617) 892-9230

Address all advertising materials to:
Michael DesChenes — Advertising Production
ANALOG Computing
565 Main Street, Cherry Valley, MA 01611

ANALOG COMPUTING (ISSN 0744-9917) is published monthly for $28 per year by **ANALOG** Magazine Corp., 565 Main Street, Cherry Valley, MA 01611, Tel. (617) 892-3488. Second-class postage paid at Worcester, MA and additional mailing offices. POSTMASTER: Send address changes to **ANALOG COMPUTING**, P.O. Box 615, Holmes, PA 19043. No portion of this magazine may be reproduced in any form without written permission of the publisher. Program listings should be provided in printed form. Articles should be furnished as typed copy in upper and lower case with double spacing. By submitting articles to **ANALOG COMPUTING**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ANALOG**. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply self-addressed, stamped envelope. U.S.A. Newstand distribution by Eastern News Distributors, Inc., 111 Eighth Ave., New York, NY 10011.

EDITORIAL

by Brian Moriarty

I was sifting through a pile of mail a few months ago, when I came across an intriguingly thick shipping envelope. It contained a draft for an article, a disk and the following cover letter:

Dear Sir:

Enclosed please find my submission for possible inclusion in your publication. It is a character set editor/tutorial that, quite frankly, rivals commercially available ones.

I have also enclosed a stamped, self-addressed envelope so that the diskette may be returned.

Sincerely,
Vince Erceg

"Another charset editor," I muttered as I pushed the disk into a drive and slammed BASIC into my 800. Nearly every magazine that covers the Atari has published a character editor at one time or another (including this one), and the commercial market is flooded with utilities designed to do exactly the same thing. So I didn't expect much in the way of excitement as I typed RUN "D:FONT" and hit the return key.

If you've read this far, you've probably guessed what happened. **Create-A-Font** had our programming staff hopping up and down with glee. Vince Erceg's utility combines a slick, utterly professional design with shameless visual gimmicks to make an otherwise boring chore fast and entertaining. If you do anything with redefined character sets, take the time to type in **Create-A-Font**. It's one of the nicest character editors ever created for the Atari. Need I add that Mr. Erceg's return envelope found its way into the Round File.

We decided to make **Create-A-Font** the centerpiece of an entire issue devoted to Atari computer graphics. Tom Hudson's **Solid States** is a 3D plotting package that lets you define any type of solid object and "look" at it from any perspective. **10/7 Painter** is a follow-up to Peter Budgell's **Extra Graphics Mode** article from Issue 15; it's a GTIA painting program that lets you create and save your own high-res Rembrandts on either disk or cassette.

Tom Newdome's **Bar Chart Subroutine** offers business programmers a convenient way to display statistics without a lot of head-scratching. Sally Forth presents **FPLOT**, an impressively fast routine for 2-color plotting. And Kyle Peacock wraps up his four-part **Fine Scrolling** tutorial with an *all-purpose* utility you can install in your BASIC programs to achieve impressive **Eastern Front**-type visual effects.

We didn't skimp on our regular utilities this month, either. If you've been going crazy trying to find an 850 Interface Module for your Christmas printer, look over Paul Swanson's **Low Cost Printer Interface** on page 36. It lets you hook up your printer using just *two* joystick ports (XL computer owners, take note!) for a total cash outlay of less than ten dollars. XL users will also appreciate Jerry White's **XLDEMO**, which highlights a couple of features your skimpy XL owner's manual didn't tell you about.

This month's games department introduces Dennis Fox to the pages of **ANALOG** with **Shooting Stars**, an assembly-language endurance test that will set your teeth on edge. Joel Gluck insists that his **Four Letter Words** listing on page 42 is big and ugly, but that didn't stop it from earning him a round of applause at a recent Boston user group meeting. Parents should not overlook the educational potential of **FLW**.

Demand for reprints of our **C:CHECK** and **D:CHECK2** articles has been so high that we decided to publish them again in this issue. If you're a new **ANALOG** reader, drop whatever you're doing and type in one or both of these programs immediately! It'll save you hours of frustration when debugging the other BASIC programs appearing in this and future issues.

Looking for a demo to silence that neighbor who won't stop bragging about his new Commodore 64? Sit the poor sucker down in front of Craig Patchett's **Stars 3D**. Show him how short the program is. Then tell him all about display list interrupts and color indirection, the two features that make Atari's hardware second to none when it comes to fancy graphics. You may also wish to remind him that his jaw is still hanging open.

Hope you enjoy **ANALOG**'s Atari Graphics Issue. □

READER COMMENT

Thank you so much for going monthly with your publication. I seem to find problems in other magazines that I don't find in yours, such as 1) pages of corrections in future issues, 2) characters that often don't resemble what I'm putting on the screen, 3) the frustration of different characters taking up different sizes of space, and 4) no method of checking programs for typing errors. It is really great to know that the best will be coming out to shine twice as much as it was before!

Also, I'd like to say I agree with Neil Weinstock's letter in issue #14 concerning the lifespan of the Atari 1200XL, and those of us who got a rather raw deal by buying one. A little bit of good (and fair) business on their part would be to give the owners of 1200XL's a chance to upgrade to the newest line now out, and — speaking for myself — I would waste no time in buying a 1450XLD if the company behind it could demonstrate to me that it sincerely wishes me at least a couple of years before it discontinues the product I bought.

Jim Slocum
St. Petersburg, Florida

Hexpad Update.

I have been very pleased with the quality of your writers and of the contributed software. However, I have two observations.

First, the HEXPAD program (issue #14) is quite useful and *tripled* my input rate. I am a touch typist of long standing, and re-learning the replacement of the letter hex codes was something I decided not to tolerate. The letters "a" through "f" are all reachable by the left hand and do not interfere with the redefinition of right hand keys for the digits "0" through "6." The following new Line 31013 will allow touch typists to retain at least the familiar left hand position for letters. Change Line 31013 to

```
31013 DATA 32,109,106,107,
108,117,105,111,97,98,99,1
00,101,102,48,48,49,50,51,
52,53,54,65,66,67,68,69,70
```

The second point is minor, but still is an annoyance. Each time you publish a game in assembler, the BASIC code used to build the disk image uses the common name "D:AUTORUN.SYS." This approach denies rapid access to DOS option "A" (directory listing) and assumes that each game will reside on a separate diskette. To overcome these "deficiencies," I modify the filename to reference "D:gamename.L", where "gamename" is some descriptive name. The ".L" reminds my disk users to use DOS option "L" to "run" the game.

Walter D. Lazear
Fairfax, VA

5200 Article Update.
(ANALOG #15)

Newer releases of the 5200 incorporate some minor hardware changes. Controller ports 3 and 4 have been eliminated, making POT4 through POT7, TRIG2, TRIG3, and bit 1 of CONSOL useless. A few of the connector pins have been redefined. Pin 2 of the I/O expansion connector now carries POKEY's Audio Out signal. Three pins on the cartridge connector have changed to accomodate the new 2600 adapter. The system clock, 02, is output on pin 14, isolated through a diode. An alternate video input is taken from pin 24 and is also isolated through a diode. Pin 30 provides an alternate audio input.

There is space on the newer boards for circuitry for a PAL (European TV standard) version of the 5200. Also, on power-up, the monitor program checks for the PAL version by examining the GTIA register PAL after step 2 of the initialization routine. It also checks the cartridge program for PAL compatibility. The byte at $BFE7 should read $02 if compatible, or $00 if not. This is the only important change to the monitor program. There are some additional hardware changes, but none affects the machine's operation from the programmer's view.

Claus Buchholz
Greenwich, CT

Since I tried to write some really great programs with the information contained in **Inside Atari Basic** and didn't get far, I started to look around for some help. I was pleased to find **ANALOG** at my local book store. In it I found **Space Assault** and **Observational Astronomy**. I also found tons of adds, reviews, reports, and the helpful hints and information I need. Not only did I have a learning tool, but I wound up with some great software for next to nothing.

I sat down and typed in **Space Assault** in about four hours, debugged it (I only missed two numbers in a DATA line), and called my resident game experts (Tracy, 12 and Billy, 10). They played the game and reported that the gun did not move fast enough. I fooled

around for awhile (read days) and finally found a cure. Once I got the gun going faster, it seemed only right that the game should get harder as it went.

These are the changes I made.

```
1165 IF SC>10000 THEN GOSU
B 1222
1168 IF SC>20000 THEN GOSU
B 1230
1169 IF SC>30000 THEN GOSU
B 1236
1222 IF DIR=1 THEN XSHIP=X
SHIP+3.5
1224 IF DIR=2 THEN XSHIP=X
SHIP-3.5
1225 RETURN
1230 IF DIR=1 THEN XSHIP=X
SHIP+4
1232 IF DIR=2 THEN XSHIP=X
SHIP-4
1235 RETURN
1236 IF DIR=1 THEN XSHIP=X
SHIP+5
1237 IF DIR=2 THEN XSHIP=X
SHIP-5
1238 RETURN
```

I added to line 1720:

```
1720 X=X+XPOS(ST)*2:Y=Y+YP
OS(ST)*2
```

I have one last question. I'd really like to buy some fantasy and/or adventure games, but they all seem to be on disks. Now I know disks are better and faster than my 1010 recorder, but I am not going to be buying a disk drive any time soon, if ever. I'm lucky I got the computer! And it seems to me, with interest picking up and prices going down on personal computers, that more people like me will be in the market for products that don't cost a small fortune just to run. Where can I find some games on tape that do something more than blast aliens?

Sincerely,
Bill Moore
Wilmington, DE

Thanks for the kind words. It happens that there are several reasonably good text adventures available on tape, including the entire Scott Adams Adventure Series from Adventure International. ANALOG's own ***Adventures in the 5th Dimension*** *(Issue #11) was specifically written with cassette users in mind; and we'll be publishing more tape-compatible adventures in the near future. Stay tuned!*

—B.M.

I would like to put in a word about Kyle Peacock's review of **Starbowl Football** in Issue #13. After reading his review, I immediately went out and bought the game. However, I was all prepared to meet an incredible computer team. Instead, I found that the computer gave me the option of playing a fair "college" team, or an outstanding "pro" team. Of course, for starters, I chose the college team. I beat the team by three points on my very first try. Is this a new option? If not, I still hope that you publish this letter to inform my fellow game enthusiasts that, in **Starbowl Football**, they have the option of getting smeared (as Mr. Peacock was), or having a very close game against an *average* computer opponent.

Chris Johnson
Alexandria, VA

Send letters to:
Reader Comment
P.O. Box 23
Worcester, MA 01603

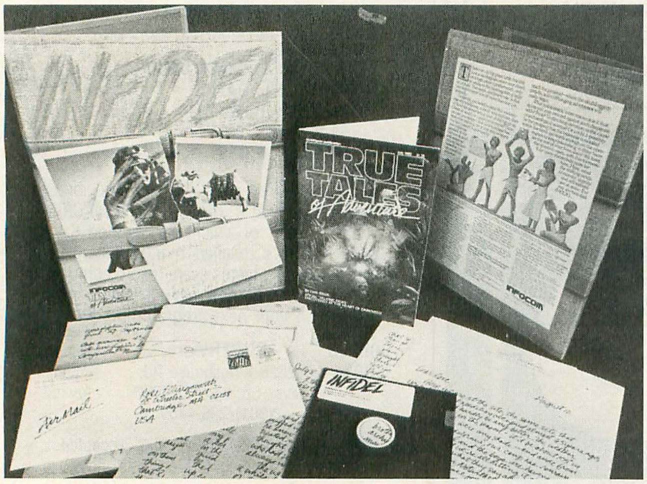
NEW PRODUCTS

by Lee Pappas

Hurtling into the year of Big Brother, wave after wave of new Atari software is trying to rid you of what's left of your holiday spending money. My taking over this column is like a typical Nor'easter, except I'm buried in a blizzard of software along with the snow.

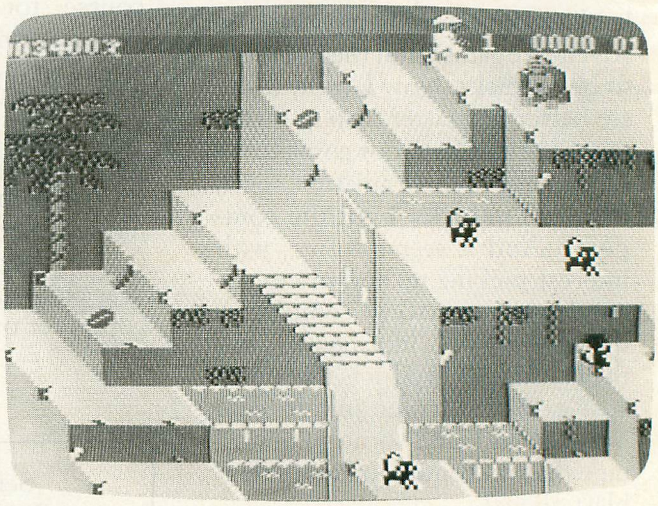
INFOCOM continues to uphold their name on the top-10 countdown with **Infidel**, **The Witness**, **Enchanter** and **Planetfall**. The latter is worth the price for its packaging alone; it looks like a mission briefing kit from Robert Heinlein's *Starship Troopers*. All four adventures are superbly put together and continue Infocom's fine non-graphics adventure tradition.

SIERRA ON-LINE is now shipping an Atari version of **Ultima I**, the super-popular Apple adventure game. **BC's Quest for Tires** is an arcade-style game, hopefully intended for the kiddies. On-Line's hottest hit in the **ANALOG** offices, **Oil's Well** (now on ROM as well as disk) turns you into "Pac Man on a stick" with added complications. (See Pat Kelley's review in this issue.) **Homeword** is On-Line's soon-to-be-released entry into the Atari word processor market. With its special graphics and joystick capabilities, this $49.95 program is touted as being especially easy to learn and operate.



Infidel.

A new company to the Atari market, STARFIRE GAMES, is releasing **Time Machine 1** and **Global Thermonuclear War**. In **Time Machine**, you pilot the fourth mission of a USAF/NASA experimental vehicle in search of the first three test pilots, who vanished into the forth dimension (you could say they had the right stuff at the wrong time!). **War** not only involves you and your Atari in World War III, but forces you to unscramble launching codes and to retarget the ICBMs, along with making retaliation decisions. Both games are on 48K disks.



Congo Bongo.

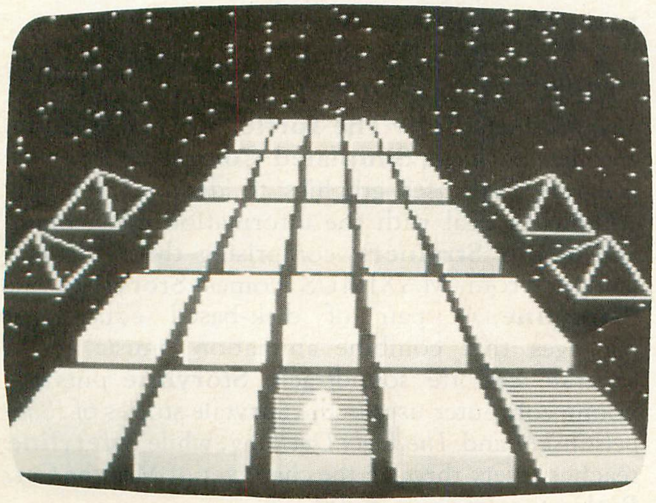
ACTIVISION is now shipping **River Raid** and **Kaboom!**, and IMAGIC's **Moonsweeper** is also available, along with **Fathom**, a graphics adventure. SIRIUS SOFTWARE has recently released **Gruds In Space**, a much better-than-average graphics adventure. And Bill Hogue's BIG FIVE SOFTWARE is ready to release **Scraper Caper**, a greatly expanded sequel to the adventures of **Miner 2049er**'s Bounty Bob.

SEGA is following up their **Star Trek Simulator** with a home version of the arcade hit **Congo Bongo**. This cartridge-based game lacks most of the screen-levels of its coin-op counterpart (only two scrolling screens). On level 1, you must avoid the green coco-

nuts hurtled by a whacked-out gorilla at the top of a series of "cliffs." As you hop over waterfalls and avoid monkeys gone bananas, you make your way to screen number 2. This is **Frogger** version one-hundred forty-seven, except now it's 3-D.

BRODERBUND is gearing up to ship **Lode Runner** on disk, and **Spare Change** on disk and ROM. With more than 150 screens, **Lode Runner** has you wandering through passages avoiding meanies and picking up treasures. An extension of the game allows you to design your own screens and play them. **Spare Change** is a game with a different twist (almost a bizarre one). As an arcade owner, you must do whatever it is an arcade owner does and do it well. In **Spare Change**, this happens to be preventing the escape of your most popular arcade game characters. They "come to life," figure it's time to split, and try to escape. It's up to you to stop'em. These two programs come after the release of **Drol**, one of those rare games that comes along and appears on **ANALOG**'s lunchtime monitors in force. Be sure to read our review on page 115.

New from EPYX is a cartridge update of **Temple of Apshai** called **Gateway to Apshai**, reviewed on page 34. Other new Epyx releases include **Pitstop**, **Gunfight** and **Seawolf**. The latter two are licensed from Bally/Midway from their coin-op versions. A preview disk with demonstrations of these four games and **Jumpman Jr**. is available for only $2.50 from Epyx. **Silicon Warrior** is a ROM cartridge for one to four players, resembling a Tic-Tac-Toe game with shooting, shields, black holes and robots. Two early video games from the arcade now translated to the Atari computer are **Starfire**, a primitive version of a **Star Raiders**-type game (remember the "Exidy" bonus ship?), and **Fire One**, one of the first sub/torpedo games. These come two games on a cassette or disk, as do **Gunfight** and **Seawolf**.

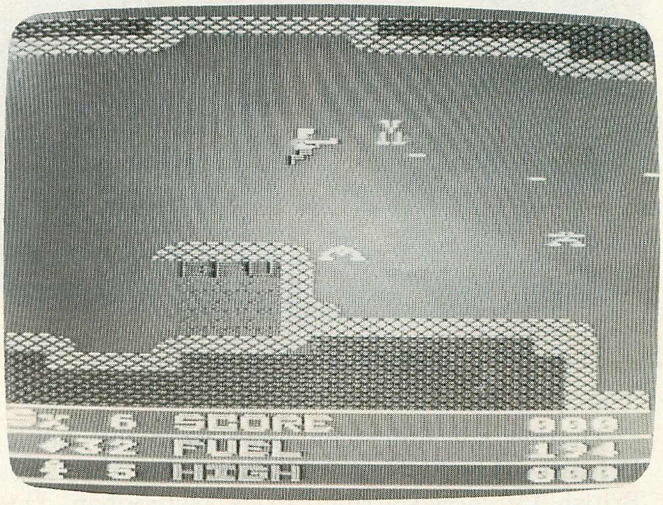


Silicon Warrior.

ADVENTURE INTERNATIONAL has a new game in release, **AREX**. This machine language program by John (Rally Speedway) Anderson pits you against enemies which must be neutralized as you progress through a maze of dungeons or an open arena. This game is available on disk or cassette.

Alley-Oops is a 16K program with 8 levels of play. A takeoff on the good ol' bowling computer games, but here you'll have to be on the lookout for beer bottles, discarded gum on the floor, and other "dangerous" obstacles. It's from ARTWORX.

From DATAMOST comes **Mr. Robot And His Robot Factory**, **Cohen's Tower**, **The Tail of Beta Lyrae**, **Cosmic Tunnels** and **Monster Smash**. These five arcade games should be out by the time you read this. INHOME SOFTWARE is shipping **Captain Beeble**, a 16K cartridge that is definitely challenging and fun (though I would have named it "Captain System Reset"). Multiple levels with explosion-inducing surprises will keep you on the edge of screaming as you avoid all of those close calls.



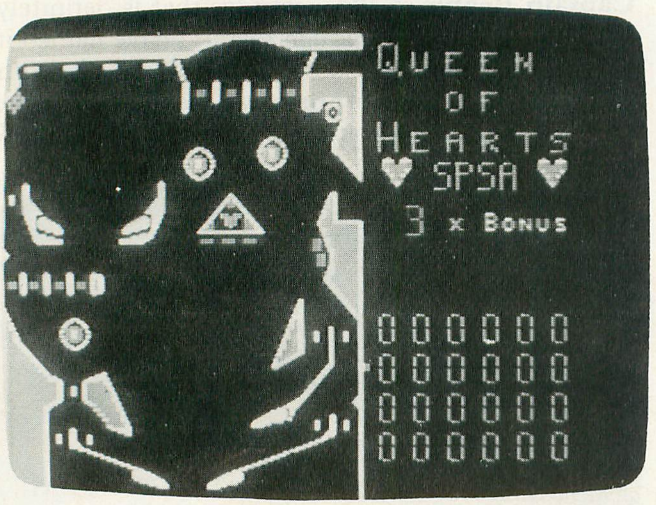
Captain Beeble.

Droids is one of several new software releases from TG SOFTWARE. **Nightshift**, **Abracadabra** and **Ozzy's Orchard** wrap up their lineup of four arcade/action games. TG also markets a joystick for left or right handed players. SCREENPLAY's **Asylum** is a 3-D scrolling graphic adventure made popular on the IBM PC, and now available for Atari computers. It retails for $29.95 and runs in 48K. This company also has a **Q*bert** clone, **Pogo Joe** (only now you're hopping on cylinders, not cubes). Another arcade conversion, **Blueprint** from CBS ELECTRONICS, is also based on a Bally/Midway game.

AVALON HILL's **TAC** (Tactical Armor Command) is a 48K disk where you command squads of infantry and tank groups. **Dino Eggs** by MICRO-FUN is a fast action game where you are transported back to prehistoric time. Your goal is to scale to the

top of cliffs containing dinosaur eggs, and bring them back to the 21st century. **Death In The Caribbean**, also by Microfun, is an adventure featuring high-resolution graphics.

If it's sadistic games you crave, then try THORN EMI's **Orc Attack**. In this cartridge you protect your castle from the Orc hoards scaling the walls on ladders. As you dodge their crossbow bolts and daggers, you discover that nice heavy rocks dropped on their heads will punch their ticket. Later you'll find that boiling oil will cook their goose, and should these medieval hoods get too close, it's headchopping time using the handy-dandy sword provided for you. Other surprises make **Orc Attack** a different kind of video game.



Queen Of Hearts.

Another adventure converted from the Apple, **Prisoner II** by EDU-WARE, is based on the 1960s British TV show starring Patrick McGoohan. This challenging product is reviewed on page 62. **Flight Simulator II** follows up SUBLOGIC's **Night Mission Pinball** with an excellent representation of piloting a Piper Cherokee Archer II.

Queen Of Hearts is a 48K disk-based pinball simulation from SSI's RAPIDFIRE division. This is one of the few pinball games that has a "tilt" feature and the ability to keep track of up to four players. DATASOFT is following up **Pooyan** with **Dallas Quest**, based on the TV series *Dallas*, and 3G COMPANY has a program designed to assist in predicting the outcome of horse racing.

Adding to the growing list of practical application software titles for the Atari is **The Home Accountant**. CONTINENTAL SOFTWARE's home finance package can handle upwards of 200 categories, 5 separate checking accounts, and has multiple print functions.

The Atari Program Exchange is now offering Chris Crawford's **Excalibur**, an adventure/role playing scenario in which you are King Arthur, the ruler of Camelot. Though loaded with music, graphics and animation, the real star of this program is the aspect of humanity — morals and chivalry play an important part in the game play. Retail cost is $29.95 and requires 48K, disk drive, and a joystick. Also released recently from APX is **Atspeller**, a spelling checker for the **AtariWriter** Word Processing cartridge. This program requires a 32K disk system and lists at $39.95. A list in the programs "dictionary" contains 30,000 words, to check against your saved text files.

Translator Disk: a new hope.

For those of you who purchased a 600XL, 800XL, or 1200XL and have had trouble running software, this disk (available from Atari) should end your problems. When booted up, The Translator Disk "flips away" the operating system in your XL Computer and replaces it with that of the 400/800 series computers. This rids the computer of any incompatibility problems, but remember — you have to boot this disk up every time a new program is to be used.

New programs to give you the smarts.

Educational software is moving into the marketplace at the same rate games have. UNICORN SOFTWARE has four titles covering the full span of ages (1 through adult): **Funbunch, Ships Ahoy, Ten Little Robots** and **Race Car 'Rithmetic**. KOALA TECHNOLOGIES has several learning programs designed for their KoalaPad touch tablet that are oriented towards the younger set. **Coloring Book, Spider Eater** and **Spellicopter** will utilize games to teach their subjects in an entertaining manner which appeals to children. DESIGNWARE, the developer of **Spellicopter**, also markets **Math Maze** and **Creature Creator** for the Atari. Both educational games feature graphics to add excitement to the learning.

A new company to the educational market, CAROUSEL SOFTWARE, has introduced three new products: **Telly The Turtle**, an expanded turtle graphics package; **Simulated Computer II**, a nice program that uses graphics to demonstrate how computers deal with the information given them; and **Brain Strainers**, comprising three separate games. From MAXIMUS comes **Storyline** and **Safetyline**, a pair of disk-based educational packages that combine animation, music and a separate cassette soundtrack. **Storyline** puts the young computer user in the fairytale stories of *Rumpelstiltskin* and *The Ugly Duckling*, while **Safetyline** teaches safety through the character of Max the Cat. Both programs run in 48K.

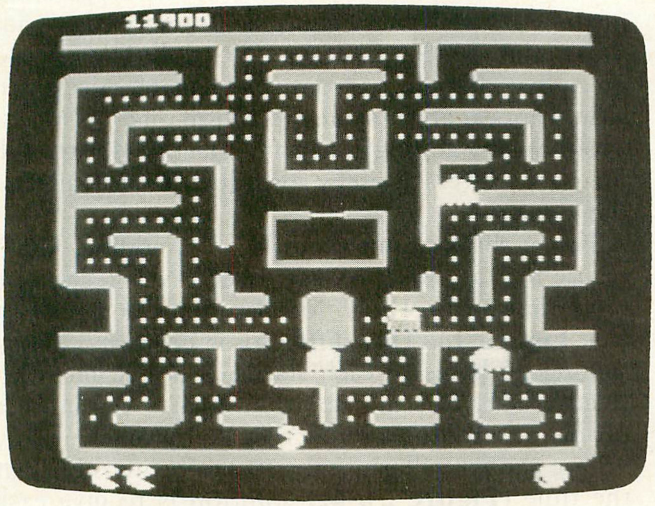
Adding to their ever-growing list of programs, DORSETT EDUCATIONAL SYSTEMS now has

over 60 courses covering a wide range of topics including reading and comprehension, electronics, accounting, business and a broad variety of math skills. JAY GEE is now shipping **Attack Of The Spelling Bees**, an arcade-style learning game where you guide your bee over letters, and "shoot" them to certain parts of the screen to complete words. This follows their previously released **Devils Dare**, a computerized version of the old Japanese "board" game, Go-Moku.

Just in time to compete with the troubled Coleco Adam comes Atari's "Programming System" All-In-One Pak. This package includes a 16K 600XL Computer, 1010 Program Recorder, five cassettes comprising the **Invitation to Programming** series (with appropriate workbooks), and two books: *Inside Atari BASIC* and *101 Programming Tips & Tricks* (with its own cassette).

The "Writing System" All-In-One Pak is made up of the 600XL, 1027 Letter Quality Printer (with stationery), the **AtariWriter** word processing cartridge, and the book *One Way to Better Writing.*

Finally, there is the "Entertainment System" All-In-One Pak, including **Donkey Kong**, the newly released **Ms. Pacman** cartridge, two joysticks and (not surprisingly) the 600XL. Also included is a booklet, *Inside Secrets* (tricks to improve your game scores) and a cartridge storage case. Prices are $379.95 for the Programmer, $599.95 for the Writer and $299.95 for the Entertainer.



Ms. Pac-Man.

Shipping (finally!) is Atari's Communicator II Package. This consists of the 835 Direct Connect Modem, **Telelink II** cartridge, and several free hours on The Source, Dow Jones and Compuserve networks. Telelink II features fine scrolling, a display buffer, autodialing from the keyboard, and can store and transmit (at the touch of a key) your log-on sequence, including password. Look for Charles Bachand's review of this system in an upcoming issue of **ANALOG**. List price is $279.95. ☐

STARS 3-D

16K Cassette or Disk

by Craig Patchett

EDITOR'S NOTE: The article accompanying this demonstration is intended primarily for advanced readers. The demo program itself, however, can be typed in and enjoyed by anybody.

Stars 3D illustrates a simple 3-dimensional graphics effect, produced by moving similar objects horizontally at different speeds. It uses a couple of assembly-language "tricks" to achieve the illusion of depth as efficiently as possible.

Listing 1 is an ATARI BASIC program that POKEs the **Stars 3D** routine into memory and activates it with a USR call. **Listing 2** is the assembly-language source code, provided for programmers who want to see how the routine works.

After entering **Listing 1** and D:CHECKing it, SAVE the demo program to disk or tape and type RUN <return>. A few seconds later your screen will be filled with what appears to be 192 stars in several planes of motion. Believe it or not, the entire starfield is actually composed of only eight individual stars!

How it's done.

The dramatic display in **Stars 3D** is accomplished by using multiple LMS (Load Memory Scan) instructions in the display list. Every DL instruction has its LMS option bit set. This allows me to have the same section of memory appear in more than one place on the screen.

When I set up the display list, I randomly pick one of the eight star memory segments and store its address in an LMS instruction. Normally, this would mean that all the stars from a given memory segment would appear at the same x-position on the screen. I avoid this by adding a random offset (0-48) to each memory address, thereby spreading the stars all over the x-range.

If you think about the offset I'm adding, you'll realize that an offset of, say, twenty will result in a mode line with twenty "orphan bytes" on the end (see **Figure 1**). This difficulty is corrected by assigning 96 bytes to each star memory segment instead of 48. But this solution creates yet another problem: scrolling a star across 96 bytes when only 48 are shown will cause the star to be invisible half the time (**Figure 2**). My solution is to give each memory segment two stars, separated by 48 bytes. Whenever one star moves off the edge of the screen, the other star will appear at the opposite side (**Figure 3**).

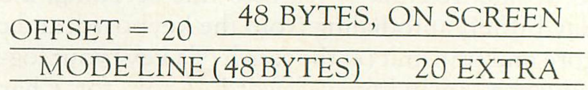


Figure 1.

48 BYTES ON SCREEN

The star is off screen and will not appear on again until it reaches the right edge of the screen area. It must scroll 48 bytes to do this.

Figure 2.

48 BYTES ON SCREEN

a)

The star on the left scrolls off the screen.

48 BYTES ON SCREEN

b)

The star on the right scrolls on.

Figure 3.

Eight luminances.

Each DL instruction in **Stars 3D** has its display list interrupt (DLI) option bit set. A small DLI service routine allows me to put a different star color on every mode line. Instead of going for a broad spectrum of colors, I decided to make each star a different shade of the same color, with slow-moving stars set to darker luminances than the fast movers. This further enhances the illusion of depth.

No fine scrolling.

Fine scrolling would have required handling each of the 192 mode lines separately. This takes a lot of processing time. Instead, I chose to manipulate each of the 16 stars (two in each memory segment) directly, using the 6502 ASL instruction. When a star is shifted out of one byte it falls into the next, and when it shifts out of its memory segment it is stuck onto the other end. Each memory segment has its own timer which determines how quickly the stars in that segment will appear to move.

Those are the basics behind **Stars 3D**. You should be able to find any details I skipped over by studying my source code. I hope this demo will show you how an unconventional approach can lead to terrific savings in both processing time and memory (the entire program requires less than 2K, including the screen RAM), both of which are critical in today's high-performance graphics programs. □

Listing 1.

```
100 REM ************************
110 REM *     STARS 3D DEMO    *
120 REM *  BY CRAIG PATCHETT   *
130 REM * ANALOG COMPUTING #16 *
140 REM ************************
150 REM
160 START=14336:REM * $3800 HEX
170 FOR I=START TO START+377
180 READ BYTE:POKE I,BYTE:NEXT I
190 X=USR(START)
200 DATA 32,49,56,32,145,56,169,0,141,
200,2,169,7,162,56,160,38,32,92,228,16
9,253,141,0,2
210 DATA 169,56,141,1,2,169,192,141,14
,212,76,35,56,32,13,57,169,0,141,173,6
3,76,98,228,169
220 DATA 189,141,189,62,169,59,141,205
,62,162,0,142,221,62,232,189,188,62,24
,105,48,157,189,62,189
230 DATA 204,62,105,0,157,205,62,169,0
,157,221,62,232,224,16,208,229,169,189
,133,204,169,59,133,205
240 DATA 160,255,162,3,169,0,145,204,1
36,192,255,208,249,202,240,5,230,205,7
6,106,56,162,0,189,189
250 DATA 62,133,204,189,205,62,133,205
,169,64,160,0,145,204,232,224,16,208,2
35,96,169,119,133,204,169
260 DATA 57,133,205,169,0,141,173,63,1
60,3,169,206,145,204,200,208,2,230,205
,173,10,210,41,7,72
270 DATA 170,189,111,57,174,173,63,157
,237,62,104,238,173,63,10,170,173,10,2
10,41,63,201,48,176,247
280 DATA 24,125,189,62,145,204,200,208
,2,230,205,189,205,62,105,0,145,204,20
0,208,2,230,205,192,67
290 DATA 208,189,169,65,145,204,200,16
9,119,145,204,141,48,2,200,169,57,145,
204,141,49,2,169,35,141
300 DATA 47,2,96,174,173,63,189,237,62
,141,10,212,141,22,208,238,173,63,64,1
62,13,32,50,57,32
310 DATA 50,57,162,12,32,50,57,32,50,5
7,202,48,17,222,87,57,208,248,32,50,57
,189,99,57,157
320 DATA 87,57,76,29,57,96,189,189,62,
133,204,189,205,62,133,205,188,221,62,
177,204,10,10,145,204
330 DATA 144,15,169,1,136,192,255,208,
2,160,47,145,204,152,157,221,62,96,8,8
,6,6,4,4,3
340 DATA 3,2,2,1,1,8,8,6,6,4,4,3,3,2,2
,1,1,34,36,38,40,42,44,46,34,112,112,2
40
```

CHECKSUM DATA

(See pp. 20-24.)

```
100 DATA 90,594,809,30,102,89,35,33,51
0,627,171,627,89,203,37,4046
250 DATA 355,705,269,975,307,668,425,3
18,819,18,4859
```

Listing 2.

```
0100 ; **********************
0110 ; *    STARS 3D DEMO   *
0120 ; *  BY CRAIG PATCHETT *
0130 ; **********************
0140 ;
0150 ; System equates
0160 ;
0170 SDMCTL = $022F
0180 NMIEN = $D40E
0190 RANDOM = $D20A
0200 WSYNC = $D40A
0210 SDLSTL = $0230
0220 VDSLST = $0200
0230 COLPF0 = $D016
0240 COLOR4 = $02C8
0250 SETVBV = $E45C
0260 XITVBL = $E462
0270 ;
0280 ; Zero-page equate
```

```
0290 ;
0300 INDRCT = $CC     ; for indirect addressing
0310 ;
0320     *=  $3800
0330 ;
0340 ;  Get things going
0350 ;
0360 INITIL
0370     JSR STRINI  ; set up stars
0380     JSR DLSINI  ; set up display list
0390     LDA #0      ; set background color
0400     STA COLOR4
0410     LDA #7      ; set up VBLANK
0420     LDX #VBLANK/256
0430     LDY #VBLANK&255
0440     JSR SETVBV
0450     LDA #DLI&255 ; get DLIs going
0460     STA VDSLST
0470     LDA #DLI/256
0480     STA VDSLST+1
0490     LDA #192
0500     STA NMIEN
0510 ALLDON
0520     JMP ALLDON  ; let things run
0530 ;
0540 ; VBLANK routine
0550 ;
0560 VBLANK
0570     JSR CNTDWN  ; take care of star movement
0580     LDA #0      ; reset index
0590     STA INDEX
0600     JMP XITVBL  ; back to system
0610 ;
0620 ; Initialize stars
0630 ;
0640 STRINI
0650     LDA #STRLIN&255 ; set up STRTPL/H arrays
0660     STA STRTPL
0670     LDA #STRLIN/256
0680     STA STRTPH
0690     LDX #0
0700     STX STRPOS
0710     INX
0720 STRBR1
0730     LDA STRTPL-1,X ; make each address a screen
0740     CLC         ;   width more than the one
0750     ADC #48     ;   before
0760     STA STRTPL,X
0770     LDA STRTPH-1,X
0780     ADC #0
0790     STA STRTPH,X
0800     LDA #0
0810     STA STRPOS,X
0820     INX
0830     CPX #16
0840     BNE STRBR1
0850     LDA #STRLIN&255 ; clear star memory
0860     STA INDRCT
0870     LDA #STRLIN/256
0880     STA INDRCT+1
0890     LDY #255
0900     LDX #3
0910     LDA #0
0920 STRBR2
0930     STA (INDRCT),Y
0940     DEY
0950     CPY #255
0960     BNE STRBR2
0970     DEX
0980     BEQ STRBR3
0990     INC INDRCT+1
1000     JMP STRBR2
1010 STRBR3
1020     LDX #0      ; give each line a star
1030 STRBR4
1040     LDA STRTPL,X
1050     STA INDRCT
1060     LDA STRTPH,X
1070     STA INDRCT+1
1080     LDA #64
1090     LDY #0
1100     STA (INDRCT),Y
1110     INX
1120     CPX #16
1130     BNE STRBR4
1140     RTS
1150 ;
1160 ; Initialize display list
1170 ;
1180 DLSINI
1190     LDA #DLIST&255 ; set up for indirect
1200     STA INDRCT  ;   addressing
1210     LDA #DLIST/256
1220     STA INDRCT+1
1230     LDA #0      ; get index ready
1240     STA INDEX
1250     LDY #3
1260 DLSBR4
1270     LDA #$CE    ; ANTIC 14, DLI, LMS line
1280     STA (INDRCT),Y
1290     INY
1300     BNE DLSBR1
1310     INC INDRCT+1
1320 DLSBR1
1330     LDA RANDOM  ; pick star type
1340     AND #7
1350     PHA
1360     TAX
1370     LDA MANCOL,X ; tell STRCOL what color
1380     LDX INDEX   ; this line is
1390     STA STRCOL,X
1400     PLA
1410     INC INDEX
1420     ASL A       ; times two so we skip over
1430     TAX         ; two screen widths...
1440 DLSBR2
1450     LDA RANDOM  ; pick random offset into
1460     AND #63     ;   line
1470     CMP #48     ; make sure it's less than
1480     BCS DLSBR2  ;   a screen width
1490     CLC
1500     ADC STRTPL,X ; ...here instead of one
1510     STA (INDRCT),Y ; put it into display list
1520     INY
1530     BNE DLSBR5
1540     INC INDRCT+1
1550 DLSBR5
1560     LDA STRTPH,X
1570     ADC #0
1580     STA (INDRCT),Y
1590     INY
1600     BNE DLSBR6
1610     INC INDRCT+1
1620 DLSBR6
1630     CPY #67     ; 192 lines done, finish up
1640     BNE DLSBR4
1650     LDA #$41
1660     STA (INDRCT),Y
1670     INY
1680     LDA #DLIST&255
1690     STA (INDRCT),Y
1700     STA SDLSTL
1710     INY
1720     LDA #DLIST/256
1730     STA (INDRCT),Y
1740     STA SDLSTL+1
1750     LDA #$23    ; give us a wide screen
1760     STA SDMCTL
1770     RTS
1780 ;
1790 ; Display list interrupt routine
1800 ;
1810 DLI
1820     LDX INDEX   ; what line are we on?
```

```
1830      LDA STRCOL,X ; load this line's color
1840      STA WSYNC    ; wait for end of last line
1850      STA COLPF0   ; store color
1860      INC INDEX    ; get ready for next line
1870      RTI
1880 ;
1890 ; Timer routine
1900 ;
1910 CNTDWN
1920      LDX #13      ; move fastest star
1930      JSR SCROLL
1940      JSR SCROLL
1950      LDX #12      ; and its twin
1960      JSR SCROLL
1970      JSR SCROLL
1980 CNTBR1
1990      DEX          ; for the rest...
2000      BMI CNTRET
2010      DEC TIMER,X ; ...countdown timer
2020      BNE CNTBR1
2030      JSR SCROLL   ; scroll if ready
2040      LDA TIMARY,X ; and reset timer
2050      STA TIMER,X
2060      JMP CNTBR1
2070 CNTRET
2080      RTS
2090 ;
2100 ; Star scroll routine
2110 ;
2120 SCROLL
2130      LDA STRTPL,X ; set up for indirect addressing
2140      STA INDRCT
2150      LDA STRTPH,X
2160      STA INDRCT+1
2170      LDY STRPOS,X ; get star position
2180      LDA (INDRCT),Y ; get byte with star in it
2190      ASL A        ; shift star left one
2200      ASL A
2210      STA (INDRCT),Y ; put it back
2220      BCC SCRBR1   ; did it fall off byte?
2230      LDA #1       ; if so, put it in next one
2240      DEY
2250      CPY #255     ; did it fall off screen?
2260      BNE SCRBR2
2270      LDY #47      ; if so, put it on other end
2280 SCRBR2
2290      STA (INDRCT),Y
2300      TYA
2310      STA STRPOS,X ; remember where it is now
2320 SCRBR1
2330      RTS
2340 ;
2350 TIMER
2360      .BYTE 8,8,6,6,4,4 ; timers for scrolling
2370      .BYTE 3,3,2,2,1,1
2380 TIMARY
2390      .BYTE 8,8,6,6,4,4 ; values to reset timers
2400      .BYTE 3,3,2,2,1,1
2410 MANCOL
2420      .BYTE $22,$24,$26 ; star colors
2430      .BYTE $28,$2A,$2C
2440      .BYTE $2E,$22
2450 ;
2460 DLIST
2470      .BYTE $70,$70,$F0 ; display list
2480      *=  *+579
2490 STRLIN
2500      *=  *+768    ; star (screen) memory
2510 STRTPL
2520      *=  *+16     ; addresses of beginning of
2530 STRTPH
2540      *=  *+16     ; each star line
2550 STRPOS
2560      *=  *+16     ; position of star on line
2570 STRCOL
2580      *=  *+192    ; color of each line
2590 INDEX
2600      *=  *+1      ; used to index into STRCOL
```

D:CHECK2

16K Disk

by Istvan Mohos and Tom Hudson

When typing programs into your computer from **ANALOG Computing**, there is always a chance of making a mistake. **D:CHECK2** will help you find such errors very easily. Type in the accompanying program and SAVE it. Follow the instructions below to check **D:CHECK2** as you would any other program.

CHECKing your typing.

1. Type in the program listing desired from the magazine. Visually check it for obvious errors (missing lines, etc.)
2. LIST the program to be checked to disk. Use the command:

```
LIST "D:PROGNAME"
```

3. LOAD **D:CHECK2** and RUN it.
4. **D:CHECK2** will ask for a filename. Respond:

```
D:PROGNAME
```

and press RETURN.

5. **D:CHECK2** will ask for an issue number. Just type the issue number and press RETURN.
6. **D:CHECK2** will execute. The screen will go black in order to speed up the program.
7. When D:CHECK2 finishes, it will display final instructions. At this time you should type NEW and press RETURN.
8. When **D:CHECK2** executed, it created a BASIC file on disk called BUG. ENTER it into your computer with the command:

```
ENTER "D:BUG"
```

This file should match the "CHECKSUM DATA" printed after the program listing you are checking. The following example shows how to check for errors.

Magazine checksum data.

```
10 DATA 34,455,234,22,55,38,93,45,114,
285,633,442,453,23,31,2957
160 DATA 82,94,64,73,347,199,287,84,15
6,368,59,40,98,9,342,2302
310 DATA 65,356,101,25,547
```

D:CHECK2 output.

```
10 DATA 34,455,234,22,55,38,244,45,114
,285,633,442,453,23,31,3108
160 DATA 82,94,64,73,347,199,287,84,15
6,368,59,40,98,9,342,2302
310 DATA 65,101,34,200
```

Each line of the program being checked has its own checksum value. If any characters in the line are incorrect, the checksum value will be different from the corresponding value in the magazine. The checksum data is set up so that there are 15 checksum values in each line, with the 16th value containing the total of the checksums.

The line number of the checksum line tells which line number is first in the checksum group. In the example above, the first line checked in the first checksum line is 10, and its checksum is 34. The first line checked in the second checksum line is 160, and its checksum is 82. The first line checked in the third checksum line is 310, and its checksum is 65.

Let's assume the CHECKSUM DATA above was listed in the magazine, and you typed in the program and checked it with **D:CHECK2**.

The first thing to do would be to look at the total of the values in the first line. This value should be 2957, as shown in the magazine CHECKSUM DATA. However, in the results in the BUG file, the total is 3108. This means that there is an error in the 15 checksum values in this line. Comparing the magazine checksums to the BUG checksums, we find that the seventh checksum is 244 in the BUG data, and should be 93. This means that there is an error in the seventh line of the program. Note the error and continue checking. The rest of the line is correct, so we go on to the second line.

Now we check the total of the second line of checksum data. The total of 2302 in our BUG file matches the total in the magazine, so we can go on to the third checksum line.

The third checksum line is different from the others in that it only checks four lines. This is because it is at the end of the program, and the program did not have an even multiple of 15 lines. The line is checked the same as the others. As you can see, the total of the line should be 547, but is only 200 in the BUG file. Looking at the BUG file, you will notice that there is one less checksum value (the 356 in the magazine checksum data). This means that the first line in the program after line 310 is missing. The last checksum in this line is also incorrect. It is a 34 and should be 25. This means that the third line after line 310 in the program is incorrect.

To summarize, there were 3 errors in the program we checked. Two errors were caused by mistakes in the lines, and a third appeared because a whole line was missing.

Once you have noted all errors, type NEW and press RETURN. This erases the D:CHECK2 program. Next, bring the program being checked into memory by typing:

```
ENTER "D:PROGNAME"
```

If the program had errors, correct the lines in error. If there were no errors, the program is correct and ready to run. □

```
10 REM CHECK DEBUGGING AID
   BY  ISTVAN MOHOS
20 REM VERSION 2 MODS BY TOM HUDSON
30 GRAPHICS 0:? :? "This run will LIST
 data statements     with the name: BUG
, to the disk."
40 ? :? "The BUG DATA is created by ev
aluating each character of a user prog
ram,      LISTed to disk.":?
50 DIM FI$(15)
60 CLOSE #1:? "ENTER FILENAME";:INPUT
FI$
70 PIK=PEEK(559):Z=0:REM constants
80 ? :? "ENTER ISSUE NUMBER";:TRAP 80:
INPUT ISSUE
90 TRAP 60:OPEN #1,4,0,FI$
100 ON X GOTO 180,280
110 ? "K":? "DISABLING SCREEN...STAND
BY...":FOR I=1 TO 800:NEXT I:POKE 559,
Z:REM debug before poking
120 LINECOUNT=Z:DIM I$(126)
130 TRAP 150:INPUT #1;I$:LINECOUNT=LIN
ECOUNT+1
140 GOTO 130
150 CLOSE #1:Q=INT(LINECOUNT/15):DIM C
(LINECOUNT),R(Q),S$(5):IF (LINECOUNT=Z
 OR I$="") THEN 530
160 IF ASC(I$(1,1))<48 OR ASC(I$(1,1))
>57 THEN 530
170 X=1:GOTO 90
180 RANGE=Z:LINE=Z:FOR I=1 TO 5:S$(I,I
)=" ":NEXT I
190 COUNT=Z
200 INPUT #1;I$:T=1:COUNT=COUNT+1
210 IF I$(T,T)<>" " THEN S$(T,T)=I$(T,
T):T=T+1:GOTO 210
220 LINE=VAL(S$)
230 R(RANGE)=LINE:RANGE=RANGE+1
240 TRAP 270:INPUT #1;I$
250 COUNT=COUNT+1:IF COUNT=15 THEN 190
260 GOTO 240
270 CLOSE #1:X=2:GOTO 90
280 FOR I=1 TO LINECOUNT:CHECKSUM=Z
290 GET #1,NUMBER:PRODUCT=X*NUMBER:CHE
CKSUM=CHECKSUM+PRODUCT:X=X+1:IF X=4 TH
EN X=1
300 IF NUMBER=155 THEN 320
310 GOTO 290
320 CHECKSUM=CHECKSUM-1000*INT(CHECKSU
M/1000):C(I)=CHECKSUM:IF ISSUE>9 THEN
X=2
330 NEXT I
340 CLOSE #1:OPEN #1,8,0,"D:BUG":LINE=
R(Z):ITEM=Z
350 COUNT=15:TOTAL=Z:IF LINECOUNT<15 T
HEN COUNT=LINECOUNT
360 PRINT #1;LINE;" DATA ";
370 FOR I=1 TO COUNT:DATUM=C(15*ITEM+I
):PRINT #1;DATUM;",";:TOTAL=TOTAL+DATU
M:NEXT I
380 PRINT #1;TOTAL
390 ITEM=ITEM+1:LINECOUNT=LINECOUNT-15
:IF LINECOUNT<1 THEN 420
400 LINE=R(ITEM)
410 GOTO 350
420 CLOSE #1:POKE 559,PIK
430 ? "KTo check BUG data against pri
nted     data statements, type NEW. Th
en type:"
440 ? "ENTER ";CHR$(34);"D:BUGRETURN .
                   Type LIST after the
READY prompt."
450 ? :? "The line number of each data
 statementcoincides with the first lin
e of the"
460 ? "user program which the data sta
tement evaluates."
470 ? "Numbers within each data statem
ent     represent consecutive lines of
the     user program."
480 ? "The last number is the total."
490 ? :? "Check the last number of eac
h state-  ment against the printed ver
sion;"
500 ? "only in case of a discrepancy c
heck    each number in the data stateme
nt."
510 ? "Make note of the lines containi
ng the bugs. Then ENTER ";CHR$(34);"D:
yourprogRETURN"
520 ? "to make the corrections.":END
530 POKE 559,PIK:? "K":? "Your typed-
in program was not properlyLISTed to d
isk."
540 ? :? "Please LIST your program to
disk, thenRUN ";CHR$(34);"D:CHECK";CHR
$(34);" again.":CLR :END
```

•

CHECKSUM DATA

```
10 DATA 44,815,767,524,686,389,806,850
,86,721,921,593,591,704,974,9471
160 DATA 482,125,389,696,567,797,442,5
61,230,89,717,216,943,541,299,7094
310 DATA 719,711,741,427,244,435,288,5
84,553,441,711,499,803,322,515,7993
460 DATA 246,684,406,232,123,700,480,7
74,500,4145
```

•

(For Atari cassette owners who wish to confirm the accuracy of their programs, see C:CHECK*, next page.)*

C:CHECK

16K Cassette

by Istvan Mohos and Tom Hudson

When typing programs into your computer from **ANALOG Computing**, there is always a chance of making a mistake. **C:CHECK** will help you find such errors very easily. Type in the accompanying program and SAVE it. Follow the instructions below to check **C:CHECK** as you would any other program.

CHECKing your typing.

1. Type in the program listing from the magazine. Visually check it for obvious errors (missing lines, etc.).
2. LIST the program to be checked to cassette. Use the command:

   LIST "C:"

3. LOAD **C:CHECK** and RUN it.
4. **C:CHECK** will ask you if you want the output to go to the screen or printer. Type S for screen or P for printer and press RETURN.
5. **C:CHECK** will ask for an issue number. Just type the issue number and press RETURN.
6. Position the tape to the beginning of the program to be checked and press PLAY on the program recorder. Press RETURN.
7. **C:CHECK** will begin reading the program from tape and generate a checksum table. This data should match the "CHECKSUM DATA" printed after the program listing you are checking. The following example shows how to check for errors.

Magazine checksum data.

```
10 DATA 34,455,234,22,55,38,93,45,114,
285,633,442,453,23,31,2957
160 DATA 82,94,64,73,347,199,287,84,15
6,368,59,40,98,9,342,2302
310 DATA 65,356,101,25,547
```

C:CHECK output.

```
10 DATA 34,455,234,22,55,38,244,45,114
,285,633,442,453,23,31,3108
160 DATA 82,94,64,73,347,199,287,84,15
6,368,59,40,98,9,342,2302
310 DATA 65,101,34,200
```

Each line of the program being checked has its own checksum value. If any characters in the line are incorrect, the checksum value will be different from the corresponding value in the magazine. The checksum data is set up so that there are 15 checksum values in each line with the 16th value containing the total of the checksums.

The line number of the checksum line tells which line number is first in the checksum group. In the example above, the first line checked in the checksum line is 10, and its checksum is 34. The first line checked in the second checksum line is 160, and its checksum is 82. The first line checked in the third checksum line is 310, and its checksum is 65.

Let's assume the CHECKSUM DATA above was listed in the magazine, and you typed in the program and checked it with C:CHECK.

The first thing to do would be to look at the total of the values in the first line. This value should be 2957, as shown in the magazine CHECKSUM DATA. However, in the results in the C:CHECK output, the total is 3108. This means that there is an error in the 15 checksum values in this line. Comparing the magazine checksums to the C:CHECK output, we find that the seventh checksum is 244 in the C:CHECK data, and should be 93. This means that there is an error in the seventh line of the program. Note the error and continue checking. The rest of the line is correct, so we go on to the second line.

Now we check the total of the second line of checksum data. The total of 2302 in our C:CHECK data matches the total in the magazine, so we can go on to the third checksum line.

The third checksum line is different from the others in that it only checks four lines. This is because it is at the end of the program, and the program did not have an even multiple of 15 lines. The line is checked the same as the others. As you can see, the total of the line should be 547, but is only 200 in the C:CHECK data. Looking at the

C:CHECK output, you will notice that there is one less checksum value (the 356 in the magazine checksum data). This means that the first line in the program after line 310 is missing. The last checksum in this line is also incorrect. It is a 34 and should be 25. This means that the third line after line 310 in the program is incorrect.

To summarize, there were 3 errors in the program we checked. Two errors were caused by mistakes in the lines, and a third appeared because a whole line was missing.

Once you have noted all errors, type NEW and press RETURN. This erases the C:CHECK program. Next, bring the program being checked into memory by positioning the tape and typing:

ENTER "C:"

If the program had errors, correct the lines in error. If there were no errors, the program is correct and ready to run. □

Basic listing.

```
100 REM CHECK DEBUGGING AID
    BY  ISTVAN MOHOS
110 REM VERSION 2 MODS AND CASSETTE
120 REM VERSION BY TOM HUDSON
130 GRAPHICS 0:? :? "This run will LIS
T data statements     to the screen or
printer."
140 ? :? "This DATA is created by eval
uating     each character of a user pro
gram,      LISTed to tape.":?
150 DIM OUT$(1),I$(128),CR$(1)
160 ? "OUTPUT TO SCREEN OR PRINTER";:I
NPUT OUT$:IF OUT$<>"S" AND OUT$<>"P" T
HEN 160
170 IF OUT$="S" THEN OPEN #2,8,0,"E:":
GOTO 200
180 CLOSE #2:? "READY PRINTER AND PRE
SS RETURN";:INPUT CR$
190 TRAP 180:OPEN #2,8,0,"P:"
200 ? :? "ENTER ISSUE NUMBER";:TRAP 20
0:INPUT ISSUE
210 ? :? "READY TAPE AND PRESS RETURN"
;:OPEN #1,4,0,"C:":? :?
220 Z=0:LINECOUNT=Z:PLIN=Z:X=2
230 TRAP 340:INPUT #1,I$:LINECOUNT=LIN
ECOUNT+1:LINUM=VAL(I$(1,5))
240 NLCK=NLCK+1:IF NLCK>1 AND NLCK<16
THEN 290
250 IF LINECOUNT=1 THEN 280
260 ? #2;TOTAL:NLCK=1
270 IF OUT$="S" THEN PLIN=PLIN+1:IF PL
IN=10 THEN ? "PRESS RETURN TO CONTINUE
";:INPUT CR$:PLIN=0
280 TOTAL=Z:? #2;LINUM;" DATA ";
290 CHKSUM=Z:IF ISSUE>9 THEN X=2
300 FOR I=1 TO LEN(I$):PRODUCT=X*ASC(I
$(I,I)):CHKSUM=CHKSUM+PRODUCT:X=X+1:IF
 X=4 THEN X=1
310 NEXT I:CHKSUM=CHKSUM+X*155:X=X+1:I
F X=4 THEN X=1
320 CHKSUM=CHKSUM-1000*INT(CHKSUM/1000
)
330 ? #2;CHKSUM;",";:TOTAL=TOTAL+CHKSU
M:GOTO 230
340 CLOSE #1:IF LINECOUNT=Z THEN 370
350 ? #2;TOTAL
360 CLOSE #2:END
370 ? "K":? "Your typed-in program wa
s not properlyLISTed to tape."
380 ? :? "Please LIST your program to
tape, thenRUN ";CHR$(34);"CHECK";CHR$(
34);" again.":CLOSE #2:CLR :END
```

CHECKSUM DATA

```
100 DATA 198,759,11,135,191,594,198,80
6,763,467,931,100,465,572,107,6297
250 DATA 764,922,11,168,375,783,304,25
9,534,890,875,136,732,361,7114
```

GRIFFIN'S LAIR

EDUCATIONAL PROGRAMS REVIEW

by Braden E. Griffin, M.D.

I find myself in a quandary. An abundance of excellent educational software is sitting by my computer, but because of limited magazine space and time, much of it cannot be reviewed. New products will get a cursory review (no pun intended) in a monthly column by Lee Pappas. In-depth evaluation of selected educational programs will appear in this space.

Now that the weather is worsening, I have greater access to my expert testers. It is impossible to get a fourteen year old boy or an eleven year old girl, with or without their friends, into the house when it is nice outside. The same is true for the pre-schoolers in my neighborhood, who are usually quite excited to come in and try Dr. Brad's new game. Once in the house and with homework out of the way, all I have to compete with is *The A Team* or **Archon**. Not only is a child's response essential in evaluating educational software, it makes the entire process easier. Their grasp of a program's goals and mechanics is infinitely superior to mine. Play testing, or "learn testing" with formulation of an opinion on a product, has been an added educational experience for them.

I am beginning to get a feel about providing a useful service to readers without showing product-specific bias. I would not like to ignore a good program because a brand new, sparkling educational package of a similar nature just arrived. For the moment, I would like to work on a theme concept for each month's column. Examples might include the following: math, language skills, typing tutors, geography, computer literacy, "oldies but goodies," and adventures. Reviewing programs of similar theme in the same issue may provide readers with a better idea of which ones meet their needs. In addition to the theme-related reviews, a "wild card" selection will be reviewed. This will be a product which does not fit into a particular theme, or one so good that I cannot wait to tell you about it. It also provides the editors with an opportunity to cut back on my column when there is a space crunch. If a wild card selection does not appear in a column, it should surface sooner or later.

Suggestions for specific themes or other comments concerning this column would be appreciated. (I already have a copy of Strunk and White's *The Elements of Style*, thank you.) Being in a bit of a vacuum, I have the urge to yell out, "Hey Mom! How'm I doing?"

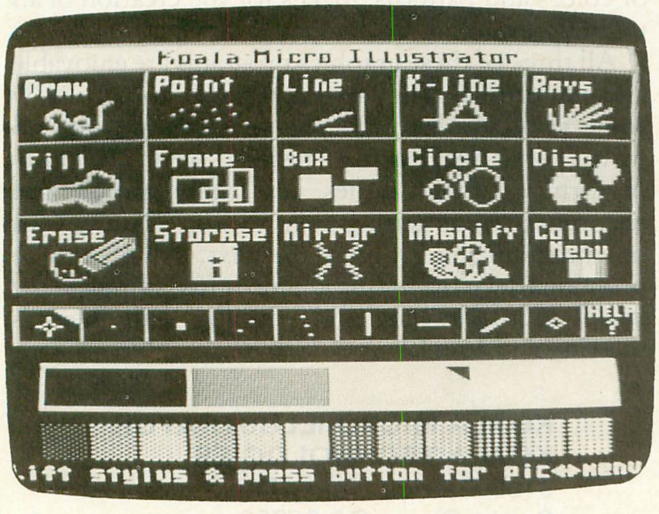
Ars gratia artis.

I'm not going to try to convince you of the importance of art in education. Nor am I going to discuss computer art and whether it has merit. Once past

the age of fingerpainting, many of us have ceased expressing ourselves artistically. The programs reviewed here will rekindle those hidden talents in the older of us, and initiate the same in the young. We can create a masterpiece....or not. Still, no paper is wasted. Proposition #X has resulted in the reduction, even elimination, of art and music education in many of our schools. Although the prospects look gloomy, these computerized ventures into the world of art offer some hope.

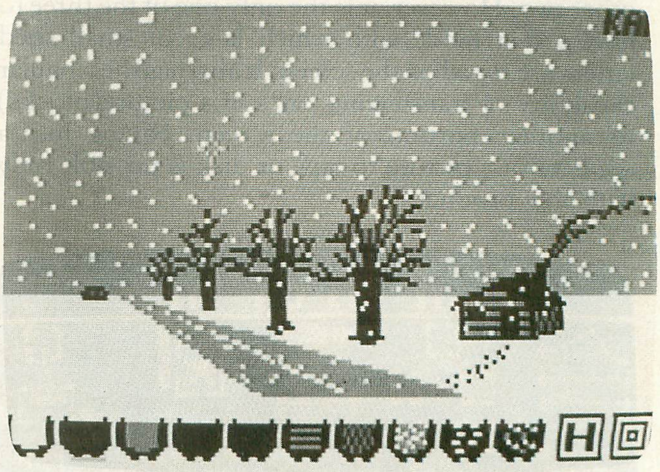
Three recently available art programs will be evaluated and compared. I will try to highlight the unique capabilities of each.

All three programs have similar basic functions. An integral part of each program is the MENU. The menus of **Micro Illustrator** and **Fun With Art** consist of graphic representations of the various functions accessed by the cursor. **Paint** has a menu with alphabetic characters as mnemonics to delineate its different modes. The menus are extremely conducive to quick and easy manipulation of several graphic techniques. The ease with which one alternates between the drawing screen and menu and the ability to select different design modes "on the fly" eliminate much potential frustration. Any child of school age can sit down and immediately begin using and enjoying any of these three programs.



Micro Illustrator.

The ability to DRAW is central to any program of this type. Use of the KoalaPad Touch Tablet makes **Micro Illustrator** unique. Although the other two use joysticks, **Fun With Art** is more like drawing with a pen or pencil, while **Paint** gives the feeling of painting with a brush. Several brush stroke widths and a number of different styles of brush tips are available to budding artists. ERASING is simple, but protected from accidental loss by requiring a confirmatory "yes" to the now classic "Are you sure?" from the computer. Other common features include modes which FRAME or outline, draw a straight LINE between two points, and draw CIRCLES and RECTANGLES of different sizes. The FILL command allows one to fill a geometric shape with an endless variety of colors or designs. Not only is this a powerful tool, but it is great fun to watch as one fills distinct areas with brilliant colors.



Paint.

If drawing is a manifestation of the body, then most certainly color is the soul. The capacity to select from one hundred and twenty-eight colors and explore various color combinations within the same picture is a prime example of the flexibility of these programs. Mixing colors, varying shades, and altering background colors with a variety of textures and patterns set this apart from conventional art.

The ZOOM or magnifying mode gives one the opportunity to view and alter a picture one pixel at a time. This feature allows for meticulous detail. It is remarkable to see a picture or design up close and notice how it differs with the change of perspective. The work of the French Impressionists with pointillism is brought into (or out of) focus, and adds to one's appreciation of their work.

Paint's pictures can be stored only on diskettes, while the other two offer both tape and diskette storage capabilities. These utilities are easily accessed from the menu and permit formatting a disk at any point. None has the ability to produce a hard copy with a printer. **Micro Illustrator** and **Fun With Art** include information on the use of the created pictures with other programs. There is extensive documentation and even a tutorial on disk covering programming techniques included with the KoalaPad Touch Tablet.

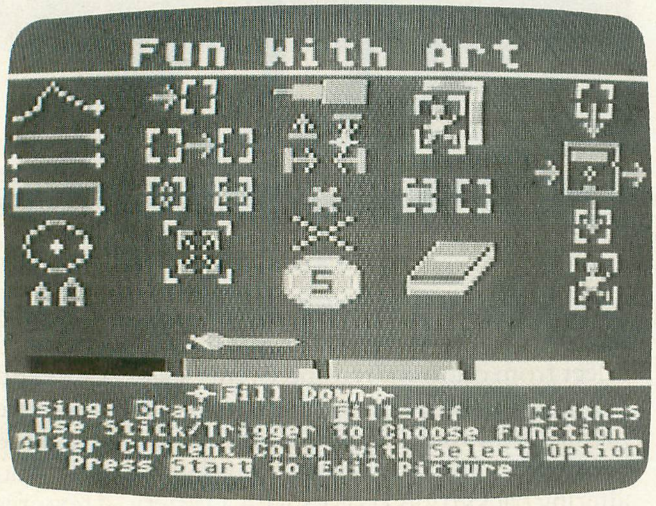
Different Strokes.

When Yogi Berra's son, Dale, was asked about his father's knack for malapropisms and his own seemingly genetic predisposition to the same, he replied, "Our similarities are different!" The advantages and disadvantages of each of these programs

will be discussed. Any mention of unique features refers only to the comparison of the three programs at hand.

Fun With Art.

This cartridge-based program does not offer a large variety of brush styles. Color selection and change are a little more burdensome than with the other two. However, it is the only one of the three to have a text mode with which large or small sized letters are typed on the screen. It also has the singular capability of transferring parts of a picture(blocks) to another portion of the screen. Saving just a segment of a picture to tape or disk is possible only with this product.



Fun With Art.

The block transfer mode enables one to append an image, zoom in on a block's contents or produce a mirror image.

The ability to set color priorities is also unique to **Fun With Art**.

The four basic colors can be set to draw over or under each other, depending on their respective color height or priority. The simplicity afforded with a cartridge makes this a particularly attractive package for young children.

Paint.

The inability to use these pictures in other programs is a significant detraction. I suspect there is a way to accomplish this, but there is no supporting documentation for such. The lack of the mirror image makes creation of intricate, symmetrical designs difficult, if not impossible.

There are many attributes. Eighty-one different styles and sizes of brush strokes are available. Being able to vary brush speed is a nice touch. The ability to internally or externally fill an area is found only in this program. If one wants to view a number of "paintings" sequentially, the ART SHOW function allows selection of up to twenty-four different pictures to be shown in a continuous display.

A one hundred and forty-seven page book accompanies this program. It was written for children and is exceptional. Documentation on the use of the software is adequate. Chapters on computer functions, imaging, computer artists, and a potpourri of art history are included. The highlight of the book is the chapter entitled "Idea Shop." The twenty pages of this chapter, designed to stimulate the imagination, suggest painting a lie, or peace, or pride, or making a painting to make one dizzy. Clever and informative, this book is a real bonus.

Micro Illustrator.

The only negative aspect of this package is the cost. The KoalaPad Touch Tablet is a lot more expensive than a joystick, but is there ever a big difference. Drawing on the touch tablet with finger or stylus is a joy. The ability to create lines that radiate from a single point is a distinctive feature. The lines, shapes, and other images are very sharp. The best drawn circle is found here, and is as easy as *pi*(sic)—(sick!) to use. There is an option in the color menu which changes any single color in a picture to a moving rainbow of colors. It is a dramatic special effect.

The mirror function may be used with most other modes and allows one to draw frames, circles, rays, etc. and have exact replicas in all four corners. This feature, along with the fill mode and use of a variety of colors and patterns, makes for the creation of an unlimited number of distinctive designs.

All three of these graphics programs are enjoyable and educational. They are easy enough to be used by some pre-schoolers. No one will tire of them, no matter what age. As my "testers" were rigorously performing their duties, the only problem I encountered was getting them to stop drawing with one program and get on to the next. Whether these programs develop artistic ability or just stimulate the imagination, drawing and painting with them is relaxing and fun. □

KOALA MICRO ILLUSTRATOR
KOALA TECHNOLOGIES
3100 Patrick Henry Drive
Santa Clara, CA 95050
16K/32K Catridge/Disk $99.00

PAINT
RESTON SOFTWARE
11480 Sunset Hills Rd.
Reston, VA 22090
48K(BASIC) Disk $39.95

FUN WITH ART
EPYX COMPUTER SOFTWARE
1043 Kiel Court
Sunnyvale, CA 94089
32K Cartridge $39.95

ATARI SINGS YOUR FAVORITE SONGS!!!

THE Original VOICE BOX Speech Synthesizer by the ALIEN GROUP has received rave reviews:

MICRO COMPUTING—"The VOICE BOX injects an endearing personality to your computer. The possibilities are enormous."

COMPUTE—"The VOICE BOX offers more human-like tones and does not blank out the screen."

CREATIVE COMPUTING—"English text and phonetic code may be freely intermixed rather than requiring separate modes as is the case without exception with every other speech system. A mode called talking face displays an animated face with impressive lip sync animation."

ANTIC—"There is a great potential for teaching children to spell and an added dimension to games overall. I believe the VOICE BOX is well worth the price tag."

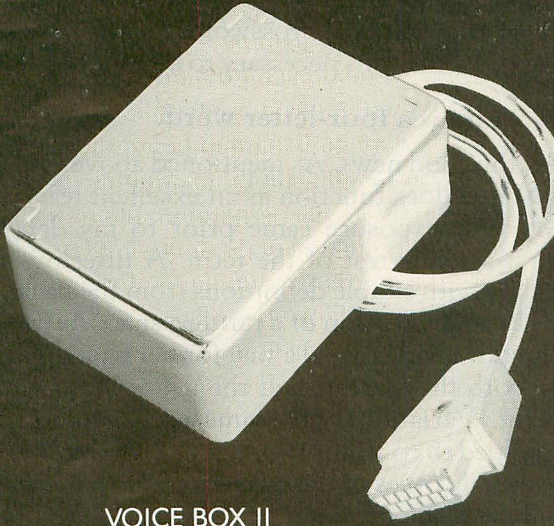
ANALOG—"For ATARI owners who want to add speech to their programs, the Alien Group VOICE BOX is probably the best choice."

POPULAR SCIENCE—"The speech quality is excellent. Besides creating speech, the software has a bit of fun with graphics."

and on the new VOICE BOX II......

TIME MAGAZINE—"Machine of the Year" "The VOICE BOX by the Alien Group enables an ATARI to say aloud anything typed on its keyboard in any language. It also sings "Amazing Grace" and "When I'm 64" or anything else that anyone wants to teach it.

INCORPORATE THE SINGING HUMAN FACE INTO YOUR PROGRAMS AND GAMES



VOICE BOX II
Speech & Singing Synthesizer

To order by mail send a check or money order to the ALIEN GROUP for $169.

THE ALIEN GROUP
27 West 23rd Street (212) 741-1770
New York, NY 10010

The New **VOICE BOX II** for **ATARI** plugs into the serial port of the **ATARI** 400/800 with sound coming out of the TV/monitor. 48K DISK is required. It has all of the features of the original **VOICE BOX** plus many exciting new hardware and software features:

- The ability to sing with voice and 3 part music.
- A library of 30 famous songs.
- A comprehensive music system that allows the user to easily enter or modify new songs.
- Software that can convert the bottom two rows of the **ATARI** keyboard into a piano with a range of 3½ octaves using the shift and control keys.
- Programmable musical sound effects such as tremolo, vibrato, glissando and click track.
- A singing *human* face with lip-sync animation designed by Jerry White.
- A talking or singing ***ALIEN*** face with software that allows the user to change the face and 8 mouth patterns as he sees fit.
- The ability to speak with inflection and feeling.
- Can speak in a foreign language with correct foreign spelling as input.
- A talk and spell program by Ron Kramer. Users can program any vocabulary for this spelling game. In fact, this program can even speak in a foreign language like French, where the user must spell the correct word in English, or vice versa.
- **GREEN GOBLINS**—A talking arcade game by John Wilson.
- Random Sentence Generator—An amusing grammar game that helps teach school children to identify parts of speech and recognize a variety of sentence structures.
- **NUMBER SPEAK**—A subroutine by Scott Matthews that converts up to a 9 digit number into normal English pronunciation. Ideal for building your own math games.
- **STUD POKER**—A talking poker game by Jerry White.
- The screen never blanks out while talking or singing.
- Singing or speaking subroutines can be incorporated into your programs, requiring as little as 100 bytes of RAM plus 5 bytes for each word.
- Price $169.00 includes **VOICE BOX II** and all of the above software.
- Inquire about our discounts for educational institutions.

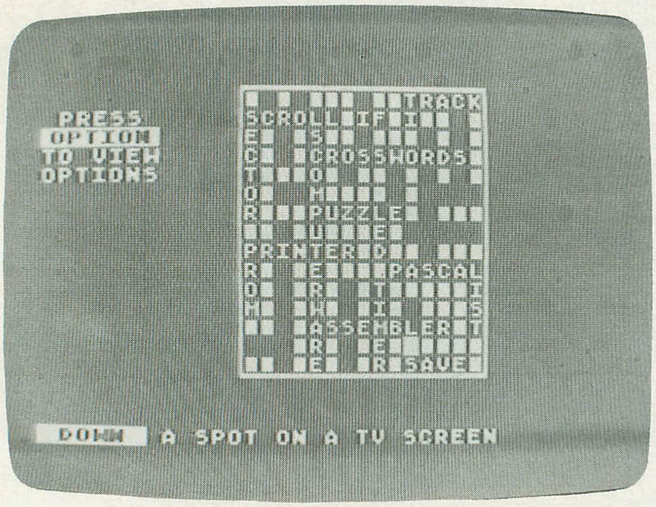
ALSO AVAILABLE AT LEADING COMPUTER STORES THROUGHOUT THE WORLD.

CROSSWORD MAGIC

Softsmith Corp.
1431 Doolittle Drive, San Leandro, CA 94577
48K Disk $49.95

by Braden E. Griffin, M.D.

Crossword Magic allows one to create a puzzle from a size 3 boxes by 3 boxes, to as large as 20 by 20. An option exists to choose the puzzle size before starting or to use the automatic puzzle sizing feature. I much preferred the latter. By simply typing a word and hitting RETURN, it is entered into a displayed puzzle grid. As subsequent words are entered, they are automatically interconnected with words already in the puzzle. If a particular word does not fit, it is stored away for future use. A word may be relocated to other available spots as desired. After all the words have been entered, clues for each word may be typed in. Clues may be as long as ninety characters and can be reviewed and retyped whenever one wishes. The puzzle may be saved to either the **Crossword Magic** disk (maximum of twenty puzzles) or any blank disk. An incomplete puzzle may be saved and completed later.



Crossword Magic.

There are seven options available. First is the CREATE mode, as discussed above. Additionally, one may COMPLETE an incomplete, previously saved puzzle, DELETE any puzzle from a disk, or TRANSFER puzzles from one disk to another (primarily for back-up purposes). An EDIT feature is provided enabling one to alter a completed puzzle. More words may be added or clues retyped; however, words already in the puzzle grid cannot be changed. Any completed puzzle may be PLAYED on the computer by simply typing letters onto the screen grid. The ARROW KEYS move the cursor about. PRINTing a hard copy of the puzzle is this program's most choice feature. The puzzle grid is reproduced with appropriate numbering of letter squares and clues. A pause allows one to advance the paper, after which, the solution is printed. Most standard printers are compatible, with two minor exceptions. The Microline 82A and 83A printers require the "Okigraph" ROM upgrade kit, and the A Epson MX-80 printer requires the "Graftrax" ROM upgrade kit. The final product is quite authentic and professional looking.

A bitter kvetch.

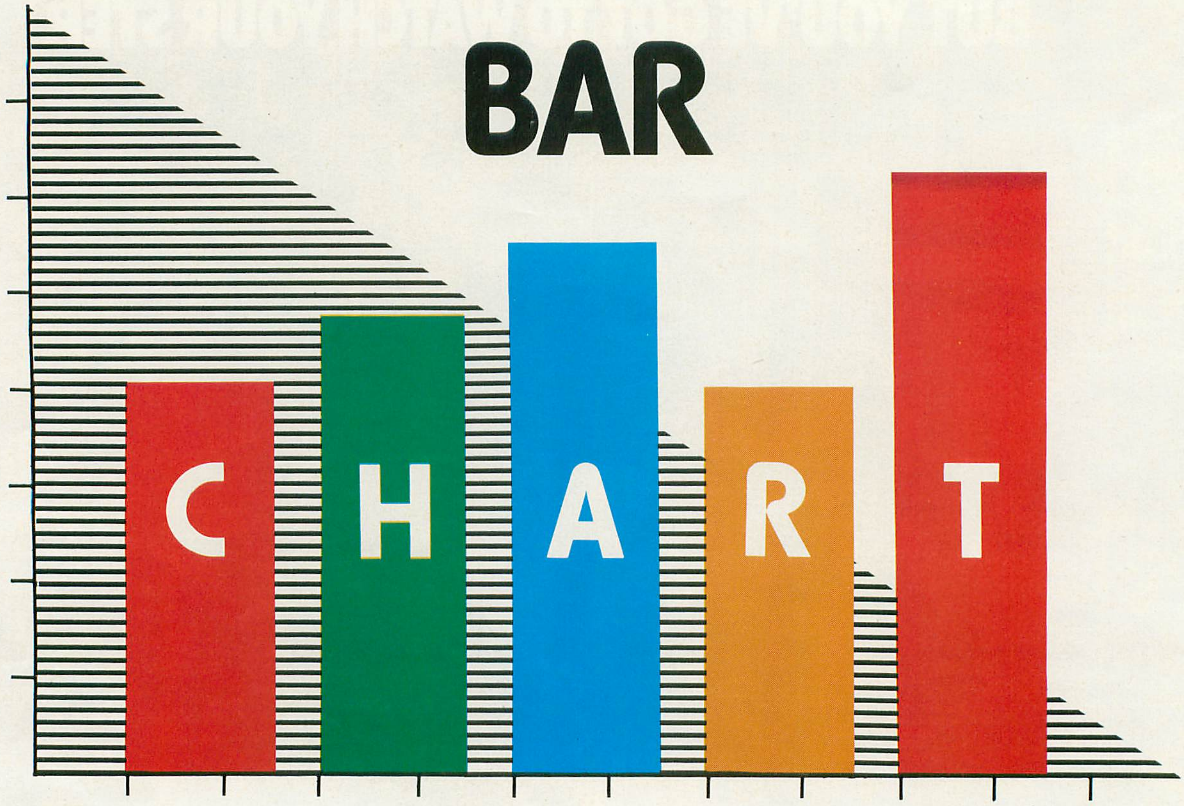
So why does **Crossword Magic** fall short of satisfying the true crossword aficionado? As each word is entered, it is interconnected with words already in the puzzle. The program will not permit the formation of words not entered or incompletely formed. This eliminates the creation of the solid block of words typically seen in standard crossword puzzles. For example, if the word "Atari" were entered as one across and "analog" as one down an "o" could not be inserted as the second letter of two down, even though it would spell "to" one way and "no" the other. In this case, the word "to" would be inserted using the "o" in "analog" as its final letter. The beauty of the standard crossword puzzle is in the unusual combinations necessary to accomplish this.

Life is a four-letter word.

Now, the good news. As mentioned above **Crossword Magic** does function as an excellent teaching aid. My first exposure came prior to my daughter's first spelling test of the term. A fifteen-word vocabulary with simple definitions from the back of the book made creation of a puzzle a snap. In about ten minutes, I had created a way for her to study her words, both the spelling and the definitions, while having fun. Variations of the same words on another puzzle helped to continue the review. As a sidelight, I made several copies of the puzzle for her classmates. Not only was she proud of her "clever" father, but her teacher told her how nice it was to have a father take such an interest in his child's schoolwork. Well, Softsmith gets credit for the "clever" part. The second occasion came with my son's confrontation with the periodic table. With this program, I was able to create a number of puzzle variations using the elements and their symbols. It made what first seemed drudgery an enjoyable learning experience. In the process, my son learned how never to forget the symbol for gold (Au). "Aaa.. You! Want some gold?" Anyway, both did well on their respective tests and we now have crossword puzzles all over the house on just about every subject; and on just about every piece of furniture, my wife says!

In summary, **Crossword Magic** is well done and has made learning in our house a lot more fun. Isn't that the way it should be? ☐

BAR CHART SUBROUTINE



16K Cassette or Disk

by Thomas P. Newdome

The advent of computer processing has brought the capability to provide a quick analysis of data via a graphic display. One type of display, the bar chart, is commonly used for displaying data that has values occuring at discrete intervals. The quarterly dividend of a given stock, the annual rainfall at a specific location, and the amount of money spent per month on food by a family are a few of many possible examples.

The following program, written as a subroutine, can be used to display a bar chart of data supplied by a user's program. The data is plotted against one of three different time intervals, with the user's program determining the range of the vertical axis of the plot. Included in the listing is a short program which demonstrates the use of the subroutine.

The three different time intervals which data may be plotted against are:

Type 1: The horizontal axis is divided into 24 monthly increments;

Type 2: The horizontal axis is divided into 28 quarterly (i.e. 3-month) increments;

Type 3: The horizontal axis is divided into 26 annual increments.

The vertical axis range of the chart is determined by the user. The actual range specified will, of course, be dictated by the range of data to be plotted.

The chart is constructed in Graphics mode 0. A high-resolution display is obtained through the use of a redefined character set. For a given bar, a resolution of up to 160 points is possible.

In order to use the subroutine, the calling program must supply values for six items;

1. The beginning year (two digits) to be used for labeling the horizontal axis.
2. The desired type (1-3) of time interval for the horizontal axis, as described above.
3. The minimum value to be used for labeling of the vertical axis. Typically, this value may be zero; however, a negative value (in the event a negative bar is to be plotted) or a value greater than zero is also allowed.
4. The vertical axis is "marked" off into 20 segments. Each segment has a resolution equal

to 1/8 of its range. The value supplied here will be the range between each segment (mark). For example, if a vertical axis which varies from 0 to 1000 is desired, then a zero would be specified for the minimum (item 3 above) and 50 would be specified for the range of each vertical mark (50 x 20 = 1000). The value for this parameter cannot be less than .01.

If a negative number is specified as the minimum (item 3 above), then the range of each vertical mark must be large enough to insure that zero (0.00) is included as a label on the vertical axis. The subroutine will automatically take care of this condition and include zero on the vertical axis, although the minimum may not be as desired.

The range of the vertical axis must be an interval between -99, 999.99 and 999,999.99.

5. The data to be plotted.

6. A descriptive title for the bar chart. This title should be 30 characters or less and use all upper case characters.

The values for the first five items above are stored by the user's program in an array, DAT ( ). The beginning year, plot type, minimum vertical label and vertical mark range are stored in DAT (1), DAT (2), DAT (3) and DAT (4) respectively. The data is then stored chronologically in DAT (5) through DAT (32). The desired title is stored in TITLE$, and a call to the subroutine can then be made (GOSUB 560). The subroutine plots the data and then returns to the user's program after any key is hit.

The following listings are composed of 3 segments. The first, Lines 100 through 490, is an initialization sequence which sets up the redefined characters that are used to construct the horizontal axis, vertical axis and data bars. Since the standard alphanumeric characters are also required, the upper case letters and numbers are copied from BASIC's standard character set to be used in the new character set. This is accomplished in Lines 180 through 220.

Lines 240 through 490 provide the definition of the redefined characters. Notice that this initialization also provides storage for DAT( ) and TITLE$ (Line 150). This initialization needs to be executed only once. Once accomplished, a branch to the user's program (second segment) is made in Line 510.

The user's program for this example is listed in Lines 1480 through 1600. The data values in Line 1520 are the chart specifications for DAT(1) — DAT(4). It will be instructive to make changes to the data values in this line to see how the chart construction is affected. Basically, the user's program puts the appropriate values into array DAT( ), sets TITLE$ equal to the desired title, and then calls the bar chart subroutine with GOSUB 560.

The third segment (Lines 520 through 1470) is the subroutine itself. Lines 610 through 830 draw the horizontal axis. The vertical axis is drawn by Lines 840 through 1040. Use is made of a short subroutine (Lines 1360 through 1470) in order to right-justify the numbers for the vertical axis labels.

The data is plotted sequentially at Lines 1050 through 1320. At Line 1070, the value of a given data point is converted to the number of blocks that will be displayed to construct the bar. Each block has a height equal to the height of a Graphics 0 character; as little as 1/8 of a block can be plotted. The number of full character blocks to be plotted is determined at Line 1110, and the fractional part of a block to be added is evaluated at Line 1130. Lines 1170 through 1230 draw a bar for positive-valued data while Lines 1240 through 1320 draw a bar for negative-valued data. A return to the user's program is accomplished at Line 1350.

Precautions.

There are a couple of precautions which should be noted. If the vertical range exceeds 999,999.99 or is below -99,999.99, then the subroutine will execute a program stop at Line 1380. Should this occur, readjustment of the minimum vertical value, DAT(3), or the vertical mark range, DAT(4), will be needed.

(continued next page.)

The redefined character set used by the subroutine is stored at the highest 1K block of RAM which begins at a 1K boundary. This beginning address is calculated as the value for variable NB at Line 160. When writing your application program, care must be taken to insure that this area of RAM is not over-written.

After the initialization sequence and subroutines have been typed in, the following command, entered in the direct mode while in Graphics 0, will indicate the actual amount of RAM available for your program.

```
? FRE(0)-PEEK(741)-1024
```

As your program is added, the value shown will decrease, but as long as it is greater than zero, memory will be available. With a 16K RAM machine, at least 8K bytes should be available for your program.

Additionally, your program must not use any graphics mode which requires more RAM than Graphics 0 (see Table 9.1 in the *Atari BASIC Reference Manual*). This means your programs will be limited to graphics 4 or below. Otherwise, RAM required by the screen display will modify memory where the redefined character set is stored. □

```
100 REM SAMPLE PROGRAM AND SUBROU-
110 REM TINE FOR PRODUCING A BAR
120 REM CHART.    WRITTEN BY
130 REM THOMAS P. NEWDOME
140 REM INITIALIZE CHARACTER SET
150 DIM DAT(32),TITLE$(30)
160 GRAPHICS 0:CB=57344:NB=(PEEK(742)-
4)*256
170 GRAPHICS 17:POSITION 4,10:? #6;"IN
ITIALIZING"
180 FOR II=0 TO 58:XX=II*8
190 A1=CB+XX:A2=NB+XX
200 FOR JJ=0 TO 7
210 POKE A2+JJ,PEEK(A1+JJ)
220 NEXT JJ:NEXT II
230 GRAPHICS 17:POSITION 6,10:? #6;"ST
ANDBY"
240 ME=NB+512
250 RESTORE 290
260 FOR II=0 TO 20:A2=ME+II*8
270 FOR JJ=0 TO 7:READ CH:POKE A2+JJ,C
H
280 NEXT JJ:NEXT II
290 DATA 255,255,24,24,24,24,24,24
300 DATA 255,255,24,24,0,0,0,0
310 DATA 255,255,0,0,0,0,0,0
320 DATA 3,3,3,3,3,3,3,31
330 DATA 3,3,3,3,3,3,3,127
340 DATA 0,0,0,0,0,0,0,0
350 DATA 0,0,0,0,0,0,0,126
360 DATA 0,0,0,0,0,0,126,126
370 DATA 0,0,0,0,0,126,126,126
380 DATA 0,0,0,0,126,126,126,126
390 DATA 0,0,0,126,126,126,126,126
400 DATA 0,0,126,126,126,126,126,126
410 DATA 0,126,126,126,126,126,126,126
420 DATA 126,126,126,126,126,126,126,1
26
430 DATA 126,126,126,126,126,126,126,0
440 DATA 126,126,126,126,126,126,0,0
450 DATA 126,126,126,126,126,0,0,0
460 DATA 126,126,126,126,0,0,0,0
470 DATA 126,126,126,0,0,0,0,0
480 DATA 126,126,0,0,0,0,0,0
490 DATA 126,0,0,0,0,0,0,0
500 REM BRANCH TO USERS PROGRAM
510 GOTO 1480
520 REM BAR CHART SUBROUTINE
530 REM ARRAY DAT() AND TITLE$
540 REM SHOULD BE PREPARED BY THE
550 REM USERS PROGRAM
560 PT=INT(DAT(2)):YR=DAT(1):LO=DAT(3)
565 IF PT<1 OR PT>3 THEN ? "PLOT TYPE
MUST BE 1,2, OR 3":STOP
570 VI=DAT(4):HF=36:LI=PT+1
575 IF VI<0.01 THEN ? "THE RANGE FOR E
ACH VERTICAL MARK MUST BE > .01":STOP
580 IF PT=2 THEN HF=38
590 IF PT=1 THEN HF=34:LI=11
600 GRAPHICS 0:POKE 756,NB/256:POKE 75
5,0
610 REM HORIZONTAL AXIS
620 POSITION 10,20:? CHR$(2)
630 POSITION 11,20
640 ? CHR$(0);
650 IF PEEK(91)>=HF THEN 710
660 FOR XX=1 TO LI
670 ? CHR$(1);
680 NEXT XX
690 GOTO 640
700 REM HORIZONTAL LABELS
710 HZ=11
720 IF PT>1 THEN 780
730 POSITION HZ,21
740 ? "JFMAMJJASONDJFMAMJJASOND"
750 POSITION HZ,22:? YR
760 POSITION 23,22:? YR+1
770 GOTO 830
780 POSITION HZ,21:? YR
790 IF PT=2 THEN YR=YR+1:HZ=HZ+4
800 IF PT=3 THEN YR=YR+5:HZ=HZ+5
810 IF YR>=100 THEN YR=YR-100
820 IF PEEK(91)<35 THEN 780
830 POSITION 24-(LEN(TITLE$))/2,23:? T
ITLE$;
840 REM VERTICAL AXIS
850 FOR XX=0 TO 19
860 POSITION 9,XX:? CHR$(3)
870 NEXT XX
880 VA=0:MB=0:V2=VI*2
890 ZE=19+INT(LO/VI)
900 IF LO>=VI THEN ZE=19:VA=LO:MB=LO/V
I
910 IF ZE<0 THEN ZE=0
920 VT=ZE:GOSUB 1380
930 VT=VT+2
940 IF VT>19 THEN 980
950 VA=VA-V2
960 GOSUB 1380
970 GOTO 930
980 VA=0:VT=ZE
990 IF LO>=VI THEN VA=LO
1000 VT=VT-2
1010 IF VT<0 THEN 1060
1020 VA=VA+V2
1030 GOSUB 1380
1040 GOTO 1000
1050 REM PLOT DATA
1060 FOR II=11 TO HF
1070 BK=DAT(II-6)/VI-MB
1080 SN=SGN(BK):BK=ABS(BK)
1090 IF SN=-1 AND ZE+BK>19 THEN BK=19-
ZE
1100 IF SN=1 AND ZE-BK<-1 THEN BK=ZE+1
1110 FB=INT(BK)
1120 XX=(BK-FB)*8
1130 PB=INT(XX)
1140 IF (XX-PB)>=0.5 THEN PB=PB+1
1150 IF PB=8 THEN FB=FB+1:PB=0
1160 IF SN=-1 THEN 1240
1170 IF FB=0 THEN 1220
1180 FOR JJ=0 TO FB-1
1190 POSITION II,ZE-JJ
1200 ? CHR$(13):NEXT JJ
1210 IF ZE-BK<=-1 THEN 1320
1220 POSITION II,ZE-FB:? CHR$(6+PB)
1230 GOTO 1320
1240 POSITION II,ZE:? CHR$(6)
1250 IF FB=0 THEN 1290
1260 FOR JJ=1 TO FB
1270 POSITION II,ZE+JJ
1280 ? CHR$(13):NEXT JJ
```

```
1290 IF PB=0 THEN 1320
1300 POSITION II,ZE+FB+1
1310 ? CHR$(21-PB)
1320 NEXT II
1330 POKE 764,255
1340 IF PEEK(764)=255 THEN 1340
1350 RETURN
1360 REM SUBROUTINE FOR RIGHT
1370 REM JUSTIFICATION
1380 IF VA>999999.99 OR VA<-99999.99 T
HEN STOP
1390 V3=ABS(VA):HZ=5
1400 IN=INT(V3):FA=INT((V3-IN)*100)
1410 IF IN=0 THEN 1430
1420 HZ=5-INT(CLOG(IN))
1430 IF VA<0 THEN 1450
1440 POSITION HZ,VT:? IN;".";FA;"0";:G
OTO 1460
1450 POSITION HZ-1,VT:? "-";IN;".";FA;
"0"
1460 POSITION 9,VT:? CHR$(4)
1470 RETURN
1480 REM BEGINNING OF USER PROGRAM
1490 RESTORE 1520
1500 FOR X=1 TO 32:READ DATA
1510 DAT(X)=DATA:NEXT X
1520 DATA 80,1,-5,2
1530 DATA 0,.25,.5,.75,1,1.25,1.5,1.75
,2
1540 DATA 0,-1,-2,-3,-4,-5
1550 DATA 0,4,6,8,10,12,24,32,12,10,9,
8,6
1560 TITLE$="PLOT TYPE 1 - HIT ANY KEY
"
1570 GOSUB 560
1580 DAT(2)=2:TITLE$="PLOT TYPE 2":GOS
UB 560
1590 DAT(2)=3:TITLE$="PLOT TYPE 3":GOS
UB 560
1600 GRAPHICS 0
```

CHECKSUM DATA
(see pp. 20-24)

```
100 DATA 7,285,886,714,486,174,311,663
,340,983,137,302,952,653,949,7842
250 DATA 215,598,8,970,937,345,303,16,
874,724,835,304,363,474,943,7909
400 DATA 974,85,554,86,986,933,465,365
,312,844,739,965,311,51,727,8397
550 DATA 42,901,878,570,852,95,813,117
,312,722,478,683,713,497,693,8366
680 DATA 746,742,817,303,523,591,464,2
04,553,739,212,193,178,674,495,7434
830 DATA 587,12,400,711,747,809,616,35
2,835,747,576,837,861,840,745,9675
980 DATA 68,331,668,923,879,953,700,52
6,643,9,361,163,531,100,162,7017
1130 DATA 195,160,847,638,885,656,173,
896,251,297,712,621,893,630,168,8022
1280 DATA 904,903,167,205,723,15,859,7
93,163,423,800,623,375,922,923,8798
1430 DATA 910,905,907,825,798,50,192,8
80,114,431,441,814,318,105,755,8445
1580 DATA 536,540,60,1136
```

GATEWAY TO APSHAI
by The Connelley Group
EPYX COMPUTER SOFTWARE
1043 Kiel Court
Sunnyvale, California 94089
16K Cartridge $39.95

by Charles Bachand

I'm mad! Very, very mad! I just lost my last man to a Mamba Snake on level six of **Gateway To Apshai**, the new machine-language adventure cartridge from Epyx. I'm so upset, I think I'm going to play it over and over again!

When I get really angry with a game, it's usually because the game play is rotten, the instructions make absolutely no sense, or I'm constantly getting myself killed off and can't seem to gain any ground. None of these are true in **Gateway**. Here, the game frustration is genuine because when you lose, you really *want* to keep on playing. Game play is fast and furious because all inputs are limited to a joystick and three console buttons (START, SELECT and OPTION). The instructions are so simple that you need only read two or three pages to get started. As for the time it takes to get killed, I've been averaging one to two hours of play to advance into the higher dungeon levels.

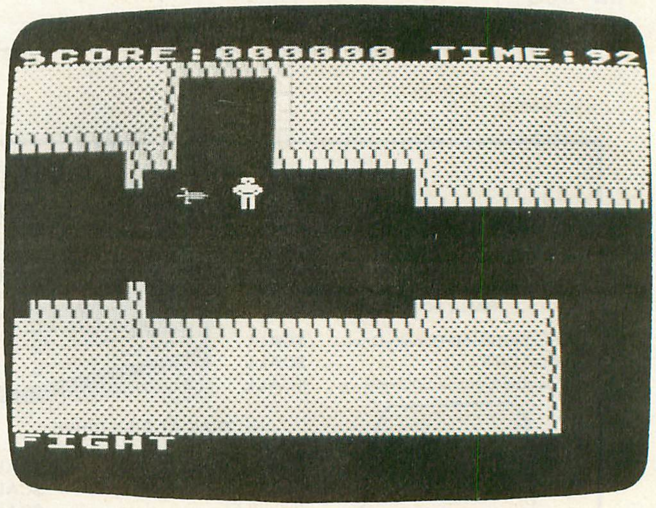
The basic scenario is that of a brave adventurer who has volunteered to explore the dungeons of Apshai. You are armed with only a dagger, a suit of leather armor and a prayer; as you explore the many rooms that make up each dungeon. Because you can only see rooms that have already been explored, entering unexplored rooms may reveal valuable treasure, magic talismans, stronger armor, deadly weapons or instant death! **Gateway** lets you choose from 16 different game levels, each of which is made up of eight dungeons, for a grand total of 128 dungeons with over seven thousand rooms. That should be enough to keep anyone busy for quite a while.

Gateway is an improved machine-language version of Epyx's best-selling adventure game **Temple of Apshai**. The original used keyboard input for everything, was relatively slow, and took forever to load (at least the cassette version did, and that was all I had when **Temple** first came out). To elaborate on how slow the original was, every time you entered a different part of the dungeon, the computer had to redraw the walls of the room to be displayed. This took from five to fifteen seconds. By contrast, **Gateway**'s wall-drawing is almost instantaneous and once drawn, stays drawn. This is accomplished by treating the screen as a window into a much larger dungeon, and using fine horizontal and vertical scrolling to view different sections. If you try to move your character off-screen (provided he doesn't bump into a wall first), the visible part of the dungeon slides off the screen and is replaced by a new area. Combined with the animation effects of your player and the creatures that you encounter in your travels, the scrolling dungeon effect adds greatly to the playability of the game.

Gripes.

One negative aspect of **Gateway** is that you can't save your games for later continuation. This is somewhat understandable due to the amount of data making up each dungeon, but it is still something that I wish was incorporated into the cartridge. If you want to play a game to its conclusion, you have to do it all in one sitting. There is also no straightforward way to pause the game, although you can fake it by going into one of the status screens until you're ready to go on.

Another gripe: Because you are constantly using the console buttons to select different options, you must be very careful not to accidentally hit the SYSTEM RESET key. This reboots the game from the beginning; it's happened to me on more than one occasion.



It is a fact that one's desire to play a computer game is inversely proportional to the time it takes to load that game. If **Gateway** was on cassette or disk, I would probably play with it two or three times and then put it away. However, being a cartridge, I have been playing **Gateway** during lunch, coffee breaks and even after work. [*Let's not forget those work-hour sessions, either. — Ed.*] I don't think I've played a game this much since I sat down to Sierra On-Line's **Ali Baba And The Forty Thieves** for eight straight hours. When the GAME OVER prompt finally appears, you look up and realize the two hours of your life have slipped through your fingers.

If you're looking for just another arcade shoot'em up, then this game may not be for you. But if you're into D&D games or enjoyed the original **Temple**, then I heartily recommend **Gateway To Apshai**. □

Lyco Computer Marketing & Consultants

TO ORDER
CALL US
TOLL FREE **800-233-8760**
In PA 1-717-327-1824

PRINTER INTERFACING
Available for IBM PC, Apple, Atari, Vic 20 & Vic 64

PRINTER PAPER AVAILABLE

SAVE on these in-stock PRINTERS

LETTER QUALITY

SMITH CORONA TP2...$449.00

DIABLO 630 ..$1719.00

ALPAHCOM 42 ....$89.00
ALPHACOM 81 ....$129.00
NEC 8023 ....$369.00
NEC 8025 ....$699.00
NEC PC-8200 COMPUTER ....$CALL

EPSON

RX-80 ....$SAVE$
RX-80FT ....ON
FX-80 ....In-Stock
FX-100..................EPSON
MX-80FT ....PRINTERS
MX-100............$$CALL$$

MANNESMANN TALLY

SPIRIT 80 ....$CALL
MT 160L...............$CALL

OKIDATA

80 ....$SAVE$
82A..................CALL for
83A....................LOWEST
84 ....PRICES
92 ....on these
93.......................In-Stock
PACEMARK 2350...PRINTERS

ATARI 850 REPLACEMENTS IN-STOCK

CITOH

GORILLA GX100 .....$179.00
PROWRITER 8510 ...$339.00
PROWRITER II $659.00
8600 ....$1025.00
STARWRITER.......$1099.00
PRINTMASTER $1499.00

STAR MICRONTICS

GEMINI 10X..........$269.00
GEMINI 15X............$CALL
DELTA 10............$479.00

MODEMS

ANCHOR MARK I ..$79.00
ANCHOR MARK II..$79.00
HAYES SMART ...$239.00
HAYES MICRO II $309.00
Micro Bit
MPP-1000.........$129.75
NOVATION
CAT $144.00
D-CAT $155.00
J-CAT $115.00
APPLE CAT II $279.00
212 APPLE CAT . $589.00

MONITORS

Sakata Color $229.00
Amdek Color I $275.00
Amdek 300 Green $149.00
Amdek 300 Amber $149.00
Gorilla Green $99.00

PARKER BROTHERS

Tutankham R............$33.75
Super Cobra R $33.75
Astro Chase R...........$33.75
Frogger R $33.75
QBert R $33.75
Popeye R................$33.75
Risk R $42.75
Chess R $42.75

SPINNAKER

Story Machine R $26.75
Face Maker R $24.75
Kinderomp R............$20.75
Fraction Fever R........$24.75
Delta Drawing R.........$26.75

SSI

Battle of Shilo C/D.........$26.75
Tigers in the Snow C/D....$26.75
Battle for Normandy C/D ..$26.75
Knights of the Desert C/D .$26.75
Cosmic Balance C/D $26.75

ON-LINE

Frogger $24.95
Wizard & Prin $26.95

ROKLAN

Wizard of War.............$29.75
Gorf $29.75
Delux Invader $27.95

BIG 5

Miner 2049................$32.75

APX

3R Math....... $19.95
Typo Attack $24.95
Family Budget $19.95
F. Cash Flow.............. $19.95

EASTERN HOUSE

Monkey Wrench 2 ....$52.75

BRODERBUND

Bank Street Writer D..... $49.75
AE D.........................$24.75
Apple Panic D.............$23.75
Choplifter ROM $32.75
David's Midnight..........$24.75
Stellar Shuttle C/D........$18.75
Ft. Apocalypse............$24.75

SYNAPSE

BLUE MAX C/D $24.75
Ft. APOCALYPSE C/D...$24.75
PHAROAH'S CURSE C/D...$24.75

DUST COVERS

800 ....$3.99
400 ....$3.99
1200.....................$3.99
410 ....$3.99
810 ....$3.99
1050.....................$5.99
PROWRITER $5.99
GEMINI 10X.............$5.99
PERCOM DISK $5.99

ALIEN GROUP

Voice Box 2 . $99.75

DON'T ASK

Sam $41.75
Abuse $15.95
Teleatri......$27.95
Poker Sam .. $24.95

PACMAN $29.75
DONKEY KONG $29.75
DIG DUG $29.75
DEFENDER $29.75
CENTIPEDE ... $29.75

EPYX

GATEWAY TO ASPHI R..............$28.75
JUMPMAN JR R $28.75
PIT STOP R $28.75
GATEWAY TO

ATARI

Computers for people.

600XL ...$CALL
800XL........for
1400XL...Lowest
1450 .....Prices

1020 PRINTER..........NOW
1025 PRINTER.............IN
1027 PRINTER ....STOCK
1050 DISK DRIVE ....$SAVE$
1010 RECORDER......$74.75

DEADLINE ..$34.75
ENCHANTER .$34.75
INFIDEL .$34.75
PLANETFALL .$34.75
STAR CROSS..$34.75
SUSPENDED ..$34.75
WITNESS ...$34.75
ZORK I ...$34.75
ZORK II ...$34.75

BLANK DISKETTES

ELEPHANT
Single Side SD (10).......$17.75
Single Side DD (10).......$21.75
Double Side DD (10)......$26.75

MAXELL
MD I (10) $28.75
MD II (10) $38.75

CERTRON CASSETTES

CC-10 12 for $15.99
CC-20 12 for $17.99

INNOVATIVE CONCEPTS

Disk Storage (holds 10)....$4.95
Disk Storage (holds 15)....$9.95
Disk Storage (holds 50)....$26.95

TRAK DISK DRIVES

AT-D1 ....$379.00
AT-D2................$399.00
PRINTER CABLE $22.95
Software for ATD-2 ....$22.95

RANA DISK DRIVE

COMPUTER CARE

BIB
5¼ DISK DRIVE CLEANER............$12.75
COMPUTER CARE KIT $19.75

HARD DISK DRIVES for APPLE IBM-PC

5MEG.....$1349.00
10MEG ...$1599.00
15MEG ...$1999.00
20MEG ...$2359.00
Add $30.00 for TRS 80 Drives

TEXAS INSTRUMENT

Disk Drive...$245.00

PERCOM

FOR ATARI COMPUTERS

AT88S1 ....$329.00
AT88S2 ....$535.00
AT**S1PD...$439.00
RFD40S1...$399.00
RFD40S2...$675.00
RFD44S1...$449.00
AT88 doubler

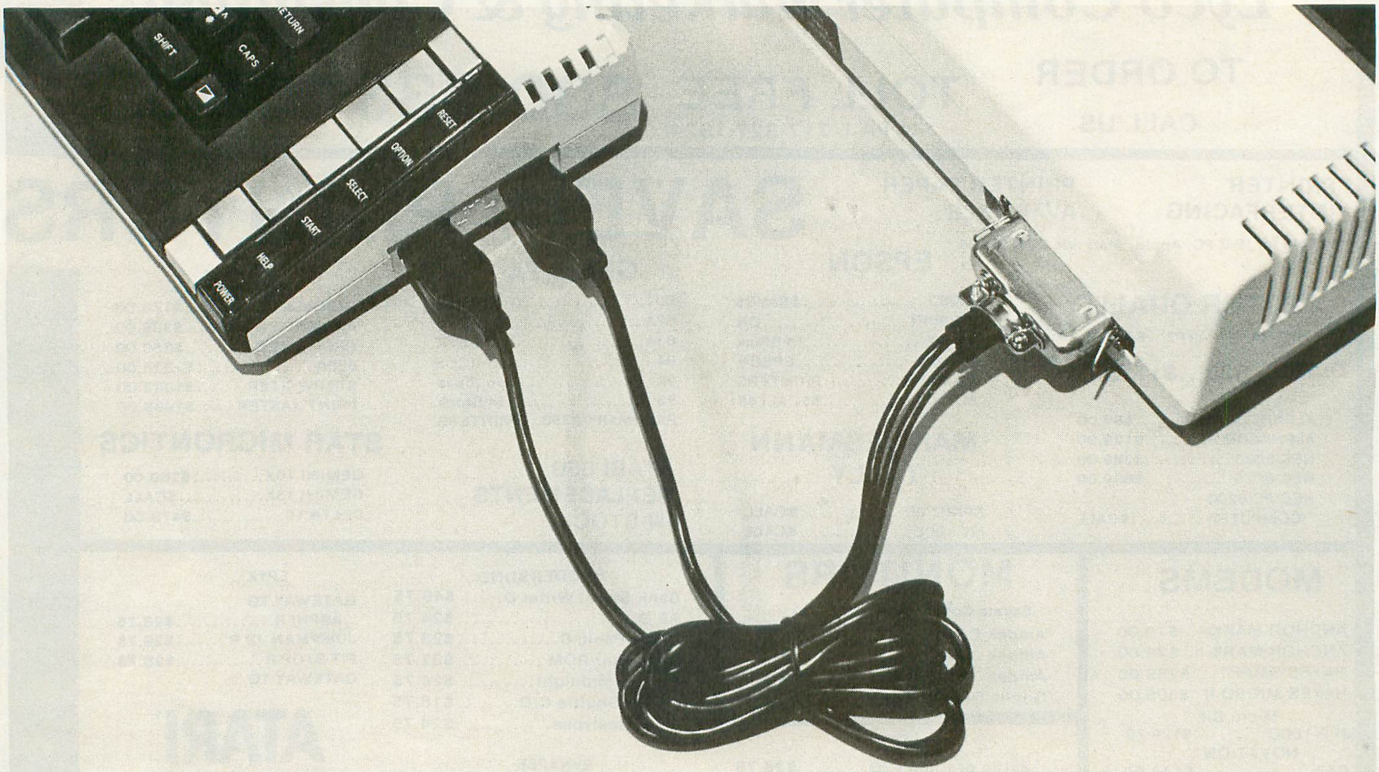
MasterCard VISA

TO ORDER
CALL TOLL FREE
800-233-8760
Customer Service 1-717-327-1825

or send order to
Lyco Computer
P.O. Box 5088
Jersey Shore, PA 1774C

POLICY

In-stock items shipped within 24 hours of order. Personal checks require four weeks clearance before shipping. No deposit on C.O.D. orders. Free shipping on prepaid cash orders within the continental U.S. PA residents add sales tax. All products subject to availability and price change. Advertised prices show 4% discount offered for cash, add 4% for Master Card or Visa. DEALER INQUIRIES INVITED.



BUILD A LOW COST PRINTER INTERFACE

16K Cassette or Disk

by Paul S. Swanson

The step up to a printer can be very costly, especially if an Atari 850 Interface Module is required. Of the printers not requiring an interface offered by Atari, the 1027 Letter-Quality Printer may prove too slow, and the 1025 Dot Matrix Printer has no graphics capability. There are many used Centronics-compatible printers available at very reasonable prices; but the 850 Interface you need to use them adds another $200 to the cost.

Even if you already own an 850, there are other potential problems with using it to drive a printer. One example is trying to run a printer in combination with a modem connected to the Interface Module. While the modem is in use, any attempt to send data to the printer will cause the system to lock up. This occurs because the 850's RC232C post depends on serial bus interrupts, effectively disabling all other I/O activity while the port is open.

Fortunately, it is possible to connect a printer to your Atari without an 850 by using a pair of joystick ports. It's not a perfect solution, but it does solve the problem mentioned above.

Restrictions.

A stick-port printer interface requires a special software handler, which must be installed in your system before you can use the printer. Therefore, this interface will not work with any commercial software that cannot be LOADed from BASIC. The machine-language handler is located in the upper half of page 6 (starting at address 1664 or $680 hex), so it will not work with any program that uses this area of memory.

These restrictions eliminate a few uses for the interface, but several others remain. For instance, programs you write yourself in BASIC can use this interface, and can be LISTed to a printer through it. Public-domain software written in BASIC can also benefit, because the handler can be initialized before the application is LOADed. This interface also allows you to print while an 850 modem channel is open, if your printer is fast enough to keep up with incoming data. Unlike other stick-port interfaces, my design uses only two ports instead of three or four, so it is fully compatible with Atari's new XL

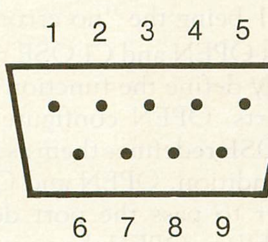
line of computers. And because 850 Interfaces are hard to come by nowadays, this low-cost interface might make a good temporary backup while you wait for your 850 to arrive.

Hardware considerations.

The hardware part of the stick-port interface is just about the simplest possible. All you need are three plugs. One connects to your printer; you will have to determine which type of plug your printer requires. The other two are standard 9-pin joystick plugs, available from any Radio Shack store for $2.49 each (catalog #276-1538). The plugs are wired together with a 12-conductor ribbon cable. I cheated by using a 14-conductor cable, leaving two conductors unused.

Refer to the documentation that came with your printer to determine the pin assignments for the plug at that end of the cable. This interface requires data lines 0 through 6. If the printer takes 8 data bits, the manual should indicate whether you should make Line 7 high or low for a 7-bit interface. If it doesn't say anything about it, connect Line 7 to ground.

You'll also need a strobe line and a busy line as well as the ground line. If no busy line is present, the "acknowledge" line may work. The ground line must be connected to the ground on your computer for reference.



Joystick pin assignments.

PIN	JACK 1	JACK 2
1	data 0	data 4
2	data 1	data 5
3	data 2	data 6
4	data 3	strobe
5	N/C	N/C
6	N/C	Busy
7	N/C	N/C
8	N/C	ground
9	N/C	N/C

Figure 1.

Figure 1 shows how the wires should be connected to joystick ports 1 and 2. The pin numbering on the plugs can be easily determined by holding the connector so that the 5-pin row is on top, with male connectors facing towards you and females facing away. With the connector oriented in this manner, pin 1 is in the upper left corner. Pins are numbered left to right across the top, and continue left to right across the bottom.

Using **Figure 1** and your printer documentation as a guide, get out your soldering iron and connect the ribbon cable to the three plugs. Be very careful to solder the wires and pins correctly.

Checking it out.

Before installing the software part of the interface, you can test your wiring with some immediate-mode commands in BASIC. BASIC is too slow to check the acknowledge line on most printers, but the following tests can be used to verify all of the other lines.

1. Turn your computer and printer off. Connect your new cable assembly to the stick ports and to the printer. Make sure the BASIC cartridge is installed and turn on the computer. If it doesn't initialize itself normally, shut it off immediately and check your wiring. Look particularly for shorts between pins 7 and 8 on either joystick plug.

2. When the computer displays the BASIC "Ready" prompt, turn on the printer. Again, if anything unusual happens, shut off the computer and printer immediately and check your wiring.

3. If everything appears normal, you are ready to test the connections. Refer to your printer's manual to determine the phase of its strobe line. This article assumes an active high strobe, such as required by an Epson MX-80; but a number of printers, including an IDS IP-225 that I have interfaced in this manner, require an active low strobe.

4. You must now set up PORTA on your Atari so that it is configured for output. Carefully type in the following immediate-mode lines and press RETURN:

```
POKE 54018,56:POKE 54016,255:POKE 5401
8,60
```

5. The simplest interface test is to send a carriage return to the printer. For printers with active high strobes, type the following:

```
POKE 54016,141:POKE 54016,13
```

For printers with active low strobes, swap the 13 and 141. Bit 7 is the strobe line in this interface, so 141 (which is 13+128) is used to set that line high.

Two things could have happened when you hit RETURN. Nothing is one possibility. If this is the case, check first to see if the printer is actually turned on and, if required, selected for "on-line." If your printer has a self-test function, you may want to try that also.

Some printers require line feeds. If yours is one of these, try the following immediate-mode commands:

```
POKE 54016,138:POKE 54016,10
```

If everything is connected, the printer is on-line and still nothing happens, it's time to start checking your wiring again. The most likely error is soldering the ribbon wires to the printer plug in reverse order.

6. If the carriage-return commands moved the paper, you're ready to try other things. Send the printer different characters using their equivalent codes. These can be found in the back of the *Atari BASIC Reference Manual*, or possibly also in your printer's manual. Remember that nothing will happen until you send the printer a carriage return. The EOL character generated by the Atari's RETURN key (CHR$(155)) will not work; you must use a decimal 13. If your printer needs a line feed, use a decimal 10 as shown above.

There are too many different types of printers around to cover all of the possibilities. If your printer has a parallel interface and doesn't follow the standards I'm describing, then you'll have to do a little research. There should be enough information in this article and in your printer's manual for you to figure it out yourself. Things that could vary include the polarity of the strobe and busy lines, and the presence or absence of bit 7. In a few instances, you may find that the data are inverted, in which case you must subtract from decimal 128 the 7-bit code for each character. This isn't very common, so your printer manual will probably mention it if applicable.

Your programming now has direct control over the voltages on the wires you connected. If you POKE a value into location 54016, its binary representation will be converted to a pattern of voltages on the wires. A binary 0 should correspond to less than 0.8 volts, binary 1 to at least 2.0 volts. This can be tested with a voltmeter by connecting its ground to the ground line of the interface cable, and using the "live" probe of the meter to measure each of the eight data lines.

The software.

Once everything checks out, you are ready to install the software portion of the interface. The Atari computer makes this an easy job for anyone familiar with assembly language. **Listing 1** is the assembly source code for an Epson printer interface.

This printer handler makes use of a set of utility routines in the Atari operating system called CIO (Central Input/Output). CIO handles most of the logic required to implement the OPEN, CLOSE, PUT and GET commands for the printer. If you perform a PRINT to the printer through BASIC, CIO will automatically break the print line down into individual characters, so that the interface will have to deal with only one at a time. LPRINT is a special BASIC command that is the equivalent of an OPEN/PRINT/CLOSE sequence. CIO will take care of all of these details for you.

CIO uses tables to find the locations of device handlers and other information it needs. The handler tables start at hexidecimal address $31A. Each I/O device is represented by three bytes. The first byte is the ATASCII representation of the device name, "P" or hex $50 in this case. The other two bytes are a pointer to the handler entry table, which can be located anywhere in memory.

An entry table contains a sequence of 2-byte addresses called vectors, which tell CIO where to branch for each supported function. The vectors must be arranged in the order shown in Lines 360 through 410 in **Listing 1** (addresses $680 through $68B). GETB, which gets one byte from the device; STATUS, which returns a device status code in response to a BASIC STATUS command; and SPECIAL, used to implement anything not covered like XIO commands, are not used in this handler, so these vectors all point to a routine that returns a "function not implemented" error code to CIO. CIO uses the 6502 Y register to pass error code numbers, with 1 being the "no error" indication.

The handler's OPEN and CLOSE routines (Lines 500-570) simply define the function of the pins on the joystick ports. OPEN configures the pins for output and CLOSE redefines them as input, which is their default condition. OPEN and CLOSE use the 6502 A register to pass the port definition byte, which is 255 ($FF) to OPEN the ports and 0 to close them. Both routines also use the PASTEUP (Port A Setup) subroutine to configure the ports.

PUTBYTE (740-1050) is the routine that actually sends a character byte to the printer. It begins by checking the busy line to make sure the printer is ready. For printers with busy lines that are active high, the BNE instruction in Line 760 must be changed to a BEQ instruction.

Line 770 of **Listing 1** is not required, and may be changed to a CLI or NOP instruction. Leaving the SEI instruction in place disables all other non-maskable interrupts from occuring during the PUTBYTE routine. If you want to use the printer concurrently with a modem, change Line 770 to a CLI ($58).

Setting up bytes for printing requires translation of the Atari End-Of-Line character (155 or $9B hex). This is checked first at Lines 810-830. Next, the high-order bit is set with the ORA #$80 instruction in Line 850. For printers with active low strobes, this must be changed to AND #$7F to clear the high-order bit. The STA PORTA instruction

sends the processed byte to the printer.

After a character has been sent, the printer must be given time to respond. The JSR JINIT instruction in Line 870 refers to a dummy routine set up to waste 8 machine cycles or about 2 microseconds, after which time the strobe line polarity is reversed by clearing bit 7 with AND #$7F. For active low strobe printers, change this instruction to ORA #$80.

The next sequence (930-980) is a timing loop that waits for the printer's busy line to respond. Epson's busy line is active high. Change the BNE PUT3 instruction in Line 960 to BEQ PUT3 if your printer's busy line is active low. This routine will return a time-out error code ($8B) to CIO if the printer fails to acknowledge by the time the loop counts down to zero.

The three unimplemented functions mentioned above are handled in Lines 1090-1140. This routine returns an error 146 if called.

The initialization routine is the final part of this program. Lines 1180-1270 find the current entry for the "P" device in the handler table at $31A, and replace it with the address of the new handler table. The JMP out of the initialization routine at Line 1410 will be defined by the BASIC program that loads the handler into page 6. This will make it possible for the handler to re-initialize itself whenever the SYSTEM RESET key is pressed.

The BASIC implementation.

In **Listing 2**, the assembly routine in **Listing 1** has been converted to decimal DATA statements. Try not to add or delete any bytes if you must change the machine code to fit a different printer protocol. If you must add or delete, be sure to alter the calling addresses in the handler entry table. Note that the entries in the table are the addresses of the entry points minus one.

The BASIC program first POKEs the machine-language code into its proper location in memory. Then the JMP vector at the end of the handler is adjusted to point to the address contained in DOSINI (location 12 or $0C hex). This assures that the normal SYSTEM RESET routine will be completed after initializing our handler. Finally, the system's DOSINI vector is "stolen" to point to the beginning of the handler's initialization code.

The last statement executed by the BASIC program is NEW, which clears the BASIC loader from memory. The interface handler is still tucked safely away in page 6, but it has yet to be installed. Press the SYSTEM RESET key and your new interface software will be initialized and ready to use.

Testing the handler.

The stick-port printer handler should act almost the same way as the normal 850 Interface handler as far as BASIC is concerned. One important difference is that the OPEN command does *not* check to see if a printer is actually on-line. Normally you can TRAP an OPEN command to see if a printer is available. For this new handler, a similar test can be performed by TRAPping a PRINT statement and looking for an error 138 (device timeout). All other functions (including LPRINT) should work identically to the resident printer handler.

If you're programming specifically for the stick-port interface and your printer's busy line is active high, you can use BASIC's STRIG(0) function to check for a "printer ready" condition. With the printer plugs connected, STRIG(0) will return a value of 0 if and only if the printer is connected, active and ready for output. Otherwise, STRIG(0) will return a value of 1.

Using the interface.

There are many possible uses for this type of printer interface. In addition to using it as a replacement for an 850 Interface, or for printing while using a modem, it also provides an easy way to add a second printer to your system. You'll have to alter some of the assembly code to identify the second printer as a new device. Instead of searching for an ATASCII "P" in the OS handler table, look for an unused table entry, represented by a $00 in the handler identification byte. Insert some other identifier, like "Q." You can then OPEN the "Q" device and PRINT to it when you want the output to go to the second printer.

This general scheme can be used to implement almost any 7-bit parallel output device. It doesn't even have to be a printer. It could be, for example, another computer. You could use such a system to transfer data from your Atari to another micro simply by PRINTing it. The restriction is that only seven bits are supported. There are also a number of plotters and other devices that could make use of this type of interface.

Your printer may provide a couple of signal lines that are not supported by this interface. A fault line is one example. This line is used to indicate that the printer encountered an error. This line isn't really necessary because the busy line will always reflect the fact that the printer is not ready. If you want to attach a fault line so that it can be checked, connect it to pin 6 on controller jack #1 and you will be able to read its state by using STRIG(0).

With a little imagination, the ideas presented in *this article* can be used to connect your Atari to almost anything that uses 5-volt logic. Even serial devices could be accessed by writing the routines required to convert back and forth between serial and parallel. With serial addressing, it is possible to interface to devices capable of both input and output. Enough lines are available to implement a direction control bit and bi-directional transfers of four-bit groups. By applying the principles introduced here, connecting other devices to your Atari should present very few problems. □

Listing 1.

```
;             JOYSTICK PORT PRINTER HANDLER
;             EPSON VERSION
;             ----------------------------
;
;             Program by Paul S. Swanson
;
; OS equates
;
HATABS = $031A    ; device handler address table
PACTL = $D302     ; port A control register
PORTA = $D300     ; port A I/O register
TRIG1 = $D011     ; stick trigger 1 register
;
; Program equates
;
ORIGIN = $0680    ; start of new handler
DUMMY = $FFFF     ; dummy address for JMP vector
;
     *=  ORIGIN
;
; Start of new handler table
;
NEWTABLE
;
; These are the new handler vectors,
; presented in the order CIO expects them
;
     .WORD OPEN-1
     .WORD CLOSE-1
     .WORD GETBYTE-1
     .WORD PUTBYTE-1
     .WORD STATUS-1
     .WORD SPECIAL-1
;
; JMP to init (also expected by CIO)
;
JINIT
     JMP EXIT      ; dummy init routine
;
; OPEN subroutine
;
OPEN
     LDA #$FF      ; set port A to "output"
     BNE PASETUP
;
; CLOSE subroutine
;
CLOSE
     LDA #0        ; set port A to "input"
;
; Configure port A
;
PASETUP
     LDX #$38
     STX PACTL     ; enable data direction control
     STA PORTA     ; specify "input" or "output"
     LDA #$3C
     STA PACTL     ; reset addressing mode
     LDA #$FF
     STA PORTA     ; clear the port
     LDY #1        ; OK status for CIO
     RTS
;
; PUT BYTE subroutine
;
PUTBYTE
```

```
0750      LDY TRIG1
0760      BNE PUTBYTE ; wait for busy line
0770      SEI
0780 ;
0790 ; Process byte to send
0800 ;
0810      CMP #$9B    ; is this an Atari EOL?
0820      BNE PUT1    ; ignore if not
0830      LDA #$0D    ; else convert to printer CR
0840 PUT1
0850      ORA #$80    ; set for active high strobe
0860      STA PORTA   ; and send byte to printer
0870      JSR JINIT   ; waste a few cycles
0880      AND #$7F    ; end the strobe pulse
0890      STA PORTA
0900 ;
0910 ; Wait for busy line
0920 ;
0930      LDY #0
0940 PUT2
0950      LDA TRIG1   ; printer ready?
0960      BNE PUT3    ; yes - continue
0970      DEY
0980      BNE PUT2    ; else keep waiting
0990      LDY #$8A    ; timeout error code
1000      CLI
1010      RTS
1020 PUT3
1030      LDY #1      ; no errors
1040      CLI
1050      RTS
1060 ;
1070 ; Unimplemented functions
1080 ;
1090 GETBYTE
1100 STATUS
1110 SPECIAL
1120      LDY #$92    ; error code
1130 EXIT
1140      RTS
1150 ;
1160 ; Initialization code
1170 ;
1180 PINIT
1190      LDY #0      ; init index
1200 PLOOP
1210      LDA HATABS,Y ; get an ID byte
1220      CMP #'P     ; is this the "P" entry?
1230      BEQ FOUND   ; yes, so change entry
1240      INY         ; otherwise skip 3 bytes
1250      INY
1260      INY
1270      BNE PLOOP   ; and keep looking for "P"
1280 ;
1290 ; Change table entry so that
1300 ; it points to our new handler
1310 ;
1320 FOUND
1330      LDA #NEWTABLE&255 ; lsb of table addr
1340      STA HATABS+1,Y
1350      LDA #NEWTABLE/256 ; msb
1360      STA HATABS+2,Y
1370 ;
1380 ; The following JMP vector will be
1390 ; set up when the handler is loaded
1400 ;
1410      JMP DUMMY
1420 ;
1430      .END
```

•

Listing 2.

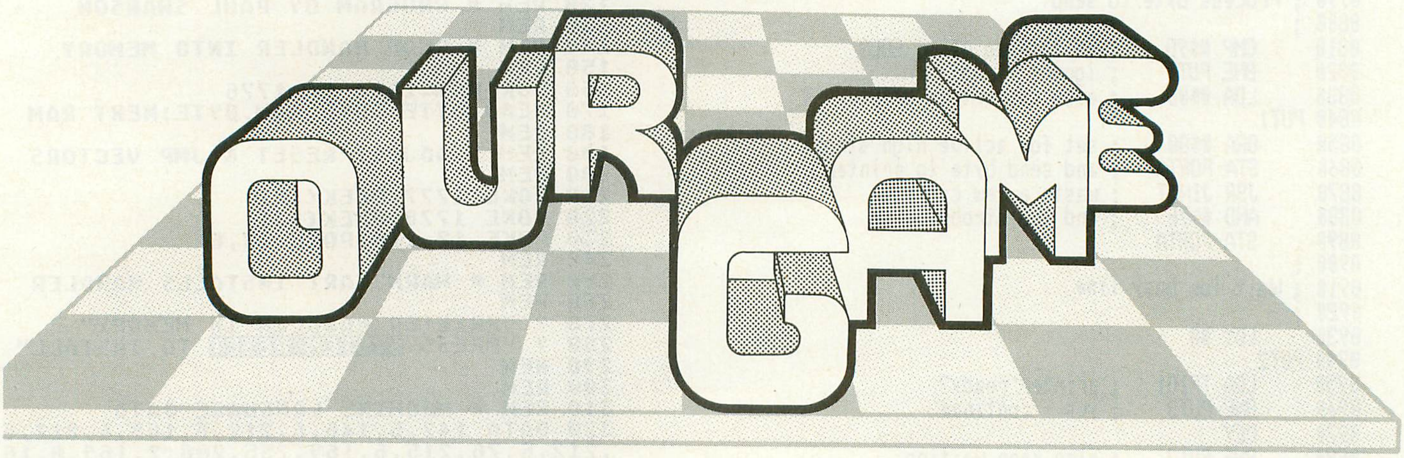
```
100 REM * STICK-PORT PRINTER HANDLER
110 REM * BASIC LOADER
120 REM * PROGRAM BY PAUL SWANSON
130 REM
140 REM * READ HANDLER INTO MEMORY
150 REM
160 FOR RAM=1664 TO 1776
170 READ BYTE:POKE RAM,BYTE:NEXT RAM
180 REM
190 REM * ADJUST RESET & JMP VECTORS
200 REM
210 POKE 1777,PEEK(12)
220 POKE 1778,PEEK(13)
230 POKE 12,216:POKE 13,6
240 REM
250 REM * WARMSTART INSTALLS HANDLER
260 REM
270 ? "PRINTER HANDLER IN MEMORY"
280 ? "PRESS SYSTEM RESET TO INSTALL"
290 NEW
300 REM
310 REM * MACHINE LANGUAGE DATA
320 DATA 142,6,146,6,212,6,169,6,212,6
,212,6,76,215,6,169,255,208,2,169,0,16
2,56,142,2
330 DATA 211,141,0,211,169,60,141,2,21
1,169,255,141,0,211,160,1,96,172,17,20
8,208,251,120,201,155
340 DATA 208,2,169,13,9,128,141,0,211,
32,140,6,41,127,141,0,211,160,0,173,17
,208,208,7,136
350 DATA 208,248,160,138,88,96,160,1,8
8,96,160,146,96,160,0,185,26,3,201,80,
240,5,200,200,200
360 DATA 208,244,169,128,153,27,3,169,
6,153,28,3,76
```

•

CHECKSUM DATA
(See pp. 20-24.)

```
100 DATA 108,146,639,83,868,89,965,234
,98,830,76,951,957,358,88,6490
250 DATA 68,94,603,341,111,78,424,918,
87,237,633,480,4074
```

•



by Joel Gluck

This is no ordinary **Our Game**, no siree! This is a special edition. What makes it special? Well, for one thing, there'll be none of the usual features — no viewer mail, no tutorials. Instead, there's a big, ugly program listing and a whole bunch of excuses and explanations.

Excuses and explanations.

Listing 1 is a prototype of a game — sort of. I wrote it in order to give you, the reader, a good idea of what a prototype looks like, and to introduce you to the style of programming that our game (the game we are going to be writing together in the coming months) is going to be written in.

In some ways, I have failed to achieve my purpose. The program is rather large (almost 9.5K) — something a prototype shouldn't be. This is because I succumbed to the temptation to include features that should only appear in a final version. I had good reason for doing this; why should you have to spend valuable time typing in what is merely an unexciting prototype and not a full-fledged game?

This pseudo-prototype game is called **Four Letter Words** (or **FLW**). The object is to spell a dozen unique combinations of the given four letters before your opponent can do the same. Most of the combinations are nonsense words, which count nonetheless. You spell words by moving your "man" (represented by either a "1" or a "2") and bumping into the desired letter on the playfield. Repeating letters in a word is not permitted.

The above describes what would have been a very nice prototype. But noooo! I just wasn't satisfied.

Superfluous Feature 1: There is a special box in the middle of the playing field. By bumping into it, you can delete the last letter of the word you are currently building.

Superfluous Feature 2: If you attempt to spell a word that is already on your opponent's list, as a penalty the last word on your list is transferred to your opponent's list. (Note: If you try to repeat a word already on your own list, you are buzzed at but nothing happens.)

Superfluous Feature 3: There is a "Forbidden Word" which is announced at the beginning of each game. If you spell the word, you lose the game. Simple enough?

Superfluous Feature 4: If you bump into your opponent's man, the two word-building displays switch. This means that whatever word you were working on, your opponent is now working on, and vice versa.

Superfluous Feature 5: There are plenty of elementary graphics and sound effects thrown in, just to liven things up.

That, in a nutshell, is the game. It may not sound like much fun. I have found that to be true of many games — a description of the game can sound very dry (especially in today's world of television, advertising and hype). However, I assure you that **FLW** is good. I've had about ten people playtest the game, and all but one like it. As a matter of fact, many of the features mentioned above stem from comments I elicited from playtesters.

Best of all, the game tries to achieve some of the game writing goals we've talked about in **Our Game**. For example, **FLW** is nonviolent, equally appealing to both sexes (and, to be hoped, a variety of age groups), and is a simultaneous two-player game.

So what's missing?

If **FLW** is so great, what's missing? Well, for one thing, the graphics are less than spectacular. I used character graphics (mode one), and did not modify

the character set to improve the game's appearance.

Also, the game is not complete. It is missing a title sequence, instructions, options, and a proper ending. As a matter of fact, when the game ends, it just sits and waits until either player pushes his/her trigger —and then starts over. To exit the game you have to hit BREAK, SYSTEM RESET or the power switch.

These deficiencies are not really deficiencies at all; they are common features of typical prototypes. The whole point of a prototype is to bring the basic idea of a game to life, not to create a finished product.

Big and ugly.

The listing itself, as I said before, is big and ugly. To be exact, it is 205 logical lines long. And, since many of these lines take up more than one physical line, the resulting length is about 13 Atari-screen fulls.

The ugliness of the listing can be attributed to a few things. For one, it has not been renumbered. This is for a reason; I frequently use line numbers to set off different parts of the program. This helps during programming and debugging; if I need to see a part of the program, I usually know the appropriate location of that part by line number. The strange numbering also helps reinforce the idea that the program is somewhat structured; it is divided into independent procedures.

The program is also ugly in that there are few REMarks, and most of those are not very understandable. This will not happen when we write our game. (Note: when typing in the listing, leave in all REMs that appear alone on a line. Frequently, these lines are referenced by GOSUBs or GOTOs.)

There is other, random ugliness. Many lines are long, and some are almost maximum length. This is not a good thing. Also, there are many complex string statements that look like pure nonsense. I would have avoided them if I could, but I couldn't, so I didn't.

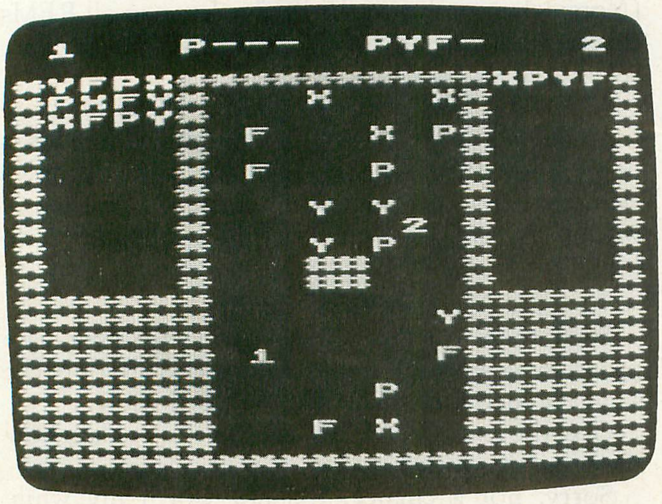
The final insult.

Well, there you are. I present you with a big, ugly program to type in to your computer, with only the slightest guarantee that it will be worthwhile.

Gee, the least I could do would be to explain how the program works!

Sorry, not a chance. It's not worth my while because it wouldn't do you any good. I've found that one does not learn much from typing in a program, even if there are accompanying explanations. If you'd like to learn from the experience, get a good book on Atari BASIC and figure out for yourself how what happens on the screen coincides with what's going on in the program. If you're more advanced than that,

try dissecting some of the more complex string statements in the program.



FLW

However, the best way to learn programming is to write you own programs, from scratch, often. Set problems or puzzles for yourself and solve them on your computer. Spend time experimenting with graphics, sound, and forms of input like the keyboard and the joysticks. Try new things, take on new challenges. The more you do your own programming, the better a programmer you will be.

Some things never change.

No, **Our Game** wouldn't be quite the same without the traditional "call for mail."

If you liked **FLW** and have ideas that could go into a final version of it, or if you didn't like it and have suggestions for improvement, or if you just have a better name for the game than **FLW**, send us mail!

If you have *any* ideas for **Our Game**, send us mail!

If you have any ideas about the state of computer/-video games in general, send us mail!

The address is:

OUR GAME
c/o ANALOG Magazine
P.O. Box 23
Worcester, MA 01603

Next month, it's back to the regular format. Don't miss the next **Our Game**! □

Big, Ugly BASIC Listing 1.

```
1000 REM - FOUR LETTER WORDS
1005 REM - the Prototype
1010 REM - Joel Gluck, Nov/Dec '83
1020 REM - 2 players, joysticks
1030 REM
1200 GOSUB 2000:REM - INITIALIZE
1300 GOSUB 3000:REM - GAME
1400 END
2000 REM - INITIALIZE
2100 DIM XSTIK(15),YSTIK(15),GMAN(1),N
WORDS(1),XWORD(1),XQUE(1),XMAN(1),YMAN
(1),NLTS(1),WORD$(200),LTTER$(10)
2110 DIM STP$(4),ADJUST(1),CONVERT$(4)
2150 FOR Z=5 TO 15:READ X,Y:XSTIK(Z)=X
:YSTIK(Z)=Y:NEXT Z
2160 DATA 1,1,1,-1,1,0,0,0,-1,1,-1,-1,
-1,0,0,0,0,1,0,-1,0,0
2200 WALL=138:DELETE=131:BLANK=32:SLOT
=13
2250 XFLD=6:YFLD=3:WDSLEFT=23
2300 GMAN(0)=49:GMAN(1)=178:XWORD(0)=5
:XWORD(1)=11
2310 XQUE(0)=1:XQUE(1)=15:ADJUST(0)=-3
2:ADJUST(1)=96
2320 LTTER$="1234567890"
2330 FOR Z=0 TO 19:WORD$(Z*10+1,Z*10+1
0)=LTTER$(1,10):NEXT Z
2350 FOR Z=1 TO 4
2360 STP$(Z,Z)=CHR$(65+INT(RND(1)*26))
2370 IF Z=1 THEN 2410
2380 FOR N=1 TO Z-1
2390 IF STP$(N,N)=STP$(Z,Z) THEN POP :
GOTO 2360
2400 NEXT N
2410 NEXT Z
2500 GOSUB 8000:REM - flash killer wor
d
2505 GRAPHICS 1+16
2510 SETCOLOR 0,12,6:SETCOLOR 2,4,6:SE
TCOLOR 3,9,4:SETCOLOR 1,1,6
2550 COLOR WALL:PLOT 5,2:DRAWTO 14,2:D
RAWTO 14,23:DRAWTO 5,23:DRAWTO 5,2
2552 FOR Z=0 TO 4:PLOT Z,14:DRAWTO Z,2
3:PLOT Z+15,14:DRAWTO Z+15,23:NEXT Z
2553 PLOT 0,2:DRAWTO 0,13:PLOT 19,2:DR
AWTO 19,23
2554 COLOR DELETE:PLOT 9,12:PLOT 9,13:
PLOT 10,12:PLOT 10,13
2560 COLOR GMAN(0):PLOT 1,0:COLOR GMAN
(1):PLOT 18,0
2570 FOR LT=1 TO 4:GLT=ASC(STP$(LT,LT)
)+32
2580 FOR N=1 TO 4:REM - number of each
 letter on field
2600 GOSUB 2900:REM - PLACE LETTER GLT
2620 NEXT N:NEXT LT
2660 FOR P=0 TO 1
2670 GLT=GMAN(P):GOSUB 2900:REM - PLAC
E MAN
2680 XMAN(P)=X:YMAN(P)=Y
2685 GOSUB 7000:REM - CLEAR SLOTS
2690 NWORDS(P)=0
2700 NEXT P
2890 RETURN
2900 REM - PLACE LETTER GLT (rtn X,Y)
2910 X=(INT(RND(1)*4)+1)*2+XFLD-1:Y=IN
T(RND(1)*10)*2+YFLD
2930 LOCATE X,Y,GS:IF GS<>BLANK THEN 2
910
2940 SOUND 0,X+Y,12,8
2950 COLOR GLT:PLOT X,Y
2955 SOUND 0,0,0,0
2960 RETURN
3000 REM - GAME
3020 P=INT(RND(1)*2)
3100 P=1-P
3120 S=STICK(P):IF S=15 THEN 3100
3130 GOSUB 4000:REM - MOVE
3140 GOTO 3100
4000 REM - CHECK MOVE
4100 XD=XSTIK(S):YD=YSTIK(S):POKE 77,0
4110 LOCATE XMAN(P)+XD,YMAN(P)+YD,G:IF
 G=WALL THEN GOSUB 7050:RETURN
4112 IF G=GMAN(1-P) THEN GOSUB 6000:RE
TURN
4115 IF G=DELETE THEN GOSUB 7100:RETUR
N :REM - DELETE CHAR
4120 IF G<>BLANK THEN GOSUB 4200:RETUR
N :REM - BUMPED SUMTHIN'
4125 REM - MOVE MAN
4130 COLOR BLANK:PLOT XMAN(P),YMAN(P)
4135 SOUND 0,P*30+10,8,8
4140 XMAN(P)=XMAN(P)+XD:YMAN(P)=YMAN(P
)+YD
4150 COLOR GMAN(P):PLOT XMAN(P),YMAN(P
)
4155 SOUND 0,0,0,0
4160 RETURN
4200 REM - BUMP
```

```
4210 IF G=EXCLAM THEN 6000
4220 COLOR G+128:PLOT XMAN(P)+XD,YMAN(
P)+YD
4230 IF NLTS(P)=0 OR NLTS(P)=4 THEN GO
SUB 7000:GOTO 4300:REM -1st letter
4240 FOR Z=1 TO NLTS(P)
4250 IF CHR$(G+ADJUST(P))=LTTER$(P*5+Z
,P*5+Z) THEN 4270
4260 NEXT Z:GOTO 4300
4270 SOUND 0,0,4,12:FOR PAUZ=1 TO 50:N
EXT PAUZ:SOUND 0,0,0,0:REM - repeated
letter
4280 COLOR G:PLOT XMAN(P)+XD,YMAN(P)+Y
D
4290 RETURN
4300 REM - letter is OK
4310 NLTS(P)=NLTS(P)+1
4320 LTTER$(P*5+NLTS(P),P*5+NLTS(P))=C
HR$(G+ADJUST(P))
4330 COLOR BLANK:PLOT XMAN(P)+XD,YMAN(
P)+YD
4340 COLOR G+ADJUST(P):PLOT XWORD(P)-1
+NLTS(P),0
4350 FOR SFX=20 TO 50 STEP 2:SOUND 0,5
FX+P*20,10,8:NEXT SFX:SOUND 0,0,0,0
4360 GOSUB 4125:REM - move man
4370 GLT=G:GOSUB 2900:REM - replace le
tter on field
4380 IF NLTS(P)<4 THEN RETURN
4400 REM - WORD COMPLETED
4410 FOR Z=1 TO 4
4420 IF ASC(LTTER$(P*5+Z,P*5+Z))<>ASC(
STP$(Z,Z))+ADJUST(P)+32 THEN 4450
4430 NEXT Z
4440 NWORDS(1-P)=13:POSITION 5,1:? #6;
"forbidden⊢":GOTO 5000:REM - word was
deadly
4450 REM - DID OPPONENT USE WORD?
4455 IF NWORDS(1-P)=0 THEN 4600:REM -
opponent has no words
4460 FOR Z=1 TO 4
4470 CONVERT$(Z,Z)=CHR$(ASC(LTTER$(P*5
+Z,P*5+Z))-ADJUST(P)+ADJUST(1-P))
4480 NEXT Z
4490 FOR CH=1 TO NWORDS(1-P)
4500 IF CONVERT$(1,4)=WORD$((1-P)*96+C
H*4,(1-P)*96+CH*4+3) THEN 4520:REM - w
ord was used already
4510 NEXT CH:GOTO 4600
4520 GOSUB 7200:REM - transfer a word
4530 GOSUB 7000:REM - clear slots
4540 IF NWORDS(1-P)>=12 THEN 5000
4550 RETURN
4600 REM - DID P(LAYER) USE WORD?
4610 IF NWORDS(P)=0 THEN 4700:REM - P
has no words
4620 FOR CH=1 TO NWORDS(P)
4630 IF LTTER$(P*5+1,P*5+4)=WORD$(P*96
+CH*4,P*96+CH*4+3) THEN 4650:REM - wor
d was used already
4640 NEXT CH:GOTO 4700
4650 FOR Z=0 TO 3:COLOR ASC(WORD$(P*96
+CH*4+Z,P*96+CH*4+Z))-ADJUST(P)+128
4655 PLOT Z+XQUE(P),CH+1:NEXT Z
4660 FOR SFX=15 TO 0 STEP -0.2:SOUND 0
,15-SFX,4,SFX:NEXT SFX
4670 FOR Z=0 TO 3:COLOR ASC(WORD$(P*96
+CH*4+Z,P*96+CH*4+Z))
4675 PLOT Z+XQUE(P),CH+1:NEXT Z
4680 GOSUB 7000:RETURN
4700 REM - THE WORD IS BRAND NEW!
4710 NWORDS(P)=NWORDS(P)+1
4720 FOR Z=1 TO 4
4730 WORD$(P*96+NWORDS(P)*4+Z-1,P*96+N
WORDS(P)*4+Z-1)=LTTER$(P*5+Z,P*5+Z)
4740 NEXT Z
4745 Y=NWORDS(P)+1:IF Y>23 THEN Y=1
4750 POSITION XQUE(P),Y:? #6;WORD$(P*9
6+NWORDS(P)*4,P*96+NWORDS(P)*4+3);
4770 FOR SFX=150 TO 50 STEP -20:SOUND
0,SFX,10,10:FOR PAUZ=1 TO 10:NEXT PAUZ
:SOUND 0,0,0,0:NEXT SFX
4780 SOUND 0,0,0,0
4785 WDSLEFT=WDSLEFT-1
4790 GOSUB 7000
4800 IF WDSLEFT>0 AND NWORDS(P)<12 THE
N RETURN
5000 REM - WE HAVE A WINNER
5100 WINNER=P:IF NWORDS(1-P)>NWORDS(P)
 THEN WINNER=1-P
5110 X=XQUE(WINNER)
5120 POSITION X,0:? #6;"win⊢"
5125 FOR Z=X TO X+3:LOCATE Z,0,G:COLOR
 G:PLOT Z,14:DRAWTO Z,23:NEXT Z
5130 FOR N=4 TO 15
5140 FOR SFX=100 TO 0 STEP -N
5150 SOUND 0,SFX,10,8:SOUND 1,SFX/2,8,
4:SETCOLOR 4,0,SFX
5160 NEXT SFX:NEXT N
5170 SOUND 0,0,0,0:SETCOLOR 4,0,0:SOUN
D 1,0,0,0
5180 IF STRIG(0)=0 OR STRIG(1)=0 THEN
RUN
5190 GOTO 5180
6000 REM - SWAP TOP DISPLAYS
6005 IF NLTS(0)=0 AND NLTS(1)=0 THEN G
OSUB 7050:RETURN
6010 SOUND 2,10,12,8:SOUND 3,12,12,8
6020 CONVERT$(1,4)=LTTER$(1,4)
6030 FOR Z=1 TO 4
6040 IF Z<=NLTS(1) THEN LTTER$(Z,Z)=CH
R$(ASC(LTTER$(5+Z,5+Z))-ADJUST(1)+ADJU
ST(0))
6050 IF Z<=NLTS(0) THEN LTTER$(5+Z,5+Z
)=CHR$(ASC(CONVERT$(Z,Z))-ADJUST(0)+AD
JUST(1))
6060 NEXT Z
6070 Z=NLTS(0):NLTS(0)=NLTS(1):NLTS(1)
=Z
6080 SETCOLOR 4,0,15
6100 FOR N=0 TO 1
6110 FOR Z=1 TO 4
6120 IF Z>NLTS(N) THEN COLOR SLOT:PLOT
 XWORD(N)+Z-1,0:GOTO 6140
6130 COLOR ASC(LTTER$(N*5+Z,N*5+Z)):PL
OT XWORD(N)+Z-1,0
6140 NEXT Z:NEXT N
6150 SETCOLOR 4,0,0
6160 FOR SFX=10 TO 0 STEP -1:SOUND 2,5
FX,12,SFX:SOUND 3,SFX+2,12,SFX:NEXT SF
X
6170 RETURN
7000 REM - CLEAR SLOTS
7010 COLOR SLOT:PLOT XWORD(P),0:DRAWTO
 XWORD(P)+3,0
7020 NLTS(P)=0
7030 RETURN
7050 REM - BUMP SOUND
7060 FOR SFX=15 TO 0 STEP -1.5
7070 SOUND 0,255-SFX,10,SFX:FOR N=1 TO
 1:NEXT N
7080 SOUND 0,0,0,0:FOR N=1 TO 1:NEXT N
:NEXT SFX
7090 RETURN
7100 REM - DELETE LETTER
7110 IF NLTS(P)=0 THEN GOSUB 7050:RETU
RN
7120 FOR SFX=50 TO 0 STEP -3:SOUND 0,5
FX,8,10:NEXT SFX:SOUND 0,0,0,0
7130 COLOR SLOT:PLOT XWORD(P)+NLTS(P)-
1,0
7140 NLTS(P)=NLTS(P)-1
7150 RETURN
7200 REM - TRANSFER A WORD FROM P TO 1
-P
7250 Y=CH+1:IF Y>23 THEN Y=1
7260 POSITION XQUE(1-P),Y:? #6;LTTER$(
P*5+1,P*5+4);
7265 SOUND 0,254,10,8:SOUND 1,255,10,8
7270 IF NWORDS(P)=0 THEN 7400
7280 Y=NWORDS(P)+1:IF Y>23 THEN Y=1
7290 POSITION XQUE(P),Y:? #6;"    ":RE
M - 4 SPACES
7295 NWORDS(1-P)=NWORDS(1-P)+1
7297 Y=NWORDS(1-P)+1:IF Y>23 THEN Y=1
7300 FOR Z=0 TO 3
7305 CONVERT$(1,1)=CHR$(ASC(WORD$(P*96
+NWORDS(P)*4+Z,P*96+NWORDS(P)*4+Z))-AD
JUST(P)+ADJUST(1-P))
7310 WORD$((1-P)*96+NWORDS(1-P)*4+Z,(1
-P)*96+NWORDS(1-P)*4+Z)=CONVERT$(1,1)
7315 COLOR ASC(CONVERT$(1,1)):PLOT XQU
E(1-P)+Z,Y
7320 NEXT Z
```

```
7325 FOR SFX=0 TO 50 STEP 4:SOUND 2,SF
X,8,15:NEXT SFX:SOUND 0,0,0,0:SOUND 1,
0,0,0:SOUND 2,0,0,0
7330 NWORDS(P)=NWORDS(P)-1
7400 FOR SFX=0 TO 240 STEP 10:SOUND 0,
SFX,10,8:SOUND 1,SFX+10,10,8:NEXT SFX:
SOUND 0,0,0,0:SOUND 1,0,0,0
7500 RETURN
8000 REM - FLASH SECRET WORD
8010 GRAPHICS 2+16
8020 SETCOLOR 0,0,15:SETCOLOR 1,4,4
8050 POSITION 4,4:? #6;"do not spell"
8100 FOR Z=1 TO 7
8110 POSITION 8,6:? #6;STP$
8120 FOR SFX=15 TO 0 STEP -0.7+0.6*(Z=
7)
8130 SETCOLOR 0,Z,SFX:SOUND 0,135,12,S
FX/2:SOUND 1,SFX*2,8,4:NEXT SFX:SOUND
1,0,0,0
8140 POSITION 8,6:? #6;"    ":REM - fo
ur spaces
8150 FOR SFX=1 TO 20:NEXT SFX
8160 NEXT Z
8170 RETURN
```

•

CHECKSUM DATA

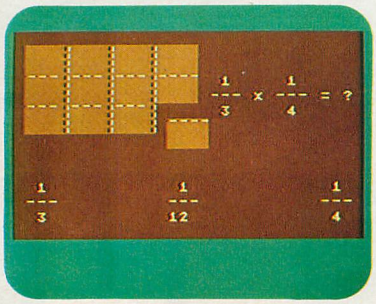
(See pp. 20-24.)

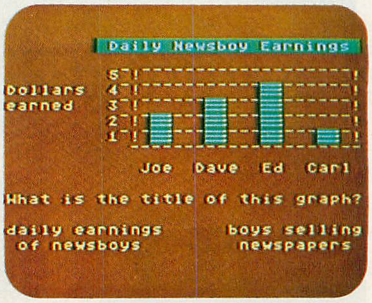
```
1000 DATA 16,103,109,491,278,148,205,2
65,897,761,760,718,798,328,964,6841
2300 DATA 151,701,900,306,186,600,564,
732,954,513,550,29,357,295,251,7089
2552 DATA 439,29,421,498,362,753,895,8
48,166,729,324,328,412,528,814,7546
2900 DATA 164,910,191,721,170,242,814,
966,398,508,236,257,714,760,886,7937
4110 DATA 213,589,696,0,715,167,913,70
5,250,222,794,20,151,771,872,7078
4240 DATA 80,188,829,648,242,800,641,2
02,989,950,310,317,375,116,50,6737
4400 DATA 602,189,610,556,934,636,879,
194,840,561,814,654,125,892,954,9440
4540 DATA 90,805,399,14,391,926,134,43
3,420,850,413,422,991,260,546,7094
4720 DATA 199,202,566,454,201,549,233,
421,962,67,526,332,929,649,328,6618
5130 DATA 362,648,106,813,633,336,745,
954,95,273,760,183,707,883,551,8049
6070 DATA 465,505,149,184,308,92,558,6
54,939,799,101,31,843,794,889,7311
7060 DATA 814,938,938,800,286,228,579,
459,207,799,335,748,232,407,537,8307
7280 DATA 439,53,972,992,187,221,967,1
60,558,756,546,82,806,925,346,8010
8020 DATA 423,631,196,521,20,598,225,8
71,558,803,4846
```

•

NEW! For Your Atari Computer

TRS 80 Color, I, III, 4 or Apple Computer





Over 1000 Programs with Full Time Audio Narration, Pictures & Text!

Interactive Tutorial Programs

All along you've heard there isn't much of a selection of low cost microcomputer courseware. Well, do you want to learn Physics, Psychology, or Philosophy? Accounting or Auto Mechanics? Sociology, Supervision or Statistics? Economics or Electronics? How about English as a Second Language? Would you like to teach your children Math or Reading? If you do, then we have the educational programs for you—and at affordable prices.

It's true there aren't many companies that offer full-length courses in subjects other than reading and arithmetic, and what is offered seems to be drills, tests, games, or simulations. What you really want is a course that covers the subject with, say, 16 full-length lessons called tutorial programs, where you interact with an expert programmer backed by a staff of experts. That's exactly what we have.

Is there something wrong with our software? Well, we don't ask you to enter your name so we can drop it into some later text. We don't ask you to type in your answer and refuse to accept it if it's not spelled just right. And, we don't branch around a lot when you make an error. Our programs simply let you know if you're wrong by proceeding only when you select the right multiple-choice answer.

This proven learning-by-positive-reinforcement method lets you proceed quickly and smoothly through the programs, without a lot of cute tricks. But, if you're a devoted computerist or game freak you may be disappointed at the lack of motion in most of the pictures, or our special visual effects. We have some dandies, like the mushroom cloud that rises over Hiroshima in our History series, but your learning is not distracted by needless special effects.

OVER 1000 PROGRAMS

We have 64 courses of 16 half-hour programs: 1024 programs! All are easy-reading, upper- and lower-case. All are in color. All are illustrated by frequent graphics composed of special and regular characters. And, best of all, every frame of every program is accompanied by high-quality, full-time audio narration by professional voice talent. David Stanton, James Mathers, Pam Barrymore. Recorded and played back, not synthesized or digitized!

— If your child is having trouble with Reading or Math, or if he or she is exceptional, and could benefit from professionally programmed lessons or courses, then you need our educational programs for your Atari, Apple, or TRS 80 to help your child.

— If you want to build your math skills, we have programs on Numbers and their meanings, Addition, Subtraction, Multiplication, Division, Fractions, Decimals, Percents, Angles, Graphs, Word Problems, Algebra, Statistics. There are ten series in all, 160 math programs. Again, all with color, pictures, and a friendly tutor's voice—all the time.

— If your verbal skills need polishing, try our Talk & Teach programs on the alphabet, spelling skills, and every level of vocabulary; Sight Words or Learn by Phonics; develop Reading Comprehension skills from stories and articles. There's even a series on the Great Classics.

— For self-development, try a 16-program course in Economics or Psychology. Or Supervision, Sociology, Counseling (Personal, School, or Employee), Writing, Business, Philosophy, Government, World History, U.S. History, or Accounting.

— For vocational skills, study our 16-program courses in Electronics, Fluid Power, Auto Mechanics, Shop, Carpentry, Construction, Meat Processing, Military Skills (64 programs), and there are many more.

There are 832 programs for your Atari, Apple and TRS 80 Mod I/III/4 and 160 for the Color Computer. Send for a free catalog.

Would you like to do something more than play games on your computer? Tired of squeaks and robotics and want to hear a human voice? Do you want to further your education or help your children along in theirs? Now you can. For just **$8.80**, we'll send you one cassette with 2 programs from the course of your choice, 100% guaranteed to work in your Atari, Apple, or TRS 80 computer, and you can try us out. Better yet, get one full course of 16 programs on 8 cassettes for only **$59.90**. Atari Master Cartridge, **$9.95**; Apple T/T Board, **$99.00**; and T/T Player, **$79.00**. That's less than books and tuition for most college classes. And we offer a 10 day, 100% exchange allowance. Does your alma mater? Send us your check, or call us at **1-800-654-3871** with your Visa/Master Card number. We'll send your educational software pre-paid. Please allow 15 days delivery. Or see your computer dealer. He may have some of our courses in stock. You've got nothing to lose but your games!

We're Your Educational Software Source.

Course	No. of Programs
Reading	256
Mathematics	128
Comprehension	48
History	32
Algebra	16
Spelling	16
Government	16

16 Programs in each of the following:
Carpentry-Electronics-Health Services
Office Skills-Statistics-First Aid/Safety
Economics-Business-Accounting-Psychology
And Many More!

For your Atari 400/600/800/1200, you will need the Atari cassette recorder and the Dorsett 4001 Educational Master Cartridge, $9.95. For your Apple II, you will need the Dorsett M402 T/T plug-in board, $99.00, and the M401 stereo cassette player, $79.00. All programs listed are available for TRS 80, I, III, 4, which require the M203 speaker converter, $99.00, and 401 stereo cassette player, $79.00.

Send for a catalog of over 1000 programs for Atari, TRS 80, Apple, etc.

For more information, write or call: **TOLL FREE 1-800-654-3871**

Dealer inquiries welcome

VISA MasterCard

DORSETT
Educational Systems, Inc.
P.O. Box 1226, Norman, OK 73070

GRAPHICS 10/7 PAINTER

by Peter C. Budgell

16K Cassette or 24K Disk

The following program can be used to create 7-color pictures under joystick and keyboard control if you have the GTIA chip.

A variety of uses exist for pictures which can be drawn on the screen, saved, and brought back under program control. Several commercial packages are available to do this, and many users will have seen the results in graphics adventure programs. Most commercial packages work in the hi-res Graphics 7½ mode, and compensate for the 4-color limit by mixing colors in some very interesting textures. Screens are stored either by dumping screen memory to storage media, or by storing the sequence of commands used to create the pictures and later reconstructing the pictures at machine language speeds.

The following program can be used to create pictures in the Graphics 10/7 mode which I introduced in **ANALOG** Issue 14, taking advantage of its 7-color capability, and 80*80 pixel resolution. Because this screen occupies 3200 bytes without the text window, its storage is very rapid. It is easy to add more features to the program, and relatively easy to convert it to most of the other graphics modes.

Featured in the program is a machine language fill routine. It is the familiar type which flows out in a diamond shape to fill any bordered area or to overwrite any solid color. The source code for the fill routine is included for those who would like to modify it to produce textured surfaces, by changing the color to be plotted in accordance with an imaginative algorithm. The routine is written to formally use CIO to do the plotting work. As written, it works in any graphics mode except Graphics 8, and requires only a change to the routines that check the screen size limits.

The program begins by POKEing the fill routine into page 6. Buffers which store the coordinates of points plotted by the fill routine are defined, and their locations in memory are POKEd for the fill routine to use. Then Graphics 10/7 is set up.

The user is presented with a blank screen and a flashing cursor. The cursor is moved with the joystick. To paint on the screen, press the trigger.

A variety of commands are available, and are

selected by pressing certain keys. The menu is presented if any non-command key is pressed. The menu includes:

D - DRAWTO — Pairs of points will be connected by a straight line.

F - FILL — A bordered shape will be filled.

C - COLOR — The available colors are selected in sequence.

X - CLEAR — The screen is cleared.

L - LOAD — Load a picture from storage media.

S - SAVE — Save the screen on disk or cassette. The higher speed short-IRG cassette mode is used.

R - COLOR REGISTER — Modify color register to gain complete color control.

These routines start at Line 1000. More features can be selected starting at Line 280, and put in subroutines at the end of the program.

Other graphics modes can be used if:

1. Screen limits in Lines 170 and 180 are changed.
2. Lines 82 thru 93 are deleted and replaced with a call to Graphics 3 through 7.
3. Lines 71 through 80 are removed.
4. Line 68 is changed to read:

```
68 DIM C(3):C(0)=0:C(1)=1:C(2)=2:C(3)=3
```

5. Line 3000 has CVAL= changed to CVAL=4 or CVAL=2 as appropriate.
6. In Lines 4040 and 5040, locations 856 and 857 are POKEd thusly:

Graphics Mode	3	4	5	6	7
POKE 856,	200	144	32	64	128
POKE 857,	0	1	3	6	13

7. Fill routine limits are changed thusly:

Line 34, BASIC program, change 80 to screen width.

Line 38, BASIC program, change 80 to screen height.

An understanding of the program can be used to implement Graphics 9 through 11, with or without a text window.

The subroutine at Lines 4030 to 4070 can be used in another program to load a picture from disk, if the user provides the file name in Line 4030. If the cassette is used, then Line 4030 should read:

```
4030 TRAP 4070:OPEN #1,4,128,"C:"
```

The background in a graphics screen is in COLOR 0. In order to erase part of a picture, select COLOR 0 and overwrite the unwanted image.

The biggest feature a user might wish to add would be a routine to draw a circle or polygon. Remember to trap errors, especially in plotting off the screen.

The fill routine alone can be used in other programs. The speed of the fill is limited by CIO, and commercial programs frequently go straight to screen memory to avoid CIO's overhead, which is caused by its generality.

Imagination is the most important ingredient in using a painter program. A slide show can be set up to show the results if the pictures are brought up sequentially by a loop in a program. The easiest way is to have the file names in data statements. Fun is available for those 3 years of age and up! □

```
5 GRAPHICS 2:? #6;"GRAPHICS 10/7":? #6
;"please stand by..."
6 POKE 752,1:GOSUB 400
10 REM  GRAPHICS 10/7 PAINTER
11 REM  INCLUDING FILL ROUTINE
12 REM  P. BUDGELL   1983
13 REM
14 DIM A$(256),B$(256),C$(256),D$(256)
:REM THE BUFFERS
15 DIM XSTICK(15),YSTICK(15),NAME$(15)
,MOVE$(6)
16 REM
17 REM STICK DIRECTIONS
18 REM
19 RESTORE 20:FOR H=1 TO 15:READ I:XST
ICK(H)=I:READ I:YSTICK(H)=I:NEXT H
20 DATA 0,0,0,0,0,0,0,0,1,1,1,-1,1,0,0
,0,-1,1,-1,-1,-1,0,0,0,0,1,0,-1,0,0
21 REM
22 REM THE FILL SUBROUTINE.  LIMITS
   SET AT 80 PIXELS ACROSS, 80 DOWN.
23 REM WE USE PAGE 6
24 REM
25 RESTORE 26:FOR I=1536 TO 1791:READ
H:POKE I,H:NEXT I
26 DATA 104,104,104,133,209,173,254,6
27 DATA 133,212,173,255,6,133,213,169
28 DATA 0,133,231,160,1,165,85,145
29 DATA 203,133,233,133,214,165,84,145
30 DATA 205,133,234,133,215,132,232,32
31 DATA 184,6,133,230,197,209,208,1
32 DATA 96,32,221,6,177,203,133,214
33 DATA 177,205,133,215,230,214,165,21
4
34 DATA 201,80,176,8,32,184,6,208
35 DATA 3,32,164,6,198,214,198,214
36 DATA 165,214,201,255,240,8,32,184
37 DATA 6,208,3,32,164,6,230,214
38 DATA 230,215,165,215,201,80,176,8
39 DATA 32,184,6,208,3,32,164,6
40 DATA 198,215,198,215,165,215,201,25
5
41 DATA 240,8,32,184,6,208,3,32
42 DATA 164,6,198,232,240,4,200,76
43 DATA 52,6,164,231,240,95,132,232
44 DATA 177,212,145,205,177,207,145,20
3
45 DATA 136,208,245,160,1,169,0,133
46 DATA 231,76,52,6,32,221,6,230
47 DATA 231,240,75,164,231,165,214,145
48 DATA 207,165,215,145,212,164,235,96
49 DATA 32,200,6,169,7,157,66,3
50 DATA 32,86,228,164,235,197,230,96
51 DATA 132,235,165,214,133,85,165,215
52 DATA 133,84,162,96,169,0,157,72
53 DATA 3,157,73,3,96,32,200,6
54 DATA 169,11,157,66,3,165,209,32
55 DATA 86,228,164,235,96,165,234,133
56 DATA 84,165,233,133,85,96,32,237
57 DATA 6,104,104,96,0,0,0,0
58 REM
59 REM SET UP BUFFER ADDRESSES FOR ML
   ROUTINE ACCESS
60 REM
61 J=INT(ADR(A$)/256):I=ADR(A$)-J*256:
POKE 203,I:POKE 204,J
62 J=INT(ADR(B$)/256):I=ADR(B$)-J*256:
POKE 205,I:POKE 206,J
63 J=INT(ADR(C$)/256):I=ADR(C$)-J*256:
POKE 207,I:POKE 208,J
64 J=INT(ADR(D$)/256):I=ADR(D$)-J*256:
POKE 1790,I:POKE 1791,J
65 REM
66 REM  GRAPHICS 10/7 SETUP
```

```
67 DIM DLI$(32)
68 DIM C(6):C(0)=0:C(1)=2:C(2)=3:C(3)=
9:C(4)=10:C(5)=11:C(6)=8
69 GRAPHICS 7:POKE 752,1
70 REM
71 POKE 623,128:REM GTIA GR.10
72 POKE 87,10:REM FOOL SCREEN HANDLER
73 REM
74 POKE 704,0:REM COLOR 0
75 POKE 705,12:REM COLOR 2
76 POKE 706,38:REM COLOR 3
77 POKE 708,100:REM COLOR 9
78 POKE 709,148:REM COLOR 10
79 POKE 710,202:REM COLOR 11
80 POKE 712,252:REM COLOR 8
81 REM
82 RESTORE 92:FOR X=0 TO 31:READ I:POK
E ADR(DLI$)+X,I:NEXT X
83 POKE PEEK(560)+256*PEEK(561)+84,141
84 POKE 513,INT(ADR(DLI$)/256):POKE 51
2,ADR(DLI$)-PEEK(513)*256
85 POKE 54286,192:REM ENABLE DLI
86 ? "      THIS IS GRAPHICS 10/7":? "
   P. BUDGELL 1983":FOR I=1 TO 250:NE
XT I
87 OPEN #2,4,0,"K:"
88 REM   DISABLE BREAK KEY
89 REM
90 D=PEEK(16)-128:IF D<0 THEN 92
91 POKE 16,D:POKE 53774,D
92 DATA 72,169,0,141,10,212,141,27,208
,141,26,208,169,144,69,79,37
93 DATA 78,141,24,208,169,10,69,79,37,
78,141,23,208,104,64
94 REM
95 RESTORE 96:FOR I=1 TO 6:READ A:MOVE
$(I,I)=CHR$(A):NEXT I
96 DATA 104,162,16,76,86,228
97 X=40:Y=40:COLOR C(1):CWORK=C(1):CVA
L=1:CUNDER=0
98 GOSUB 500:REM INTRODUCTION
99 A=PEEK(764):IF A<255 THEN GOSUB 200
100 GOSUB 170:GOSUB 101:GOTO 99
101 LOCATE X,Y,CUNDER:COLOR CWORK:PLOT
 X,Y:INVC=(CWORK=0)*C(1):FOR WAIT=1 TO
 3:NEXT WAIT
110 COLOR INVC:PLOT X,Y:FOR WAIT=1 TO
2:NEXT WAIT
120 COLOR CUNDER:PLOT X,Y
130 IF STRIG(0)=0 THEN COLOR CWORK:PLO
T X,Y
140 RETURN
170 I=STICK(0):X=X+XSTICK(I):IF X=80 T
HEN X=79
175 IF X=-1 THEN X=0
180 Y=Y+YSTICK(I):IF Y=80 THEN Y=79
185 IF Y=-1 THEN Y=0
186 IF I<15 THEN LOCATE X,Y,CUNDER2
190 RETURN
200 GET #2,CHAR:POKE 764,255
210 IF CHAR=ASC("D") THEN GOSUB 1000:G
OTO 500
220 IF CHAR=ASC("F") THEN GOSUB 2000:G
OTO 500
230 IF CHAR=ASC("C") THEN GOSUB 3000:G
OTO 500
240 IF CHAR=ASC("X") THEN PRINT CHR$(1
25):PRINT #6;CHR$(125):GOTO 400
250 IF CHAR=ASC("L") THEN GOSUB 4000:G
OTO 500
260 IF CHAR=ASC("S") THEN GOSUB 5000:G
OTO 500
270 IF CHAR=ASC("R") THEN GOSUB 6000:G
OTO 500
400 ? CHR$(125);" GRAPHICS 10/7 PAINTE
R MENU : "
410 ? " D=DRAWTO MODE  F=FILL  C=COLOR
CHANGE"
420 ? " X=NEW SCREEN   L=LOAD  S=SAVE
SCREEN"
430 ? " R=COLOR REG.        SPACE=REGULAR
 MODE";
440 RETURN
500 ? " REGULAR DRAW MODE - PRESS TRIG
GER"
510 ? "   COLOR   ";CVAL
520 ? " (TO ERASE USE COLOR 0 )"
525 D=PEEK(16)-128:IF D<0 THEN 530
526 POKE 16,D:POKE 53774,D
530 RETURN
1000 ? CHR$(125);"DRAWTO MODE: PRESS T
RIGGER TO DEFINE"
1010 ? "NEW STARTPOINT; PRESS AGAIN TO
 DRAWTO":? "PRESS ANY KEY TO EXIT"
1015 GOSUB 101:IF STRIG(0)=0 THEN POKE
 53279,0:X0=X:Y0=Y:GOSUB 1099
1020 A=PEEK(764):IF A<255 THEN RETURN
1030 GOSUB 170:GOTO 1015
1099 IF STRIG(0)=0 THEN 1099
1100 GOSUB 170:IF STRIG(0)=0 THEN COLO
R CWORK:PLOT X0,Y0:DRAWTO X,Y:POKE 532
79,0:GOTO 1500
1110 A=PEEK(764):IF A<255 THEN POP :RE
TURN
1111 GOSUB 101:GOTO 1100
1500 IF STRIG(0)=0 THEN 1500
1599 RETURN
2000 COLOR CUNDER2:PLOT X,Y:POSITION X
,Y:I=USR(1536,CWORK):CUNDER2=CWORK:RET
URN
3000 CVAL=CVAL+1:IF CVAL=7 THEN CVAL=0
3010 COLOR C(CVAL):CWORK=C(CVAL):? "
COLOR  = ";CWORK
3020 RETURN
4000 ? CHR$(125);"   LOAD WHAT FILE?   "
:TRAP 4070
4010 ? "GIVE NAME SUCH AS D1:XX.Y  OR
C:"
4020 ? " FILE NAME ";:INPUT NAME$:IF N
AME$="C:" OR NAME$="C" THEN GOTO 4100
4030 OPEN #1,4,0,NAME$
4040 POKE 850,7:POKE 852,PEEK(88):POKE
 853,PEEK(89):POKE 856,128:POKE 857,12
:REM SET UP IOCB #1
4050 I=USR(ADR(MOVE$))
4060 FOR I=0 TO 8:GET #1,A:POKE 704+I,
A:NEXT I
4070 CLOSE #1:POKE 54286,192:TRAP 4000
0:RETURN
4100 ? "       HIT RETURN    ":OPEN #1,4,12
9,"C:":GOTO 4040:REM SHORT IRG FOR CAS
SETTE
5000 ? CHR$(125);"   SAVE TO WHAT FILE?
 ?":TRAP 5070
5010 ? " GIVE NAME eg. D1:XX.Y or C:"
5020 ? "FILE NAME ";:INPUT NAME$:IF NA
ME$="C:" OR NAME$="C" THEN GOTO 5100
5030 OPEN #1,8,0,NAME$
5040 POKE 850,11:POKE 852,PEEK(88):POK
E 853,PEEK(89):POKE 856,128:POKE 857,1
2:REM SET UP IOCB #1
5050 I=USR(ADR(MOVE$)):REM MULTIPLE OF
 128 BYTES MUST BE MOVED
5060 FOR I=0 TO 8:PUT #1,PEEK(704+I):N
EXT I:REM COLOR REGISTERS LAST
5070 CLOSE #1:POKE 54286,192:TRAP 4000
0:RETURN
5100 ? " HIT RETURN AFTER PREPARING SA
VE TAPE ";
5101 TRAP 5120:LPRINT :REM INITIALIZE
5120 TRAP 5070
5130 OPEN #1,8,128,"C:":REM SHORT IRG
5150 GOTO 5040
6000 ? CHR$(125);"GIVE REGISTER # TO C
HANGE":TRAP 6090
6010 ? "BETWEEN 704 AND 712 INCLUSIVE"
6020 ? "  REGISTER ";:INPUT REG
6030 IF REG<704 OR REG>712 THEN 6010
6040 ? " GIVE BRIGHTNESS 0 - 15 ";:INP
UT BRIGHT
6050 IF BRIGHT<0 OR BRIGHT>15 THEN 604
0
6060 ? " GIVE COLOR  0 - 15 ";:INPUT C
OL
6070 IF COL<0 OR COL>15 THEN 6060
6080 POKE REG,COL*15+BRIGHT
6090 TRAP 40000:RETURN
```

(CHECKSUM DATA next page.)

CHECKSUM DATA

(See pp. 20-24.)

```
5 DATA 832,23,848,232,325,260,259,776,
269,367,275,670,321,256,578,6291
23 DATA 636,265,379,195,220,919,45,14,
757,728,257,703,920,999,421,7458
38 DATA 954,635,280,602,914,728,279,75
5,453,66,102,664,999,52,945,8428
53 DATA 335,958,250,763,295,283,527,26
1,117,132,147,594,276,798,978,6714
68 DATA 793,317,263,810,217,272,552,29
9,330,506,775,739,489,268,886,7516
83 DATA 442,634,12,218,507,193,292,945
,405,944,271,279,576,375,491,6584
98 DATA 171,700,163,107,976,796,15,593
,275,541,657,548,586,608,667,7403
210 DATA 627,633,634,571,650,661,664,6
25,862,804,771,599,617,81,966,9765
525 DATA 246,276,598,391,892,655,666,9
,681,641,552,995,657,821,104,8184
3000 DATA 852,391,785,550,843,166,794,
667,108,943,692,367,133,797,889,8977
5030 DATA 808,357,333,116,694,843,220,
698,668,732,646,603,158,388,794,8058
6050 DATA 949,863,31,683,923,3449
```

•

Assembly Listing.

```
0100 ;   FILL ROUTINE USING CIO
0110 ;
0120 ;  CIO EQUATES USED
0130 ;
0140 CIO =   $E456
0150 ICCOM = $0342
0160 ICBLEN = $0348
0170 CGBINR = $07     ; GET BINARY RECORD
0180 CPBINR = $0B     ; PUT BINARY RECORD
0190 ;
0200 BUFA =  $CB      ; ADRESSES FOR IND,Y
0210 BUFB =  $CD      ; ADDRESSING FOR
0220 BUFC =  $CF      ; PIXEL COORDINATE
0230 BUFD =  $D4      ; STORAGE
0240 ;
0250 XPOS =  $55      ; X COORDINATE see OS
0260 YPOS =  $54      ; Y COORDINATE  "  "
0270 ;
0280 COLOR = $D1      ; COLOR TO PLOT
0290 ;
0300 ; THE FOLLOWING LOCATIONS CAN ONLY
0310 ; BE USED TEMPORARILY, BECAUSE THEY
0320 ; ARE RESERVED FOR THE FLOATING POINT
0330 ; ROUTINES.
0340 ;
0350 COLOVER = $E6    ; COLOR TO COVER
0360 COUNTNEW = $E7   ; # NEW PIXELS
0370 COUNTOLD = $E8   ; # PIXELS TO TEST
0380 ;                  AROUND
0390 XPSTOR = $E9     ; STORE ORIGINAL XPOS
0400 YPSTOR = $EA
0410 XP  =   $D6      ; X COORDINATE
0420 YP  =   $D7      ; Y COORDINATE
0430 TEM =   $EB      ; TO STORE Y INDEX
0440 ;
0450 ;  THE FILL ROUTINE USES PAGE 6
0460 ;
0470     *=  $0600
0480 ;
0490     PLA
0500     PLA
0510     PLA          ; THE COLOR TO PLOT
0520     STA COLOR
0530     LDA $06FE    ; BUFD ADDRESS STORED
0540     STA BUFD     ; HERE BECAUSE OF THE
0550     LDA $06FF    ; LIMITED FREE SPACE
0560     STA BUFD+1   ; IN PAGE 0
0570 ;
0580     LDA #0
0590     STA COUNTNEW ; INITIALIZATION
0600     LDY #1       ; ONE PIXEL FIRST TIME
0610     LDA XPOS     ; FROM O.S.
0620     STA (BUFA),Y
0630     STA XPSTOR
0640     STA XP       ; FOR THE FIRST LOCATE
0650     LDA YPOS     ; FROM O.S.
0660     STA (BUFB),Y
0670     STA YPSTOR
0680     STA YP       ; FOR THE FIRST LOCATE
0690     STY COUNTOLD ; ONE PIXEL
0700     JSR LOCATE   ; GET COLOR TO COVER
0710     STA COLOVER
0720     CMP COLOR    ; ITSELF ?
0730     BNE PRELOOP
0740     RTS          ; IF ITSELF QUIT OR GET
0750 ;                  INFINITE LOOP !
0760 ;
0770 PRELOOP JSR PLOT
0780 LOOP LDA (BUFA),Y
0790     STA XP       ; X FOR CURRENT PIXEL
0800     LDA (BUFB),Y
0810     STA YP       ; Y FOR CURRENT PIXEL
0820 ;
0830 LOOP0 INC XP     ; TEST TO RIGHT
0840     LDA XP
0850     CMP #80      ; RHS LIMIT TO SCREEN
0860     BCS LOOP1
0870     JSR LOCATE   ; IF WITHIN SCREEN
0880 ;                  THEN SEE IF THE PIXEL
0890 ;                  CONTAINS THE COLOR TO
0900 ;                  BE OVERWRITTEN. THE
0910 ;                  LOCATE ROUTINE
0920 ;                  CONTAINS THE COMPARE.
0930 ;
0940     BNE LOOP1
0950     JSR KEEP     ; PLOT IT AND MARK IT
0960 ;                  FOR ITS OWN TEST
0970 ;                  NEXT TIME THROUGH THE
0980 ;                  LIST OF LOCATIONS.
0990 ;
1000 LOOP1 DEC XP
1010     DEC XP
1020     LDA XP
1030     CMP #255     ; CHECK SCREEN LHS
1040     BEQ LOOP2
1050     JSR LOCATE
1060     BNE LOOP2
1070     JSR KEEP
1080 ;
1090 LOOP2 INC XP
1100     INC YP       ; TEST BELOW
1110     LDA YP
1120     CMP #80      ; BOTTOM OF SCREEN
1130     BCS LOOP3
1140     JSR LOCATE
1150     BNE LOOP3
1160     JSR KEEP
1170 ;
1180 LOOP3 DEC YP
1190     DEC YP
1200     LDA YP
1210     CMP #255
1220     BEQ LOOP4
1230     JSR LOCATE
1240     BNE LOOP4
1250     JSR KEEP
1260 ;
1270 LOOP4 DEC COUNTOLD
1280     BEQ DONELOOP ; ALL POINTS DONE
1290     INY          ; ELSE DO NEXT POINT
1300     JMP LOOP
1310 ;
1320 DONELOOP LDY COUNTNEW
```

```
1330      BEQ RETURN0 ; IF NO NEW POINTS
1340 ;                  IN THE COLOR TO
1350 ;                  COVER REMAIN
1360 ;
1370      STY COUNTOLD ; THIS BECOMES THE
1380 ;                   NEW # PIXELS TO
1390 ;                   PLOT AND TEST
1400 ;
1410 TRANSFER LDA (BUFD),Y
1420      STA (BUFB),Y
1430      LDA (BUFC),Y
1440      STA (BUFA),Y
1450      DEY
1460      BNE TRANSFER ; MOVE BUFFERS
1470      LDY #1
1480      LDA #0
1490      STA COUNTNEW ; INITIALIZE
1500      JMP LOOP   ; BEGIN AGAIN
1510 ;
1520 ;    THE SUBROUTINES
1530 ;
1540 KEEP JSR PLOT    ; Y REG. IN TEM
1550      INC COUNTNEW
1560      BEQ RETURN2 ; BUFFER OVERFLOWS
1570      LDY COUNTNEW
1580      LDA XP      ; STORE THE COORD.
1590      STA (BUFC),Y ; OF THIS PIXEL
1600      LDA YP      ; FOR PLOTTING AND
1610      STA (BUFD),Y ; TESTING.
1620      LDY TEM     ; RECOVER Y REG.
1630      RTS
1640 ;
1650 LOCATE JSR POS
1660      LDA #CGBINR
1670      STA ICCOM,X
1680      JSR CIO     ;   CIO LOCATE
1690      LDY TEM
1700      CMP COLOVER ; PIXEL IN A REG.
1710      RTS
1720 ;
1730 POS STY TEM
1740      LDA XP
1750      STA XPOS    ; POSITION X
1760      LDA YP
1770      STA YPOS    ; AND Y
1780      LDX #$60    ; TO USE IN LOCATE
1790      LDA #0      ; AND IN PLOT
1800      STA ICBLEN,X ; ONE PIXEL AS
1810      STA ICBLEN+1,X ; IN ACCUM.
1820      RTS
1830 ;
1840 PLOT JSR POS
1850      LDA #CPBINR
1860      STA ICCOM,X
1870      LDA COLOR   ; THE ONE TO PLOT
1880      JSR CIO
1890      LDY TEM
1900      RTS
1910 ;
1920 RETURN0 LDA YPSTOR
1930      STA YPOS
1940      LDA XPSTOR
1950      STA XPOS
1960      RTS
1970 ;
1980 RETURN2 JSR RETURN0
1990      PLA         ; WE EXITED A SUBROUTINE
2000      PLA         ; SO POP RETURN ADDRESS
2010      RTS         ; TO BASIC
2020 ;
2030 ;    TO USE FOR ANY GRAPHICS
2040 ;    MODE OTHER THAN 8, FIX
2050 ;    THE SCREEN LIMIT TESTS.
```

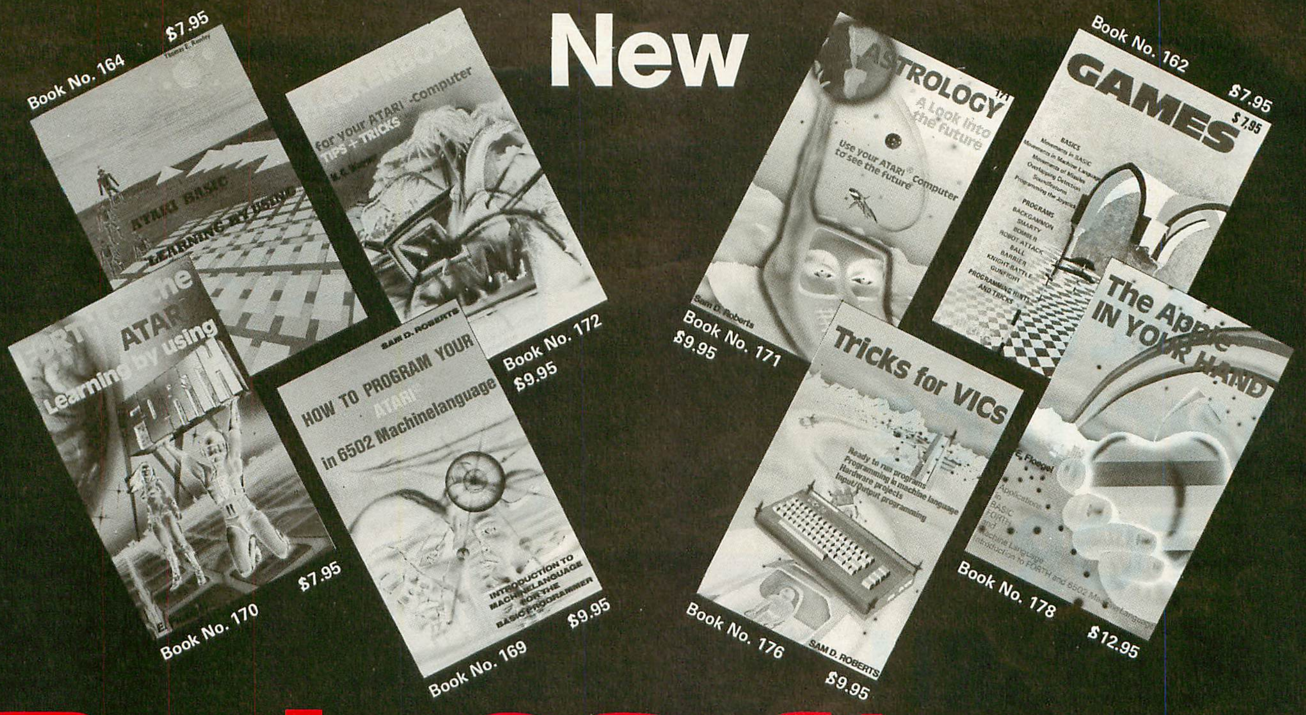
•

Transfer program.

```
0100 ;  PICTURE TRANSFER
0110 ;
0120      *=  $0600
0130 ;
0140      PLA           ; TERMS
0150      PLA
0160      STA $0355     ; ICBADR+1+$10
0170      PLA
0180      STA $0354     ; ICBADR+$10
0190      LDA #0
0200      STA $0358
0210      LDA #$20      ; MAX POSSIBLE LENGTH
0220      STA $0359
0230      LDA #7        ; C. GET BIN. REC.
0240      STA $0352
0250      LDX #$10
0260      JSR $E456     ; IGNORE ERRORS
0270      RTS
0280 ;
```

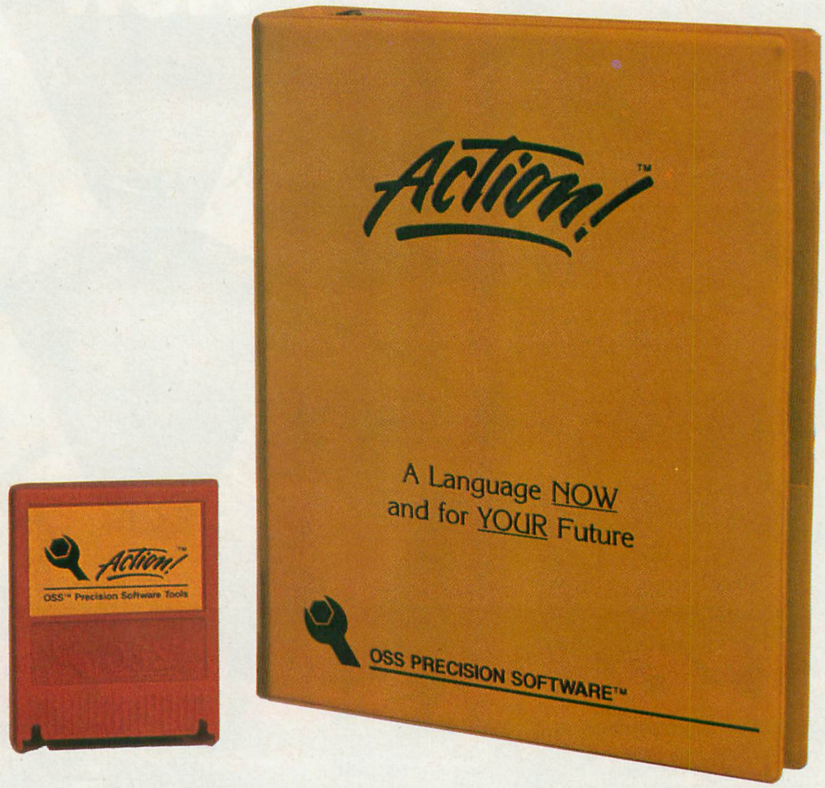
•

New



Book No. 164 $7.95
Book No. 172 $9.95
Book No. 170 $7.95
Book No. 169 $9.95
Book No. 171 $9.95
Book No. 162 $7.95
Book No. 176 $9.95
Book No. 178 $12.95
HOW TO PROGRAM YOUR ATARI in 6502 Machinelanguage
Tricks for VICs
GAMES
ASTROLOGY A Look into the future
The Apple IN YOUR HAND

A NEW LANGUAGE FOR THE ATARI!



ACTION!
by Clinton Parker
OPTIMIZED SYSTEMS SOFTWARE, Inc.
1173-D Saratoga, Sunnyvale Road
San Jose, California 95129
16K SuperCartridge $99.00

by Brian Moriarty

Atari users have a surprisingly wide selection of programming languages from which to choose. We've got three dialects of BASIC, four C compilers, eight or nine FORTHs, a pair of Pascals, PILOT, Logo, WSFN, a Lisp interpreter, numerous 6502 assemblers and a couple of hybrids like BASM and Mirth. Not bad for a "game machine," eh?

Leave it to Optimized Systems Software to come up with yet another way to tell your Atari what to do. OSS has been the leading purveyor of alternative operating systems and languages for the Atari since before I can remember. **Action!** is only the first of a whole new line of OSS products that's been causing quite a stir in the Atari underground. It's been touted as the first programming environment developed specifically for the 6502, and the fastest high-level language available for the Atari. These are pretty strong claims which, after playing with the system for several weeks, appear to be totally justified. As you are about to read.

New! Improved!

In syntax and overall structure, **Action!** bears a strong resemblance to Pascal, C and other members of the Algol family. It's a procedure-oriented language featuring global and local variables, user-definable functions, parameter passing and powerful structures like DO loops, FOR-TO, WHILE, UNTIL and IF-THEN-ELSE. Three basic data types are recognized: 8-bit BYTEs (or CHARacters), 16-bit signed INTegers and 16-bit unsigned CARDinals. The system also supports a variety of extended data types including pointers, subscripted arrays, strings and records.

AND	FI	OR	UNTIL	=	(
ARRAY	FOR	POINTER	WHILE	<>	)
BYTE	FUNC	PROC	XOR	#	.
CARD	IF	RETURN	=	>	[
CHAR	INCLUDE	RSH	-	>=	]
DEFINE	INT	SET	*	<	"
DO	LSH	STEP	/	>=	'
ELSE	MOD	THEN	&	$	;
ELSEIF	MODULE	TO	%	<	
EXIT	OD	TYPE	!	@	

Listing 1.
Reserved keywords.

Listing 1 includes all of the keywords reserved for use by the **Action!** system. These are used to declare variables, define new procedures and/or functions and

to control the operation of the compiler. BASIC veterans will note with alarm the total lack of keywords that do interesting things in and of themselves, like SETCOLOR or DRAWTO. They're missing for a very good reason. Unlike BASIC, **Action!** does not limit your programming to a limited number of safe little commands. It invites you (indeed, *forces* you) to invent the commands you need to solve problems yourself. The keywords in **Listing 1** are the tools the system gives you to, in effect, write your own language. If this prospect doesn't excite you, maybe BASIC has been holding your hand for too long.

Print	PrintE	PrintD	PrintDE	PrintB	PrintBE
PrintBD	PrintBDE	PrintC	PrintCE	PrintCD	PrintCDE
PrintI	PrintIE	PrintID	PrintIDE	Put	PutE
PutD	PutDE	InputS	InputSD	InputMD	Open
Close	XIO	Note	Point	Graphics	SetColor
Plot	DrawTo	Fill	Position	Sound	SndRst
SCopy	SCopyS	SAssign	StrB	StrC	StrI
Break	Error	Zero	SetBlock	MoveBlock	

Listing 2.
Library procedures.

Don't get the impression that **Action!** leaves you completely on your own, though. The cartridge includes a library of useful I/O, graphics and system-level routines that you can use to start building more elaborate programs. **Listings 2** and **3** will give you an idea of what's available. The resemblance of many **Action!** library words to Atari BASIC commands is intentional; the kindly folks at OSS want to make your transition from BASIC to **Action!** as painless as possible. This concern for familiarity unfortunately extends to the **Action!** graphics library, which offers exactly the same (limited) access to the hardware as Atari BASIC. Other weak points of the cartridge library include inadequate control over memory allocation and a mysterious lack of support for the Atari's built-in floating point math package.

InputB	InputC	InputI	InputBD	InputCD	InputID
GetD	Locate	Paddle	PTrig	Stick	STrig
SCompare	ValB	ValC	ValI	Rand	Peek
PeekC	Poke	PokeC			

Listing 3.
Library functions.

Most of the elements in an **Action!** program are delimited by space characters — as many as you like! You don't have to keep track of line numbers, semicolons, brackets or any other nuisances that can make you feel more like a bookkeeper than a programmer. Just follow a few simple rules regarding commas and parentheses, and you're all set. **Action!**'s modern design encourages a wide-open style of program composition, with plenty of freedom regarding the use of blank lines, upper and lower-case characters, indentation, comments and other flourishes that improve readability and make coding more fun.

A four-part system.

Internally, the **Action!** system consists of four distinct modules. There's an *editor* for creating and modifying program source text, a *compiler* which translates source text into executable machine code, a *run-time library* that supports the compiled code (described above), and a *monitor* which acts as a switchboard between the other three modules and (if you're using a disk drive) DOS.

A very important distinction between **Action!** and every other compiled language for the Atari is that these modules do not have to be loaded in separately from disk. All four are tucked away inside the SuperCartridge, safe from accidental erasure and ready whenever you need them. Further, the system is arranged so that your source text and compiled code can reside in memory at the same time. This self-contained design combines the performance of a compiled language with a degree of interactiveness usually associated with an interpreter. A stroll through the modules will show you what I mean.

The editor.

Somebody at OSS once told me that the text editor in the **Action!** cartridge was originally going to be marketed by itself as a word processor. It isn't hard to believe. There are so many features and options in the **Action!** editor that I can only touch on the most interesting here.

Action!'s editor uses your TV as a virtual window into a text area that can extend well beyond the edges of the screen. Unlike the standard Atari screen editor, you can type up to 240 characters on a single line with no cursor wraparound. How? When your cursor reaches the right edge of the screen, the line you're working on (and *only* that line) starts to coarse-scroll to the left. You can keep right on typing until a buzzer informs you that you've reached the rightmost position in that line — the right "edge" of the text window. Move your cursor back towards the left, and the line scrolls to the right until you hit the left edge of the window. This design neatly eliminates the usual confusion between "logical" and "physical" lines of text.

Hitting CTRL/SHIFT/">" or "<" instantly moves you to the rightmost or leftmost character in the current line, respectively. You can also change the maximum width of the text window to any convenient value, such as how many characters will fit on your printer.

The **Action!** editor allows you to create a second text window, co-resident in memory but otherwise completely independent from the main window. The 2-window editing mode is represented visually by a split screen, with the bottom half of the image devoted

to the auxiliary window. You can jump back and forth between the two windows and transfer blocks of text if desired; the editor remembers where you were working in each window and automatically returns you to that point when you return. Additionally, you can save, load or delete text in one window without disturbing the contents of the other. That means, for example, that you could load a library of routines into the auxiliary window, review them and copy the ones you need into your main program, which has been in full view the whole time! Sure beats LISTing and ENTERing lines of BASIC, doesn't it?

Other noteworthy capabilities of the **Action!** editor include global search and replace, instant access to the beginning or end of a file and the ability to delete, move and copy selected blocks of text. The block move and copy functions are implemented so nicely that I have to tell you about them. When you hit the SHIFT/DELETE keys, the line you're working on disappears, just as with the Atari screen editor. But the line isn't gone forever. It's being held in a buffer, waiting to be moved or copied to anywhere else in your text window(s). Simply move the cursor to a likely spot and hit CTRL/SHIFT/"P" (for paste) to dump the contents of the buffer. Several adjacent lines of text can be sent to the buffer by repeatedly "deleting" them with SHIFT/DELETE. **Action!**'s method of picking up and dropping blocks of text feels very natural if you're used to the Atari screen editor, and it also eliminates the annoyance of losing a line of work by accidentally hitting SHIFT/DELETE. Incidentally, you can automatically undo any changes you have made to a line of text by hitting CTRL/SHIFT/"U" before leaving the line. Luxurious.

Before you toss out your **AtariWriter** cartridge, let me point out a couple of small but irritating problems in the **Action!** editor. There's a feature called *tagging* which allows you to mark any location in your text by assigning it a unique one-character identifier. You can later return to that point in the text at any time by calling its ID code. It's a good idea that, unfortunately, isn't pulled off particularly well. If you set a tag in a line and change even a single character in that line, the tag disappears. This restriction (which is documented) considerably reduces the usefulness of the tagging option, to say the least.

My other gripe is with the way the cursor appears to flash and jump around the screen when it is being moved up or down, as if it isn't sure where to go next. The solid command line on the bottom of the screen also seems to jerk

occasionally as you cursor around. Minor cosmetic points, perhaps, but an unstable cursor seems out of place in this otherwise superb little text editor.

The monitor.

After you've put the finishing touches on an **Action!** program and saved it out to disk, what next? Press the CTRL/SHIFT/"M" keys simultaneously and you'll find yourself staring at a barren white bar across the top of your screen. This is **Action!**'s monitor, the central interface between the editor, compiler, machine and user.

Monitor functions are invoked by typing a one-character code letter. You can select various compilation options, save and load compiled programs, examine the values of variables and memory locations and trace the execution of your programs. You can even use the X (execute) directive to interactively test almost any procedure or function. This capability is very unusual (and useful) in a compiled programming language.

The compiler.

Unlike Atari BASIC, which compiles each line of program text as it is typed, **Action!** requires that your program be explicitly translated into machine code before it can be executed. This isn't nearly as formidable as it sounds. All you have to do is type the letter C from within the **Action!** monitor.

The compiler accepts source text from either the editor (default), or from a text file saved onto cassette or disk. If you've been using both text windows, **Action!** will compile only the text in the window you last edited. Compilation is almost unbelievably rapid, especially when the source is the editor. I've never seen **Action!** take more than a few seconds to compile even a fairly large program that was in the editor. Small programs are compiled before you take your finger off the RETURN key. You can optionally instruct the compiler to list each line of source text to the screen or a printer as it is being compiled. This slows the compilation considerably, however.

A compile error causes the system to display the line where the error occurred, along with an error message number. Surprisingly for an OSS product, there are no English error messages. If you re-enter the editor after a compile error, you'll find the cursor obligingly positioned over the questionable spot in your text.

Successfully compiled code is executed by typing the letter R (run) from within the **Action!** monitor. If you're accustomed to the leisurely pace of Atari BASIC, get ready for a shock. OSS isn't kidding when they say **Action!** is fast.

How fast is fast?

Execution speed is very important to Atari programmers. Why? Because much of the software written for the Atari relies heavily on graphics, where a few extra machine cycles in the wrong place can make the difference between a spectacular special effect and an interesting but unmarketable demo. High speed isn't likely to hurt a non-graphics program, either. This is in accordance with Moriarty's Maxim: *It is much easier to slow down a computer program than it is to speed it up.*

A number of attempts have been made to devise a universal method for comparing the speed performance of computer languages and hardware. In September of 1981, *Byte* magazine published an iterative number-crunching algorithm called the **Sieve of Eratosthenes**, which calculates all of the 1,899 prime numbers between 3 and 16,384.* The **Sieve** has since become the informal industry standard for clocking the speed of microcomputer languages.

Listing 4 is an implementation of the **Sieve** in Atari BASIC. It requires 19,490 jiffies or approximately 5½ minutes to execute on an unmodified 48K Atari 800 system. I recognize that **Listing 4** is not the most efficient way to write the **Sieve** in Atari BASIC, but it is the clearest and most portable way, and that's what counts in this application. You might like to try rewriting the **Sieve** for better speed performance. I've achieved improvements of better than 30% with tricky recoding.

---

*Jim Gilbreath, "A High-Level Language Benchmark." *Byte*, VI, 9 (September 1981), pp. 180-198.

Listing 4.

```
10 REM * ERATOSTHENES SIEVE
11 DIM FLAG$(8191)
12 POKE 559,0
13 POKE 19,0:POKE 20,0
14 COUNT=0
15 FOR I=1 TO 8191
16 FLAG$(I,I)="T"
17 NEXT I
18 FOR I=0 TO 8190
19 IF FLAG$(I+1,I+1)="F" THEN 27
20 PRIME=I+I+3
21 K=I+PRIME
22 IF K>8190 THEN 26
23 FLAG$(K+1,K+1)="F"
24 K=K+PRIME
25 GOTO 22
26 COUNT=COUNT+1
27 NEXT I
28 TIME=PEEK(20)+256*PEEK(19)
29 POKE 559,34
30 ? COUNT;" PRIMES IN"
31 ? TIME;" JIFFIES"
```

•

CHECKSUM DATA
(See pp. 20-24.)

```
10 DATA 941,347,921,5,772,308,90,393,3
14,892,919,849,588,596,860,8795
25 DATA 623,689,395,581,796,17,419,352
0
```

•

Although I love standards, I don't like the **Sieve**. It's not easy for beginners to understand, it takes too long (in BASIC, anyway), and it doesn't test the Atari under real-world conditions, with lots of 6502 processor time being "stolen" by Antic for video DMA. I wanted a benchmark that anybody could appreciate, operating under the kind of DMA conditions an Atari program is likely to find itself up against.

Back in Issue 11, I devised a little program that fills a GRAPHICS 24 screen with color, one byte (eight pixels) at a time. It was used to compare a couple of BASIC compilers at the time, but it's equally valid in any run-time environment. My definitive BASIC implementation of this test appears in **Listing 5**. **Screen-Fill**, as the program shall henceforth be known, executes in 4025 jiffies or about 67 seconds on a 48K 800. (Again, improvements are possible, but for the sake of clarity let's stick to **Listing 5**.) I'll be using **Screen-Fill** in conjunction with the **Sieve** to judge the performance of every new language I review from now on. So let it be written; so let it be done.

Listing 5.

```
10 REM * SCREEN-FILL BENCHMARK
11 GRAPHICS 24
12 POKE 19,0:POKE 20,0
13 SCREEN=PEEK(88)+256*PEEK(89)
14 FOR I=0 TO 31
15 FOR J=0 TO 239
16 POKE SCREEN+J,255
17 NEXT J
18 SCREEN=SCREEN+240
19 NEXT I
20 TIME=PEEK(20)+256*PEEK(19)
21 GRAPHICS 0
22 PRINT TIME;" JIFFIES"
```

CHECKSUM DATA
(See pp. 20-24.)

```
10 DATA 206,5,2,185,233,103,695,394,78
6,399,557,157,527,4249
```

•

OSS includes a implementation of the **Sieve** benchmark in their **Action!** documentation. I rewrote the code slightly to make it match my BASIC implementation more closely; the modified program is shown in **Listing 6**. It executes in 89 jiffies or just under a second and a half. I'll save you a calculation by pointing out that the **Sieve** runs about 219 times faster in **Action!** than it does in Atari BASIC.

Listing 6.

```
BYTE RTCLOK=20, ; addr of sys timer
     SDMCTL=559 ; DMA control

BYTE ARRAY FLAGS(8190)

CARD COUNT,I,K,PRIME,TIME

PROC SIEVE()

 SDMCTL=0 ; shut off Antic
 RTCLOK=0 ; only one timer needed

 COUNT=0           ; init count
 FOR I=0 TO 8190 ; and flags
  DO
  FLAGS(I)='T
  OD

 FOR I=0 TO 8190
  DO
  IF FLAGS(I)='T THEN
    PRIME=I+I+3
    K=I+PRIME
    WHILE K<=8190
      DO
      FLAGS(K)='F
      K==+PRIME
      OD
    COUNT==+1
  FI
  OD

 TIME=RTCLOK ; get timer reading
 SDMCTL=34   ; restore screen

 PRINTF("%E %U PRIMES IN",COUNT)
 PRINTF("%E %U JIFFIES",TIME)

RETURN
```

•

Unconvinced? **Listing 7** is an **Action!** implementation of **Screen-Fill**. This demanding little gem executes in 32 jiffies (slightly more than half a second), or 126 times faster than its BASIC counterpart under maximum DMA handicap. And if you cheat by replac-

ing the nested FOR-TO loops with an **Action!** SETBLOCK procedure in the form:

```
SETBLOCK(SCREEN,7680,255)
```

you'll obtain an execution time of just five jiffies. This is essentially the same amount of time it takes the equivalent machine-language code to do the same job. No other high-level Atari language that I am aware of can match this kind of speed performance.

Listing 7.

```
BYTE RTCLOK=20,   ; addr of sys timer
     SAVMSCL=88,  ; lsb of screen addr
     SAVMSCH=89,  ; msb

     I,J,TIME     ; declare variables

CARD SCREEN

PROC BENCH()

  GRAPHICS(24)
  RTCLOK=0

  SCREEN=SAVMSCL+256*SAVMSCH

  FOR I=0 TO 31
    DO
    FOR J=0 TO 239
      DO
      POKE(SCREEN+J,255)
      OD
    SCREEN==+240
    OD

  TIME=RTCLOK

  GRAPHICS(0)
  PRINTF("%E %U JIFFIES",TIME)

RETURN
```

•

Pulling the wings off a butterfly.

Once I got a taste of **Action!**'s dizzying speed, I had to find out what was going on inside that demonic little cartridge. So I used the W (write object code) option of the **Action!** monitor to send a copy of the compiled **Screen-Fill** benchmark to a disk file. Then I read it back into Ralph Jones' **Ultra Disassembler** (published by Adventure International), massaged the labels and commented the code to make it correspond to the **Action!** source text, line by line. The result appears in **Listing 8**.

Assembly programmers will appreciate the extraordinary efficiency of the **Action!** compiler. The code in **Listing 8** is totally non-recursive. It uses no special stacks or indirect pointers to control the flow of execution, just pure in-line machine code with an occasional JSR into a cartridge library routine. This is "native mode" compilation at its best: simple, clean, and very, very swift. The output of a typical C or Pascal compiler looks like spaghetti by comparison.

Because compiled **Action!** programs refer to subroutines that reside inside the **Action!** cartridge, you can't run a program without the cartridge in place. This may come as a disappointment to users who want to give copies of their latest **Action!** game to friends who don't have **Action!** OSS plans to remedy this situation by offering a Personal Run-Time Package to licensed **Action!** users for around $30. It's a utility that will let you turn any **Action** program into a self-standing entity that will run with no help at all from the **Action!** cartridge, thank you. A commercial run-time package will also be offered for a one-time licensing fee of approximately $300. Both may be available by the time you read this; contact OSS directly for more information.

Another $30 will get you OSS's Programmer's Aid Disk (PAD), a collection of demonstration programs and library routines that wouldn't fit into the already crowded **Action!** cartridge. The libraries include badly-needed support for player/missile graphics, memory management and floating point math, precisely the weaknesses I noted above. The demo programs are very instructive and help to clarify some of the obscure features of the language. You even get a full-blown game program, written in **Action!** by our very own Joel Gluck.

The PAD squarely addresses many of the shortcomings of the **Action!** cartridge and documentation, and is an absolute must for all serious owners of the **Action!** system. In fact, this material ought to be included with every new system sold, even if it means bumping up the price a bit.

You can bank on it.

The 16K **Action!** "SuperCartridge" is a technically interesting device in and of itself. It employs a hardware technique called bank-selecting to make itself "look" like an 8K cartridge. This gives you access to the 8K of RAM between $8000-$9FFF that is de-selected and thus rendered useless by a conventional 16K cartridge, such as **AtariWriter**.

The bottom half of the SuperCartridge ($A000-$AFFF) is divided into three independently addressable 4K banks of ROM, which are automatically switched in and out depending on what part of the system is in use. If your Atari has 48K or more memory, it's even possible to address the 4K bank of RAM that resides "under" this half of the cartridge. OSS's new **DOS XL** operating system takes advantage of this capacity in a most ingenious manner. Look for a report in a future issue.

The bank-select cartridge is a nearly ideal home for Atari software. It gives the cartridge designer a full 16K to work with, enough room for plenty of bells and whistles. It gives the user an instant-loading, highly reliable environment with up to 40K of workspace. And because three of the memory banks occupy the same 4K address range, a bank-select cartridge is very difficult to pirate. Let's hope that more manufacturers start taking advantage of bank-selecting to enhance the value and security of their products.

Advice and admiration.

I'm sorry to report that the *Action! Reference Manual* doesn't do the language justice. In a commendable attempt to satisfy beginners and experts alike, the *Manual* suffers from lack of confidence, uncertain organization and a shortage of good, hard technical data. Thank goodness for the numerous sample programs, which communicate a lot more about the system than the text surrounding them.

Having once written the manual for a new (and mercifully obscure) programming language, I can appreciate the difficulties involved in deciding how much needs to be said, to whom, and in what order. Nevertheless, a new language can only be as good as its documentation. Until somebody sits down, rolls up his or her sleeves and writes an authoritative book about **Action!**, it will have a hard time attaining the wide acceptance it so obviously deserves. I conclude this diatribe by acknowledging that the latest edition of the *Reference Manual* (in the small yellow notebook) shows a marked improvement over the first release.

The **Action!** cartridge itself has gone through a couple of changes since its first appearance in August 1983. You can tell which version you have by using the "?" (display memory) command in the monitor to examine cartridge address $B000. If this byte equals $31 hex, you have the original Version 3.1. A value of $33 indicates Version 3.3, in which a number of minor 3.1 bugs have been corrected. The final version is 3.6 ($36 at $B000), which should be ready soon after you read this. OSS has always been very good about maintaining their products, so you shouldn't have any trouble getting an upgrade if you need one. Consult OSS for prices and availability.

I hope my kvetching about the documentation doesn't scare you away. If sensible, structured code and edge-of-the-art speed are what you crave in a high-level language, **Action!** is exactly what you need. OSS's hideous orange cartridge joins the ranks of **valFORTH**, **Omnimon!**, **ABC** and **MAC/65** as one of the most valuable development tools ever published for the Atari. Congratulations and thanks to Clint Parker and OSS for bringing us such an advanced product. You can expect to see plenty of support for this exciting new language in future issues of **ANALOG.** □

Listing 8.

```
0100 ;           DISASSEMBLY OF COMPILED
0110 ;           ACTION! SCREEN-FILL
0120 ;           BENCHMARK (LISTING 7)
0130 ;           ----------------------
0140 ;
0150 ;          DEFINE ADDRESS CONSTANTS
0160 ;          ----------------------
0170 RTCLOK = 20
0180 SAVMSCL = 88
0190 SAVMSCH = 89
0200 ;
0210 ;         GLOBAL VARIABLE STORAGE
0220 ;         ----------------------
0230     *=  ORIGIN
0240 I   *=  *+1      ; reserve 1 byte for
0250 J   *=  *+1      ; each BYTE variable,
0260 TIME *= *+1
0270 SCREEN *= *+2    ; 2 bytes for CARDs
0280 ;
0290 ;         PROC BENCH()
0300 ;         ------------
0310     JMP START
0320 ;
0330 ; If our procedure used local variables,
0340 ; they would have been stored here.
0350 ; That's why the above JMP is included.
0360 ;
0370 ;         GRAPHICS(24)
0380 ;         ------------
0390 START
0400     LDA #24
0410     JSR GRAPHICS
0420 ;
0430 ;         RTCLOK=0
0440 ;         --------
0450     LDY #0
0460     STY RTCLOK
0470 ;
0480 ;         SCREEN=SAVMSCL+256*SAVMSCH
0490 ;         --------------------------
0500     LDA #0
0510     STA TEMP1+1
0520     LDA SAVMSCH ; move SAVMSCH into
0530     STA TEMP1   ; TEMP1
0540     LDA # >256  ; msb of multiplier
0550     TAX
0560     LDA # <256  ; lsb
0570     JSR MULTIPLY
0580     STA TEMP4
```

```
        TXA          ; save (256*SAVMSCH)
        STA TEMP4+1  ; into TEMP4
;
        CLC
        LDA SAVMSCL  ; add SAVMSCL to
        ADC TEMP4    ; (256*SAVMSCH) and
        STA SCREEN   ; store in SCREEN
        LDA #0
        ADC TEMP4+1
        STA SCREEN+1
;
;            FOR I=0 TO 31 DO
;            ----------------
        LDY #0
        STY I        ; init I-loop
ILOOP
        LDA #31
        CMP I        ; reached limit yet?
        BCS JINIT    ; no - do another J-loop
        JMP GETIME   ; else get timing
;
;           FOR J=0 TO 239 DO
;           -----------------
JINIT
        LDY #0
        STY J        ; init J-loop
JLOOP
        LDA #239
        CMP J        ; reached limit yet?
        BCS DOPOKE   ; no - poke another byte
        JMP ADD240   ; else update SCREEN
;
;          POKE(SCREEN+J,255)
;          ------------------
DOPOKE
        CLC
        LDA SCREEN   ; add SCREEN to
        ADC J        ; J, and
        STA TEMP2    ; save in TEMP2
        LDA SCREEN+1
        ADC #0
        STA TEMP2+1
;
        LDY #255
        LDX TEMP2+1
        LDA TEMP2    ; poke (SCREEN+J) with
        JSR POKE     ; a 255
;
;          OD (for J loop)
;          ---------------
        INC J
        JMP JLOOP
;
;            SCREEN==+240
;            ------------
ADD240
        CLC
        LDA SCREEN   ; add SCREEN and
        ADC #240     ; 240; store result
        STA SCREEN   ; in SCREEN
        LDA SCREEN+1
        ADC #0
        STA SCREEN+1
;
;           OD (for I loop)
;           ---------------
        INC I
        JMP ILOOP
;
;            TIME=RTCLOK
;            -----------
GETIME
        LDA RTCLOK
        STA TIME
;
;            GRAPHICS(0)
;            -----------
        LDA #0
        JSR GRAPHICS
;
;            PRINTF("%E %U JIFFIES",TIME)
;            ----------------------------
        JMP OVER     ; skip over in-line string
STRING
        .BYTE 13     ; length of string
        .BYTE "%E %U JIFFIES"
OVER
        LDA #0       ; msb of TIME
        STA TEMP3    ; into TEMP3
        LDY TIME     ; lsb into Y
        LDX # >STRING ; msb of string addr
        LDA # <STRING ; lsb
        JSR PRINTF
;
;            RETURN
;            ------
        RTS          ; from procedure
;
        RTS          ; back to Action! monitor
```

•

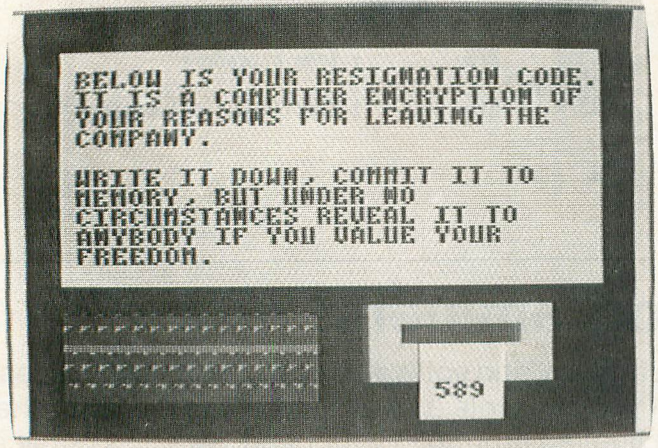
PRISONER II
by David Mullich
EDU-WARE SERVICES, Inc.
P.O. Box 22222
Agoura, California 91301
48K Two-Disk Set $39.95

by C.J. Thorns

A top-level British secret agent wants "out." Slamming his resignation onto the Company desk, he streaks back to his flat in a homebuilt Lotus 7, intending to pack his suitcase and escape to a quiet vacation in the sun. Then he hears the hiss of gas and, fighting against it, slips unconscious to the floor.

Waking, the agent looks around at his familiar bedroom. Was it a dream? He opens the window curtains and stares with astonishment at a baroque landscape of candy-colored Italian cottages. This isn't London!

So began *The Prisoner*, the unforgettable 1968 television series starring Patrick McGoohan that became a legend among fantasy/science fiction addicts. Inhabitants of the mysterious "Village" in which McGoohan's character found himself were identified only by number. Some were prisoners, others not; there was no way of knowing if an inmate was a former agent or a "custodian" who, using various psychological tricks, would persuade you into revealing "information" (the reason for the resignation?).



The Prisoner.

A few years ago, I made a pilgrimage to Portmeirion, the village in North Wales where the series had actually been filmed. As I walked the little streets, I could almost hear the ghostly Muzak playing, and I felt certain that, if I turned the next corner quickly enough, I would see a golf cart ahead of me. The only souvenir I have of that first visit is a paperback copy of *The Prisoner* by Thomas Disch. But I recently returned to the Village — through the keyboard of my Atari.

Edu-Ware Service's new **Prisoner II** is an animated graphics adventure that recreates the Village (although they have rebuilt it as an "Island") and populates it with electronic frustrations no less sophisticated than the original. The packaging describes the product as a "Science Fiction Nightmare," but that description doesn't nearly do it justice.

Prisoner comes with a confidential dossier which outlines the problems you will face. After loading one of the two disks in the set, I was given a three-digit code which, I was told, was an encrypted reason for my resignation. I then found myself at an Airport, where I attempted to take a flight to Tokyo, or Honolulu, or *anywhere* — but I found myself on a one-way trip to the Island.

After navigating an invisible maze, I entered the Village with an invitation to visit "The Caretaker" at my earliest convenience. His house wasn't hard to locate, but the door was locked. Other buildings with obscure purposes were easier to enter. An occasional step in the wrong direction sent me back to the maze — with all the passages rearranged! Climbing the fence that surrounds the Island alerted an electronic watchdog called Pax, which also sent me back to the maze.

I was able to contact an underground organization called The Brotherhood; they assured me that, if I satisfied them of my integrity, they would help me to escape. But can I trust them? If I reveal my resignation code I will lose, but at times the pressure is so great that it would almost be a pleasure to submit.

The screen of **Prisoner II** consists of one line for text entry, another for the computer's response and a high-resolution display of your current location. Sounds, colors, animation and an (at times) intentionally unresponsive keyboard combine to raise your frustration level as you move around the Island. The interiors of buildings often change when you return to them; the standard adventure trick of "save the game and see what happens" just doesn't work here! You can record your current score and posses-

sions using the STASIS command — which may or may not be available, depending on your location. When you reboot, you will be dumped back into the invisible maze. If you want a fresh start, perhaps to let a friend play, you must (unfortunately) give up by revealing your resignation code. This results in a humorous response and resets your environment.

The **Prisoner II** disks are copy-protected, so you cannot back them up. This is always a nuisance, especially in this case, where you can't write-protect the disk if you want to save your status. The program also requires you to swap disks from time to time; there is no support for multiple-drive play. I wonder how many times I can swap my master disks before they become unreadable? The only fault I could find in the documentation was that the program should be booted from Disk #1, not Disk #2 as stated.

Prisoner II is a superb package (I dare not call it a "game") that should provide weeks of entertainment. It goes far beyond the traditional "collect the right combination of treasures" adventure, and includes some diabolical arcade-like sequences to frustrate you even more. I'm told that it is possible to escape from the Island. If you succeed, you will never forget it.

Be seeing you. □

Tired of high prices, poor service and hidden charges you get from other mail order companies . . .

Try RISING SUN SOFTWARE!

ENTERTAINMENT

ARTWORX
Strip Poker (D) $25.00
S.P. Data Disks 20.00

ATARI (ROM only)
Centipede $32.00
Defender 32.00
Dig Dug 32.00
Donkey Kong 35.00
Eastern Front 35.00
Galaxian.............. 32.00
Joust................ 35.00
Missle Command 27.00
Ms. Pac-Man.......... 35.00
Pac-Man.............. 32.00
Pengo 32.00
Qix.................. 32.00
Robotron 32.00
Space Invaders 27.00
Star Raiders 32.00

BIG FIVE
Miner 2049er (R) $35.00

BRODERBUND
A.E. (D) $25.00
Arcade Machine (D)... 42.00
Choplifter (D) 25.00
Choplifter (R) 32.00
Oper. Whirlwind (D) ... 25.00
Sea Fox (D)........... 21.00
Sea Fox (R)........... 28.00
Sky Blazer (D) 28.00

BUDGECO
Raster Blaster (D).... $21.00

DATAMOST
Airstrike (D/C)....... $28.00
Bilestoad (D) 28.00
Mating Zone (D) 25.00

DATAMOST
Micropainter (D) $25.00
Pooyan (D/C)......... 21.00
Zaxxon (D/C)......... 28.00

DON'T ASK
Poker Sam (D/C) $20.00
S.A.M. (D) 42.00

EDU-WARE
Prisoner 2 (D) $28.00
Rendezvous (D)....... 28.00

ELECTRONIC ARTS
Archon (D) $28.00
Hard Hat Mack (D).... 25.00
M.U.L.E. (D) 28.00
Murder on the Zunderneuf (D) 28.00
Pinball Construction Set (D) 28.00
Worms? (D)........... 25.00

EPYX
Jumpman (D) $28.00
Jumpman Jr. (D)...... 28.00

INFOCOM (disk only)
Deadline $35.00
Enchanter 35.00
Planet Fall 35.00
Starcross 28.00
Suspended 35.00
The Witness 35.00
Zork I 28.00
Zork II.............. 28.00
Zork III.............. 28.00

INTELLIGENT STATEMENTS
Pro. Blackjack (D) ... $49.00

LIGHTNING SOFTWARE
Master Type (D) $28.00

MUSE
Castle Wolfenstein ... $21.00

ODESTA
Chess (D) $49.00
Checkers (D) 35.00
Odin (D)............. 35.00

PARKER BROTHERS
Astrochase (R)....... $35.00
Chess (R) 42.00
Frogger (R) 34.00
Popeye (R) 35.00
Q-Bert (R) 35.00
Risk (R) 42.00
Super Cobra (R) 35.00
Tutankam (R)......... 35.00

ROKLAN
Deluxe Invaders (R) .. $28.00
Gorf (D) 28.00
Gorf (R) 32.00
Wizard of Wor (R) 32.00
Wizard of Wor (D) 28.00

SIERRA ON-LINE
Crossfire (R)........ $25.00
Crossfire (D/C) 21.00
Frogger (D/C) 25.00
Mission Asteroid (D) .. 20.00
Sammy Lightfoot (R) .. 27.00
Ultima II (D).......... 42.00
Ulysses (D) 28.00
Wiz & Princess (D) 25.00

STRATEGIC SIMULATIONS
Battle for Normandie (D/C) .. $28.00
Battle of Shiloh (D/C)....... 28.00
Combat Leader (D)...... 28.00
Cosmic Balance (D/C) 28.00
Cosmic Balance II (D/C) 28.00
Cytron Masters (D).... 28.00
Galactic Gladiator (D) . 28.00
Knights of the Desert (D/C0 .... 28.00
Shattered Alliance (D). 28.00
Tigers in the Snow (D/C) 28.00

SYNAPSE
Blue Max (D/C)...... $25.00
Dimension X (D/C).... 25.00
Ft. Apocalypse (D/C).. 25.00
Necromancer (D/C).... 25.00
Pharoah's Curse (D/C) 25.00
Shadow World (D/C) .. 25.00
Shamus (D/C)........ 25.00
Shamus (R) 32.00
Shamus II (D/C)...... 25.00
Survivor (D/C)........ 25.00
Zepplin (D/C) 25.00

THORN EMI
Hockey (R) $28.00
Jumbo Jet Pilot (R).... 35.00
River Rescue (R)...... 28.00
Soccer (R)........... 35.00
Submarine Commander (R)................ 35.00

HARDWARE DEALS

MODEMS
Apple Cat II ..$299.00
Micromodem II 265.00
Micromodem w/terminal prog..... 295.00
212 Apple Cat . 580.00

MONITORS
Amdek
Color I $299.00
Color II RGB ... 599.00
Color III RGB .. 399.00
RGB Card 149.00
USI
Pi1 9" Green .$119.00
Pi2 12" Green . 149.00
Pi3 12" Amber. 159.00
Pi4 9" Amber .. 129.00
Color 1400 .... 299.00

PRINTERS
C. Itoh
GX-100.......$229.00
Prowriter 399.00
Okidata
Microline 80 ... 349.00
Microline 82A . 449.00
Microline 92 ... 549.00

UTILITIES & LANGUAGES

ADVENTURE INTERNATIONAL
Diskey (D).......... $35.00

ATARI
Assembler Editor (R) . $45.00
Atari Basic (R)........ 42.00
Macro Assembler (R).. 68.00
Microsoft Basic II (R).. 68.00
PILOT (R) 60.00

DATASOFT
BASIC Compiler (D) .. $63.00
Lisp Interpreter (D) .. 70.00
Edit 6502 (R)........ 125.00

OPTIMIZED SYSTEMS
Basic A+ (D)........ $56.00
Bug 65 (D) 25.00
Mac/65 (D) 56.00

EDUCATION

ATARI
Conversational Languages
French (C)......... $42.00
German (C)......... 42.00
Italian (C) 42.00
Spanish (C)......... 42.00
Educator Kit 117.00
Inv. to Programming 1 ..... 18.00
Inv. to Programming 2 ..... 23.00
Inv. to Programming 3 ..... 23.00
Juggles House (D) 23.00
Juggles House (C) 18.00
Juggles Rainbow (D) .. 23.00
Juggles Rainbow (C) .. 23.00
My First Alphabet (D) . 27.00
Programmer Kit....... 53.00
Tough Typing (cass) .. 18.00

EDU-WARE
Compu-Read (D)..... $21.00
Compu-Read (C)...... 15.00

SPINNAKER
Face Maker (D) $28.00
Hey Diddle Diddle (D). 21.00
Kindercomp (D) 21.00
Most Amazing Thing (D)................ 28.00
Rhymes & Riddles (D). 21.00
Snooper Troops 1 (D) . 32.00
Snooper Troops 2 (D) . 32.00

RISING SUN SOFTWARE
4200 PARK BLVD.
OAKLAND, CALIFORNIA 94602
(415) 482-3391

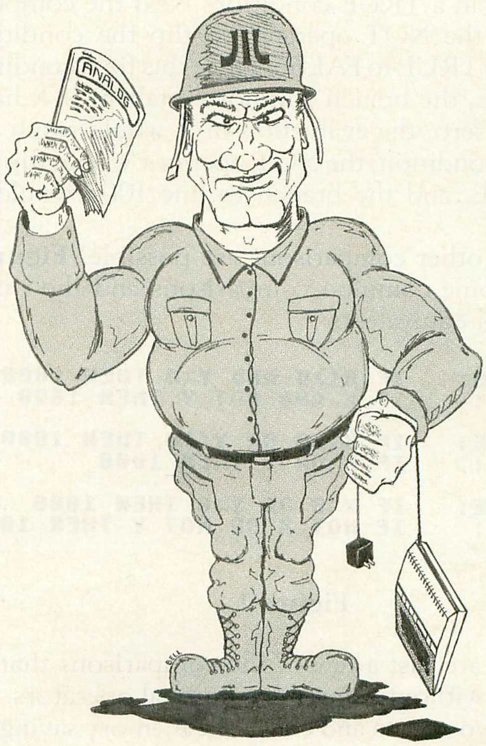
Ordering Information: We'll accept any form of payment—cash, personal check, money order, VISA/MasterCard, or C.O.D. Send cash at your own risk. Add $2.00 for UPS shipping; $3.00 for Blue Label Air. California residents add applicable sales tax. ALL orders shipped same day received. If we are out of stock on a particular item we will include a special bonus with your order when shipped.

CALL TOLL FREE 24 HOURS (ORDERS ONLY)
(800) 227-1617, ext. 114 (Outside California)
(800) 772-3545, ext. 114 (Inside California)

master charge

VISA

BASIC TRAINING



by Tom Hudson

In last issue's **BASIC Training**, we started examining the use of the IF/THEN statement in Atari BASIC. This issue, we continue our in-depth study with more examples and a useful PRINT USING subroutine.

IF/THEN statements without relational operators?

Usually, IF/THEN statements are used with relational operators. No, that doesn't mean they can be found working at the phone company, but that they use the symbols = (equal to,) < (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to) OR <> (not equal to).

In some cases, these operators are not necessary, and you will see IF/THEN statements that look like:

```
IF NUMBER THEN 4000
IF NOT NUMBER THEN 4000
IF NUMBER OR NUMBER2 THEN 4000
IF NUMBER AND NUMBER2 THEN 4000
```

These are perfectly valid comparisons. They aren't seen very often, but they are easy ways to conserve several bytes of memory as well as several keystrokes.

How does this type of comparison work? Remember that the computer deals with two conditions in comparisons: TRUE and FALSE. If there is no relational operator present, the computer will examine the expression and determine if it is zero or non-zero. If the expression is zero, the FALSE condition will be set. If the expression is not zero the TRUE condition will be set.

With this information in hand, such comparisons as:

IF X <> 0 THEN 1000
can be replaced with:
IF X THEN 1000

If the value of X is non-zero, the TRUE condition will be set and the comparison will operate the same. In addition, the new line is eight bytes shorter than the original! Eight bytes may not seem like much, but when repeated dozens of times in a program the memory savings can add up to an impressive amount.

What if we want to branch when the expression is zero? In this case we must use the NOT operator. This simply flips the result of the comparison. If the result was TRUE, the NOT operator will change it to FALSE. If the result was FALSE, the NOT operator will change it to TRUE. Using this information, it is easy to check for zero values. We simply use the same comparison as for non-zero and add the NOT operator to flip the result! A simple zero check is shown below.

IF NOT X THEN 1000

Let's assume that X has a value of 5, which is, of course, non-zero and should result in a FALSE condition when X is compared to zero. First the

program will evaluate X and find that it is non-zero, resulting in a TRUE condition. Next the computer will use the NOT operator to flip the condition, changing TRUE to FALSE. Since this final condition is FALSE, the branch will not be taken. If X had a value of zero, the evaluation of X would result in a FALSE condition, the NOT operator would flip this to TRUE, and the branch to line 1000 would be taken.

What other comparisons are possible? **Figure 1** shows some common comparisons and their non-relational equivalents.

```
BEFORE:   IF X<>0 AND Y=0 THEN 1000
AFTER:    IF X AND NOT Y THEN 1000

BEFORE:   IF X<>0 OR Y<>0 THEN 1000
AFTER:    IF X OR Y THEN 1000

BEFORE:   IF X=0 OR Y=0 THEN 1000
AFTER:    IF NOT X OR NOT Y THEN 1000
```

Figure 1.

These are just a few of the comparisons that are possible without using the relational operators. Try some of your own and check the memory savings by using the FRE(0) function.

What about strings?

So far we've only talked about numeric variables in comparisons. The IF/THEN statement can also compare alphanumeric information in strings. If you are not sure what strings are or how they work, you may want to read "Strings in ATARI BASIC" in **ANALOG** #11. This in-depth article describes how strings work and demonstrates the comparison of strings with BASIC.

String comparisons are basically the same as numeric comparisons. The only difference is the structure of the expression within the IF/THEN statement. Let's look at how string comparisons work.

In numeric comparisons, the relational operators =, <, >, etc. determine the relationship between the two values being compared.

In strings, the ATASCII code values of the characters in the strings are used to perform the comparison. Comparison proceeds from the leftmost character to the rightmost. As each character is encountered, it is converted into its ATASCII value and compared to the corresponding character in the other string. If the ATASCII values of the characters compared are the same, the two strings are considered equal. If the ATASCII value of one character is higher than the other, that string is considered greater. Conversely, if the character's ATASCII value is lower than the other, the string is considered less than the other.

Luckily, the ATASCII values are configured so that the closer the letter is to the beginning of the alphabet, the lower its ATASCII code number is. This makes it very easy to alphabetize lists of names. For example, the name "FRED" would come before "RALPH" because the ATASCII value for F (70) is lower than R (82).

Enter the listing from **Figure 2** into your computer, type in the example strings shown below the listing and observe the results.

```
10 DIM A$(5),B$(5)
20 PRINT "ENTER STRING 1";:INPUT A$
30 PRINT "ENTER STRING 2";:INPUT B$
40 IF A$=B$ THEN PRINT A$;"=";B$
50 IF A$>B$ THEN PRINT A$;">";B$
60 IF A$<B$ THEN PRINT A$;"<";B$
70 GOTO 20
```

Figure 2.

TRY	STRING 1	STRING 2
1	FRED	RALPH
2	RALPH	FRED
3	123	1234
4	12345	12345
5	HELLO	hello

Comparison number 1 will result in FRED < RALPH because, as noted earlier, the letter F is lower than the letter R in the ATASCII code sequence.

Comparison number 2 will result in RALPH > FRED. This is the same result as the first comparison, but we have just switched the order of the comparison.

Comparison number 3 will result in 123 < 1234. The first three characters of the strings are equal, but when the computer tries to compare the fourth character of each, it finds that the string "123" only has three characters, while the string "1234" has four. This causes the computer to decide that "1234" is greater than "123."

Comparison number 4 will result in 12345 = 12345, a fairly obvious result.

Comparison number 5 is very interesting. It will result in HELLO < hello. The reason for this becomes clear when one studies the ATASCII code chart in the ATARI BASIC reference manual. The upper case letters' ATASCII codes range from 65 to 90, while the lower case range from 97 to 122. As a result of this structure, lower case words will always be greater than their upper case counterparts.

Try your own comparisons with the above program and observe the results. You'll soon understand the logic of string comparisons.

USING comparisons.

Now that we've covered the decision-making features of ATARI BASIC, let's look at a program that applies most of these principles.

The program listing in **Figure 3** will provide ATARI BASIC users with a handy PRINT USING function. This function allows the printing of

numbers in specific forms, such as dollars and cents, dates and fixed decimal places.

Type this program into your computer and check it for typing accuracy with C:CHECK or D:CHECK II. Lines 10000-10250 of this listing make up the PRINT USING subroutine, and can be added to any program. Before running the program, let's look at how the PRINT USING function works.

The PRINT USING function allows the programmer to print numbers in specific formats. Instead of numbers appearing like 7654293.5, the PRINT USING function prints the numbers in an easily-readable form, such as 7,654,293.50. Dates can be shown in the standard format 12/14/83 if desired.

In order to use the PRINT USING function, two things are needed: An EDITING PATTERN and a VALUE to be edited. The editing pattern determines the printed form of the value, and can be set up in almost any form. There are several rules that must be followed when setting up an editing pattern.

1. The editing symbol "#" is used wherever a number is to appear. To show a five-digit number, the pattern "#####" would be used.
2. The editing symbol "." is used to show were the decimal point is to appear. Only one decimal point should appear in the editing pattern. To show a three-digit number with two decimal positions, the pattern "###.##" would be used.
3. The editing symbol "," is used to show where commas are to be inserted into the number. Use as many commas as necessary. To show a dollar amount in the millions of dollars, the editing pattern "#,###,###.##" would be used.
4. Other editing symbols, such as "/", can be placed in the editing pattern. They will be inserted into the edited result. An example of this is an ordinary date edit pattern, for which "##/##/##" would be used.
5. The editing pattern should be large enough to hold any values that are edited with it. For example, if the largest number in a dollar amount edit would be $1000, the pattern "#,###.##" is the smallest that should be used. If the number exceeds the pattern length, the subroutine will return all "*" for the edited number.
6. The subroutine does not round when decimals are truncated. For example, if a pattern of "###.#" is used and the number is 23.0577, the edited result will be "23.0".

The BASIC program in **Figure 3** will allow you to type ##,###.## and presss RETURN. When asked for a value to edit, type 1895.546 and press RETURN. The computer should display "EDITED VALUE = 1,895.54" and ask for another pattern. Try some of your own to get acquainted with the PRINT USING function.

How it works.

The PRINT USING subroutine is easy to use. Simply set up the string variables P$, O$ and F$ as in line 80 of the demonstration program, and place lines 10000-10250 in your program. When you want to edit a number, just place an editing pattern in P$, put the number to be edited in O$ (this is easily done with the statement "O$=STR$(NUMBER)"), and GOSUB 10000. The subroutine will edit the number and return the result to you in F$. You can then print F$ to the screen or printer as you wish.

Line 80 — Sets up the editing work string variables.

Line 90 — Accepts editing pattern from keyboard and places it in P$.

Line 100 — Accepts value to be edited from keyboard and places it in O$.

Line 110 — GOSUBS 10000 to edit the value.

Line 120 — Prints the edited value to the screen.

Line 130 — Loops back to line 90 to get another pattern and number.

Line 10030 — Saves the lengths of the pattern (LP) and the length of the number (LO), then moves spaces to the final result string (F$).

Line 10040 — This line locates the pattern decimal (if any) and saves its position in PD.

Line 10050 — If there is no decimal point in the pattern, this line sets the pattern decimal location (PD) to zero.

Line 10060 — This line locates the decimal point in the number to be edited (if any) and saves the position in OD.

Line 10070 — If there is no decimal point in the number, this line sets the object decimal position (OD) to zero.

Line 10080 — If there is a decimal in the pattern, this line sets up decimal point work pointers (PWX and PDX).

Line 10090 — If there is no decimal point in the pattern, this line sets the decimal point work pointers accordingly.

Line 10100 — If the number to be edited has a decimal point, this line sets up decimal point work pointers (OWX and ODX).

Line 10110 — If there is no decimal point in the number to be edited, this line sets the decimal point work pointers accordingly.

Line 10120 — If there is no decimal point in the pattern, the program continues editing at line 10220.

Line 10130 — This line places the decimal point in the final result string.

Line 10140 — If there are no decimal places in the object string, this line passes control to

line 10190 in order to fill the right side of the decimal point with zeroes.

Line 10150 — If the copy of the object string's decimal digits is complete, this line passes control to line 10190 to zero fill the rest.

Line 10160 — If there are no more decimal places in the pattern, this line passes control to line 10220 in order to handle the left side of the decimal point.

Line 10170 — If this position of the pattern is not a "#", this line moves what is in the pattern to the final result string, increments the pattern pointer, and goes to line 10160 to continue copying.

Line 10180 — This line moves the number in the object string to the final result string, increments the pattern pointer and object pointer, and goes to line 10150 to continue copying.

Line 10190 — This line begins the zero-fill section. If the end of the pattern has been reached, no more filling is needed and control goes to line 10220 to do the left side of the decimal point.

Line 10200 — If the current position in the pattern is not a "#," this line moves what is in the pattern to the final result string, increments the pattern pointer, and goes to line 10190 to continue zero-filling.

Line 10210 — This line places a zero in the current position of the final result string, increments the pattern pointer and loops back to line 10190 to continue zero-filling.

Line 10220 — If there are no more digits to the left of the object string's decimal point, then the edit is complete, and the subroutine RETURNs to the main program.

Line 10230 — If there are no more editing positions to the left of the pattern's decimal point, then the number is too large for the pattern. When this occurs, the program places all "*" in the final result string and RETURNs.

Line 10240 — If the current position in the pattern is not a "#," this line places the character in the final result string, decrements the pattern pointer and loops back to line 10230 to continue copying the left side of the decimal point.

Line 10250 — This line moves the number from the object string to the final result string, decrements the pattern pointer and object string pointer, the loops back to line 10220 to continue copying the left side of the decimal point.

```
10 REM ******************************
20 REM *       PRINT USING DEMO     *
30 REM *                            *
40 REM *        BY TOM HUDSON       *
50 REM *                            *
60 REM * A.N.A.L.O.G. COMPUTING #16 *
70 REM ******************************
80 DIM P$(20),O$(20),F$(20)
90 ? :? "ENTER EDITING PATTERN";:INPUT
 P$
100 ? :? "ENTER VALUE TO EDIT";:INPUT
O$
110 GOSUB 10000
120 ? :? "EDITED VALUE = ";F$
130 GOTO 90
10000 REM **************************
10010 REM * PRINT USING SUBROUTINE *
10020 REM **************************
10030 LP=LEN(P$):LO=LEN(O$):F$(1)=" ":
F$(20)=" ":F$(2)=F$:F$=""
10040 FOR X=1 TO LP:IF P$(X,X)="." THE
N PD=X:GOTO 10060
10050 NEXT X:PD=0
10060 FOR X=1 TO LO:IF O$(X,X)="." THE
N OD=X:GOTO 10080
10070 NEXT X:OD=0
10080 IF PD THEN PWX=PD-1:PDX=PD+1:GOT
O 10100
10090 PWX=LP:PDX=0
10100 IF OD THEN OWX=OD-1:ODX=OD+1:GOT
O 10120
10110 OWX=LO:ODX=0
10120 IF  NOT PD THEN 10220
10130 F$(PD,PD)="."
10140 IF  NOT OD THEN 10190:REM ***ZER
O FILL***
10150 IF ODX>LO THEN 10190
10160 IF PDX>LP THEN 10220
10170 IF P$(PDX,PDX)<>"#" THEN F$(PDX,
PDX)=P$(PDX,PDX):PDX=PDX+1:GOTO 10160
10180 F$(PDX,PDX)=O$(ODX,ODX):ODX=ODX+
1:PDX=PDX+1:GOTO 10150
10190 IF PDX>LP THEN 10220
10200 IF P$(PDX,PDX)<>"#" THEN F$(PDX,
PDX)=P$(PDX,PDX):PDX=PDX+1:GOTO 10190
10210 F$(PDX,PDX)="0":PDX=PDX+1:GOTO 1
0190
10220 IF  NOT OWX THEN RETURN
10230 IF  NOT PWX THEN FOR X=1 TO LP:F
$(X,X)="*":NEXT X:RETURN
10240 IF P$(PWX,PWX)<>"#" THEN F$(PWX,
PWX)=P$(PWX,PWX):PWX=PWX-1:GOTO 10230
10250 F$(PWX,PWX)=O$(OWX,OWX):OWX=OWX-
1:PWX=PWX-1:GOTO 10220
```

•

CHECKSUM DATA
(See pp. 20-24.)

```
10 DATA 771,338,225,293,229,755,783,19
,464,477,993,412,505,568,70,6902
10020 DATA 572,131,806,220,806,221,966
,660,949,639,288,473,620,338,330,8019
10170 DATA 575,824,336,571,332,148,399
,812,105,4102
```

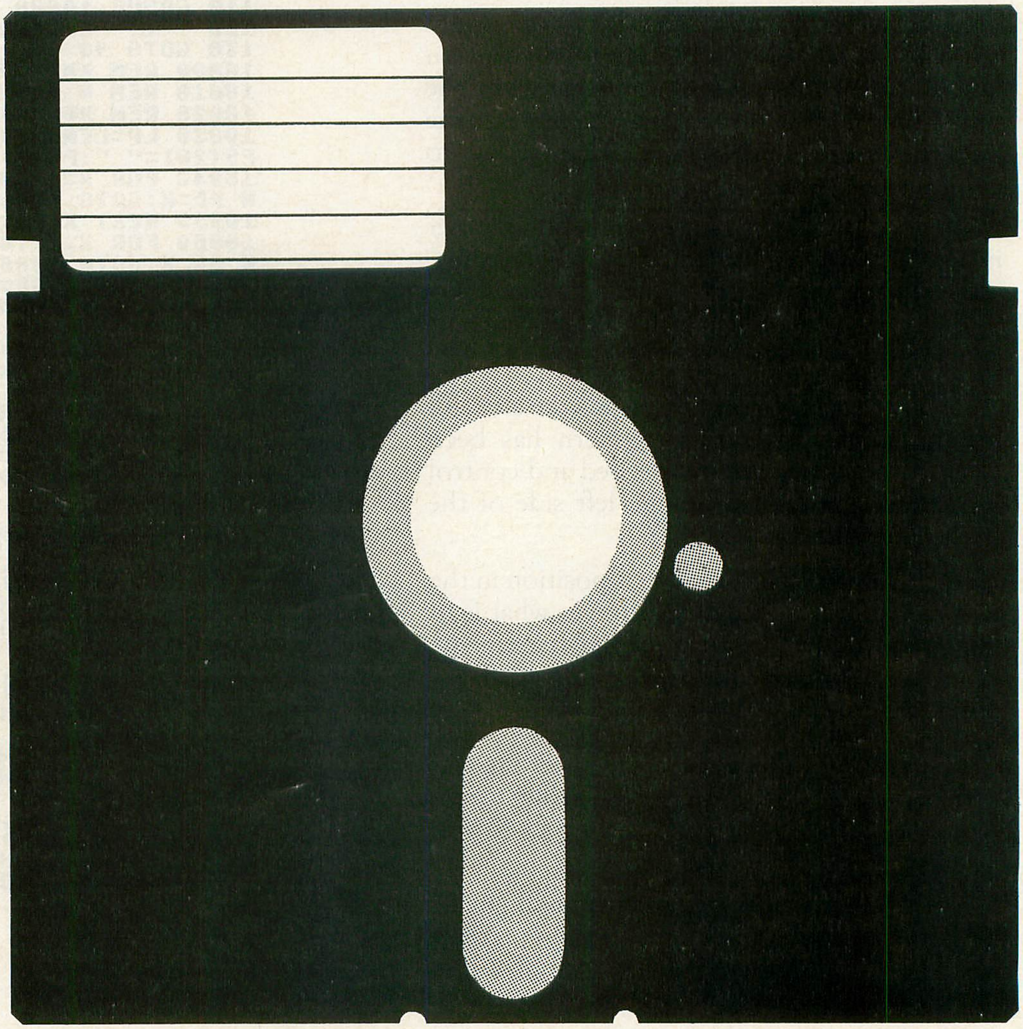
•

Hopefully, this article will give some *helpful* insights into the decision-making power of your ATARI computer system, as well as some ideas to help you make your programs more efficient. ☐

DO YOU SUBSCRIBE TO ANALOG COMPUTING ON DISK OR CASSETTE?

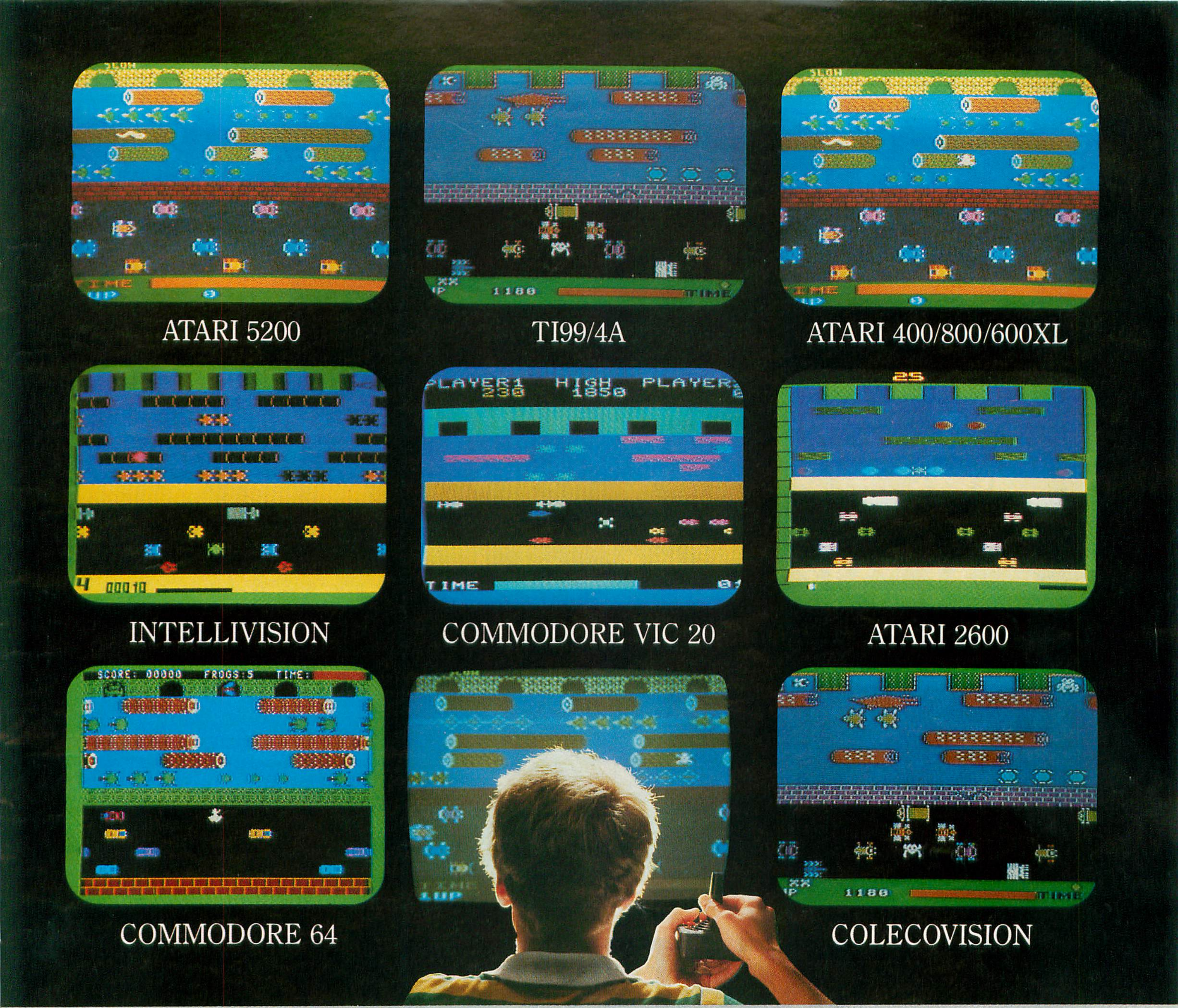
. . . if not, then you should. Since issue 1, **ANALOG**'s cassette and disk subscriptions have eliminated the need for you to spend hours typing in programs from the magazine. All of the programs in the magazine are on the cassette and disk versions. These programs are in the public domain, and are not copy-protected.

1 year (12 issues) on cassette, $90.00, disk, $130.00; ½ year (6 issues) cassette, $48.00, disk, $72.00.



To subscribe on cassette or disk, send check or money order to:

CASSETTE SUBSCRIPTION
or
DISK SUBSCRIPTION
P.O. BOX 23
WORCESTER, MA 01603

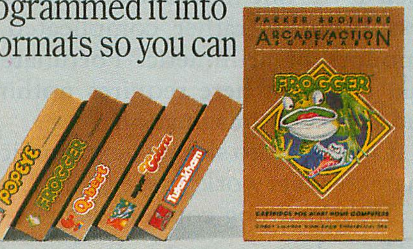


8 WAYS YOU CAN PLAY FROGGER*™ AT YOUR PAD.

FROGGER is one of the all-time great award-winning home video games. And now Parker Brothers has programmed it into all the most popular video and computer formats so you can keep things hopping in your own home.

Catch Frogger along with POPEYE,® Q*bert,™ TUTANKHAM™ and SUPER COBRA™ where you buy your video and computer games. You'll find it absolutely ribbitting.

®PARKER BROTHERS



CREATE-A-FONT

24K Cassette or Disk

by Vince Erceg

The ability to create alternate character sets is an extremely powerful feature of the Atari computer. The method for creating such a set is relatively easy. I say "relatively" because if you don't have a program such as the one that follows this article, then the process becomes rather complicated, involving large amounts of arithmetic. Fortunately, the program that I present here requires nothing but a joystick and an active imagination.

If you want to skip my explanation of how altered character sets work, just type in **Listing 1** and RUN it. The operating commands are self-explanatory.

How do you create an alternate character set? First, you must set aside some memory space (1024 bytes for a full set or 512 bytes for a half-set). Then you either copy the Atari's ROM-based character set into the space, or fill it with your own revised character data. Finally, you tell the Atari where you have placed your set with one simple POKE to location 756 ($2F4 hex).

Details—Details.

Each letter, number, or graphics figure in a character set is created by 64 individual pixels (the size of a dot in GRAPHICS 8), arranged in an eight by eight grid. The way that the actual character appears depends on which pixels are "on" or "off." **Figure 1** shows Atari's pixel representation of the letter "A."

```
              BITS
  7   6   5   4   3   2   1   0
                                  0
              X   X               1   B
          X   X   X   X           2   Y
      X   X           X   X       3   T
      X   X           X   X       4   E
      X   X   X   X   X   X       5   S
      X   X           X   X       6
                                  7
```

Figure 1.

As you can see, there are eight bits per byte and eight bytes per character. Since there are 128 separate characters in a complete character set, and each requires eight bytes, that means we must reserve 128 X 8 or 1024 bytes of RAM for our new set. If you are using GRAPHICS modes 1 or 2, then you need reserve only 512 bytes since you can only display half of the set in either of these modes at a time.

How are pixel patterns actually stored in memory? **Figure 2** shows how the position of a pixel is converted into a number.

BITS							
7	6	5	4	3	2	1	0
128	64	32	16	8	4	2	1

Figure 2.

The decimal value of each pixel position in a byte is shown under its respective bit number (0-7). The total value of the byte is obtained by adding together the decimal values of all of the bits that are "ON," or set. If all of the bits are set, the value of the byte is 128+64+32+16+8+4+2+1, or 255. (255 is the maximum number that a byte can hold, and also the maximum number that can be POKEd using BASIC.) If bits 7 and 0 alone were set, the byte value would be 128+1, or 129. Remember, when a bit is set it equals 1. It does NOT equal zero.

Referring back to **Figures 1** and **2**, let's figure out the byte values of the letter "A."

BYTE #	BITS ON	BYTE VALUE
1	0	= 0
2	16+8	= 24
3	32+16+8+4	= 60
4	64+32+4+2	=102
5	64+32+4+2	=102
6	64+32+16+8+4+2	=126
7	64+32+4+2+	=102
8	0	= 0

Now we are ready to relocate the character set. When you create your own character set and plan to store it in memory, you must reserve at least 1K (1024 bytes) for a full set and you must make sure that it begins on a memory address that is evenly divisible by 1024 (such as 38912). An easy way to do this is:

```
POKE 106,PEEK(106)-4
```

Always execute a GRAPHICS command after POKEing location 106 so that screen memory will be location 756 will be the value found in location

If you are using GR.1 or GR.2, you need only reserve 512 bytes of RAM. This area must start at an address that is evenly divisible by 512 (such as 38912, again). This memory can be reserved by:

```
POKE 106,PEEK(106)-2
```

The location of your new character set will now begin at PEEK(106)*256. The number to poke in location in 756 will be the value found in location 106. That's about all there is to it, except for a few things that must (unfortunately) be considered at all times:

1) Make sure that your new character set does not overlap player/missile data.

2) Scrolling the text window in a graphics mode scrolls up to 800 bytes past the top of memory! This effectively ruins your character set. The easiest way around this is not to scroll the text window.

3) Hitting SYSTEM RESET or executing a DOS command will reset the character set pointer at 756 to 224. You must remember to change it back to your value if any one of these things happens.

Any easy way to avert danger from steps two or three is to simply subtract an extra four pages from location 106. It wastes memory, but it relieves you of some of the worries of possibly destroying your set. The beginning of your character set will then reside at (PEEK(106)+4)*256.

One last thing. When altering a character set, remember that the characters are not in ATASCII order; rather, they are in the hardware's own internal order. To change the character data in the new set, look at the internal code table and find the character (see the *Atari BASIC Reference Manual*, page 55) and its associated value on the left. Multiply this value by eight and add it to the start of your set. The following demo program will show you what I mean.

```
10 CH=57344:REM Start of the ROM set
20 START=PEEK(106)-5
30 CHBAS=(START+1)*256
40 POKE 106,START:GRAPHICS 0
50 FOR X=0 TO 1023:REM Move ROM set
60 POKE CHBAS+X,PEEK(CH+X):NEXT X
70 POKE 756,CHBAS/256
80 FOR X=CHBAS+33*8 TO CHBAS+33*8+7
90 POKE X,255:NEXT X
```

This program changes the letter 'A' to an inverse blank (cursor).

The program.

Now that you're familiar with how a character set works, type in **Listing 1.** After pressing START and a small delay, you will be facing the main display screen. In the upper right is an 8 X 8 grid with the heart character (CTRL/comma) enlarged. Pressing the joystick button will cause the pixel under the flashing dot to be toggled. (If it's on it will be turned off, and vice-versa.) You will notice that this has a visible effect on the character located in the middle of the screen. Any action that affects the enlarged display will also affect the same character anywhere it appears. You can move the cursor around with the joystick and change the shape of any character as you please.

Next to the blown-up image of the character being "edited," you will find the main menu. Here is a summary of all options and their functions:

Edit: Select a character to edit. Cursor moves to middle of screen. Find character and press joystick button.

Reverse: Reverses current character. Same effect as Atari logo key when in BASIC.

Data: Prints values of the eight bytes in the current character and provides a look at the character when presented in a group.

Invert: Flips current character upside down.

Clear: Clears current character. Sets all values to zero.

Save Font: Saves redesigned set to tape or disk.

Load Font: Loads previously saved set from tape or disk.

From (Copy): Copies character image from selected character to current character.

To (Copy): Copies current character image to selected character.

Scroll up: Scrolls character image up.

Scroll down: Scrolls character image down.

Scroll left: Scrolls character image left.

Scroll right: Scrolls character image right.

Undo: Undoes all changes made to current character since the last Edit.

Kolor changes:(Sorry about the spelling.) Allows you to change any color register.

Quit: Exit program.

Antic 4/5: Changes display to Antic mode 4 (see below) and changes appearance of grid.

LOGO: Atari logo key copies image from ROM character set.

START: Allows you to type several characters in succession, so you can see how they look next to each other (e.g.; two characters making a car or boat).

OPTION: Disk directory.

The Antic 4/5 command changes the display list so that the characters are now in mode 4. Pressing A again will switch you back to the original display.

Most of the characters in mode 4 are indistinguishable because they are only four pixels wide instead of eight. This loss of resolution is the price that you must pay in order to get multicolored characters. Not only did the display list change, but the editing grid did also. This was done because the color that is displayed depends upon which binary number you place in each pixel. These numbers are related to the Atari's color registers as follow:

00	Background
01	Playfield 0
10	Playfield 1
11	Playfield 2

Playfield 3 can be used (binary 11), but only with an inverse character.

If you are in the "printing" (START) mode and you choose GRAPHICS 1 or 2, you may view each half of the character set alternately by repeatedly pressing RETURN.

When using "K"olor change, you must specify both color and luminance. The luminance can be any even value between 0 and 14; the color values range from 0-15, and are interpreted as follows:

0	Grey
1	Gold
2	Orange
3	Red
4	Pink
5	Purple
6	Blue
7	Blue
8	Med. Blue
9	Dk. Blue
10	Blue-Grey
11	Olive
12	Med. Green
13	Dk. Green
14	Orange Green
15	Orange

LINES added FROM ANALOG #15 - PAGE 9

GTIA colors may be significantly different. □

```
10 REM CREATE-A-FONT by Vince Erceg
20 REM
30 REM PROGRAM WRITTEN: 8/2/83
40 REM
50 GOSUB 1500:DIM FN$(14),EOR$(18),C$(
18),A$(25),XFR$(32),CLEAR$(42),B(C7),A
(C7),U(C7),ESC$(C1),LF$(C1),DN$(C1)
55 LF$=CHR$(30):DN$=CHR$(29):A$=" ":A$
(C2)=DN$:A$(C3)=LF$:A$(25)=" ":A$(C4)=
A$:C$="ERDICSLFT-=+*UKQ:A"
70 ESC$=CHR$(27):GRAPHICS C0:GOSUB 122
0:FOR I=C1 TO 18:READ J:EOR$(I)=CHR$(J
):NEXT I
75 FOR I=C1 TO 32:READ J:XFR$(I)=CHR$(
J):NEXT I
```

```
80 FOR X=C0 TO C7:B(C7-X)=INT(C2^X+0.5
):NEXT X
90 X=PEEK(C106)-C8:PM=(X+C2)*C256:POKE
 623,C1:POKE 54279,X
95 POKE 53277,C3:X=X-C8:Y=USR(ADR(XFR$
),57344,X*C256)
100 GRAPHICS C0:POKE C559,C0:POKE 538,
155:POKE C16,C64:POKE 53774,C64
110 POKE C756,X:CHBAS=X*C256:A=USR(ADR
(CLEAR$),PM-C256*C2,C256*C4):POKE 705,
148:POKE 710,C0:POKE 712,148
120 POKE 53248,168:POKE 53250,168:POKE
 53251,168:POKE 704,C255:POKE 706,C4
125 POKE 707,C6:POKE 53256,C3:POKE 532
58,C3
130 POKE 752,C1:FOR I=C0 TO C3:FOR J=C
0 TO 31:POSITION J+C4,I+C15:? ESC$;CHR
$(I*32+J);:NEXT J:NEXT I
140 POKE 53259,C3:FOR I=PM-107 TO PM-C
64:POKE I,C8:NEXT I:POKE 53253,160
150 DL=PEEK(C560)+PEEK(C561)*C256+C5*C
5:POKE DL-19,C13:POKE DL-C7,C13:POKE D
L,C13
160 FOR I=PM+281 TO PM+310 STEP C8:FOR
 J=C0 TO C3:POKE I+J,85:POKE I+J+C128,
170:POKE I+J+C4,170
170 POKE I+J+132,85:NEXT J:NEXT I:PM=P
M+C5*C5:GOSUB 180:Z=C64:POKE C559,46:G
OTO 430
180 POSITION C2,C3:? "Edit":? "Reverse
":? "Data":? "Invert"
185 ? "Clear":? "Save Font":? "Load Fo
nt":? "From (Copy)"
190 POKE 82,14:POSITION 14,C3:? "To (C
opy)":? ESC$;CHR$(156);" Scroll UP"
195 ? ESC$;CHR$(157);" Scroll DN":? ES
C$;CHR$(158);" Scroll Left":? ESC$;CHR
$(159);" Scroll RT"
200 ? "Undo":? "Kolor change":? "Quit"
:POKE 82,C2:POSITION C2,11:? "Antic 4/
5   LOGO Atari":RETURN
210 AM=C0:POKE 694,C0:POKE 702,C64:IF
PEEK(CONSOL)=C6 THEN 990
220 IF PEEK(CONSOL)=C3 THEN 1400
230 POSITION XC+30,YC+C3:? "█":POSITIO
N XC+30,YC+C3:? " "
235 IF STICK(C0)=C15 AND STRIG(C0) AND
 PEEK(CH)=C255 THEN 210
240 IF  NOT STRIG(C0) THEN 980
250 S=STICK(C0):IF PEEK(CH)<C255 THEN
330
260 XC=XC+(S>C4 AND S<C8)-(S>C8 AND S<
C12)
270 YC=YC+(S=C5)+(S=C9)+(S=C13)-(S=C6)
-(S=C10)-(S=14)
280 IF XC<C0 THEN XC=C7
290 IF XC>C7 THEN XC=C0
300 IF YC<C0 THEN YC=C7
310 IF YC>C7 THEN YC=C0
320 POKE CONSOL,C0:POSITION XC+30,YC+C
3:? "█";:FOR X=C1 TO C5*C5:NEXT X:GOTO
 210
330 IF PEEK(CH)=39 THEN POKE CH,66
340 CLOSE #C1:OPEN #C1,C4,C0,"K:":GET
#C1,X:X=X*(X>41 AND X<86):IF  NOT X TH
EN 210
350 FOR I=C1 TO 18:IF C$(I,I)<>CHR$(X)
 THEN NEXT I:GOTO 210
360 ON I GOTO 420,440,450,500,510,520,
550,580,610,630,650,670,700,730,740,76
0,780,1150
370 TRAP 390:X1=26:FOR X=160 TO 40 STE
P -C4:POKE 53253,X:POSITION X1+C1,C3:?
 A$:X1=X1-C1:IF X1<C1 THEN 390
380 NEXT X
390 FOR X1=X TO 160:POKE 53253,X1:NEXT
 X1:POSITION C2,C3:RETURN
400 IF  NOT STRIG(C0) THEN 400
410 GOTO 210
420 GOSUB 800
430 CHAR=CHBAS+Z*C8:FOR X=C0 TO C7:U(X
)=PEEK(CHAR+X):NEXT X:GOTO 950
440 FOR X=PM TO PM+31:POKE X,C255-PEEK
(X):SOUND C0,X,C10,C8:NEXT X:GOTO 970
450 GOSUB 370:POKE 538,C0:FOR I=C0 TO
C7:POSITION 23,I+C3:? PEEK(CHAR+I);:NE
XT I
455 POSITION C2,C0:? "  PRESS JOYSTICK
 ";
460 ? "BUTTON TO CONTINUE";:LOCATE CX+
C4,CY+C15,Z:POSITION CX+C4,CY+C15:? ES
C$;CHR$(Z);
465 FOR I=C1 TO C4:FOR J=C1 TO C4
470 POSITION I+C3,J+C5:? ESC$;CHR$(Z);
:POSITION I+C9,J+C5:? ESC$;CHR$(Z+C128
);:NEXT J:NEXT I
480 IF STRIG(C0) THEN 480
490 POKE 538,155:GOSUB 370:GOSUB 180:G
OTO 210
500 FOR I=C0 TO C7:A(I)=PEEK(PM+I*C4):
NEXT I:FOR I=C0 TO C7:FOR J=C0 TO C3:P
OKE PM+I*C4+J,A(C7-I):NEXT J:NEXT I
505 GOTO 970
510 FOR X=PM TO PM+32:POKE X,C0:SOUND
C0,X,C6,C8:NEXT X:GOTO 970
520 GOSUB 370:? "SAVE":? :? "FILE (dev
:filename.ext)":INPUT FN$:TRAP 520:CLO
SE #C1
525 OPEN #C1,C8,C0,FN$:FOR X=C0 TO 102
3
530 I=PEEK(CHBAS+X):POKE CHBAS+X,C255:
POKE CHBAS+X,I:PUT #C1,PEEK(CHBAS+X):N
EXT X:GOSUB 370
540 ? CHR$(253);"SAVE COMPLETE":FOR X=
C1 TO 100:NEXT X:GOSUB 370:CLOSE #C1:G
OSUB 180:GOTO 210
550 GOSUB 370:? "LOAD":? :? "FILE (dev
:filename.ext)":INPUT FN$:TRAP 570:CLO
SE #C1
555 OPEN #C1,C4,C0,FN$:FOR X=C0 TO 102
3
560 GET #C1,Q:POKE CHBAS+X,C255:POKE C
HBAS+X,Q:NEXT X:GOSUB 370
565 ? CHR$(253);"LOAD COMPLETE":FOR X=
C1 TO 100:NEXT X
570 GOSUB 370:CLOSE #C1:GOSUB 180:GOTO
 430
580 SAX=CX:SAY=CY:GOSUB 370:? "LOCATE
CHARACTER TO COPY":? "AND PRESS JOYSTI
CK BUTTON":GOSUB 800:CX=SAX:CY=SAY
590 FOR X=C0 TO C7:POKE CHAR+X,PEEK(CH
BAS+Z*C8+X):NEXT X:GOSUB 370:GOSUB 180
600 FOR I=C0 TO C7:FOR J=C0 TO C3:POKE
 PM+I*C4+J,PEEK(CHBAS+Z*C8+I):NEXT J:N
EXT I:GOTO 210
610 SAX=CX:SAY=CY:GOSUB 370:? "LOCATE
CHARACTER TO ":? "REPLACE AND PRESS":?
 "JOYSTICK BUTTON":GOSUB 800:CX=SAX
620 CY=SAY:FOR X=C0 TO C7:POKE CHBAS+Z
*C8+X,PEEK(PM+X*C4):NEXT X:GOSUB 370:G
OSUB 180:GOTO 210
630 FOR I=C0 TO C7:A(I)=PEEK(PM+I*C4):
NEXT I:X=A(C0):FOR I=C1 TO C7:A(I-C1)=
A(I):NEXT I:A(C7)=X:FOR I=C0 TO C7
640 FOR J=C3 TO C0 STEP -C1:POKE PM+I*
C4+J,A(I):SOUND C0,100-I*C4-J,10,C8:NE
XT J:NEXT I:GOTO 970
650 FOR I=C0 TO C7:A(I)=PEEK(PM+I*C4):
NEXT I:X=A(C7):FOR I=C7 TO C1 STEP -C1
:A(I)=A(I-C1):NEXT I:A(C0)=X
660 FOR I=C7 TO C0 STEP -C1:FOR J=C0 T
O C3:POKE PM+I*C4+J,A(I):SOUND C0,I*J,
C12,C8:NEXT J:NEXT I:GOTO 970
670 FOR I=C0 TO C7:A(I)=PEEK(PM+I*C4):
A(I)=A(I)*C2
680 IF A(I)>C255 THEN A(I)=A(I)-C255:G
OTO 680
690 NEXT I:FOR I=C0 TO C7:FOR J=C0 TO
C3:POKE PM+I*C4+J,A(I):SOUND C0,I,C8,C
4:NEXT J:NEXT I:GOTO 970
700 FOR I=C0 TO C7:A(I)=PEEK(PM+I*C4):
J=A(I)/C2:IF J=INT(J) THEN A(I)=J:GOTO
 720
710 A(I)=INT(J)+C128
720 NEXT I:FOR I=C0 TO C7:FOR J=C0 TO
C3:POKE PM+I*C4+J,A(I):SOUND C0,C7-I,C
8,C4:NEXT J:NEXT I:GOTO 970
730 FOR X=C0 TO C7:POKE CHAR+X,U(X):NE
XT X:GOTO 950
740 GOSUB 370:TRAP 740:? "COLOR REGIST
ER";:INPUT R:? :? "COLOR";:INPUT C:? :
? "LUMINANCE";:INPUT L
742 R=R+708:IF R<708 OR R>712 OR C<C0
OR C>C15 OR L<C0 OR L>C15 THEN 740
```

```
750 POKE R,C*C16+L:POKE 705,PEEK(712):
GOSUB 370:GOSUB 180:IF  NOT AM THEN 22
0
755 GOTO 1060
760 GOSUB 370:? "PRESS 'Y' TO EXIT PRO
GRAM":CLOSE #C1:OPEN #C1,C4,C0,"K:":GE
T #C1,X:IF X=89 THEN 790
770 GOSUB 370:GOSUB 180:GOTO 210
780 FOR I=C0 TO 31:POKE PM+I,PEEK(5734
4+CHAR-CHBAS+INT(I/C4)):NEXT I:GOTO 97
0
790 GRAPHICS C0:POKE C559,34:POKE 5327
7,C0:POKE C756,CHBAS/C256:POKE 538,C0:
END
800 LOCATE CX+C4,CY+C15,Z:ORIG=Z
810 XC=C0:YC=C0:LOCATE CX+C4,CY+C15,Z:
POSITION CX+C4,CY+C15:? ESC$;CHR$(Z+C1
28-C128*(Z=C27));
820 FOR X=C1 TO C10:IF STICK(C0)=C15 A
ND STRIG(C0) THEN NEXT X:GOTO 810
830 IF  NOT STRIG(C0) THEN 910
840 POSITION CX+C4,CY+C15:? ESC$;CHR$(
ORIG);
850 S=STICK(C0):CX=CX+(S>C4 AND S<C8)-
(S>C8 AND S<C12):CY=CY+(S=C5)+(S=C9)+(
S=C13)-(S=C6)-(S=C10)-(S=14)
860 IF CX<C0 THEN CX=31
870 IF CX>31 THEN CX=C0
880 IF CY<C0 THEN CY=C3
890 IF CY>C3 THEN CY=C0
900 POKE CONSOL,C0:FOR X=C1 TO C10:NEX
T X:GOTO 800
910 IF Z>C127 THEN Z=Z-C128:GOTO 910
920 IF (Z>C64 AND Z<96) OR (Z>31 AND Z
<65) THEN Z=Z-32:GOTO 940
930 IF Z>-C1 AND Z<32 THEN Z=Z+C64
940 POSITION CX+C4,CY+C15:? ESC$;CHR$(
ORIG);:RETURN
950 SOUND C0,C0,C0,C0:FOR I=PM TO PM+3
1 STEP C4:FOR J=C0 TO C3:POKE I+J,PEEK
(CHAR+(I-PM)/C4):NEXT J:NEXT I
960 GOTO 210
970 SOUND C0,C0,C0,C0:FOR X=PM TO PM+2
8 STEP C4:POKE CHAR+(X-PM)/C4,PEEK(X):
NEXT X:GOTO 210
980 X=USR(ADR(EOR$),PEEK(PM+YC*C4),B(X
C)):FOR I=C0 TO C3:POKE PM+YC*C4+I,X:P
OKE CHAR+YC,X:NEXT I:GOTO 400
990 AM=C1:POKE 752,C1:POKE 538,C0:POSI
TION C3,C0:? "PRESS SELECT TO RETURN T
O EDITING ";
1000 GOSUB 370:? "1 - ANTIC MODES 4 &
5":? "2 - GRAPHICS MODES 1 & 2":? "3 -
 TEXT MODE 0":? :? "WHICH";:TRAP 990
1010 POKE 752,C0:INPUT M:POKE 752,C1:I
F M<C1 OR M>C3 THEN 990
1020 LIM=38:IF M=C2 THEN LIM=18
1030 IF M=C1 THEN POKE DL+C1,C4:POKE D
L+C2,C5
1040 IF M=C2 THEN POKE DL+C1,C6:POKE D
L+C2,C7
1050 ? :? "OPTION Clear display":IF M=
C1 THEN ? "START Change colors"
1060 POSITION C2,21:FOR X=C2 TO LIM:CL
OSE #C1:OPEN #C1,C4,C0,"K:":POKE CH,C2
55
1070 IF PEEK(CONSOL)=C5 THEN POKE 538,
155:GOTO 1130
1080 IF PEEK(CONSOL)=C6 AND M=C1 THEN
740
1090 IF PEEK(CONSOL)=C3 THEN GOSUB 114
0:GOTO 1060
1100 IF PEEK(CH)=C255 THEN 1070
1110 IF M<>C2 OR PEEK(CH)<>C12 THEN 11
20
1115 I=PEEK(C106)-C16:J=PEEK(C756):POK
E C756,J+C2*(I=J)-C2*(I<>J):POKE CH,C2
55:GOTO 1070
1120 GET #C1,I:POSITION X,21:? ESC$;CH
R$(I);:POSITION X+20*(M=C2),22-(M=C2):
? ESC$;CHR$(I);:NEXT X
1125 GOTO 1060
1130 FOR I=C1 TO C3:POKE DL+I,C2:POSIT
ION C0,21:? CHR$(156):NEXT I:GOSUB 370
:GOSUB 180:GOTO 210
1140 FOR I=C1 TO C3:POSITION C0,21:? C
HR$(156):NEXT I:RETURN
1150 X=204:Y=51:IF PEEK(PM+C256)=204 T
HEN X=85:Y=170
1160 FOR I=PM+C256 TO PM+285 STEP C8:F
OR J=C0 TO C3:POKE I+J,X:POKE I+J+C128
,Y:POKE I+J+C4,Y:POKE I+J+132,X
1170 NEXT J:NEXT I:FOR I=DL-C6 TO DL-C
1:POKE I,C2*(X=85)+C4*(X=204):NEXT I:G
OTO 210
1180 DATA 104,104,133,204,104,133,205,
104,69,204,133,213,104,69,205,133,212,
96
1190 DATA 104,104,133,213,104,133,212,
104,133,215,104,133,214,162,4,160,0,17
7,212,145,214,200,208,249,230,213
1200 DATA 230,215,202,208,240,96
1210 REM ALMOST DONE!!
1220 POKE C559,C0:DL=PEEK(C560)+PEEK(C
561)*C256:POKE 710,C0:POKE DL+C7,C7
1225 POSITION C4,C2:? "create";CHR$(14
1);:POKE 752,C1
1230 ? "a";CHR$(141);"font":POKE DL+C8
,C10
1235 FOR X=C2 TO 38:POSITION X,C0:? CH
R$(141);:POSITION X,C3:? CHR$(141);:NE
XT X
1240 POSITION C3,C7:? "'THE ULTIMATE C
HARACTER SET EDITOR'":? :? :? "    Cre
ated 8/2/83 by Vince Erceg"
1250 POSITION 19,22:? "PRESS START TO
BEGIN";:POKE DL+C13,34:POKE DL+21,50:P
OKE 54277,C7
1260 RESTORE 1230:FOR I=1536 TO 1554:R
EAD J:POKE I,J:NEXT I:POKE C559,34
1265 A=USR(1536):POSITION 19,C15:? "PL
EASE WAIT"
1270 FOR X=C7 TO C0 STEP -C1:POKE 5427
7,X:POKE 54276,X:FOR Y=C1 TO C7:SOUND
C0,X*C2,C8,C8:NEXT Y:NEXT X
1275 SOUND C0,C0,C0,C0
1280 POKE 552,111:POKE 553,C6:POKE C0,
C0:FOR I=1570 TO 1695:READ J:POKE I,J:
NEXT I:FOR I=C1 TO 42:READ J
1290 CLEAR$(I)=CHR$(J):NEXT I:RESTORE
:RETURN
1300 DATA 104,173,31,208,201,6,240,10,
232,142,10,212,142,25,208,76,1,6,96
1310 DATA 0,0,0,48,50,37,51,51,0,179,1
80,161,178,180,0,52,47,0,35,40,33,46,3
9,37,0,36,41,51,48,44
1320 DATA 33,57,0,44,41,51,52,0,0,0,0,
0,0,0,48,50,37,51,51,0,175,176,180,169
,175,174,0,38,47,50
1330 DATA 0,36,41,51,43,0,36,41,50,37,
35,52,47,50,57,0,0,165,88,133,203,165,
89,133,204,160,0,166,0,240
1340 DATA 13,185,34,6,145,203,200,192,
38,208,246,76,148,6,185,73,6,145,203,2
00,192,38,208,246,169,155,141,26,2
1350 DATA 165,0,73,1,133,0,96,104,104,
133,204,104,133,203,104,133,206,104
1370 DATA 133,205,166,206,160,0,169,0,
145,203,136,208,251,230,204,202,48,6,2
08,244,164
1390 DATA 205,208,240,198,204,160,0,14
5,203,96
1400 GOSUB 370:TRAP 1460:CLOSE #C1:OPE
N #C1,C6,C0,"D:*.*"
1410 FOR X=C3 TO 14:INPUT #C1;FN$:FN$=
FN$(C3):IF FN$(C3,C12)="FREE SECTO" TH
EN 1460
1420 IF X/C2=INT(X/C2) THEN POSITION C
15,(X+C1)/C2:? FN$;:GOTO 1440
1430 POSITION C2,(X+C2)/C2:? FN$;
1440 NEXT X
1450 ? :? :? "PRESS ANY KEY TO CONTINU
E":GOSUB 1470:GOTO 1410
1460 ? :? :? "END..PRESS ANY KEY":GOSU
B 1470:GOSUB 180:GOTO 210
1470 POKE CH,C255
1480 IF PEEK(CH)<C255 THEN POKE CH,C25
5:GOTO 370
1490 GOTO 1480
1500 C0=0:C1=1:C2=C1+C1:C3=C2+C1:C4=C3
+C1:C5=C4+C1:C6=C5+C1:C7=C6+C1
1510 C8=C7+C1:C9=C8+C1:C10=C9+C1:C12=C
10+C2:C13=C12+C1:C15=C13+C2
1520 C16=C15+C1:C27=C12+C15:C64=C8*C8:
C106=106:C127=127:C128=C127+C1
```

```
1530 C255=255:C256=C255+C1:C559=559:C5
60=C559+C1:C561=C560+C1:C756=756:CH=C7
56+C8:CONSOL=53279
1540 RETURN
```

•

CHECKSUM DATA
(See pp. 20-24.)

```
10 DATA 609,253,228,257,336,739,580,95
4,221,386,605,540,605,798,457,7568
130 DATA 623,45,108,254,372,233,666,14
7,679,821,735,56,627,652,987,7005
250 DATA 287,206,206,117,110,100,93,57
9,623,17,221,996,906,786,624,5871
400 DATA 945,697,986,874,690,10,492,96
5,222,791,381,758,924,733,492,9960
520 DATA 428,836,133,668,409,841,358,7
36,487,981,178,524,756,220,538,8093
640 DATA 48,595,261,21,386,985,805,500
,312,737,585,823,555,972,573,8158
770 DATA 26,787,922,173,495,703,975,60
4,313,52,41,100,101,565,653,6510
920 DATA 929,541,987,668,722,849,537,3
18,28,722,155,597,605,237,644,8539
1070 DATA 305,71,348,941,936,669,269,7
23,661,823,596,259,232,65,224,7122
1200 DATA 444,172,902,303,697,948,783,
142,7,179,61,824,637,200,673,6972
1310 DATA 74,593,373,81,679,759,100,66
3,788,283,23,545,900,911,129,6901
1480 DATA 995,737,819,343,647,156,798,
4495
```

•

Ask Sally Forth

by Sally Forth

Here's FPLOT, the ultrahigh-speed graphics plotting routine I promised to bring you last month. It's designed exclusively for Atari's high-resolution 2-color mode "F" (GRAPHICS mode 24 in BASIC). To use it, you must have **valFORTH**'s standard graphics library and assembler LOADed into your system. The definition eats up a hefty chunk of dictionary space, but who cares? FPLOT is *fast*.

Although the source screens are mostly self-documenting, a few clarifications are in order for the sake of beginners. FPLOT uses a technique known as "table lookup" to eliminate the time-consuming calculations used by the PLOT routine in Atari's Central Input/Output (CIO) utility. The twin ALLOT words in screen 1 set aside 384 bytes for the lookup tables; screens 2 and 3 set up negative and positive bit-masks for the plot logic. Screens 4 through 7 define an assembly-language word (PLOTSETUP) that initializes the lookup tables. It must be executed immediately after setting up the high-res screen (with a 24 GR. instruction) or FPLOT will not work. I could have written PLOTSETUP in FORTH, but speed is the reason for this entire exercise, right?

Screens 8 through 12 contain the assembly-language definition of FPLOT. Syntax is identical to the PLOT word in **valFORTH's** standard graphics package. The color to be plotted (0-1) is specified with the **valFORTH** word COLOR, just like the ordinary PLOT. Strange things will happen if your plotting parameters exceed 319 for the X-coordinate, or 191 for the Y. FPLOT's lack of automatic range checking is part of the price you pay for extra speed.

You can use FPLOT with the **valFORTH** word DR. (DRAWTO) because it saves the X and Y coordinates in the operating system locations COLCRS and ROWCRS. If you're not using DR. and you want to make FPLOT even faster, eliminate Line 9 in screen 9 and Lines 6, 7, and 9 in screen 10.

The demo words in screens 13-17 will show you just how much speed you can expect from FPLOT. I obtained screen-fill times of 9957 jiffies for the CIO plot and just 1632 jiffies for PLOT, a 6-times improvement. In case you're wondering, it takes Atari BASIC 24554 jiffies to fill a mode F screen with PLOTs!

Next month, I'll show you how to implement megaspeed line-drawing with FPLOT. □

(FORTH screens next page.)

SCREEN #1

```
( HIGH SPEED MODE F PLOTTER )
( Reserve space for tables )
DECIMAL
LABEL YLOWS 192 ALLOT   ( lsbs )
LABEL YHIGHS 192 ALLOT ( msbs )
( OS equates )
84 CONSTANT ROWCRS
85 CONSTANT COLCRS
88 CONSTANT SAVMSC
                              -->
```

SCREEN #2

```
( HIGH SPEED MODE F PLOTTER )
2 BASE ! ( for convenience )
LABEL NMASKS ( plot masks )
     11111110 C,  11111110 C,
     11111101 C,  11111101 C,
     11111011 C,  11111011 C,
     11110111 C,  11110111 C,
     11101111 C,  11101111 C,
     11011111 C,  11011111 C,
     10111111 C,  10111111 C,
     01111111 C,  01111111 C,
                              -->
```

SCREEN #3

```
( HIGH SPEED MODE F PLOTTER )
LABEL PMASKS ( more masks )
        0 C,         1 C,
        0 C,        10 C,
        0 C,       100 C,
        0 C,      1000 C,
        0 C,     10000 C,
        0 C,    100000 C,
        0 C,   1000000 C,
        0 C,  10000000 C,
DECIMAL
                              -->
```

SCREEN #4

```
( HIGH SPEED MODE F PLOTTER )
ASSEMBLER ( new vocabulary )
        N 4 + CONSTANT PNTR
        N 3 + CONSTANT XHI
        N 2 + CONSTANT XLO
          N CONSTANT YPOS
( PLOTSETUP initializes the
( YLOWS and YHIGHS tables for
( use by the high-speed plot
( routine.  It should be called
( immediately after setting up
( graphics mode 24. )
                              -->
```

SCREEN #5

```
( HIGH-SPEED MODE F PLOTTER )
CODE PLOTSETUP
    XSAVE STX, ( preserve X     )
SAVMSC 1+ LDA, ( get msb of     )
     YPOS STA, ( screen addr    )
   SAVMSC LDY, ( and also lsb   )
     # 0 LDX, ( init index      )
         CLD, ( for safety      )
BEGIN,        ( start a loop    )
     YPOS LDA, ( put msb into   )
YHIGHS ,X STA, ( msb table      )
                              -->
```

SCREEN #6

```
( HIGH-SPEED MODE F PLOTTER )
           TYA, ( put lsb into  )
  YLOWS ,X STA, ( lsb table     )
           CLC, ( for addition  )
      # 40 ADC, ( new offset    )
CS IF,          ( if carry set, )
      YPOS INC, ( increment msb )
ENDIF,
           TAY, ( save new lsb  )
           INX, ( update index  )
     # 192 CPX, ( done 192 yet? )
                              -->
```

SCREEN #7

```
( HIGH-SPEED MODE F PLOTTER )
EQ UNTIL,       ( loop until    )
                ( index = 192   )
     XSAVE LDX, ( restore X-reg )
      NEXT JMP, ( back to FORTH )
             C; ( end PLOTSETUP )
                              -->
```

SCREEN #8

```
( HIGH-SPEED MODE F PLOTTER )
( FPLOT is the actual plotting
( word.  Syntax is the same as
( the standard CIO-type PLOT:
(          x-pos y-pos ---
( X-pos can range from 0-319.
( Y-pos range is 0-191.  No
( range checks are performed
( to save time, so be careful!
( Plot color is controlled by
( the standard valFORTH word
( COLOR.  Legal COLOR values
( are 0 and 1. )              -->
```

SCREEN #9

```
( HIGH SPEED MODE F PLOTTER )
CODE FPLOT
      # 2 LDA, ( # DROP values )
   SETUP JSR, ( move into N     )
   XSAVE STX, ( preserve X      )
    YPOS LDX, ( y-pos is used )
  ROWCRS STX, ( as an index     )
 YLOWS ,X LDA, ( into lsb/msb   )
    PNTR STA, ( offset tables )
YHIGHS ,X LDA, ( to create a    )
 PNTR 1+ STA, ( z-page pntr     )
                              -->
```

SCREEN #10

```
( HIGH-SPEED MODE F PLOTTER )
( PNTR now contains the
( absolute RAM address of the
( desired mode line.)
      XHI LDA, ( get msb x-pos )
 COLCRS 1+ STA, ( for OS usage  )
      XLO LDA, ( get lsb x-pos )
   COLCRS STA, ( also for OS    )
      XHI LSR, ( divide x-pos   )
       .A ROR, ( by 2           )
       .A LSR, ( then 4         )
       .A LSR, ( and then 8     )
         TAY, ( save x/8 in Y )
                              -->
```

SCREEN #11

```
( HIGH-SPEED MODE F PLOTTER )
      XLO LDA, ( get lsb again )
    # 7 AND, ( mask to get
               ( bit position
               ( in plot byte   )
       .A ASL, ( multiply by 2 )
   CLRBYT ORA, ( & superimpose
               ( bit0 color
               ( data for use   )
          TAX, ( as an index
               ( into the mask
               ( tables         )
   PNTR )Y LDA, ( get plot byte )
NMASKS ,X AND, ( mask plot bit )
                              -->
```

SCREEN #12

```
( HIGH-SPEED MODE F PLOTTER )
PMASKS ,X ORA, ( superimpose )
               ( the plot bit )
   PNTR )Y STA, ( and show the )
               ( new byte       )
    XSAVE LDX, ( restore X-reg )
     NEXT JMP, ( and return     )
            C; ( end FPLOT      )
                              -->
```

SCREEN #13

```
( FPLOT DEMONSTRATION )
( PLOTEST and FPLOTEST compare
( the speed of the standard
( CIO-type PLOT with FPLOT.
( They PLOT all 61,440 pixels
( on a full Mode F screen,
( using the system jiffy
( timers to clock the speed.
( Be sure you've LOADed the
( valFORTH graphics library
( into your system before
( LOADing these words! )
                              -->
```

SCREEN #14

```
( FPLOT DEMONSTRATION )
19 CONSTANT TOCK ( the timer )
20 CONSTANT TICK ( locations )
0 VARIABLE TIMING
( Set up screen & colors )
: GREADY
  24 GR.
  1 0 14 SE. ( white pixels )
  2 0 0 SE. ( black bkgnd )
  1 COLOR ;
                              -->
```

SCREEN #15

```
( FPLOT DEMONSTRATION )
( Calculate & display
( timer reading )
: SHOWTIME
  TOCK C@ 256 *
  TICK C@ +
  TIMING !
  0 GR.
  TIMING @ .
  ." jiffies with " ;
                              -->
```

SCREEN #16

```
( FPLOT DEMONSTRATION )
( Fill screen with CIO PLOTs )
: PLOTEST
  GREADY
  0 TOCK ! ( zero timers )
  192 0 DO
        320 0 DO
              I J PLOT
        LOOP
  LOOP
  SHOWTIME
  ." CIO PLOT." CR ;
                              -->
```

SCREEN #17

```
( FPLOT DEMONSTRATION )

( Fill screen with FPLOTs )

: FPLOTEST
  GREADY
  PLOTSETUP ( init plot tables )
  0 TOCK !  ( zero timers      )
  192 0 DO
        320 0 DO
              I J FPLOT
        LOOP
  LOOP
  SHOWTIME
  ." FPLOT." CR ;

```

•

Sally welcomes your questions about the FORTH programming language, and will publish the most interesting letters in future columns. Write to her c/o:

ANALOG Computing
P.O. Box 23
Worcester, MA 01603

XLDEMO

16K Cassette or Disk

by Jerry White

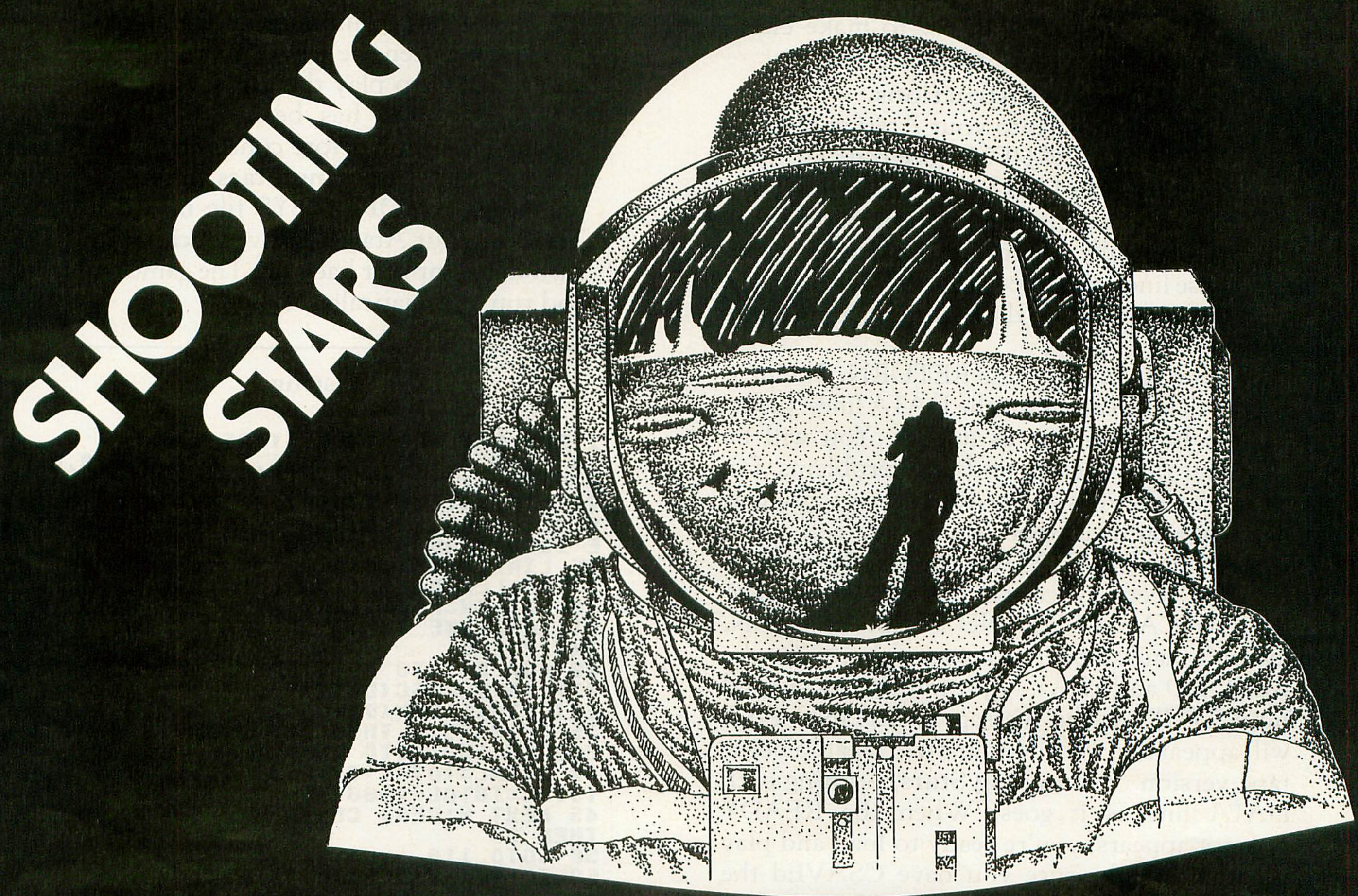
Atari threw a couple of surprising capabilities into the new XL operating system. This self-documenting tutorial program will show you how to access some of these features from Atari BASIC. XLDEMO works on any Atari XL-Series computer, including the old 1200XL. — Editor

```
10 REM  ATARI XL COMPUTER DEMO
20 REM   (C)1983 by Jerry White
30 REM
40 REM The following routines will
50 REM demonstrate some of the unique
60 REM features of ATARI XL computers
70 REM
80 REM  CLEAR SCREEN & SET MARGINS
90 GRAPHICS 0:POKE 82,2:POKE 83,39
100 REM
110 REM  SET SCREEN COLORS
120 SETCOLOR 2,9,2:SETCOLOR 4,1,10:SET
COLOR 1,9,15
130 REM
140 REM  TURN OFF THE CURSOR & BEGIN
150 POKE 752,1:GOSUB 910
160 GOSUB 1180:LIST 10,90
170 REM
180 ? :? "    Programs may be listed wi
th fine"
190 ? :? "scrolling by first poking a 
non-zero"
200 ? :? "into location 622 prior to o
pening"
210 ? :? "the screen editor, then usin
g the"
220 ? :? "ATARI BASIC LIST command."
230 GOSUB 1080
240 REM
250 REM  SETUP FOR FINE SCROLLING
260 POKE 622,1
270 REM NOTE THAT IN THIS PROGRAM WE
280 REM DID NOT HAVE TO OPEN AN IOCB
290 REM FOR THE SCREEN EDITOR SINCE
300 REM THE BASIC GRAPHICS 0 COMMAND
310 REM DID IT FOR US.
320 REM
330 POKE 752,1:GOSUB 1190:LIST 10,430:
GOSUB 1080
340 REM
350 REM  POKE 622,0 TO RETURN TO
360 REM  NORMAL FASTER SCROLLING
370 REM
380 REM  KEY REPEAT RATE
390 REM
400 ? :? "    There is an initial delay
 of .8"
410 ? :? "seconds before a key will re
peat."
420 ? :? "The default value stored in 
location"
430 ? :? "729 is 48.  This value is th
e number"
440 ? :? "of 60ths of a second to dela
y before"
450 ? :? "a key will repeat (48/60=.8)
."
460 ? :? "    To change the delay to 1/
3 of a"
470 ? :? "second, POKE 729,20.  To mak
e this"
480 ? :? "delay one full second, POKE 
729,60."
490 GOSUB 1080
500 ? :? "    The value stored in locat
ion 730"
510 ? :? "is the delay between repeats
.  The"
520 ? :? "default is 6 or 6/60ths of a
 second."
530 ? :? "At this rate, a key will rep
eat ten"
540 ? :? "times per second.  To slow t
he repeat"
550 ? :? "rate down to five times per 
second,"
560 ? :? "POKE 730,12.  To double the 
speed,"
570 ? :? "POKE 730,3.":POKE 729,20:POK
E 730,3
580 GOSUB 1080
590 REM  KEY CLICK SOUND
600 REM
610 ? :? "    The click sound you hear 
whenever"
620 ? :? "you type can also be disable
d on XL"
630 ? :? "computers.  The default valu
e stored"
640 ? :? "in location 731 is 0 and ind
icates"
650 ? :? "that the click is enabled.  
To dis-"
660 ? :? "the click, simply POKE in a 
255."
670 POKE 731,255
680 GOSUB 1080:POKE 82,3
690 ? :? "    This program has just set
 the"
700 ? :? "the delay before repeat to 1
/3 of"
710 ? :? "a second, and doubled the sp
eed of"
720 ? :? "the repeat rate.  Fine scrol
l has"
730 ? :? "been left on, and the keyboa
rd click"
740 ? :? "has been disabled."
750 ? :? "    I now return control of t
his"
760 ? :? "computer to you and ATARI BA
SIC."
770 POSITION 26,18:? "Jerry White"
780 POKE 82,2:POSITION 2,17:POKE 752,0
```

```
790 ? :? "BASIC":? "IS";:END
800 REM
810 REM  HELP KEY SUBROUTINE
820 REM    by Jerry White
830 POKE 732,0:REM CLEAR HELP KEY
840 HELP=PEEK(732):POSITION 7,23
850 IF PEEK(53279)=6 THEN START=1:RETU
RN
860 IF NOT HELP THEN 840
870 IF HELP=17 THEN ? " YOU PRESSED TH
E HELP KEY ";
880 IF HELP=81 THEN ? " YOU PRESSED S
HIFT/HELP ";
890 IF HELP=145 THEN ? "  YOU PRESSED
CTRL/HELP ";
900 RETURN
910 ? :? "    This program demonstrates
 how to"
920 ? "check for the HELP key from BAS
IC on"
930 ? "an ATARI XL computer."
940 LIST 1000,1090
950 ? :? " Press HELP, SHIFT/HELP, CTR
L/HELP,"
960 ? "      or Press  START  to exit."
:START=0
970 GOSUB 1230
980 POKE 53279,8:GOSUB 830
990 IF START THEN RETURN
1000 POSITION 16,21:? "         ";:POKE
540,120
1010 IF PEEK(540) THEN 1010
1020 POSITION 7,23:? "
     ";
1030 POSITION 16,21:? " START ";:GOTO
980
1040 REM
1050 REM  WAIT FOR START KEY
1060 REM  PRESS AND RELEASE.
1070 REM
1080 GOSUB 1230
1090 POSITION 8,23:? "PRESS  START  TO
 CONTINUE";
1100 POKE 540,60
1110 IF PEEK(53279)=6 THEN 1180
1120 IF PEEK(540) THEN 1110
1130 POSITION 14,23:? " START ";
1140 POKE 540,60
1150 IF PEEK(53279)=6 THEN 1180
1160 IF PEEK(540) THEN 1150
1170 GOTO 1090
1180 ? CHR$(125):REM CLEAR SCREEN
1190 SETCOLOR 2,9,2:SETCOLOR 4,1,10:SE
TCOLOR 1,9,15:RETURN
1200 REM
1210 REM  SOFT BELL SOUND
1220 REM
1230 FOR JW=15 TO 0 STEP -0.5:SOUND 0,
0,2,JW:NEXT JW:RETURN
```

CHECKSUM DATA
(See pp. 20-24.)

```
10 DATA 247,289,255,703,327,690,263,84
3,613,74,506,346,83,369,170,5778
160 DATA 338,95,615,54,904,410,251,814
,88,487,780,124,909,882,963,7714
310 DATA 224,84,962,90,560,937,99,786,
105,204,643,974,237,721,994,7620
460 DATA 893,526,34,836,508,531,365,88
4,3,978,567,335,835,812,84,8191
610 DATA 512,748,54,124,669,716,267,95
,344,886,734,271,735,508,992,7655
760 DATA 976,235,826,393,88,366,384,42
1,218,371,396,206,82,21,597,5580
910 DATA 121,830,406,325,293,591,832,2
70,813,976,189,575,718,279,141,7359
1060 DATA 952,282,941,560,135,880,195,
969,139,884,211,724,266,540,281,7959
1210 DATA 892,283,818,1993
```



16K Cassette or Disk

by Dennis Fox

Shooting Stars is an assembly-language game that strands you on an asteroid with almost zero gravity. Red-hot particles of space debris are bouncing all around you. Death is virtually certain in this hostile environment. The question is, how long can you survive?

Your objective is to avoid contact with the "shooting stars" for as long as possible. You are represented by a little astronaut figure, which can be moved around the screen with a joystick plugged into port #1.

You begin the game with five lives. The number of lives remaining is displayed in the top part of the screen, along with a clock which shows how long you have survived. An extra life is awarded at the 3-minute mark, and another at the 5-minute mark. A bell will signal the earning of an extra life. The longer you remain alive, the faster the flying particles will move.

I hope you enjoy playing **Shooting Stars**. See if you can beat my high score of five minutes and fourteen seconds.

Typing the program.

Before typing anything into your computer, take a look at the program listings accompanying this article.

Listing 1 is the main data and data checking routine. This listing is used to create both the cassette and disk versions of **Shooting Stars**. The DATA statements are listed in hexadecimal (base 16) so that the program will fit in a 16K cassette-based system. This makes typing a little more difficult, but it's a necessary evil.

Listing 2 must be added to **Listing 1** if you are using a 410 or 1010 cassette drive.

Listing 3 must be added to **Listing 1** if you are using a disk drive.

Listing 4 is the assembly-language source code for **Shooting Stars**, created with the Atari Macro Assembler. You do *not* have to type in this listing to play the game! It's provided for those readers who want to study how the program works.

Follow the instructions below to make either a cassette or disk version of **Shooting Stars**.

Cassette instructions.

1. Type **Listing 1** into your Atari and verify your typing with C:CHECK (see page 20).

2. After **Listing 1** has been entered into your computer, type in **Listing 2**. The program lines will automatically merge with **Listing 1**. Make sure these lines were typed exactly as shown. It's a good idea to CSAVE the entire program at this point.

3. Type RUN and press RETURN. The program will begin checking the DATA lines, printing the line numbers as it goes. You will be alerted if the program finds any problems. Fix any incorrect lines and re-RUN the program as necessary until all errors are eliminated.

4. When all the DATA is correct, the computer will "beep" twice and you will be asked to READY CASSETTE AND PRESS RETURN. Insert a blank tape in your recorder, press the RECORD and PLAY keys simultaneously and hit RETURN. The message WRITING FILE will appear and the computer will create a boot-tape version of **Shooting Stars**, printing each DATA line as it goes. When the READY prompt appears, you're ready to load and play the game. Make sure you have CSAVEd the BASIC program on a separate tape before continuing.

5. From this point on, when you want to play **Shooting Stars**, do the following: Rewind the boot tape created by the BASIC program to the beginning. Turn your Atari off and remove all cartridges. Press the PLAY key on your recorder and turn your Atari back on while holding down the START key. The computer will "beep" once. Press RETURN and **Shooting Stars** will load and run automatically.

Disk instructions.

1. Type **Listing 1** into your Atari and verify your typing with D:CHECK2 (see page 16.)

2. After **Listing 1** is correctly entered, carefully type in **Listing 3**. The program lines will merge with those in **Listing 1**. It's a good idea to SAVE the entire BASIC program at this point.

3. Type RUN and press RETURN. The program will begin verifying each DATA line, printing the line numbers as it goes. You will be alerted if the program finds any problems. Fix incorrect lines and re-RUN the program as necessary until all errors are eliminated.

4. When all DATA lines are correct, you will be prompted to INSERT DISK WITH DOS, PRESS RETURN. Insert a disk containing Atari DOS 2.0S into drive #1 and press RETURN. The message WRITING FILE will appear, and the computer will create an AUTORUN.SYS file on the disk, printing each line number as it proceeds. When the READY prompt appears, you're ready to play the game. Make sure the BASIC program has been SAVEd under a separate filename before continuing.

5. To play **Shooting Stars**, insert the disk with the AUTORUN.SYS file into drive #1. Turn off your Atari, remove all cartridges and turn the computer back on. The game will load and run automatically. □

Listing 1.

```
1 REM *** SHOOTING STARS ***
10 DATA 0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,
0,0,10,11,12,13,14,15
20 DIM DAT$(91),HEX(22):FOR X=0 TO 22:
READ N:HEX(X)=N:NEXT X:LINE=990:RESTOR
E 1000:TRAP 60:? "CHECKING DATA"
25 LINE=LINE+10:? "LINE:";LINE:READ DA
T$:IF LEN(DAT$)<>90 THEN 110
28 DATLIN=PEEK(183)+PEEK(184)*256:IF D
ATLIN<>LINE THEN ? "LINE ";LINE;" MISS
ING!":END
30 FOR X=1 TO 89 STEP 2:D1=ASC(DAT$(X,
X))-48:D2=ASC(DAT$(X+1,X+1))-48:BYTE=H
EX(D1)*16+HEX(D2)
35 IF PASS=2 THEN PUT #1,BYTE:NEXT X:R
EAD CHKSUM:GOTO 25
40 TOTAL=TOTAL+BYTE:IF TOTAL>999 THEN
TOTAL=TOTAL-1000
45 NEXT X:READ CHKSUM:IF TOTAL=CHKSUM
THEN 25
50 GOTO 110
60 IF PEEK(195)<>6 THEN 110
100 ? "WRITING FILE":PASS=2:LINE=990:R
ESTORE 1000:TRAP 60:GOTO 25
110 ? "BAD DATA: LINE ";LINE:END
1000 DATA A209A9FF9D30259DDE26CA10F7A2
29A92585D6A93A85D5A009B9122591D58810F8
A5D518690A85D5A5D6690085,695
1010 DATA D6CA10E6A9CA8D3002A9248D3102
A90085CD85CB85CF85D185D3A93385CEA93485
CCA93585D0A93685D2A93785,788
1020 DATA D4A9108D42068D43068DA006A900
8DA1068DA2068DA3068D6006A9058D5006A9B4
8D0006AD0AD24A1869378D01,948
1030 DATA 06A200AD0AD24A4A69329D1006E8
E006D0F1A2008A0A0A0A0A69379D2006E8E0
06D0F0A200AD0AD229039D30,321
1040 DATA 06E8E006D0F3A9308D07D4A93E8D
2F02A9038D1DD0A900A891CB91CD91CF91D191
D3C8D0F3A9008D45068D1ED0,300
1050 DATA A9468DC002A90F8DC1028DC2028D
C302A9968DC502A9368DC602A9F88DC402A903
8D4106A907A224A080205CE4,446
1060 DATA AD1FD0C906D0034C6824AD780229
01D003CE0106AD7802290 8D003EE0006AD7802
2902D003EE0106AD78022904,186
1070 DATA D003CE0006AD0006C9C8D003CE00
06AD0006C932D003EE0006AD0106C934D003EE
0106AD0106C9C8D003CE0106,391
1080 DATA AD00068D00D0A200AC0106BDFF24
91CBE8C8E013D0F5A200BD3006D006FE1006DE
2006BD3006C901D006FE1006,306
1090 DATA FE2006BD3006C902D006DE1006FE
2006BD3006C903D006DE1006DE2006E8E006D0
C9A200BD1006C9CAD012BD30,69
1100 DATA 06D008A9039D30061890 05A9029D
3006BD1006C930D014BD3006C902F008A9009D
30061890 05A9019D3006BD20,766
1110 DATA 06C934D012BD3006D008A9019D30
061890 05A9029D3006BD2006C9D5D014BD3006
C901D008A9009D3006189005,729
1120 DATA A9039D3006E8E006D093AD10068D
01D0AD11068D02D0AD12068D03D0AD13068D05
D0AD14068D06D0AD15068D07,24
1130 DATA D0A900AC200691CFA910C891CFC8
91CFA900C891CFA900AC210691D1A910C891D1
C891D1A900C891D1A900AC22,98
```

```
1140 DATA 0691D3A910C891D3C891D3A900C8
91D3A9EFAC230631CD91CDA910C811CD91CDA9
10C811CD91CDA9EFC831CD91,628
1150 DATA CDA9FBAC240631CD91CDA904C811
CD91CDA904C811CD91CDA9FBC831CD91CDA9BF
AC250631CD91CDA940C811CD,19
1160 DATA 91CDA940C811CD91CDA9BFC831CD
91CDEE4006AD4006C910D03AA9008D4006CE41
06AD4106D005A9038D4106A9,239
1170 DATA C88D03D2AD4106C903D008A9788D
02D2189014AD4106C902D008A9648D02D21890
05A9558D02D2AD42068D2525,812
1180 DATA AD43068D2625ADA0068D2325ADA0
06C913D015ADA106D010EE5006EEA106A9A48D
01D2A9018DA206ADA006C915,632
1190 DATA D017ADA106C901D010EE5006EEA1
06A9A48D01D2A9018DA206AD4506F0034C1724
A9504E42060838ED4206282E,6
1200 DATA 4206A200EDA006E8E003D0F8AAA0
00C8D0FDCAD0F8ADA206C901D017A9328D00D2
EEA306ADA306C925D008EEA2,154
1210 DATA 06A9008DA306ADA206C902D017A9
288D00D2EEA306ADA306C925D008EEA206A900
8DA306ADA206C903D017A920,39
1220 DATA 8D00D2EEA306ADA306C925D008EE
A206A9008DA306ADA206C904D00DA9008D00D2
8D01D2A9008DA306A98AA200,65
1230 DATA 9D2925E8EC5006D0F74C0E21CE50
06A9A88D01D28D03D2A9328D00D2481869648D
02D28DC00268A000A200E8D0,192
1240 DATA FDC8C019D0F6186901C964D0DFA9
008D00D28D01D28D02D28D03D2A9468DC002A9
00AC5006992925AD5006F003,491
1250 DATA 4C7920A9008D5006AD8402F007AD
1FD0C906D0F4A900854D4C5B20AD09D00D0AD0
0D0BD00D0DD00D0ED00D0FD0,877
1260 DATA 29018D4506EE6006AD6006C93CD0
26A9008D6006EE4306AD4306C91AD017A9108D
4306EE4206AD4206C916D008,77
1270 DATA EEA006A9108D42064C62E4707070
471C2509090909090909090909090909090909
090909090909090909090909,5
1280 DATA 0909090909090909090909090909
090941CA24007C546C7C447C1010FEBABABA38
382828EE00800000000000000000,486
1290 DATA 0000010000746960650000001A0000
00000000000000000000000000000000000000
000000000000000000000000000,944
```

CHECKSUM DATA
(see pp. 20-24)

```
1 DATA 881,955,686,427,745,192,617,545
,276,445,496,549,150,973,995,8932
1020 DATA 768,669,955,856,671,709,750,
27,551,477,210,152,143,547,83,7568
1170 DATA 840,823,674,934,969,941,836,
724,876,758,202,441,487,9505
```

Listing 2.

```
2 REM *** CASSETTE VERSION ***
65 IF PASS=2 THEN FOR X=1 TO 18:PUT #1
,0:NEXT X:CLOSE #1:END
70 ? "READY CASSETTE AND PRESS RETURN"
;:OPEN #1,8,128,"C:":RESTORE 200:FOR X
=1 TO 40:READ N:PUT #1,N:NEXT X
200 DATA 0,11,216,31,255,31,169,0,141,
47,2,169,60,141,2,211,169,0,141,231,2,
133,14,169,56,141,232,2
210 DATA 133,15,169,0,133,10,169,32,13
3,11,24,96
```

Listing 3.

```
2 REM *** DISK VERSION ***
65 IF PASS=2 THEN PUT #1,224:PUT #1,2:
PUT #1,225:PUT #1,2:PUT #1,0:PUT #1,32
:CLOSE #1:END
70 ? "INSERT DISK WITH DOS, PRESS RETU
RN";:DIM IN$(1):INPUT IN$:OPEN #1,8,0,
"D:AUTORUN.SYS"
90 PUT #1,255:PUT #1,255:PUT #1,0:PUT
#1,32:PUT #1,69:PUT #1,37
```

Listing 4.

```
;
; Shooting stars by Dennis Fox
;

; Page zero usage

PBASE  =  $CB        ;POINTERS TO...
MBASE  =  $CD        ;MISSILES,
BASE2  =  $CF        ;PLAYER1,
BASE3  =  $D1        ;PLAYER2,
BASE4  =  $D3        ;PLAYER3
ADDR   =  $D5        ;2-BYTE POINTER

; Page 6 usage

X      =  $600
Y      =  $601
XM     =  $610
YM     =  $620
DM     =  $630
T1     =  $640
T2     =  $641
TI1    =  $642
TI2    =  $643
HIT    =  $645
LIFE   =  $650
TI3    =  $660
TI4    =  $6A0
ELF    =  $6A1
SF1    =  $6A2
STIME  =  $6A3

; Operating system equates

ATTRAC =  $4D        ;ATTRACT MODE POINTER
PCOLR0 =  $2C0       ;PLAYER COLORS
PCOLR1 =  $2C1
PCOLR2 =  $2C2
PCOLR3 =  $2C3
COLOR0 =  $2C4       ;PLAYFIELD COLORS
COLOR1 =  $2C5
COLOR2 =  $2C6
COLOR3 =  $2C7
COLOR4 =  $2C8
SDMCTL =  $22F       ;DMA CONTROL
SDLSTL =  $230       ;DISP LIST POINTER
SDLSTH =  $231
STICK0 =  $278       ;JOYSTICK
STRIG0 =  $284       ;STICK BUTTON
HITCLR =  $D01E      ;COLLISION CLEAR
HPOSP0 =  $D000      ;P/M HOR. POSITIONS
HPOSP1 =  $D001
HPOSP2 =  $D002
HPOSP3 =  $D003
HPOSM0 =  $D004
HPOSM1 =  $D005
HPOSM2 =  $D006
HPOSM3 =  $D007
```

```
M1PL   =   $D009         ;COLLISION REGISTERS
M2PL   =   $D00A
M3PL   =   $D00B
P1PL   =   $D00D
P2PL   =   $D00E
P3PL   =   $D00F
GRACTL =   $D01D         ;GRAPHIC CONTROL
CONSOL =   $D01F         ;CONSOLE BUTTONS
AUDF1  =   $D200         ;AUDIO CONTROLS
AUDC1  =   $D201
AUDF2  =   $D202
AUDC2  =   $D203
RANDOM =   $D20A         ;RANDOM #
PMBASE =   $D407         ;P/M START ADDR
SETVBV =   $E45C         ;VERT BLANK SET VECTOR
XITVBV =   $E462         ;VERT BLANK EXIT

; set up screen and display list

 ORG $2000

SSTAR  LDX #9
       LDA #$FF
SETTB  STA SCRNA,X       ;SET TOP...
       STA SCRNC,X       ;AND BOTTOM...
       DEX               ;OF SCREEN
       BPL SETTB
       LDX #41
       LDA #SCRNB/256    ;POINT TO...
       STA ADDR+1        ;MIDDLE OF SCREEN
       LDA #SCRNB&255
       STA ADDR
SETMID LDY #9
MIDLP  LDA MIDBYT,Y      ;COPY MIDDLE BYTES...
       STA (ADDR),Y      ;TO SCREEN RAM
       DEY
       BPL MIDLP
       LDA ADDR          ;NOW ADD 10 BYTES...
       CLC               ;TO POINT TO...
       ADC #10           ;NEXT LINE
       STA ADDR
       LDA ADDR+1
       ADC #0
       STA ADDR+1
       DEX               ;ANOTHER LINE?
       BPL SETMID        ;YES!
       LDA #DLIST&255    ;POINT TO
       STA SDLSTL        ;DISPLAY
       LDA #DLIST/256    ;LIST
       STA SDLSTH
       LDA #$00          ;SET LOW BYTES...
       STA MBASE         ;OF P/M POINTERS
       STA PBASE
       STA BASE2
       STA BASE3
       STA BASE4
       LDA #$33          ;NOW SET HI BYTES
       STA MBASE+1       ;TO POINT TO
       LDA #$34          ;VARIOUS P/M AREAS
       STA PBASE+1
       LDA #$35
       STA BASE2+1
       LDA #$36
       STA BASE3+1
       LDA #$37
       STA BASE4+1

; Set players starting position

START2 LDA #16           ;INITIALIZE TIMERS
       STA TI1
       STA TI2
       STA TI4
       LDA #$00
       STA ELF
       STA SF1
       STA STIME
       STA TI3
       LDA #5            ;START W/5 LIVES
       STA LIFE
START  LDA #180          ;SET UP...
       STA X             ;ASTRONAUT X
       LDA RANDOM        ;NOW GET...
       LSR A             ;RANDOM #
       CLC               ;AND SET UP
       ADC #55           ;ASTRONAUT Y
       STA Y

; Set missles horizontal position

       LDX #$00          ;GET RANDOM...
L1     LDA RANDOM        ;HORIZONTAL
       LSR A             ;POSITION,
       LSR A             ;MAKE SURE IT'S
       ADC #50           ;ON SCREEN,
       STA XM,X          ;STORE IT
       INX               ;NEXT MISSILE
       CPX #6            ;ALL DONE?
       BNE L1            ;NOT YET!

; Set missles vertical position

       LDX #$00          ;START W/MISSILE 0
L2     TXA               ;GET MISSILE #
       ASL A             ;MULTIPLY...
       ASL A             ;MISSILE...
       ASL A             ;NUMBER...
       ASL A             ;BY 32,
       ASL A             ;MAKE SURE IT'S...
       ADC #55           ;ON SCREEN,
       STA YM,X          ;AND STORE IT!
       INX               ;NEXT MISSILE
       CPX #6            ;ALL DONE?
       BNE L2            ;NOT YET!

; Set missiles' starting directions

       LDX #$00
L3     LDA RANDOM        ;GET RANDOM...
       AND #$03          ;DIRECTION
       STA DM,X          ;AND STORE
       INX               ;NEXT MISSILE
       CPX #6            ;ALL DONE?
       BNE L3            ;NOT YET!

; Set up player missile graphics
; Single line resolution

       LDA #$30          ;POINT TO P/M AREA
       STA PMBASE
       LDA #62           ;TURN ON SCREEN
       STA SDMCTL
       LDA #3            ;SET P/M GRAPHICS
       STA GRACTL
       LDA #$00          ;NOW CLEAR OUT...
       TAY               ;ALL MISSILES
A1     STA (PBASE),Y     ;AND PLAYERS
       STA (MBASE),Y
       STA (BASE2),Y
       STA (BASE3),Y
       STA (BASE4),Y
       INY
       BNE A1
       LDA #$00          ;NO COLLISION!
       STA HIT
       STA HITCLR
       LDA #$46          ;SET UP COLORS
       STA PCOLR0
       LDA #15
       STA PCOLR1
       STA PCOLR2
       STA PCOLR3
       LDA #150
       STA COLOR1
       LDA #54
       STA COLOR2
       LDA #$F8
       STA COLOR0
       LDA #3
       STA T2
       LDA #$07          ;NOW START UP...
       LDX #VBI/256      ;OUR VERTICAL BLANK...
       LDY #VBI&255      ;INTERRUPT!
       JSR SETVBV

; The game begins here

BACK   LDA CONSOL        ;CHECK CONSOLE KEYS
       CMP #6            ;START PRESSED?
       BNE JOY2          ;NOPE!
       JMP DEATH         ;GO TO RESTART!
```

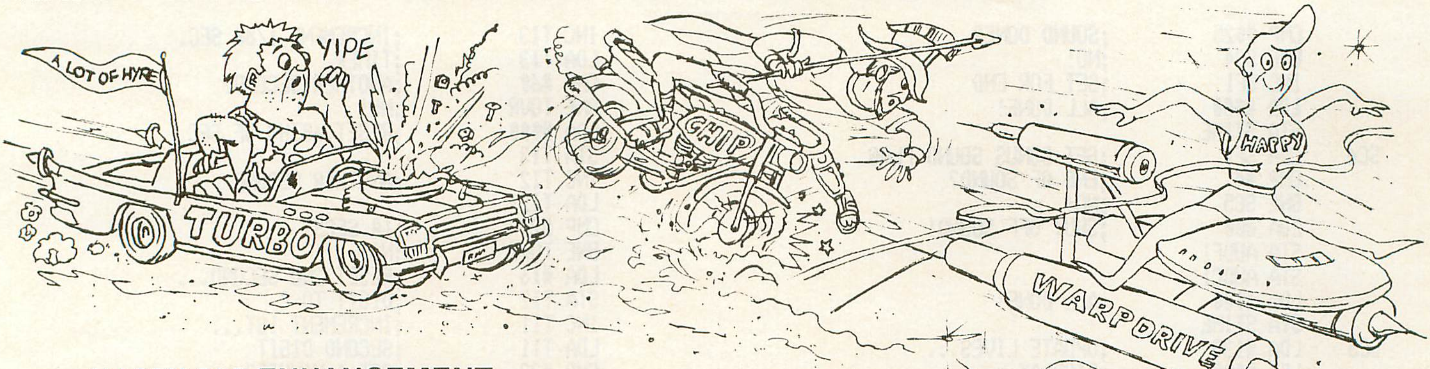
```
; Read joystick

JOY2    LDA STICK0      ;GET ASTRONAUT...
        AND #1          ;DIRECTION
        BNE O1
        DEC Y           ;UP
O1      LDA STICK0
        AND #$08
        BNE O2
        INC X           ;RIGHT
O2      LDA STICK0
        AND #$02
        BNE O3
        INC Y           ;DOWN
O3      LDA STICK0
        AND #$04
        BNE O4
        DEC X           ;LEFT
O4      LDA X           ;GET ASTRO. X
        CMP #200        ;TOO FAR RIGHT?
        BNE O5          ;IT'S OK
        DEC X           ;TOO FAR!
O5      LDA X
        CMP #50         ;TOO FAR LEFT?
        BNE O6          ;OK.
        INC X           ;TOO FAR!
O6      LDA Y           ;GET ASTRO. Y
        CMP #52         ;TOO FAR UP?
        BNE O7          ;OK.
        INC Y           ;TOO FAR!
O7      LDA Y
        CMP #200        ;TOO FAR DOWN?
        BNE O8          ;OK.
        DEC Y           ;TOO FAR!

; Put astronaut on playfield

O8      LDA X           ;GO AHEAD AND SET...
        STA HPOSP0      ;HOR. POSITION
        LDX #$00        ;NOW COPY THE...
        LDY Y           ;ASTRONAUT IMAGE...
B1      LDA ANAUT,X     ;INTO PLAYER 1
        STA (PBASE),Y
        INX
        INY
        CPX #19
        BNE B1

; Move missiles

        LDX #$00
D0      LDA DM,X        ;GET MISSILE DIR.
        BNE D1
        INC XM,X        ;RIGHT...
        DEC YM,X        ;AND UP
D1      LDA DM,X
        CMP #1
        BNE D2
        INC XM,X        ;RIGHT...
        INC YM,X        ;AND DOWN!
D2      LDA DM,X
        CMP #2
        BNE D3
        DEC XM,X        ;LEFT...
        INC YM,X        ;AND DOWN!
D3      LDA DM,X
        CMP #3
        BNE DO1
        DEC XM,X        ;LEFT...
        DEC YM,X        ;AND UP!
DO1     INX             ;NEXT MISSILE
        CPX #6          ;ALL DONE?
        BNE D0          ;NO, DO OTHERS!

; Check if missile has hit wall

        LDX #$00
C0      LDA XM,X        ;GET X POS.
        CMP #202        ;TOO FAR RIGHT?
        BNE C1          ;IT'S OK.
        LDA DM,X        ;IT'S TOO FAR,
        BNE DI1         ;WE MUST REVERSE
        LDA #3          ;THE DIRECTION!
        STA DM,X
        CLC
        BCC C1
DI1     LDA #2
        STA DM,X
C1      LDA XM,X        ;GET X POS.
        CMP #48         ;TOO FAR LEFT?
        BNE C2          ;OK.
        LDA DM,X        ;REVERSE DIRECTION!
        CMP #2
        BEQ DI2
        LDA #0
        STA DM,X
        CLC
        BCC C2
DI2     LDA #1
        STA DM,X
C2      LDA YM,X        ;GET Y POS.
        CMP #52         ;TOO FAR UP?
        BNE C3          ;OK.
        LDA DM,X        ;REVERSE!
        BNE DI3
        LDA #1
        STA DM,X
        CLC
        BCC C3
DI3     LDA #2
        STA DM,X
C3      LDA YM,X        ;GET Y POS.
        CMP #213        ;TOO FAR DOWN?
        BNE C4          ;OK.
        LDA DM,X        ;REVERSE!
        CMP #1
        BNE DY2
        LDA #$00
        STA DM,X
        CLC
        BCC C4
DY2     LDA #3
        STA DM,X
C4      INX             ;NEXT MISSILE
        CPX #6          ;ALL DONE?
        BNE C0          ;NO, DO OTHERS!

; Plot missiles on playfield

        LDA XM          ;SET ALL MISSILES...
        STA HPOSP1      ;HORIZONTAL...
        LDA XM+1        ;POSITIONS!
        STA HPOSP2
        LDA XM+2
        STA HPOSP3
        LDA XM+3
        STA HPOSM1
        LDA XM+4
        STA HPOSM2
        LDA XM+5
        STA HPOSM3
        LDA #$00        ;PLOT 1ST MISSILE
        LDY YM          ;IN PLAYER 1
        STA (BASE2),Y
        LDA #$10
        INY
        STA (BASE2),Y
        INY
        STA (BASE2),Y
        LDA #$00
        INY
        STA (BASE2),Y
        LDA #$00        ;PLOT 2ND MISSILE
        LDY YM+1        ;IN PLAYER 2
        STA (BASE3),Y
        LDA #$10
        INY
        STA (BASE3),Y
        INY
        STA (BASE3),Y
        LDA #$00
        INY
        STA (BASE3),Y
        LDA #$00        ;PLOT 3RD MISSILE
        LDY YM+2        ;IN PLAYER 3
        STA (BASE4),Y
        LDA #$10
        INY
        STA (BASE4),Y
        INY
        STA (BASE4),Y
        LDA #$00
```

```
        INY
        STA (BASE4),Y
        LDA #$EF        ;PLOT 4TH MISSILE
        LDY YM+3        ;IN MISSILE 1
        AND (MBASE),Y
        STA (MBASE),Y
        LDA #$10
        INY
        ORA (MBASE),Y
        STA (MBASE),Y
        LDA #$10
        INY
        ORA (MBASE),Y
        STA (MBASE),Y
        LDA #$EF        ;PLOT 5TH MISILE
        INY             ;IN MISSILE 2
        AND (MBASE),Y
        STA (MBASE),Y
        LDA #$FB
        LDY YM+4
        AND (MBASE),Y
        STA (MBASE),Y
        LDA #$04
        INY
        ORA (MBASE),Y
        STA (MBASE),Y
        LDA #$04
        INY
        ORA (MBASE),Y
        STA (MBASE),Y
        LDA #$FB        ;PLOT 6TH MISSILE
        INY             ;IN MISSILE 3
        AND (MBASE),Y
        STA (MBASE),Y
        LDA #$BF
        LDY YM+5
        AND (MBASE),Y
        STA (MBASE),Y
        LDA #$40
        INY
        ORA (MBASE),Y
        STA (MBASE),Y
        LDA #$40
        INY
        ORA (MBASE),Y
        STA (MBASE),Y
        LDA #$BF
        INY
        AND (MBASE),Y
        STA (MBASE),Y

; Create sound

        INC T1
        LDA T1
        CMP #$10
        BNE SO1
        LDA #$00
```

```
        STA T1
        DEC T2
        LDA T2
        BNE SO2
        LDA #3
        STA T2
SO2     LDA #$C8        ;SET SOUND CONTROL
        STA AUDC2
        LDA T2          ;GET SOUND NUMBER
        CMP #3          ;SOUND 3?
        BNE S2          ;NO!
        LDA #120        ;SET 3RD...
        STA AUDF2       ;SOUND FREQUENCY
        CLC
        BCC SO1         ;GO SHOW TIME
S2      LDA T2
        CMP #2          ;SOUND 2?
        BNE S3          ;NO!
        LDA #100        ;SET 2ND...
        STA AUDF2       ;SOUND FREQUENCY
        CLC
        BCC SO1         ;AND SHOW TIME
S3      LDA #85         ;SET 1ST
        STA AUDF2       ;SOUND FREQUENCY
SO1     LDA TI1         ;SHOW DIGIT 1...
        STA SCREEN+9    ;OF SECONDS
        LDA TI2         ;SHOW DIGIT 2...
        STA SCREEN+10   ;OF SECONDS
        LDA TI4         ;SHOW MINUTES!
        STA SCREEN+7

; Check for extra life

        LDA TI4         ;GET MINUTES
        CMP #19         ;3 MINUTES?
        BNE OELC        ;NO!
        LDA ELF         ;DONE EXTRA LIFE YET?
        BNE OELC        ;YES, NO MORE!
        INC LIFE        ;ANOTHER LIFE
        INC ELF         ;SET EXTRA LIFE FLAG
        LDA #$A4        ;SET UP FOR...
        STA AUDC1       ;EXTRA LIFE SOUND
        LDA #$01
        STA SF1
OELC    LDA TI4         ;GET MINUTES
        CMP #21         ;5 MINUTES?
        BNE OEL         ;NO!
        LDA ELF         ;GOT 2ND EXTRA?
        CMP #1
        BNE OEL         ;NO MORE!
        INC LIFE        ;ANOTHER LIFE!
        INC ELF         ;INC EXTRA LIFE FLAG
        LDA #$A4        ;SET UP FOR...
        STA AUDC1       ;EXTRA LIFE SOUND
        LDA #$01
        STA SF1
```

```
; Check for ending sequence

OEL     LDA HIT         ;ASTRO HIT DEBRIS?
        BEQ NEND        ;NO, HE'S OK.
        JMP END         ;UH-OH!
NEND    LDA #80         ;THIS SECTION
        LSR TI1         ;CREATES A TIME
        PHP             ;DELAY USING
        SEC             ;THE TIME VARIABLES
        SBC TI1         ;TI1 AND TI4
        PLP
        ROL TI1
        LDX #$00
SMOR    SBC TI4
        INX
        CPX #3
        BNE SMOR
        TAX
TD1     LDY #$00        ;THIS IS A
TD2     INY             ;TIME DELAY
        BNE TD2         ;ROUTINE!
        DEX
        BNE TD1

; Play extra life sound

        LDA SF1         ;GET BONUS SOUND FLAG
        CMP #1          ;FIRST FREQ?
        BNE SC2         ;NO!
        LDA #50         ;SET 1ST BONUS FREQ.
        STA AUDF1
        INC STIME       ;INC SOUND TIMER
        LDA STIME
        CMP #$25        ;END OF SOUND?
        BNE SC2         ;NOT YET!
        INC SF1         ;NEXT FREQ
        LDA #$00        ;NO MORE SOUND!
        STA STIME
SC2     LDA SF1         ;GET BONUS SOUND FLAG
        CMP #2          ;SECOND FREQ?
        BNE SC3         ;NO!
        LDA #40         ;SET FREQ.
        STA AUDF1
        INC STIME       ;INCREMENT TIMER
        LDA STIME
        CMP #$25        ;SOUND DONE?
        BNE SC3         ;NO!
        INC SF1         ;NEXT FREQ
        LDA #$00        ;ALL DONE!
        STA STIME
SC3     LDA SF1         ;GET BONUS SOUND FLAG
        CMP #3          ;3RD FREQ?
        BNE SC4         ;NO!
        LDA #32         ;SET FREQ.
        STA AUDF1
        INC STIME       ;INC SOUND TIMER
        LDA STIME
```

HAPPY WINS THE RACE WITH WARP DRIVE SPEED!



HAPPY 810 ENHANCEMENT

- The only change needed to run all WARP DRIVE software
- Plug in P.C. board requires no permanent modifications
- Proven reliable in thousands of installations, reduces disk drive wear
- Comes completely assembled and tested, just plug in and use
- Full one year parts and labor guarantee, compatible with existing software
- High quality printed circuit board with gold connectors

NO ONE ELSE HAS THIS PERFORMANCE
Unenhanced whole disk (ATARI rev B format) read time: 112 seconds
Unenhanced whole disk (ATARI rev C fast format) read time: 89 seconds
ENHANCED 810 whole disk (any format) read time with standard software: 68 seconds
ENHANCED 810 whole disk (any format) read time with WARP DRIVE software: 43 seconds
Standard software whole disk write and verify time: 238 seconds
WARP DRIVE software whole disk write and verify time: 62 seconds

NEW HAPPY WARP DRIVE SOFTWARE

WARP SPEED HAPPY BACKUP PROGRAM

- Completely automatic: nothing to figure out, insert disks and press return
- Only program on the market guaranteed to backup any disk
- Can write to a blank disk: format write and verify in one operation
- Automatic program tracing: copies only the tracks that are used
- Efficient memory utilization: reduces the number of disk insertions
- Requires only one ENHANCED disk drive, backups will work on a standard drive

WARP SPEED MULTI DRIVE HAPPY BACKUP PROGRAM

- Same features as above plus support of multiple ENHANCED drives
- Can be used with up to 4 ENHANCED drives
- Source and all destination drives read and write in parallel
- Format write and verify 3 complete disks in less than 3 minutes

WARP SPEED HAPPY COMPACTOR PROGRAM

- Reduces the number of disks required to backup your library
- Combines up to 8 self booting disks into 1 disk with a menu
- Compacted disks run only on an ENHANCED drive
- Pays for itself by saving on disks
- Single or dual ENHANCED drive operation

HAPPY WARP DRIVE DOS

- Improves ATARI DOS 2.0S to use warp speed reading and write with verify
- Use all features of BASIC, PILOT, FMS, and DUP at top warp speed
- Warp speed I/O software module available separate from DOS

HAPPY WARP DRIVE SECTOR COPY PROGRAM

- Standard format whole disk read, write and verify in 105 seconds
- Use with sngle or dual drives, mix ENHANCED and NON-ENHANCED drives

HAPPY CUSTOMIZER PROGRAM (sold separately $99.95)

- Creates custom format disks of any specification
- Any type bad sector, duplicate sector numbers, or interleave
- Easy to use but requires an advanced level user to interpret the results

REVIEWED IN POPULAR MAGAZINES

A.N.A.L.O.G. COMPUTING—July/August 1983 *"...The installation instructions for the Happy 810 Enhancement are among the best I have ever seen. ...The Happy 810 Enhancement is one of the most powerful hardware modifications available to ATARI computer owners."*

ANTIC—July 1983 *"The difference between a normal ATARI 810 disk drive and one equipped with Happy is like the contrast between mass transit and the automobile. A car costs you more initially, but improves the quality of your life. Similarly, if you use your disk drive a lot, installing Happy will markedly enhance your programming life."*

SPECIAL SUGGESTED RETAIL PRICE BEFORE FEBRUARY 28, 1984: *Get the HAPPY 810 ENHANCEMENT with the single and multi drive HAPPY BACKUP PROGRAM, plus the HAPPY COMPACTOR PROGRAM, plus the HAPPY DRIVE DOS, plus the HAPPY SECTOR COPY, all with WARP DRIVE speed, including our diagnostic for $249.95. Existing registered ENHANCEMENT owners may upgrade to WARP DRIVE speed for $15.00 with no hardware changes.!*

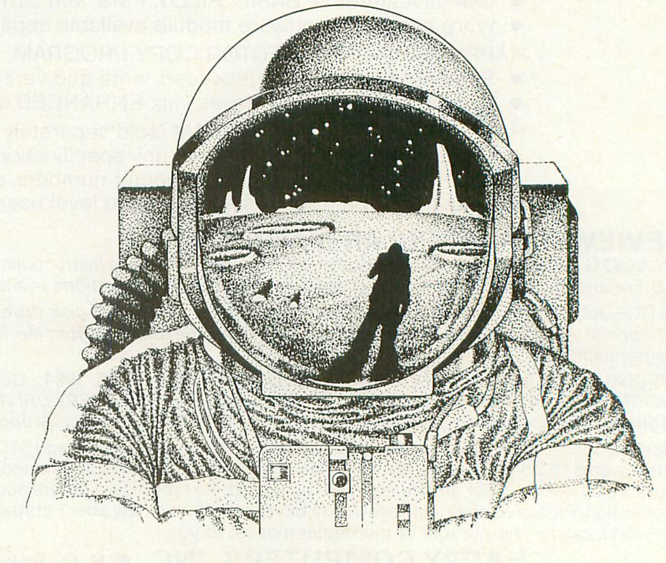
Price includes shipping by air mail to U.S.A. and Canada. Foreign orders add $10.00 and send an international money order payable through a U.S.A. bank. California orders add $16.25 state sales tax. Cashiers check or money order for immediate shipment from stock. Personal checks require 2-3 weeks to clear. Cash COD available by phone order and charges will be added. No credit card orders accepted. ENHANCEMENTS for other ATARI compatible drives coming soon, call for information. Please specify -H model for all drives purchased new after February 1982, call for help in ENHANCEMENT model selection. Dealers now throughout the world, call for the number of the dealer closest to you.

ATARI 810 is a registered trademark of Atari, Inc.

HAPPY COMPUTERS, INC. • P. O. Box 1268 • Morgan Hill, California 95037 • (408) 779-3830

```
        CMP #$25        ;SOUND DONE?
        BNE SC4         ;NO!
        INC SF1         ;SET FOR END
        LDA #$00        ;ALL DONE!
        STA STIME
SC4     LDA SF1         ;GET BONUS SOUND FLAG
        CMP #4          ;END OF SOUND?
        BNE SC5         ;NO!
        LDA #00         ;SHUT OFF SOUND!
        STA AUDF1
        STA AUDC1
        LDA #$00        ;ALL DONE!
        STA STIME
SC5     LDA #138        ;UPDATE LIVES...
        LDX #$00        ;DISPLAY
DMORE   STA SCREEN+13,X ;ALL LIVES
        INX
        CPX LIFE
        BNE DMORE
        JMP BACK
END     DEC LIFE        ;1 LESS LIFE
        LDA #$A8        ;INIT DEATH SOUND
        STA AUDC1
        STA AUDC2
        LDA #50         ;INIT FREQ.
SA      STA AUDF1
        PHA
        CLC
        ADC #100        ;INCREMENT SOUND
        STA AUDF2
        STA PCOLR0      ;FLASH ASTRONAUT
        PLA
        LDY #$00        ;TIME DELAY
DLY2    LDX #$00
DLY1    INX
        BNE DLY1
        INY
        CPY #25
        BNE DLY2
        CLC
        ADC #$01
        CMP #100        ;END OF DEATH?
        BNE SA          ;NO, LOOP BACK
        LDA #$00        ;SHUT OFF SOUND
        STA AUDF1
        STA AUDC1
        STA AUDF2
        STA AUDC2
        LDA #$46        ;RESET...
        STA PCOLR0      ;ASTRONAUT COLOR
        LDA #$00        ;REMOVE...
        LDY LIFE        ;LIFE INDICATOR...
        STA SCREEN+13,Y ;FROM SCREEN
        LDA LIFE        ;ANY MORE LIVES?
        BEQ DEATH       ;NO! END OF GAME!
        JMP START       ;ANOTHER LIFE.
DEATH   LDA #00         ;ZERO LIVES
        STA LIFE
AGAIN   LDA STRIG0      ;STICK TRIGGER PRESSED?
        BEQ RESTRT      ;YES!
        LDA CONSOL      ;GET CONSOLE KEY
        CMP #6          ;START PRESSED?
        BNE AGAIN       ;NO!
RESTRT  LDA #$00        ;RESET...
        STA ATTRAC      ;ATTRACT MODE
        JMP START2      ;RESTART GAME.

; Vertical blank interrupt

VBI     LDA M1PL        ;GET...
        ORA M2PL        ;ALL...
        ORA M3PL        ;COLLISION...
        ORA P1PL        ;REGISTERS...
        ORA P2PL        ;AND CHECK...
        ORA P3PL        ;FOR...
        AND #1          ;ASTRONAUT...
        STA HIT         ;COLLISION!
        INC TI3         ;INCREMENT 1/60 SEC.
        LDA TI3         ;TIMER
        CMP #60         ;ANOTHER SECOND?
        BNE TOVR        ;NO!
        LDA #$00        ;RESET 60THS OF SEC
        STA TI3
        INC TI2         ;ANOTHER SECOND
        LDA TI2
        CMP #26         ;10 SECS?
        BNE TOVR        ;NO!
        LDA #16         ;RESET 2ND SECOND...
        STA TI2         ;DIGIT TO 0
        INC TI1         ;INCREMENT 1ST...
        LDA TI1         ;SECOND DIGIT
        CMP #22         ;GOT A MINUTE?
        BNE TOVR        ;NO!
        INC TI4         ;INCREMENT MINUTES
        LDA #16         ;RESET 1ST SECOND...
        STA TI1         ;DIGIT TO 0
TOVR    JMP XITVBV      ;EXIT VERT BLANK!

; Set up display list

DLIST   DB  112,112,112,71
        DW  SCREEN
        DB  9,9,9,9,9,9,9,9,9,9
        DB  9,9,9,9,9,9,9,9,9,9
        DB  9,9,9,9,9,9,9,9,9,9
        DB  9,9,9,9,9,9,9,9,9,9
        DB  9,9,9,9
        DB  65
        DW  DLIST

; Data for astronaut image

ANAUT   DB  $00,$7C,$54,$6C,$7C,$44,$7C,$10,$10,$FE,$BA,$BA
        DB  $BA,$38,$38,$28,$28,$EE,$00

; Set up screen memory

MIDBYT  DB  128,0,0,0,0,0,0,0,0,1
SCREEN  DB  0,0,116,105,109,101,0,0,26,0,0,0,0,0,0,0,0,0,0
SCRNA   DS  10
SCRNB   DS  420
SCRNC   DS  10

        END SSTAR
```

●



SOLO FLIGHT
by Sid Meier
MICROPROSE SOFTWARE
10616 Beaver Dam Road
Hunt Valley, MD 21030
48K Cassette or Disk $34.95

---

by Lee Pappas

Finally a flight simulator for Atari Home Computers. This review includes, at no extra charge, your first ground school lesson (unless you're already a pilot).

Solo Flight allows you to practice several types of flying scenarios, including cross-country and instrument-only exercises. The screen is divided into two parts. Your instrument panel resides on the bottom half; above it is a display of your plane as seen from behind, along with appropriate surroundings. The latter may consist of the ground, clouds or lots more clouds. Should you see "lots more clouds," you'll have to fly IFR. IFR is pilot talk for Instrument Flight Rules, as opposed to VFR, Visual Flight Rules.

Your flight instruments consist of an altimeter (altitude gauge), airspeed indicator, a small artificial horizon (attitude gauge), power indicator and fuel gauge. Numeric readouts include pitch, flap angle, magnetic heading (compass) and climb rate. Two important readouts, VOR 1 and 2, are numeric. This differs significantly from the VOR units in real aircraft, as they are analog gauges. An ILS (Instrument Landing System) readout, landing gear up/down, brakes on/off and engine overheat indicators are also included.

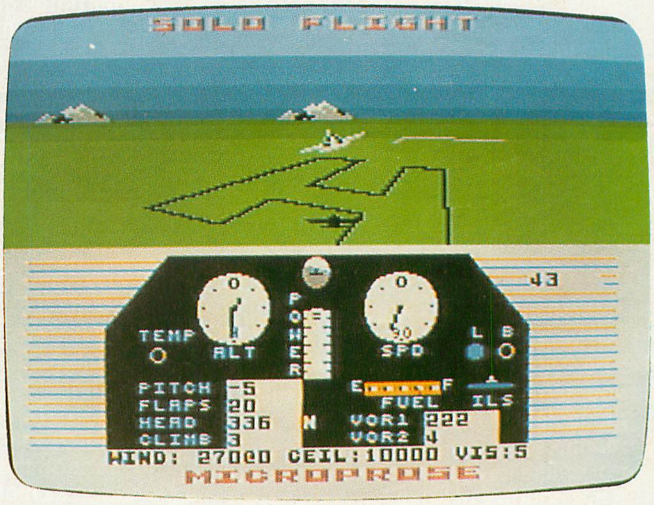
Control of the aircraft is achieved using the joystick and keyboard. Pushing the stick to the left or right will turn your plane in that direction. Pulling back will make the plane climb; pushing forward will tilt the plane's nose down. The computer's number keys control your speed (*a la* **Star Raiders**), the F key will put the flaps down in 20-degree increments, and the S key actuates or releases the breaks.

The view of "what's outside" centers on the aircraft you're controlling. You can "look" out the left, right or back window by pushing the corresponding cursor control key. As you pilot, the aircraft will accurately respond to your actions; even the pitch of the plane changes, which I thought was really neat.

If you opt to play IFR, you'll lose your top visual display (it turns grey), and it'll be just you and your instrument panel. If you keep the plane climbing, you'll eventually rise above the cloud layer and get your visual display back — except the ground will be thousands of feet below, and you'll have no idea where you are. Now you'll really have to know how to use your navigational instruments. The **Solo Flight** reference manual explains the use of these, along with rudimentary flight procedures. For experienced (real life) pilots: while you may not have to file a flight plan, you'll have fun showing off to your friends who never "had time" to go up with you.

Flying IFR can be a lot of fun, but can easily spell curtains until you get the hang of things. The sense of accomplishment you feel when you come out of the clouds, right on top on the airport you were flying to, far exceeds that of just getting a high score zapping space invaders. At the end of an IFR run, you'll be shown an on-screen map of your flight route (this could resemble anything from a clear cut line to somebody's signature!).

The nicest parts of **Solo Flight** are its navigation features. Three rough maps are included: Eastern Kansas, Western Washington state, and mid-Colorado, each representing a different type of terrain. All three regional maps include 7 airports and 2 VOR towers. Kansas is better for "student" pilots; the other two states are trickier because of mountains and higher ground elevations. Other features of the simulation allow you to practice landings, "staying in the pattern for touch and go's" (landings and takeoffs), and windy conditions.



The ground display is made up of green multi-shades, and the sky, blue shades. Several puffy cumulous clouds reside on the distant horizon. The detail of ground objects in **Solo Flight** leaves a bit to be desired; however, I have yet to see truly realistic ground objects on any home flight simulator. Visible objects include airfields, mountains, cities and VOR towers. Airports are black, and roughly resemble their real-life counterparts. Cities look like horizontal outlines in grey. The mountains have the weakest imaging of all. They are merely white outlines, which you can see through (and even fly through!).

As a whole, **Solo Flight** is the best Atari flight simulator published to date. Though the graphics are somewhat rough, and the control panel is not up to what it should be (there is no stall indicator, and non-standard VORs), if you've been looking for a program of this type, you'll have a lot of fun on your **Solo Flight**. ☐

FINE SCROLLING PART IV

TAKING THE PLUNGE

16K Cassette or Disk

by Kyle Peacock

Well, this is it. For the past two issues I've been babbling about how much we need assembly language to accomplish smooth fine scrolling. Hopefully you've also been keeping up with Tom Hudson's **Boot Camp** series. If not, don't sweat. I've included both BASIC and assembly listings so that novices and advanced programmers alike can enjoy the pleasures of fine scrolling. If you've been keeping up and have a basic understanding of the material covered in issues 13 through 15, you are already well on your way to mastering fine scrolling (and maybe even publishing a best-selling game).

The following BASIC listing contains all the code we need to secure our objective. The assembly-language routines are contained in the DATA statements on Lines 8000-10000. All you need to do is get a grip on the parameters passed to my USR routine and the restrictions on said parameters.

The USR call in Line 790 will initialize things. This line can be located anywhere you like, as long as all the proper strings have been initialized. I've done my best to be as diverse as possible. Consequently, there are slightly over a dozen parameters to worry about. Let's examine the USR call and its associated parameters:

```
A=USR(INIT,VBLANK,DLIST,LINES,MINX,MAX
X,MINY,MAXY,HMAX,VMAX,HSPEED,VSPEED,ST
ICK,HINVERT,VINVERT,PAGES)
```

INIT. This is the address where the USR call will begin execution. It is specified by using the ADR function in BASIC. The argument of the ADR function is INIT$, which is initialized with the data in Lines 8000 to 8999.

VBLANK. This is the address of my vertical blank routine. Its definition follows the same criteria of the INIT parameter, and is initialized with the data in Lines 9000 to 9999.

DLIST. This is the address of your freshly-generated display list. You should initialize DLIST$ by placing your display list data into each consecutive string element. You should also DIMension DLIST$ to the number of bytes in your display list, *plus two*. Your display list *must* end with the jump and wait for vertical blank (JVB) opcode. Do *not* try to install the associated JBV operands yourself! Lines 480 to 510 will do it for you.

LINES. This is the total number of actual lines used by your display. It is *not* the number of lines displayed at any one time. Rather, it is the total range of lines of text (or bit-mapped graphics) that will be smooth-scrolled. This number will be equal to the number of hi/lo pointers in the PAGES parameter (see next page).

MINX. The minimum value for horizontal scrolling.

MAXX. The maximum value for horizontal scrolling.

MINY. The minimum value for vertical scrolling.

MAXY. The maximum value for vertical scrolling.

The above MIN and MAX parameters dictate the outer boundaries of the scrolling area. Since smooth-scrolling requires the changing of hardware registers and the updating of LMS operands, the assembly-language routine needs to

keep track of the number of operand updates (e.g.; byte shifts). If the specified boundaries are exceeded, scrolling is halted. MINX and MINY are associated with the leftmost and uppermost boundaries, respectively. MAXX and MAXY are associated with the rightmost and lowermost boundaries. MAXY should be initialized to LINES, minus the total number of LMS instructions in your display list.

HMAX. This is the maximum value of the hardware register HSCROL. It is dependent on the graphics mode. The following table lists graphics modes and their associated HMAX value.

Graphics Mode	HMAX Value
0	3
1	7
2	7
3	15
4	15
5	7
6	7
6+	7
7	3
7+	3
8	3

VMAX. This is the maximum value of the hardware register VSCROL. If you'll recall, last issue I listed a table of graphics modes and their associated VSCROL ranges (Issue 15, page 106). Reference this table to find the correct value of VMAX for your application.

HSPEED. The speed of horizontal scrolling.

VSPEED. The speed of vertical scrolling.

The SPEED parameters dictate how fast the display will scroll. The value stored in these parameters should be specified in jiffies (60ths of a second). For example, a value of 1 will cause the display to scroll one position every 1/60 of a second. A value of 60 will cause the display to scroll once every second. A value of zero will terminate scrolling altogether. A negative value in these parameters will probably screw things up, so don't try it!

STICK. This is the joystick number to be read. Valid ranges are 0 to 3. Every time the associated stick is moved, the display will scroll accordingly. This parameter is handy for multiple-user applications.

HINVERT. The horizontal scroll inversion flag.

VINVERT. The vertical scroll inversion flag.

The INVERT parameters invert the scrolling direction when they have a non-zero value. For example, if HINVERT is zero, the display will scroll to the left when the joystick is moved to the left. If HINVERT is one, the display will scroll to the right when the joystick is moved to the left. These parameters are useful when you wish to create a scrolling map-like display.

PAGES. This parameter is the address of a list of hi/lo pointers used for screen RAM. PAGES$ should be initialized with the high and low pointers of the RAM areas you wish scrolled. For example, if your screen RAM uses page 8, page 12 and page 20, PAGES$ should contain 8, 0, 12, 0, 20 and 0. Keep in mind that PAGES$ is structured in hi/lo byte fashion. The LINES parameter should equal the number of pairs of screen RAM pointers (three in this case).

Disclaimer.

Although I have tried to incorporate as many features as possible into these routines, some of you out there may find they will not work for your particular application. For example, the routines will bomb out if you have an LMS instruction that is over 255 bytes away from the beginning of the display list. This problem cannot be easily remedied, as it was a design consideration. Should you encounter any other strange problems, feel free to send me a disk or tape with your program, an explanation of the problem, and a self-addressed stamped envelope. Time permitting, I'll asess the damage and (hopefully) come up with a solution.

Wow! Other than the program listing, I'm all done. I had no idea how involving this column would be. From now on I'll stick to playing and reviewing games. Maybe one day I'll review your masterpieces, incorporating fine scrolling. Until then, see you in the funny papers. □

BASIC Listing.

```
100 REM ***************************
110 REM *    SMOOTH SCROLL DEMO   *
120 REM *     BY  KYLE PEACOCK    *
130 REM * A.N.A.L.O.G. COMPUTING *
140 REM *        ISSUE 16         *
150 REM * ALL RIGHTS UNRESERVED! *
160 REM ***************************
170 REM
180 PRINT "K"
190 REM
200 REM - DIMENSION INIT$ & VBLANK$
210 REM - EXACTLY AS THEY ARE HERE.
220 REM
230 DIM INIT$(199),VBLANK$(408)
240 REM
250 REM - DIMENSIONS OF PAGES$ &
260 REM - DLIST$ WILL DEPEND ON
270 REM - APPLICATION
280 REM
290 DIM PAGES$(112),DLIST$(78)
300 PRINT "PHASE 0 COMPLETE."
310 REM
320 REM - INSTALL PAGES$ DATA.
330 REM - LOOP VARIABLE SHOULD BE 1
```

```
340 REM - TO # OF PAGE BYTES DIVIDED
350 REM - BY TWO.
360 REM
370 FOR X=1 TO 56:READ A,B
380 PAGES$(X*2-1,X*2-1)=CHR$(B)
390 PAGES$(X*2,X*2)=CHR$(A):NEXT X
400 ? "PHASE 1 COMPLETE."
410 REM
420 REM - INSTALL DLIST DATA
430 REM - LOOP VARIABLE SHOULD BE 1
440 REM - TO # OF DLIST BYTES
450 REM
460 FOR X=1 TO 76:READ A
470 DLIST$(X,X)=CHR$(A):NEXT X
480 HIGH=INT(ADR(DLIST$)/256)
490 LOW=ADR(DLIST$)-HIGH*256
500 DLIST$(X,X)=CHR$(LOW)
510 DLIST$(X+1,X+1)=CHR$(HIGH)
520 ? "PHASE 2 COMPLETE."
530 REM
540 REM - INSTALL VBLANK INITIALIZER
550 REM
560 FOR X=1 TO 199:READ A
570 INIT$(X,X)=CHR$(A):NEXT X
580 ? "PHASE 3 COMPLETE."
590 REM
600 REM - INSTALL VBLANK ROUTINE
610 REM
620 FOR X=1 TO 408:READ A
630 VBLANK$(X,X)=CHR$(A):NEXT X
640 REM
650 REM - ALL DONE...
660 REM
670 ? "PHASE 4 COMPLETE."
680 REM
690 REM - POKE A 1 INTO MEM. LOC.
700 REM - 1791 TO HALT EVERYTHING.
710 REM - IF ANY PARAMETERS NEED TO
720 REM - BE CHANGED ONCE THINGS GET
730 REM - GOING, POKE 1791,1, THEN
740 REM - PERFORM THE USR CALL WITH
750 REM - THE NEW PARAMETERS.
760 REM - FINALLY, POKE 1791,0
770 REM
780 POKE 1791,1
790 A=USR(ADR(INIT$),ADR(VBLANK$),ADR(
DLIST$),56,0,255,0,32,7,7,1,1,0,0,0,AD
R(PAGES$))
800 POKE 1791,0
810 GOTO 810
820 REM
830 REM - RAM PAGE USAGE
840 REM
850 DATA 0,0,1,0,2,0,3,0
860 DATA 250,0,251,0,252,0,253,0
870 DATA 254,0,255,0
880 DATA 3,0,1,0,0,0,253,0
890 DATA 250,0,5,0,4,0,250,0
900 DATA 5,0,2,0,0,0,4,0
910 DATA 253,0,241,0,222,0,3,0
920 DATA 247,0,212,0,2,0,1,0
930 DATA 3,0,5,0,252,0,253,0
940 DATA 255,0,254,0,253,0,252,0
950 DATA 251,0,250,0,249,0,248,0
960 DATA 1,0,3,0,2,0,5,0
970 DATA 4,0,248,0,252,0,253,0
980 DATA 0,0,3,0,6,0,4,0
990 DATA 254,0,255,0
1000 REM
1010 REM - DISPLAY LIST
1020 REM
1030 DATA 112,112,112,118,0,0
1040 DATA 118,0,0,118,0,0,118,0,0
1050 DATA 118,0,0,118,0,0,118,0,0
1060 DATA 118,0,0,118,0,0,118,0,0
1070 DATA 118,0,0,118,0,0,118,0,0
1080 DATA 118,0,0,118,0,0,118,0,0
1090 DATA 118,0,0,118,0,0,118,0,0
1100 DATA 118,0,0,118,0,0,118,0,0
1110 DATA 118,0,0,86,0,0,65
8000 REM
8010 REM - VBLANK INITIALIZER
8020 REM
8030 DATA 165,16,41,127,133,16,141
8040 DATA 14,210,216,104,104,141,28
8050 DATA 6,104,141,27,6,104,141
8060 DATA 1,6,104,141,0,6,104
8070 DATA 104,141,8,6,104,104,141
8080 DATA 2,6,141,18,6,104,104
8090 DATA 141,4,6,104,104,141,3
8100 DATA 6,141,19,6,104,104,141
8110 DATA 5,6,104,104,141,6,6
8120 DATA 104,104,141,7,6,104,104
8130 DATA 141,10,6,104,104,141,11
8140 DATA 6,104,104,141,9,6,104
8150 DATA 104,141,12,6,104,104,141
8160 DATA 13,6,104,133,204,104,133
8170 DATA 203,169,1,141,16,6,141
8180 DATA 17,6,165,205,72,165,206
8190 DATA 72,173,0,6,133,205,173
8200 DATA 1,6,133,206,160,0,162
8210 DATA 0,142,15,6,142,15,6
8220 DATA 177,205,141,26,6,41,15
8230 DATA 240,33,173,26,6,41,64
8240 DATA 240,26,200,200,173,26,6
8250 DATA 41,15,201,1,240,18,173
8260 DATA 26,6,41,48,240,8,152
8270 DATA 56,233,2,157,29,6,232
8280 DATA 200,208,211,142,20,6,104
8290 DATA 133,206,104,133,205,169,7
8300 DATA 174,28,6,172,27,6,32
8310 DATA 92,228,96
8320 REM * 199 BYTES
9000 REM
9010 REM - VERTICAL BLANK ROUTINE
9020 REM
9030 DATA 173,255,6,240,3,76,98
9040 DATA 228,216,173,0,6,141,48
9050 DATA 2,173,1,6,141,49,2
9060 DATA 206,16,6,208,124,173,10
9070 DATA 6,141,16,6,240,116,174
9080 DATA 9,6,189,120,2,172,12
9090 DATA 6,208,12,41,12,201,8
9100 DATA 240,103,201,4,240,12,208
9110 DATA 93,41,12,201,4,240,91
9120 DATA 201,8,208,123,238,14,6
9130 DATA 173,14,6,205,6,6,240
9140 DATA 112,144,110,173,18,6,205
9150 DATA 2,6,208,7,206,14,6
9160 DATA 240,97,208,95,206,18,6
9170 DATA 169,255,72,48,7,169,0
9180 DATA 141,14,6,240,80,141,22
9190 DATA 6,104,141,23,6,174,8
9200 DATA 6,202,160,0,177,203,24
9210 DATA 109,22,6,145,203,200,177
9220 DATA 203,109,23,6,145,203,200
9230 DATA 202,16,236,173,22,6,48
9240 DATA 211,16,34,240,38,208,36
9250 DATA 206,14,6,16,31,173,18
9260 DATA 6,205,4,6,208,7,238
9270 DATA 14,6,240,18,208,16,238
9280 DATA 18,6,169,0,72,169,1
9290 DATA 16,182,173,6,6,141,14
9300 DATA 6,206,17,6,208,110,173
9310 DATA 11,6,141,17,6,240,102
9320 DATA 174,9,6,189,120,2,172
9330 DATA 13,6,208,12,41,3,201
9340 DATA 2,240,49,201,1,240,12
9350 DATA 208,79,41,3,201,1,240
9360 DATA 37,201,2,208,69,206,15
9370 DATA 6,16,64,173,19,6,205
9380 DATA 3,6,208,7,238,15,6
9390 DATA 240,51,208,49,206,19,6
9400 DATA 173,7,6,141,15,6,240
9410 DATA 38,208,36,238,15,6,173
9420 DATA 15,6,205,7,6,240,25
9430 DATA 144,23,173,19,6,205,5
9440 DATA 6,208,7,206,15,6,240
9450 DATA 10,208,8,238,19,6,169
9460 DATA 0,141,15,6,173,14,6
9470 DATA 141,4,212,173,15,6,141
9480 DATA 5,212,173,20,6,240,83
9490 DATA 141,21,6,165,205,72,165
9500 DATA 206,72,162,0,142,24,6
9510 DATA 173,19,6,10,141,25,6
9520 DATA 173,48,2,133,205,173,49
9530 DATA 2,133,206,172,25,6,177
9540 DATA 203,72,174,24,6,189,29
9550 DATA 6,168,200,104,145,205,238
9560 DATA 25,6,172,25,6,177,203
9570 DATA 72,189,29,6,168,200,200
9580 DATA 104,145,205,238,24,6,238
9590 DATA 25,6,206,21,6,208,210
9600 DATA 104,133,206,104,133,205,76
```

```
9610 DATA 98,228
9620 REM * 408 BYTES
```

•

CHECKSUM DATA
(See pp. 20-24.)

```
100 DATA 526,476,135,536,542,770,544,9
5,988,101,426,664,82,977,88,6950
250 DATA 244,226,971,100,692,844,81,32
1,649,729,188,96,254,794,626,6815
400 DATA 895,83,537,651,890,95,126,944
,705,9,524,871,906,91,167,7494
550 DATA 97,473,854,927,109,517,87,441
,410,96,687,102,929,108,391,6228
700 DATA 275,677,843,103,562,905,727,1
07,7,81,984,717,94,228,100,6410
850 DATA 748,499,339,866,349,748,372,3
20,336,503,504,765,375,771,347,7842
1000 DATA 275,69,277,91,579,580,581,58
2,583,584,578,36,289,995,291,6390
8030 DATA 887,727,417,132,632,360,583,
441,137,645,602,597,879,883,462,8384
8180 DATA 674,461,577,153,472,606,609,
422,417,608,872,762,339,468,561,8001
9000 DATA 291,457,293,620,480,337,650,
474,402,337,885,587,443,344,886,7486
9150 DATA 342,484,688,407,336,434,886,
912,486,653,594,195,472,192,601,7682
9300 DATA 453,616,618,347,586,583,460,
418,369,480,373,492,175,640,390,7000
9450 DATA 625,174,448,594,664,616,330,
670,492,516,784,664,670,923,625,8795
9600 DATA 953,188,554,1695
```

•

Assembly Listing.

```
0100      .OPT NO LIST
0110 ;
0120 ; *************
0130 ; SYSTEM EQUATES
0140 ; *************
0150 ;
0160 POKMSK = $10       ;POKEY INTERRUPT
0170 SDLSTL = $0230     ;DLIST POINTER
0180 STICK0 = $0278     ;JOYSTICK PORT
0190 IRQEN = $D20E      ;INTER. REQUEST
0200 HSCROL = $D404     ;SCROLL REGISTER
0210 VSCROL = $D405     ;SCROLL REGISTER
0220 SETVBV = $E45C     ;SET TIMERS
0230 XITVBV = $E462     ;EXIT DEF VBLANK
0240 ;
0250 ; *********************
0260 ; MEMORY USAGE & EQUATES
0270 ; *********************
0280 ;
0290 LISTLO = $0600     ;DLIST LO-BYTE
0300 LISTHI = $0601     ;DLIST HI-BYTE
0310 MINX =   $0602     ;LEFTMOST LIMIT
0320 MINY =   $0603     ;UPPER LIMIT
0330 MAXX =   $0604     ;RIGHTMOST LIMIT
0340 MAXY =   $0605     ;LOWER LIMIT
0350 HMAX =   $0606     ;LARGEST HSCROL
0360 VMAX =   $0607     ;LARGEST VSCROL
0370 ALINES = $0608     ;# OF D-LINES
0380 STICK = $0609      ;JOYSTICK #
0390 HSPEED = $060A     ;HOR. SPEED
0400 VSPEED = $060B     ;VER. SPEED
0410 HINVERT = $060C    ;INVERSION FLAG
0420 VINVERT = $060D    ;INVERSION FLAG
0430 PAGES = $CB        ;SCREEN PAGES
0440 ZPAGE = $CD        ;ZERO PAGE INDEX
0450 ;
0460 ; ************************
0470 ; MISCELLANEOUS MEMORY USAGE
0480 ; ************************
0490 ;
0500 HBIT =  $060E      ;'HSCROL' COPY
0510 VBIT =  $060F      ;'VSCROL' COPY
0520 HTIME = $0610      ;'HSPEED' COPY
0530 VTIME = $0611      ;'VSPEED' COPY
0540 HPOINT = $0612     ;SHIFT COUNTER
0550 VPOINT = $0613     ;SHIFT COUNTER
0560 DLINES = $0614     ;LMS OPCODES CNT
0570 COUNT = $0615      ;'DLINES' COPY
0580 TEMP =  $0616      ;TEMP STORAGE
0590 TEMP1 = $0617      ;TEMP STORAGE
0600 XHOLD1 = $0618     ;TEMP STORAGE
0610 XHOLD2 = $0619     ;TEMP STORAGE
0620 OPCODE = $061A     ;DLIST OPCODE
0630 VBLANK = $061B     ;VBLANK VECTOR
0640 OFFSETS = $061D    ;OFFSETS TO LMS
0650 STOPALL = $06FF    ;HOLD EVERYTHING
1000      .INCLUDE #D:VAR.ASM
1010 ;
1020 ;*********************
1030 ;INITIALIZATION ROUTINE
1040 ;*********************
1050 ;
1060      *=  $4000
1070      LDA POKMSK    ;GET IRQ. INT.
1080      AND #$7F      ;NO BREAK KEY
1090      STA POKMSK    ;THE BREAK KEY
1100      STA IRQEN     ;NO LONGER WORKS
1110      CLD           ;CLEAR DECIMAL
1120      PLA           ;# OF ARGUEMENTS
1130      PLA           ;VBLANK HI/BYTE
1140      STA VBLANK+1
1150      PLA           ;VBLANK LO/BYTE
1160      STA VBLANK
1170      PLA           ;DLIST HI/BYTE
1180      STA LISTHI
1190      PLA           ;DLIST LO/BYTE
1200      STA LISTLO
1210      PLA           ;DISCARD
1220      PLA           ;GET 'LINES'
1230      STA ALINES
1240      PLA           ;DISCARD
1250      PLA           ;GET 'MINX'
1260      STA MINX
1270      STA HPOINT    ;RESET TO START
1280      PLA           ;DISCARD
1290      PLA           ;GET 'MAXX'
1300      STA MAXX
1310      PLA           ;DISCARD
1320      PLA           ;GET 'MINY'
1330      STA MINY
1340      STA VPOINT    ;RESET TO START
1350      PLA           ;DISCARD
1360      PLA           ;GET 'MAXY'
1370      STA MAXY
1380      PLA           ;DISCARD
1390      PLA           ;GET 'HMAX'
1400      STA HMAX
1410      PLA           ;DISCARD
1420      PLA           ;GET 'VMAX'
1430      STA VMAX
1440      PLA           ;DISCARD
1450      PLA           ;GET 'HSPEED'
1460      STA HSPEED
1470      PLA           ;DISCARD
1480      PLA           ;GET 'VSPEED'
1490      STA VSPEED
1500      PLA           ;DISCARD
1510      PLA           ;GET 'STICK'
1520      STA STICK
1530      PLA           ;DISCARD
1540      PLA           ;GET 'HINVERT'
1550      STA HINVERT
```

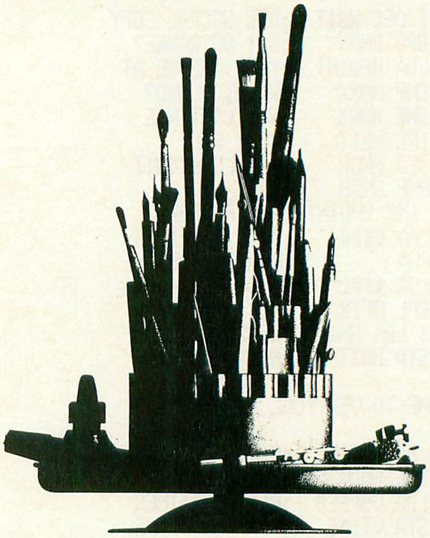
```
1560      PLA           ;DISCARD
1570      PLA           ;GET 'VINVERT'
1580      STA VINVERT
1590      PLA           ;PAGE USAGE HI
1600      STA PAGES+1   ;STORE IT
1610      PLA           ;PAGE USAGE LO
1620      STA PAGES     ;STORE IT
1630      LDA #$01      ;INITIALIZE RAM
1640      STA HTIME     ;COPY OF SPEED
1650      STA VTIME     ;TIMERS.
1660 ;
1670 ;********************************
1680 ;EXAMINE DLIST, NOTE LMS OPCODES
1690 ;********************************
1700 ;
1710      LDA ZPAGE     ;SAVE WHATEVER
1720      PHA           ;IS IN MEM. LOC.
1730      LDA ZPAGE+1   ;SAVE WHATEVER
1740      PHA           ;IS IN MEM. LOC.
1750      LDA LISTLO    ;GET DLIST/LO &
1760      STA ZPAGE     ;PUT IN WORKAREA
1770      LDA LISTHI    ;GET DLIST/HI &
1780      STA ZPAGE+1   ;PUT IN WORKAREA
1790      LDY #$00      ;SET UP COUNTER
1800      LDX #$00      ;SET UP COUNTER
1810      STX VBIT      ;SET UP SCROLL
1820      STX VBIT      ;BITS TO ZERO
1830 DLOOK LDA (ZPAGE),Y ;GET DISPLAY
1840      STA OPCODE    ;LIST OPCODE
1850      AND #$0F      ;IS IT A BLANK
1860      BEQ NEXT      ;LINE OPCODE?
1870      LDA OPCODE    ;IS IT AN LMS
1880      AND #$40      ;OPCODE?
1890      BEQ NEXT
1900      INY           ;IF SO, SKIP
1910      INY           ;LMS OPERANDS
1920      LDA OPCODE
1930      AND #$0F      ;IS IT A JVB
1940      CMP #$01      ;OPCODE?
1950      BEQ DONE      ;IF SO, STOP.
1960      LDA OPCODE    ;ARE SCROLL BITS
1970      AND #$30      ;OF OPCODE SET?
1980      BEQ NEXT
1990      TYA           ;THIS IS AN LMS
2000      SEC           ;OPCODE W/SCROLL
2010      SBC #$02      ;BITS SET. SAVE
2020      STA OFFSETS,X ;THE OFFSET.
2030      INX
2040 NEXT INY           ;MOVE TO NEXT
2050      BNE DLOOK     ;DLIST OPCODE.
2060 DONE STX DLINES    ;SAVE # OF LINES
2070      PLA           ;RESTORE MEM.
2080      STA ZPAGE+1   ;LOCATION.
2090      PLA           ;RESTORE MEM.
2100      STA ZPAGE     ;LOCATION.
2110      LDA #$07      ;TELL SYSTEM TO
2120      LDX VBLANK+1  ;SET UP DEF.
2130      LDY VBLANK    ;VERTICAL BLANK
2140      JSR SETVBV    ;ROUTINE.
2150      RTS           ;ALL DONE. BYE!
7000      .INCLUDE #D:VAR.ASM
7010 ;
7020 ;*************************
7030 ;TIME TO HORIZONTAL SCROLL?
7040 ;*************************
7050 ;
7060      *=  $5000
7070      LDA STOPALL   ;IS IT OKAY TO
7080      BEQ PLUNGE    ;EXECUTE?
7090      JMP XITVBV    ;NO! BYE!
7100 PLUNGE CLD         ;YES, CLEAR DEC.
7110      LDA LISTLO    ;TELL ANTIC
7120      STA SDLSTL    ;WHERE YOUR NEW
7130      LDA LISTHI    ;DISPLAY LIST IS
7140      STA SDLSTL+1
7150      DEC HTIME     ;DECREMENT TIMER
7160      BNE ENDIT     ;IF <> 0, STOP!
7170      LDA HSPEED    ;RESET TIMER
7180      STA HTIME
7190      BEQ ENDIT     ;IF = 0, STOP!
7200 ;
7210 ;  NOW READ CORRECT JOYSTICK
7220 ;
7230      LDX STICK     ;GET POSITION OF
7240      LDA STICK0,X  ;RIGHT JOYSTICK
7250      LDY HINVERT   ;SHOULD WE IN-
7260      BNE HOPP      ;VERT HOR. MOVE
7270 ;
7280 ;SCROLL DIRECTION ISN'T INVERTED
7290 ;
7300      AND #$0C      ;SCAN SELECT
7310      CMP #$08      ;JOYSTICK BITS
7320      BEQ HLEFT     ;MOVE LEFT
7330      CMP #$04
7340      BEQ HRIGHT    ;MOVE RIGHT
7350      BNE ENDIT
7360 ;
7370 ;SCROLL DIRECTION IS INVERTED
7380 ;
7390 HOPP AND #$0C      ;SCAN SELECT
7400      CMP #$04      ;JOYSTICK BITS
7410      BEQ HLEFT     ;MOVE LEFT
7420      CMP #$08
7430      BNE ENDH      ;MOVE RIGHT
7440 ;
7450 ;  SCROLL RIGHT
7460 ;
7470 HRIGHT INC HBIT    ;INCREMENT RAM
7480      LDA HBIT      ;'HSCROL' COPY
7490      CMP HMAX      ;IS IT ABOVE
7500      BEQ ENDH      ;VALID SCROLL
7510      BCC ENDH      ;LIMIT?
7520      LDA HPOINT    ;YES! ARE WE AT
7530      CMP MINX      ;LEFTMOST BOUND?
7540      BNE HOR5      ;NO, CONTINUE
7550      DEC HBIT      ;YES! HALT
7560      BEQ ENDH      ;SCROLL & QUIT
7570      BNE ENDH
7580 HOR5 DEC HPOINT    ;CONTINUE.
7590      LDA #$FF      ;PERFORM BYTE
7600      PHA           ;SHIFTING W/HFIX
7610      BMI HFIX      ;ROUTINE
7620 HOR55 LDA #$00     ;RESET 'HSCROL'
7630      STA HBIT      ;RAM COPY.
7640      BEQ ENDH      ;ALL DONE.
7650 ;
7660 ;INC/DEC LMS LO/BYTE OPERANDS
7670 ;
7680 HFIX STA TEMP      ;GET BYTES TO
7690      PLA           ;INC/DEC LO-BYTE
7700      STA TEMP1     ;OF LMS OPERANDS
7710      LDX ALINES    ;GET # OF LINES
7720      DEX           ;MINUS ONE
7730      LDY #$00      ;SET INDEX TO 0
7740 HFIX5 LDA (PAGES),Y ;GET OPERAND
7750      CLC
7760      ADC TEMP      ;INC/DEC IT
7770      STA (PAGES),Y ;PUT IT BACK
7780      INY           ;CHECK FOR WRAP
7790      LDA (PAGES),Y ;AROUND OF LO
7800      ADC TEMP1     ;BYTE INTO HI
7810      STA (PAGES),Y ;BYTE. FIXITUP
7820      INY           ;ADJUST POINTER
7830      DEX           ;DID WE DO ALL?
7840      BPL HFIX5     ;NO! CONTINUE
7850      LDA TEMP      ;YES! DECIDE
7860      BMI HOR55     ;WHO CALLED THIS
7870      BPL HOR66     ;ROUTINE. RETURN
7880 ;
7890 ;INTERMEDIATE BRANCH
7900 ;
7910 ENDIT BEQ ENDH     ;NO MATTER WHAT
7920      BNE ENDH      ;BRANCH TO END.
7930 ;
```

```
7940 ; SCROLL LEFT
7950 ;
7960 HLEFT DEC HBIT ;DEC HSCROL COPY
7970       BPL ENDH ;OUT OF RANGE?
7980       LDA HPOINT ;YES! ARE WE AT
7990       CMP MAXX ;RIGHT BOUND?
8000       BNE HOR6 ;NO! CONTINUE
8010       INC HBIT ;YES! HALT
8020       BEQ ENDH ;SCROLL & QUIT
8030       BNE ENDH
8040 HOR6 INC HPOINT ;NO! PERFORM
8050       LDA #$00 ;BYTE SHIFTING
8060       PHA ;W/HFIX ROUTINE
8070       LDA #$01
8080       BPL HFIX
8090 HOR66 LDA HMAX ;RESET 'HSCROL'
8100       STA HBIT ;RAM COPY
8110 ;
8120 ; TIME TO VERTICAL SCROLL?
8130 ;
8140 ENDH DEC VTIME ;DEC. TIMER
8150       BNE ENDV ;IF <> 0, QUIT
8160       LDA VSPEED ;RESTORE TIMER
8170       STA VTIME ;FOR NEXT TIME
8180       BEQ ENDV ;IF = 0, QUIT
8190 ;
8200 ; READ CORRECT JOYSTICK
8210 ;
8220       LDX STICK ;GET JOYSTICK #
8230       LDA STICK0,X ;GET READING
8240       LDY VINVERT ;IS VER. SCROLL
8250       BNE VOPP ;INVERTED?
8260 ;
8270 ;VERTICAL SCROLL ISN'T INVERTED
8280 ;
8290       AND #$03 ;SCAN SELECT
8300       CMP #$02 ;BITS
8310       BEQ VUP ;SCROLL UP
8320       CMP #$01
8330       BEQ VDOWN ;SCROLL DOWN
8340       BNE ENDV
8350 ;
8360 ;VERTICAL SCROLL IS INVERTED
8370 ;
8380 VOPP AND #$03 ;SCAN SELECT
8390       CMP #$01 ;BITS
8400       BEQ VUP ;SCROLL UP
8410       CMP #$02 ;SCROLL DOWN
8420       BNE ENDV ;DO NADA!
8430 ;
8440 ;SCROLL DOWN
8450 ;
8460 VDOWN DEC VBIT ;DEC VSCROL COPY
8470       BPL ENDV ;IN VALID RANGE?
8480       LDA VPOINT ;NO! ARE WE AT
8490       CMP MINY ;UPPER BOUND?
8500       BNE VER5 ;NO! CONTINUE
8510       INC VBIT ;YES! STOP
8520       BEQ ENDV ;SCROLL & QUIT
8530       BNE ENDV
8540 VER5 DEC VPOINT ;ADJUST VERTICAL
8550       LDA VMAX ;RESET RAM COPY
8560       STA VBIT ;OF 'VSCROL'
8570       BEQ ENDV ;ALL DONE!
8580       BNE ENDV
8590 ;
8600 ;SCROLL UP
8610 ;
8620 VUP INC VBIT ;INC RAM COPY
8630       LDA VBIT ;OF 'VSCROL'
8640       CMP VMAX ;IS IT IN VALID
8650       BEQ ENDV ;RANGE?
8660       BCC ENDV
8670       LDA VPOINT ;NO! ARE WE AT
8680       CMP MAXY ;LOWER BOUND?
8690       BNE VER6 ;NO! CONTINUE
8700       DEC VBIT ;YES! HALT
8710       BEQ ENDV ;SCROLL & QUIT
8720       BNE ENDV
8730 VER6 INC VPOINT ;ADJUST VERTICAL
8740       LDA #$00 ;RESET RAM COPY
8750       STA VBIT ;OF 'VSCROL'
8760 ENDV LDA HBIT ;INSTALL COPY
8770       STA HSCROL ;INTO 'HSCROL'
8780       LDA VBIT ;INSTALL COPY
8790       STA VSCROL ;INTO 'VSCROL'
8800 ;
8810 ;INSTALL PAGE DATA INTO DLIST
8820 ;
8830 FIX LDA DLINES ;GET # OF LMS
8840       BEQ RTS ;OPCODES
8850       STA COUNT ;HOLD ON TO IT
8860       LDA ZPAGE ;SAVE WHATEVER
8870       PHA ;IS IN MEM. LOC.
8880       LDA ZPAGE+1 ;SAVE WHATEVER
8890       PHA ;IS IN MEM. LOC.
8900       LDX #$00 ;SET INDEX TO 0
8910       STX XHOLD1
8920       LDA VPOINT ;VERTICAL POINT
8930       ASL A ;MULTIPLY BY 2
8940       STA XHOLD2 ;HOLD ON TO IT
8950       LDA SDLSTL ;GET LO/DLIST &
8960       STA ZPAGE ;PUT IN WORKAREA
8970       LDA SDLSTL+1 ;GET HI/DLIST &
8980       STA ZPAGE+1 ;PUT IN WORKAREA
8990 HSTUFF LDY XHOLD2 ;PAGE INDEX
9000       LDA (PAGES),Y ;PAGE DATA
9010       PHA ;SAVE IT
9020       LDX XHOLD1 ;OFFSET INDEX
9030       LDA OFFSETS,X ;OFFSET TO LMS
9040       TAY ;HOLD IT
9050       INY ;PLUS ONE
9060       PLA ;GET PAGE DATA
9070       STA (ZPAGE),Y ;PUT IN DLIST
9080       INC XHOLD2 ;INC PAGE INDEX
9090       LDY XHOLD2 ;GET IT AGAIN
9100       LDA (PAGES),Y ;PAGE DATA
9110       PHA ;SAVE IT
9120       LDA OFFSETS,X ;OFFSET TO LMS
9130       TAY ;HOLD IT
9140       INY ;PLUS ONE
9150       INY ;PLUS TWO
9160       PLA ;GET PAGE DATA
9170       STA (ZPAGE),Y ;PUT IN DLIST
9180       INC XHOLD1 ;INC OFFSET INDEX
9190       INC XHOLD2 ;INC PAGE INDEX
9200       DEC COUNT ;ARE WE DONE?
9210       BNE HSTUFF ;NO! KEEP GOING.
9220       PLA ;RESTORE MEMORY
9230       STA ZPAGE+1 ;LOCATION
9240       PLA ;RESTORE MEMORY
9250       STA ZPAGE ;LOCATION
9260 RTS JMP XITVBV ;ALL DONE. LATER!
```

•

PAINT
by The Capital Children's Museum
ATARI, Inc.
P.O. Box 61657
Sunnyvale, California 94086
48K Disk $44.95

by Arthur Leyenberger



Atari is clearly trying to get its act together. The recent introduction of **Atariwriter**, a quality word processor with the features most users want and need, was the first sign. Next came **Family Finances**, a combination of cash flow analysis and budget planner in an easy-to-use format. Now we have **Paint**.

Developed by the Capital Children's Museum in Washington, D.C. and originally published by Reston, **Paint** has been repackaged in Atari's familiar silver, black and white format, and is now an official Atari product.

Paint is difficult to describe because it will be many things to many people. To say it is a graphics program would be accurate but an understatement. To call it an electronic canvas would be closer to the point. **Paint** is an outlet for artistic expression that will let someone of any age go as far as they want to.

The program is simple to operate and lets the budding artist draw on the screen, using the joystick as a brush. Lines and shapes may easily be created. Areas of the screen can be colored in, patterns and textures put anywhere and colors mixed together. Also, brush speeds and widths can be varied, and mistakes erased.

Paint includes three separate programs. "Artshow" automatically displays in succession images that have already been created. This particular program is accessed by both keyboard *and* joystick. The keyboard is used only to initially select this option and then for pausing the pictures. The joystick is used for selecting the screens to be shown and the order in which they will appear. The procedure is very simple, and the anticipation of the "show" makes it fun.

The second program included is called "Simple Paint." This program provides four different brushes and four different colors. Its capabilities are not as extensive as the main program, but it is probably better suited for younger children. It is so easy to use, a youngster can be "up and doodling" in no time. This rapid involvement with the program is likely to get and maintain the child's interest.

The third and main program is "Super Paint." There are over 24 different commands, requested by either a 1-letter keyboard input or through a menu selection via the joystick. There are built-in functions for drawing lines, circles and rectangles. Pure colors or textured patterns may be used to fill in any enclosed portion of the screen. Two levels of "zoom" magnification are also available.

When "Super Paint" is accessed, the bottom portion of the screen contains 10 paint-pots. There are 9 textures and colors plus the one background color. The individual paint-pot colors may be chosen by either joystick manipulation or keyboard entry. Once the desired color is chosen, the joystick is used as a painter would use a brush. There are nine different brush types and nine brush widths, for a total of 81 combinations of brushstrokes. The speed of the brush may also be selected to allow the "video artist" to become accustomed to the eye-hand coordination required to perform delicate maneuvers, such as writing in script.

Paint comes with a 175-page manual that discusses everything from computer art to the basics of computing. The manual itself could be an ideal stand-alone text for art, philosophy or even computer science classes. In addition to the specifics on how to use the paint program, there are sections on: suggestions for creative projects, a brief history of art down through the ages, diverse examples of computer art created by professional artists using a variety of computers, computer imaging, and even a tutorial on how computers work. A glossary and bibliography are also included. The book is written in a lighthearted first person style, clearly presented and enjoyable.

As I said at the beginning of this review, it is difficult to describe **Paint** because of its flexibility and extensive features. Perhaps the best description is that **Paint** is **Paint**. In any case, the product is an excellent value. The best part of all is that, unlike the kind of paints I used when I was a child, there is no mess to clean up. ☐

Great Software • Great Service • ~~Great~~ *incredible* Prices

from Atari

Pole Position

ROM Cart. $36.88

Not since the release of their Pacman cartridge has there been such great anticipation for a game program from Atari. Expectations increased though months of pre-advertising without product delivery. Well, Pole Position is out now and it's well worth the wait. It's an exciting rendition of the coin-op game that was one of the two most popular (the other is Q*Bert) of 1983.

All Atari programs including Ms. Pacman, Dig Dug, Qix and Donkey Kong in stock and available at similar savings.

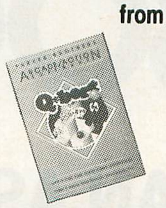
from Electronic Arts

Pinball Construction Set

48K Disk $27.88

Steve Wozniak of Apple Computer Corp. has called Bill Budge's Pinball Construction Set "the best program ever written for an 8-bit computer." You can build your own pinball machine from the parts provided — bumpers, flippers, targets, etc. — and, then, save and play them. You can also control the machine's logic, affecting the scoring, the sound, or even the amount of gravity inside the machine!

All Electronic Arts programs including Archon, M.U.L.E. and Murder on the Zinderneuf in stock and available at similar savings.



from Parker Brothers

Q*Bert

ROM Cart. $33.88

It's everybody's favorite arcade character of 1983. It's lovable Q*Bert. This is the official Atari cartridge version of the smash coin-op hit. Excellent graphics and game play.

All Parker Brothers programs including Popeye, Super Cobra, Frogger and Astro Chase in stock and available at similar savings.

from Broderbund

Lode Runner

48K Disk $22.88

Runner-up (to Miner 2049er) as the outstanding original arcade game of 1983 in our in-house poll. 150 thrill-packed levels plus a game generator feature which permits you to design and play your own levels. If you're starting to "burn out" on arcade games, this program will rekindle your enthusiasm.

All Broderbund Software including The Arcade Machine, Operation Whirlwind and David's Midnight Magic in stock and available at similar savings.

from Synapse

Blue Max

32K Tape or Disk $22.88

The premier Atari software house brings you still another fabulous program. As a member of the Royal Air Force, you will command a biplane fighter/bomber and will battle the Axis on the ground and in the skies. Excellent scrolling and 3-D graphics.

All Synapse programs including Rainbow Walker, Encounter and Pharoah's Curse in stock and available at similar savings.

from Microprose

Solo Flight

48K Tape or Disk $22.88

The first real flight simulator for the Atari. Features include: color, three-dimensional graphics; dual radio navigation instruments; full-feature, actual configuration instrument panel; multiple air navigation charts; multiple airports and runways; day, night, cross-wind and instrument flying scenarios; takeoffs and landings; emergency procedure practice; and (as they say) much, much more. An outstanding program that realistically simulates the thrill and challenge of flying.

All Microprose Software including NATO Commander, Hellcat Ace and Floyd of the Jungle in stock and available at similar savings.

NEW RELEASES

Save Big on a Dozen Recent Releases!

Bank St. Writer (Broderbund) D	$44.88
Eagles (SSI) D	$27.88
Enchanter (Infocom) D	$34.88
Infidel (Infidel) D	$34.88
LOGO (Atari) R	$74.88
Oil's Well (Sierra On-Line) R	$23.88
Pitstop (Epyx) R	$27.88
Planetfall (Infocom) D	$34.88
Spellunker (Micrographic) D	$27.88
Trains (Spinnaker) D	$27.88
Ultima III (Origin) D	$39.88
Zaxxon (Datasoft) T&D	$24.88

SUPERSPECIALS

Save 33% to 43% on these Classic Programs!

Baja Buggies (Gamestar) T&D	$19.88
Bandits (Sirius) D	$19.88
Castle Wolfenstein (Muse) D	$18.88
Choplifter (Broderbund) D	$22.88
Crush, Crumble & Chomp (Epyx) T&D	$18.88
Facemaker (Spinnaker) D	$22.88
Jumbo Jet Pilot (Thorn-EMI) R	$29.88
Labyrinth (Broderbund) T&D	$18.88
Mastertype (Lightning) D	$24.88
Serpentine (Broderbund) T&D	$19.88
Soccer (Thorn-EMI) R	$29.88
Visicalc (Visicorp) D	$139.88

Prices good thru Mar. 5, 1984

Call or write for FREE CATALOG (specify Atari)

Mail order prices do not apply at our retail location.

HOW TO ORDER:

Use mail or phone. For immediate shipment pay by Money Order, Certified Check, Cashiers Check, VISA, or Master Card (add 4% for credit card). Personal checks are accepted but require 1-3 weeks to clear. 10% deposit required for C.O.D.

SHIPPING & HANDLING:

$2.50 for orders under $50; free shipping over $50. Mass. residents add 5% sales tax. Prices are mail order only and are subject to change without notice.

SOFTWARE PLUS

(617) 369-1992

for orders and information
phone hours: 10-6 EST Mon.—Sat.

45 Walden St., Box AA, Concord, Mass. 01742

BOOT CAMP

AN ASSEMBLY LANGUAGE TUTORIAL COLUMN

by Tom Hudson

I hope all **Boot Camp** readers have been practicing their addition, subtraction and X-Y register manipulations, because we're moving on to bigger and better things. We'll be dabbling with comparisons, branching and indexing this month, giving you even more tools to work with in assembly language.

First things first.

Last month, I gave you a simple data manipulation problem:

PROBLEM: Write a program which starts with A=$03, X=$07 and Y=$14. Then write the code necessary to change these registers so that when the program ends, the registers are A=$07, X=14 and Y=$03.

As most readers know, there are hundreds of ways to solve any programming problem, and this one is no exception. The objective is not just to solve the problem, but to do it in the most efficient way possible. I'll show you two ways to solve the above problem, and discuss the pros and cons of each.

```
10         STA AHOLD
20         STX XHOLD
30         STY YHOLD
40         LDA XHOLD
50         LDX YHOLD
60         LDY AHOLD
70         BRK
80 AHOLD *=*+1
90 XHOLD *=*+1
0100 YHOLD *=*+1
0110          .END
```

Figure 1.

Figure 1 shows an easy-to-understand, straightforward solution to our problem. It stores each register in hold areas, then loads the registers from the appropriate hold area. Lines 10-60 perform the register exchange function, and Lines 80-100 set up the one byte storage areas.

This solution is very easy to understand by simply looking at it, and is a solution that most beginners would probably use. However, from a memory usage standpoint, this routine requires 22 bytes. We can do the same exchange in only 10 bytes with the routine in **Figure 2**.

```
10         STY HOLD
20         TAY
30         TXA
40         LDX HOLD
50         BRK
60 HOLD    *=*+1
70         .END
```

Figure 2.

As you can see, this code uses two of the transfer instructions, TAY and TXA, to eliminate two of the temporary storage areas used in **Figure 1**. Since the transfer instructions use only one byte versus the six bytes for a LDA and STA instruction, this version of the exchange code uses less than half the memory of **Figure 1**.

Although we gain memory savings by using the code in **Figure 2**, we do lose some readability. Let's say you use the routine in **Figure 1** in a program and don't look at the program for a year. If you need to make a change, it's easy to see what the routine does. The code in **Figure 2** may not be so easy to decipher. Since you never know when you'll have to make a change to a program, it's a very good idea to COMMENT your code heavily, in order to let yourself know what you were doing.

What if. . .?

The great thing about computers is that they can perform calculations very quickly. Without the ability to make decisions, though, a computer would be almost useless.

For this reason, the 6502 microprocessor in your Atari is equipped with 14 comparison instructions. These instructions are designed to test the values contained in the Accumulator, X and Y registers. Each of these instructions compares the desired register with the memory byte specified in the operand and sets the 6502 status flags accordingly.

The Accumulator comparison instructions are:

CMP #n	(IMMEDIATE)
CMP nn	(ABSOLUTE)
CMP n	(ZERO PAGE)
CMP (n,X)	(PRE-INDEXED INDIRECT)
CMP (n),Y	(POST-INDEXED INDIRECT)
CMP n,X	(ZERO PAGE INDEXED X)
CMP nn,X	(INDEXED X)
CMP nn,Y	(INDEXED Y)

The X register comparison instructions are:

CPX #n	(IMMEDIATE)
CPX nn	(ABSOLUTE)
CPX n	(ZERO PAGE)

The Y register comparison instructions are:

CPY #n	(IMMEDIATE)
CPY nn	(ABSOLUTE)
CPY n	(ZERO PAGE)

All comparison instructions affect only three status flags. These are the SIGN, ZERO and CARRY flags.

What happens in a comparison? Internally, the computer will subtract the operand byte from the register contents, set the status flags just like a subtract, but will NOT alter the register. Simple, right? Let's look at a few examples.

Assume the accumulator contains $45, and we execute the instruction:

CMP #$31

Inside the computer, the faithful 6502 would subtract $31 from $45 and obtain the following result:

```
$45  = 0 1 0 0 0 1 0 1
$31  = 0 0 1 1 0 0 0 1
       ---------------
       0 0 0 1 0 1 0 0 = $14
```

Since the result is not zero, the ZERO flag is set to 0. The SIGN flag is set to bit 7 of the result, which is 0. The CARRY flag is set to 1, since no borrow was required. The CARRY flag is always the inverse of the borrow status.

By looking at the result of this comparison, we can say that the accumulator is NOT EQUAL to $31, since the result of the compare was not zero. We can also say that the accumulator is GREATER THAN $31, since the CARRY flag is set.

Assume the X register contains $7F and we want to compare it with $7F. We would use the following instruction:

CPX #$7F

The subtract operation inside the 6502 would look like:

```
$7F  = 0 1 1 1 1 1 1 1
$7F  = 0 1 1 1 1 1 1 1
       ---------------
       0 0 0 0 0 0 0 0 =     $00
```

The result is zero, so the ZERO flag is set to 1. The 7 bit of the result is 0, so the SIGN flag is set to 0. No borrow was required, so the CARRY flag is set to 1.

After this comparison is complete, we can conclude that the register is EQUAL to $7F because the ZERO flag is set.

Assume the Y register contains $12 and we want to compare it to $4E. We would use the following instruction:

CPY #$4E

The subtract operation inside the 6502 would look like:

```
$12  = 0 0 0 1 0 0 1 0
$4E  = 0 1 0 0 1 1 1 0
       ---------------
       1 1 0 0 0 1 0 0 =     $C4
```

Before you get confused with the above binary operation, remember how subtraction works in base 10. If the number being subtracted (minuend) is larger than the subtrahend, a BORROW is necessary from the next higher digit. This case of the compare requires a borrow.

In this case, the ZERO flag will be set to zero, indicating a non-zero result. The SIGN flag will be set to the contents of bit 7 of the result, which is a 1. The CARRY flag will be set to 0, the inverse of the borrow status.

From these flags, we can conclude that the Y register is *less* than $4E because the CARRY flag is cleared (0).

That's all there is to using the compare instructions. They work the same way, regardless of the addressing mode.

Comparisons are just about worthless without the ability to do something based on the result of a comparison, so next we'll look at the 6502 branch-on-condition instructions.

Branches conveniently located.

So far, the only means of transferring program execution we've looked at has been the JMP (JUMP TO LOCATION) instruction. Now we'll look at the 8 branch-on-condition instructions used by the 6502. The 8 formats are:

BCS n	(BRANCH IF CARRY = 1)
BCC n	(BRANCH IF CARRY = 0)
BEQ n	(BRANCH IF ZERO = 1)
BNE n	(BRANCH IF ZERO = 0)
BMI n	(BRANCH IF SIGN = 1)
BPL n	(BRANCH IF SIGN = 0)
BVC n	(BRANCH IF OFLOW = 0)
BVS n	(BRANCH IF OFLOW = 1)

Observant readers may note that the operand of the branch instructions consists of only one byte. As you may recall, the JMP instruction was able to jump to any memory location because its operand consisted of two bytes. Branches are another story altogether.

With only one byte in their operands, branch instructions are only able to branch backward 128 bytes or forward 127 bytes. This is known as RELATIVE addressing. Fortunately, most assemblers will calculate the distance of a branch for you. However, if a branch distance is more than the branch limit, you'll have to restructure your branch by using a JMP or multiple branch instructions.

Let's look at a few typical branch applications. **Figure 3** shows the comparison/branch structure for the condition:

IF X = 7 THEN GOTO START

```
      CPX #7
      BEQ START
       .
       .
       .
START
```

Figure 3.

As you can see, the CPY instruction is followed by a branch instruction. In this case, if the X register is EQUAL TO 7, the program will go to the location labeled START.

For the condition:

IF A <> 52 THEN GOTO POINTA

we would use the code in **Figure 4**.

```
       CMP #52
       BNE POINTA
        .
        .
        .
POINTA
```

Figure 4.

Multiple conditions may require some extra effort, such as the condition:

IF Y <= 242 THEN GOTO MAIN

The code for this condition is shown in **Figure 5**.

```
     CPY #242
     BEQ MAIN
     BMI MAIN
      .
      .
      .
MAIN
```

Figure 5.

These multiple conditions are really quite easy, you just have to use the instructions provided.

The nice thing about branch instructions is that you don't have to use them after a compare instruction. You can place them anywhere in a program. For example, in addition or subtraction instructions, which set the status flags just like a compare, a zero result in an operation will set the proper branch flags. Look at the following code:

```
LDA BYTE1
SEC
SBC BYTE2
CMP #0
BEQ ZERO
```

The CMP #0 instruction is not necessary, since the SBC operation sets the flags for us! The optimized code would look like:

```
LDA BYTE1
SEC
SBC BYTE2
BEQ ZERO
```

Remember, branches can be done anywhere the status flags are altered, giving you incredible flexibility in program design.

"I wish I was indexing..."

Now we can start combining some of our new programming tools to do meaningful work. With the added function of branching, we can start using the X and Y registers as counters and indexes.

Indexing was discussed in the second installment of **Boot Camp** in **ANALOG #14**, so I won't repeat all the basics. The first example I'll show is the use of the X and Y registers as counters.

Let's say we want to execute a section of code ten times. Since the program uses the Accumulator and X register in the loop, we'll use the Y register as a counter to control the loop.

In order to use the X and Y registers as indexes, we have been given four instructions:

INX	(INCREMENT X BY 1)
INY	(INCREMENT Y BY 1)
DEX	(DECREMENT X BY 1)
DEY	(DECREMENT Y BY 1)

These four instructions simply add or subtract one from the X or Y registers, allowing you to use the registers as indexes easily. These registers affect the ZERO and SIGN flags.

Figure 6 shows the code necessary to perform a loop ten times.

```
      LDY #10
LOOP   .
       .
       .
      DEY
      BNE LOOP
```

Figure 6.

This is a very simple counter example. Note that, in this case, we have set up the Y register as a count-down counter, from 10 to 0. After the DEY instruction is executed, we BNE LOOP. If the Y register decremented to zero, the program will not take the branch, and the loop is finished. No CPY #0 instruc-

tion was needed, since the DEY instruction set the zero flag for us.

We could have used the Y register as a count-up counter, from 0 to 10, as in **Figure 7**.

```
        LDY #0
LOOP    .
        .
        .
        INY
        CPY #10
        BNE LOOP
```

Figure 7.

Note that in the count-up example an extra compare is needed (CPY #10) to see the Y register has reached ten yet. If it has not, the program will take the BNE LOOP branch to continue looping.

Using the X and Y registers for *indexing* is similar to using them for counters. The main difference is that the register is used inside the loop to point to varying places in memory. **Figure 8** shows an example of indexing that will copy the six bytes of TABLE1 into TABLE 2.

```
10         LDX #5
20 COPY    LDA TABLE1,X
30         STA TABLE2,X
40         DEX
50         BPL COPY
60         BRK
70 TABLE1  .BYTE 10,20,30,40,50,60
80 TABLE2  *=*+6
90         .END
```

Figure 8.

The program in **Figure 8** begins with the X register set at 5. Remember, when referencing individual elements in a table, the indexes for the elements range from zero to one less than the number elements. In this case, the element numbers range from 0-5. As the loop (labeled COPY) executes, each byte of TABLE1 will be moved to TABLE2. This looping will continue until the X register is decremented past zero, where it will equal 255 due to wraparound. At this point, the SIGN flag will be 1, indicating a negative number. When this happens, the BPL COPY instruction will be ignored and the looping will end. Try assembling this routine into memory and tracing its execution.

What if we want to copy TABLE1 into TABLE2 in REVERSE ORDER? This is a nifty little problem that will help you understand X-Y indexing more thoroughly. Try writing the code necessary, using as many memory locations as necessary. Next issue, I'll show a way to do this with only three changes to **Figure 8**.

No more time.

I had wanted to cover multi-byte math this issue, but due to space limitations I'll have to delay this until next issue. Until then, play around with comparisons and branching, and try to find a solution to the above problem. □

FAST REPEAT KEY

16K Cassette or Disk

by Sammie J. McCaa, Jr.

While working on a rather large machine-language program, I became impatient with the long delay in the Atari keyboard's key repeat function. I found myself pressing the CTRL/arrow keys to move ahead, instead of just holding down the space bar and waiting for the repeat. So I put away what I was doing and pulled out my operating system manuals to figure out a way to get around the problem.

I discovered that ths OS uses a variable called SRTIMR at location 555 decimal ($22B hex) to determine how long to wait before starting the repeat function. It works liks this: Every time you press a key, the keyboard interrupt routine stores a value of 30 (for a 30-jiffy or 1/2 second delay) into SRTIMR. This location is then decremented every 1/60th of a second. If the key is still being pressed when SRTIMR reaches zero, the OS will repeat the key until you let it go. All you have to do to change the repeat delay is to control the value that is stored into SRTIMR when a key is first pressed.

I began by writing a program that copied the OS keyboard interrupt routine from ROM into RAM, changed the value stored into SRTIMR and, of course, altered the interrupt vector to point to the new routine. It worked, but I wasn't too sure if it would work on other versions of the OS (I have an old Atari with the original "A" ROMs). I went down to a local department store to try my routine on a 1200XL. The program didn't stand a chance. Talk about incompatibility! Not only is the 1200 interrupt handler in a different ROM location (this wasn't really a problem, since I could check the "reserved" interrupt vector to find out where it was), but it was three, yes, *three* times as long as the handler in the 400/800 OS!

After a few more attempts, including one that tried to determine which OS you had by checking the size of the interrupt handler, I finally got it right. The current version continuously checks SRTIMR with a small custom vertical blank interrupt routine. Whenever SRTIMR gets bigger than I want it to be, I just change it. Simple, right?

I got tired of re-initializing the routine every time I hit SYSTEM RESET, so out came the manuals again. I discovered how to trick Atari into believing there has been a cassette boot, thereby enabling me to steal the system long enough to keep my fast repeat-key routine active. The routine is small enough (only 37 bytes) to safely fit at the bottom of the 6502 hardware stack on page 1. This keeps page 6 free for other machine-language routines.

I've included two versions of the program. **Listing 1** is the BASIC loader, while **Listing 2** is the assembly-language source code. Both versions are set up for use with a disk drive. To use the routine with a cassette, change the third byte in Line 120 of the BASIC listing from 3 to 2, or Line 140 of the assembly listing from LDA #3 to LDA #2.

To execute the BASIC loader, just type it in, CHECK it with D:CHECK1 or C:CHECK and RUN it. The assembly version requires a G100 from the Editor/Assembler cartridge's DEBUG mode. I selected a new key delay value of 10 (1/6 of a second). By changing the PAUSE variable in the assembly version (Line 270) and re-assembling, you can experiment with different time delays. □

BASIC Listing.

```
10 REM * REPEAT KEY BASIC LOADER
20 REM * BY SAMMIE J. MCCAA, JR.
30 REM
40 FOR RAM=256 TO 292
50 READ BYTE:POKE RAM,BYTE:NEXT RAM
60 POKE 255,104:REM * PLA FOR BASIC
70 X=USR(255):REM * INIT ROUTINE
80 ? "FAST REPEAT INSTALLED"
90 END
100 DATA 160,22,162,1,169,6,32,92
110 DATA 228,169,0,133,2,169,1,133
120 DATA 3,169,3,133,9,96,173,43
130 DATA 2,201,11,144,2,169,10,141
140 DATA 43,2,76,95,228
```

•

CHECKSUM DATA
(See pp. 20-24.)

```
10 DATA 228,60,255,869,896,390,347,31,
259,683,917,523,923,626,7007
```

•

Assembly Listing.

```
0100 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
0110 ;                              ;
0120 ;         REPEAT KEY           ;
0130 ;                              ;
0140 ; BY SAMMIE J. MCCAA, JR.      ;
0150 ;                              ;
0160 ;  ANALOG COMPUTING #16        ;
0170 ;                              ;
0180 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
0190 ;
0200 ; ** EQUATES **
0210 ;
0220 BOOT =  $09
0230 CASINI = $02
0240 SRTIMR = $022B
0250 SETVBV = $E45C
0260 SYSVBV = $E45F
0270 PAUSE = 10
0280 ORG =   $0100
0290 ;
0300   *=  ORG
0310 ;
0320 INIT
0330     LDY #WAIT&255 ; CHANGE IMMEDIATE
0340     LDX #WAIT/256 ; VERTICAL BLANK
0350     LDA #6      ; VECTOR
0360     JSR SETVBV
0370     LDA #ORG&255 ; TRICK COMPUTER
0380     STA CASINI  ; INTO THINKING
0390     LDA #ORG/256 ; THERE HAS BEEN
0400     STA CASINI+1 ; A CASSETTE BOOT
0410     LDA #3
0420     STA BOOT
0430     RTS
0440 ;
0450 WAIT
0460     LDA SRTIMR  ; GET TIMER VALUE
0470     CMP #PAUSE+1 ; IS IT > PAUSE?
0480     BCC STORE
0490     LDA #PAUSE  ; MAKE IT = PAUSE
0500 STORE
0510     STA SRTIMR  ; SAVE NEW RESULT
0520     JMP SYSVBV  ; AND RETURN
0530 ;
0540     .END
```

•

OIL'S WELL
by Thomas Mitchell
SIERRA ON-LINE
Sierra On-Line Building
Coarsegold, California 93614
16K Cartridge $39.95

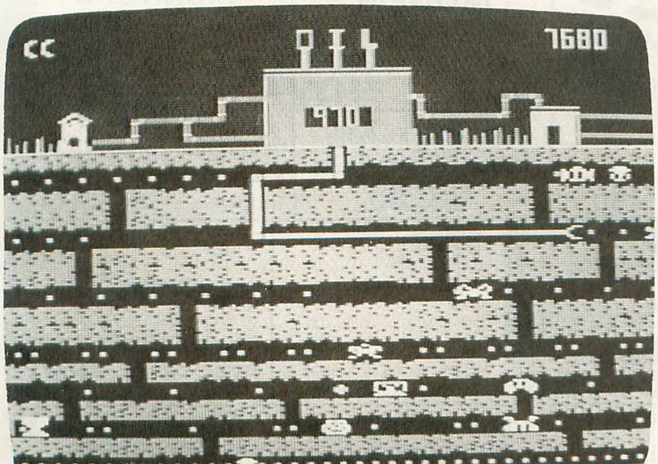
DROL
by Aik Beng
BRODERBUND SOFTWARE
17 Paul Drive
San Rafael, California 94903
32K Disk $39.95

ORC ATTACK
by Dean M. Lock
THORN-EMI VIDEO
1370 Avenue of the Americas
New York, NY 10019
16K Cartridge $39.95

by Patrick J. Kelley

With the mid-winter doldrums upon us, many will be looking for interesting new games to divert and entertain until spring's kindly thaw. This review will cover three possible candidates, catering to a wide range of tastes.

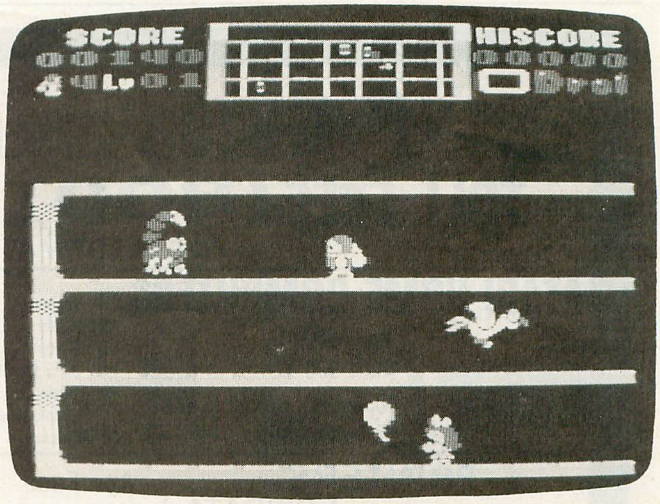
The first is **Oil's Well** from Sierra On-Line. Initially, the cartridge may appear to be just another variation on the "gobble the dots" theme, but this arcade-style game offers action that is both truly different and challenging. Your joystick controls a **PacMan**-like drilling implement on a retractable tether. You must maneuver your drill bit through a network of subterranean oil pipes, consuming units of energy and avoiding foreign objects flushed into the network to hinder your progress. The goal is to successfully navigate the network without colliding with the obstacles, and reach the lower segments of the screen.



Oil's Well.

Simple as it may sound, **Oil's Well** is *very* difficult, offering fast play, good graphics, and that certain something that makes you want to keep playing long after many other games have lulled you into boredom.

If the scenario of **Oil's Well** isn't unusual enough for you, then Broderbund's **Drol** is the ticket. This game puts you into a topsy-turvy fantasy world of screeching witch doctors, renegade vacuum cleaners, bounding monsters and plants with a taste for murder. You are the heroic soul sent into this bizarre never-never land on a desperate rescue mission. Equipped with an anti-gravity back pack and a gun that blasts monsters to oblivion with "reality pellets," you must comb a scrolling maze in search of a hostage family held prisoner by an evil witch doctor. Battling monsters and dodging weapons hurled at you by the witch doctor in your quest, your job is to scoop up the captives as they guilelessly wander around.



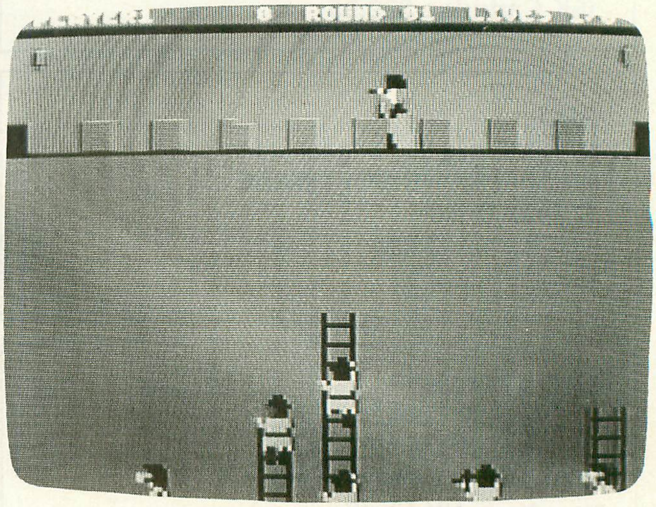
Drol.

Besides the mother, daughter and little boy that you must find, you are also in search of the family pets (a lizard and a crocodile), also imprisoned by the heartless blackguard. This aspect of the game I find especially charming; the pets float through the maze smiling happily and zooming about on their own private jet packs or "beanie belts." (This is reminiscent of a scene in William Peter Blatty's film *The Ninth Configuration*, which also featured characters ambiguously floating by, propelled in a similar manner.) **Drol** is the most lighthearted of these three games, and by far the best thought-out. It also features refreshing sound effects and some of the best pseudo-3D graphics I've ever seen.

Last, but by no means least, is **Orc Attack** by Thorn /EMI. **Attack** is easily the most violent and gratuitously satisfying shoot-'em-up on the market today (although "drop-'em-down" might be a more accurate label). In **Orc Attack**, you are the sole defender of a castle besieged by hellish Orcs, whose sole purpose is to scale the walls and invade. You prevent this by hurtling chunks of mortar down into the rampaging Orcs, splitting their scaling ladders and soft Orc skulls. A direct hit on an enemy Orc rewards you with a sickening *thunk* and a splash of blood.

As the invasion progresses, so do your methods of defense. Periodically during each attack wave, you can make use of a cauldron of boiling oil, which horribly (and probably painfully) eradicates the arrogant Orcs. Bonus levels force you to deal with a necromancer who calls forth spirits of the dead, an alchemist who hurls lightning bolts, or a horde of fast moving and deadly spiders.

Orc Attack is essentially an endurance contest in which you annihilate as many Orcs as possible before they scale the wall and send your severed head over the side. The graphics are adequate and, well... graphic. It's great if you want to get rid of the frustrations of a hard day, or if you are a budding sadist at heart. So take some of your remaining holiday money and ask your local computer store to show you **Oil's Well**, **Drol** and **Orc Attack**. Whatever your choice, you won't be sorry. □



Orc Attack.

FASTER THAN A SPEEDING BASIC . . .

Able to leap tall assemblers in a single byte. More powerful than simple binary code, it's . . .

BASM

- Runs up to 231 times faster than Atari BASIC. Up to 217 times faster than Commodore 64 BASIC.
- Programs in BASIC. Uses assembly language datatypes and addressing modes.
- Facilitates learning of assembly language programming.
- Reduces writing time for graphics programs and games by more than 50%.
- Provides free updates with small handling charge.

More importantly, BASM stands for truth, justice and a better way to program!

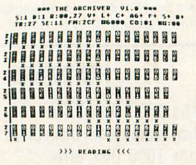
Only $99.95

Dealer inquiries encouraged.

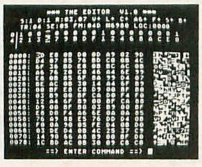
COMPUTER ALLIANCE
21115 Devonshire St., Suite 132
Chatsworth, California 91311
(213) 368-4089

Add $2.00 for shipping.
Add $1.50 for C.O.D.
California residents add 6½% sales tax.

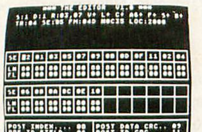
810 POWER!



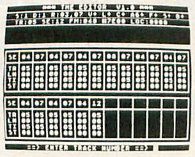
ARCHIVER



EDITOR



FORMATTER



MAPPER

The **CHIP**™ with Archiver Software allows:
Automatic Back-ups of **any** Disk Program. Easy enough for a beginner to use.
Also includes Sector Editor, Disassembler, Mapper, and Custom Formatter.

All this for **only $99.95** plus shipping

Happy owners?.........
Update your Happy 810 Enhancement now with **ARCHIVER/EDITOR** Software.
Supports multiple drives and makes your Happy drive compatible with the **CHIP!**

Just **$39.95** plus shipping

Visa & MasterCard or C.O.D. call now (815) 229-2999 for immediate shipment or send check or M.O. to:

MIGHTY BYTE COMPUTER
828 Green Meadow Ave., Dept. AG
Rockford, Illinois 61107

For Technical Support and Drive Service
Call 7pm-11pm CST

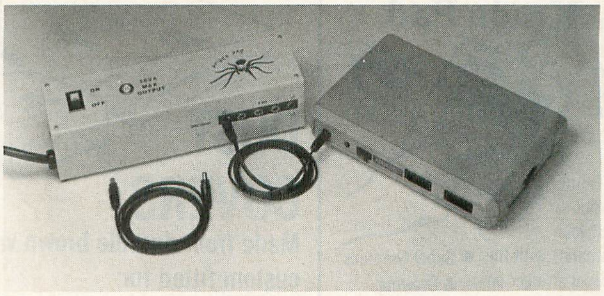
Thank You for making our 1983 an Outstanding Success!

The CHIP and ARCHIVER Software are products of Spartan Software of MN, Inc.

SPIDER PAC

pat. pend.

DOES IT ALL!



One **SPIDER PAC** takes the place of 5 transformers, a surge protector, a five stage noise suppressor, and up to 2 power strips! On one side there are 5 filtered low voltage outlets that supply power to all of your Atari® 9VAC components (400, 800, 1200XL, disc drives, 850 interface, 1010 recorder, etc.). On the other side you have three filtered 115 VAC outlets for monitors, printers or whatever.

IMPROVED, now handles several disk drives at a time and with the new XL-DAPTOR you can power up the just introduced 600-1400XL series.

- Provides a centralized power source for all of your components.
- Compatible with the Rana® and Trak® disc drives.
- Five 4 ft. long detachable low voltage cords are included.
- Avoids potential fire and electrical shock hazards associated with overloading extension cord devices with too many transformers.
- Protects your hardware/software investments with spike suppression and noise filtration (Five stage EMI & RFI).
- 15 amp. circuit breaker.
- Makes your Atari® system more versatile and portable.
- Great for schools — daisy chain up to 4 **SPIDER PACS** in a row (you can run up to 29 components with this configuration).
- Illuminated on/off switch.
- All low voltage outlets are individually isolated.
- A retail price very competitive with power conditioners that don't provide the added feature of low voltage outlets.

SPIDER PAC-MODEL # SP-1B	$134.95
SPIDER PAC JR.-MODEL # SP-JR B	$107.95
(Same as SP-1 except without filters and circuit breaker)	
XL-DAPTOR-MODEL # XL-DB	$24.95
(needed only for the new 600XL, 800XL, and 1400XL series of computers).	

Check with your local dealer or order factory direct.

Use your MC or Visa by calling our order lines, 24 hours a day. AZ residents must pay 6% Sales Tax. Free Postage & Handling in Cont. U.S., add $3.95 for outside Cont. U.S.

Continental U.S.A., P.R., & V.I. **1-800-824-7888**
Alaska & Hawaii **1-800-824-7919**

Ask for Operator No. 93

or send a certified check or money order to Compu-Tech, Dept. R, P.O. Box 5058, Glendale, AZ 85306

COMPU-TECH Corporation Headquarters
12033 N. 60th Ave.
Glendale, AZ 85304 (602) 979-2100

Atari is a registered trademark of Warner Communications, Inc.
Trak is a registered trademark of microcomputer corporation.
Rana is a registered trademark of Rana Systems.

COMPUTER PALACE

Over 1000 Items For Your Atari!

ATARI®

We Know ATARI Computers... Call Us For The Latest Info: (503)683-5361

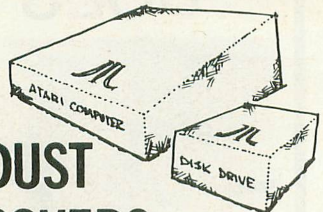
the Koala Pad

From Koala Tech.

The future is here... Input your commands with the touch of a finger on your Koala Pad.

- Super Graphics & Drawing Program Included with Pad.

Retail $100.00
$87.00 Plus Freight



DUST COVERS

Made from durable brown vinyl & custom fitted for:

- ATARI 400, 800, 600XL, 1200XL Computers
- ATARI 410 Recorder, 810 Disk Drive
- EPSON MX-80 or FX-80 Printers

$8.95 EACH Please specify which model

FREE Catalog

With any order, or send $1 (refundable with purchase). Send us your name & address for FREE QUARTERLY FLYERS with SPECIALS!

Over 1000 items for your Atari...

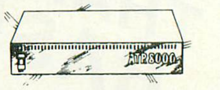
- Mosaic
- Percom
- Broderbund
- APX
- Roklan
- Datasoft
- Synapse
- A I
- On-Line
- Atari
- Hayes
- Epson
- Visicalc
- More!

Use Your Credit Cards!

We honor VISA, American Express, & MasterCard.

24 & 48 Hour Shipping Is Available... CALL!

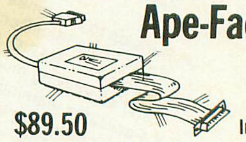
ATR-8000 EXPANSION UNIT



- Run CP/M on your Atari
- Parallel printer port
- Printer spooler
- Serial port
- DD controller

	Plus Freight
16K model	299.00
64K model (for CP/M)	499.00
2-DDDS Drives (for 64K model)	695.00

Ape-Face



$89.50 Parallel Printer Interface

PINBALL (Much more than a game!) CONSTRUCTION SET

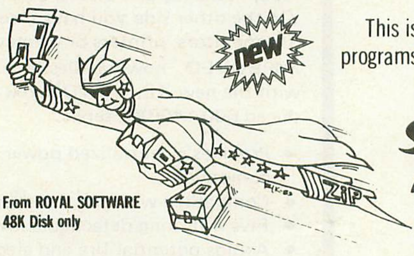
48K Disk

New from Electronic Arts **$35.10**

- Design the pinball game of your dreams, or play one of five games on the disk! Share your games with friends!

ALSO FROM ELECTRONIC ARTS...

M.U.L.E (D)	35.10
Hard Hat Mack (D)	31.50
Archon (D)	35.10
Axis Assassin (D)	31.50
Murder on the Zinderneuf (D)	35.10
Worms (D)	31.50



From ROYAL SOFTWARE
48K Disk only

This is one of the **Most Versatile** data-base programs available for business or home use!

Super Mailer PLUS

Much More Than A Mailing List!

Maintain Your Book Library
Organize Your Record Collection
Keep Track Of Your Recipies
Are You A Stamp Collector?
Let Super Mailer Plus keep track of all these and more!

Dealer Inquiries Invited

Our Price $49.95

You can accomplish this **and more** because of the **"User Re-defineable fields"!** The program has many other features: Multiple Search–Special Code Field–Single/or/Double Density Operation–Create Sub-files–Auto-delete–Print Labels 1/2/or 3-up–Print File Copy–Merge/Search/Sort–Built-in Official Post Office State Abbreviations–More...

$54.00 The Monkey Wrench II

A cartridge for the right-hand Slot of your Atari 800!

This is a BASIC & Machine-language **Programmers Aid** for the 800. It plugs into the right slot, and works with Atari BASIC to add 16 new direct-mode commands(Incl. Auto line numbering, Delete lines, Renumber BASIC, Hex-Dec conv., Find string, Move lines, Exchange string, Disk directory from BASIC, Mem. test, More...). Also includes a monitor with 15 commands to interact with the 6502. This has been proven to be a very helpful tool when writing your own programs!

ALL-TIME FAVORITES

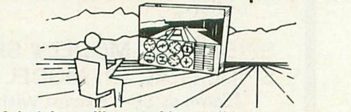
Star Raiders(C)	38.20
Eastern Front (D,T)	29.95
Pacman (C)	39.50
Temple of Apshi (D,T)	35.10
Zork I, II, or III (D) each	35.10
Centipede (C)	39.50
Missle Command (C)	29.70
Shamus (D,T)	31.50
Ghost Hunter (D)	34.95
Ghost Hunter (T)	29.95

TOP-TEN PROGRAMS

Pole Position (C)	44.90
Donkey Kong (C)	44.90
Blue Max (D,T)	31.50
Baseball (D,T)	28.90
Pinball Construction Set (48K, Disk)	35.10
Football (D,T)	28.90
Monkey Wrench II (not a game) (C)	54.00
Ultima II (D)	52.50
Archon (D)	35.10
Dig Dug (C)	44.90

FLIGHT SIMULATOR II

From Sublogic — 48K Disk **$44.90**



Colorful, scrolling graphics—Full instrumentation
The program includes a World War I "Air-Ace" battle game, too!

ULTIMA III/Exodus

$49.50 From Origin Systems

Here is the latest in the famous series of fantasy/role-playing adventure games.

HARDWARE

Astra 1620 Disk Drive	529.00
Micromainframe Disk Drive	499.00
Rana 1000 Disk Drive	349.00
Trak A2-D2 Disk Drive	398.00
Super-5 Printer	329.00
Panasonic Monitor (Grn/Scrn, w/sound)	199.00

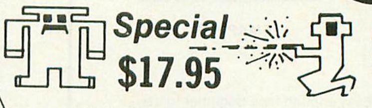
SOFTWARE

Archon (D,T)	35.10
Joust (C)	44.90
Robotron (C)	39.50
Microsoft Basic II (D)	79.10
Diskey Utility (D)	44.90
Quasimodo (D,T)	31.50
Murder on the Zinderneuf (D)	35.10
Blue Max (D,T)	31.50
Odessa Chess (D)	59.50
Pharoah's Pyramid (D)	31.50
Rally Speedway (C)	44.90
Ultima II (D)	52.50
Ultima III/Exodus (D)	49.50
Tax Advantage (D) (Not a game)	59.95

SUPER SPECIALS—UP TO 60% OFF!

Meteor Storm
24K Disk Only — From Royal Software

Special $9.95



***Special* $17.95**

K-Razy Cartridge ***Shootout***

From CBS

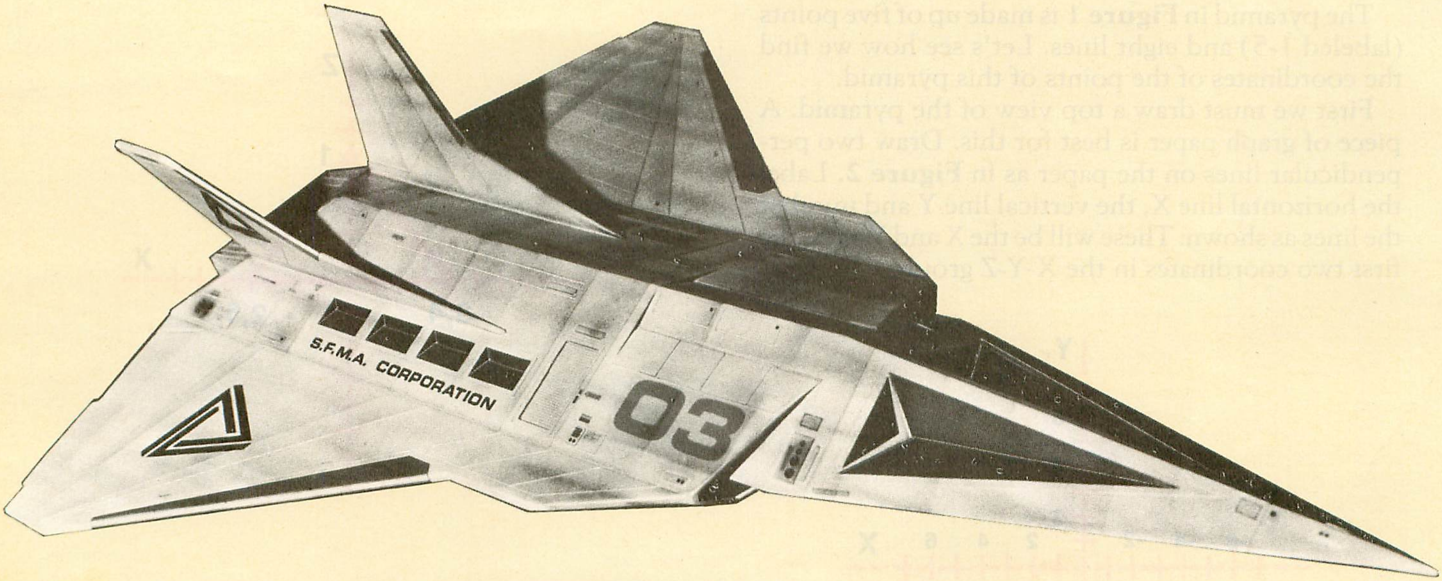
COMPUTER PALACE

OPEN Monday - Saturday, 9 to 6
2160 W. 11th Avenue Eugene, Oregon 97402

USE YOU CREDIT CARD & CALL TOLL-FREE 1-800-452-8013
You Never Pay Extra To Use You Credit Card At Computer Palace!

Our Toll-Free Number Is For ORDERS ONLY, PLEASE! Call 503-683-5361 *for info.*

Shipping Note: UPS & Parcel Post, Add $2.00—UPS Blue Label (Air), Add $3.90—Hardware and Paper Supplies are Extra—Call for exact cost on your order. **24 & 48 hour shipping available—Call 503-683-5361 for information.**



SOLID STATES

A 3D OBJECT PLOTTING SYSTEM

24K Cassette or 32K Disk

by Tom Hudson

Most people who are familiar with the Atari personal computers will agree that their graphics are superior to any other computer in their price range. By using the Atari's high-resolution GRAPHICS 8 mode, one can produce very respectable images on the video screen. Combine this ability with Atari 1020 printer/plotter hard-copy, and you've got an excellent, low-cost graphics machine.

This article presents a program which lets you show three-dimensional objects on the screen or print them out on a 1020 plotter. You can view the objects from any angle, with true perspective. If you like, object data can be stored on cassette or disk for future use.

The basics of 3-D.

In order to describe any object in our three-dimensional world, we must give at least three coordinates. These coordinates are usually labeled X (length), Y (width) and Z (height). This 3-D program is no exception. When you want to show an object, you must break it down into a number of points. Each point has its own set of X, Y and Z coordinates, which tells the computer where the point is located in space.

In addition to the locations of the points, we must tell the computer how these points are connected to form the sides of the object. The final result will be a "wire-frame" graphic representation of the object (so called because the object looks like it is constructed out of thin wires strung between the individual points of the object). **Figure 1** shows the wire-frame representation of a pyramid.

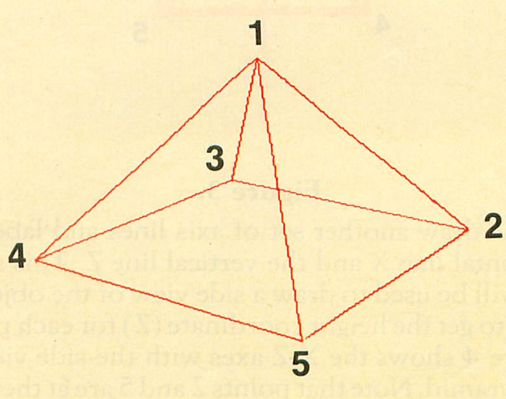


Figure 1.

The pyramid in **Figure 1** is made up of five points (labeled 1-5) and eight lines. Let's see how we find the coordinates of the points of this pyramid.

First we must draw a top view of the pyramid. A piece of graph paper is best for this. Draw two perpendicular lines on the paper as in **Figure 2**. Label the horizontal line X, the vertical line Y and number the lines as shown. These will be the X and Y axes, the first two coordinates in the X-Y-Z group.

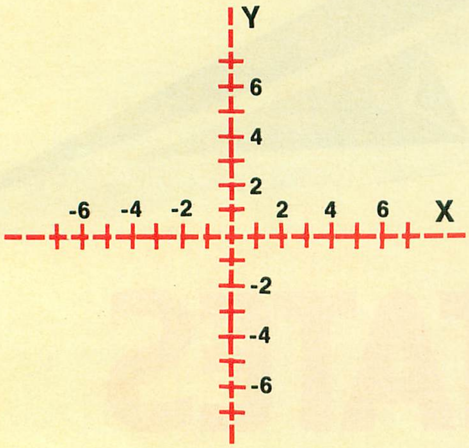


Figure 2.

Next, draw the top view of the 3-D object and number each point where two or more lines intersect. **Figure 3** shows the top view of our pyramid with its five points labeled.

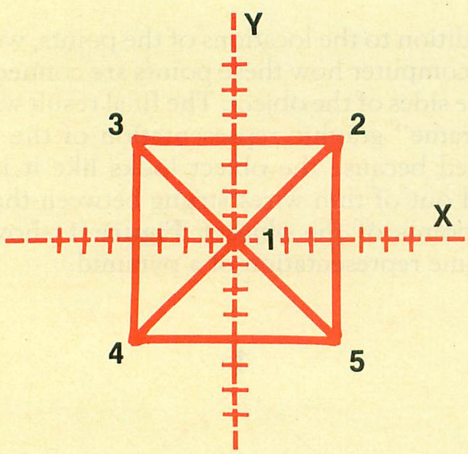


Figure 3.

Now draw another set of axis lines and label the horizontal line X and the vertical line Z. This set of axes will be used to draw a side view of the object in order to get the height coordinate (Z) for each point. **Figure 4** shows the X-Z axes with the side view of our pyramid. Note that points 2 and 5 are at the same location on the side view, as are points 3 and 4. This is because they have the same X and Z locations.

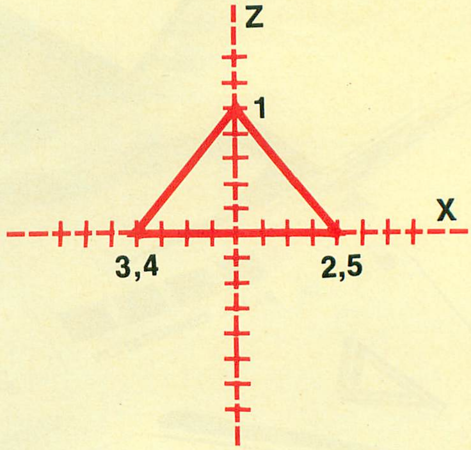


Figure 4.

Now that we've drawn our pyramid, we can write down the coordinates for each point. First let's find the coordinates for point 1.

To find the X coordinate, we can look at either **Figure 3** or **Figure 4**. In either case, the X coordinate for point 1 is 0.

To find the Y coordinate, we look at **Figure 3**. The Y coordinate for point 1 is also 0.

To find the Z coordinate, look at **Figure 4**. The Z coordinate for point 1 is 5.

Repeat this procedure for the remaining four points in the pyramid and you should have a list that looks like **Figure 5**.

POINT #	X	Y	Z
1	0	0	5
2	4	4	0
3	-4	4	0
4	-4	-4	0
5	4	-4	0

Figure 5.

After defining all the points, we're ready to tell the computer how to draw the pyramid. As stated earlier, this pyramid is made up of eight lines. In order to draw each line, the computer must know which two points make up the endpoints of the line. **Figure 6** shows the endpoints of the eight lines.

LINE #	FROM POINT	TO POINT
1	1	2
2	1	3
3	1	4
4	1	5
5	2	3
6	3	4
7	4	5
8	5	2

Figure 6.

With this information in hand, you're ready to enter it into the 3-D image program in **Listing 1**. Type in the program and check your typing with C:CHECK or D:CHECK2. When you're sure the program is correct, RUN it.

What a view.

When the 3-D object plot program is executed, the computer will ask you several questions.

1. **DMA OFF?** If you answer Y, the computer will turn off the screen while performing the complex 3-D math. This will speed up the program considerably. If you don't care about speed, type N.
2. **OUTPUT TO PLOTTER?** If you own an Atari 1020 plotter and would like hard-copy of your 3-D objects, ready your plotter and answer Y to this question. Otherwise, answer N.
3. **FILE OR KEYBOARD INPUT?** Type an F or a K depending on your choice.
4. If you want a file input (F), the computer will ask for the name of a 3-D object file. If you're using the cassette, position your tape to the beginning of the 3-D file, press PLAY, type C: and press RETURN. For disk, respond D:FILENAME.EXT and press RETURN. The computer will load the desired 3-D object data and continue at step 6.
5. If you want to define a new 3-D object (K), you will be asked how many points there are in the object. Our pyramid has 5 points. You will then be asked for the X,Y and Z coordinates of each point. For example, to enter the coordinates of point 2 of the pyramid, you would type 4,4,0 and press RETURN. Next you will be asked how many lines there are in the object. Type this number and press RETURN. After this you must enter the endpoints of each line. Enter these point numbers as in **Figure 6**. Finally, you'll be asked if you'd like to save the object in a file. If you type Y, the computer will ask for a filename. For cassette, position your tape, press RECORD and PLAY, type C: and press RETURN. When the computer beeps, press RETURN again and the file will be saved. For disk, type D:FILENAME.EXT and press RETURN.
6. You're now ready to look at the object. You will be asked:

ENTER OBSERVER LOCATION X,Y,Z OR E FOR EDIT

If you'd like to examine and/or change the object data, type E, press RETURN and go to step 8. Otherwise, type in the X, Y and Z coordinates of the point in space where your imaginary "eye" is located and press RETURN. A good observer location for our pyramid is 5,7,3. Next you will be asked:

ENTER COORDINATES LOOKED AT X,Y,Z

Now you should enter the X, Y, and Z coordinates of the place where you want to look. If, for example, you want to look at the center of our pyramid, you should type 0,0,2.5 and press RETURN. Finally, you'll be asked:

ENTER ZOOM FACTOR

The normal zoom factor is 1. To enlarge the object image, type a larger number; to make the object smaller, enter a smaller number.

7. At this point, the object will be drawn on your screen and on the plotter, if selected. When drawing is complete, a short tone will sound. Press START, SELECT, OPTION or the joystick #1 trigger to start a new plot at step 6.
8. The EDIT section of the program has three options: PRINT the object data, EDIT POINTS or EDIT LINES. These options are self-prompting and similar to the original data entry procedure above. After editing, you may save the edited object data to a file, if desired, as in step 5.

More complex objects.

You can design and plot extremely complex objects with this image processor. The procedure is the same as the simple pyramid used in the above illustrations; there are just more points and lines involved.

Figure 7 is the data for a futuristic 3-D space shuttle. Try entering this data into your computer and viewing it.

POINTS:44

POINT 1:	-5.5	-5.5	0
POINT 2:	-8	-1	0
POINT 3:	-8	3	0
POINT 4:	-16	14	0
POINT 5:	-16	17	0
POINT 6:	-5.5	21	0
POINT 7:	0	16	2.5
POINT 8:	0	0	2.5
POINT 9:	-2.5	0	1.25
POINT 10:	-2.5	-6.5	1.25
POINT 11:	0	-7.5	2.5
POINT 12:	-1.25	-2	1.5
POINT 13:	-1.25	-7.5	1.5
POINT 14:	-2.5	18.5	1.25
POINT 15:	-4.5	22	6.5
POINT 16:	-4.5	18	6.5
POINT 17:	-3.2	13	3
POINT 18:	-2.5	5.5	1.25
POINT 19:	-4	-6	0
POINT 20:	-4	-7.5	0
POINT 21:	0	-26	0
POINT 22:	4	-7.5	0
POINT 23:	4	-6	0
POINT 24:	1.25	-7.5	1.5
POINT 25:	1.25	-2	1.5
POINT 26:	2.5	0	1.25
POINT 27:	2.5	-6.5	1.25
POINT 28:	5.5	-5.5	0
POINT 29:	5.5	21	0
POINT 30:	2.5	18.5	1.25
POINT 31:	4.5	22	6.5
POINT 32:	4.5	18	6.5
POINT 33:	3.2	13	3
POINT 34:	2.5	5.5	1.25
POINT 35:	16	17	0
POINT 36:	16	14	0
POINT 37:	8	3	0
POINT 38:	8	-1	0
POINT 39:	3.5	-9.5	0.25
POINT 40:	0.5	-9.5	2
POINT 41:	0.5	-19	0.6
POINT 42:	-0.5	-9.5	2
POINT 43:	-3.5	-9.5	0.25
POINT 44:	-0.5	-19	0.6

LINES:58

LINE 1:	7 TO 6
LINE 2:	6 TO 29
LINE 3:	29 TO 7
LINE 4:	29 TO 35
LINE 5:	35 TO 36
LINE 6:	36 TO 37
LINE 7:	37 TO 38
LINE 8:	38 TO 28
LINE 9:	28 TO 29
LINE 10:	6 TO 5
LINE 11:	5 TO 4
LINE 12:	4 TO 3
LINE 13:	3 TO 2
LINE 14:	2 TO 1
LINE 15:	1 TO 6
LINE 16:	7 TO 8
LINE 17:	8 TO 9
LINE 18:	9 TO 10
LINE 19:	10 TO 1
LINE 20:	8 TO 26
LINE 21:	26 TO 27
LINE 22:	27 TO 28
LINE 23:	30 TO 31
LINE 24:	31 TO 32
LINE 25:	32 TO 33
LINE 26:	33 TO 34
LINE 27:	34 TO 30
LINE 28:	14 TO 15
LINE 29:	15 TO 16
LINE 30:	16 TO 17
LINE 31:	17 TO 18
LINE 32:	18 TO 14
LINE 33:	10 TO 13
LINE 34:	13 TO 12
LINE 35:	12 TO 9
LINE 36:	12 TO 8
LINE 37:	13 TO 11
LINE 38:	11 TO 8
LINE 39:	8 TO 25
LINE 40:	25 TO 26
LINE 41:	11 TO 24
LINE 42:	24 TO 25
LINE 43:	27 TO 24
LINE 44:	24 TO 23
LINE 45:	23 TO 22
LINE 46:	22 TO 24
LINE 47:	13 TO 19
LINE 48:	19 TO 20
LINE 49:	20 TO 13
LINE 50:	11 TO 21
LINE 51:	21 TO 22
LINE 52:	20 TO 21
LINE 53:	42 TO 43
LINE 54:	43 TO 44
LINE 55:	44 TO 42
LINE 56:	39 TO 40
LINE 57:	40 TO 41
LINE 58:	41 TO 39

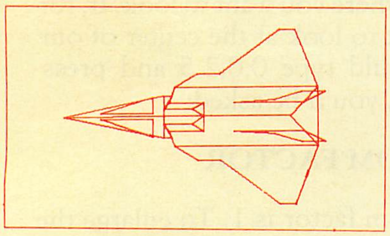
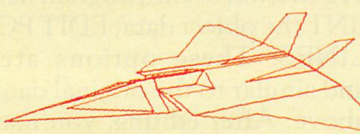


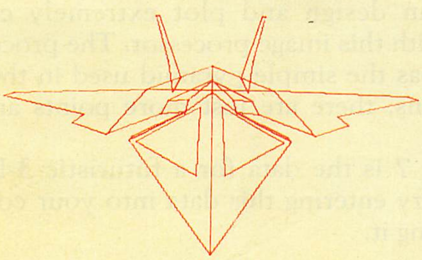
Figure 7.

Some suggested views for this shuttle are:

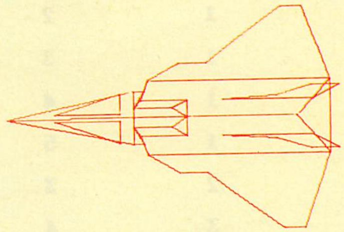
OBSERVER LOC.	LOOKED AT	ZOOM
60,-60,40	0,0,0	1



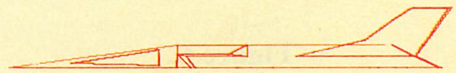
OBSERVER LOC.	LOOKED AT	ZOOM
0,-45,6	0,0,0	1



OBSERVER LOC.	LOOKED AT	ZOOM
1,0,500	0,0,0	4.5



OBSERVER LOC.	LOOKED AT	ZOOM
500,0,0	0,0,0	6



Who gets the credit?

To be honest, I can't take all the credit for this program. I got it from a Compucolor II computer bulletin board several years ago; the author was not credited. I modified the program to work on the Atari computers, added the editing, screen clipping, plotter output and file handling.

Although the code has been heavily modified, the author may recognize the general structure, and is encouraged to let me know his or her name. I'll be glad to give credit where it is due.

Attention, artists.

I'd be interested in seeing some of the 3-D objects created by our readers with this program. Just send in a printout of your object's data along with your name, and you may see your work displayed in a future issue! □

```
100 GRAPHICS 0
110 ? "****************************"
120 ? "*   3-D IMAGE PLOT SYSTEM   *"
130 ? "*      BY TOM HUDSON        *"
140 ? "*   ANALOG COMPUTING # 16   *"
150 ? "****************************"
160 DIM A$(5),F$(20),DMA$(1),O$(1),EG$
(2),IN$(1):EG$=CHR$(27):EG$(2)=CHR$(7)
170 XL=0:XR=319:YT=0:YB=191:OFF=40000
180 ? :? "DMA OFF";:INPUT DMA$:IF DMA$
<>"Y" AND DMA$<>"N" THEN 180
190 ? :? "OUTPUT TO PLOTTER";:INPUT O$
:IF O$<>"Y" AND O$<>"N" THEN 190
200 IF O$="N" THEN 220
210 TRAP 190:CLOSE #3:OPEN #3,8,0,"P":
? #3;EG$;"*H*I*M0,-300*I":PC=0
220 ? :? "FILE OR KEYBOARD INPUT";:INP
UT A$:IF A$="F" THEN 1100
230 IF A$<>"K" THEN 220
240 ? :? "HOW MANY POINTS ARE THERE";:
INPUT PS
250 DIM X(PS),Y(PS),Z(PS),P(PS,2),VIS(
PS)
260 ? "ENTER X,Y,Z COORDINATES FOR EAC
H POINT"
270 FOR I=1 TO PS:? "POINT ";I;:INPUT
Q1,Q2,Q3:X(I)=Q1:Y(I)=Q2:Z(I)=Q3:NEXT
I
280 ? :? "HOW MANY LINES ARE THERE";:I
NPUT LS:DIM LN(LS,1)
```

```
290 ? :? "NOW ENTER POINT INFORMATION"
300 ? "FOR EACH LINE."
310 FOR I=1 TO LS:? :? "LINE ";I:? "FR
OM POINT";:INPUT Q1:LN(I,0)=Q1:? "  TO
 POINT";:INPUT Q1:LN(I,1)=Q1:NEXT I
320 ? :? "DO YOU WANT TO SAVE THIS OBJ
ECT";:INPUT A$:IF A$="Y" THEN 1250
330 IF A$<>"N" THEN 320
340 REM ***************************
350 REM *    TIME FOR NEW PLOT     *
360 REM ***************************
370 ? :? "ENTER OBSERVER LOCATION X,Y,
Z"
380 ? "OR E FOR EDIT":TRAP 1340
390 ZOOM=1
400 INPUT OX,OY,OZ
410 ? :? "ENTER COORDINATES LOOKED AT
X,Y,Z"
420 TRAP 410:INPUT VX,VY,VZ
430 ? :? "ENTER ZOOM FACTOR":TRAP 430:
INPUT ZOOM:TRAP OFF
440 X(0)=VX:Y(0)=VY:Z(0)=VZ
450 D0=1:IF DMA$="Y" THEN POKE 559,0
460 REM ***************************
470 REM *  CALCULATE PERSPECTIVE   *
480 REM ***************************
490 DX=VX-OX:DY=VY-OY:DZ=VZ-OZ
500 U1=SQR(DX*DX+DY*DY+DZ*DZ):IF U1=0
THEN U1=1E-06
510 CX=DX/U1:CY=DY/U1:CZ=DZ/U1
520 S3=SQR(1-CZ*CZ):S2=SQR(1-CY*CY)
530 QX=OX+D0*CX:QY=OY+D0*CY:QZ=OZ+D0*C
Z
540 FOR I=0 TO PS:XW=X(I):YW=Y(I):ZW=Z
(I):GOSUB 610:NEXT I
550 FOR I=0 TO PS:IF VIS(I)=0 THEN 570
560 XW=X(I):YW=Y(I):ZW=Z(I):GOSUB 610:
GOSUB 670
570 NEXT I:GOTO 740
580 REM ***************************
590 REM *  IS THE POINT VISIBLE?   *
600 REM ***************************
610 VIS(I)=1:VCX=XW-OX:VCY=YW-OY:VCZ=Z
W-OZ
620 IF DX*VCX+DY*VCY+DZ*VCZ>0 THEN RET
URN
630 VIS(I)=0:RETURN
640 REM ***************************
650 REM *  NOW CALC PLOT COORDS    *
660 REM ***************************
670 K=D0/(VCX*CX+VCY*CY+VCZ*CZ)
680 AX=OX+K*VCX:AY=OY+K*VCY:AZ=OZ+K*VC
Z
690 IF S3=0 THEN 720
700 P(I,1)=((AX-QX)*CY-(AY-QY)*CX)/S3
710 P(I,2)=(AZ-QZ)/S3:RETURN
720 P(I,1)=((QX-AX)*CZ+(AZ-QZ)*CX)/S2
730 P(I,2)=(AY-QY)/S2:RETURN
740 REM ***************************
750 REM *     SCALE THE IMAGE      *
760 REM ***************************
770 T=450*ZOOM:FOR I=0 TO PS
780 P(I,1)=P(I,1)*T
790 P(I,2)=P(I,2)*T
800 NEXT I
810 XAD=160-P(0,1):YAD=96-P(0,2):FOR I
=1 TO PS:P(I,1)=P(I,1)+XAD:P(I,2)=P(I,
2)+YAD:NEXT I
820 REM ***************************
830 REM *   NOW DRAW THE IMAGE!    *
840 REM ***************************
850 GRAPHICS 24:SETCOLOR 2,0,0:COLOR 1
:TRAP OFF
860 IF O$="Y" THEN ? #3;"M0,0*D480,0*D
480,288*D0,288*D0,0"
870 FOR I=1 TO LS:TV=VIS(LN(I,0))+VIS(
LN(I,1)):IF TV=0 THEN 1010
880 IF TV=2 THEN 980
890 QT=0:ISAVE=I:IF VIS(LN(I,0))=0 THE
N I1=LN(I,0):I2=LN(I,1):I=LN(I,0):GOTO
 910
900 I1=LN(I,1):I2=LN(I,0):I=LN(I,1)
910 XT1=X(I1):YT1=Y(I1):ZT1=Z(I1):XT2=
X(I2):YT2=Y(I2):ZT2=Z(I2):FV=0:FH=0
920 XW=(XT1+XT2)/2:YW=(YT1+YT2)/2:ZW=(
ZT1+ZT2)/2:GOSUB 610
930 IF VIS(I)>0 THEN XT2=XW:YT2=YW:ZT2
=ZW:GOTO 950
940 XT1=XW:YT1=YW:ZT1=ZW
950 QT=QT+1:IF QT<15 THEN 920
960 XW=XT2:YW=YT2:ZW=ZT2:GOSUB 610
970 GOSUB 670:P(I,1)=P(I,1)*T+XAD:P(I,
2)=P(I,2)*T+YAD:VIS(I)=0:I=ISAVE
980 X1=P(LN(I,0),1):Y1=191-P(LN(I,0),2
):X2=P(LN(I,1),1):Y2=191-P(LN(I,1),2):
GOSUB 1550
990 IF O$="N" OR POK=0 THEN 1010
1000 ? #3;"M";X1*1.5;",";(191-Y1)*1.5;
"*D";XW*1.5;",";(191-YW)*1.5:PC=PC+1
1010 NEXT I
1020 IF PC>0 THEN ? #3;"H*M0,-300*I":P
C=0
1030 FOR X=15 TO 0 STEP -1:SOUND 0,128
,10,X:NEXT X
1040 IF PEEK(53279)=7 AND STRIG(0)=1 T
HEN 1040
1050 GRAPHICS 0:? "LAST PARAMETERS:"
1060 ? :? "OBSERVER: ";OX;",";OY;",";O
Z:? "VIEWPOINT:";VX;",";VY;",";VZ:? "Z
OOM:";ZOOM:GOTO 340
1070 REM ***************************
1080 REM *   LOAD 3-D IMAGE FILE    *
1090 REM ***************************
1100 CLOSE #1:? "ENTER FILENAME TO LOA
D";:INPUT F$:TRAP 1200:OPEN #1,4,0,F$:
TRAP 1180
1110 INPUT #1,PS:DIM X(PS),Y(PS),Z(PS)
,P(PS,2),VIS(PS)
1120 FOR X=1 TO PS:INPUT #1,Q1:X(X)=Q1
:NEXT X
1130 FOR X=1 TO PS:INPUT #1,Q1:Y(X)=Q1
:NEXT X
1140 FOR X=1 TO PS:INPUT #1,Q1:Z(X)=Q1
:NEXT X
1150 INPUT #1,LS:DIM LN(LS,1)
1160 FOR X=1 TO LS:INPUT #1,Q1:LN(X,0)
=Q1:INPUT #1,Q1:LN(X,1)=Q1:NEXT X
1170 CLOSE #1:TRAP OFF:GOTO 340
1180 ? :? "FILE FORMAT ERROR!":GOTO 1
210
1190 ? :? "I/O ERROR - ";PEEK(195):GO
TO 1210
1200 ? :? "CAN'T OPEN FILE!"
1210 ? "PRESS RETURN":INPUT IN$:RUN
1220 REM ***************************
1230 REM *    SAVE 3-D IMAGE FILE   *
1240 REM ***************************
1250 CLOSE #1:? "ENTER FILENAME TO SAV
E";:INPUT F$:TRAP 1210:OPEN #1,8,0,F$:
TRAP 1190
1260 ? #1;PS
1270 FOR X=1 TO PS:? #1;X(X):NEXT X
1280 FOR X=1 TO PS:? #1;Y(X):NEXT X
1290 FOR X=1 TO PS:? #1;Z(X):NEXT X
1300 ? #1;LS:FOR X=1 TO LS:? #1;LN(X,0
):? #1;LN(X,1):NEXT X:GOTO 1170
1310 REM ***************************
1320 REM * EDIT THE 3-D IMAGE DATA *
1330 REM ***************************
1340 TRAP OFF:? :? "PRINT, EDIT OR QUI
T";:INPUT A$:IF A$="E" THEN 1410
1350 IF A$="Q" THEN 340
1360 IF A$<>"P" THEN 1340
1370 TRAP 1340:LPRINT "POINTS:";PS:LPR
INT
1380 FOR X=1 TO PS:LPRINT "POINT ";X;"
: ";X(X),Y(X),Z(X):NEXT X:LPRINT
1390 LPRINT "LINES:";LS:LPRINT
1400 FOR X=1 TO LS:LPRINT "LINE ";X;":
 ";LN(X,0);" TO ";LN(X,1):NEXT X:LPRIN
T :GOTO 1340
1410 TRAP OFF:? :? "EDIT POINT OR LINE
 OR EXIT";:INPUT A$:IF A$="L" THEN 148
0
1420 IF A$="E" THEN 320
1430 IF A$<>"P" THEN 1410
1440 ? :? "ENTER POINT# OR RETURN";:TR
AP 1410:INPUT PT:IF PT>PS OR PT<0 THEN
 1440
1450 ? :? "X=";X(PT),"Y=";Y(PT),"Z=";Z
(PT)
1460 ? :? "ENTER NEW X,Y,Z OR RETURN":
TRAP 1410
```

```
1470 INPUT Q1,Q2,Q3:X(PT)=Q1:Y(PT)=Q2:
Z(PT)=Q3:GOTO 1410
1480 ? :? "ENTER LINE# OR RETURN";:TRA
P 1410:INPUT LN:IF LN>LS OR LN<0 THEN
1480
1490 ? :? "FROM POINT:";LN(LN,0):? "
TO POINT:";LN(LN,1)
1500 ? :? "ENTER NEW LINE POINTS OR RE
TURN":TRAP 1410
1510 ? "FROM POINT:";:INPUT Q1:IF Q1>P
S THEN 1510
1520 LN(LN,0)=Q1
1530 ? "  TO POINT:";:INPUT Q1:IF Q1>P
S THEN 1530
1540 LN(LN,1)=Q1:GOTO 1410
1550 REM *****************************
1560 REM *  GRAPHICS CLIP ROUTINE    *
1570 REM *****************************
1580 L1=0:L2=0:R1=0:R2=0:T1=0:T2=0:B1=
0:B2=0:POK=0
1590 IF X1<XL THEN L1=1:GOTO 1610
1600 IF X1>XR THEN R1=1
1610 IF Y1>YB THEN B1=1:GOTO 1630
1620 IF Y1<YT THEN T1=1
1630 IF X2<XL THEN L2=1:GOTO 1650
1640 IF X2>XR THEN R2=1
1650 IF Y2>YB THEN B2=1:GOTO 1670
1660 IF Y2<YT THEN T2=1
1670 IF L1+L2=2 OR R1+R2=2 OR T1+T2=2
OR B1+B2=2 THEN RETURN
1680 X3=X1:Y3=Y1:X4=X2:Y4=Y2:GOSUB 173
0
1690 L1=L2:R1=R2:T1=T2:B1=B2
1700 X1=XW:Y1=YW:X3=X2:Y3=Y2:X4=X1:Y4=
Y1:GOSUB 1730
1710 IF X1<XL OR X1>XR OR Y1<YT OR Y1>
YB OR XW<XL OR XW>XR OR YW<YT OR YW>YB
 THEN RETURN
1720 PLOT X1,Y1:DRAWTO XW,YW:POK=1:RET
URN
1730 IF L1+T1+B1+R1=0 THEN XW=X3:YW=Y3
:RETURN
1740 IF L1 THEN XW=XL:YW=Y3+(Y4-Y3)*(X
L-X3)/(X4-X3):X3=XW:Y3=YW:IF Y3>=YT AN
D Y3<=YB THEN RETURN
1750 IF R1 THEN XW=XR:YW=Y3+(Y4-Y3)*(X
R-X3)/(X4-X3):X3=XW:Y3=YW:IF Y3>=YT AN
D Y3<=YB THEN RETURN
1760 IF B1 THEN YW=YB:XW=X3+(X4-X3)*(Y
B-Y3)/(Y4-Y3):X3=XW:Y3=YW:IF X3>=XR AN
D X3<=XL THEN RETURN
1770 IF T1 THEN YW=YT:XW=X3+(X4-X3)*(Y
T-Y3)/(Y4-Y3):X3=XW:Y3=YW
1780 RETURN
```

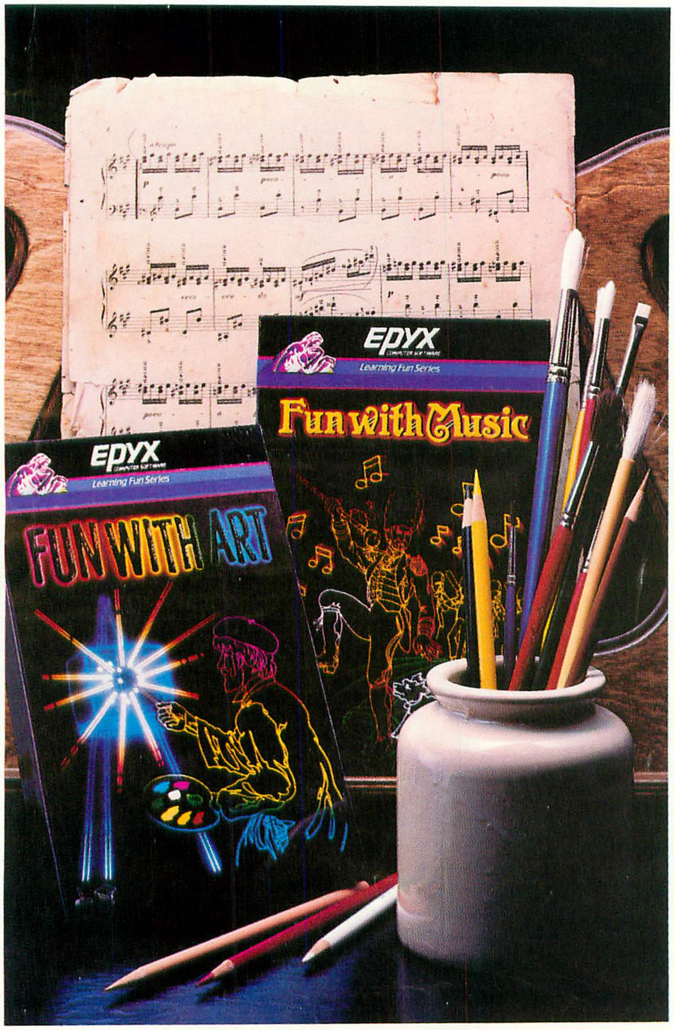
●

CHECKSUM DATA

(See pp. 20-24.)

```
100 DATA 884,171,93,680,49,183,281,21,
19,761,786,810,646,727,345,6456
250 DATA 7,361,741,555,54,975,741,76,7
36,358,140,364,580,623,747,7058
400 DATA 684,758,904,746,634,492,366,6
49,372,463,470,216,489,389,83,7715
550 DATA 868,182,868,374,466,352,181,8
44,719,364,318,370,435,571,501,7413
700 DATA 848,937,854,936,366,889,372,9
35,581,587,742,548,362,193,368,9518
850 DATA 158,523,258,560,857,273,581,7
27,815,679,764,109,918,17,713,7952
1000 DATA 873,485,698,241,637,893,973,
550,249,552,639,802,953,957,961,10463
1150 DATA 713,132,170,273,397,595,386,
551,302,553,702,526,491,494,497,6782
1300 DATA 385,553,479,555,355,614,183,
673,58,513,334,553,586,183,960,6984
1450 DATA 81,478,859,659,899,355,534,1
72,417,104,563,746,565,49,77,6558
1600 DATA 825,36,833,89,832,55,840,626
,913,636,520,958,894,177,945,9179
1750 DATA 964,922,270,808,2964
```

INDEX TO ADVERTISERS

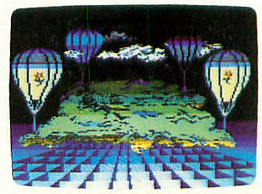
GIVE YOUR KIDS A LESSON THEY'LL NEVER FORGET.

When kids have fun and learn at the same time, they're more likely to remember more of what they've learned. What's more, when they associate the two together—learning becomes an enjoyable activity. So they'll do more of it.

That's the basic principle behind the Learning Fun Series from EPYX.

FUN WITH MUSIC. PARENTS TAKE NOTE.

Fun with Music™ is designed to appeal to both you and your children. It comes with a songbook and has two modes of play. In the first mode, you can enter songs from the songbook—or compose your own. Then play the songs back, adding or removing notes and changing tempo or key. Your computer is like a musical instrument with memory, and you see every note of it displayed on your screen.

The second mode lets you play your song in a fun-filled action game. You control a drum major trying to touch the notes before a small but pesky poodle catches up to him and slows down the parade.



Either way, Fun with Music gives you and your whole family the perfect mix of learning and play.

FUN WITH ART. JUST PICTURE IT.

Creating art on the video screen is one of the newest forms of "high-tech" play for kids. And adults, for that matter. Plus there's nothing to clean up afterwards.

Fun with Art™ uses the computer and joystick to transform your TV screen into an artist's canvas with astonishing results. 128 colors, numerous brush strokes, all kinds of geometric shapes, and special fill-in and magnifying options are some of the 24 different modes and features available to create works of art never before possible.

Fun with Art brings out the artist in you, no matter what your age!

MORE LEARNING FUN ON THE WAY.

These two are the first of an extensive series of Learning Fun games we have planned. Look for these, as well as other EPYX titles, wherever computer software is sold.

EPYX™ COMPUTER SOFTWARE

LEARNING FUN SERIES.


John Anderson's
RALLY SPEEDWAY
SLIP INTO SOMETHING TURBO-CHARGED
like the road-scorching speedster in RALLY SPEEDWAY — the game that turns your ATARI into a full-blown, four-wheeled demon! Take on our demanding course, or use the unique Construct-A-Track option to design your own. Push yourself to the limit for a better lap time, or get a friend in on the action for a one-on-one duel to the finish line — it's your choice!
CALL TOLL FREE 1-800-327-7172
MAKE TRACKS TO YOUR DEALER TODAY!
ATARI 400/800/1200 CARTRIDGE
053-0171 . . . . $49.95
If your dealer doesn't have Rally Speedway in stock, phone 1-800-327-7172 with your order, or write us at Adventure International, P.O. Box 3435, Longwood, FL 32750 for your FREE catalog.
DEALER ORDERS WELCOME
155
COPYRIGHT 1983
Adventure INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC.