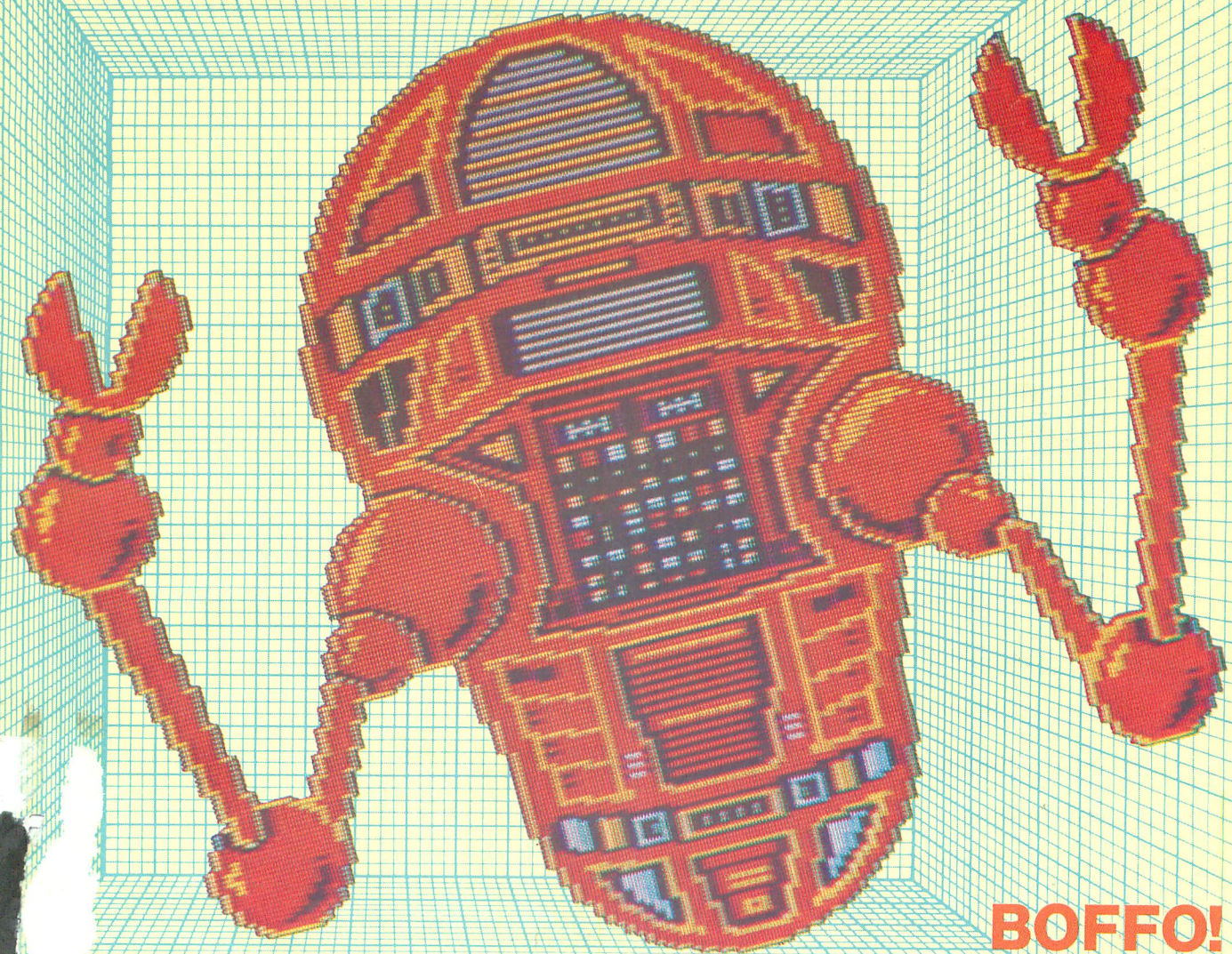
NO. 24 NOVEMBER 1984 $3.00

THE MAGAZINE FOR ATARI® COMPUTER OWNERS

ANALOG COMPUTING



BOFFO!
Unicheck
Bopotron!
Race in Space
AtariCon Report
Circuit Database
Cassette Compressor

OSS WRITES ONLY PRECISION SOFTWARE... OUR CUSTOMERS WRITE OUR BEST ADS!

BASIC XL

"BASIC XL is a fast and powerful extension of Atari BASIC, totally compatible with virtually all software. Its many features make programming easy, especially games that require player/missile graphics. For people writing business software or translating existing programs from other computers, the new string arrays and other string-handling features make the task manageable. BASIC XL is a truly professional language that should become standard in all future Atari computers. Overall Rating—A." *The Addison-Wesley Book of Atari Software 1984*

BASIC XL SuperCartridge & Manual ***(Requires 16K Memory)*** **$99.00**

ACTION!

"For those who have found BASIC to be too slow or assembler too difficult, ACTION! is the logical alternative. ACTION! programs can increase speed from 50 to 200 times that of BASIC." *Jerry White, Antic, February 1984*

ACTION! SuperCartridge & Manual ***(Requires 16K Memory)*** **$99.00**

MAC/65

"For the serious machine language programmer or anyone interested in programming in 6502 machine language, this package is a must. A lot of the good professional software on the market, games or otherwise, was written using this brute. Coding machine language with anything else is like trying to swim upstream in quicksand." *ACE Of West Hartford, May 1984*

MAC/65 SuperCartridge & Manual ***(Requires 16K Memory)*** **$99.00**

New Tools To Help You Write Better Programs

THE BASIC XL TOOLKIT

Packed with many useful subroutines to expand the capabilities of your BASIC XL programming and games that demonstrate BASIC XL's speed and versatility.

THE BASIC XL TOOLKIT Disk & Manual **$39.95**

THE ACTION! TOOLKIT

A useful library of ACTION! subroutines: **PMG.ACT** allows easy implementation of ATARI's player/missile graphics; **REAL.ACT** implements the use of floating point numbers from within ACTION! programs; **IO.ACT** allows many advanced disk file manipulations; many more procedures and demo games.

THE ACTION! TOOLKIT Disk & Manual **$39.95**

THE MAC/65 TOOLKIT

A special library of macros that will quickly add the macro equivalent of many BASIC commands to your source file programs, such as: **POKE, GOSUB, GET, SETCOLOR, PLOT, player/missile graphics, smooth scrolling,** plus many more commands.

THE MAC/65 TOOLKIT Disk & Manual **$39.95**

Now Available At Your Software Dealer

QSS™

Optimized Systems Software, Inc.

1221B Kentwood Avenue, San Jose, California 95129 (408) 446-3099

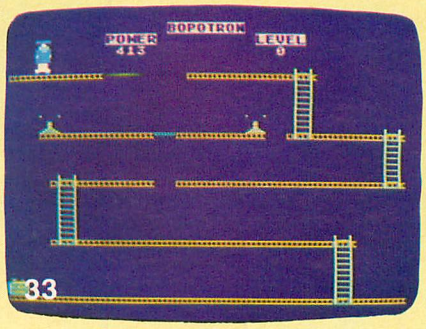
CIRCLE #101 ON READER SERVICE CARD

NO. 24 NOVEMBER 1984

THE MAGAZINE FOR ATARI® COMPUTER OWNERS™

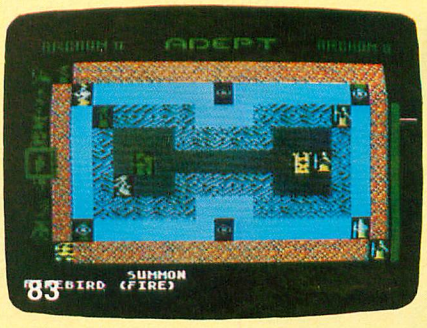
ANALOG COMPUTING

FEATURES

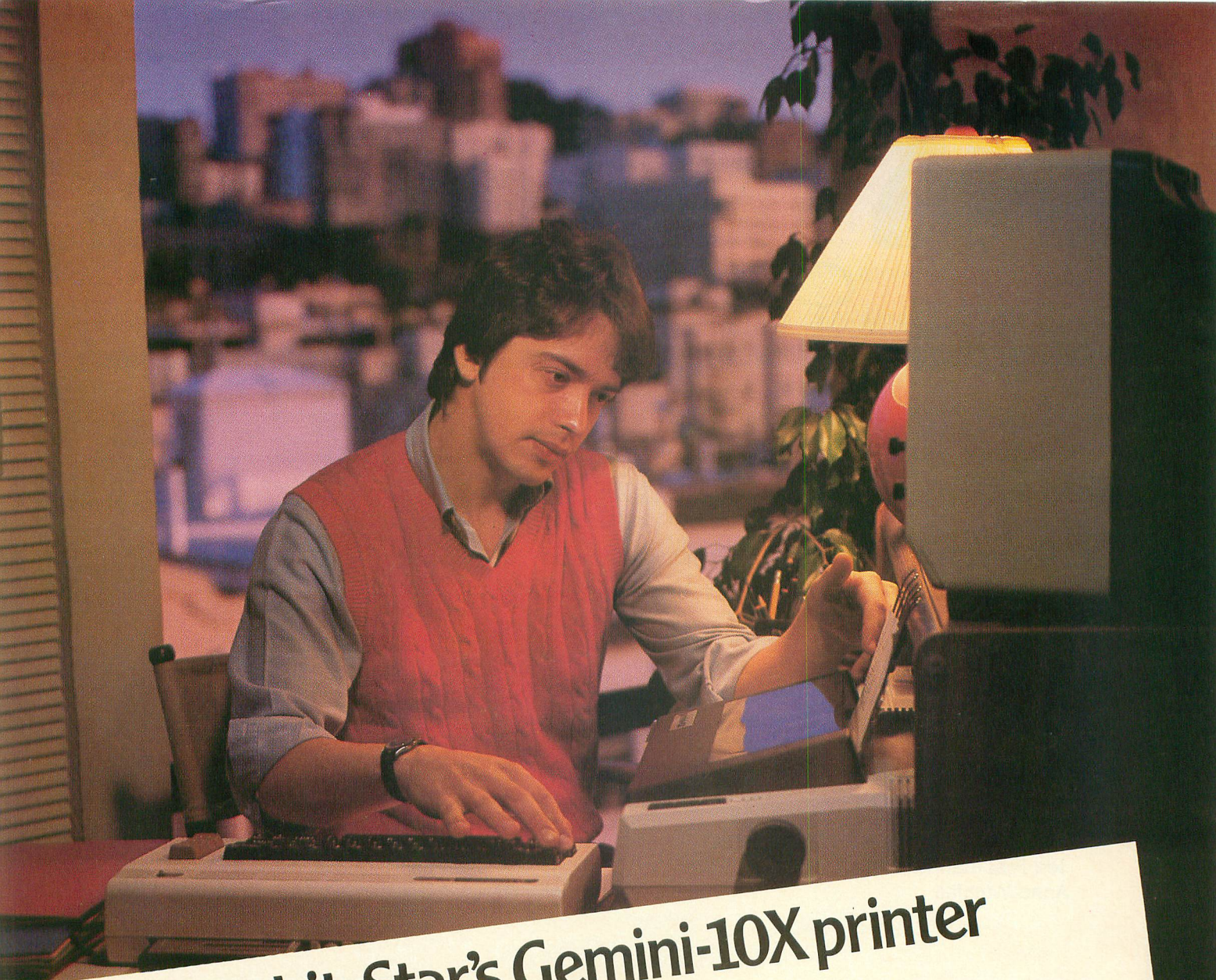


REVIEWS

COLUMNS



ANALOG COMPUTING (ISSN 0744-9917) is published monthly for $28 per year by ANALOG 400/800 Corp., 565 Main Street, Cherry Valley, MA 01611. Second class postage paid at Worcester, MA and additional mailing offices.



Bit by bit, Star's Gemini-10X printer will grow with you.

"Having a quickly growing marketing business means more and more paperwork. And the way I type, I knew early on that I needed a printer.

"I chose the Gemini-10X dot matrix printer because I wanted a printer that could do it all.

"First, the Gemini type is excellent. Actually good enough for letters. And I can also create my own type. For example, I designed "SteveScript." It's a combination of two different type styles and it makes my letters really stand out.

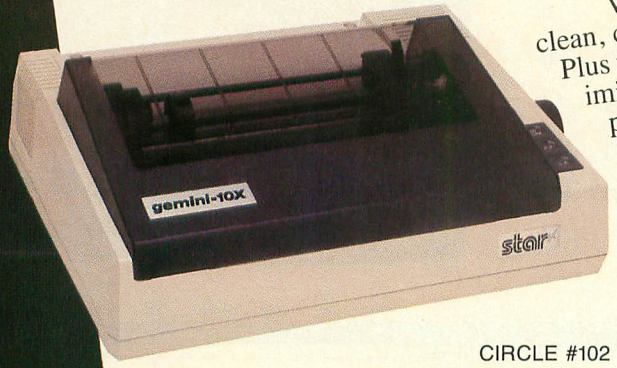
"Plus, my Gemini can do all sorts of advanced graphics, which a daisywheel can't.

"And it's much faster. No more time wasted typing. It was a pleasure to throw the whiteout away."

Steven Clarke—San Francisco, California

With Star's Gemini-10X printer you get 120 characters per second of clean, crisp type, multi-function versatility and steady, dependable service. Plus the addition of a Gemini "PC Printer" specifically designed to maximize the performance of all IBM* PCs and PC-compatibles. All at a price that works.

It's everything you need in one printer.



*IBM is a registered trademark of International Business Machines Corp.

star micronics, inc

THE POWER BEHIND THE PRINTED WORD.

200 Park Avenue, New York, NY 10166

Dallas, TX (214) 456-0052 • Boston, MA (617) 329-8560
Chicago, IL (312) 773-3717 • Irvine, CA (714) 586-8850

CIRCLE #102 ON READER SERVICE CARD

ANALOG COMPUTING
STAFF

Editors/Publishers
MICHAEL DESCHENES
LEE H. PAPPAS

Managing Editor
JON A. BELL

Production Editor
DIANE L. GAW

Contributing Editors
DONALD FORBES
BRADEN GRIFFIN, M.D.
TONY MESSINA

East Coast Editor
ARTHUR LEYENBERGER

West Coast Editor
JIM DUNION

Art Director
BOB DESI

Contributing Artists
AMY DESCHENES
GARY LIPPINCOTT
DAVID NOZZOLILLO
LINDA RICE

Technical Division
CHARLES BACHAND
TOM HUDSON
KYLE PEACOCK

Advertising Manager
MICHAEL DESCHENES

Distribution
PATRICK J. KELLEY

Production/Distribution
LORELL PRESS, INC.

Contributors
RANDOLPH CONSTAN
HAROLD JOHNSON
ROBERT LUCE
STEVE PANAK

For subscription information and service call toll-free:

1-800-345-8112

in Pennsylvania call
1-800-662-2444

ADVERTISING SALES

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

Los Angeles	—	**(213) 467-2266**
San Francisco	—	**(415) 864-3252**
Chicago	—	**(312) 445-2489**
Denver	—	**(303) 595-4331**

Gerald F. Sweeney & Associates
P.O. Box 662
New York, NY 10113
(212) 242-3540

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 400/800 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. 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 Computing**. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply self-addressed, stamped envelope. U.S.A. newstand distribution by Eastern News Distributors, Inc., 111 Eighth Ave., New York, NY 10011.

READER COMMENT

Not a BBS.

The BBS known as Software Safaris (otherwise listed as Soft Safaris or Sw. Safaris) was discontinued about one year ago, so that I might use the Atari for my own projects.

Since that time, I have seen my number published as an active BBS several times! Please don't call any of these numbers expecting a computer linkup: (405) 685-2027, 636-0218.

Thank you,
Michael Ray
Oklahoma City, OK

Ape-Face feedback.

In response to your review of the **Ape-Face** printer interface, I would like to update you on Digital Devices' product line.

Mr. Bachand evaluated one of the *very* first interfaces we ever produced, and it did have a rather unconventional picture of an ape. Now, however, the **Ape-Face** used is much more aesthetically and visually appealing.

The **Ape-Face** is now available with an extra I/O connector for daisy-chaining other peripherals, and it costs no more than the older model. Digital Devices has developed two new products which will be released soon.

One is the **U-Print**, a printer interface for Commodore machines, and the other is a low cost **Printer Buffer**, which works with the **Ape-Face**, as well as many other standard computers and printers.

Thank you for evaluating **Ape-Face** in **ANALOG Computing**.

Sincerely,
Charles L. Frazier
President, Digital Devices Corp.
Atlanta, GA

Compiling in 3-D.

I am impressed on how the 3-D image plotter (**Solid States**, issue 16) by Tom Hudson worked, and I have a couple questions.

First of all, I want to know how to change the Atari 1020 plotter routine to fit my Prowriter.

Second, I want to compile this program on my **A.B.C.** (A Basic Compiler) **Compiler**, but there's a square root on Lines 500 and 520. According to COMPUTE! Book's *Mapping the Atari*, there is a routine at $BEB1 (48817 dec.) that will calculate the square root of the number at locations $D4-$D9(212-217). I tried this by taking my number (N) and doing the following steps:

```
10 N=1000
20 M5B=INT(N/256)
30 L5B=N-M5B*256
40 POKE 212,L5B
50 POKE 213,M5B
60 ? USR(48817)
```

For some reason, this came back with *ERROR - 9* or *Array or String DIM Error.* Is this supposed to print my answer or is it supposed to die? Could you please show me how to use the floating point package for this application?

Thank you.
Jeff Lamb
Livermore, CA

If you'd like to print ***Solid States*** *graphic images on your Prowriter, check out* ***Son of Solid States*** *in* ***ANALOG Computing*** *issue 22. There's also a routine to print the images on Epson printers in the same issue.*

Unfortunately, the ***A.B.C. Compiler*** *can't be used to compile* ***Solid States****. The three-dimensional transformation calculations require the use of floating-point values, and* **A.B.C.** *works only with integers.*

The square root routine you mention is not located in the system's floating-point library, but is in the BASIC cartridge and also requires the use of floating-point values, stored in BCD (Binary Coded Decimal) format. It is not reecommended that you JSR to this routine directly, nor is it easy to do in BASIC.

—TH

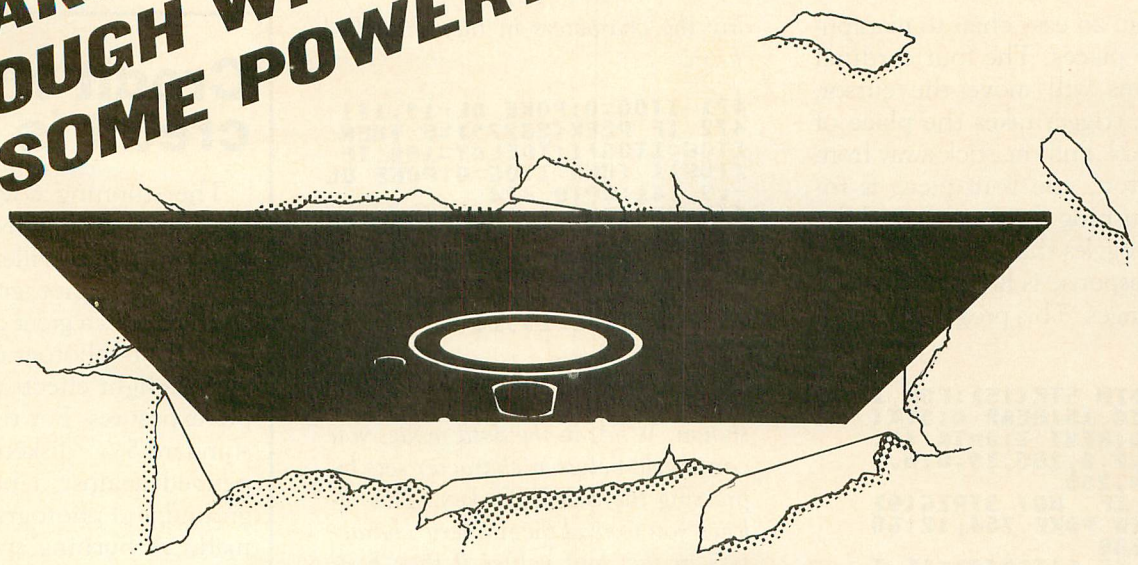
Remote Micro-Puzzler.

I am a great fan of Atari Graphics 7½ and have spent many hours in creating and modifying the wonderful pictures that can be produced in this mode. Consequently, I was fascinated by **Micro-Puzzler** (issue 22). Mr. Hearin is to be commended on a superb job!

My son and I spent a few hours in front of the screen attempting to piece together some of our favorite files with moderate success. Since I do spend a number of hours at the keyboard, I like to lean back and relax when I "play."

If one takes the time to key the following lines into Larry's excellent program, the joystick can be

BREAKS THROUGH WITH AWESOME POWER!



TOP-DOS

POWER YOU COMMAND.

FRIENDLY POWER

TOP-DOS unleashes the latent power of your ATARI computer—an amazing machine. TOP-DOS puts this power under your control. Here are just a few of its friendly features:

COMMAND MENU & HELP FILES speed your mastery of the system.
SUPERIOR STATUS DISPLAY keeps you informed. Shows free memory, disk-drive configuration, and state of TOPS-DOS options.
ERROR-CODE TRANSLATOR deciphers numeric codes into English.
BREAK-KEY ABORT lets you change your mind in mid-command.
FULL SCREEN USE shows you what you've done. Keeps 23 ines of past operations on display.
UNDELETE COMMAND rescues an accidentally-deleted file.

POWER TO PLEASE

Upgrade to TOP-DOS. Owners are delighted. You will be too. Only $49.95. No risk, 30-day MONEYBACK GUARANTEE.

SPECIFICATIONS

Memory required: 32K
Computers: ALL ATARI
Disk drives: 1 to 8
5¼": Single-density, Double-density, Double-sided Double-density
8": Double-density.
Ramdisk: AXLON or MOSAIC.
Files: Single density: 64*
Double density: 128*
Sectors: Single density: Up to 944
Double density: Up to 1968
Memory–residency: 0700-1A80 (hex)
(Same as ATARI DOS-2)
Commands: 55
(All ATARI DOS-2 + 40 more)
Command options: 35

See TOP-DOS at your dealer. If not available, you may order direct from ECLIPSE SOFTWARE, 1058-G Marigold Court, Sunnyvale, CA 94086,(408) 246-8325.

DEALER INQUIRIES WELCOME

SOPHISTICATED POWER

TOP-DOS offers professional features found only in systems on much larger machines. Whatever your experience level, you will appreciate the flexibility and power of this advanced system. Here are some examples of TOP-DOS's powerful features:

MACHINE LANGUAGE MONITOR & EDITOR allows you to access and change bytes in memory.
COMMAND FILE CAPABILITY permits you to simply and rapidly execute a complex sequence of commands.
"HELLO" FILE executes automatically on boot-up.
SET COMMAND enables you to customize your system: Configure disk drives and select TOP-DOS options.
FILE DIRECTORY COMMAND lets you choose: Alphabetization, the number of columns in the listing, and the inclusion of deleted & open files.
MEMORY MAP shows you the memory areas used by the Binary Load command.
ONE-LINE COMMANDS saves you time and conserves screen space, once you are familiar with the command syntax.
DOS-RESIDENT OPTION speeds your transfer between TOP-DOS & BASIC, or other programs.

ECLIPSE

TOP-DOS includes all the features of its predecessor, DOS-MOD. TOP-DOS and DOS-MOD are trademarks of ECLIPSE. ATARI is a registered trademark of Atari, Inc. Prices are subject to change without notice. Shipping charges are prepaid. California residents add 6.5% sales tax ($49.95 + 3.25 = $53.20).

*An advanced version is available to TOP-DOS owners (at additional cost), which doubles the number of files, as well as adding a number of other features.

used from an easy chair to manipulate the pieces. The four cardinal directions will move the cursor, and the trigger takes the place of RETURN. Pull the stick away from the button, and your piece is rotated. Pushing the stick toward the button toggles the display. The keyboard response is little affected by the changes. This program is a delight!

```
15 DIM STK(15):FOR I
=5 TO 15:READ A:STK(
I)=A:NEXT I:DATA 40,
255,7,0,255,39,6,0,1
5,14,255
372 IF  NOT STRIG(0)
 THEN POKE 764,12:GO
TO 380
374 IF STICK(0)=15 T
HEN 380
376 POKE 764,STK(STI
CK(0))
395 IF A=255 THEN 37
2
```

Sincerely,
Chet Walters
Girard, OH

More on Create-A-Font.

I agree with Mr. Randolf Constan (issue 18 **Reader Comment**) in his praise of the **Create-A-Font** program published in issue 16. I also noticed that the menu slowly becomes unreadable, as more and more letters are changed.

Mr. Constan's display list interrupt modification does clear that problem up nicely, but it causes difficulty elsewhere. . .When one selects the data option (wherein the program displays a four-by-four grid of both normal and inverse video characters and the numeric values for each line), the characters displayed are always hearts (the values are correct for the character selected). If it would be possible to communicate this to Mr. Constan for his attention, a valuable option of the program would be rescued.

I have scheduled "Understanding Display List Interrupts" for myself too far in the future to attend to this matter myself. . .

Donald McEntee
Webster Groves, MO

The following modifications work with those in issue 18 to allow displaying the characters in their modified form.

```
471 ITOG=0:POKE DL-19,141
472 IF PEEK(53279)=5 THEN
ITOG=ITOG+1:IDELAY=100:IF
ITOG>1 THEN ITOG=0:POKE DL
-19,141:GOTO 474
473 IF ITOG=1 THEN POKE DL
-19,13
474 IF IDELAY>0 THEN IDELA
Y=IDELAY-1:GOTO 474
480 IF STRIG(C0) THEN 472
481 POKE DL-19,13
```

Now, when the data mode is selected, the modified characters will be shown. While in the data mode, you can toggle between character sets by pressing the SELECT key. This allows you to read the numeric characters in the data values if they have been changed. Simply add these lines to your **Create-A-Font** *program (as modified in issue 18).*

—TH

I am slightly confused about copyright laws on public domain software. . .and I'm wondering if you could help me.

If a program published in **ANALOG Computing**—or any other magazine—is copyrighted and has a good program design or efficient subroutines that can be used in other programs, is it breaking the copyright for a programmer to use those routines or general program design in his own programs if these are to be published?

I certainly would appreciate any assistance you could give me.

Chris Cammack
Oviedo, FL

The best rule of thumb to follow in these cases is to contact the magazine in question. Each publisher may have a different view of how to handle this problem.

For programs from **ANALOG Computing**, *just write us a letter, telling us which program is involved and what you plan to do with your program when it's completed. We normally only ask for written credit in the program documentation. Naturally, if you're writing the program for publication in* **ANALOG Computing**, *no permission is necessary.*

—TH

Credit where credit is due.

The stunning cover for our last issue was provided by Bruce Bennett, a Hicksville, NY-based professional photographer.

Bruce does a great deal of special effects photography, using various light effects and multiple exposures. For the cover, a standard 5¼" diskette was suspended against a black background and photographed normally. A burning sparkler was positioned behind the disk and exposed several times, using different colored filters for each exposure.

The result — an explosive magazine cover (and, probably, a disk you could never hope to use again).

PRINTERS

Alphacom 40C w/interface . 99.95
Alphacom 80C w/Interface 189.95
Axiom AT-550 .... 279.00
Epson.....Call
Prowriter I....Call
Riteman.....Call
Silver Reed...Call
Toshiba 1351..Call
Toshiba 1340..Call
Legend 880....239

Triangle Replacement Keyboard for 400 ..... 54.95

Back Chair.....79.95
(Ergonomic Computer Chair)

SUPERPRINTER PACKAGES

Gemini 10X and Apeface 333
Prowriter and Apeface 409
Legend 880 and Apeface 308

No additional ship. charges on printer packages in Continental USA

MOSAIC

48K RAM 94.00
64K RAM/400....149.00
64K RAM/800+ Cable Kit #1 .....169.00
64K Expander for 600 XL 99.95

Bring the trivia craze home with **P.Q. The Party Quiz Game** for the Atari 800 & 800XL (disk only) ..... Call

ATARI

ATARI is a trademark of ATARI, INC.

Atari Inc. has cut all hardware and software prices please call for latest prices!!!

star micronics·inc

THE POWER BEHIND THE PRINTED WORD.

Gemini 10X ...265
Gemini 15X ...389
Delta 10X 419
Delta 15X 579
Radix 10X 579
Radix 15X 699
Powertype 339

DISK DRIVES

Indus GT .....Call
Percom .....Call
Trak .....Call
Astra 2001 .....Call

MODEMS

Hayes Smart Modem 300 .....Call
Mark II 79.00
Mark VII/Auto Ans/ Auto Dial .....Call
Mark XII/1200 Baud .Call
MPP 1000 C .....Call
R-Verter Modem Adaptor 39.95
Prometheus .....Call

INTERFACES

Aid Interfast I ...Call
Ape Face .....Call
R-Verter Modem Adaptor 39.95
MPP 1150 .....Call

☆ *Call for prices on joysticks, printer cables, blank floppy disks, and other computer accessories* ☆

A T A R I S O F T W A R E

ACCESSORIES

Ape-Link 29.95
Big Foot 16K Buffer .....Call
Big Foot 32K Buffer .....Call
Big Foot 64K Buffer .....Call
Gemini 10X 8K Upgrade.....Call
Koala Pad - D 69.95
Koala Pad - Cart 74.95
Humpty Dump - D 29.95
Monitors .....Call
Compuserve Starter 27.95
Vidtex 29.95
EIS Subscription Kit 64.95
Elephant SS/SD 17.00
Verbatim SS/DD 20.00
Elephant SS/DD 20.00
Alien Voice Box II-D 99.95
Analog Compendium 9.95
Atari Assembler 14.95
Compute's Machine Lang/Beg 12.95
Inside Atari Dos 19.95
Mapping the Atari 14.95
Compute's 1st-3rd Books Atari-EA 12.95
Printer Stand 15.95
Omnimon 82.95
Omniview 80 .....39.95
Bulk Diskettes SS/DD w/sleeves/100 .....1.29 Each
Printer Ribbons .....Call
Ramrod XL .....Call
WICO Joysticks .....Call

ACTIVISION

Decatholon - Cart 29.95
Beamrider - Cart 29.95
Hero - Cart 29.95
Pitfall II - Cart 29.95
River Raid - Cart 29.95

AVALON HILL

Call for items and prices

BRODERBUND

Bank Street Writer - D 49.95
Choplifter - D 23.95
Drol - D 23.95
Gumball - D 20.95
Loderunner - D 23.95
Mask of the Sun - D 27.95
Operation Whirlwind - D 27.95
Spelunker - D 23.95
Stelth - D 23.95
Whistler's Brother - 23.95

CBS SOFTWARE

Call for items and prices

DATASOFT

Bruce Lee - D/T 23.95
Dallas - D 23.95
Heathcliff - D/T 23.95
Letter Wizard - D 34.95
Micropainter - D 23.95
Spell Wizard - D 34.95
Juno First - D/T 20.95
Lost Tomb - D/T 27.95
Mancopter - D/T 27.95
Meridan III - D/T 27.95

ELECTRONIC ARTS

Archon - D 29.95
Pinball Construction - D 29.95
M.U.L.E - D 29.95
Murder/Zinderneuf - D 29.95
One On One - D 29.95
Archon II - D 29.95
Financial Cookbook - D 37.95
Music Construction - D 29.95

EPYX

Dragons/Pern - D 27.95
Fun With Art - Cart 27.95
Gateway to Apshai-Cart....27.95
Jumpman Jr. - Cart 27.95
Jumpman - D/T 27.95
Pitstop II - Cart 27.95
Puzzlemania - D 23.95
Summer Games - D 27.95
Temple of Apshai - D/T....27.95

INFOCOM

Deadline - D 29.95
Enchanter - D 23.95
Infidel - D 34.95
Planetfall - D 24.95
Sorcerer - D 34.95
Starcross - D 29.95
Suspended - D 29.95
Witness - D 34.95
Zork I, II or III - D 27.95
Sea Stalker - D 24.95
Cutthroats - D 24.95

MICROPROSE

Chopper Rescue - D/T 23.95
Floyd/Jungle - D/T 23.95
Hellcat Ace - D/T 23.95
Mig Alley Ace - D/T 23.95

MICROPROSE (cont'd.)

NATO Commander - D/T 23.95
Solo Flight - D/T 23.95
Spitfire Ace - D/T 23.95
Wingman - D/T 23.95
Air Rescue I - D/T 23.95
Challenger - D/T 23.95
F-15 Strike Eagle - D/T 23.95

MISCELLANEOUS ATARI

Diskey - D 34.95
Ultra Disassembler - D 34.95
Zombies - D 23.95
Codewriter - D 69.95
Star League Baseball D/T...22.95
Star Bowl Football - D/T....22.95
Death in the Caribbean-D...27.95
Dino Eggs - D 27.95
The Heist - D 23.95
Gyruss - Cart 34.95
Star Wars - Cart 34.95
Master Type - D/Cart 27.95
Flight Simulator II - D 37.95
S.A.M. - D 41.95
Castle Wolfenstein - D 20.95
Compuserve Starter Kit 27.95
Home Accountant - D 49.95
Ken Uston's Blackjack - D ..49.95
Megafont II - D 19.95
Monkey Wrench II-Cart....37.95
Movie Maker - D 41.95
Pogo Joe - D 20.95
Ultima III - D 41.95
Jupiter Mission - D 34.95
Boulder Dash - D/T 20.95
Scraper Caper - Cart 34.95
Miner 2049'er - Cart 34.95
Spy Hunter - Cart 30.95
Tapper - Cart 29.95
Up 'N Down - Cart 29.95
Beach Landing - D 27.95
Pic Builder - D 27.95
Astrochase - D 20.95
Flip-Flop - D/T 20.95
Sargon II - D/T 16.95
Odesta Chess - D 49.95
Millionaire - D 37.95
Spy vs. Spy - D 23.95
Q-Bert-Cart.....34.95

MISC. ATARI (cont'd.)

Lifestyle - D 34.95
MPP Modem Driver - D 19.95
Microfiler - Cart 34.95
Microcheck - D 34.95
Montezuma's Revenge - Cart 34.95
Mr. Do's Castle - Cart 34.95
Frogger II - Cart 34.95
Net Worth - D 54.95
Sticky Bear - D 27.95
Fischer Price - Cart 19.95
Windham Classics - D 19.95
Omnitrend Universe - D 69.95
Raid Over Moscow - D 27.95
Space Beagle - D 23.95
Adventure Writer - D 41.95
Beachhead - D 23.95
Letter/Spell Perfect - D 74.95
Harcourt/Bruce S.A.T. - D...59.95
S.A.G.E. Graphics Editor - D 39.95
Strip Poker - D 23.95
Millionaire - D 37.95

OPTIMIZED SYSTEMS

Action - Cart 69.95
Basic XL - Cart 69.95
MAC/65 - Cart 69.95
DOS XL - D 27.95
Action Aid - D 27.95
C65 - D 59.95
Handy-Writer - D 89.95
Postal Tool - D 41.95
Print Tool - D 41.95

SIERRA ON-LINE

Aquatron - D 20.95
Dark Crystal - D 27.95
Homeword Speller - D 34.95
Homeword - D 49.95
Oil's Well - D 23.95
Prisoner - D 23.95
Quest For Tires - D 23.95
Ultima II - D 41.95
Ultima I - D 23.95
Wiztype - D 23.95

SPINNAKER

Adventure Creator - D 22.95
Aerobics - D 27.95
Alf in the Color Caves - C ...22.95
Alphabet Zoo - Cart 22.95

SPINNAKER (cont'd.)

Cosmic Life
Delta Drawing - Cart 22.95
Facemaker - Cart 22.95
Fraction Fever - Cart 22.95
Grandma's House - D 19.95
Kids on Keys - Cart 22.95
Kindercomp - Cart 22.95
Search/Amazing Thing - D .22.95
Snooper #1 - D 22.95
Snooper #2 - D 22.95
Story Machine - Cart 22.95
Trains - D 22.95

SSI

Carrier Force - D 41.95
Combat Leader - D/T 27.95
Cosmic Balance II - D 27.95
Cosmic Balance - D 27.95
Broadsides - D 27.95
War in Russia - D 55.95
50 Mission Crush - D 27.95
Questron - D 34.95
Rails West - D 27.95
Bomb Alley - D 41.95
Computer Ambush - D 41.95
Galactic Adventures - D 41.95
Computer Baseball - D 27.95
Reforger 88 - D 41.95
Objective Kursk - D 27.95

SYNAPSE

Air Support - D/T 23.95
Alley Cat - D/T 23.95
Blue Max - D/T 23.95
Dimension X - D/T 23.95
Drelbs - D/T 23.95
Electrician - D/T 23.95
Encounter - D/T 23.95
Fort Apocalypse - D/T 23.95
Necromancer - D/T 23.95
New York City - D/T 23.95
Pharoah's Curse - D/T 23.95
Quasimodo - D/T 23.95
Rainbow Walker - D/T 23.95
Relax Stress Reduction System 99.95
Shamus Case II - D/T 23.95
Zepplin - D/T 23.95
Synfile - D 54.95
Syncalc - D 54.95
Syntrend - D 54.95
Synchron - D 29.95
Syncomm - D 29.95
Synstock - D 27.95

D-Disk T-Cassette Cart-Cartridge

To Order Call Toll Free
800-558-0003

For Technical Info, Order Inquiries, or for Wisc. Orders -
414-351-2007

NEW LOWER PRICES

NEW PRODUCTS

ComputAbility™

NO SURCHARGE FOR MASTERCARD MasterCard OR VISA VISA

ORDERING INFORMATION. Please specify system. For fast delivery send cashier's check, money order or direct bank transfers. Personal and company checks allow 2 weeks to clear. Charges for COD are $3.00. School Purchase Orders welcome. In CONTINENTAL USA, include $3.00 shipping per software order. Include 3% shipping on all Hardware orders, minimum $3.00. Mastercard & Visa please include card # and expiration date. WI residents please add 5% sales tax. HI, AK, FPO, APO, Canadian orders—add 5% shipping, minimum $5.00. All other foreign orders, please add 15% shipping, minimum $10.00. All goods are new and include factory warranty. Due to our low prices, all sales are final. All defective returns must have a return authorization number. Please call 414-351-2007 to obtain an RA# or your return will NOT be accepted for replacement or repair. Prices and availability are subject to change without notice.

COMPUTABILITY
P.O. Box 17882
Milwaukee, WI 53217

ORDER LINES OPEN
Mon-Fri 11 AM - 7 PM CST
Sat 12 PM - 5 PM CST

CIRCLE #106 ON READER SERVICE CARD

Griffin's Lair
Educational Programs Review

by Braden E. Griffin, M.D.

Procrastination is the mother of circumvention. I realize that my motto for success may not stir the hearts of men, but the light bulb has already been invented. In spite of more pressing matters—like the 23rd Olympiad and ideal golfing weather—I have managed to meet yet another deadline.

I always wondered what perverse individual was responsible for the textbook titles of yesteryear: *Adventures in Math* or *Adventures in Civics*. "Adventures in civics?" Gimme a break. The only adventure in civics was trying to stay awake. However, this month we will look at some computer adventures designed specifically for children. These games provide much in the way of educational benefits, as well as being quite entertaining. Anyway, someone here promised I would do it. Does the name *Jon* ring a *bell?*

Adventure games encourage a uniquely personal interaction with the computer. Many of them promote the development of reading skills, particularly in the area of comprehension. The logical thinking used to solve problems—an essential for success—is a prime ingredient in adventures. Every game of this genre involves a journey of sorts, and an aptitude for knowing where you are (and where you are going) is developed, since it is of prime importance. The concept of mapping out an area is basic to one's quest for the Holy Grail.

With increasing complexity, attention to detail is imperative, and note taking skills become invaluable. All in all, adventure games motivate children to develop proficiency in many areas, and mental exercise is as important as physical. I doubt that anybody will record a hit song "Let's Get Mental," but let's.

SEASTALKER
INFOCOM, INC.
55 Wheeler Street
Cambridge, MA 02138
40K Disk $39.95

The first computer adventure game was created by Willie Crowther and Don Woods not far from the home of Infocom. This classic journey through Colossal Cave opened the portal to an area of computer software which has grown to unfathomable proportions.

The early adventure games were composed entirely of text, with the player initiating action through an interpreter or "puppet." Simple two-word commands, like *TAKE AXE*, or *GO NORTH*, made it possible to achieve this interaction. The unusual images created provided a unique experience for everyone. Because of the exceptional graphic capabilities of the computer, illustrated text adventures using the

basic two-word commands soon followed. Not far behind were graphic adventures controlled entirely by joystick and even action adventures, using an arcade-style format.

The parallel to the evolution, or maybe devolution, of leisure time activities from books to television to video games (Tinkers to Evers to Chance) is interesting. It would be great if someone could refashion the rather simplistic format of the original text adventures without stifling the imagination with pictures. Enter Infocom.

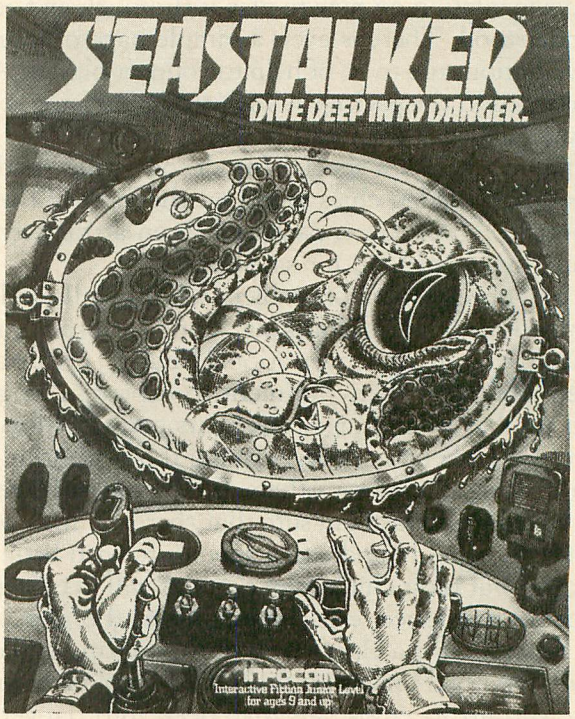
Beginning with **Zork I: The Great Underground Empire**, Infocom introduced Interlogic machine language adventures, where one communicates in complete sentences, rather than two-word commands. The **Zork** trilogy and a wide variety of other scenarios from Infocom are among the most popular adventures, text or otherwise, on the market today. Following in the footsteps of this august family of interactive fictional escapades is **Seastalker**, written with the beginning adventurer in mind.

Before discussing the particulars of **Seastalker**, it is important to examine the use of what is referred to as Interlogic machine language. As mentioned earlier, communication involves the use of complete sentences. Multiple objects may be used, and more than one command may be input at a time. *TURN ON THE HYDROJET THEN SET THE THROTTLE TO FAST* is a valid command, with the computer executing each in order and responding in kind.

Most of the over 800 words in the game's vocabulary require correct spelling, at least of the first six letters of the word, which is what the computer recognizes. Using *N* for *NORTH*, or similar abbreviations, is permitted. When entering a room for the first time, a complete description is given, including the objects in view and visible exits. On re-entering the same room, a less extensive description is given, unless the VERBOSE command is employed to elicit the complete version (or, conversely, the SUPERBRIEF mode, which just displays the name of the room). Other frequently used commands include INVENTORY (a list of possessions), LOOK, SAVE (up to five different game locations may be saved), RESTORE (previous game position), DIAGNOSE (assesses physical condition) and WAIT (allowing time to pass). An extremely helpful feature is the SCRIPT command, which allows the use of a printer and provides a transcript of the interaction. Reviewing the conversations and descriptions from previous sessions may prevent needless duplication of effort and is always quite interesting.

Direct conversation with the characters in the story is accomplished by simply ASKing the individual ABOUT something or someone. The emphasis on the input of commands which are grammatically correct is very welcome. The computer may respond to improperly phrased commands with comments like *YOU MUST SUPPLY A VERB* or *I FOUND TOO MANY NOUNS IN THAT SENTENCE* or even *YOU CAN'T USE MULTIPLE DIRECT OBJECTS WITH ATTACK*. Any game capable of improving a child's vocabulary and spelling, while stressing fundamentals of grammar, has merits far in excess of its intrinsic entertainment value.

Seastalker has been produced by the combined effort of Stu Galley (Infocom's **The Witness**) and Jim Lawrence. Lawrence, an author of juvenile fiction for years, has ghostwritten for series such as Nancy Drew and the Hardy Boys. The result of this collaboration is an absorbing adventure in which the characters are so well developed that they practically come to life.



Seastalker.

The story centers around the undersea research station of Inventions Unlimited, or the Aquadome, and the perils that surround it. The player assumes the role of the main character who has just developed a super submarine, the ultramarine bioceptor Scimitar, at a distant research lab. Suddenly, a message is received concerning an attack on the Aquadome by a monster of some sort—the crew there is in grave danger.

Here I come to save the day! One must pilot the previously untested sub to the Aquadome through Frobton Bay, avoiding the speedboats and ships above, as well as treacherous shoals and submerged wrecks below the surface of the water. Navigation can be tricky, but attempts at sabotage make the journey even more dangerous. Having arrived at the Aquadome, the adventure is just beginning.

A crack team of experts is available to help with the dilemma—well, maybe *some* of them have less than the most honorable intentions. I don't want to

ruin the story (the good guys do not *always* win), but suffice it to say that Murphy's First Law prevails—everything seems to go wrong. The problems must be approached logically, and the crew's expertise used to solve them.

Any game, particularly a text adventure that causes sweaty palms and the feeling that one's heart is trying to exit through the throat as each command is anxiously entered, has succeeded in creating realistic interaction.

As usual, Infocom's packaging adds even more realism. Included is a submarine logbook containing scale drawings of the research lab, the Aquadome and the Scimitar, as well as serving as the user's manual. The documentation on how to play is thorough and easily understood. There is even a sample page of interactive fiction sure to be of help to younger players. Also included in the package is a nautical chart, top secret Infocards with information about the crew and the equipment, and a special Infodecoder film used for revealing hidden clues found on the back of the Infocards. All of this is packaged in an attractive and functional portfolio for safekeeping.

The development of problem solving skills and the emphasis on logical thinking, along with the unique interaction using conversational English, create a superb educational environment to explore. Designed for the beginner of age nine years and up, hints are provided when necessary, making this adventure less frustrating than many others. (Like Frosted Flakes, it is not just for kids!) I'm sure that there are nine-year-olds who could handle this adventure with little difficulty, but most children of that age will need a good deal of assistance, especially early in the game.

Seastalker is a class act. It's fun, exciting and educational. (Did I mention fun?) All young adventurers are certain to enjoy this dive deep into danger.

DRAGON'S KEEP
TROLL'S TALE
SIERRA ON-LINE, INC.
Coarsegold, CA 93617
48K Disk $29.95

I do not know if Sierra On-Line was the first to introduce the illustrated adventure for the Atari, but **Wizard and the Princess** was the first one that I had seen and, subsequently, purchased. That initial endeavor remains one of the finest adventures ever produced.

Now the same company has introduced a junior adventure for children ages eight years and older, using their successful illustrated text format. For those of you unfamiliar with this type of adventure, it follows the same basic format of a text adventure with graphic enhancements.

Two-word commands are used to communicate with the computer (e.g., *OPEN DOOR*). Many objects, characters, etc. specifically named in text adventures must be discerned from their appearance in the illustrated version. A table may be the only readily visible fixture in a room, the secret note beneath it not graphically depicted until the command to look under the table is given. Although sometimes limited in the degree of complexity compared to text-only adventures, the illustrated adventure offers a distinctly different challenge with its visual clues.

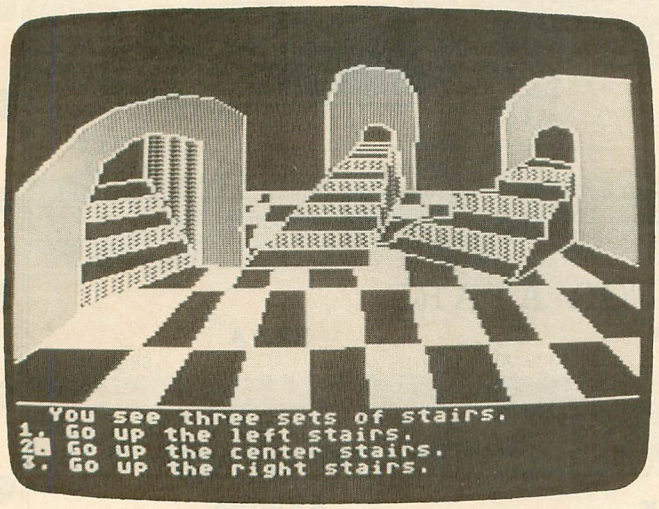
Dragon's Keep.

In **Dragon's Keep** and **Troll's Tale**, Sierra On-Line has maintained the same approach to adventuring, with one significant difference. Instead of having to decide the precise command to enter—probably the single most difficult aspect of an adventure game—the player is given an option of what to do. On entering a room, one decides whether to LOOK IN THE BOX, GO TO THE WINDOW or GO BACK OUT THE DOOR. These commands are entered by moving the cursor with the space bar until it's over the desired command and hitting the RETURN key. A child must be able to read at second or third grade level, but does not have to type in the commands.

The two game themes are appropriately geared for young children. In **Dragon's Keep**, a magical dragon holds sixteen animals captive in a variety of places. The child must find and then release all of them. In **Troll's Tale**, the object is to discover sixteen hidden treasures and return them to the Dwarf King. In both adventures, if the "bad guy" appears in a particular location, the player must return there later to accomplish the specific task.

Although challenging to the youngster, the frustration of playing "What am I thinking?" with the computer is eliminated. Persistence and trying all of the options will eventually lead to success (just like real life). One doesn't have to be concerned with entering a room with a single, locked exit without the key—and being stranded forever. Best of all, the ever present fear of extinction, so much a part of the usual adventure, does not exist.

There are a number of commands which, when selected, result in silly responses. These diversions are of little consequence in the adventure itself, but are right on target when it comes to hitting a child's funny bone.



Troll's Tale.

In addition to the usual educational benefits of adventures, these two programs stimulate the use of reading skills in concert with object recognition. This perceptual coordination is of great importance in the further development of reading comprehension. To assist in the strengthening of skills necessary to map out an area, a map is included with each game. There are also stickers of found objects, which can be placed on the map in the appropriate locale.

The instructions for playing are on the disk and are very easy to understand, requiring little, if any, adult supervision. The graphic illustrations and command options are very well coordinated—not in the least confusing. In spite of the titles, these adventures are devoid of any evidence of violence.

The next step to higher levels of adventure gaming—and the development of the more complex skills required to play them—have been made much easier with the introduction of these two games. **Dragon's Keep** and **Troll's Tale** will provide hours of stimulating fun and excitement for children with an adventuresome spirit.

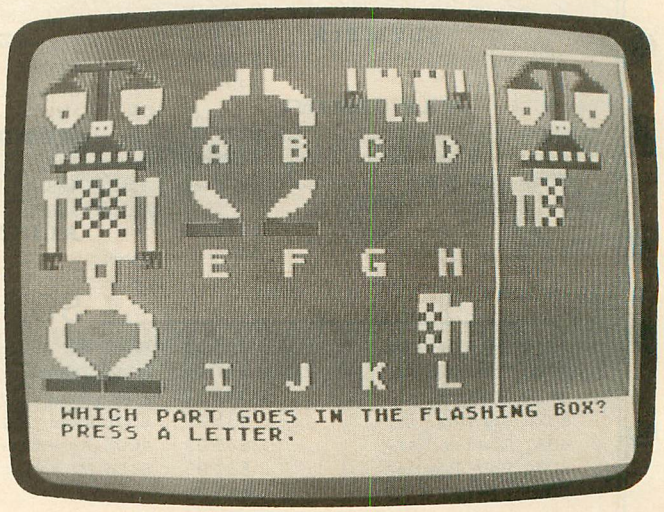
TONK IN THE LAND OF BUDDY-BOTS
Sprout Software
MINDSCAPE, INC.
3444 Dundee Road
Northbrook, IL 60062
48K Disk $34.95
(314) 480-7667

Emergency! Emergency! There is trouble in Buddy-Bot Land! A Buddy-Bot's parts are scattered everywhere. The Buddy-Bot needs Tonk to collect his parts and put him together again.

So begins an adventure of Herculean proportion, at least for the four- to eight-year-olds for whom this game is intended. When the staff at **ANALOG Computing** gave me this game for possible review, I was fascinated by its title. That, as much as anything, was the reason I loaded it up immediately, instead of sticking it in the "programs to review tomorrow" stack, as usual. I'm glad I did, because here is a game with an adventure theme for very young children. . .which fits in perfectly with the other reviews this month.

Having arrived in Buddy-Bot Land, **Tonk** searches for the lost parts of a robot-like character called a Buddy-Bot. Depending on the level of difficulty (1-4), one of 128 varieties of these creations is divided into as few as three and as many as twelve parts, which are scattered throughout the land. Using a joystick, the player collects these parts by touching them, continuing on until the sum of all parts equals the whole (I've *always* wanted to say that).

It's not as easy as it may sound. A number of pitfalls await the brave **Tonk** as the quest progresses. Mean old Gork is a mischievous dude who lives in a castle at the edge of the CrissCross Sea. His soldiers are out on patrol, and if **Tonk** is captured, he is sent to Gork's castle—and must search for a lost part there before escaping. Gork's soldiers move pretty fast and are not easy to outmaneuver. Black holes, which suddenly appear throughout the land, and sky holes, found in the castle, are additional perils to be avoided.



Tonk in the Land of the Buddy-Bots.

Travel is mostly by foot; however, a cable car and a raft provide welcome transportation through some parts of Buddy-Bot Land. As a boon to our bold traveler, there are special caves to be found in the kingdom. One may enter these caves and play one of five different games with a missing Buddy-Bot part as the reward for winning.

The games are fun in themselves and stimulate the

ALL THE BEST PRICES

ATARI HOME COMPUTERS

850 Interface	$159.00	Touch Tablet/Software	$64.99
1010 Recorder	$71.99	Light Pen/Software	$72.99
1020 Color Printer	$219.00	CX22 Track Ball	$39.99
1025 Dot Matrix Printer	$249.00	7097 Atari Logo	$74.99
1027 Letter Quality Printer	$269.00	4018 Pilot (Home)	$57.99
1030 Direct Connect Modem	$99.99	405 Pilot (Educ.)	$99.99
1050 Disk Drive	$329.00	8036 Atari Writer	$77.99
1064 Memory Module	$125.00	5049 VisiCalc	$139.99
CX30 Paddles	$11.99	488 Communicator II	$119.99
CX40 Joystick	$7.99		

600XL ..... **$ CALL**
800XL ..... **$ CALL**
1200XL ..... **$CALL**
While Supplies Last

DISK DRIVES FOR ATARI

INDUS
GT Drive $379.99

TRAK
AT-D2 ..... $389.00
AT-D4 ..... $539.00
TRAK PRINTER CABLE $34.99

RANA
1000 ..... $299.00

PRINTERS

AXIOM
AT-100 ..... $199.00
AT-700 ..... $489.00
AT-846 Interface $69.99

BMC
BX 80 ..... $269.00

EPSON
RX & FX Series ..... $Call

MANNESMAN TRALLY
Spirit 80 ..... $289.00
MT-160L $589.00

NEC
8027 Dot Matrix $389.00

OKIDATA
82, 83, 92, 93 ..... $Call

OLYMPIA
OL2 w/Keyboard $469.00
OLRO $499.00

STAR GEMINI
Gemini 10X $279.00
Gemini 15X $399.00
Radix 10 ..... $599.00

TEC
8510P $379.00
F10-20 ..... $569.00

MONITORS

AMDEK
300 Green $149.00
300 Amber $159.00
300 Color 289.00
500 Composit/RGB/VCR $399.00

BMC
Color 9191 + ... $249.00

BMC
Color 1215 ..... $259.00

SAKATA
Color 100 ..... $249.00

TAXAN
Color 210 ..... $299.00

USI
Color 1400 ..... $249.99

ZENITH
ZE122 Amber $99.99
ZE123 Green $89.99
ZE135 Color $469.00

MEMORY BOARDS
Axlon 32K $59.99
Axlon 48K $99.99
Axlon 128K $299.00

ALIEN VOICEBOX
Atari $119.99

MODEMS

ANCHOR
Volksmodem $59.00
Atari Cable $9.99

HAYES
300 ..... $199.00

MICRO BITS
MB1000 ..... $139.00

NOVATION
J-CAT $99.99

ALIEN
VOICEBOX $119.00

AXION
32K Memory $59.99
98K Memory $99.99
128K Memory $299.00

BIT-3
Full View 80 ..... $249.00

KOALA
Koala Pad (Rom) $79.99

KRAFT & WICO
Controllers ..... $Call

MICRO BITS
MB1150 Printer Interface $89.99

MONITOR & PRINTER
Cables and Interface ..... $Call

SOFTWARE

4011 Star Raiders	$31.99	8043 Ms. Pacman	$37.99	The Learning Company	$Call
4022 Pac Man	$31.99	8044 Joust	$37.99	Broderbund	$Call
4025 Defender	$31.99	8045 Pengo	$33.99	Creative Software	$Call
8026 Dig Dug	$31.99	8052 Moon Patrol	$33.99	Electronic Arts	$Call
8031 Donkey Kong	$35.99	4003 Assembler	$44.99	Epyx	$Call
8034 Pole Position	$37.99	8126 Microsoft Basic I or II	$64.99	Infocom	$Call
8040 Donkey Kong Jr.	$33.99			Sega	$Call

COMPUTER MAIL ORDER

VISA

east
800-233-8950
In PA call (717)327-9575, Dept. 117
Order Status Number: 327-9576
Customer Service Number: 327-1450
477 E. 3rd St., Williamsport, PA 17701

canada
Ontario/Quebec 800-268-3974
Other Provinces800-268-4559
In Toronto call (416)828-0866, Dept. 117
Order Status Number: 828-0866
2505 Dunwin Drive, Unit 3B
Mississauga, Ontario, Canada L5L1T1

west
800-648-3311
In NV call (702)588-5654, Dept. 117
Order Status Number: 588-5654
P.O.Box 6689
Stateline, NV 89449

MasterCard

No risk, no deposit on C.O.D. orders and no waiting period for certified checks or money orders. Add 3% (minimum $5) shipping and handling on all orders. Larger shipments may require additional charges. NV and PA residents add sales tax. All items subject to availability and price change. Call today for our catalog.

CIRCLE #107 ON READER SERVICE CARD

(continued from page 11)
development of skills in the areas of memory, concentration and visual discrimination. The Different/Alike game displays six Minibots (a smaller version of the Buddy-Bot?) and requires the child to select the pair that is either the same as or different from the others. Further practice with recognition of shapes and patterns is provided in Match the Shadow, where a Minibot shown on one side of the screen must be matched by choosing from a group of four to ten shadows of Minibots. Minibot Shuffle, not to be confused with "The Curly Shuffle," is a variation of the old shell game, which makes it necessary to concentrate on the particular box hiding the Minibot. In Remember Me, a Buddy-Bot is displayed and then vanishes from the screen. The task is to select individual body parts from a group of four and reassemble an exact replica of the original. This is a real toughie and will certainly enhance memory skills.

Tonk in the Land of the Buddy-Bots

The last game is Buddy-Bot Puzzle. Here, a Buddy-Bot is shown on the left; its component parts (twelve in all) are randomly scattered in the middle of the screen. In an area on the right side of the screen, a red box flashes in a variety of positions—where specific body parts belong. The appropriate part is selected as the puzzle is pieced together. Thoughtfully, an option exists allowing one to just play the games without having to participate in the adventure.

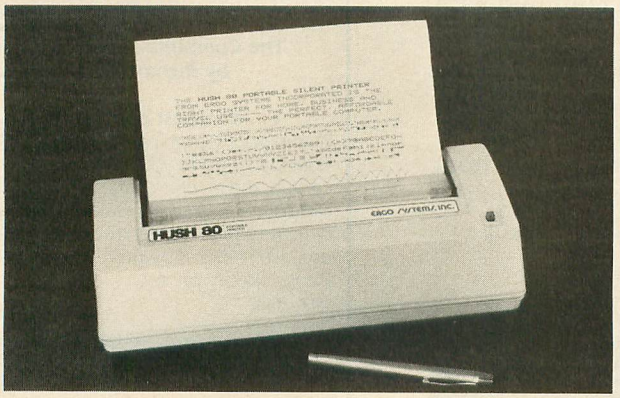
Tonk in the Land of the Buddy-Bots is an exceptional example of educational software. The colorful graphics, clever animation and original music further enhance this quality product. The experience of learning one's way about in a strange land—and the excitement of danger without the possibility of any real harm befalling our hero—makes a nice combination. This multifaceted adventure game for very young children looks like a winner. □

NEW PRODUCTS

by Lee Pappas

ERGO SYSTEMS' PORTABLE PRINTER

Ergo Systems' latest offering is claimed to be the lowest priced 80-column dot matrix thermal printer in the world. The **HUSH 80S** portable printer features bidirectional printing in a full 80 columns, at 80 characters/second and 4800 dots/inch graphics.



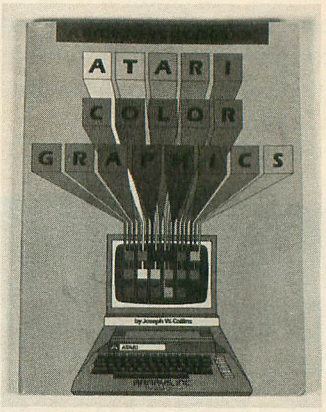
This 28-ounce printer is available in two Atari-applicable models, each with optional rechargeable Ni-Cad batteries.

The **HUSH 80S** is serial interface, and the **HUSH 80P** parallel interface. The printers are packaged with an interface cable, paper and wall transformer.

$159.00. Ergo Systems, 1360 Willow Road, Menlo Park, CA 94025 — (415) 322- 3746.

BEGINNER'S GRAPHICS WORKBOOK

Atari Color Graphics: A Beginner's Workbook is designed to teach the reader/user about graphics modes 0 through 15 and how to achieve results in all of those formats.



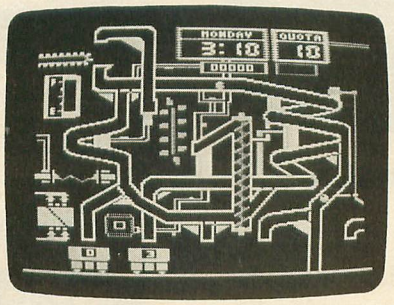
In-depth coverage, along with chapters on programming and many demo-programs, complete this well-presented book of 202 pages, softbound.

$12.95, by Joseph W. Collins from Arrays, Inc., 11223 South Hindry Ave., Los Angeles, CA 90045.

WANTED: GUMBALL FACTORY MANAGER

Broderbund's new **Gumball** game has you working as a "bin filler" at the Sticky Sole Gumball Factory. Your mission there (should you decide to accept it) begins by screening the gumballs as they roll down the various candy chutes and conveyors.

After you've sorted the proper colored candies into the correct bin, you must get them over to shipping to meet your day's quota (keeping an eye on the time clock). Should Mr. Nitpicker catch an error in your work, he'll charge in and tip your bin over.



And, just when things were finally beginning to get *worse*, the word goes out that spazzed-out dental assistants are trying to sabotage the factory by blowing it up!

48K disk, $29.95, Broderbund Software, 17 Paul Dr., San Rafael, CA 94903 — (415) 479-1170.

GUARD YOUR DATALINK

Dataguard is designed to provide security and privacy to modem users whose phone line doubles as a general voice communication line. This device protects the line from accidental pickups on extension phones, which can effect data loss or disconnection.

Two units are available, an "in-phone" model—easily installed in a couple of minutes and invisible after hookup—plus a "snap-in" model that is 12 feet long, replacing your standard line cord.

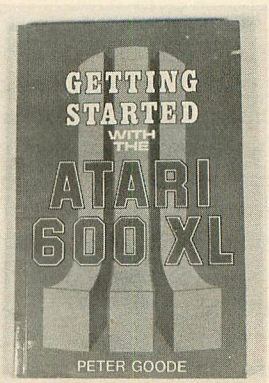
Dataguard doesn't interfere with normal phone operations or permanently tie up a phone strictly for data communications. It carries a full one-year warranty and is FCC approved.

$39.95, Control Industries, Box 6292, Bend, OR 97708 — (503) 389-1969.

GETTING STARTED — A HEAD START

Getting Started with the Atari 600XL is a worth-looking-into new book that covers the BASIC basics, from the ground up.

Some of the chapters help the reader understand and utilize sound effects, simple graphics and plotting, and some of the Atari peripherals. *Getting Started. . .* also contains one of the best explanations of the ERROR codes I've seen to date.

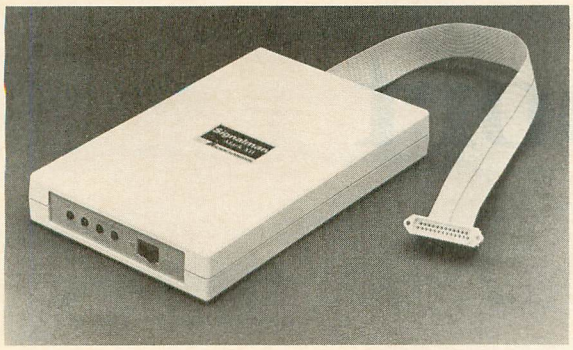


BASIC commands, with their proper syntax, are explained here in a way that makes understanding easy. Short, sample programs will help you learn how to use certain commands, such as: strings, graphic routines, the joystick/paddles and sound.

Getting Started is 140 pages long, softbound.

Written by Peter Goode, the book retails for $12.95. It's distributed by David & Charles, Inc., North Pomfret, VT 05053.

HIGH-END MODEM FROM ANCHOR



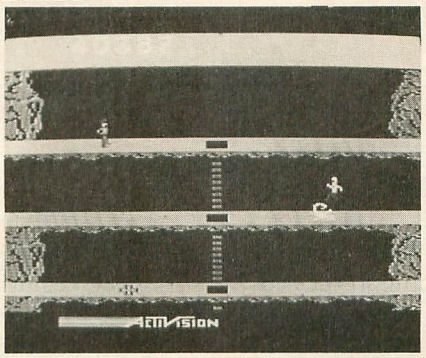
Anchor Automation offers a new modem in its **Signalman** line, the **Mark XII**. Emulating the command structure of the well-respected Hayes **Smartmodem**—with even more enhancements, **Signalman** can be operated manually through a keyboard (without computer coding) or automatically, with the ability to answer or originate calls at 1200 baud (special lines) or 300 baud (standard lines).

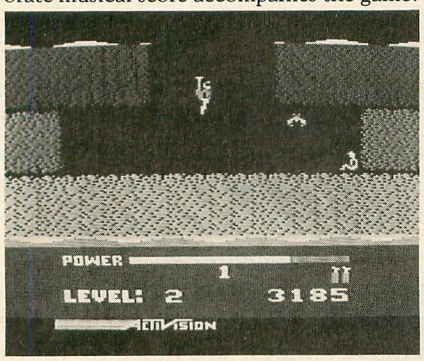
The **Mark XII** detects dial tone and busy signals, then displays the status on the CRT. It uses RS-232 serial interfacing and comes with a built-in cable and two telephone jacks.

The cost of the **Mark XII** is $399.00; from Anchor Automation, 6913 Valjean Ave., Van Nuys, CA 91406 — (213) 997-6493.

ACTIVISION ANNOUNCES TWO NEW GAMES

Pitfall II: Lost Caverns takes us on the further adventures of **Pitfall** Harry. Similar in concept to its best-selling predecessor, **Pitfall II** finds Harry leaving the jungle for the far away mountains of Peru, with you controlling his arms and legs. As Harry—with his niece, Rhonda, and pet mountain cat, Quickclaw—sets out to explore a vast underground complex full of killer frogs, eel-infested waters, bats, scorpions and much more, he's on the lookout for the lost Raj Diamond. Eventually, Harry must also find his niece and cat, as they all split up when they entered the Lost Caverns. Many surprises await, and Harry can now swim, fly (by grabbing a balloon) and climb. An elaborate musical score accompanies the game.



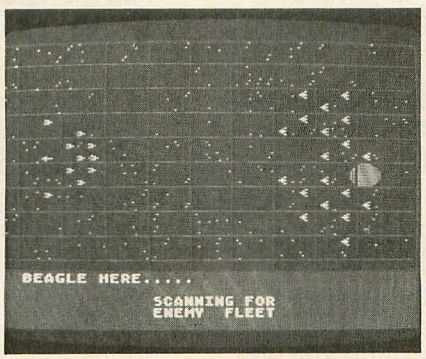


Activision has also released **H.E.R.O.** (Helicopter Emergency Rescue Operations), where a "man on a prop-pac" must find and bring back trapped miners from a huge cave network.

Pitfall II and **H.E.R.O.** list for $34.95 from Activision, 2350 Bayshore Frontage Road, Mountain View, CA 94043 — (415) 960-0410.

QUEST OF THE SPACE BEAGLE

Avalon Hill's latest interactive adventure is a sequel to **Jupiter 1999** and is, like its predecessor, set in outer space. As the only survivor from a previous other-worldly episode, you are (unfortunately) the candidate for hero in this battle between two alien worlds.

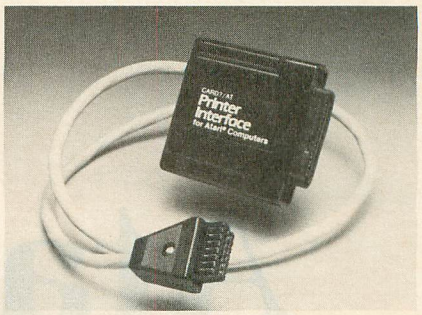


The peaceful Faunians have chosen you as their savior to fight the evil Gentuzians, in a pre-emptive strike against the Gentuzian homeworld. After you win the battle for the "good guys" (assuming that you *do* win), they'll want you as emperor. . . and why not? But all you desire is to repair the **Space Beagle** and get out.

Other surprises are in store for you in the 48K game, which can be saved to continue play at a later date. $35.00 from Avalon Hill Microcomputer Games, 4517 Hartford Rd., Baltimore, MD 21214 — (310) 254-9200.

CARD?/AT FROM CARDCO

CardPrint/AT is a new parallel interface which enables simple plug-in and print capability for the Atari computer owner. It's compatible with any standard parallel input printer, whether it be dot matrix, thermal, daisy wheel or letter quality. . . even multi-color pen printer/ plotters.



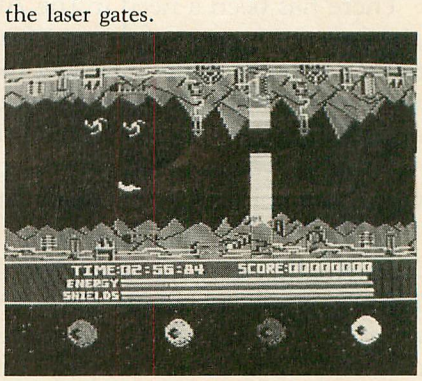
This unit negates the need for an Atari 850 interface module or special programming, and accepts all the standard Atari print commands. All cables and connectors are included, along with a lifetime warranty.

$79.95, CARDCO, Inc., 300 South Topeka, Wichita, KS 67202.

IMAGIC'S 1-2-3

This new disk from Imagic contains three new games for the Atari home computers.

Laser Gates (my favorite) is a **Defender**-type shoot-'em-up with horizontal scrolling and plenty of things to fire at. The goal is to reach the CPU and destroy it, then proceed to the next planet, and so on. You must negotiate the solid grey walls, missiles swirling asteroids, floating hamburgers and the laser gates.



In **Wing War**, you must hunt and destroy the Dragon's enemies and change them into crystals. Crystals, when brought back to the Dragon's lair, increase your firepower and overall crystal strength. Other crystals can be found at the geyser and volcano. When Gargok the Guardian is destroyed, firepower and defensive energy are gained.

Quick Step is a sort of **Q*bert** "jump on the food" game, where you must also outsmart your opponent by using magic and bites.

Cost: $24.95 for 32K disk. From Imagic, 981 University Ave., Los Gatos, CA 95030 — (408) 399-2200.

A Report on AtariCon '84

by Arthur Leyenberger

The first international Atari users convention, or AtariCon, was held in Southfield, Michigan on August 25 and 26, 1984. The convention was sponsored by the Michigan Atari Computer Enthusiasts (MACE) and the Capitol Hill Atari Owners Society (CHAOS). They should be given credit for their hard work in planning, organizing and staging the convention.

There has been a need in the Atari users community for a convention of this sort for some time. Other major computer groups have met like this, so why not Atari users? With the recent buyout and reorganization of Atari under Jack Tramiel, the need to make Atari users' voices heard is even more important. It was in this spirit that this year's AtariCon was held.

Starting over.

Originally, AtariCon was scheduled for October of 1983. It was being sponsored by MACE alone and was to have over fifty exhibit booths, dozens of seminars and a major Atari display. With visions of trying to duplicate CES (the Consumers Electronics Show), plans became too ambitious. This, together with a lack of support from the local group, doomed the 1983 convention.

The fact that AtariCon '83 was a bust hurt the organizers of this year's efforts. Many companies became disillusioned with MACE and the idea of an Atari-only convention after the first time. It was also felt that last year's attempt at a show was viewed as a MACE event, rather than a national Atari convention. Therefore, the two local user groups, CHAOS and MACE, teamed up to organize and run the August convention.

This year, it was decided that there would be a show, regardless of how small it turned out to be. As long as there were a few vendors willing to participate, the organizers were determined to get the first Atari convention off the ground.

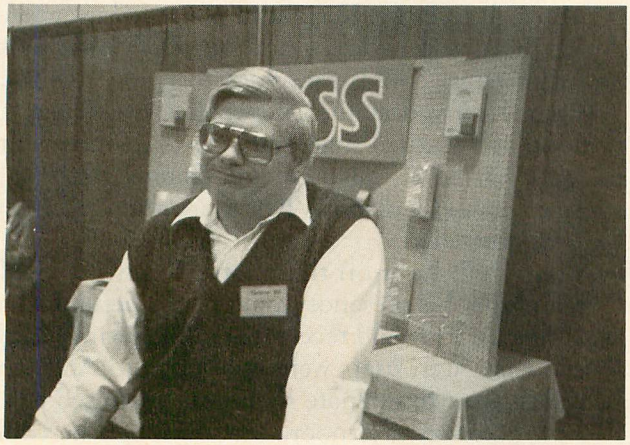
Ike Hudson from CHAOS and Mike Lechkun from MACE were the two coordinators of this AtariCon. MACE provided about 75% of the $5000 budget, and the two groups split the work evenly. About thirty people volunteered their efforts to help with the convention. The Jersey Atari Computer Group and the Penninsula Atari group of Virginia were contributors, and the Wiesbaden (West Germany) Atari group sent all they could afford—mostly moral support.

In a nutshell.

There were approximately twenty booths displaying their wares at AtariCon. The most notable, OSS (Optimized Systems Software), was not only hosting a booth, but also gave most of the seminars on programming and other topics. Friendly Bill Wilkinson was there, as was Clinton Parker of Action! fame. OSS was showing several new products. New "toolkits" for BASIC XL, Action! and MAC/65 were being demonstrated—and selling fast. These products contain utilities, subroutines and macros for getting the most out of a particular programming language. They retail for about $40 and are available now.

OSS was also demonstrating a new word processor called **The Writer's Tool**. It comes on the now-famous OSS (bank-selectable) super-cartridge and supports single or double density disk drives. It looks like a powerful, yet easy-to-use word processor. It will be-

come available by the time you read this and will sell for about $130.

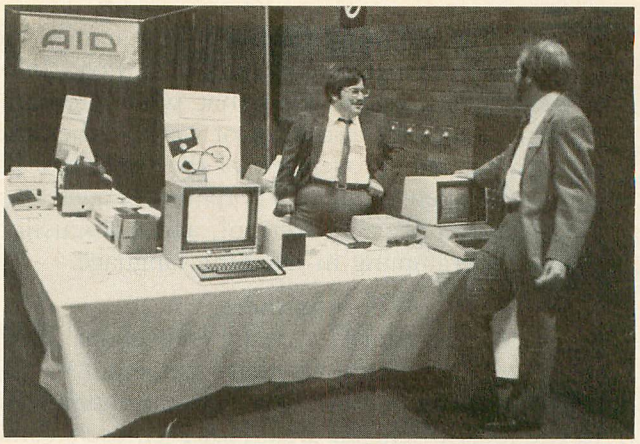


The amiable Bill Wilkinson of OSS.

Another vendor present was Advanced Interface Devices (AID), whose rep drove all the way from Florida to make the show. AID's president, Herman Price, said that there was a lot of interest in his products. AID makes the **R-Verter**, a $49.95 serial bus modem adapter. It allows most modems and RS-232 devices to work with the Atari, without requiring an 850 interface. Their other product is called the **Interfast-1**, a buffered (4K memory) printer interface for Atari computers. It also allows various character sets to be created, stored and used on dot matrix printers. Currently available, it sells for $129.95.

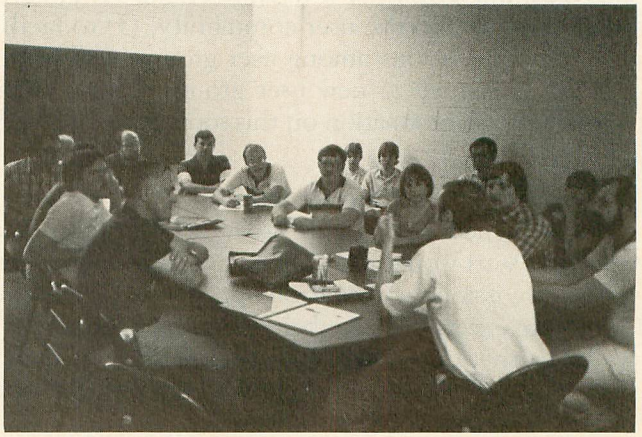
Show coordinators deep in discussions. In the foreground: Mike Lechkun of MACE (left), from CHAOS, Rob Peck (center) and Ike Hudson (right).

Alpha Systems was also there, selling their utility software. They have a sector copier/analyzer, a cartridge backup program, a keyboard customizer that allows macros to be defined and used with BASIC, and a graphics screen dump. George Morrison from Alpha told me that they were doing a brisk business. He must have been right, since there were always people crowding around his booth.



Herman Price, president of Advanced Interface Devices, talks with a potential customer after a long day.

The other booths were occupied by either local retailers or other Atari user groups. MACE, JACG, MilAtari, Penninsula ACE, London Ontario group and CHAOS all had booths at the convention. These were good places to introduce yourself and make new friends. Many of the user groups had t-shirts for sale. An interesting one being sold by MACE was a white shirt with a Commodore logo on it. Overlaid on the Commodore logo was the international road sign symbol for "not," a red circle with a slash through it. Very clever.



Thirteen user groups discuss their future and AUGI (Atari User Groups International).

Even though the vendor turnout was light, all these companies should be given credit for attending and supporting the user convention. All of the vendors reported that they made money at AtariCon, even though Atari did not show up. The rest were glad that they came and promised to show up next year at AtariCon '85.

Learning opportunities.

Quite a few seminars were held during the convention. Eric Hanson of the MilAtari group gave a very interesting lecture on 3-D graphics. Starting with simple examples, and showing more complex techniques as he went along, Hanson had the audience of about fifty listening attentively. Mark Rose of OSS gave a

talk on the ins and outs of being a programmer. He used examples of programming in BASIC, Logo, Pilot and machine language to show that, while programming can be fun, it's also a lot of work. Other and seminars included an overview of the three BASIC languages, an open forum for a potpourri of questions, a good lecture on OSS's Action!, and a discussion on game programming in machine language.

Who was there.

Attendance at AtariCon numbered about 1800. People of all ages—many families—came out for the day. The majority of Atari users who attended were from the local Michigan area. However, users came from as far away as Oklahoma, New Jersey, Canada and Panama. Interestingly, no West Coast user groups or members attended the show. Unfortunately, as a result of the smaller than expected crowd attending the show, MACE and CHAOS will lose approximately $2500 on the event.

The most positive result from AtariCon '84 was that thirteen user groups met and decided to form a coalition. Called Atari User Groups International (AUGI), this group will allow Atari users to have a single voice when communicating with Atari—or anybody else. AUGI's goals are: (1) to provide a common voice for Atari users with Atari and other vendors; (2) to provide a common point of communications with Atari, other vendors and the user community; (3) to facilitate communications among user groups; and (4) to provide assistance to new user groups. User groups will be getting information on this soon from CHAOS or MACE.

Can we talk?

As it turned out, a side story at AtariCon was whether Atari would show up or not. Through a series of discussions with various people at Atari, the show's organizers had a verbal commitment that Atari would be there, complete with their CES booth, and "in a big way." At one point, it was believed that one of the Tramiels would also attend.

Had Atari been present at the convention, chances are that the two sponsoring user groups would have at least broken even. The events that led up to Atari's cancelling at the last minute are interesting—and a classic example of miscommunication.

Ike Hudson, president of CHAOS and one of the principal coordinators of the convention, began talking to Earl Rice and Mark Cator of Atari user group support in January of 1984. By July, these two men, long known to user groups, were no longer working for Atari. Throughout July, several discussions were held with West Shell, Atari's Director of Marketing. On July 31, West Shell finally confirmed to CHAOS that Atari would "attend in a big way."

In mid-August, calls were placed to Atari to confirm their attendance and to make any last minute arrangements. Atari management was unavailable, and the calls were not returned. It was also learned that West Shell was no longer with Atari. Finally, on August 24, the day before the convention, Mike Aldrich, another of the convention's organizers, was able to get through to Brian Kerr, head of special events for Atari. Brian said he knew nothing about AtariCon, and that Atari could not afford to come, anyway.

When I arrived at the convention late Friday afternoon and learned that Atari was a no-show, I immediately tried to call Leonard Tramiel. Instead, I spoke to James Copland, VP of Marketing and an ex-Commodore man.

After a half-hour on the phone, I convinced Mr. Copland that his presence would help amend the bad press that Atari would receive for not showing up. He agreed that Atari users were important to Atari and he didn't want to alienate them. He promised to arrive on Saturday afternoon and, although he could not talk about products, he would be glad to make a statement. I went to sleep that night thinking that Copland was a sincere fellow and glad that at least *someone* from Atari would be at AtariCon.

The following morning, I learned that Mr. Copland would not be attending. A letter of apology with an offer of $500 to help defray costs was sent instead. Mr. Copland stated that he was unable to arrange flight connections that would allow him to return to Sunnyvale for a Sunday morning meeting.

The new Atari had an excellent opportunity to "hit the ground running" by attending AtariCon. Since the sale of Atari to Jack Tramiel on July 2, no information had come out of Sunnyvale. Atari could have used the convention as a means of providing information on their plans for the future—and to show that they truly support Atari users.

I believe that Atari's absence was due not to malice but, rather, to things just slipping through the cracks. July and August were difficult times for the new Atari. Having just been sold, they were trying desperately to plug the dollar leaks. Jack Tramiel is running a start-up company now, and all of Atari's efforts are directed to putting the company in a position to make money for the first time in two years.

Epilogue.

In a follow-up conversation, James Copland told me that the new Atari will not let this kind of thing happen again. It was a case of complete miscommunication. Atari did not want to offend the user groups—or any Atari user. From talking to Copland, I got the impression that Atari *would* be present at the next AtariCon.

In spite of Atari's non-attendance, AtariCon '84 was a positive experience. It showed how dedicated Atari users are and what they can do. Personally, I was able to meet many of the fine user group volunteers who help make Atari computing as rewarding as it is. I look forward to AtariCon '85 as being an even better event. We'll see *you* there. □

CAN FLYING FEET AND FISTS CONQUER THE EVIL WIZARD'S FORTRESS?

What's it like to have the lightning feet and fatal fists of Bruce Lee?

You'll find out in this death-defying game.

You have to kick, slash and punch your way through an array of deadly chambers. Where the brutal Green Yamo, terrible Ninja, exploding bushes and other dangers lurk.

Even if you survive all that, the Evil Wizard is waiting to do you in with an arsenal of flaming fireballs.

Destroy him and his fortune is yours.

Now, have you got what it takes to play Bruce Lee?

For Commodore 64, Apple II, Atari and IBM PC & PC/JR systems.

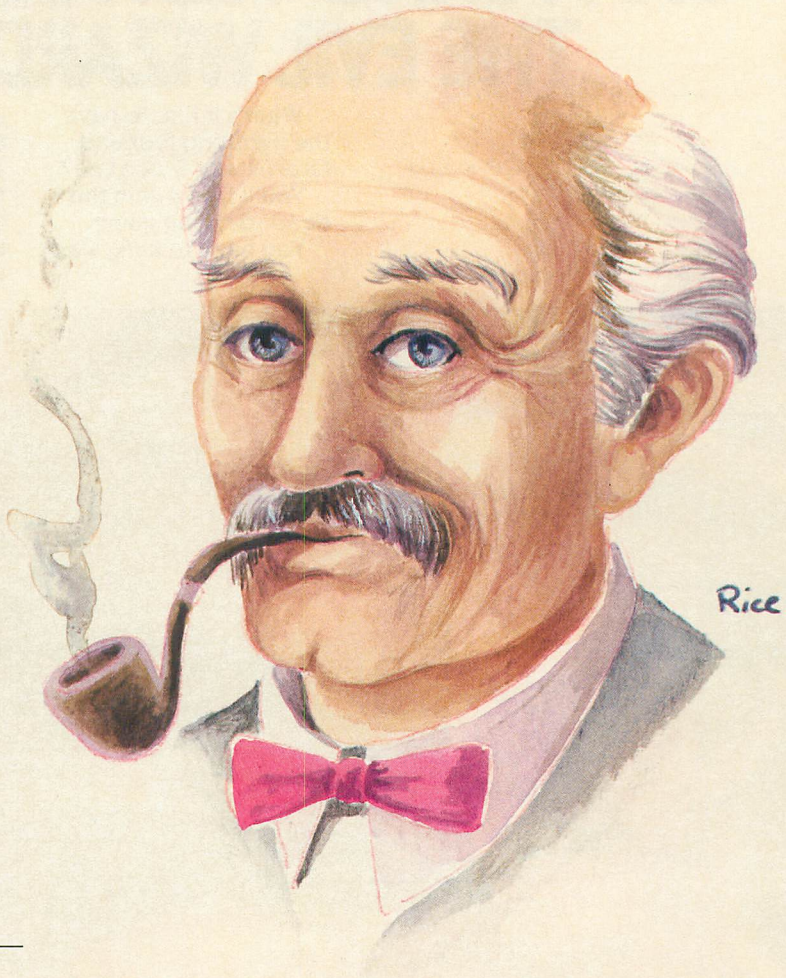
FAMOUS FACES™

Datasoft®

WE CHALLENGE YOU.

Datasoft, Inc., 19808 Nordhoff Place, Chatsworth, CA 91311 • Phone (818) 701-5161

Ask Mr. Forth



by Donald Forbes

If you wish to demonstrate FORTH, there is no better way than to follow the steps of the masters. Even pygmies see farther than giants, if they stand on their shoulders.

The first question of any newcomer to FORTH is: what does it look like? We can do no better than to take the example of best-selling author Leo Brodie and of fig-founder Kim Harris in the early pages of *Starting Forth*. They show how a small letter *F* on the keyboard can become a large letter *F* on the screen. Here is their code, just as they present it.

```
SCR # 1
 0 ( brodie large letter F )
 1 : STAR  42 EMIT ;
 2 : STARS  0 DO STAR LOOP ;
 3 : MARGIN  CR 15 SPACES ;
 4 : BLIP  MARGIN STAR ;
 5 : BAR  MARGIN 5 STARS ;
 6 : F  BAR BLIP BAR BLIP BLIP CR ;
```

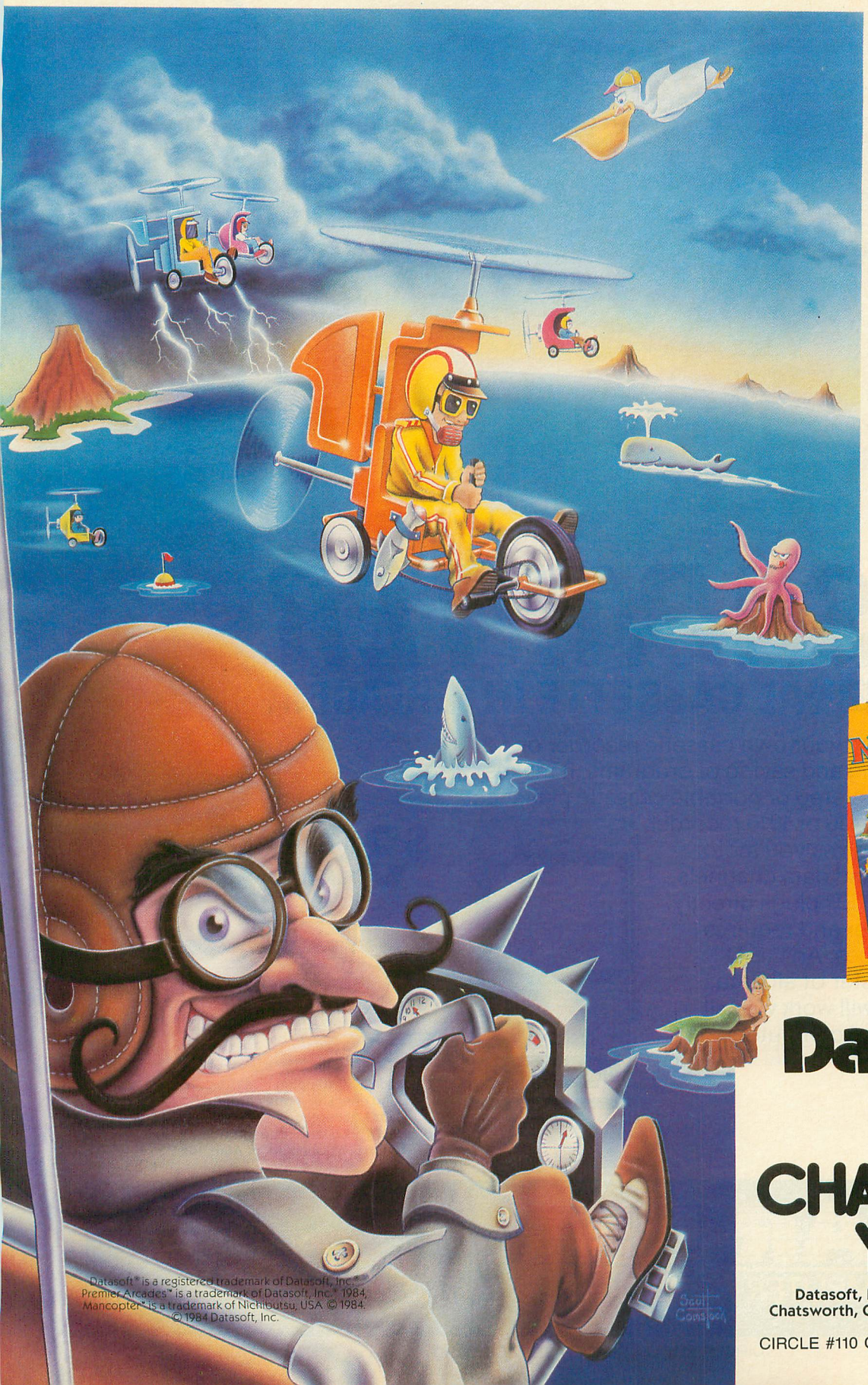
In your demo, you must point out that, in FORTH, you create a new definition with a colon followed by a name (in this case, *STAR*) and then spell out what the definition does. Here it takes *42*, which happens to be the ASCII number for an asterisk (*45* would be a *$*), followed by *EMIT*, which puts the asterisk on-screen. The next definition, *STARS*, puts *STAR* inside a DO. . .LOOP to draw multiple stars. *MARGIN* is used to do a carriage return followed by a line feed and fifteen spaces. *BAR* and *BLIP* also draw on the previous definitions. In the final line, *F* puts them all together, so that an *F* on the keyboard puts a large *F* on the screen.

You can do this demo from the keyboard, but it makes more sense to put your code on disk as well. Then you can do your first demo from the keyboard but repeat it at will by loading it from the disk.

The six-line demo is an excellent illustration of the compactness of FORTH, but is apt to be confusing to a newcomer. A much smarter idea is to put the demo on one or two screens, with only one word or instruction per line—and *copious* comments after each word. This method uses much more space but makes everything many times more clear. Your audience will now be able to check that they understand the purpose of each word in the final program.

One of the pleasures of FORTH is that it is nice and compact, so that you write tight code with little typing effort. You should resist the temptation to do this at the outset. If anyone else is going to be reading your code, think about putting only one word (or a few) per line and then filling up the line with lots of comments. This takes more typing, but your audience will thank you. Furthermore, when you go back

CAN YOU PEDAL FAST ENOUGH TO WIN THE GREAT MANCOPTER™ RACE?



Can you pedal fast enough to keep your human-powered copter up in the air?

Can you dodge blade-biting birds, strange swamp creatures and dastardly villains?

Can you keep your copter from falling into the mouths of hungry sharks and giant squids?

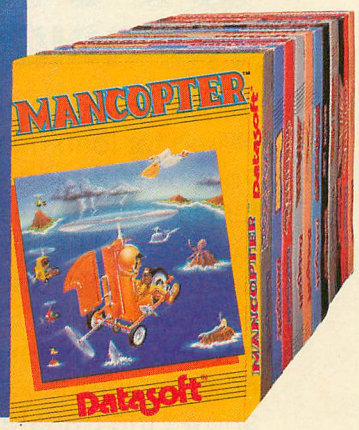
Can you fly through treacherous mangroves and lightning storms?

Can you take all this fun and frustration?

Then you're ready for the great Mancopter race.

For Commodore 64, Atari and IBM PC & PC/JR systems.

Premier Arcades™

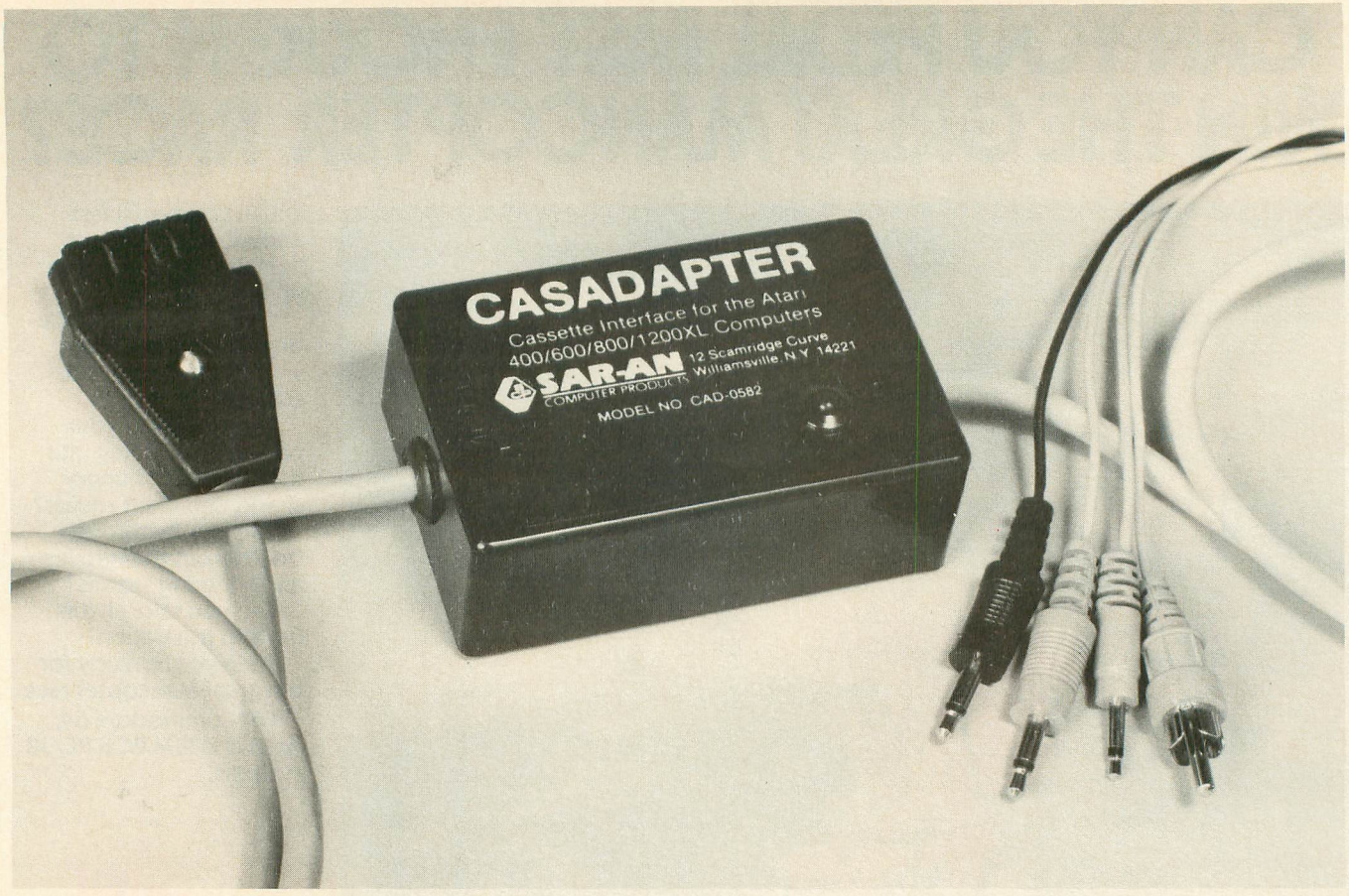


Datasoft®

WE CHALLENGE YOU.

Datasoft, Inc., 19808 Nordhoff Place,
Chatsworth, CA 91311 • Phone (818) 701-5161

CIRCLE #110 ON READER SERVICE CARD



CASADAPTER

THE ORIGINAL CASSETTE INTERFACE...

Allows you to use your own cassette recorder or stereo for loading and saving of programs. It makes your own home stereo or portable cassette player act just like an Atari 410 or 1010 recorder. CASADAPTER gives you motor control, audio, and data channels. The CASADAPTER plugs directly into the computer and requires no other interfaces. An L.E.D. shows the transfer of all data. CASADAPTER will work with all of the Atari brand computers.

$39.95

Add $2.50 shipping
Send check or money order
C.O.D. accepted
New York State Residents add 7% tax
®Atari is a trademark of Atari, Inc.

SAR-AN COMPUTER PRODUCTS

12 Scamridge Curve
Williamsville, New York 14221
(716) 632-3441

CIRCLE #111 ON READER SERVICE CARD

to review your old code, you won't have to rack your brains to remember what you did. Most importantly, when you start to present the advanced features of FORTH to beginners, you will need to make sure that they follow you every step of the way, or you'll lose them in a hurry.

Here is the one-word-per-line version of Brodie's program.

```
SCR # 2
 0 ( brodie    large letter F )
 1 : STAR      place a star on screen
 2   42        save an ASCII asterisk
 3   EMIT ;    put asterisk on screen
 4 : STARS     display multiple stars
 5   0         starting index of loop
 6   DO        begin looping N times
 7   STAR      display an asterisk
 8   LOOP ;    branch back to DO
 9 : MARGIN    make 15 space margin
10   CR        car. return, line feed
11   15        save number 15
12   SPACES ;  display these spaces
13 : BLIP      single asterisk
14   MARGIN    15-space margin
15   STAR ;    asterisk
16 : BAR       display 5 asterisks
17   MARGIN    15-space margin
18   5         save number of stars
19   STARS ;   display asterisks
20
21 : F         show large F
22
23   BAR       five stars
24   BLIP      one star
25   BAR       five stars
26   BLIP      one star
27   BLIP      one star
28   CR ;      carriage return and
29             line feed
30
31 ( Type F to show large F )
```

If you walk your users through the program line by line, they should have a good basic understanding of what a FORTH program looks like.

This program only puts one letter on the screen. How about putting five letters on the screen? That will certainly make a more impressive demo. We can put the word *FORTH* on the screen by merely repeating and adapting the code which we used for the letter *F*.

I made some changes to get the code to fit one or two screens. Everything is standard fig-FORTH. Note that it will run just as well on either **QS Forth** by James Albanese from Quality Software, **valFORTH** from Valpar International or **Team Atari Forth** by Steve Calfee and others. No matter which FORTH you are using, I urge you to get a copy of Calfee's public domain FORTH—which may be available on a bulletin board from the Bay Area Atari User Group in San Jose, CA. There is nothing to compare with it for instructional purposes—it follows religiously the fig-FORTH model and has a valuable decompiler and disassembler which I have not seen elsewhere. Sending a $15 check to the Jersey Atari Computer Group, attention of Don Ursem, 37 Clover Lane, Randolph, NJ 07869 will get you both disks number 19 and 20.

```
SCR # 3
 0 ( forth demo          7/08 )
 1 DECIMAL 125 EMIT : FF 70 EMIT ;
 2 : FSTARS 0 DO FF LOOP ;
 3 : FMAR CR 1 SPACES ;
 4 : FBLIP FMAR FF ;
 5 : FBAR FMAR 5 FSTARS ;
 6 : F FBAR FBLIP FBAR FBLIP ;
 7 : 00 79 EMIT ; : SP SPACES ;
 8 : OSTARS 0 DO 00 LOOP ;
 9 : OMAR CR 8 SP ;
10 : OBAR OMAR 00 SP 00 ;
11 : OBLIP OMAR 00 SPACE 3 OSTARS ;
12 : O OBLIP OBAR OBAR OBLIP ;
13 : RR 82 EMIT ;
14 : RSTARS 0 DO RR LOOP ;
15 : RMAR CR 15 SP ;
16 : RBAR RMAR 4 RSTARS ;
17 : RBLIP RMAR RR 3 SP RR ;
18 : R RBAR RBLIP RBAR RBLIP ;
19 : TT 84 EMIT ;
20 : TSTARS 0 DO TT LOOP ;
21 : TMAR CR 21 SP ;
22 : TBAR TMAR 5 TSTARS ;
23 : TBLIP TMAR 2 SP TT ;
24 : T TBAR TBLIP TBLIP TBLIP ;
25 : HH 72 EMIT ;
26 : HSTARS 0 DO HH LOOP ;
27 : HMAR CR 28 SP ;
28 : HBLIP HMAR HH 3 SP HH ;
29 : HBAR HMAR 5 HSTARS ;
30 : H HBLIP HBLIP HBAR HBLIP ;
31 : FOR 125 EMIT F O R T H QUIT ;
```

The first line has *DECIMAL* followed by *125 EMIT*. The word *DECIMAL* was included to make sure that we did not default to *HEX*, as Calfee's disk will do. The statement *125 EMIT* clears the screen, since *125* is the ATASCII clear screen character. *SP* is just my abbreviation here for *SPACES*. Instead of *STAR*, I used *FF* for *F* (which is *70 EMIT*), *00* for *0* (which is *79 EMIT*), and so on. *MARGIN*, for example, was abbreviated to *FMAR* and *TMAR* to squeeze each instruction into one line. As the last line shows, typing *FOR* will put up the word *FORTH*, so as to fill the whole screen. This makes for a much nicer demo.

Explaining FORTH.

Now is a good time to bring up the question: what is FORTH? Whether you want to pose this question to your audience is up to you. If they are just looking for entertainment, you may want to postpone it. But, if they have any background in computer hardware, assembly language or mathematics, they may find it fascinating. And, once anyone is hooked on FORTH, there is no turning back.

We can always explain, as most people want to do, that FORTH is a computer language and operating system, but this seems like trying to describe an automobile as a "horseless carriage" to an Eskimo who has never seen a horse. He will keep wondering about the strange animal called a horse and never give a thought to the carriage. When you ask him to repeat the explanation, he won't know where to begin.

What does FORTH do? A good question. Before answering, how about another question: what does an automobile do?

"The way to describe an automobile is first by thinking about what it is for, about its function, and note

the list of items that make up its structure," according to Harry Katzan in his book on systems design and documentation. "If you think about its function (what it is for), you won't describe it by talking about its four wheels, its engine, size, and so on. You will think about it as a means of transporting a few people from one place to another at a certain cost."

We can take the same approach. In any computer system you have a user who is looking for answers, a piece of hardware (in our case, a keyboard, screen, CPU, memory, disk and printer) and a set of software which, in the final analysis, invokes some mathematical operation or algorithm. The mathematical operation may be arithmetic (+, −, *, /, ∧) or logical (AND, OR, NOT) or comparison (=, <, >). The mathematical operation hands its answer back to the software, which passes it back on to the hardware, which hands it to the user via the screen, the printer or perhaps even the disk.

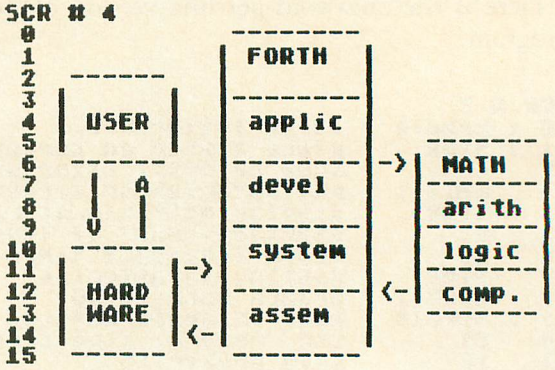
Where does FORTH fit in this pattern? The central and inescapable fact is that FORTH is the complete software interface. This one reality sets FORTH apart from all other software. There is no software in your Atari until you've loaded the FORTH disk—FORTH is the genie that brings your hardware to life and prepares it to do your bidding.

FORTH history.

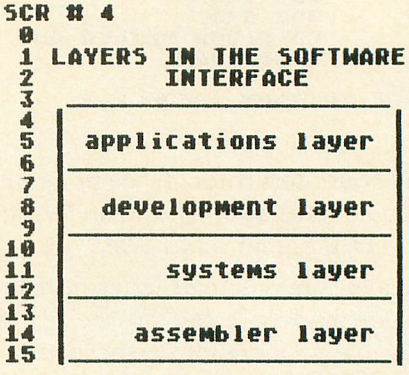
This is really how FORTH got its start. Bill Ragsdale (an electronics engineer from California, who also belongs to the Society of American Magicians) founded the Forth Interest Group and served as its president for five years, until last April. He remarks: "FORTH then (1978) was really an operating system for what I call a crippled computer. FORTH has been treated in a receptive way by users of computers with very limited resources in terms of memory, mass storage or input/output. The Varian 620i was a crippled computer. Some of the early Hewlett-Packards, the 2100 series for instance, had very limited manufacturer support. In such cases, FORTH has been graded with very high marks." To which he adds the following comment: "On the other hand, we in the FORTH community face a very real problem, in that, as the manufacturers have provided increased quality in software, the need and demand for FORTH appears to diminish. FORTH was providing some irreplaceable attributes five years ago. Now it appears that a number of those attributes are no longer as attractive as they were. For example, there is more memory space available, I/O is faster, more disk space is available, file structures are less limiting. This puts an increased challenge on people using and writing FORTH systems. Are they going to stay back in the "crippled computer" mentality, or are they going to continue to grow and follow industry needs?"

Despite the industry changes, FORTH holds a unique position as a complete self-contained software interface to the computer hardware.

Here is one diagram you can create for yourself. It shows how the user invokes the hardware to invoke the software to invoke a function that returns an answer via the software and the hardware, back to the user.



Here is another diagram for you. It may look like a chocolate layer cake, but really spells out the different layers that must be present in any working software interface to any computer hardware system. In fact, the diagram really shows why FORTH is FORTH—why FORTH is unique and can do all those marvelous things that no other software interface can.



The applications layer represents the applications programs and systems with which most users of personal (and other) computers are familiar. Word processing, database management, spread sheets, graphics packages, accounting systems and spelling verifiers are a few examples.

On the development layer, we have the tools that a programmer uses to develop his applications, including a slew of languages, from FORTRAN, COBOL, PL/1 and PASCAL to SNOBOL, APL and LISP (Lots of Irritating, Superfluous Parentheses!) to one of the latest and the one with the shortest name, C (which, in turn followed from the B language written by computer chess expert Ken Thompson in 1970). On the development layer, we also have such programs as editor and debugging packages.

At the systems layer, the software has to do cold and warm starts, as well as talk to the files and the

peripherals (which go under such names as OS, DOS, MS-DOS, PC-DOS, UNIX, VENIX and XENIX.

Underneath all these layers lies the assembly (or machine language) layer, which consists simply of *0*s and *1*s, and is the only language that the computer hardware can understand.

The ultimate justification for FORTH is that it *spans* all these layers, can do what all these layers are supposed to do and, in addition, can do—at the level of a high-level language—what no other software can. If FORTH were just another high-level computer language or just another operating system, there would be little reason to take the time to learn it; the advantages of compactness, speed and virtual storage have been neutralized to some extent by advances in computer hardware over the past seven years. Smaller and faster chips, however, have had no impact on our unique ability to remain in FORTH and float up, down and back from the applications to the development, to the systems and assembly layers as we choose —and within a single definition.

Our simple little program to put a letter *F* on the screen serves not only as an excellent demonstration of structured coding without GOTOs, but also of the multiple layers of FORTH.

The applications layer, of course, is our demonstration of the word *FORTH* on the screen.

The development layer is clearly visible on the disk screen where we created the original program by the use of the editor and the associated debugging features.

The systems layer can best be appreciated by executing the DECOMP instruction in either **valFORTH** or **Team Atari FORTH** (which makes this FORTH a worthwhile investment). If we do *DECIMAL DECOMP STAR*, we get approximately the following:

```
5AE6 :  LIT 42
5AEA :  EMIT
5AEC :  ;S
```

If we do *DECOMP STARS*, we get a clear illustration of how the systems layer of FORTH takes our coding from the development layer and converts it into executable code. This is what we get:

```
5AF8 :  0
5AFA :  (DO)
5AFC :  5TAR
5AFE :  (LOOP) TO 23292
5B02 :  ;S
```

Here FORTH's systems layer inserts the forward and backward branches that are needed for the specified iterations of the DO loop. You can DECOMP the rest of the words for a better appreciation of the FORTH systems layer.

How about the assembly layer? For that, we need some extra apparatus. We need a way to see what the assembler code looks like. We can do this easily with nothing more than the instruction *DECIMAL 0 200 TYPE*, which will type out (beginning at address *0*) the next 200 bytes of memory. This output is hard to read. We can improve readability by creating a few simple dump routines to display memory in graphics characters, or by bytes in decimal or hexidecimal. In these examples, *AAA* is the starting address and *BBB* the ending address.

```
: DUMPG BBB AAA
  DO I 1 TYPE SPACE LOOP ;
: DUMPD BBB AAA
  DO I C@ . SPACE LOOP ;
: DUMPX BBB AAA
  DO HEX I C@ . DECIMAL LOOP ;
```

Let us now dump the assembly code for our letter *F* program. The address of the code will be different, depending on what FORTH we are using, and how many words are in our dictionary. The general pattern, however, remains the same. To find the starting address, we only need ' *STAR* . (pronounced "tick STAR dot). From this information we can arrive at the starting and ending addresses for the dump.

In the dump, we see the names of the words in our program: STAR, STARS, MARGIN, BAR, BLIP and F. Each word begins with a so-called head, which con-

(continued on next page)

tains an indicator of its length in bytes, followed by the name of the word, then a pointer to the previous word (to allow dictionary searches), followed by a code field address (points to the code to be executed) and, finally, a "body" with one or more parameter field addresses (these point to previously defined words which we included in the definition of our original word).

The byte structure of each word is a topic that is longer than you can afford to include in your demo at this stage. There is one important point, however, which must not be overlooked. Dr. C.H. Ting of the Lockheed Missiles and Space Company wrote a book called *Systems Guide to fig-FORTH* ($25.00, Mountain View Press, P.O. Box 4656, Mountain View, CA 94040), which is the only one I have seen designed specifically to "deal with the inner mechanisms on how the FORTH system operates, which is essential to the understanding and effective utilization of the FORTH language." He notes that "the FORTH language is a major synthesis of many of the concepts and techniques used for some time in the computer industry, such as stacks, dictionary, virtual memory and interpreter. The single most important invention by Charles Moore in developing this language, which wrapped all these elements and rolled them into a small but powerful operating system, is the code field in the header of a definition. The code field contains the address of a routine to be first executed when the definition is called."

In the May issue of *Forth Dimensions*, he appends, "Code fields and the associated inner interpreters are the sole inventions Mr. Charles Moore brought us in FORTH. Stacks, the dictionary, indirect threaded code and virtual memory were all well-developed techniques before FORTH was invented. Using the code field to identify a specific interpreter to execute a particular command was not obvious or considered useful prior to that time. The code field sets FORTH apart from any other type of language or programming constructs, and it is the most unique feature in FORTH or FORTH-like systems. Many of the attributes associated with the FORTH language, such as compactness, simplicity and extensibility, can only be realized with the use of the code field."

The idea to keep in mind is that the CFA or code field address plays a central role in the operation of FORTH. To know how FORTH works, we must have a clear picture of the function CFA performs when it interacts with other FORTH components. A proper understanding should, as Dr. Ting says, "be able to cut through much of the mythical fog often surrounding FORTH."

No demo of FORTH, however, should get bogged down in a discussion of points of theory, no matter how important. So let us add some final fireworks to keep our audience psyched up.

This program will put a colorful pattern on the screen:

```
: MOIRE 24 GR. 1 0 14 SETCOLOR
  2 0 0 SETCOLOR 318 0 DO 1 COLOR
  159 0 PLOT I 191 DRAWTO 3
  +LOOP 30000 0 DO LOOP ;
```

If you have QS Forth (and Ekkehard Floegel's book *Forth for the Atari*), note that the inputs to the SETCOLOR statement are reversed, while PLOT and DRAWTO have the COLOR statement embedded. Instead, do this:

```
: MOIRE 24 GR. 14 0 1 SETCOLOR
  0 0 2 SETCOLOR 318 0 DO 159 0
  1 PLOT I 191 1 DRAWTO 3
  +LOOP 30000 0 DO LOOP ;
```

This program provides a flashy display:

```
SCR # 6
 0 ( color display 6/30 )
 1 DECIMAL : DELAY 0 DO LOOP ;
 2 : CENTER
 3   712 C@ 710 C@ 712 C!
 4   709 C@ 710 C! 709 C! ;
 5 : FLASH 100 0 DO CENTER
 6   1000 DELAY LOOP ;
 7 : BACKGROUND 200 0 DO I 712 C!
 8   1000 DELAY 2 +LOOP ;
 9 : FOREGROUND 200 0 DO I 710 C!
10   1000 DELAY 2 +LOOP ;
11 : WELCOME 125 EMIT 10 10 POS.
12  ." Welcome to Forth "
13   FLASH BACKGROUND FOREGROUND
14   0 GR. ;
```

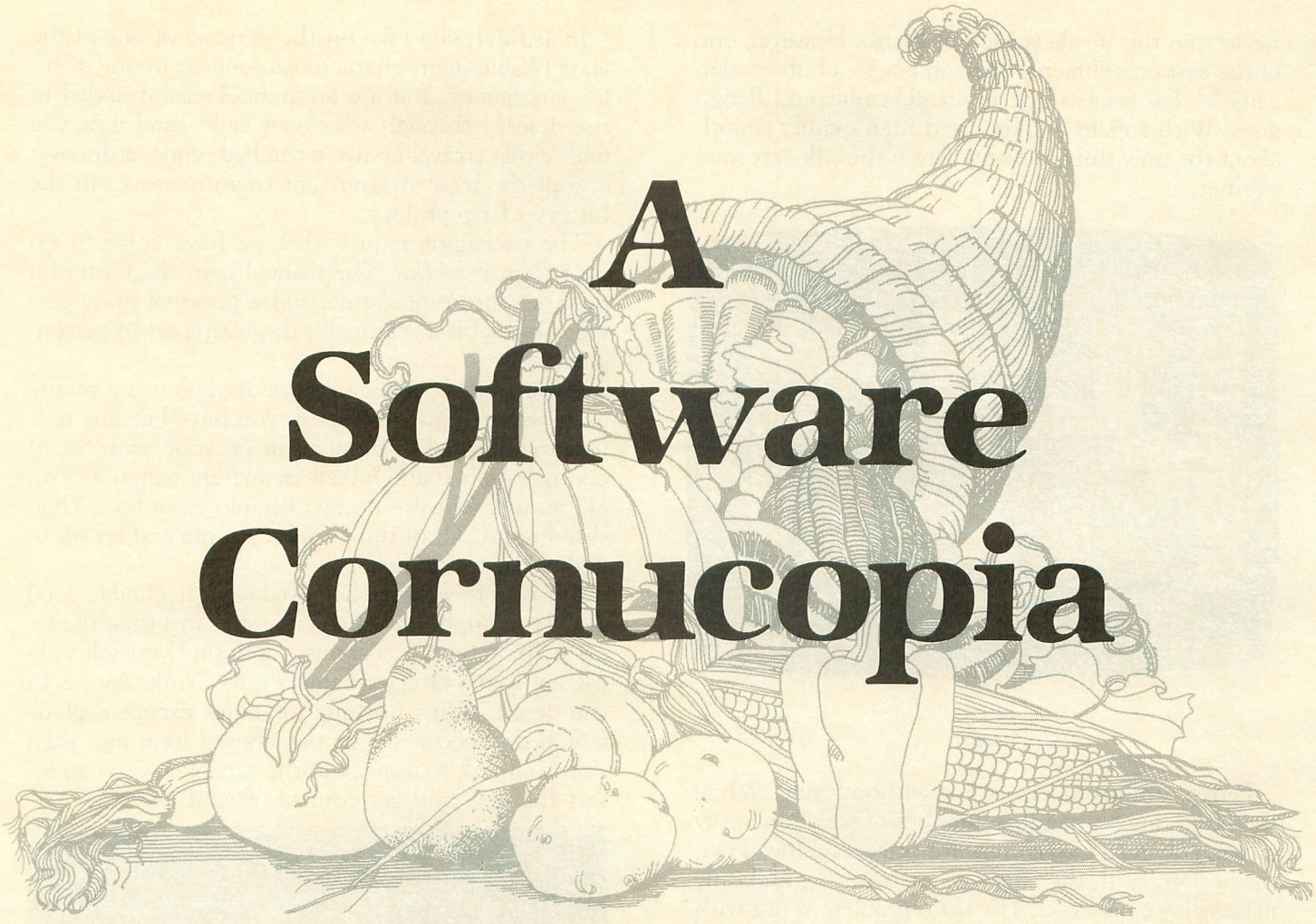
Be sure to tell your audience to watch for the next exciting installment in the continuing sage of the wonderful world of FORTH! □

Send letters to:

Ask Mr. Forth

P.O. Box 23

Worcester, MA 01603



A Software Cornucopia

by Steve Panak

Spring has again grown into summer, which in turn decayed into fall, prompting us to focus our attention on end-of-the-year events: lousy weather, school (for those of us *lucky* enough to still be attending) and, of course, the big one—Christmas. Here we'll take a look at a number of entertainment programs which may find themselves on Christmas lists throughout softwareland. So read carefully, then go out and take a look for yourself. Make that list, check it twice, and you'll end up with a pleasant holiday season rather than a miserable, monotonous one.

PENGO
ATARI, INC.
Sunnyvale, CA 94086
16K Cartridge $44.95

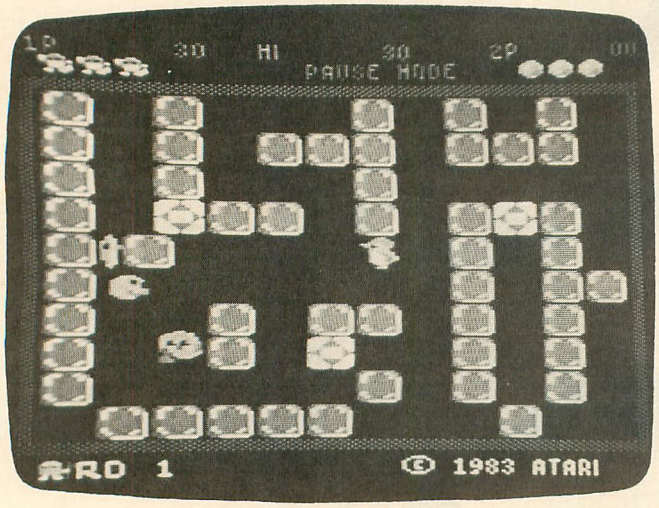
Pengo takes the player to an Antarctic wasteland, where he must do battle with the nasty Sno-bees in this **Pac-Man** derivative.

Pengo is a penguin fighting a never-ending battle to survive. As if it isn't hard enough to find food and keep warm, he is terrorized by creatures whose touch means certain death. But all is not hopeless; he has numerous methods by which to defeat these menaces. He can crush them with ice blocks which he kicks across the screen, or he can freeze and destroy them in one of two ways—he can kick the wall and freeze a Sno-bee against it, or he can line up the three diamond blocks for extra points, again freezing his foes in their tracks.

Once frozen, the tables turn; now **Pengo**'s touch is deadly to his helpless adversaries. But new Sno-bees are continuously hatching. Only by remembering just which cubes have eggs can **Pengo** destroy the creatures in their shells. Once a level has been cleared of all the Sno-bees, play continues with faster and smarter antagonists. Finishing a round in record time awards extra points and, at 30,000 points, a new life is obtained.

Pengo is an arcade conversion, licensed from Sega, and the inevitable question is: how does it compare to its coin-op counterpart? The problem I have is that I live in an area that the city-folk refer to as the country, and the country folk refer to as the boondocks; we have just graduated from **Space Invaders** (remember that one?) to **Pac-Man**. Well, it's not *quite* that bad, but we are anywhere from a year to two behind the rest of civilization. So, the point is that I have

never seen the arcade version of **Pengo**. However, one of the best compliments one can pay to a home video game is that it *looks* like an arcade game, and **Pengo** does. With superb graphics and high quality sound, about the only thing missing here is the silk-screened cabinet.



Pengo.

Pengo, unfortunately, is not without fault. While the program can handle a number of simultaneously moving objects, too many will noticeably slow down the action. Still, the effect is not irritating and only makes play a bit easier. The main problem is not with the program, but with the game concept itself. **Pengo** is just the same thing done over and over and *over* again, with no change in its scenery, strategy or substance. Unlike some games with various changing screens (**Ms. Pac-Man**, **Donkey Kong**, etc.), the action in **Pengo** only gets a little more intense, like the original **Pac-Man**. Perhaps I'm different from everyone else, but I need a little more. **Pac-Man** was great in its time (*how* many years ago?), but many a **Pac-Man** machine now lies dormant in favor of those with more variety. **Pengo** provides no incentive to continue, except to get *the* high score. For me, that is simply not enough.

INFIDEL
by Michael Berlyn
INFOCOM, INC.
55 Wheeler Street
Cambridge, MA 02138
32K Disk $42.95

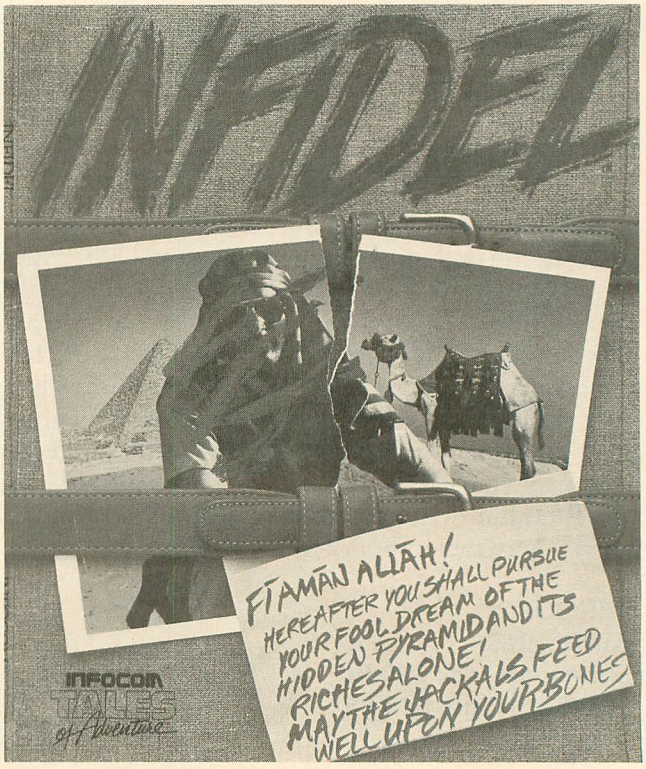
Infidel (*n.* a disbeliever in something specified or understood) is Infocom's first entry in its newest series, **Tales of Adventure**. While their previous voyages have taken us to distant galaxies or faraway magical lands, **Infidel** is set in Egypt, and so, in that respect, it's down to earth. But the quality of this game is nothing less than out of this world.

In **Infidel**, you take on the persona of one of the least likable main characters to appear in any of Infocom's games. You are an archeologist stranded in the desert—through your own fault—and now you must either shrivel up like a parched prune or discover one of the most magnificent treasure troves in the history of Egyptology.

The packaging is just what we have come to expect from Infocom. The manual is in the form of a pulp adventure magazine, and a personal diary provides you with a reasonably disgusting set of personality traits.

It seems that, after a rather disappointing tenure as an archeologist's assistant, you have become restless and decided to branch out on your own. So, of course, you lied and bilked an ancient map out of an old spinster who was trying to employ your boss. Then you dumped all of those moral people and set off to grab it all for yourself.

But your greed overshadowed what negligible good sense you might have had and, after insulting the local help's religious customs, you found yourself waking up from a drugged camel's milk drink. Alone. In the desert. With nothing for miles except a plane which dropped a single, parachuted item and then slowly shrank to a speck in the sky. Now you can insert the disk and get control of your life.



Infidel.

In the first phase of the game, you must survive to find the pyramid. This is really not a very difficult task, unless you forget an important item from your camp—or run out of food, water or time. Your greatest challenges await you in the great pyramid's chambers.

Each area contains an ancient puzzle, issued eons ago. **Infidel**'s solutions do not rely on magical items or spells (but there may be a curse or two; no self-respecting pyramid would be without one). Each trap seems as if it may have been designed by the pyramid's builders to prevent looting, and they're all reminiscent of the *Raiders* films of late. Careful reasoning and observation will allow you to solve these problems. But there are also clues.

On the walls of the mighty pyramid are hieroglyphics. If these are deciphered, they offer valuable clues to aid you in your quest for riches. Interpreting these is a relatively good exercise in its own right. Though solving them is not essential to finish, they are an extra which provides the game with depth. Write down all the marks you can find, then translate them using traditional code-breaking methods (i.e., spotting common words). This is another challenge you accept when playing **Infidel**.

All of the basic Infocom program features are present; SAVE game, as always, is a must for the weary adventurer. In fact, this product is, overall, just what we have come to expect from the people at Infocom —high quality entertainment, imitated but never equalled. **Infidel** is a worthy addition to their software library, and to yours.

MR. ROBOT and his ROBOT *FACTORY*
DATAMOST
9748 Cozycroft Avenue
Chatsworth, CA 91311
(213) 366-7160
48K Disk $34.95

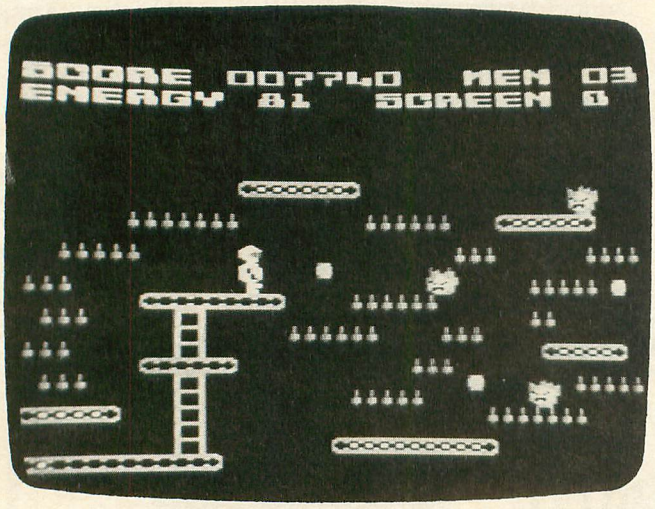
Mr. Robot and his Robot Factory is a game which supports the old proverb warning us never to judge a book by its cover. A rather spartan container holds what, surprisingly, turns out to be a relatively decent game. It consists of two parts: **Mr. Robot** (the game) and the **Robot Factory** (a program which allows you to custom design your own playing screens).

Mr. Robot is a true hybrid game, with themes borrowed from many predecessors. You must clear dots (**Pac-Man**) from an irregular, ever-changing screen of ladders, poles and other pathways (**Donkey Kong Junior**), while jumping over or destroying alien fireballs (**Donkey Kong**). An energizer enables you to successfully defeat these fireballs. However, there are also original (I *think*) concepts here, as well.

Some screens contain bombs you must cross over. Stepping on them ignites their fuses, which burn for a short time before the devices explode. If you happen to be *on* one of them when this happens, **Mr. Robot** is reduced to Mr. Junk.

The bombs are also very effective against fireballs. Trampolines allow you to survive longer falls; magnets, when touched, divert your vertical fall either to the left or the right. Transporters allow you to move around the screen—just like Mr. Spock. Extra lives are earned, not by exceeding a stated score, but by touching life tokens scattered on various levels.

Overall, the game is okay. The graphics are mixed. While most of the obstacles are fairly detailed (the fuses on the bombs burn nicely), **Mr. Robot** himself seems only slightly above the graphic capability of the old 2600. There are twenty-two screens to challenge you, and you can start at any level you wish. Music (by Paradise, whatever or whoever that is) supplies background sound at least as good as any Top 40 radio station. The real gimmick is the **Factory**.



Mr. Robot

By using the **Robot Factory**, you can design your own screens and store up to twenty-six of them, to play in your own order. Using a method much like that in **Pinball Construction Set**, you select components via the joystick and move them to any desired position on the screen. Then you can test, play, store and edit them, until you have your own ultimate game.

This feature is well done but does have a few faults. The disk controls are directly beneath the game components, so, if you happen not to be paying close attention, you are likely to move the cursor over one of these and erase your current creation. Also, they seem to have forgotten to place a "free life" token among the components. Thus, you're unable to incorporate this feature in any of your screens.

Documentation is shabby, drab and colorless, and will probably stop any potential buyers at the showcase. My recommendation is that you go further. Ask to load the game and play it a couple of times. Look at all the screens and feel the power it gives you.

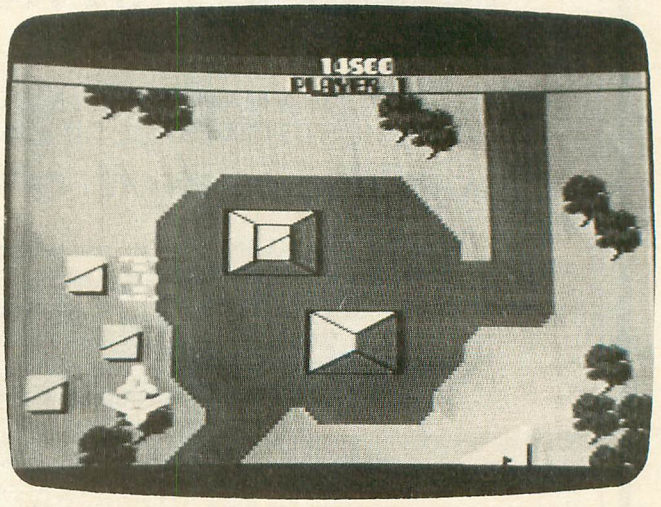
While **Mr. Robot** may not be a welcome addition to a library full of other, similar arcade games, it may be the best buy for a first game. . .and it's a must for anyone who frequently mutters, "I could have designed that game better."

FLAK
FUNSOFT (IJG, INC.)
1953 W. 11th Street
Upland, CA 91786
32K Disk $34.95

Flak disputes the theory (you can't judge a book by its cover) supported by **Mr. Robot**; it turns out to be a disappointing endeavor to copy a superb arcade game. Again we see a package that is somewhat less than physically attractive. Being of a curious and open-minded nature, we open it up to take a look.

Surprise! **Flak** is a cheap attempt to rip off the arcade game **Xevious**. I say "cheap" because it captures none of the excitement of the original—it's not cheap, in the true financial sense, at $34.95. And I call it an "attempt" because it looks half-finished. **Flak** could be the worst mistake your wallet ever made; avoid it like radioactive waste.

In **Xevious**, you swooped in low over magnificent cartoon-like graphics, using air-to-ground bombs to decimate oncoming fighters, saucers, bunkers and tanks, on your way to a climactic encounter with the monster mother ship. In **Flak**, you fly over adequate vertically-scrolling landscape, firing upon land bases only, on your journey to do battle with the mad CPU (give me a break). It seems as though someone made a start at copying **Xevious** and, for whatever reason (copyright, time, money, expertise) found they could not do it. . .but they sold the partially finished product, anyway.



Flak.

There are more problems. **Flak** places you, when destroyed, near your last position in play. However, this often results in your being right above a firing bunker. Poof! Without a chance, you're a pulsating fireball (that's my imagination; the graphics aren't that good).

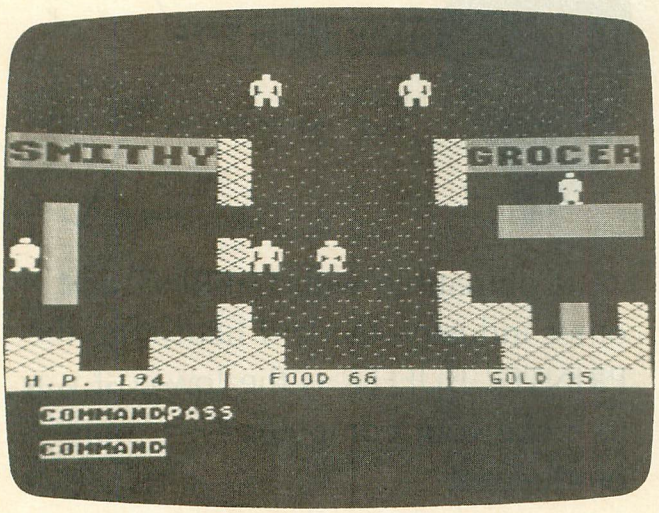
The documentation is poor, in the truest sense of the word. The manual looks almost photocopied. If that isn't enough of a curse, the book then makes a pitiful effort to create an interesting background history for the game. Again, **Flak** fails miserably.

As if you'd want one, it gives you an extra life every 10,000 points; fortunately, I rarely reached that level of expertise. The sound effects are barely above those of the earliest **Pong** game, and they affect the human nervous system in a manner similar to new chalk scraped down a blackboard.

Flak is truly a book to judge by its cover. . .and a good argument in favor of book burning. When the best thing you can say about a game is that it has adequate graphics, you know it's in trouble. My sympathy to anyone who's stuck with this dog.

QUESTRON
STRATEGIC SIMULATIONS, INC.
883 Stierlin Road
Mountain View, CA 94043
48K Disk $49.95

Questron is a meaningless Xerox copy of **Ultima**, with a few new twists scattered throughout. However, it does cover well-worn ground in a competent way, and would make a fine first and/or only addition of a role-playing game to your software library.



Questron.

In **Questron**, you start the game as a lowly serf. The only characteristic you can control is your name, while the computer assigns the familiar D&D attributes of strength, intelligence, etc. Then, it's off to travel, battle, plunder, loot, kill and maim in the quest for fame and fortune. Over fine-scrolling graphics, you travel to cathedrals, towns, ports and dungeons. You pray to be healed, buy and sell goods, and search for treasures. An amusing addition to this formula is the gambling parlor. These are found in many of the towns and allow you to quickly increase (or, more likely, lose) your gold holdings.

The graphics are nearly identical to those in **Ultima**: top view maps of landscapes, towns and buildings; first person point of view line vector graphics in the dungeons. All graphics are good, although not very original—the biggest problem with this game.

We've seen all this before. We need another **Ultima** no more than we need another **Pac-Man**. With no unique quest in mind, plot development only in the form of short sentences (usually the same in every town) and endless battles, the game is like a Twinkie without the creme filling—an attractive exterior hiding absolutely nothing.

Another irritant is the number of disk swaps necessary to play **Questron**. While I don't mind waiting for my slow 810 to feed my starving 800 the required data, continually changing disks drives me into a frenzy. Normally, the solution is two drives (for those of us who have the cash reserves), but the manual says *nothing* about two-disk support. A call to the SSI technical hotline netted me the response that only one drive could be used with the Atari version of **Questron**. Those with two drives are as powerless here as we who have only one.

But **Questron** is well done and shows a lot of effort. It only fails for me because I have done *it all* before, many times. **Ultima** excited me, spurred me to finish it in record time—a dollar (or more) well spent. **Ultima II** I never finished. I had quested to destroy evil once before. Without more filler to create a more real universe, I quickly lost interest. Even with the extensive documentation that supplies the game with a history and describes its monsters and attributes, the voyage was shallow for me. . .like wading into the water but never having to battle to keep my head above the waves.

Still, I feel no qualms about recommending **Questron** as a first adventure of this type. If you are considering it as your second or third adventure and are looking for something different, then (at $49.95) it would serve you well to examine **Questron** closely before buying.

In retrospect.

We have looked at a number of games here, all of different types. From the superb prose of **Infidel** to the abysmal machine language mess called **Flak**, we've observed the best and worst characteristics in current software.

Using this as a guide, go out and look for yourself, digging into and scrutinizing the disks, cassettes and cartridges in search of real quality. Only through careful purchasing habits can we avoid the unpleasant feeling we all know—that of wasting our time and money on a worthless bit of rubbish.

The author would like to thank Perfect Computers of Nile, Ohio, for valuable assistance in the creation of this article.

NEW!

PRINTER ACCESSORIES FROM

DIGITAL DEVICES T.M.

U•PRINT T.M.

Expand your Atari® or Commodore® computer with Digital Devices *U • PRINT*. We make it simple to add any printer you choose. *U • PRINT* interfaces feature industry standard Centronics parallel connectors to hook up an Epson, Star, NEC, C.Itoh, Okidata, or any other printer.

- EXTRA SERIAL PORT FOR DAISY CHAINING OTHER PERIPHERALS.
- COMPATIBLE WITH ALL ATARI HARDWARE AND SOFTWARE.

U • PRINT MODEL A

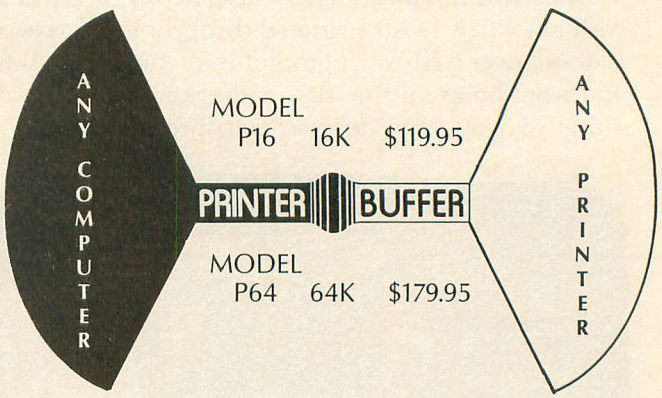
- EMULATION OF COMMODORE PRINTERS, INCLUDING GRAPHICS.
- COMPATIBLE WITH ALL COMMODORE HARDWARE AND SOFTWARE.

U • PRINT MODEL C

Compact, easy to install, and costing only $89.95, U•PRINT gives you a choice!

PRINTER BUFFER T.M.

Ever get stuck while your printer catches up? The PRINTER BUFFER eliminates waiting by rapidly accepting data in memory, then relaying it at the printer's rate, freeing the computer for your next job. User-upgradable memory (16k to 64k) allows up to 32 pages of data to be stored.

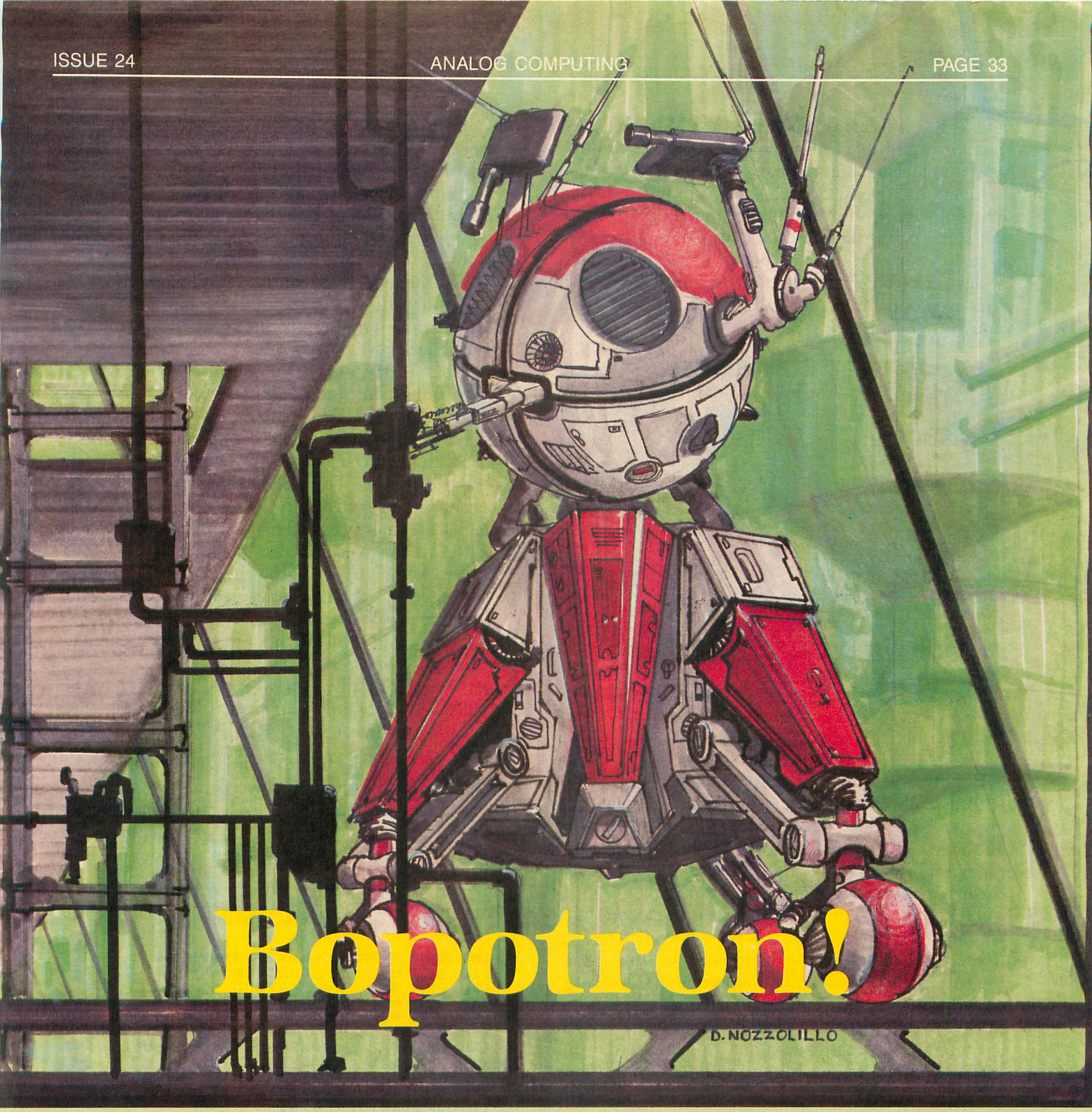


Compatible with U•PRINT and other industry-standard hardware, the PRINTER BUFFER is the low-cost way to make your computer even more productive!

Quality Products Made In USA From

DIGITAL DEVICES Corporation

430 Tenth Street, Suite N205 Atlanta, Georgia 30318
In Georgia (404) 872-4430; Outside Georgia (800) 554-4898



Bopotron!

16K Cassette or 24K Disk

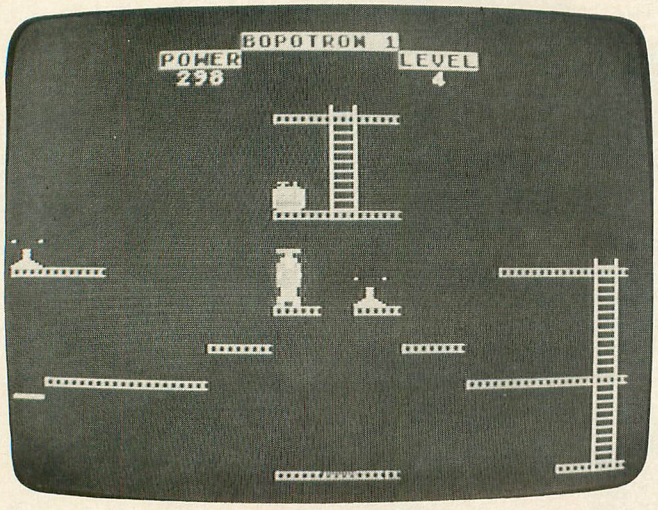
by Kyle Peacock

Mankind has finally abandoned its futile efforts to establish the nuclear superiority of any one nation. The major powers have joined together for the sake of scientific advancement rather than political dominance. Space exploration has become the major interest of today's society. Unfortunately, the fantasies of warp drive and speed-of-light travel are still unattainable ideals. Cryogenics and patience are the only feasible methods of reaching the stars.

You are a small worker, **Bopotron**, on the starship Quab IV. While the crew lies in suspended animation, your job is to handle minor maintenance and repairs. It's just your luck that Ted, the last human to enter the cryogenic vault, left the ship in an uproar. The control platforms that allow **Bopotrons** to move about are on full automatic, and many of the ship's power units are on constant drain.

Being a noble **Bopotron**, you set out to perform

a task that will last through the next four centuries. You are responsible for the recharging of the drained power units. Each unit requires 100 EUs (energy units) to function properly. Charging a unit drains your own internal supply, so you'll have to juice up at one of the ship's power packs periodically. Should you fail to keep your own internal power above zero, you'll no longer be able to function, the ship will go dead and everyone on board will be iced. *(Aren't they already? —Ed.)*



Bopotron.

Bopotron is a one-player game written in BASIC, with assembly language subroutines. The game begins with the player receiving five lives to complete all the levels. This magazine version has only five levels. Future levels can be added via the use of the **Bopotron Construction Set** (see page 55).

The screen layout for **Bopotron** is pretty straightforward. Girders and ladders are safe areas to bop (travel) on. Power packs are represented by small, glowing batteries with plus and minus terminals. To charge up, all you must do is stand on either side of a power pack. Power units more or less resemble television antennas. Standing next to one will discharge your internal power supply while charging the unit. Once a unit is fully charged, it will glow rapidly. Fully charged units require zero maintenance and expend none of your energy. Once you've charged up all of the power units on a particular level, you can leave the level by bopping onto the exit girder (a small, glowing girder segment somewhere on the screen).

The framework of the Quab IV is non-contiguous; not all of the ladders and girders connect. For this reason, the Quab engineers have provided maintenance platforms to aid the **Bopotrons**. These platforms travel in a pre-programmed manner at regular intervals. Whenever you bop onto a platform, it will carry your **Bopotron** along its pre-programmed vector. The use of such platforms is essential for completing your task.

Although mechanical, a **Bopotron** is extremely fragile. It will be destroyed under one of two conditions: (1) the **Bopotron** falls a long distance off of a girder, ladder or platform; or (2) the **Bopotron**'s brain bubble (located at the top of its head) is struck by a girder or platform. Should one of these conditions arise, it's bye-bye **Bopotron**.

Although pretty much an "as is" game, **Bopotron** does allow for slight variations. Examination of Line 160 in the program listing reveals the following variables:

MAXLIFE — This is the maximum number of **Bopotron** lives per game. No extra lives are awarded. This value should be kept below ten.

MAXFALL — This is the maximum number of character segments the **Bopotron** can fall before dying. The height of one girder is four character segments. So, as an example, if MAXFALL equals twenty-four, your **Bopotron** can fall the height of six girders before dying.

MAXLEVEL — This is the maximum number of levels in the game. As levels are added, this number would increase. Those with only 16K cannot add levels without first deleting already existing ones.

STARTLEVEL — This is the starting level of the game. Should you wish to skip levels one, two and three, set STARTLEVEL equal to four.

Once again, I'd like to thank Tom Hudson for his assistance on **Bopotron**. Levels one, two and five are of my own design, while levels three and four were concocted by Tom. You just can't keep a good programming team down. Keep on boppin'! □

BASIC listing.

```
100 REM ************************
110 REM *      BOPOTRON        *
120 REM *    KYLE PEACOCK      *
130 REM *  ANALOG COMPUTING    *
140 REM ************************
150 REM
160 MAXLIFE=5:MAXFALL=11:MAXLEVEL=5:ST
ARTLVL=1:LVL=STARTLVL
170 READ N1,N2,N3,N4,N5,N6,N7,N8,N9,N0
180 DIM VBLANK$(723),INIT$(103),DLI$(1
2),BOP$(30):LIFE=N1
190 POKE 106,PEEK(106)-N4:CSET=PEEK(10
6)*256:GRAPHICS 0:POKE 1536,CSET/256:P
OKE 54286,192:POKE 752,N1:GOSUB 610
200 VBLANK=ADR(VBLANK$):BOP=ADR(BOP$):
INIT=ADR(INIT$)
210 FOR X=N1 TO 723:READ N:VBLANK$(X)=
CHR$(N):NEXT X
220 FOR X=N1 TO 103:READ N:INIT$(X)=CH
R$(N):NEXT X
230 FOR X=N1 TO 12:READ A:DLI$(X)=CHR$
(A):NEXT X:X=ADR(DLI$):HIGH=INT(X/256)
:LOW=X-HIGH*256
240 POKE 512,LOW:POKE 513,HIGH
250 FOR X=N1 TO 30:READ N:BOP$(X)=CHR$
(N):NEXT X
260 PMBASE=INT((PEEK(145)+N3)/N4)*N4:P
OKE 54279,PMBASE:PMB=PMBASE*256:POKE 5
59,46:POKE 53277,N3:POKE 623,33
270 FOR X=K0 TO 127:READ A:POKE CSET+N
8+X,A:NEXT X
```

```
280 DLIST=PEEK(561)*256+PEEK(560):POKE
 DLIST+N8,N2+128:FOR X=DLIST+N9 TO DLI
ST+28:POKE X,N4:NEXT X
290 K=INT(RND(N1)*16):SETCOLOR N2,K,N2
:SETCOLOR N4,K,N2:FOR X=K0 TO N3:SOUND
 X,K0,K0,K0:NEXT X
300 REM
310 REM *** CONSTRUCT BOARD ***
320 REM
330 GOSUB 4000:GOSUB 5000:GOSUB 6000:G
OSUB 7000:GOSUB 8000:GOSUB 2000:GOSUB
3000:POKE 54286,192
340 GOSUB 610:POKE 1537,K0
350 REM
360 REM *** GET THINGS GOING
370 REM
380 POSITION N0,N2:? ENG;" ";:POKE 77,
K0
390 V=PEEK(1537):CHAR=PEEK(1612):TRIG=
STRIG(K0)
400 IF V=K0 AND ENG>K0 THEN 440
410 POKE 1537,N1:FOR X=K0 TO 255 STEP
N5:SOUND N1,X,N8,N0
420 K=INT(RND(N1)*256):T=INT(RND(N1)*N
3):SETCOLOR N2,K,K:SETCOLOR N4,K,K:POK
E 53256,T:POKE 53257,T:NEXT X
430 SOUND K0,K0,K0,K0:SOUND N1,K0,K0,K
0:LVL=LVL-N1:LIFE=LIFE+N1:GOTO 580
440 SOUND K0,200-100*(STICK(0)<>15)-50
*(TRIG=K0),N6,N8
450 IF CHAR<12 OR CHAR>13 THEN 530
460 X=INT((PEEK(1541)-45)/N4):Y=INT((P
EEK(1545)-N6)/N4):SOUND K0,K0,K0,K0
470 FOR R=255 TO K0 STEP -N5:SOUND N2,
R,N0,N0:SOUND N3,R+N1,N0,N0:ENG=ENG-N2
480 POSITION N0,N2:? ENG;" ";:T=PEEK(1
612):IF T<12 OR T>13 THEN 520
490 NEXT R
500 POSITION X-(N1*(CHAR=13))+(N1*(CHA
R=12)),Y:? "■";
510 POSITION X-(N1*(CHAR=13))+(N1*(CHA
R=12)),Y-N1:? "■";:ACTIVE=ACTIVE-N1
520 SOUND N2,K0,K0,K0:SOUND N3,K0,K0,K
0
530 IF CHAR<136 OR CHAR>137 THEN 550
540 IF ENG<=MAXENG THEN ENG=ENG+N0:FOR
 X=15 TO K0 STEP -N2:SOUND N1,200,N0,X
:NEXT X:GOTO 560
550 ENG=ENG-N1-N2*(TRIG=K0)
560 IF CHAR<>144 OR ACTIVE<>K0 THEN 38
0
570 POKE 1537,N1:FOR T=K0 TO N4:FOR X=
K0 TO 255 STEP 20:SOUND K0,X,N0,N0:SOU
ND N1,X+N1,N0,N0:NEXT X:NEXT T
580 POKE 1537,N1:FOR X=K0 TO N3:SOUND
X,K0,K0,K0:POKE 53248+X,K0:NEXT X:GOSU
B 660
590 LVL=LVL+N1*(LVL<>MAXLEVEL):IF LIFE
>MAXLIFE THEN 630
600 GOTO 290
610 POSITION 14,0:? "■BOPOTRON■";CHR$(L
IFE+176):POSITION N9,N1:? "■POWER■":POSI
TION 24,N1:? "■LEVEL■"
620 POSITION 26,N2:? LVL;:RETURN
630 LIFE=N3:GOSUB 610:POSITION N1,N3:?
 "GAME OVER - PRESS BUTTON TO PLAY AGA
IN":SETCOLOR N2,K0,K0
640 IF STRIG(K0) THEN 640
650 GOSUB 660:LVL=STARTLVL:LIFE=N1:GOT
O 290
660 POSITION K0,N2:FOR X=K0 TO 24:? "■
";:NEXT X:RETURN
1000 REM
1010 REM *** CONSTANTS
1020 REM
1030 DATA 1,2,3,4,5,6,7,8,9,10
1040 REM
1050 REM *** VERTICAL BLANK ROUTINE
1060 REM
1070 DATA 216,238,199,2,173,1,6,240,3,
76,98,228,165,203,72,165,204,72,160,0,
162,0,189,5,6,157,0,208
1080 DATA 189,9,6,141,2,6,24,113,205,1
41,3,6,200,140,4,6,224,0,240,13,165,20
3,24,105,128,133,203,165
1090 DATA 204,105,0,133,204,173,2,6,56
,233,10,168,169,0,145,203,200,204,2,6,
144,248,173,3,6,24,105,10
1100 DATA 168,169,0,145,203,136,204,3,
6,176,248,188,13,6,16,24,188,15,6,189,
9,6,56,249,45,6,201,11
1110 DATA 144,10,201,245,176,6,169,0,2
40,7,144,154,172,4,6,177,205,172,2,6,1
45,203,238,2,6,238,4,6
1120 DATA 173,2,6,205,3,6,208,203,189,
13,6,16,29,169,0,157,13,6,224,2,144,20
,188,15,6,185,35,6
1130 DATA 157,5,6,185,45,6,157,9,6,169
,15,157,17,6,172,4,6,232,224,4,144,188
,104,133,204,104,133,203
1140 DATA 174,78,6,48,58,222,19,6,208,
53,189,15,6,240,7,169,255,157,15,6,48,
41,188,17,6,185,25,6
1150 DATA 157,19,6,189,7,6,217,55,6,20
8,59,189,11,6,217,65,6,208,51,169,1,15
7,15,6,169,15,157,19
1160 DATA 6,16,8,48,6,16,198,240,84,20
8,82,254,17,6,189,17,6,201,5,208,4,169
,0,240,6,201,10,208
1170 DATA 2,169,5,157,17,6,168,185,25,
6,240,227,16,51,189,7,6,24,125,21,6,15
7,7,6,189,11,6,24
1180 DATA 125,23,6,157,11,6,236,81,6,2
08,26,173,5,6,24,125,21,6,141,5,6,141,
6,6,173,9,6,24
1190 DATA 125,23,6,141,9,6,141,10,6,20
2,16,165,169,255,141,81,6,174,78,6,48,
90,173,5,6,56,253,7
1200 DATA 6,16,5,73,255,24,105,1,201,7
,176,23,189,11,6,56,237,9,6,201,12,208
,5,142,81,6,208,7
1210 DATA 176,5,169,1,141,1,6,169,0,15
7,21,6,157,23,6,188,17,6,189,7,6,217,5
5,6,144,6,240,9
1220 DATA 169,255,48,2,169,1,157,21,6,
189,11,6,217,65,6,144,6,240,9,169,255,
48,2,169,1,157,23,6
1230 DATA 202,16,163,165,88,133,207,16
5,89,133,208,173,9,6,56,233,6,74,74,17
0,240,16,165,207,24,105,40,133
1240 DATA 207,165,208,105,0,133,208,20
2,16,238,173,5,6,56,233,44,74,74,168,1
77,207,174,81,6,48,2,169,1
1250 DATA 141,76,6,41,127,141,75,6,173
,9,6,201,27,144,23,165,207,56,233,80,1
33,207,165,208,233,0,133,208
1260 DATA 177,207,201,1,240,109,201,8,
176,105,173,75,6,240,48,206,77,6,48,3,
76,98,228,72,169,1,174,16
1270 DATA 208,240,1,10,141,77,6,104,20
1,2,144,4,201,8,144,26,173,81,6,16,53,
173,9,6,41,1,208,7
1280 DATA 173,9,6,41,3,208,39,238,82,6
,169,1,208,22,173,120,2,201,13,208,9,1
73,9,6,201,98,208,235
1290 DATA 240,16,201,14,208,12,169,255
,24,109,9,6,141,9,6,141,10,6,173,75,6,
240,81,173,82,6,205,83
1300 DATA 6,144,5,169,1,141,1,6,169,0,
141,82,6,173,120,2,201,7,208,22,173,5,
6,201,198,176,49,173
1310 DATA 75,6,201,8,240,42,201,12,240
,38,169,1,208,24,201,11,208,30,173,5,6
,201,47,144,23,173,75,6
1320 DATA 201,9,240,16,201,13,240,12,1
69,255,24,109,5,6,141,5,6,141,6,6,76,9
8,228
1330 REM * 723 BYTES
1340 REM
1350 REM *** VBLANK INITIALIZER
1360 REM
1370 DATA 216,165,16,41,127,133,16,141
,14,210,104,104,141,79,6,104,141,80,6,
104,133,204,104,133,203,104
1380 DATA 133,206,104,133,205,169,1,14
1,1,6,165,204,72,162,3,160,0,152,145,2
03,200,208,251,230,204,202
1390 DATA 16,246,162,9,157,25,6,224,4,
176,3,157,13,6,202,16,243,141,82,6,141
,17,6,141,19,6
1400 DATA 141,20,6,142,78,6,169,5,141,
18,6,104,133,204,174,79,6,172,80,6,169
,7,76,92,228
1410 REM * 103 BYTES
1420 REM
1430 REM *** DLI ROUTINE
```

```
1440 REM
1450 DATA 72,173,0,6,141,10,212
1460 DATA 141,9,212,104,64
1470 REM
1480 REM *** PLAYER IMAGE DATA ***
1490 REM
1500 DATA 12,126,0,60,126,126,126,126
1510 DATA 126,0,0,0,102
1520 DATA 12,0,24,60,24,0,66,0,0,60
1530 DATA 60,126,0
1540 DATA 1,255
1550 DATA 1,255
1560 REM
1570 REM *** CHARACTER DATA
1580 REM
1590 DATA 0,0,0,0,85,68,68,85
1600 DATA 8,8,8,10,89,72,72,90
1610 DATA 0,0,0,170,85,68,68,170
1620 DATA 128,128,128,128,149,132,132,
149
1630 DATA 8,8,8,10,8,8,8,10
1640 DATA 0,0,0,170,0,0,0,170
1650 DATA 128,128,128,128,128,128,128,
128
1660 DATA 191,191,170,42,85,68,68,85
1670 DATA 254,254,170,168,85,68,68,85
1680 DATA 63,12,21,170,170,150,170,170
1690 DATA 48,48,84,170,154,86,154,170
1700 DATA 2,10,42,170,85,68,68,85
1710 DATA 128,160,168,170,85,68,68,85
1720 DATA 128,60,12,15,2,61,2,253
1730 DATA 2,60,48,240,128,124,128,127
1740 DATA 0,0,0,0,255,204,204,255
2000 REM
2010 REM *** SET UP BOPOTRON
2020 REM
2030 POKE 704,102:POKE 53256,K0
2040 POKE 705,136:POKE 53257,K0
2050 POKE 706,151:POKE 707,231:POKE 16
19,MAXFALL
2060 RESTORE 2080+(20*LVL)
2070 READ X,Y,ENG,MAXENG:POKE 1541,X*N
4+48:POKE 1542,X*N4+48:POKE 1545,Y*N4+
N6:POKE 1546,Y*N4+N6
2080 RETURN
2100 DATA 0,4,500,500
2120 DATA 37,23,200,500
2140 DATA 0,12,300,500
2160 DATA 16,13,300,500
2180 DATA 37,23,300,650
3000 REM
3010 REM *** 'EXIT' GIRDER
3020 REM
3030 RESTORE 3080+(20*LVL)
3040 READ X,Y:POSITION X,Y:? "[0]";:RETU
RN
3100 DATA 0,4
3120 DATA 0,23
3140 DATA 38,5
3160 DATA 23,23
3180 DATA 35,5
4000 REM
4010 REM *** FLOOR DRAW
4020 REM
4030 RESTORE 4080+(20*LVL)
4040 READ AMOUNT:FOR T=N1 TO AMOUNT:RE
AD STRT,END,YPOS:FOR X=STRT TO END:POS
ITION X,YPOS:? "!";:NEXT X:NEXT T
4050 RETURN
4100 DATA 9,0,8,4,17,26,4,3,13,9,16,23
,9,30,35,9,7,13,13,16,35,13,7,30,18,0,
38,23
4120 DATA 9,7,15,7,18,23,7,14,15,12,18
,19,12,6,10,16,23,28,16,35,36,16,2,5,1
9,0
4121 DATA 38,23
4140 DATA 8,0,5,5,0,5,13,0,5,23,34,38,
5,34,38,13,34,38,23,12,19,9,22,27,9
4160 DATA 12,16,23,4,16,23,9,0,5,12,30
,38,12,16,18,14,21,23,14,12,15,16,24,2
7,16,2
4161 DATA 11,18,28,38,18,34,38,23,16,2
3,23
4180 DATA 20,22,27,5,7,13,8,16,17,8,18
,19,9,20,21,10,30,35,10,22,23,11,24,25
,12,5
4181 DATA 13,13,26,27,13,28,29,15,30,3
1,16,32,33,17,34,35,18,20,21,19,18,19,
20,11,17
4182 DATA 21,9,10,22,2,8,23,32,38,23
5000 REM
5010 REM *** LADDER DRAW
5020 REM
5030 RESTORE 5080+(20*LVL)
5040 READ AMOUNT:FOR T=N1 TO AMOUNT:RE
AD STRT,END,XPOS:FOR Y=STRT TO END:POS
ITION XPOS,Y:? "%&";:NEXT Y
5050 POSITION XPOS,STRT:? CHR$(34);"#$
";:POSITION XPOS,END:? CHR$(34);"#$";:
NEXT T
5060 RETURN
5100 DATA 4,4,9,27,9,13,36,13,18,4,18,
23,31
5120 DATA 4,12,16,11,12,16,20,19,23,6,
16,23,29
5140 DATA 2,13,23,1,13,23,36
5160 DATA 3,4,9,19,18,23,36,12,18,36
5180 DATA 3,5,16,36,18,23,36,8,13,2
6000 REM
6010 REM *** BATTERY DRAW
6020 REM
6030 RESTORE 6080+(20*LVL)
6040 READ AMOUNT:FOR T=N1 TO AMOUNT:RE
AD XPOS,YPOS:POSITION XPOS,YPOS:? "[0]"
;
6050 POSITION XPOS,YPOS-N1:? "[3]";:NEX
T T:RETURN
6100 DATA 1,0,23
6120 DATA 1,14,23
6140 DATA 1,34,13
6160 DATA 1,16,9
6180 DATA 1,0,23
7000 REM
7010 REM *** POWER UNIT DRAW
7020 REM
7030 RESTORE 7080+(20*LVL)
7040 READ ACTIVE:FOR T=N1 TO ACTIVE:RE
AD XPOS,YPOS:POSITION XPOS,YPOS:? ",-"
;
7050 POSITION XPOS,YPOS-N1:? "./";:NEX
T T:RETURN
7100 DATA 2,3,9,23,9
7120 DATA 2,4,16,37,16
7140 DATA 2,0,5,12,9
7160 DATA 2,0,12,21,14
7180 DATA 4,20,5,0,8,8,13,10,13
8000 REM
8010 REM *** PLATFORM PROGRAMMING
8020 REM
8030 A=USR(INIT,VBLANK,PMB+512,BOP)
8040 RESTORE 8080+(20*LVL)
8050 READ NUMPLAT:FOR A=N1 TO NUMPLAT:
ADD=K0+N5*(A>N1):READ NUMVEC:FOR B=N1
TO NUMVEC
8060 READ STARTX,STARTY,XEND,YEND,SPEE
D:STARTX=STARTX*N4+48:STARTY=STARTY*N4
+18:XEND=XEND*N4+48:YEND=YEND*N4+18
8070 IF B=N1 THEN POKE 1541+N1+A,START
X:POKE 1545+N1+A,STARTY
8080 POKE 1561+ADD,SPEED:POKE 1571+ADD
,STARTX:POKE 1581+ADD,STARTY:POKE 1591
+ADD,XEND:POKE 1601+ADD,YEND
8090 ADD=ADD+N1:NEXT B:POKE 1614,NUMPL
AT-N1:NEXT A:RETURN
8100 DATA 2,3,14,17,14,9,2,14,9,14,13,
1,14,13,14,17,1,2,9,4,15,4,1,15,4,9,4,
3
8120 DATA 2,4,16,23,16,4,3,24,7,34,23,
2,16,4,16,23,3,24,7,34,23,4,4,7,12
8121 DATA 24,12,2,5,7,0,17,1,24,12,7,1
2,2,5,7,0,17,1
8140 DATA 2,5,18,9,18,5,2,18,5,6,5,2,6
,5,6,23,2,6,23,18,23,2,18,23,18,23,1,5
,20,23
8141 DATA 32,23,2,32,23,32,5,2,32,5,20
,5,2,20,5,20,9,2,20,23,20,23,1
8160 DATA 2,3,0,19,0,23,1,0,23,32,23,1
,28,13,24,13,2,4,19,23
8161 DATA 19,14,1,37,12,24,4,2,14,4,6,
12,2,6,12,6,18,2
8180 DATA 2,4,30,23,14,18,2,5,8,5,8,3,
5,8,5,14,2,28,14,28,5,2,3,8,23
8181 DATA 14,8,2,14,8,14,8,3,28,5,34,1
1,2
```

CHECKSUM DATA.

(see page 90)

```
100 DATA 35,417,588,935,47,89,580,742,
892,697,20,988,307,835,973,8145
250 DATA 242,395,500,710,508,78,180,84
,545,250,93,892,99,323,34,4933
400 DATA 80,149,420,691,611,513,927,63
8,552,779,800,388,257,617,615,8037
550 DATA 645,84,648,336,621,722,256,67
1,886,399,182,776,275,130,277,6908
1030 DATA 309,279,686,281,141,301,180,
38,880,760,477,858,898,581,593,7262
1180 DATA 149,632,501,574,782,30,433,5
14,278,757,167,889,901,127,368,7102
1330 DATA 522,288,353,290,621,357,246,
133,513,289,156,291,597,848,294,5798
1480 DATA 366,296,184,516,700,172,893,
894,296,557,298,206,410,410,382,6580
1630 DATA 70,75,385,18,242,18,259,711,
220,626,237,628,277,864,279,4909
2030 DATA 825,848,930,183,978,789,435,
527,700,524,547,279,195,281,185,8226
3040 DATA 588,825,666,683,874,684,281,
947,283,190,64,790,367,311,882,8435
4140 DATA 675,686,503,220,549,928,283,
76,285,195,23,573,793,796,95,6680
5140 DATA 283,916,698,285,593,287,200,
973,442,110,973,977,154,118,287,7296
7010 DATA 989,289,205,375,824,264,712,
254,683,578,289,583,291,540,211,7087
8050 DATA 677,403,84,439,975,729,381,5
85,486,894,228,670,912,298,7761
```

●

Assembly language listing.

```
;
; ****************************
; *        BOPOTRON          *
; *   by Kyle S. Peacock     *
; *    ANALOG COMPUTING      *
; ****************************
;
; ********************
; * ZERO PAGE USAGE *
; ********************
;
PLADR    =   $CB      ;ADDR PLR AREA
PMSTR    =   $CD      ;ADDR PLR DATA
SCREEN   =   $CF      ;ADDR OF SCREEN
;
; ****************************
; * HARD/SOFTWARE REGISTERS *
; ****************************
;
POKMSK   =   $10      ;BREAK DISABLE
SAVMSC   =   $58      ;SCREEN ADDR.
STICK    =   $0278    ;JOYSTICK PORT
COLOR3   =   $02C7    ;COLOR REG. #3
HPOSP0   =   $D000    ;HOR. POS. PLR0
TRIG0    =   $D010    ;JOYSTICK BUTTON
RANDOM   =   $D20A    ;RANDOM NO.
IRQEN    =   $D20E    ;BREAK DISABLE
CHBASE   =   $D409    ;CHAR SET ADDR
WSYNC    =   $D40A    ;WAIT FOR SYNC.
SETVBV   =   $E45C    ;SET VBLANK
XITVBV   =   $E462    ;END OF VBLANK
;
; *****************
; * MEMORY USAGE *
; *****************
;
         *=  $0601
ACTIVATE *=  *+1      ;VBLANK ACTIVE
STARTPT  *=  *+1      ;PLR Y START
ENDPT    *=  *+1      ;PLR Y END
DATAPT   *=  *+1      ;DATA POINTER
XPOS     *=  *+4      ;PLR0 X-COORDS.
YPOS     *=  *+4      ;PLR0 Y-COORDS.
STATUS   *=  *+4      ;PLR0 DEATH FLAG
PNTR     *=  *+2      ;DATA POINTERS.
XSPEED   *=  *+2      ;PLATFORM SPEED
DELTAX   *=  *+2      ;PLAT. X-DELTA
DELTAY   *=  *+2      ;PLAT. Y-DELTA
SPEEDS   *=  *+10     ;PLAT. SPEED
XSTART   *=  *+10     ;PLAT. X-START
YSTART   *=  *+10     ;PLAT. Y-START
XEND     *=  *+10     ;PLAT. X-END
YEND     *=  *+10     ;PLAT. Y-END
CHAR     *=  *+2      ;STANDING ON #.
MOVETIME *=  *+1      ;MOVEMENT TIMER
TASKERS  *=  *+1      ;# OF PLATFORMS
XHOLD    *=  *+1      ;X-REG. HOLD
YHOLD    *=  *+1      ;Y-REG. HOLD
SLIP     *=  *+1      ;ON PLAT. FLAG
FALLCOUNT *= *+2      ;HOW FAR TO FALL...
;
; ******************************
; * VERTICAL BLANK INITIALIZER *
; ******************************
;
         CLD          ;CHILL DECIMAL.
         LDA POKMSK   ;DISABLE BREAK
         AND #$7F     ;KEY. THE BREAK
         STA POKMSK   ;KEY NO LONGER
         STA IRQEN    ;WORKS. (SO?!?)
         PLA          ;# OF VARIABLES
         PLA          ;VBLANK HI/BYTE
         STA XHOLD    ;HOLD ON TO IT.
         PLA          ;VBLANK LO/BYTE
         STA YHOLD    ;HOLD ON TO IT.
         PLA          ;PLAYERS HI/BYTE
         STA PLADR+1  ;STORE IT.
         PLA          ;PLAYERS LO/BYTE
         STA PLADR    ;STORE IT.
         PLA          ;PLAYER STRING
         STA PMSTR+1  ;HI/BYTE.
         PLA          ;PLAYER STRING
         STA PMSTR    ;LO/BYTE.
         LDA #$01     ;STOP OLD VBLANK
         STA ACTIVATE
         LDA PLADR+1  ;HOLD ON TO ADDR
         PHA          ;OF PM AREA.
         LDX #$03     ;LOAD X-REG.
         LDY #$00     ;LOAD Y-REG.
         TYA          ;LOAD A-REG.
CLEAR
         STA (PLADR),Y ;CLEAR PM0
         INY          ;MOVE TO NEXT.
         BNE CLEAR    ;ALL DONE?
         INC PLADR+1  ;MOVE TO NEXT.
         DEX          ;ALL DONE?
         BPL CLEAR    ;NO. GO BACK!
         LDX #$09     ;SET X FOR CLEAR
CLEAR2
         STA SPEEDS,X ;CLEAR OLD
         CPX #$04     ;MOTION TIMERS
         BCS CLEAR3   ;& STATUS. PRE-
         STA STATUS,X ;PARE FOR NEW
CLEAR3
         DEX          ;DATA CONTROL
         BPL CLEAR2   ;INFORMATION.
         STA FALLCOUNT
         STA PNTR
         STA XSPEED+0 ;STOP PREVIOUS
         STA XSPEED+1 ;PLATFORMS.
         STX TASKERS  ;SOUND OFFICIAL?
         LDA #$05     ;I DIDN'T THINK
         STA PNTR+1   ;SO!!!!!!!!
         PLA          ;YES. RESTORE
         STA PLADR+1  ;PM AREA.
         LDX XHOLD    ;VBLANK HI/BYTE
         LDY YHOLD    ;VBLANK LO/BYTE
         LDA #$07     ;DEF. VBLANK.
         JMP SETVBV   ;SET IT UP/BOOGIE...
;
; ***************************
; * DISPLAY LIST INTERRUPT *
; ***************************
;
         *=  $2000
         PHA          ;SAVE ACC.
         LDA $0600    ;CHR. HI/BYTE.
         STA WSYNC    ;WAIT A WHILE.
         STA CHBASE   ;SAVE CHR. SET.
         PLA          ;RESTORE ACC.
         RTI          ;BOOGIE...
;
; ***********************
; * PLAYER DRAW ROUTINE *
; ***********************
;
         CLD          ;CHILL DECIMAL.
         INC COLOR3   ;CHANGE COLOR.
         LDA ACTIVATE ;VBLANK ACTIVE?
         BEQ FUNCTION ;YES! BRANCH!
         JMP XITVBV   ;NO. LATER.
FUNCTION
         LDA PLADR    ;PLR ADDR. LO.
         PHA          ;SAVE IT.
         LDA PLADR+1  ;PLR ADDR. HI.
         PHA          ;SAVE IT.
         LDY #$00     ;BLANK Y REG.
         LDX #$00     ;BLANK X REG.
DRAW5
         LDA XPOS,X   ;PLR X-POS.
         STA HPOSP0,X ;STORE IT.
         LDA YPOS,X   ;PLR Y-POS.
         STA STARTPT  ;SAVE IT.
         CLC          ;GET PLAYER
         ADC (PMSTR),Y ;LENGTH. ADD
         STA ENDPT    ;AND STORE IT.
         INY          ;MOVE TO 1st
         STY DATAPT   ;DATA BYTE.
         CPX #$00     ;PLAYER #0?
         BEQ PASS0    ;YES. BRANCH.
         LDA PLADR    ;INCREMENT
         CLC          ;PLAYER ADDR.
         ADC #128     ;FOR DRAWING
         STA PLADR    ;NEXT PLAYER.
         LDA PLADR+1  ;ADJUST FOR
         ADC #$00     ;256 BYTE PAGE
         STA PLADR+1  ;WRAP-AROUND.
PASS0
         LDA STARTPT  ;STARTING PT.
         SEC          ;SUBTRACT 1
         SBC #1       ;BYTES.
         TAY          ;MOVE TO Y-REG.
         LDA #$00     ;CLEAR ACC.
INSERT1
         STA (PLADR),Y
         INY          ;CLEAR OUT
         CPY STARTPT  ;TOP & BOTTOM
         BCC INSERT1  ;OF PLATFORM
         LDA ENDPT    ;GET ENDPOINT
         CLC          ;AND ADD 1
         ADC #1       ;DECIMAL.
         TAY          ;MOVE TO Y-REG.
         LDA #$00     ;CLEAR ACC.
INSERT2
         STA (PLADR),Y ;STORE BLANK
         DEY          ;MOVE TO NEXT.
         CPY ENDPT    ;AT END?
         BCS INSERT2  ;NO. BLANK MORE.
PASS1
         LDY STATUS,X ;GET STATUS.
         BPL PASS2    ;IF >=0, BRANCH.
         LDY PNTR-2,X ;GET POINTER.
         LDA YPOS,X   ;GET Y-COORD.
         SEC          ;SUBTRACT START-
         SBC YSTART,Y ;ING POINT.
         CMP #$0B     ;IS IT < 12?
         BCC PASS2    ;YES. BRANCH.
         CMP #245     ;IS IT >= 245?
         BCS PASS2    ;YES. BRANCH.
         LDA #$00     ;NO. CLEAR ACC.
         BEQ PASSX    ;BRANCH.
BACKUP
         BCC DRAW5    ;BRANCH POINT.
PASS2
         LDY DATAPT   ;GET POINTER.
         LDA (PMSTR),Y ;GET PLR DATA.
PASSX
```

```
        LDY STARTPT ;GET START PT.
        STA (PLADR),Y ;PUT IT PLR AREA
        INC STARTPT ;INC. AREA PTR.
        INC DATAPT  ;INC. DATA PTR.
        LDA STARTPT ;GET AREA PTR.
        CMP ENDPT   ;AT END?
        BNE PASS1   ;NO. BRANCH.
        LDA STATUS,X ;GET STATUS.
        BPL OUT     ;IF >=0 BRANCH.
        LDA #$00    ;CLEAR ACC.
        STA STATUS,X ;SAVE STATUS.
        CPX #$02    ;DRAWING BOPOTRON?
        BCC OUT     ;YES. BRACH.
        LDY PNTR-2,X ;GET PNTR.
        LDA XSTART,Y ;GET X-START
        STA XPOS,X  ;SAVE AS X-COORD
        LDA YSTART,Y ;GET YSTART
        STA YPOS,X  ;SAVE AS Y-COORD
        LDA #15     ;PAUSE FOR A
        STA XSPEED-2,X ;WHILE.
OUT
        LDY DATAPT  ;RESTORE Y-REG.
        INX         ;MOVE TO NEXT.
        CPX #$04    ;AT END?
        BCC BACKUP  ;NO. BRANCH.
        PLA         ;RESTORE OLD
        STA PLADR+1 ;PLAYER ADDR.
        PLA         ;POINTERS.
        STA PLADR   ;
;
; *************************
; * BIRDER TASKED PLAYERS *
; *************************
;
        LDX TASKERS ;# OF PLATFORMS.
        BMI FORWARD ;NONE. BRANCH.
START
        DEC XSPEED,X ;TIME TO MOVE?
        BNE FORWARD ;NO BRANCH.
        LDA STATUS+2,X ;PLAT ALIVE?
        BEQ SELECT  ;YES. BRANCH.
        LDA #$FF    ;SET UP FOR NEW
        STA STATUS+2,X ;VECTOR.
        BMI FORWARD ;BRANCH.
SELECT
        LDY PNTR,X  ;GET PNTR.
        LDA SPEEDS,Y ;GET PLAT SPEED
        STA XSPEED,X ;SAVE IT.
        LDA XPOS+2,X ;IS PLATFORM AT
        CMP XEND,Y  ;DESTINATION?
        BNE ADDUP   ;NO. BRANCH.
        LDA YPOS+2,X ;NOW CHECK
        CMP YEND,Y  ;Y-COORDS.
        BNE ADDUP   ;NOT THERE. BRANCH.
        LDA #$01    ;CHANGE STATUS
        STA STATUS+2,X ;TO ONE.
        LDA #15     ;PAUSE FOR A
        STA XSPEED,X ;WHILE.
        BPL LOAD@   ;BRANCH.
        BMI LOAD@   ;BRANCH.
BACKTRACK
        BPL START   ;BRANCH POINT.
FORWARD
        BEQ NEXT    ;BRANCH POINT.
        BNE NEXT    ;BRANCH POINT.
LOAD@
        INC PNTR,X  ;AT DESTINATION.
        LDA PNTR,X  ;TIME TO GET
        CMP #$05    ;NEXT PREPROG-
        BNE LOAD1   ;RAMMED VECTOR.
        LDA #$00    ;BUT DON'T GET
        BEQ LOAD2   ;VECTOR IF IT
LOAD1
        CMP #1@     ;IS NOT VALID.
        BNE LOAD2   ;IF INVALID,
        LDA #$05    ;KEEP SEARCHING
LOAD2
        STA PNTR,X  ;FOR VALID
        TAY         ;VECTOR.
        LDA SPEEDS,Y
        BEQ LOAD@
        BPL NEXT
ADDUP
        LDA XPOS+2,X ;ADD PROPER
        CLC         ;DELTA TO PLAT-
        ADC DELTAX,X ;FORM X-COORD.
        STA XPOS+2,X ;SAVE IT.
        LDA YPOS+2,X ;ADD PROPER
        CLC         ;DELTA TO PLAT-
        ADC DELTAY,X ;FORM Y-COORD.
        STA YPOS+2,X ;SAVE IT.
        CPX SLIP    ;BOPOTRON ON
        BNE NEXT    ;THIS PLATFORM?
        LDA XPOS    ;YES. ADD
        CLC         ;X-DELTA TO BOP-
        ADC DELTAX,X ;OTRON X-COORD
        STA XPOS    ;AND SAVE IT.
        STA XPOS+1
        LDA YPOS    ;ADD Y-DELTA TO
        CLC         ;BOPOTRON
        ADC DELTAY,X ;Y-COORD AND
        STA YPOS    ;SAVE IT.
        STA YPOS+1
NEXT
        DEX         ;HANDLE NEXT
        BPL BACKTRACK ;PLATFORM.
;
; *************************
; * BOPOTRON ON PLATFORM? *
; *************************
;
        LDA #$FF    ;CLEAR 'ON
        STA SLIP    ;PLATFORM' FLAG.
        LDX TASKERS ;# OF PLATFORMS
        BMI DELTASDONE ;NONE. QUIT.
SLIPTEST
        LDA XPOS    ;SUBTRACT BOPO-
        SEC         ;TRON X-COORD
        SBC XPOS+2,X ;FROM PLATFORM'S
        BPL NOABS   ;IF >=0 BRANCH.
        EOR #$FF    ;TAKE ABSOLUTE
        CLC         ;VALUE.
        ADC #$01    ;
NOABS
        CMP #$07    ;IS IT >=7?
        BCS SETDELTAS ;YES. BRANCH.
        LDA YPOS+2,X ;NO. SUBTRACT
        SEC         ;BOPOTRON
        SBC YPOS    ;Y-COORD.
        CMP #12     ;IS IT = 12?
        BNE DIETEST ;NO. BRANCH.
        STX SLIP    ;SET FLAG.
        BNE SETDELTAS ;BRANCH.
DIETEST
        BCS SETDELTAS ;IS IT < 12?
        LDA #$01    ;YES!!! BOPOTRON
        STA ACTIVATE ;DIES PAINFULLY
SETDELTAS
        LDA #$00    ;CLEAR OUT OLD
        STA DELTAX,X ;DELTA VALUES.
        STA DELTAY,X ;
        LDY PNTR,X  ;GET PNTR.
        LDA XPOS+2,X ;COMPARE DESTIN-
        CMP XEND,Y  ;ATION TO ACTUAL
        BCC PLUS1X  ;POSITION.
        BEQ NEWDELTA ;
        LDA #$FF    ;DELTA IS -1.
        BMI SETDELTAX
PLUS1X
        LDA #$01    ;DELTA IS +1
SETDELTAX
        STA DELTAX,X ;SAVE X-DELTA.
NEWDELTA
        LDA YPOS+2,X ;COMPARE DES-
        CMP YEND,Y  ;TINATION TO
        BCC PLUS1Y  ;ACTUAL POSITION
        BEQ DELTASDONE
        LDA #$FF    ;DELTA IS -1
        BMI SETDELTAY
PLUS1Y
        LDA #$01    ;DELTA IS +1
SETDELTAY
        STA DELTAY,X ;SAVE Y-DELTA.
DELTASDONE
        DEX         ;CHECK NEXT
        BPL SLIPTEST ;PLATFORM.
;
; ***************************
; * CHARACTER TRACE ROUTINE *
; ***************************
;
PASS3
        LDA SAVMSC  ;GET 1st ADDR.
        STA SCREEN  ;OF SCREEN
        LDA SAVMSC+1 ;MEMORY & SAVE
        STA SCREEN+1 ;IT.
        LDA YPOS    ;BOPOTRON Y.
        SEC         ;SUBTRACT SEX
        SBC #$06    ;DECIMAL.
        LSR A       ;DIVIDE BY FOUR.
        LSR A
        TAX         ;MOVE TO X-REG.
PASS4
        BEQ PASS5   ;IF =0 BRANCH.
        LDA SCREEN  ;GET SCREEN &
        CLC         ;ADD ONE LINE.
        ADC #4@     ;(4@ BYTES)
        STA SCREEN  ;SAVE IT.
        LDA SCREEN+1 ;CORRECT FOR
        ADC #$00    ;PAGE WRAP-
        STA SCREEN+1 ;AROUND.
        DEX         ;AT BOPOTRON'S
        BPL PASS4   ;Y-COORD?
PASS5
        LDA XPOS    ;YES. GET
        SEC         ;X-COORD & SUB-
        SBC #44     ;TRACT 44.
        LSR A       ;DIVIDE BY
        LSR A       ;FOUR.
        TAY         ;MOVE TO Y-REG.
        LDA (SCREEN),Y ;CHARACTER #.
        LDX SLIP    ;BOPOTRON ON
        BMI PASS6   ;PLATFORM?
        LDA #$01    ;YES! CHAR=1.
PASS6
        STA CHAR+1  ;SAVE CHARACTER
        AND #$7F    ;REMOVE @@@
        STA CHAR    ;& SAVE AGAIN.
        LDA YPOS    ;IS BOPOTRON Y-
        CMP #27     ;COORD <27
        BCC PASS7   ;YES! BRANCH.
        LDA SCREEN  ;CHECK SCREEN
        SEC         ;POSITION 2
        SBC #8@     ;LINES UP TO SEE
        STA SCREEN  ;IF BOPOTRON IS
        LDA SCREEN+1 ;CRASHING HIS
        SBC #$00    ;DOME.
        STA SCREEN+1 ;(8@ BYTES)
        LDA (SCREEN),Y ;CHARACTER #.
        CMP #$01    ;IS IT A BIRDER?
        BEQ ZAP     ;YES BRANCH.
        CMP #$08    ;IS IT >=8?
        BCS ZAP     ;YES BRANCH.
PASS7
;
; *************************
; * BOPOTRON MOTION ROUTINE *
; *************************
;
        LDA CHAR    ;GET CHARACTER #
        BEQ FALL    ;IF = @ BRANCH.
        DEC MOVETIME ;DEC TIMER.
        BMI MOTION  ;IF <@ MOVE.
        JMP XITVBV  ;ELSE, QUIT.
MOTION
        PHA         ;SAVE ACC.
        LDA #$01    ;RESET TIMER.
        LDX TRIG@   ;IS BUTTON HELD?
        BEQ RESET   ;NO. BRANCH.
        ASL A       ;YES. MOVE FAST.
RESET
        STA MOVETIME ;SAVE TIMER.
        PLA         ;RESTORE ACC.
        CMP #$02    ;IS CHARACTER <2
        BCC FALLTEST ;IF YES BRANCH.
        CMP #$08    ;IS CHARACTER <8
        BCC CHECK6  ;IF YES BRANCH.
FALLTEST
        LDA SLIP    ;ON PLATFORM?
        BPL CHECK8  ;IF YES BRANCH.
        LDA YPOS    ;MODIFY BOPO-
        AND #$01    ;TRON'S Y-COORD
        BNE FALL    ;SO HE'S ALWAYS
        LDA YPOS    ;ON TOP OF A
        AND #$03    ;BIRDER.
        BNE CHECK8  ;
FALL
        INC FALLCOUNT ;FALLING...
        LDA #$01    ;ADD ONE TO
        BNE ADDY    ;Y-COORD.
CHECK6
        LDA STICK   ;GET JOYSTICK.
        CMP #13     ;PUSHED DOWN?
        BNE CHECK7  ;NO. BRANCH.
        LDA YPOS    ;IS Y-COORD AT
        CMP #98     ;LOWER LIMIT?
        BNE FALL    ;NO. BRANCH.
        BEQ CHECK8  ;YES. BRANCH.
CHECK7
        CMP #14     ;PUSHED UP?
        BNE CHECK8  ;NO. BRANCH.
        LDA #$FF    ;SET TO MOVE UP.
ADDY
        CLC         ;ADD MOVEMENT
        ADC YPOS    ;DELTA TO BOP-
        STA YPOS    ;OTRON'S Y-COORD
        STA YPOS+1  ;AND SAVE IT.
CHECK8
        LDA CHAR    ;GET CHARACTER #
        BEQ ALLDONE ;IF =@ QUIT.
        LDA FALLCOUNT ;IS FALLCOUNT
        CMP FALLCOUNT+1 ;OVER LIMIT?
        BCC CHECK9  ;NO. BRANCH.
ZAP
        LDA #$01    ;STOP VBLANK &
        STA ACTIVATE ;INFORM BASIC.
CHECK9
        LDA #$00    ;NOT FALLING.
        STA FALLCOUNT ;HALT COUNT.
        LDA STICK   ;GET STICK.
        CMP #$07    ;MOVED RIGHT?
        BNE CHECK1@ ;NO. BRANCH.
        LDA XPOS    ;CHECK FOR END
        CMP #198    ;OF SCREEN OR
        BCS ALLDONE ;OBSTRUCTIONS.
        LDA CHAR    ;IF ANY, DO NOT
        CMP #$00    ;ALLOW BOPOTRON
        BEQ ALLDONE ;TO MOVE RIGHT.
        CMP #12     ;
        BEQ ALLDONE ;
        LDA #$01    ;O.K. TO MOVE
        BNE ADDX    ;RIGHT. BRANCH.
CHECK1@
        CMP #11     ;PUSHED LEFT?
        BNE ALLDONE ;NO BRANCH.
        LDA XPOS    ;CHECK TO SEE IF
        CMP #47     ;BOPOTRON IS AT
        BCC ALLDONE ;SCREEN END OR
        LDA CHAR    ;OBSTRUCTED. IF
        CMP #$09    ;SO DO NOT ALLOW
        BEQ ALLDONE ;MOVEMENT TO THE
        CMP #13     ;LEFT.
        BEQ ALLDONE ;
        LDA #$FF    ;O.K. TO MOVE LEFT.
ADDX
        CLC         ;ADD RIGHT OR
        ADC XPOS    ;LEFT DELTA TO
        STA XPOS    ;BOPOTRON'S X-
        STA XPOS+1  ;COORD & SAVE.
ALLDONE
        JMP XITVBV  ;LATE HOME BOY...
;
; ****************************
; * REDEFINED CHARACTER CHART *
; ****************************
;
; CHAR #    TYPE
;
;    @ BLANK SPACE
; !  1 BIRDER
; "  2 BIRDER & LADDER (left)
; #  3 BIRDER & LADDER (middle)
; $  4 BIRDER & LADDER (right)
; %  5 LADDER (left)
; &  6 LADDER (middle)
; '  7 LADDER (right)
; (  8 BATTERY (bottom left)
; )  9 BATTERY (bottom right)
; * 1@ BATTERY (top left)
; + 11 BATTERY (top right)
; , 12 POWER UNIT (bottom left)
; - 13 POWER UNIT (bottom right)
; . 14 POWER UNIT (top left)
; / 15 POWER UNIT (top right)
; @ 16 EXIT BIRDER
```

●

Circuit Database

32K Disk

by Randolph Constan

Being an electronics hobbyist, I have accumulated thousands of schematic diagrams over the years, for all kinds of circuits. A schematic, as you may already know, is a symbolic representation of an electronic circuit. If you're acquainted with them, you also know that they *always* end up on scraps of paper which mysteriously slip into the fourth dimension as soon as you try to locate one. "Where's that diagram you promised me for the video output board for my Atari 400?" asks my friend Jim impatiently. "I'm sorry, buddy," I say in an embarrassed tone, "I just can't seem to find it *anywhere*."

Well, if this sounds all too familiar, boot up your Atari and get ready to kiss this problem good-bye forever. **Circuit Database** will allow you to easily edit, save and retrieve up to sixty-one circuit diagrams, complete with all the necessary symbols and explanatory text, on a single side of a normal density disk. In fact, at only eight sectors per screen, sixty-nine screens could be saved, if DOS 2.0S allowed a bigger directory.

Typing it in.

First, prepare a newly-formatted disk with DOS.SYS. As DUP.SYS is not needed, it can be deleted if you wish. Type in Listing 2 and save it on another disk. RUN it with your new disk inserted, and a special file called D:CIRCHAR.SYS will be created. This file will be used by **Circuit Database** to load the electronic symbol character set, along with a few machine language routines to increase program efficiency. A checksum value will warn you of an error, except in the unlikely event that two or more errors cancel each other out.

When the file has been successfully created and a copy of Listing 2 has been saved, reboot your Atari or type *NEW*, press RETURN and begin typing in Listing 1. When you have finished, make sure a copy is saved before typing *RUN*. Also, the program should be saved on your newly-prepared disk, with the filename D:CIRCUIT.SYS.

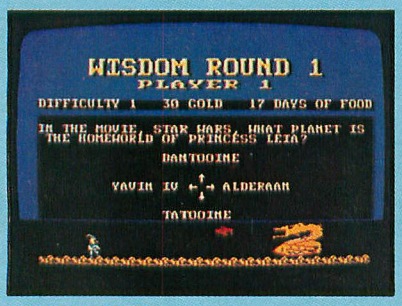
The reason for the .SYS extensions will be explained later. My apologies for the long list of data statements

SURVIVAL OF THE SMARTEST . . .

EXCITING, NEW & DIFFERENT!

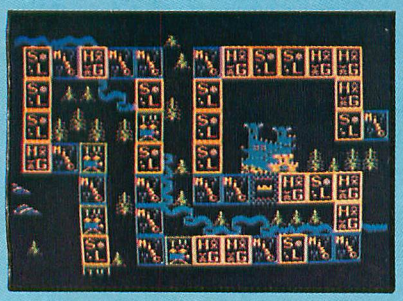
Outsmart your Friends.
Outwit the Dragon.
Join the Quest.

Here are just a few of many screens.



THE CHALLENGE.
Over 2000 stimulating trivia/fact questions will send you rummaging through your personal memory banks. Answer correctly and advance in your quest while adding more gold to your treasure. Fail and you lose ground.

THE ACTION.
You may have to face and battle a ferocious dragon. If you prevail, the dragon's cache of gold will be added to your treasure. If you fail, your journey will be slowed while you heal your wounds.



THE STRATEGY
Each questing party consists of three characters, a Knight, Prince and Page. Each has distinctly different physical characteristics, and all must complete the Quest. You must decide when to send them on to the next challenge.

48K Atari Disk
$39.95

Challenging excitement for 1 to 4 adventurers

This exciting new concept in computer entertainment will keep you and your friends involved for hours and hours. This is probably the best party-game ever developed, and new question disks will be available. **Optional:** Utility disk which allows you to create unlimited trivia questions and answers for educational or entertainment. The utility disk also includes over 1000 additional questions. **Utility disk $24.95.**

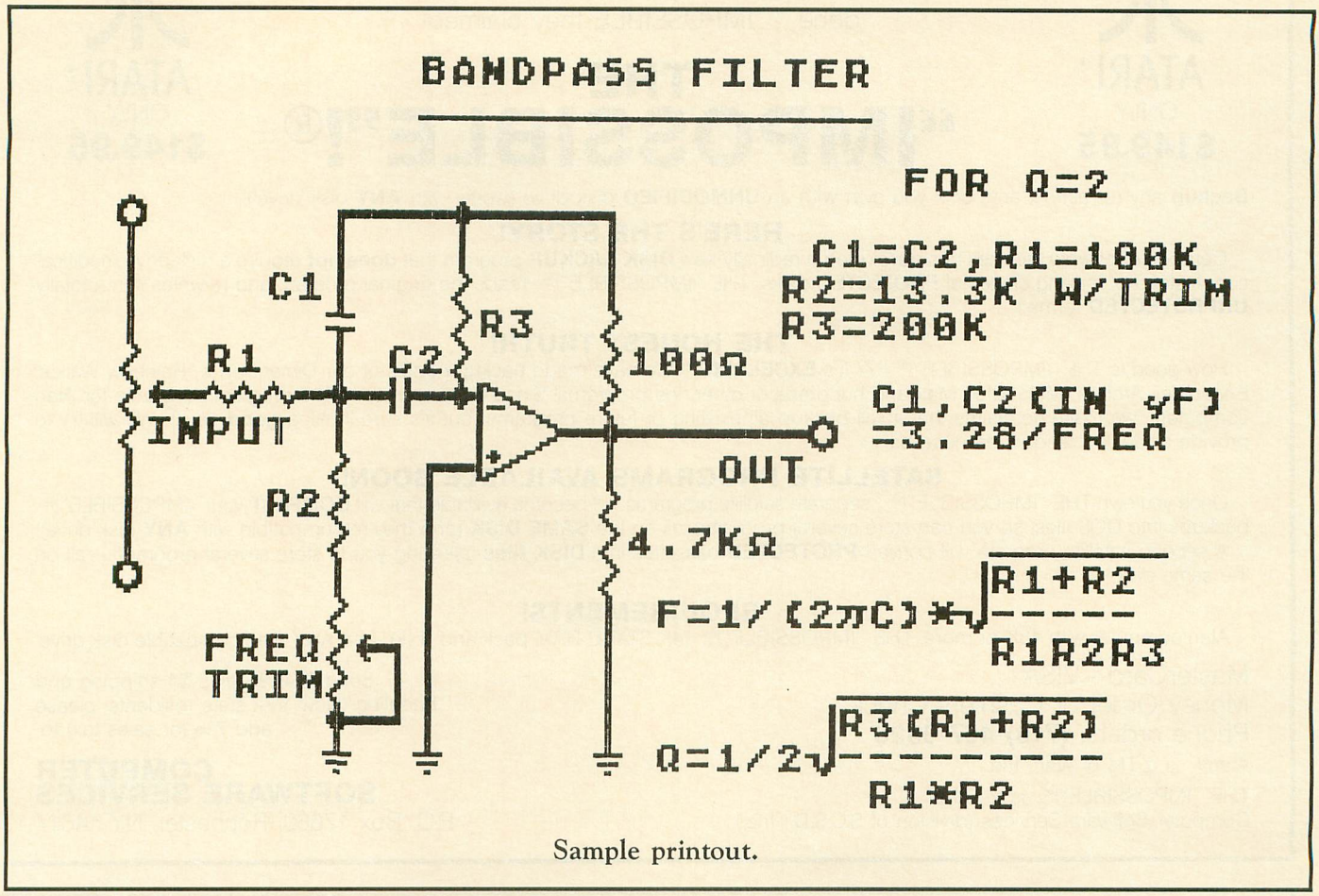
Royal Software

"Software fit for a king!"

2160 W. 11th • Eugene, OR 97402 • (503) 683-5361

Ask for Trivia Quest at your favorite Atari Computer Store or order directly from Royal Software. Use your MasterCard, Visa, American Express, or send check or Money Order including $2.90 shipping and handling.

CIRCLE #116 ON READER SERVICE CARD



Sample printout.

in Listing 2. The character set is heavily edited and so, unfortunately, requires a full 1024 bytes, plus an additional 169 bytes for machine language routines.

About the program.

The obvious advantage of having the main program load in character set and machine language data from a separate file is speed. I hate long initializations and have found this method far superior to waiting for data to be POKEd into memory. This method, together with the CIO fast SAVE/LOAD routines from **ANALOG Computing** issue 13, allow **Circuit Database** to fully initialize within five seconds of typing *RUN*. A side benefit of this method is a significant savings in memory.

If you have finished typing in Listing 1 and have it saved as described above, RUN it. After a short initialization, a menu will appear with three choices. For now, select option 1. *Do not* press RETURN! The program will ask you for a filename. Right now, we are only interested in gaining familiarity with the editor, so just hit RETURN and plug a joystick into port 1. As you move the joystick, the program will begin to draw "wires" on the screen.

Notice that this is a graphics 0 screen, and yet—no matter how you turn, intersect or cross over existing wires—the display looks correct. The screen editor uses a method similar to bit-mapping for drawing these "wires" on the screen. Each time you move the joystick, these mapping routines examine the present cursor location, the characters already above, below and on each side of your present location, and the direction in which you pushed the joystick. This data is compiled into a string variable called TEMP$. Another string variable, WIRE$, is then searched by a fast machine language routine, to "look up" the character which corresponds to the data stored in TEMP$. That character is then printed on the screen. As a result, the drawing action seems a bit more like drawing on a graphics 7 screen than the graphics 0 screen that it actually is. If you notice any inconsistencies in this drawing action, carefully check the characters in Line 15175. This is the actual "look-up table" for the action described above. All circuit drawing is done with the joystick and one-key commands.

Command summary.

When the initial menu appears after first RUNning the program, three options are given. Later, when returning to the menu after viewing or creating a ciruit, a fourth choice will be given. These choices are:

1. DRAW NEW CIRCUIT — Allows entry of a filename, then proceeds to the editor screen. You must enter a valid filename if you wish your finished screen to be saved. *Do not* use the *D:* prefix!

2. VIEW A PREVIOUS CIRCUIT — Allows filename entry and displays selected circuit on the screen. After displaying the circuit, press an *M* to return to the menu, or hit *P* to print the screen on a graphics-compatible Epson or Gemini printer.

3. DISK CIRCUIT DIRECTORY — Will display all files on the disk *without* a .SYS extension. In this way, no space is wasted displaying DOS.SYS or the program files. Up to thirty-nine files can be displayed simultaneously, and, if there are more, you'll be prompted before the directory listing continues. Also, the number of screens you can store with the remaining disk space is calculated and displayed at the bottom of the screen. But note that DOS allows a maximum of sixty-four files.

4. RETURN TO PRESENT CIRCUIT FOR EDITING — This choice is displayed only when a circuit has already been loaded in or has been freshly drawn.

Screen editor commands.

The following is a descriptive listing of all the joystick and keyboard functions available while in the DRAW/EDIT mode (menu items 1 and 4). These functions have many subtle features. Experiment for a while, and you'll find no end to the number of symbols you can create by combining commands with careful editing.

JOYSTICK — Use for drawing wires. These wires automatically join or cross over other wires as you move about the screen. To move without drawing, hold the joystick button down as you move.

E (ERASE MODE) — The cursor can now be moved as an eraser. Push the joystick button to return to normal edit (you *must* return to normal edit to execute other commands).

H (HELP SCREEN) — Displays a summarized listing of all editing commands. Press any key to return to your drawing.

The following commands will only function when the last joystick command was a move to the right and when adequate space is available between the cursor and screen borders:

Q (TRANSISTOR) — Hit *Q* up to six times in succession to get the symbol you want. Two bipolar, two FETS and two unijunction types are given. Hit any key to return.

A (OP-AMP) — Operational amplifier.

I (I.C. GATE) — Hit *I* up to four times for the most appropriate symbol. Press any key to return.

G (GROUND) — For this, the last joystick command must be a downward move.

The following commands will work regardless of joystick direction, provided enough space is available. Otherwise, the "component" will not be drawn—or, in some cases, will be shortened.

R — Resistor.

C — Capacitor.

D — Diode. Polarity will depend on last joystick directional command.

S — Switch.

L — Loop (coil, inductor).

T — Termination point.

ARROW KEYS — Draw arrow chosen. Useful in creating potentiometers and other variable devices.

The following commands are executed while holding down the CONTROL key:

CTRL-P — Print screen. This command will print the circuit display on a Gemini or an Opson printer with the Graftrax graphics option. This process takes several minutes. The cursor will stop flashing during the printing process. When complete, the cursor will resume flashing, and you can continue editing the circuit.

CTRL-S — Solder. Use to solder two completely crossed wires. Position cursor over the crossed pair and push *CTRL-S*. It will then be necessary

(continued on next page)

to hold the joystick button down when moving away, or the "unconnected" default character will re-appear.

CTRL-CLEAR — Clears screen. Caution: picture cannot be recovered.

CTRL-ARROW keys — Scrolls screen in chosen direction. Erasing will occur at screen edges!

CTRL-C — Color change. Changes the background, text and border colors. Many combinations are possible.

CTRL-T — Text mode (see below).

Text mode.

After entering text mode, you can use the CTRL-ARROW keys to move the cursor anywhere on the screen. You can then enter text containing alphabetic and numeric characters. The space bar, backspace and CTRL-ARROWs are the only active editing keys. Also, the following special symbols are available:

CTRL-P — Pi.
CTRL-O — Ohms symbol.
CTRL-M — Micro symbol.
CTRL-S — Square root symbol.

When you have finished entering text, press the ESCape key, and the following options will be displayed on a "mini-menu."

S (SAVE) — Saves screen to disk if a valid filename has been entered. Otherwise, you will be returned to the mini-menu.

E (EDIT) — Return to circuit screen editor and joystick control.

M (MENU) — Brings you back to the main menu. Option 4 will now be available if you wish to return to editing.

A few final notes.

The BREAK key has not been disabled, to permit easy code modification and debugging. If you accidently hit BREAK, GOTO Line 1000 to safely re-enter the program. Also, if you hit SYSTEM RESET, you will have to re-RUN.

Avoid the CAPS/LOWER key, since only capital letters are accepted for input throughout the **Circuit Database** program.

Oh, and one final point: resist the temptation to remove the REM statements to save yourself typing, as I GOTO and GOSUB them frequently. The day will come when you will want to add a few customized commands to the screen editor. If you venture to print out the entire modified character set, you will see many possibilities which have not yet been implemented. However, the program flow is fairly complex, and these REM statements will prove to be an invaluable aid, should you want to make any additions. □

Listing 1.

```
10 REM CHARACTER SET FILEMAKER FOR
20 REM CIRCUIT DATABASE
30 OPEN #1,8,0,"D:CIRCHAR.SYS"
35 B=0:FOR I=1 TO 1024:READ A
40 PUT #1,A:B=B+A
50 NEXT I
60 IF B<>66693 THEN ? "ERROR IN CHARAC
TER DATA STATEMENTS":STOP
70 ? "CHARACTER SET FILE COMPLETE"
75 B=0:FOR I=1 TO 169:READ A
80 PUT #1,A:B=B+A
85 NEXT I
90 IF B<>24752 THEN ? "ERROR IN MACHIN
E LANGUAGE DATA":STOP
95 CLOSE #1
96 ? "MACHINE CODE FILE COMPLETE"
99 END
100 DATA 0,0,0,0,0,0,0,0,0,24,24,24,24
,0,24,0,0,102,102,102
102 DATA 0,0,0,0,0,0,102,255,102,102,255
,102,0,24,62,96,60,6,124,24,0
104 DATA 0,102,108,24,48,102,70,0,0,0,
0,51,102,60,12,24,0,24,24,24
106 DATA 0,0,0,0,0,0,14,28,24,24,28,14,0
,0,112,56,24,24,56,112,0
108 DATA 0,102,60,255,60,102,0,0,0,24,
24,126,24,24,0,0,0,0,0,0
110 DATA 0,24,24,48,0,0,0,126,0,0,0,0,
0,0,0,0,24,24,0
112 DATA 0,6,12,24,48,96,64,0,0,60,102
,110,118,102,60,0,0,24,56,24
114 DATA 24,24,126,0,0,60,102,12,24,48
,126,0,0,126,12,24,12,102,60,0
116 DATA 0,12,28,60,108,126,12,0,0,126
,96,124,6,102,60,0,0,60,96,124
118 DATA 102,102,60,0,0,126,6,12,24,48
,48,0,0,60,102,60,102,102,60,0
120 DATA 0,60,102,62,6,12,56,0,0,0,24,
24,0,24,24,0,0,0,24,24
```

```
122 DATA 0,24,24,48,6,12,24,48,24,12,6
,0,0,0,126,0,0,126,0,0
124 DATA 96,48,24,12,24,48,96,0,0,60,1
02,12,24,0,24,0,0,0,62,99
126 DATA 99,54,119,0,0,24,60,102,102,1
26,102,0,0,124,102,124,102,102,124,0
128 DATA 0,60,102,96,96,102,60,0,0,120
,108,102,102,108,120,0,0,126,96,124
130 DATA 96,96,126,0,0,126,96,124,96,9
6,96,0,0,62,96,96,110,102,62,0
132 DATA 0,102,102,126,102,102,102,0,0
,126,24,24,24,24,126,0,0,6,6,6
134 DATA 6,102,60,0,0,102,108,120,120,
108,102,0,0,96,96,96,96,96,126,0
136 DATA 0,99,119,127,107,99,99,0,0,10
2,118,126,126,110,102,0,0,60,102,102
138 DATA 102,102,60,0,0,124,102,102,12
4,96,96,0,0,60,102,102,102,108,54,0
140 DATA 0,124,102,102,124,108,102,0,0
,60,96,60,6,6,60,0,0,126,24,24
142 DATA 24,24,24,0,0,102,102,102,102,
102,126,0,0,102,102,102,102,60,24,0
144 DATA 0,99,99,107,127,119,99,0,0,10
2,102,60,60,102,102,0,0,102,102,60
146 DATA 24,24,24,0,0,126,12,24,48,96,
126,0,0,30,24,24,24,24,30,0
148 DATA 3,3,3,243,51,55,30,12,0,120,2
4,24,24,24,120,0,0,8,28,54
150 DATA 99,0,0,0,0,0,0,0,0,0,255,0,0,
0,255,54,54,102,204,0
152 DATA 24,24,60,127,127,60,24,24,3,3
,3,3,3,3,3,24,24,24,248
154 DATA 248,0,0,0,24,24,60,254,254,60
,24,24,0,0,0,248,248,24,24,24
156 DATA 255,192,192,192,192,192,192,2
55,192,192,192,255,255,192,192,192,248
,14,7,3
158 DATA 3,7,14,248,0,0,112,223,223,11
2,0,0,24,24,31,0,0,31,24,24
160 DATA 24,31,0,255,255,0,31,24,24,12
6,231,195,195,231,126,24,255,255,0,0
162 DATA 0,0,0,0,0,0,0,0,0,0,255,255,0
,126,126,126,126,126,126,0
164 DATA 3,3,3,255,255,3,3,3,0,0,0,31,
31,24,24,24,0,0,0,255
166 DATA 255,0,0,0,24,24,24,219,255,24
,24,24,3,51,27,255,255,27,51,3
168 DATA 3,51,99,255,255,99,51,3,192,1
92,192,192,192,192,192,192,192,0,24,60,255
170 DATA 255,60,24,24,24,24,60,255,255
,60,24,0,24,24,60,255,255,60,24,24
172 DATA 24,24,24,31,31,0,0,0,120,96,1
20,96,126,24,30,0,24,60,126,24
174 DATA 24,24,24,24,24,24,24,24,24,12
6,60,24,0,24,48,127,127,48,24,0
176 DATA 0,24,12,254,254,12,24,0,24,24
,24,255,0,126,0,24,0,32,112,216
178 DATA 141,7,2,0,24,12,6,12,24,48,96
,48,102,102,102,231,231,102,102,102
180 DATA 24,24,255,0,0,255,24,24,198,2
30,246,255,255,246,230,198,99,103,111,
255
182 DATA 255,111,103,99,255,24,60,126,
255,24,24,24,24,24,24,255,126,60,24,25
5
184 DATA 0,3,6,252,252,0,0,0,0,231,60,
102,102,102,102,60,0,192,96,63
186 DATA 31,0,0,0,24,24,24,24,24,28,6,
3,3,3,118,204,204,118,3,3
188 DATA 3,6,28,24,24,24,24,24,0,240,2
20,199,193,192,223,192,0,0,0,0
190 DATA 192,112,28,7,192,112,28,7,7,2
8,112,192,204,222,204,193,199,220,240,
0
192 DATA 7,28,112,192,0,0,0,0,6,6,6,25
5,254,6,6,6,6,6,0,0
194 DATA 0,0,0,0,0,0,0,0,0,0,6,6,48,96
,192,0,240,224,240,152
196 DATA 48,96,192,128,216,120,120,248
,88,248,96,0,0,96,240,96,14,56,227,135
198 DATA 7,3,0,0,24,220,220,192,96,96,
48,48,24,24,24,24,24,24,24,24
200 DATA 0,126,120,124,110,102,6,0,8,2
4,56,120,56,24,8,0,16,24,28,30
202 DATA 28,24,16,0
300 DATA 104,104,133,204,104,133,203,1
04,104,133,205,160,4,177,203,197,205,2
40,9,152
310 DATA 24,105,5,168,192,84,208,241,2
00,132,212,169,0,133,213,96
400 DATA 104,104,133,204,104,133,203,1
04,133,206,104,133,205,169,0,133,208,1
60,0,162
410 DATA 0,177,203,133,207,177,205,197
,207,208,1,232,200,192,4,208,240,224,4
,240
420 DATA 30,230,208,165,208,201,16,201
,17,240,20,165,203,24,105,5,133,203,16
5,204
430 DATA 105,0,133,204,160,0,162,0,24,
144,206,177,203,133,212,169,0,133,213,
96
500 DATA 104,104,133,207,104,170,104,1
33,206,104,133,205,104,133,204,104,133
,203,160,0
510 DATA 177,203,145,205,202,208,6,165
,207,240,17,198,207,230,203,208,2,230,
204,230
520 DATA 205,208,2,230,206,24,144,228,
234,234,234,234,96
```

•

CHECKSUM DATA.

(see page 90)

```
10 DATA 10,278,930,599,155,380,504,805
,849,163,401,799,523,515,286,7197
100 DATA 969,124,763,345,308,841,843,9
82,962,949,183,219,664,511,708,9371
130 DATA 85,893,417,551,684,974,656,42
7,653,814,11,613,147,621,619,8165
160 DATA 656,794,15,33,173,532,5,295,1
87,786,458,0,954,646,5,5539
190 DATA 919,20,299,248,103,957,499,68
2,741,637,204,394,82,693,494,6972
520 DATA 881,881
```

•

Listing 2.

```
10 REM CIRCUIT COMPILER AND EDITOR
11 REM BY R.CONSTAN, 1984
12 REM ANALOG COMPUTING
13 REM
15 GOSUB 15000
20 GRAPHICS C18:POSITION C7,C3:? #C6;"
SYSTEM          INITIALIZATION":POSITI
ON C1,C8:? #C6;"ONE MOMENT PLEASE"
30 GOSUB 15100:GRAPHICS C0
48 GOTO 1000:REM JUMP TO MENU
49 REM
50 REM JOYSTICK/COMMAND SUBROUTINE
51 IF PEEK(764)<>C255 AND  NOT ERASE T
HEN GOTO 2000:REM ESCAPE SEQUENCE
52 POKE 764,C255
54 DRAW=STRIG(C0):IF ERASE AND  NOT ST
RIG(C0) THEN ERASE=C0
55 TRAP 51:S=STICK(C0):GOTO S+50
57 LET COMMAND=C4:XNOW=XNOW+(XNOW<C38)
:RETURN
61 LET COMMAND=C3:XNOW=XNOW-(XNOW>C0):
RETURN
63 LET COMMAND=C2:YNOW=YNOW+(YNOW<C22)
:RETURN
64 LET COMMAND=C1:YNOW=YNOW-(YNOW>C1):
RETURN
65 POKE 755,( NOT PEEK(755))*C2:SOUND
C1,75,C10-ERASE*C2,PEEK(755)*C1:GOTO 5
1
66 REM GUARDED AREA:ADD NO LINES ABOVE
100 REM CIRCUIT EDITOR INITIALIZATION
110 GRAPHICS C0
115 GOSUB 15140:REM CHECK ML ADDRESSES
120 SETCOLOR C2,C3,C4:SETCOLOR C4,C3,C
4:POKE 82,C0
122 DRAW=C1:ERASE=C0
140 X=C20:Y=C12:XNOW=X:YNOW=Y
```

```
155 POKE 764,C255:REM CLEAR CH
300 REM BASIC LINE DRAWING ROUTINES
305 POSITION X,Y:? " ←";:REM INIT POS.
310 TEMP$="♥♥♥♥":B=C0:REM RESET
320 GOSUB 50:POKE 755,C2:SOUND C1,56,C
10-ERASE*C2,C2:REM GET COMMAND
325 IF ERASE OR  NOT DRAW THEN DELAY=5
IN(C2):GOTO 475
329 REM CHECK PRESENT POS. AND PREVENT
    OVERWRITE OF NON-LINE CHRS.
330 LOCATE XNOW,YNOW,A:POSITION X,Y:IF
 A<>32 AND A<>160 THEN A=USR(SEARCH,WI
RE,A):IF A=85 THEN 320
360 TRAP 380:LOCATE X+C1,Y,A:K$=CHR$(A
):REM RIGHT
364 A=USR(SEARCH,WIRE,A)
368 TEMP$(C1,C1)=WIRE$(A-C3)
380 TRAP 400:LOCATE X-C1,Y,A:K$=CHR$(A
):REM LEFT
384 A=USR(SEARCH,WIRE,A)
388 TEMP$(C2,C2)=WIRE$(A-C4)
400 TRAP 420:LOCATE X,Y+C1,A:K$=CHR$(A
):REM BELOW
404 A=USR(SEARCH,WIRE,A)
408 TEMP$(C3,C3)=WIRE$(A-C1)
420 TRAP 440:LOCATE X,Y-C1,A:K$=CHR$(A
):REM ABOVE
424 A=USR(SEARCH,WIRE,A)
428 TEMP$(C4,C4)=WIRE$(A-C2)
440 POKE 755,C0:POKE 82,X:PUT #6,155:R
EM CORRECT  LOCATE ERROR
441  SOUND C1,C0,C0,C0
442  TEMP$(C5-COMMAND,C5-COMMAND)="├":R
EM CURRENT COMMAND
445 LOCATE X,Y,A:POSITION X,Y:IF A=C32
 OR A=160 THEN 450:REM CURRENT LOCATIO
N
446 B=USR(SEARCH,WIRE,A):IF B=85 THEN
GOTO 480
447 B=B-C5:FOR I=C1 TO C4
448 IF WIRE$(B+I,B+I)="├" THEN TEMP$(I
,I)="├"
449 NEXT I
450 REM NOW FIND CORRECT CHR MATCH!
455 A=USR(CSEARCH,WIRE,ADR(TEMP$))
472 GOTO 480
475 IF  NOT DRAW THEN POKE 82,C0:SOUND
 C1,C0,C0,C0:GOTO 490
476 IF ERASE THEN A=C32
480 POSITION X,Y:? "E";CHR$(A);"←";:PO
KE 82,C0
490 X=XNOW:Y=YNOW:POSITION X,Y:? "→←";
:GOTO 310
499 REM ********************************
**
500 REM COMPONENT DRAW UTILITY ROUTINE
S
510 FOR I=C1 TO N
520 IF (COMMAND=C1) AND ((YNOW<C3) OR
(I=N)) THEN K$="|":I=N:GOTO 540
522 IF (COMMAND=C2) AND ((YNOW>C20) OR
 (I=N)) THEN K$="|":I=N:GOTO 540
530 IF (COMMAND=C3) AND ((XNOW<C2) OR
(I=N)) THEN K$="-":I=N
532 IF (COMMAND=C4) AND ((XNOW>C36) OR
 (I=N)) THEN K$="-":I=N
540 POSITION XNOW,YNOW:? K$:GOSUB L(CO
MMAND):NEXT I
550 POSITION XNOW,YNOW:? " ←";:X=XNOW:
Y=YNOW
560 GOTO 52
580 REM TRANSISTOR OPTIONS
581 ? "↑v|↓←←tw↓←←u|↓←";:XNOW=XNOW+C1:
YNOW=YNOW+C2:GOTO 590
582 ? "↑v|↓←←tx↓←←u|↓←";:XNOW=XNOW+C1:
YNOW=YNOW+C2:GOTO 590
583 ? "↑ ├→↓←←● ↓←← ├→";:XNOW=XNOW+C2:YN
OW=YNOW+1:GOTO 590
584 ? "↑ ├→↓←←▄ ↓←← ├→";:XNOW=XNOW+C2:YN
OW=YNOW+1:GOTO 590
585 ? "↑ |E←↓←←◆ ↓←← |E←";:XNOW=XNOW+C2:
YNOW=YNOW+1:GOTO 590
586 ? "↑ |E→↓←←◆ ↓←← |E→";:XNOW=XNOW+C2:
YNOW=YNOW+1:GOTO 590
590 GET #C2,A:IF A=81 THEN XNOW=X:YNOW
=Y:GOTO 2751
595 X=XNOW:Y=YNOW:GOTO 52
600 REM IC GATE OPTIONS
601 ? "┐↓←▛▀";:GOTO 620
602 ? "┐↓←▛▟▄";:GOTO 620
603 ? "┐↓←▜▟▄";:GOTO 620
604 ? "┐↓←▜▟▗";:GOTO 620
620 XNOW=XNOW+C4:YNOW=YNOW+C1:GET #C2,
A:IF A=73 THEN XNOW=X:YNOW=Y:GOTO 2687
625 X=XNOW:Y=YNOW:GOTO 52
1000 REM MAIN MENU
1010 GOSUB 15140:POKE 82,C0
1015 SETCOLOR C0,C2,C8:SETCOLOR C1,C4,
C10:SETCOLOR C2,C8,C2:SETCOLOR C3,C4,C
4:SETCOLOR C4,C0,C0
1020 ? "K";:POKE DL+C3,70:POKE DL+C6,C
6:POKE DL+C7,C6:POKE DL+C8,C6
1030 ? "  CIRCUIT DATABASE ──────
──── R.CONSTAN * * 1984      menu
 options"
1040 POSITION C0,C3:? "
                                "
1050 FOR I=C4 TO C18:POSITION C0,I:? "
";:POSITION C39,I:? "";:NEXT I
1060 POSITION C0,I:? "
                                "
1070 POSITION C0,C6:? "  1----------
DRAW NEW CIRCUIT":?
1075 ? "  2----------VIEW PREVIOUS C
IRCUIT":?
1080 ? "  3----------DISK CIRCUIT  D
IRECTORY":?
1090 IF DRAW THEN ? "  4----------RE
TURN TO PRESENT▶▶▶CIRCUIT FOR EDITING"
:? :?
1095 ? "        *** ENTER SELECTION **
*":POSITION C0,C0:? "→←";
1100 GET #C2,MENU:IF MENU>C128 THEN PO
KE 764,C39
1110 IF (MENU<49) OR (MENU>52-( NOT DR
AW)) THEN SOUND C1,C256-C1,C10,C10:DEL
AY=C2^C2:SOUND C1,C0,C0,C0:GOTO 1100
1120 MENU=MENU-48:IF MENU>C2 THEN 1300
1125 FILE$="D:"
1130 POSITION C4,C21:? "ENTER CIRCUIT
NAME: [              ]";:POSITION 25,21:
? " ←";
1132 FOR I=C1 TO C13
1135 GET #C2,B:IF B=155 THEN 1170
1136 IF (I=C13) AND (B<>155) THEN 1135
1138 IF B>C128 THEN POKE 764,C39:GOTO
1135
1140 IF (B=126) AND (I>C1) THEN ? "← ←
";:I=I-C1:FILE$(I+C2)="":GOTO 1135
1145 IF (B=46) OR ((B)47) AND (B(58))
OR ((B)64) AND (B(91)) THEN FILE$(I+C2
,I+C2)=CHR$(B):GOTO 1160
1150 SOUND C1,C256-C1,C10,C10:DELAY=SI
N(C2):SOUND C1,C0,C0,C0:GOTO 1135
1160 POSITION 24+I,C21:? CHR$(B);:NEXT
 I
1170 IF MENU=C1 THEN 100
1180 TRAP 1250:CLOSE #C1:OPEN #C1,4,C0
,FILE$:GRAPHICS C0:POKE 709,C0:POKE 71
0,C0:POKE 712,C0:GOSUB 15140
1200 ADDRESS=SCREEN:NUMBER=C960+C10:IO
=C1:GOSUB 6200:NUMBER=NUMBER+SCREEN-C9
:CLOSE #C1
1205 GOSUB 1650:B=USR(MOVE,C960+C10,AD
R(HOLD$),SCREEN)
1210 POSITION C2,C23:? "*** PRESS M FO
R MENU, P TO PRINT ***";:POSITION C0,C
0:? "→←";:POKE SCREEN,C0
1220 DRAW=C1:GET #C2,A:IF A>C128 THEN
POKE 764,C39:A=A-C128
1230 IF A=77 THEN 1240
1232 IF A=80 THEN GOSUB 16000
1233 GOTO 1220
1240 POSITION C0,C23:? BL$;:GOTO 1000
1250 POSITION C2,C21:? "  FILE NOT FO
UND-CAN'T ACCESS DISK!  ":FOR I=C1 TO C
10:POKE 755,C3:SOUND C1,I,C4,C10
1252 DELAY=SIN(C2):POKE 755,C2:DELAY=S
IN(C2):SOUND C1,C0,C0,C0:NEXT I
1255 POSITION C2,C21:? BL$:GOTO 1100
1300 IF MENU<>C4 THEN 1350
1301 REM RESTORE CIRCUIT
1305 POKE DL+C3,66:POKE DL+C6,C2:POKE
DL+C7,C2:POKE DL+C8,C2
```

```
1310 ? "{CLR}":B=USR(MOVE,970,SCREEN,ADR(H
OLD$)):GOSUB 1650:GOTO 122
1350 POKE 752,C1:REM DIRECTORY
1355 GOSUB 1600
1360 POSITION C0,C0:? " {UP}";
1370 TRAP 1380:CLOSE #C1:OPEN #C1,C6,C
0,"D:*.*":GOTO 1400
1380 POSITION C10,C7:? "CAN'T ACCESS D
ISK ";:SOUND C1,C128,C12,C10:DELAY=C2
^C2^C2:SOUND C1,C0,C0,C0
1390 FOR DELAY=C1 TO C10:NEXT DELAY:?
"{CLR}":POKE 752,C0:GOTO 1000
1400 TRAP 40000:FOR B=C1 TO C2:FOR I=C
1 TO 27 STEP C12:FOR J=C5 TO C17
1410 TRAP 1410:INPUT #C1,DIR$:IF DIR$(
C5)="FREE SECTORS" THEN 1440
1415 IF DIR$(C11,C13)="SYS" THEN 1410
1420 POSITION I,J:? BL$(C1,C12);:POSIT
ION I,J:? DIR$(C2,C13);:NEXT J:NEXT I
1430 POSITION C5,C20:? "CONTINUE (Y,N)
 ? ";:GET #C2,A:IF A=78 THEN 1000
1435 POSITION C0,C20:? BL$;:POSITION C
0,C0:? " {UP}":NEXT B
1440 POSITION C0,C20:? "{RIGHT}DISK SPACE ="
;INT(VAL(DIR$(C1,C3))/C8);" SCREENS.":
? "{RIGHT}PRESS ANY KEY FOR MENU";
1450 CLOSE #C1:GET #C2,A:IF (A>C128) A
ND (A<>155) THEN POKE 764,C39
1455 ? "{CLR}":POKE 752,C0:GOTO 1000
1599 STOP
1600 REM CLEAR WINDOW
1610 COLOR 160:FOR I=C5 TO C18:PLOT C1
,I:DRAWTO C38,I
1620 SOUND C1,I,C10,C10:POSITION C1,I:
? #6;BL$(C1,C38);:NEXT I:? #6:SOUND C1
,C0,C0,C0:RETURN
1640 FOR I=C1 TO C9:POKE SCREEN+C960+I
,PEEK(703+I):NEXT I:RETURN :REM SAVE C
OLORS
1650 FOR I=C1 TO C9:POKE 703+I,PEEK(SC
REEN+C960+I):NEXT I:RETURN :REM RESTOR
E COLORS
2000 REM LINES 2000 TO 4200 RESERVED
2001 REM  LINKUP AREA
2002 REM DONT FORGET POP IF EXIT!
2010 IF PEEK(764)=C39 THEN POKE 764,C3
9:GOTO 52:REM INVERSE
2015 LINK=500
2020 TRAP 52:GET #C2,LINE:POKE 755,C0:
LINE=LINE*C8+2100:GOTO LINE
2124 REM (CTRL)"C"=KOLOR CHANGE
2125 A=PEEK(710):A=A+C16:POKE 710,(A<C
256)*A+(A>=C256)*(A-C256)
2126 IF A>C256 THEN A=PEEK(709):POKE 7
09,PEEK(710):POKE 710,A
2127 A=PEEK(712):A=A+16:POKE 712,(A<C2
56)*A+(A>=C256)*(A-255)
2128 GOTO 52
2228 REM (CTRL)P=EPSON SCREEN PRINT
2229 GOSUB 16000:GOTO 52
2252 REM (CTRL)S=SOLDER
2253 LOCATE X,Y,A:IF A=C19 OR A=147 TH
EN POSITION X,Y:? "{INV} {UP}";
2254 GOTO 52
2260 GOTO 5000:REM (CTRL)"T"=TEXT MODE
2324 POSITION C0,C0:? "{INS}";BL$;:IF Y>C1
 THEN Y=Y-C1
2325 YNOW=Y:POSITION X,Y:? "{RIGHT}{UP}";:GOTO
52:REM SCROLL UP
2332 POSITION C0,C0:? "{DEL}":POSITION C0,
C23:? BL$;:IF Y<C22 THEN Y=Y+C1
2333 YNOW=Y:POSITION X,Y:? "{RIGHT}{UP}";:GOTO
52:REM SCROLL DOWN
2340 FOR I=C1 TO C22:POSITION C0,I:? "
{INV} ":SOUND C1,I,C8,C2:NEXT I:IF X>C0 TH
EN X=X-C1
2341 XNOW=X:POSITION X,Y:? "{RIGHT}{UP}";:GOTO
52:REM SCROLL LEFT
2348 FOR I=C1 TO C22:POSITION C38,I:?
" ";:POSITION C0,I:? "{DEL}":SOUND C1,I,C8
,C2:NEXT I:IF X<C38 THEN X=X+C1
2349 XNOW=X:POSITION X,Y:? "{RIGHT}{UP}";:GOTO
52:REM SCROLL RIGHT
2436 ? "{SHIFT-=}{RIGHT}";:GOTO 52
2444 ? "{SHIFT-=}{LEFT}";:GOTO 52
2460 ? "{SHIFT-=}{UP}";:GOTO 52
2588 ? "{SHIFT-=}{DOWN}";:GOTO 52
2620 REM A=OP AMP
2621 IF (COMMAND<>C4) OR (XNOW>C34) OR
 (YNOW>C20) THEN 52
2622 ? "op{LEFT}{LEFT}{DOWN}  q{DOWN}{LEFT}{LEFT}{LEFT}{LEFT}{DOWN}rs{UP}{RIGHT}{RIGHT}";:XNOW=XNO
W+C4:YNOW=YNOW+C1:X=XNOW:Y=YNOW:GOTO 5
2
2636 REM CAPACITOR DRAW
2637 N=C2:IF COMMAND<C3 THEN K$="d":GO
TO LINK
2638 K$="c":GOTO LINK
2644 REM DIODE DRAW
2645 N=C2:IF COMMAND=C1 THEN K$="g":GO
TO LINK
2646 IF COMMAND=C2 THEN K$="h":GOTO LI
NK
2647 IF COMMAND=C3 THEN K$="f":GOTO LI
NK
2648 IF COMMAND=C4 THEN K$="e":GOTO LI
NK
2652 ERASE=C1:GOTO 52
2668 REM GROUND DRAW
2669 IF COMMAND=C2 THEN ? "{INV}{LEFT}";:GOTO 5
2
2670 GOTO 52
2676 REM HELP SCREEN
2677 POSITION C0,C0:? "{RIGHT}{LEFT}";:POKE SCREE
N,C0:B=USR(MOVE,C960,ADR(HOLD$),SCREEN
)
2678 ? "{CLR}":GOSUB 4500:B=USR(MOVE,C960,
SCREEN,ADR(HOLD$)):POSITION X,Y:? "{RIGHT}{LEFT}"
;:GOTO 52
2684 REM I=IC GATE
2685 IF (COMMAND<>C4) OR (XNOW>C34) OR
 (YNOW>C20) THEN 52
2686 B=C0
2687 SOUND C1,C36+C5*B,C10,C4:DELAY=C2
^C2:SOUND C1,C0,C0,C0:B=B+C1:IF B=C5 T
HEN B=C1
2688 POSITION X,Y:GOTO 600+B
2708 REM L=COIL (LOOP)
2709 IF COMMAND=C1 AND YNOW>C5 THEN ?
"n{UP}{LEFT}m{UP}{LEFT}m{UP}{LEFT}l{UP}{LEFT}|{UP}{LEFT}";:YNOW=YNOW-C5:Y=YNOW
:GOTO 52
2710 IF COMMAND=C2 AND YNOW<C18 THEN ?
 "l{DOWN}{LEFT}m{DOWN}{LEFT}n{DOWN}{LEFT}|{DOWN}{LEFT}";:YNOW=YNOW+C5:Y=YNO
W:GOTO 52
2711 IF COMMAND=C3 AND XNOW>C5 THEN ?
"k{LEFT}{LEFT} j{LEFT}{LEFT} j{LEFT}{LEFT} j{LEFT}{LEFT}i{LEFT}{LEFT}-{LEFT}{LEFT}";:XNOW=XNOW-C6:X=X
NOW:GOTO 52
2712 IF COMMAND=C4 AND XNOW<C33 THEN ?
 "i j j jk-";:XNOW=XNOW+C6:X=XNOW:GOTO 52
2748 REM Q=TRANSISTOR
2749 IF (COMMAND<>C4) OR (XNOW>C37) OR
 (YNOW<C3) OR (YNOW>20) THEN 52
2750 B=C0
2751 SOUND C1,C36+C5*B,C10,C4:DELAY=C2
^C2:SOUND C1,C0,C0,C0:B=B+C1:IF B=C7 T
HEN B=C1
2752 POSITION X,Y:GOTO 580+B
2756 REM RESISTOR
2757 N=C4:IF COMMAND<C3 THEN K$="b":GO
TO LINK
2758 K$="a":GOTO LINK
2764 REM S=SWITCH
2765 N=C2:IF COMMAND<C3 THEN K$="{UP}":GO
TO LINK
2766 K$="z":GOTO LINK
2772 REM TERMINATION
2773 ? "{INV} {LEFT}";:GOTO 52
3100 ? "{CLR}":POSITION X,Y:? "{RIGHT}{LEFT}";:GOTO 5
2:REM CLEAR SCREEN
4200 REM END OF RESERVED AREA !
4500 POKE 755,C2:POKE 752,C1:SOUND C1,
C0,C0,C0
4502 ? "{CLR}     CIRCUIT DRAWING COMMAND S
UMMARY{RIGHT}{RIGHT}_____
_____"
4505 ? "{RVS} -aa-{RIGHT}{INV} -c-{RIGHT}{RVS} i j j k{RIGHT}{INV} -z-{RIGHT}{INV} -e
-";:?
4510 ? "{RVS} --{RIGHT}{INV} {RIGHT}{INV}{LEFT}{LEFT}{UP}{UP}{DOWN}{DOWN}{RIGHT} --DRAW ARRO
WS";:?
4515 ? "ONE DIRECTION ({RIGHT}) ONLY:"
4520 ? "    -op          {RVS} {INV}, (4 TYPES
)"
4521 ? "{INV}   | q-":?
4522 ? "    -rs          {RVS} tx    (6 TYPES
)":? :?
```

```
4548 ? "[CTRL] S = 'SOLDER' CROSSED WI
RES"
4550 ? "[CTRL] C = COLOR CHANGE"
4552 ? "[CTRL] ←↑↓→ = SCROLL SCREE
N"
4554 ? "[CTRL] CLEAR =CLEAR SCREEN"
4556 ? "[CTRL] T = ENTER TEXT MODE,[ES
C]= EXIT":?
4558 ? "E=ERASE--PUSH JOYSTICK BUTTON
TO EXIT"
4560 ? "H=HELP SCREEN":? :?
4600 ? "PRESS ANY KEY TO RETURN TO EDI
T":GET #C2,A:POKE 752,C0:? "E":RETURN
5000 TRAP 40000:REM TEXT MODE
5005 POKE 755,C2:SOUND C1,C0,C0,C0:POS
ITION C0,C0:? "→←";:POKE SCREEN,C0:B=U
SR(MOVE,C960,ADR(HOLD$),SCREEN)
5006 POSITION X,Y:? " ←";
5010 IF PEEK(764)=C256-C1 THEN 5100
5015 GET #C2,A
5016 IF A=C13 THEN POSITION X,Y:? "&";
:X=X+C1:GOTO 5050
5017 IF A=C15 THEN POSITION X,Y:? "@";
:X=X+C1:GOTO 5050
5018 IF A=C16 THEN POSITION X,Y:? "♥";
:X=X+C1:GOTO 5050
5019 IF (A=19) AND (X<C35) AND (Y>C0)
THEN POSITION X,Y:? "\↑← ┌──┘↓←←←";:X=X
+C1:GOTO 5050
5020 IF ((A>C31) AND (A<96)) OR ((A>15
9) AND (A<224)) THEN POSITION X,Y:? CH
R$(A);:X=X+C1
5022 IF (A>155) AND (A<160) THEN POKE
764,C39:A=A-C128
5025 IF (A>27) AND (A<32) THEN ? CHR$(
A);:GOTO 5000+A
5026 GOTO 5050
5028 Y=Y-(Y>C0):GOTO 5040
5029 Y=Y+(Y<C23):GOTO 5040
5030 X=X-(X>C0):GOTO 5040
5031 X=X+(X<C38):GOTO 5040
5040 POSITION X,Y:? "→←";
5050 IF (A=126) OR (A=254) THEN X=X-(X
<>C0):POSITION X,Y:? " ←";:IF A=254 TH
EN POKE 764,C39
5060 IF A=27 THEN GOTO 5200
5080 IF A=125 OR A=253 THEN B=USR(MOVE
,960,SCREEN,ADR(HOLD$)):POSITION X,Y:?
 "→←";
5090 IF X=C39 THEN X=C38:POSITION X,Y:
? "→←";
5100 DELAY=DELAY+C1:IF DELAY>C4 THEN D
ELAY=C0:POKE 755,( NOT PEEK(755))*C2
5110 GOTO 5010
5200 POKE 755,C2
5210 POSITION C0,C0:? "→←";:POKE SCREE
N,C0:GOSUB 1640:B=USR(MOVE,C960+C10,AD
R(HOLD$),SCREEN)
5220 POKE 752,C1:FOR I=C0 TO C8:POSITI
ON C10,I:? "                              ";:SOU
ND C1,I*C4,C8,C4:NEXT I
5225 SOUND C1,C0,C0,C0
5230 POSITION C12,C1:? "ENTER COMMAND:
";
5235 POSITION C16,C3:? "E=EDIT"
5240 POSITION C16,C4:? "S=SAVE"
5250 POSITION C16,C5:? "M=MENU"
5260 POSITION C21,C7:? "*";:POSITION C
0,C0:? "→←";:POKE SCREEN,C0:GET #C2,A:
IF A>C128 THEN A=A-C128:POKE 764,C39
5270 IF (A=69) OR (A=83) OR (A=77) THE
N 5300
5280 SOUND C1,200,C10,C10:DELAY=C2^C2:
SOUND C1,C0,C0,C0:GOTO 5260
5300 GOSUB 1640:B=USR(MOVE,C960+C10,SC
REEN,ADR(HOLD$))
5310 IF A=69 THEN POKE 752,C0:XNOW=X:Y
NOW=Y:POSITION X,Y:? " ←";:GOTO 52
5320 IF A=77 THEN POKE 752,C0:POP :GOT
O 1000
5330 GOSUB 1640:TRAP 5400:CLOSE #C1:OP
EN #C1,C8,C0,FILE$
5340 NUMBER=C960+C10:ADDRESS=SCREEN:IO
=C1:GOSUB 6000
5350 POSITION C10,C7:? "K ** SAVE COMP
LETE **";
5360 DELAY=C2^C2^C2:GOTO 5220
5400 SOUND C1,C256-C1,C10,C10:POSITION
 C10,C7:? "  CAN'T ACCESS DISK  ";
5410 DELAY=C2^C2:SOUND C1,C0,C0,C0:FOR
 DELAY=C1 TO C16:NEXT DELAY:GOTO 5220
5990 END
6000 REM CIO TO PUT BYTES
6010 IO=C16*IO
6020 IOCB=832+IO:POKE IOCB+C2,C11
6030 ADRHI=INT(ADDRESS/256)
6040 ADRLO=ADDRESS-C256*ADRHI
6050 POKE IOCB+C4,ADRLO:POKE IOCB+C5,A
DRHI
6060 NUMHI=INT(NUMBER/256)
6070 NUMLO=NUMBER-C256*NUMHI
6080 POKE IOCB+C8,NUMLO:POKE IOCB+C9,N
UMHI
6090 I=USR(ADR("hhh▒LV▒"),IO)
6100 CLOSE #IO/C16
6110 RETURN
6200 REM CIO TO GET BYTES
6210 IO=C16*IO
6220 IOCB=832+IO:POKE IOCB+C2,C7
6230 ADRHI=INT(ADDRESS/C256)
6240 ADRLO=ADDRESS-ADRHI*C256
6250 POKE IOCB+C4,ADRLO:POKE IOCB+C5,A
DRHI
6260 NUMHI=INT(NUMBER/C256)
6270 NUMLO=NUMBER-C256*NUMHI
6280 POKE IOCB+C8,NUMLO:POKE IOCB+C9,N
UMHI
6290 I=USR(ADR("hhh▒LV▒"),IO)
6298 RETURN
15000 REM PROGRAM INITIALIZATION
15040 REM SET UP CONSTANTS
15050 C0=0:C1=1:C2=C1+C1:C3=C2+C1:C4=C
3+C1:C5=C4+C1:C6=C5+C1:C7=C6+C1:C8=C7+
C1:C9=C8+C1:C10=C9+C1:C11=C10+C1
15055 C12=C11+C1:C13=C6+C7:C14=C7+C7:C
15=C7+C8:C16=C8+C8:C17=C9+C8:C18=C9+C9
:C19=C10+C9:C20=C10+C10:C21=C20+C1
15060 C22=C21+C1:C23=C21+C2:C30=C23+C7
:C31=C30+C1:C32=C30+C2:C33=C32+C1:C34=
C33+C1:C35=C34+C1:C36=C35+C1
15070 C37=C36+C1:C38=C37+C1:C39=C38+C1
:C40=C39+C1:C128=128:C256=C128+C128:C2
55=C256-C1:C960=960
15078 DIM L(C4):L(C0)=52:L(C1)=64:L(C2
)=63:L(C3)=61:L(C4)=57:LET COMMAND=C0
15080 REM DIMENSION STRINGS
15085 DIM WIRE$(85),TEMP$(C4),K$(C1),B
L$(C39),HOLD$(980),FILE$(C15),DIR$(C30
)
15090 REM LOWER MEMTOP
15095 POKE 106,PEEK(106)-C8:GRAPHICS C
0:DRAW=C0:RETURN
15100 ADDRESS=(PEEK(106)+C4)*C256:NUMB
ER=1024:IO=C1:OPEN #C1,C4,C0,"D:CIRCHA
R.SYS":GOSUB 6200
15110 CHAR=ADDRESS/C256:REM CH SET LOC
N
15120 ADDRESS=1536:NUMBER=169:IO=C1:GO
SUB 6200:CLOSE #C1:REM ML LOCATIONS
15140 SEARCH=1536
15150 CSEARCH=SEARCH+C36
15160 MOVE=CSEARCH+C40+C40
15170 REM SET UP STRINGS
15175 WIRE$="┣┣┣┣┣┣┣┣┣♥┣┣┣┣┣♥♥┣┣♥┣┣
┣♥┣♥┣┣♥♥┣┣♥♥♥┣♥┣┣┣┣♥┣┣♥┣♥┣♥┣┣♥┣
┣┣♥♥┣♥┣♥♥♥┣┣┣┣┣♥♥♥♥ ":REM "BITMAP"
15180 WIRE=ADR(WIRE$):BL$=" ":BL$(C39)
=BL$:BL$(C2)=BL$
15200 SCREEN=PEEK(88)+C256*PEEK(89)
15210 DL=PEEK(560)+C256*PEEK(561)
15220 CLOSE #C2:OPEN #C2,4,C0,"K:"
15230 POKE 756,CHAR
15300 RETURN
16000 REM EPSON SCREEN PRINT
16010 PSC=PEEK(88)+PEEK(89)*256:TRAP 1
6060:CLOSE #C5:OPEN #C5,C8,C0,"P:":? #
C5;"[ESC]A";CHR$(C8)
16020 FOR I=PSC TO PSC+39:? #C5;"[ESC]K";C
HR$(184);CHR$(C0);
16030 FOR J=I+880 TO I STEP -C40:CSUB=
C0:C52=C0:IF PEEK(J)>127 THEN CSUB=C12
8:C52=C255
16040 PCH=ABS(PEEK(J)-CSUB):CHLOC=PEEK
(756)*C256+PCH*C8
```

```
16050 FOR J2=CHLOC+C7 TO CHLOC STEP -C
1:PUT #C5,ABS(C52-PEEK(J2)):NEXT J2:NE
XT J:? #C5:NEXT I:? #5;"£@":CLOSE #1
16060 TRAP 40000:RETURN
```

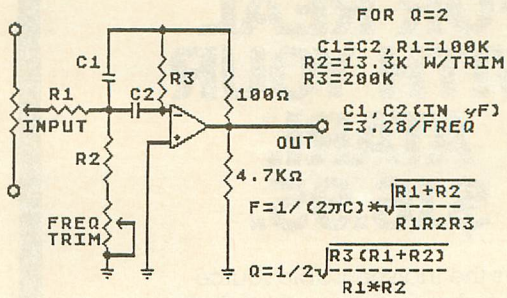
●

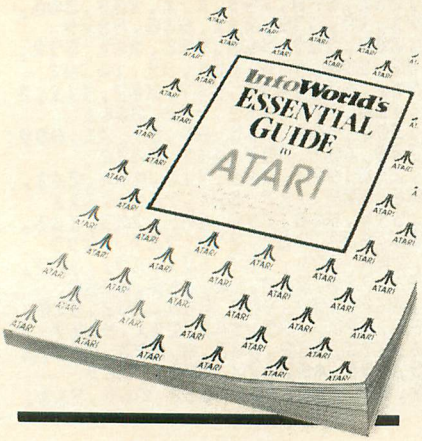
CHECKSUM DATA.

(see page 90)

```
10 DATA 48,578,295,260,851,374,494,306
,284,197,976,265,954,223,640,6745
61 DATA 408,615,416,277,414,807,147,55
1,507,755,199,967,816,177,19,7075
320 DATA 898,786,609,686,745,436,762,1
49,442,771,705,420,748,692,426,9275
428 DATA 757,779,808,172,499,260,322,2
90,755,547,850,742,366,56,842,8045
490 DATA 350,72,564,207,504,34,140,628
,236,224,520,442,896,898,358,6073
584 DATA 362,842,848,571,924,1,587,561
,592,566,544,905,787,359,75,8524
1020 DATA 316,503,804,338,843,28,877,8
62,966,88,225,149,845,953,506,8303
1132 DATA 399,80,197,87,963,524,439,25
2,99,661,446,931,911,48,860,6897
1232 DATA 564,717,746,59,846,424,87,30
0,505,227,331,956,199,762,475,7198
1390 DATA 142,697,220,653,425,564,825,
234,946,702,621,135,829,239,381,7613
1650 DATA 634,504,939,727,307,916,163,
985,371,489,171,648,703,63,232,7852
2253 DATA 811,646,767,203,319,614,157,
922,148,666,519,388,383,373,396,7312
2620 DATA 425,157,878,908,895,801,452,
893,599,596,596,825,786,972,652,10435
2676 DATA 690,730,870,769,171,269,306,
956,639,913,471,18,430,211,636,8079
2750 DATA 257,300,959,988,898,802,592,
943,849,853,236,994,938,602,91,10302
4505 DATA 75,951,130,198,947,382,911,2
88,856,61,858,420,776,812,996,8661
5005 DATA 130,308,908,753,59,91,31,899
,278,300,962,740,467,263,445,6634
5031 DATA 249,298,924,361,215,605,241,
722,180,448,954,829,117,657,711,7511
5250 DATA 697,599,900,798,856,289,335,
816,117,137,836,693,782,297,747,8899
6010 DATA 814,204,696,902,621,491,102,
758,696,378,793,710,820,24,960,8969
6240 DATA 887,627,778,108,764,702,820,
385,799,891,626,773,799,249,221,9429
15085 DATA 794,423,332,687,8,873,202,3
52,364,943,789,603,407,70,402,7249
15230 DATA 702,58,714,669,692,665,213,
933,234,4880
```

●





PLAY FOR REAL WITH YOUR ATARI.® $16.95.

Here's the indispensable source on the choice, care and feeding of your Atari computer. Everything you'll ever need to know about your Atari: inside, up-to-date advice on the hardware; critical reviews of the software; and even an exhaustive look at the peripherals.

InfoWorld's Essential Guide to Atari Computers has been written by Scott Mace, an authority on Atari, and the editors of InfoWorld, the respected voice of the industry.

Now your Atari can be more than just fun and games.

Please send InfoWorld's Essential Guide to Atari Computers. I enclose $16.95 + $1.50 postage & handling (and any sales tax).

NAME _____

ADDRESS _____

CITY _____

STATE _____ ZIP _____

[ ] MasterCard [ ] Visa [ ] AmEx

Card #: _____ Exp. ____

Send to: Harper & Row
10 East 53rd Street, 20th floor
New York, NY 10022

Or call toll free 800-638-3030.
(In MD, call collect 301-824-7300.) Or see your local bookseller or computer store.

Harper&Row

CIRCLE #121 ON READER SERVICE CARD

THANK YOU !!!

Your tremendous response has allowed ***LotsaBytes*** to hold down prices and expand our product line. Our National Public Domain Service will continue to offer **quality, quantity, selection,** and the **low prices** that you deserve. Our selections from our Master Library of thousands of Public Domain programs will save you time, tedious work, and money. The price per Public Domain disk is still only **$7.95** and there are more **FREE BONUSES** than ever before to choose from.

PUBLIC DOMAIN SOFTWARE

# 1 GAMES	# 2 UTILITIES	# 3 AMS MUSIC	# 4 GAMES
Two full disk sides packed with over 25 games including some Arcade quality. **$7.95**	25 powerful programs to help you get the most out of your Atari computer. **$7.95**	25 Advanced Musicsystem files including a new Player program. 2 sides. **$7.95**	All different! Filled on both sides with classics, new games, some Arcade types. **$7.95**
# 5 EDUCATION	**# 6 AMS MUSIC**	**# 7 GAMES**	**# 8 UTILITIES**
We loaded this disk up with something for everyone in the family. Two sides. **$7.95**	All different! Some of your favorite music. Includes Player program. 2 sides. **$7.95**	No duplicates here. Includes some of the latest Arcade types. Two full sides. **$7.95**	Another all different assortment to unleash the full power of your Atari. **$7.95**

If you purchase any three or more disks at one time you may choose any one of the following disks **FREE!!**

a. The Atari XL TRANSLATOR DISK that enables XL owners to use most 400/800 software. **FREE!!**

- - or - -

b. An all different AMS MUSIC disk with Player. **FREE!!**

- - or - -

c. Your choice of any one of the above public domain disks -- #1, #2, #3, #4, #5, #6, #7, or #8 (specify one). **FREE!!**

ADVANCED MUSICSYSTEM II
by LEE ACTOR
Allows you to create music with your Atari computer! All new machine code.
* Control over pitch duration, envelope dynamic level, meter, tempo and key.
* 4 independent voices
* 5 ½ octaves per voice
* Save up to 8200 notes
* Custom DOS
* FULL instructions
* 24K disk or 16 K cassette

Originally $29.95 **Only $14.95**

ORIGINAL ADVENTURE
by Bob Howell
For all Atari computers. The Original Colossal Cave Adventure faithfully reproduced from the 'main-frames'. This is the one that launched the whole Adventure craze of today!
* Two mazes
* 130 rooms
* Deadly Dragons
* Nasty Dwarves
* Tenacious Troll
* The Pirate & More!
* 86 coded hints
* SAVE/RESUME
* 40K disk or 32K tape

Originally $24.95 **Only $14.95**

ESI WRITER ! At last a brand-new Word Processor that has more features and is easier to use than anything else available for the Atari. Easy for the beginner to use, it asks questions and remembers the answers. ESI WRITER is so sophisticated that it has about 50 features we don't even have room to mention! Works with ANY Atari.

* Reads any text file * Built in Help screen
* Very fast! * WORKS WITH ANY PRINTER
* Instant top, bottom or text location without scrolling!
* Every printer feature * DISK ONLY (Any Atari)

TRUST US ON THIS ONE! YOU WILL LOVE IT!

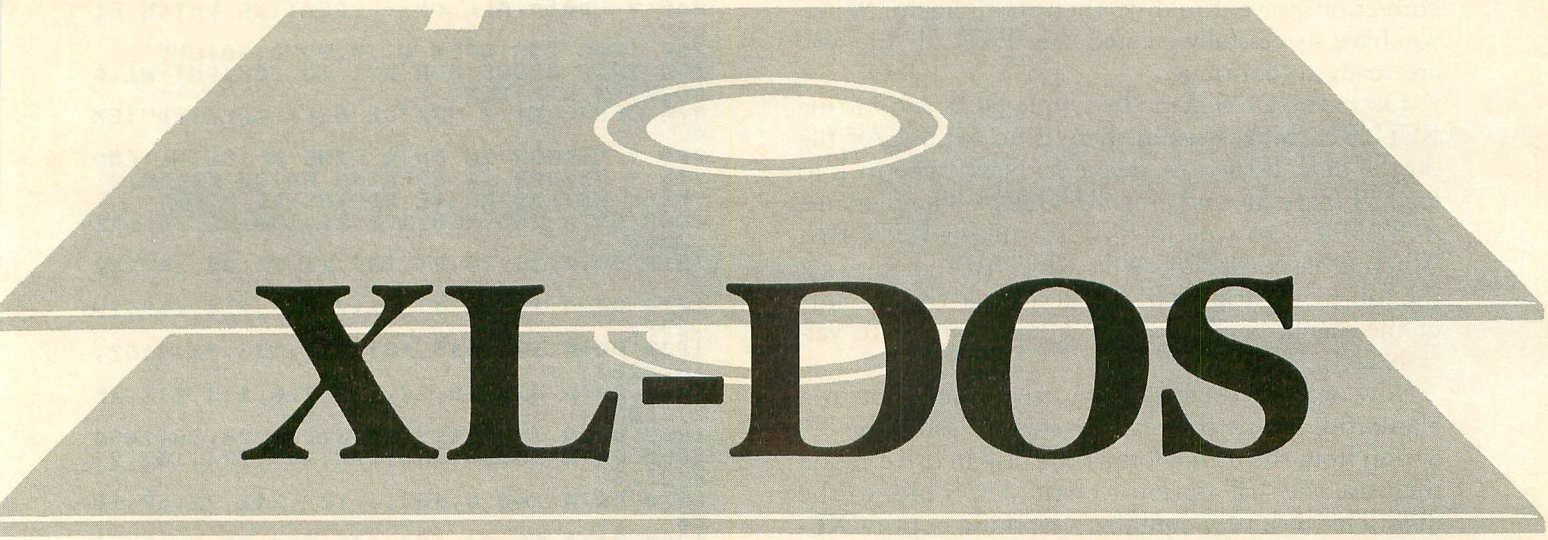
Originally $49.95 ***LotsaBytes*** **price $24.95**

Full 100% Replacement guarantee. Any disk found to be defective will be replaced free and we will also refund your return postage. All orders shipped by First Class U.S. Mail. Add $1.95 shipping and handling for 1 to 5 disks. Add $2.95 for 6 to 12 disks. California residents add 6% sales tax. Outside of U.S.A. and Canada add 15%. U.S. Funds only. We accept checks or Money Orders. Sorry, no COD or Charge Cards. Allow two weeks for personal checks to clear.

LOTSABYTES
15445 Ventura Blvd., Suite 10H
Sherman Oaks, CA 91413

Atari is the registered trademark of Atari, Inc.

CIRCLE #122 ON READER SERVICE CARD



XL-DOS

16K Disk

by Robert Luce

XL-DOS is an end-user modification to Atari's DOS 2.0S for XL computers with 64K memory. It stores both DUP.SYS and the MEM.SAV file in the normally-*unused* top 16K of memory. Both DOS functions are rewritten to allow instantaneous loading of the DUP.SYS file from memory, with MEM.SAV always active.

Maybe I'm just impatient, but that long wait for DUP.SYS to load in has always bothered me. It's always right after I hit the RETURN key that I realize I've forgotten to save my BASIC program or assembler source code before using the DOS command. I know. . .using a MEM.SAV file would avoid this situation, but then it takes an incredible amount of time just to *get* the DOS menu. Yes, I'm still using DOS 2.0S. Most of my files use DOS 2, and, after looking at DOS 3, I think I'll wait for DOS 4 (now in the rumor stage).

If you use DOS 2, have an XL computer, and are as impatient as I am, I think you're going to love **XL-DOS**!

Working with XL-DOS.

If you think you might be interested in using **XL-DOS** but don't care how it works, I suggest you skip directly to the section on typing it in. For those of you who want to know more, read on. . .

As mentioned above, **XL-DOS** uses the top 16K in a 64K XL computer to store the DUP.SYS and MEM.SAV files, so there's *no waiting* for DUP.SYS to load into memory. After you type *DOS RETURN*, you are immediately presented with the DOS menu. Not only that, but **XL-DOS** also handles the MEM.SAV function using this memory, so your program remains intact. Of course, some menu options, such as disk duplication, require use of the program area, but DOS 2 will ask your permission first, as usual when MEM.SAV is active.

When the disk is booted, you may notice that it's taking a bit longer than usual. This is because the DUP.SYS file must be read into memory. After it's in memory, the Operating System ROM is turned off, exposing the RAM underneath. The DUP.SYS file is then moved up to this hidden RAM and stored in the range from $E400-$F8FF. Because we are going to be switching back and forth between ROM and RAM at these addresses, the Atari character set data must also be copied to this RAM memory to avoid a very unsightly display. The area of memory used is $E000-$E3FF, the "normal" address of the character set data. All we do is copy from ROM to RAM at the same addresses. Finally, **XL-DOS** MEM.SAV routines use the RAM memory from $C000-$CFFF and from $D800-$DEFF. The range from $D000-$D7FF is always mapped to the hardware I/O, so it is unusable for our purposes.

Typing in XL-DOS.

First of all, you should have on hand a blank disk or a disk that can be formatted. Boot up your computer using DOS 2. If you have a 1200XL, make sure that the BASIC cartridge is installed. Next, type in the BASIC program and SAVE it to the disk before running it. There is an error-checking routine for the

data lines, so if you get the message *Error in data line #. . .*, check that line against the listing. Make your corrections, save it to disk, then run it again. When you have successfully created the "PATCH" file, you are ready to continue.

Go to the DOS menu by typing *DOS* and hitting RETURN. Make certain there is no MEM.SAV file on the disk. If there is, delete it. Next, type in *L* to load a binary file and press RETURN. The filename is "PATCH," so type in *PATCH* and hit RETURN. When it has loaded in, strike RETURN again to get the full menu displayed. If you look at the top line of the menu, you'll see that it has changed to version 2XL.

Now, remove the disk from drive 1 and put in a blank disk or one that can be erased. Using the "I" option from the menu, format the disk in drive 1, and then use the "H" option to write DOS files to the disk. This disk now contains a working copy of **XL-DOS**. Now, turn OFF the computer. *Do not* return to the cartridge or use any other functions. If you do, the machine will most assuredly lock up.

To use your new **XL-DOS**, simply reboot the computer using the disk you have just created. Type in a couple of lines of BASIC, then type DOS and hit RETURN. Fast, huh? Check out the disk directory. Nothing but DOS.SYS and DUP.SYS! Now, return to the cartridge and type *LIST*. Your BASIC program is still there!

A cautionary note.

XL-DOS doesn't use any more memory than DOS 2. It does, however, use the zero page addresses $D4-$D7 normally reserved for the floating point routines. Since **XL-DOS** uses these addresses when the floating point routines are not being used, there should be no incompatibility problems, unless a program is also using these zero page addresses. One example is the **AtariWriter** printer driver, which would not work with **XL-DOS**, but, since the **AtariWriter** never uses DUP.SYS or MEM.SAV, there is no reason to use **XL-DOS** with it, is there? Make sure to clearly mark this disk "**XL-DOS**." You may make additional copies simply by booting the **XL-DOS** disk and using the "H. Write DOS files" option. □

Listing 1.
BASIC listing.

```
100 GRAPHICS 0:POSITION 13,1:? "XL-DOS
 CREATOR"
110 ? :? "NOW CHECKING YOUR DATA LINES
:"
120 DIM F$(308)
130 LINCT=0:DONE=0
140 FOR X=1 TO 8:READ DATA
150 LINCT=LINCT+DATA
160 IF DATA=256 THEN POP :DONE=1:GOTO
190
170 F$(LEN(F$)+1)=CHR$(DATA)
180 NEXT X
190 READ CKSUM:CLINE=PEEK(183)+256*PEE
K(184)
200 LINCT=LINCT+CLINE
210 IF LINCT<>CKSUM THEN 280
220 ? "LINE #";CLINE;" IS OK!"
230 IF  NOT DONE THEN 130
240 ? "DATA ALL OK - CREATING PATCH FI
LE"
250 TRAP 290:OPEN #1,8,0,"D:PATCH"
260 TRAP 40000:FOR X=1 TO 308:PUT #1,A
SC(F$(X,X)):NEXT X
270 CLOSE #1:? "PATCH FILE CREATED":EN
D
280 ? "ERROR IN DATA LINE #";CLINE:END
290 ? "ERROR IN OPENING PATCH FILE":?
"DISK MAY BE WRITE PROTECTED.":END
1000 DATA 255,255,122,21,124,21,76,192
,2066
1010 DATA 23,70,23,137,23,32,85,24,142
7
1020 DATA 169,0,133,212,133,214,169,29
,2079
1030 DATA 133,215,169,192,133,213,162,
16,2263
1040 DATA 32,119,24,169,216,133,213,16
2,2108
1050 DATA 7,32,119,24,32,70,24,96,1454
1060 DATA 169,0,133,212,169,224,133,21
3,2313
1070 DATA 160,0,162,3,177,212,72,32,18
88
1080 DATA 85,24,104,145,212,32,70,24,1
776
1090 DATA 200,208,241,230,213,202,16,2
36,2636
1100 DATA 96,182,23,251,23,240,73,32,2
020
1110 DATA 70,23,206,158,23,48,65,32,17
35
1120 DATA 170,25,32,105,23,169,255,141
,2040
1130 DATA 158,21,141,157,21,162,16,169
,1975
1140 DATA 47,157,68,3,169,24,157,69,18
34
1150 DATA 3,32,164,21,32,85,24,162,167
3
1160 DATA 21,169,0,133,212,133,214,169
,2211
1170 DATA 31,133,215,169,228,133,213,3
2,2324
1180 DATA 119,24,32,70,24,169,0,141,17
59
1190 DATA 157,21,96,19,24,39,24,32,160
2
1200 DATA 85,24,169,0,133,214,133,212,
2170
1210 DATA 169,228,133,215,169,31,133,2
13,2501
1220 DATA 162,21,208,18,58,24,135,24,1
870
1230 DATA 32,119,24,32,70,24,206,157,1
894
1240 DATA 21,76,117,32,32,102,24,88,17
32
1250 DATA 169,112,141,14,212,165,16,14
1,2220
1260 DATA 14,210,96,120,169,0,141,14,2
024
1270 DATA 212,141,14,210,173,1,211,41,
2273
1280 DATA 254,76,107,24,173,1,211,9,21
35
1290 DATA 1,141,1,211,96,234,234,234,2
442
1300 DATA 234,32,180,25,96,160,0,177,2
204
1310 DATA 214,145,212,200,208,249,230,
213,2981
1320 DATA 230,215,202,208,242,96,63,25
,2601
1330 DATA 97,25,32,85,24,169,0,133,189
5
1340 DATA 212,133,214,169,29,133,213,1
69,2612
1350 DATA 192,133,215,162,16,32,119,24
,2243
```

(Listing continued on page 53)


IT'S HERE

COL

INTRODUCING OKIMATE 10... THE FIRST

The printer in a class by itself.

It's here! The new OKIMATE 10 Personal Color Printer. The first color printer that lets you show off and tell all. The printer that lets you print all the information you can create with your Atari® or Commodore® computer. But with the remarkable ability to create original drawings and graphics as well, in over 26 beautiful colors.

A class act! The OKIMATE 10 gives you crisp, clean term papers, school reports and homework. Word processing capability means everything you do can be printed letter quality in minutes, instead of typed in hours. OKIMATE 10 color gives you the opportunity to print graphs, charts and pictures from popular graphics and drawing programs. OKIMATE 10's brilliant color means you'll shine, every time.

OKIMATE 10 feels right at home. Anywhere.

A special PLUG 'N PRINT™ package lets you plug your new OKIMATE 10 into your Atari or Commodore computer. And print. It's that easy. In minutes you'll be printing everything from soufflé recipes to needlepoint patterns. Party invitations to kitchen inventory. Love letters to gardening directions. At 240 remarkable words per minute. And not just in black and white, but in over 26 brilliant colors!

Financial statements will keep you tickled pink for very little green.

If you use your personal computer to keep track of mortgage payments, tuition payments, balance your checkbook or jump ahead of the Dow Jones', there's good news for you. You'll find that the new OKIMATE 10 gets down to business quickly. And easily.

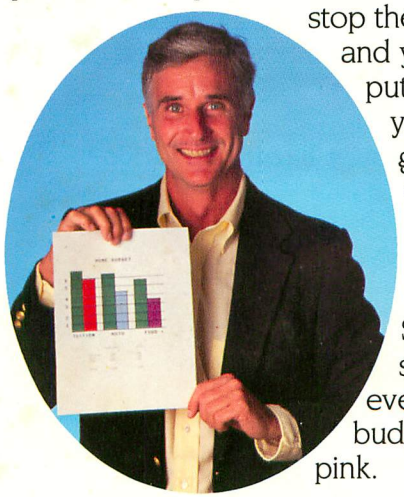
A "Learn-to-Print" diskette and tape shows you how to set up your new personal color printer and start printing. A complete OKIMATE 10 Handbook will show you how you can take your imagination to places it's never been before.

*Atari is a registered trademark of Atari Inc. *Commodore is a registered trademark of Commodore Business Machines, Inc.

OR

PERSONAL COLOR PRINTER UNDER $250.

And while your imagination is soaring, you'll be glad to know that your new printer can keep right up with it! The new OKIMATE 10 is built with the same tradition of quality and manufacturing excellence that has made Okidata the most respected name in computer printers. Okidata craftsmen specially designed and engineered the new OKIMATE 10 to be incredibly small and lightweight. And they made it quiet as a whisper. But their imagination didn't stop there. To help you and your personal computer keep within your personal budget, they made the OKIMATE 10 available at retailers everywhere for less than $250. Something that should make every personal budget tickled pink.



Color your world.

If you've been playing games on your personal computer, now you can get serious and still have fun. The new OKIMATE 10 is completely compatible with a variety of software packages that will run on your Atari and Commodore with a simple disk drive. Just load and you're off and running. Plotting charts. Designing special graphs. Creating original illustrations and pictures. Drawing special graphics. And printing them all beautifully for everyone. On most kinds of paper. In over 26 beautiful colors!

QUESTIONS & ANSWERS

Q: Why do I need a printer?

A: You might as well ask, "Why do I need crayons?" When it comes to communicating, "putting it on paper" is still the best way to get your message across. You can have lots of computer equipment, but without the OKIMATE 10, it doesn't mean very much. Unless you get your letter, report, term paper or party invitation off the screen and down on paper, nobody's going to see it.

Q: What makes the OKIMATE 10 better than any other printer?

A: Because the OKIMATE 10 is unlike any other printer. First, it prints in COLOR. Up to 26 beautiful colors. Second, it prints up to 240 words a minute, so quietly you can talk in a whisper right next to it and still hear every word! And third, it prints letter quality, every time.

Q: What about graphics and pictures?

A: The OKIMATE 10 does it all. Graphs, charts, symbols, pictures, illustrations, and special drawings! With a compatible drawing package, anything you create on your screen can be printed in full color; a disk drive is required for color screen printing.

Q: What kind of paper can I use?

A: Just about any kind of smooth paper you want. From continuous feed computer paper to single sheets. From mailing labels to plastic acetate for overhead transparencies, the OKIMATE 10 prints crisp, clean, colorful images you'll be proud to send to friends, teachers, business associates, or frame and hang right in your own living room!

Q: Is the OKIMATE 10 easy to use?

A: As easy as "PLUG 'N PRINT!" No other printer is easier to use than the OKIMATE 10. Connecting the printer to your Commodore or Atari computer is, literally, a snap. The exclusive PLUG 'N PRINT package snaps into the printer. One cable connects it directly to your computer or disk/tape drive. Turn it on and you're in business. Once your OKIMATE 10 is up and running, the "Learn-to-Print" software program (included) teaches you printer basics—the "Color Screen Print" disk (also included) automatically prints everything on the screen in a single stroke. As a matter of fact, most of your printing can be done with just one command.

Q: What's the printer like in operation?

A: In one word: easy! Incredibly easy! The ribbon comes in a "Clean Hands" cartridge. So it's as easy to change as the tape in your audio cassette player.

Q: What about reliability?

A: Okidata has built the reputation of its complete line of printers on quality, dependability and rugged construction. The OKIMATE 10 is no exception. Don't let its light weight and compact size fool you. This printer is not a toy. It's a workhorse.

OKIDATA
an OKI AMERICA company

Available at retailers everywhere.

```
1360 DATA 169,216,133,215,162,7,32,119
,2413
1370 DATA 24,32,70,24,96,50,31,52,1749
1380 DATA 31,88,76,32,256,1863
```

CHECKSUM DATA.

(see page 90)

```
100 DATA 171,475,815,559,361,848,273,5
50,782,571,84,905,606,355,318,7673
250 DATA 469,840,402,92,894,770,196,81
0,817,638,62,606,490,351,808,8245
1100 DATA 324,565,759,811,552,255,773,
630,491,200,521,851,362,321,511,7926
1250 DATA 623,301,490,524,339,359,139,
825,278,829,783,814,76,405,6785
```

Assembly language listing.

```
**********************
**                    **
**      XL-DOS        **
**    BY BOB LUCE     **
**      5/31/84       **
**                    **
**********************
;
;O.S. EQUATES
;
ICBAL    =   $0344
ICBAH    =   $0345
NMIEN    =   $D40E
IRQEN    =   $D20E
CHSET    =   $E000
;
;D.O.S. EQUATES
;
DUPFLG   =   $159D
OPT      =   $159E
SFLOAD   =   $15A4
MEMFLG   =   $179E
RRDUP    =   $1801
DUPSYS   =   $192F
CLOSX    =   $19AA
CLOS20   =   $19B4
DOS      =   $2075
;
;MISC EQUATES
;
MSAVF    =   $1D00
DOSSTART =   $1F00
MSAVT1   =   $C000
MSAVT2   =   $D800
SAVEDOS  =   $E400
TO       =   $D4
FROM     =   $D6
;
; DOS PATCHES
;
         *=  $157A
;
         JMP DUPINIT
;
         *=  $1746
;
MWRITE   JSR ON       ;AUX. MEM ON
         LDA #$00     ;SET UP
         STA TO       ;INDIRECT
         STA FROM     ;POINTERS
         LDA # >MSAVF
         STA FROM+1
         LDA # >MSAVT1
         STA TO+1
         LDX #$10     ;MOVE LOWER
         JSR MOVE     ;10 PAGES
         LDA # >MSAVT2 ;UPDATE INDIRECT
         STA TO+1     ;POINTER
         LDX #$07     ;MOVE TOP
         JSR MOVE     ;7 PAGES
         JSR OFF      ;AUX MEM OFF OS ON
         RTS
;
; COPY CHR SET TO AUX MEMORY
;
MOVECHR  LDA # <CHSET ;SET UP
         STA TO       ;INDIRECT
         LDA # >CHSET ;POINTERS
         STA TO+1
         LDY #$00     ;INIT REGS
         LDX #$03
LP1      LDA (TO),Y   ;GET CHR
         PHA          ;SAVE ACC
         JSR ON       ;AUX MEM ON
         PLA          ;RESTORE ACC
         STA (TO),Y   ;& STORE
         JSR OFF      ;AUX MEM OFF
         INY
         BNE LP1      ;IF Y=0 THEN
         INC TO+1     ;INC HI BYTE
         DEX          ;DEC LOOP INDEX
         BPL LP1
         RTS
;
         *=  $17B6
;
         BEQ RRDUP
GOOD     JSR MWRITE   ;DO MEM.SAV
         DEC MEMFLG   ;SHOW MEM SAVED
         BMI RRDUP    ;ALWAYS
;
; COLDSTART ROUTINE
;
DUPINIT  JSR CLOSX    ;CLOSE IOCB
         JSR MOVECHR  ;MOVE CHR SET
         LDA #$FF     ;CONDITION DOS
         STA OPT      ;FLAGS TO LOAD BUT
         STA DUPFLG   ;NOT RUN DUP.SYS
         LDX #$10     ;USE IOCB 1
         LDA # <DUPSYS ;TO LOAD DUP.SYS
         STA ICBAL,X
         LDA # >DUPSYS
         STA ICBAH,X
         JSR SFLOAD   ;DO LOAD
         JSR ON       ;AUX MEM ON
         LDX #$15     ;MOVE DUP
         LDA #$00     ;TO AUX MEM
         STA TO
         STA FROM
         LDA # >DOSSTART
         STA FROM+1
         LDA # >SAVEDOS
         STA TO+1
         JSR MOVE
         JSR OFF      ;AUX MEM OFF
         LDA #$00     ;SHOW DUP NOT
         STA DUPFLG   ;IN MEMORY
         RTS
;
         *=  $1813
;
RRDUP1   JSR ON       ;AUX MEM ON
         LDA #$00     ;MOVE DUP DOWN
         STA FROM     ;FROM AUX MEM
         STA TO
         LDA # >SAVEDOS
         STA FROM+1
         LDA # >DOSSTART
         STA TO+1
         LDX #$15
         BNE CONT
;
         *=  $183A
;
CONT     JSR MOVE
         JSR OFF      ;AUX MEM OFF
         DEC DUPFLG   ;SHOW DUP IN MEMORY
         JMP DOS      ;AND RUN IT
;
; ROUTINES TO ENABLE/DISABLE
; AUX MEM ($C000-$FFFF)
;
OFF      JSR AUXOFF
         CLI
         LDA #$70
         STA NMIEN
         LDA #$10
         STA IRQEN
         RTS
;
ON       SEI
         LDA #$00     ;NO INTERRUPTS!
         STA NMIEN
         STA IRQEN
AUXON    LDA $D301
         AND #$FE
         JMP ST
;
AUXOFF   LDA $D301
         ORA #$01
ST       STA $D301
         RTS
         NOP
         NOP
         NOP
         NOP
;
; MEMSVQ TELLS DOS IF MEM.SAV
; IS ACTIVE (Y=1)
;
MEMSVQ   JSR CLOS20   ;CLOSE IOCB
         RTS          ;RETURN Y=1
;
; MOVE ROUTINE
; CALLING ROUTINE SETS UP 'TO'
; AND 'FROM' AND ON ENTRY X
; IS NUMBER OF PAGES TO MOVE
;
MOVE     LDY #$00
LP       LDA (FROM),Y
         STA (TO),Y
         INY
         BNE LP
         INC TO+1
         INC FROM+1
         DEX
         BNE LP
         RTS
;
; RESTORE SAVED MEMORY
;
         *=  $193F
;
LDMEM1   JSR ON       ;AUX MEM ON
         LDA #$00     ;THEN MOVE
         STA TO       ;SAVED MEMORY
         STA FROM     ;BACK TO IT'S
         LDA # >MSAVF ;ORIGINAL
         STA TO+1     ;LOCATION
         LDA # >MSAVT1
         STA FROM+1
         LDX #$10
         JSR MOVE
         LDA # >MSAVT2
         STA FROM+1
         LDX #$07
         JSR MOVE
         JSR OFF      ;AUX MEM OFF
         RTS
;
; CHANGE DOS MENU TITLE BAR
; TO REFLECT PATCHED DOS
;
         *=  $1F32
;
         .BYTE "XL "
;
         .END
```

THE MMG BASIC COMPILER

ATARI OWNERS FINALLY!!

The BASIC Compiler for Every Need and Every Program!

Tired of using those other BASIC compilers that don't do the job for you? Is there a long list of valid BASIC commands that they don't support? Or don't they compile to true 6502 machine language for maximum speed? Or do you have to rewrite your whole BASIC program just to find out that it won't run when compiled?

Announcing
THE MMG BASIC COMPILER

THE FIRST COMPLETE BASIC COMPILER FOR THE ATARI COMPUTERS THAT PRODUCES NATIVE 6502 CODE

What is a BASIC compiler?

BASIC, as we all know, is an easy-to-use language for ATARI computers. It's only disadvantage is that it's SLOW. For some types of functions, it seems to take BASIC programs forever to execute. We all know that the fastest language available is machine language, the language of ones and zeros. But don't worry! Now you don't have to learn a whole new language just to have programs execute with machine language speed.
The MMG BASIC COMPILER takes your BASIC program and converts it to machine language for you. Furthermore, this machine language program will autorun, simply by naming it AUTORUN.SYS, putting it on a disk with the DOS 2.05 files on it, and turning on your computer with that disk in your drive.

What will a compiler do for me?

Using the MMG BASIC COMPILER, you can program in BASIC, the same BASIC you already know, and get your program up and running. Then the MMG BASIC COMPILER will convert your BASIC program for you, producing lightning-fast programs to rival those of the professionals. Imagine moving a player from the top of the screen to the bottom in less than a second! Try that using other compilers! Imagine what your programs will be like when they're compiled to true 6502 machine language. The MMG BASIC COMPILER has been used to produce commercially available arcade-type games from BASIC source code, and can do the same for you!
MMG would even be interested in marketing your results! If you produce what you believe to be a marketable program, call us for details!

Can your compiler:

- compile to fast 6502 machine language, not slow pseudocode (P-code)?
- support trigonometric functions like ATN, COS, SIN?
- support mathematical functions like CLOG, EXP, LOG, RND, SQR?
- support RUN "D:PROGRAM"?
- support ATARI string handling like A$(2,4)="BOD"?
- support COMmon variables?
- support the POP command?
- support the LPRINT command?
- support either RAD or DEG calculations?
- support both integer and floating point arithmetic?
- operate in either single or true double density?
- allow DATA statements anywhere in your program?
- produce assembly language source code of your program for your own use?

The MMG BASIC COMPILER does!

The MMG BASIC COMPILER comes with both single and double density versions on the same disk, and is available from your local computer store, or send $99.95 plus $3.00 for shipping and handling to:

MMG Micro Software P.O. Box 131 Marlboro, NJ 07746 (201) 431-3472

Visa, MasterCard, or COD orders accepted. New Jersey residents please add 6% sales tax.

CIRCLE #124 ON READER SERVICE CARD

Cassette Compressor

16K Cassette or Disk

by Harold Johnson

ANALOG Computing has established a sort of tradition—each issue contains a machine language game. The maker program provided writes the actual game to disk or cassette, but when writing to cassette, it uses the slowest possible way—long inter-record gaps. This is the same as typing in *SAVE C:* instead of *CSAVE*.

One day I got tired of waiting for my cassette boot programs to load, so I wrote the **Cassette Compressor**. It will take any program not longer than the amount of free RAM and "compress" it by rewriting it and using short inter-record gaps. This causes the program to load faster and to be more reliable.

As a test for the **Compressor**, I unplugged all of my RAM boards except one, leaving me with only 16K of RAM. I then loaded in the program **Crash Dive!** since it was the longest program published by **ANALOG Computing** up to that time. It fit quite easily into the system, leaving thousands of bytes free. After rewriting, **Crash Dive** loaded in only 27% of the original time. It also took up about a quarter as much tape. This gives even more reliability, since there is now less tape for errors to develop on.

Using the program.

First, type in the listing. It's pretty short, so you shouldn't have any problems. Next, CSAVE it a couple of times, so that you won't have to type it again the next time you want to use it. Now, just run the program. You will be prompted to insert the source cassette and press a key. The console will also beep once, reminding you to depress the PLAY button on your recorder. At this time, put the cassette containing the program to be compressed in, and rewind or advance the tape to the proper counter number.

Now, push down the PLAY button on the recorder and press RETURN. The computer will read in the source file, during which there will be a fairly long delay. When this is finished, the console will beep twice, and you will be asked to insert the destination cassette and press a key. Now, put in the destination cassette, rewind or advance the tape to the proper location, press the PLAY and RECORD buttons down on the recorder, and press RETURN. The computer will rewrite the whole file. When the computer finishes, the console will beep again, and you will be asked if you would like to write the same file again.

If you enter a *Y*, the computer will rewrite the file from memory (why read it again?), and this cycle will repeat. If you enter an *N*, you will be asked one more thing—*Compress another file?* If you enter a *Y* here, the program will rerun. If you enter an *N*, you will be returned to BASIC.

Notes.

There are just a couple of tricks employed in this program. In Line 10, the input/output noise is disabled via the POKE to location 65. POKEing a *1* to this location will turn it back on again (although I don't know why anyone would want to). This is documented in the regular BASIC reference manual from Atari. I have no idea why no one seems to use it.

Another trick is disabling the BREAK key. This just makes sure that, if you accidentally hit it, it won't mess up the I/O. This is done in Line 20 by the last two POKEs to memory locations 16 and 53774. If you use this disabling trick, however, remember that hitting SYSTEM RESET or changing graphics modes re-enables the BREAK key.

The last trick is in Lines 100 and 120. I use an input from IOCB #0. This channel is always open for BASIC itself, so, instead of opening channel 1, I use this. It also allows all screen editing features to be enabled without causing errors. Pretty neat, huh? □

Listing 1.

```
10 POKE 752,1:POKE 65,0:? "Insert sour
ce cassette and press any  key.":A=FRE
(0):IF A)32767 THEN A=32767
20 DIM A$(A-128):OPEN #1,4,0,"C:":POKE
 16,64:POKE 53774,64
30 ? "Reading file...please wait.":TRA
P 50:A=1
40 GET #1,B:A$(A)=CHR$(B):A=A+1:GOTO 4
0
50 CLOSE #1:? "Insert destination cass
ette and press any key."
60 OPEN #1,8,128,"C:"
70 ? "Writing file...please wait."
80 ? #1;A$;:CLOSE #1
90 ? "Done.  Write same file again (Y
/N)"
100 INPUT #0;A$:IF A$="Y" THEN 50
110 ? "Compress another file (Y/N)"
120 INPUT #0;A$:IF A$="Y" THEN RUN
130 END
```

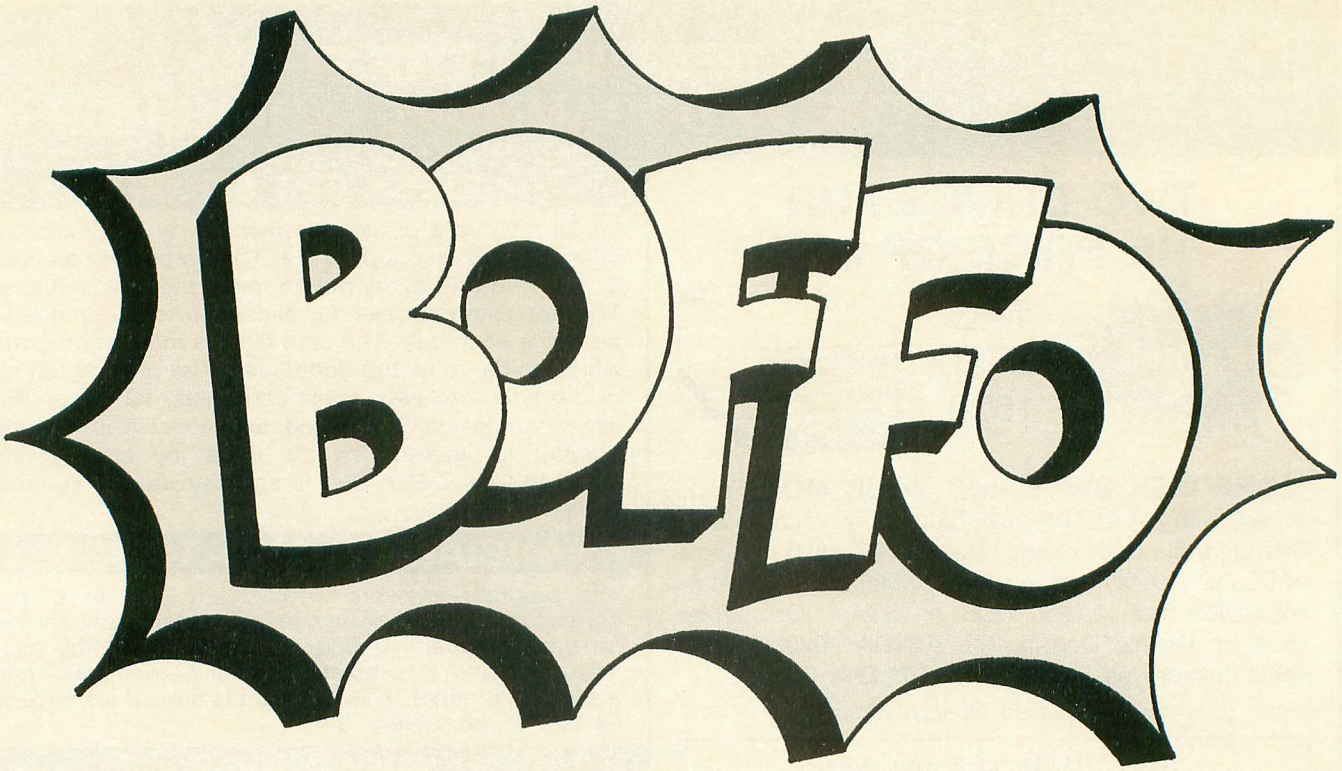
•

CHECKSUM DATA.

(see page 90)

```
10 DATA 232,907,420,648,697,539,79,410
,92,987,704,214,35,5964
```

•



16K Cassette or Disk

by Tom Hudson

Editor's note: This issue, Tom Hudson's **Boot Camp** *gives way to* **BOFFO**, *a utility for assembly language programmers.* **Boot Camp** *readers are encouraged to study the use of* **BOFFO** *for next issue's* **Boot Camp** *column, which will examine the use of assembly language subroutines in BASIC.*

For the assembly language programmer, there's probably no worse drudgery than converting object code into DATA statements for use in BASIC programs. This was a major concern for the **ANALOG Computing** programming staff (all of us being basically lazy), so **BOFFO** was developed and improved, a little bit at a time, over a period of about a year.

BOFFO will convert just about any file into DATA statements, so that it can be reconstructed or POKEd into memory by a BASIC program. The DATA statements can be created with a simple checksum if desired—a good idea for magazine programs, which are easily mistyped.

BOFFO is intended primarily for *experienced* programmers who would like an easy shortcut in generating BASIC DATA.

Using the program.

After typing in the accompanying program listing, check your typing with **D:CHECK2** and SAVE the program to disk.

When RUN, **BOFFO** will ask you several questions about what you want to do. Here's a brief explanation of each:

```
OBJ/DATA file?
```

If you're converting an assembler object file, type O and press RETURN; otherwise, type *D* and press RETURN.

Typing O will force **BOFFO** to read the machine language load header bytes, placing only actual memory bytes into the DATA statements. Note that the memory in your object file *must* be contiguous. That is, each group of loaded data must start loading in the first byte after the previous group. If not, an error message will result. Use the O option for object files *only*!

Typing *D* will cause **BOFFO** to place *every byte* of the file into the DATA statements. Use the D option for text files or object files which you'd like to completely reconstruct.

```
OBJECT filename?
 DATA filename?
```

Depending on whether you chose the O or D option, one of the above questions will appear. In either case, type in the name of the file you wish to convert. If you enter a filename without

a device, D1: is assumed.

```
BASIC filename?
```

At this point, enter the name of the file you want to create. This file will contain the BASIC DATA statements which correspond to the input file. If you have a printer, you can send the output to it by typing *P:*. Usually, you'll want to send the output to a disk file, like so: D:FILENAME.EXT.

Most errors in filename entry will be trapped, and you will be asked to try again.

```
Starting lineno?
```

Type in the line number you want the DATA statements to start on.

```
Line increment?
```

Type in the line number increment. For example, if you use a starting line number of 1000 and an increment of 5, the line numbers will proceed: 1000, 1005, 1010, and so on.

```
Decimal/Hex?
```

Typing *D* here will cause **BOFFO** to generate decimal (base 10) numbers in the DATA statements. This is the form of data that is used most often.

Typing *H* here will generate *hexadecimal* DATA statements, like those seen in most of **ANALOG Computing**'s assembly language games. The hexadecimal form saves a great deal of memory, but you must decode the hex values, which is a slow process. Look at any of **ANALOG Computing**'s assembly games to see how this is done.

```
Bytes per line?
```

Simply enter the number of data bytes you want on each DATA line. Usually, 25 is the maximum for decimal values, and 45 the maximum for hex. Of course, you could put only one byte on each line, but this would waste memory.

```
Checksum (Y/N)?
```

If you want **BOFFO** to generate checksum values, type *Y* and press RETURN. If not, type *N*.

The checksums generated by **BOFFO** are simple, modulo 1000 numbers. To get the checksum, add the value of each byte in the line to a counter. If the value of the counter ever exceeds 999, subtract 1000 from it. The checksum is placed after the last byte in each DATA line; simply READ it after you have read all the data bytes. Figure 1 shows the code necessary to process checksums for DATA statements containing 25 bytes per line.

```
10 TRAP 100
20 FOR X=1 TO 25
30 READ BYTE
40 TOTAL=TOTAL+BYTE
50 IF TOTAL>999 THEN TOTAL=TOTAL-1000
60 NEXT X
70 READ CHKSUM
80 IF TOTAL<>CHKSUM THEN ? "DATA ERROR
!"
90 GOTO 20
100 IF PEEK(195)=6 THEN ? "DATA OK!":E
ND
110 ? "DATA ERROR:";PEEK(195):END
```

Figure 1.

Note that the checksum values carry on from one DATA line to the next. In this way, if a line is missing, it can be detected and fixed.

You don't need checksums for your own programs. I usually only generate checksums if the finished program is going to be printed in **ANALOG Computing**, or where someone will have to type it in. Using checksums definitely helps the end user reduce typing errors.

Using the output.

After you've answered all **BOFFO**'s questions, it will generate the BASIC DATA and write it to the specified output file. When the READY prompt appears, you're ready to use the DATA.

To retrieve your newly-created BASIC code, you must ENTER it. The ENTER command is similar to LOAD, except that the code is *added* to whatever BASIC code is in memory. Type *ENTER"D:FILENAME.EXT"*. The DATA code will be brought into memory, ready for you to use.

At the end of the DATA statements, **BOFFO** creates a REM statement which lets you know how many bytes were converted. This is a convenient way to tell how many bytes the program needs to READ.

Some final notes.

An important restriction in **BOFFO** is that data blocks in object data files *must* be contiguous. Otherwise, there is no way to determine where in memory the code resides.

Remember that **BOFFO** is meant for fairly advanced programmers. You should know what you want to do with the code once it is converted to DATA. If you don't know what "object code" is, **BOFFO** isn't for you.

I think most assembly language programmers will agree that **BOFFO** is a utility that's worth typing in. It has saved me literally hundreds of hours—and many headaches—when setting up **ANALOG Computing**'s assembly language games. Why translate all those bytes, when your Atari can do it for you? □

(BOFFO listing starts on page 61)

ATARI®

600XL ..... CALL

800XL ..... CALL

DISK DRIVES

Rana 1000 ..... $298
Astra 2001 ..... $549
Indus GT $315
Trak AT-D2 ..... $388
Trak AT-1 ..... $319
Trak AT-D4 ..... Call
Astra 1620 (Dual) ... $499
Percom ..... Call
Atari 1050 ..... $349

MEMORIES

Microbits 64K (XL) .. $115
Mosaic 48K (400) $98
Mosaic 64K (400/800) Call
Mosaic 32K $68
Atari 64K (600XL) .... Call

INTERFACES

Axiom 846 ..... Call
Ape Face ..... Call
Atari 850 (In Stock) .. $169
Interfast 1 ..... $150
Microbits 1150 ..... Call
R-Verter ..... Call

DIRECT PRINTERS

Axiom AT-100 ..... $219
Atari 1027 ..... $285
Axiom 550 AT $319
Axiom 700 AT $469
Atari 1025 ..... $395

DIRECT MODEMS

Microbits 1000C $128

OTHER ATARI

400 Keyboard ..... Call
Koala Pad $67
Chalkboard Pad $75
Bit-3 80 Column $228
ATR-8000 (64K) $489
ATR-8000 (16K) $359
Alien Voice Box $98
1010 Recorder $74

ATARI SOFTWARE

MISCELLANEOUS

Syn Calc (D) $59
Syn File (D) $59
Syn Trend (D) $48
Syn Com (D) $29
Syn Chron (D) $29
Decathlon (R) $29
Drols (D) $23
Gyruss (R) $31
Heist (D) $23
Bruce Lee (C/D) $27
Universe (D) ..... Call
Questron (D) $34
Koala Coloring I $20
Koala Logo Design $27
Bumble Games (D) $27
Miles Accounting ..... Call
World Gtst. Baseball $23
Gridrunner (R) $20
Sargon II (C/D) $23
Millionaire (D) $34
Castle Wolfenstein (D) $20
Odesta Chess (D) $46
Financial Wizard (D) .. $41
Ultima III (D) $39

ADVENTURE INT'L

Adv. 1-12 each (C) $18
Preppie (C/D) $20
Preppie II (C/D) $23
Diskey (D) $33
Sea Dragon (C/D) $23
Lunar Lndr . (C) $11 (D) $15
Galactic Empire (C) .. $14

ADVENTURE INT'L

Ultra Disassembler (D) $33
Diskey (D) $33
Adv. 1-12 (each) (C) ... $18
Saga 1-12 (each) (D) .. $27

ATARI

Atari Writer (R) $68
Paint (D) $30
Microsoft Basic II (R) . $64
Visicalc (D) $139
Home File Mgr (D) $36
Assembler Editor (R) . $44
Qix (R) $32
Dig Dug (R) $32
Atari Logo (R) $72
Ms. Pac Man (R) $33
Joust (R) $33
Donkey Kong Jr. (R) $35
Computer Chess (R) . $24
Galaxian (R) $30
Defender (R) $30
ET $34
Pac Man (R) $30
Centipede (R) $30
Caverns of Mars (D) .. $28
Star Raiders (R) $30
Conv. Lang. Ea. (C) ... $42
Asteroids (R) $27
Space Invaders (R) ... $27
Missile Command (R) . $27
Telelink (R) $21
Family Finance (D) ... $35
Prog. 1 (C) $18

AVALON HILL

Telengard . (C) $16 (D) $19
Close Assault (C) 20 (D) 23
TAC (D) $27
Space Station Zulu .....
..... (D) $17 (C) $14
Flying Ace . (D) $21 (C) $18
GFS Sorceress .....
..... (D) $23 (C) $20

AVALON HILL(Cont'd)

Moon Patrol (C) $17
B-1 Nuc. Bomber (C) .. $12
Legionnaire (C) $23
Empire of Overmind .....
..... (D) $23 (C) $20
Tanktics ... (D) $20 (C) $17
Comptr Stock & Bonds ...
..... (D) $17 (C) $14
Galaxy .... (D) $17 (C) $14
Nuke War (C) $12
Andromeda Conquest ...
..... (D) $16 (C) $13
Close Asslt (D) $23 (C) $20

BRODERBUND

Sky Blazer (D) $22
Bank St. Writer (D) ... $46
A.E. (D) $23
Arcade Machine (D) .. $39
Choplifter . (D) $23 (R) $29
Labyrinth (C/D) $20
Serpentine . (D) $23 (R) $27
Steller Shuttle (C/D) .. $17
Apple Panic (C/D) $20
Genetic Drift (C/D) ... $20
Oper. Whirlwind (D) .. $27
Choplifter . (D) $23 (R) $29
Match Boxes (C/D) ... $20

CDY CONSULTING

Pogoman (C/D) $27

CBS

Mountain King (R) ... $27
Boulders & Bombs (R) $27
Krazy (each) (R) $27

CONTINENTAL

Home Accountant (D) $48
Tax Advantage (D) ... $45

COUNTERPOINT SW Call

DATASOFT

Pooyan (C/D) $20
Teletalk (D) $33
Graphic Master (D) ... $23
Micropainter (D) $23
Zaxxon (C/D) $27
Pac. Coast Hwy (C/D) . $20
Clowns/Balloons (C/D) $20
Spell Wizard (D) .....
Canyon Climber (C/D) $20

EASTERN HOUSE

Monkey Wrench II ... $51

EDUCATIONAL SW

Tricky 1,2,3 or 4 ..... $15
Tricky 5-13 ..... $22

EDU-WARE

Spelling Bee (D) $27
Compu/Read .....
..... (D) $21 (C) $15
Compu/Math-FR .....
..... (D) $27 (C) $21
Compu/Math-Dec .....
..... (D) $27 (C) $21

EPYX

Dragon Riders (C/D) .. $27
Temple APS (C/D) $27
Jumpman (C/D) $27

FIRST STAR

Boulder Dsh (C/D) 20 (R) 27
Bristles (C/D) $20
Flip Flop (C/D) $20

GAMESTAR

Football (C/D) $21
Baseball (C/D) $21

Printers/Etc.

DIABLO

630 Letter Qual. $1648

SILVER REED

EXP 400 Ltr. Qual. ... $310
EXP 500 Ltr. Qual. ... $379
EXP 550 Ltr. Qual. ... $449
EXP 770 Ltr. Qual. ... $888

STAR

Gemini 10X $259
Gemini 15X $378
Delta 10 ..... $378
Delta 15 ..... $539
Radix 10 ..... $549
Radix 15 ..... $645
Power Type ..... Call

TOSHIBA

1340 ..... $788
1351 ..... $1299

NEC

3510 ..... $1259
3530 ..... $1259
3550 ..... $1598
7710/7730 ..... $1719

CITOH

Prowriter $335
Prowriter II $648
Starwriter $1098
Printmaster $1448

OKIDATA

82A ..... Call
84P ..... Call
92 ..... $409
93 ..... Call

DAISYWRITER

2000 ..... $995

MANNESMANN

160L $559
Spirit $309

JUKI

6100 ..... $419

ABATI

LQ 20P Ltr. Qual. $378

PANASONIC

1090 ..... $255
1091 ..... $319
1092 ..... $458

MONITORS

AMDEK

V300 G $139
V300 A $149
V310 G (IBM) $155
V310 A (IBM) $165
Color I+ $274
Color II+ $419
Color III $349
Color IV (IBM) $699

PRINCETON GRAPHICS

MAX 12 (Amber) $178
HX 12 (RGB) $489
SR 12 (RGB) $619

SUPER 5

100A (Amber) $99
500G (IBM with tilt) .. $126
500A (IBM with tilt) .. $126

SAKATA

SC100 (Color) $239
1000G (Green) $109

TAXAN

100 Green $118
105 Amber $128
210 RGB/Composite $288
400 RGB Med-Res. .. $309
415 RGB Hi-Res. $429
420 RGB Hi-Res.(IBM) $479

NEC

JB 1260 (Grn) $99
JB 1201 (Grn) $145
JB 1205 (Amber) ... $145

ZENITH

Green $95
Amber $99

MODEMS

NOVATION

J-Cat $99
Apple Cat II $259
D-Cat $149

HAYES

Smartmodem $209
Smartmodem 1200 .. $498
Micromodem II $259
Micromodem IIe $239

PROMETHIUS

Promodem 1200 .... $349

D = DISK
R = ROM (Cartridge)
C = CASSETTE
OPEN MON-SAT

COSMIC COMPUTERS™

727 BREA CANYON RD., SUITE 16
WALNUT, CA 91789

(800) 626-7642

PLEASE FOR ORDERS ONLY
SORRY, NO COD'S

IN CALIF. (714) 594-5204

FOR TECHNICAL INFO, ORDER INQUIRIES,

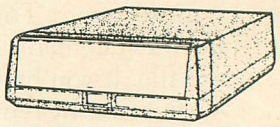
Add $2.50 shipping per software order in continental U.S. Add $5.00 shipping per software order for AK, HI, FPO-APO. Add $10.00 or 15% (whichever is greater) per software order for non-U.S. Call for cost of hardware shipping. Calif. residents add 6½% sales tax. Cashiers checks or money orders filled within 24 hours for items in stock. Personal checks require 4 weeks to clear. MasterCard and Visa OK for software only within continental U.S., add 3% surcharge. Include card no., expiration date and signature. Due to our low prices, all sales are final. All defective returns must have a return authorization number. Please call to obtain one before returning goods for replacement or repair. Prices & availability subject to change.

ASTRA

DISK DRIVE SYSTEM

MORE FOR YOUR MONEY
DOUBLE OR SINGLE DENSITY
TWO DRIVES

2001 DISK DRIVE ... $549



INDUS GT.....$315

ATARI SOFTWARE

MISCELLANEOUS

Financial Wizard (D) .. $41
Castle Wolfenstein (D) $20
Master Type (D) $27
Astro Chase (D) $22
Miner 2049er (R) $34
Megalegs (C) $23
Cap'n Cosmo (D) $19
Spy's Demise (D) $14
Galac. Chase (D)$20 (C)$18
Kid Grid (C/D) $20
Battle Trek (D) $20
Air Strike (C/D) $27
T&F-PMP Property Management (D) .. $149

GEBELLI

Firebird (R) $27
Embargo (R) $30
Candy Factory (D) $20
Match Racers (C/D) .. $20
Dr. Goodcode (D) $20

HAYDEN

Go ..... (D) $23 (C) $20
Sargon II .. (D) $23 (C) $20

HES

Coco (C/D) $34
Coco II (C/D) $27

INHOME

Baseball .. (R) $23 (D) $23
Crypts of Terror .....
..... (D) $23 (C) $20

INNOVATIVE DESIGN

Speedway Blast .....
..... (D) $20 (R) $27
Pool 400 (R) $27

JV SOFTWARE

Jrny to Plnts (C/D) $20
Action Quest (C/D) ... $20
Ghost Encount. (C/D) . $20

INFOCOM

Zork I, II or III (D) $27
Deadline (D) $34
Starcross (D) $27
Suspended (D) $34
Witness (D) $34
Planetfall (D) $34
Enchanter (D) $34
Infidel (D) $34

KRELL SAT ..... Call

INTELL. STATEMENTS

Prof. Blackjack (D) ... $46

LJK

Letter Perfect (D) $74
Data Perfect (D) $74
Spell Perfect (D) $56
Letter Perfect (R) $74

MICROPROSE

Solo Flight (D) $26
Hellcat Ace (C/D) $23

MONARCH

ABC Compiler (D) $55

OPTIMIZED SYSTEMS

Action (R) $65
Basic XL (R) $65
Mac 65 (D) $58
C-65 (D) $58
Bug 65 (D) $23

ODESTA

Chess (D) $45
Checkers (D) $34
Odin (D) $34

PARKER BROS

Astrochase (R) $33
Death Star (R) $33
Q-Bert (R) $33
Popeye (R) $33

PHOENIX

Birth of Phoenix (D) .. $16
Adv. In Time (D) $20

QUALITY

Name That Song .....
..... (D) $13 (C) $11
Return of Hercules (D) $22
Ali Baba (D) $22
Jeeper Creepers (D) .. $20

ROKLAN

Gorf ..... (D) $27 (R) $30
Dlx Invaders (D)$23 (R)$27

RESTON

Moviemaker (D) $45

SCARBOROUGH SYS.

Mastertype (D/R) $27
Songwriter (D) $27

SCHOOL WIZWARE . Call

SENTIENT

Gold Rush (D) $23
Cyborg (D) $23

SIERRA ON-LINE

Mission Asteroids (D) $17
Ulys.& Gldn Fleece (D) $27
Crossfire .. (D) $20 (R) $23
Mouseattack (D) $23
Jawbreaker (D) $20 (R) $23
Threshold (D) $27
Homeword (D) $46
Ultima II (D) $39
Dark Crystal (D) $26
Wiz. & Princess (D) ... $22
Lunar Leeper (D) $20
Wiz & Princess (D) ... $22
Frogger (C/D) $23

SIRIUS

Bandits (D) $23
Space Eggs (D) $20
Sneakers (D) $20
Way Out (D) $27
Type Attack (D) $27
Repton (D) $27
Critical Mass (D) $27
Fast Eddy (R) $23

SPINNAKER

Snooper Troop 1,2 (D) . $30
Most Amazing (D) $27
Kids on Keys (D) $20
Trains (D) $27
Delta Drawing (R) $27
Aerobics (D) $34
Hey Diddle Diddle (D) . $20
Srch Amzng Thngs (D) $27
Story Machine (D) $23
Face Maker (D) $23

STRATEGIC SIM.

Broadsides (D) $27
Carrier Force (D) $39
Combat Leader (D) ... $27
Rails West (D) $27
Epidemic (D) $23
Eagles (D) $27
Cosmic Bal I or II (D) .. $27

SUBLOGIC

Flight Simulator II (D) . $36
Pinball (C/D) $20

SWIFTY

J. White Music Lessons ..... (C/D) $20

SYNAPSE

File Manager (R) $54
Fort Apocalypse (C/D) $23
Dimension X (C/D) ... $23
Blue Max (C/D) $23
Encounter (D/R) $23
Zepplin (C/D) $23
Pharoah's Curse (C/D) $23
Protector II (D) $23 (R) $29
Shamus ... (D) $23 (R) $29
Fort Apocalypse (C/D) $23
Shamus II (C/D) $23
Necromancer (C/D) ... $23
Pharoh's Curse (C/D) . $23
Drelbs (C/D) $23
Shadow World (C/D) .. $23
Survivor (C/D) $23

THORN EMI

Soccer (R) $34
Jumbo Jet (R) $34
Submarine Comm. (R) $34

TRONIX

S.A.M. (D) $39
P.M. Animator (D) $29
Juice (C/D) $20
Chatterbee (D) $27

USA

3-D Sprgrphcs (C/D) .. $27
Survival Adv. (C/D) ... $17

BOFFO
BASIC listing.

```
10 TRAP 40000:? "{CLR}ASSEMBLY-TO-BASIC DA
TA CONVERTOR":? "{DOWN}By Tom Hudson, ANALO
G Computing"
20 DIM FILE$(15),FI$(17),D$(1),HX$(16)
,DH$(1),CK$(1),OD$(1),ODMSG$(6):HX$="0
123456789ABCDEF"
30 POSITION 2,5:? "{CTRL-T}  OBJ/DATA file";
:INPUT OD$:IF OD$="O" THEN ODMSG$="OBJ
ECT":GOTO 60
40 IF OD$<>"D" THEN 30
50 ODMSG$="  DATA"
60 POSITION 2,7:? "{CTRL-T}";ODMSG$;" filena
me";:INPUT FILE$:TRAP 120:IF LEN(FILE$
)=1 THEN 90
70 IF FILE$(2,2)=":" THEN FI$=FILE$:GO
TO 100
80 IF FILE$(3,3)=":" THEN FI$=FILE$:GO
TO 100
90 FI$="D:":FI$(3)=FILE$
100 OPEN #1,4,0,FI$:IF OD$="O" THEN GE
T #1,BYTE:GET #1,BYTE2:IF BYTE<>255 OR
 BYTE2<>255 THEN 130
110 GOTO 150
120 ? FILE$;" INVALID FILE,";:GOTO 140
130 ? FILE$;" NOT OBJ FILE,";
140 ? " PRESS RETURN";:INPUT D$:CLOSE
#1:GOTO 60
150 POSITION 2,9:? "{CTRL-T} BASIC filename"
;:INPUT FILE$
160 TRAP 200:IF FILE$(2,2)=":" THEN FI
$=FILE$:GOTO 190
170 IF FILE$(3,3)=":" THEN FI$=FILE$:G
OTO 190
180 FI$="D:":FI$(3)=FILE$
190 OPEN #2,8,0,FI$:GOTO 210
200 ? FILE$;" INVALID FILE, PRESS RETU
RN";:INPUT D$:CLOSE #2:GOTO 150
210 POSITION 2,11:? "{CTRL-T}Starting lineno
";:TRAP 210:INPUT LINE:TOTAL=0
220 POSITION 2,13:? "{CTRL-T} Line increment
";:TRAP 220:INPUT INC
230 POSITION 2,15:? "{CTRL-T}    Decimal/Hex
";:INPUT DH$:IF DH$<>"D" AND DH$<>"H"
THEN 230
240 POSITION 2,17:? "{CTRL-T} Bytes per line
";:TRAP 240:INPUT BLIN
250 POSITION 2,19:? "{CTRL-T} Checksum (Y/N)
";:INPUT CK$:IF CK$<>"Y" AND CK$<>"N"
THEN 250
260 COUNT=0:TRAP 420
270 IF OD$="D" THEN LA=0:HA=65535:TOTL
EN=65536:TLC=-1:GOTO 310
280 GET #1,LO1:GET #1,HI1:GET #1,LO2:G
ET #1,HI2:LA=LO1+HI1*256:HA=LO2+HI2*25
6:TOTLEN=HA-LA+1:TLC=-1
290 IF HA=737 AND LA=736 THEN 420
300 IF COUNT>0 AND LA<>LL+1 THEN ? "{CTRL-R}
ERROR!!! MEMORY NOT CONTIGUOUS!":END
310 LL=HA:IF TOTAL>0 THEN 330
320 X=-999
330 TLC=TLC+1:IF TLC=TOTLEN THEN 270
340 GET #1,BYTE:TOTAL=TOTAL+1:COUNT=CO
UNT+1:GNDTOT=GNDTOT+BYTE:IF CK$="Y" AN
D GNDTOT>999 THEN GNDTOT=GNDTOT-1000
350 IF X=-999 THEN ? #2;LINE;" DATA ";
:LINE=LINE+INC:X=0
360 IF DH$="D" AND X>0 AND X<BLIN THEN
 ? #2;",";
370 X=X+1:IF DH$="H" THEN BH=INT(BYTE/
16):BL=BYTE-(BH*16):? #2;HX$(BH+1,BH+1
);HX$(BL+1,BL+1);:GOTO 390
380 ? #2;BYTE;
390 IF X<BLIN THEN 330
400 IF CK$="N" THEN ? #2:GOTO 320
410 ? #2;",";GNDTOT:GOTO 320
420 IF PEEK(195)<>136 AND PEEK(195)<>0
 THEN ? "{CTRL-R}ABNORMAL TERMINATION (ERROR
 ";PEEK(195);")":END
430 IF CK$="N" OR X=-999 THEN 490
440 IF DH$="D" THEN 470
450 ? #2;"00";:TOTAL=TOTAL+1:X=X+1:IF
X<BLIN THEN 450
460 GOTO 480
470 ? #2;",0";:TOTAL=TOTAL+1:X=X+1:IF
X<BLIN THEN 470
480 ? #2;",";GNDTOT
490 ? #2:? #2;LINE;" REM * ";TOTAL;" B
YTES":CLOSE #1:CLOSE #2:END
```

•

CHECKSUM DATA.
(see page 90)

```
10 DATA 16,177,337,993,432,855,216,221
,32,650,701,698,588,720,389,7025
160 DATA 840,704,251,659,579,218,841,7
09,139,513,652,643,195,147,504,7594
310 DATA 764,586,912,363,980,825,589,8
35,89,10,703,491,184,726,662,8719
460 DATA 737,660,627,18,2042
```

•

The Latest Innovations From CDY For Your Atari 400/800/800XL

OMNIMON! Resident Monitor

OMNIMON! is a PC board which plugs into your 400/800 (soon to be available for the XLs also) and gives you complete control of your computer. Even though it is always available (by pressing SELECT and SYSTEM RESET) it takes up no user memory because it resides in the unused 4K block at $C000. Use it to interrupt, examine, and manipulate any program in memory whether it be disk, cassette, or cartridge based. It is especially good for program development or customization of existing programs. The flexible disk I/O allows you to write to or read from disk in either single or double density. You can edit raw sector data or even load a file without DOS. Many debugging tools are at your disposal: Display/Alter memory or 6502 registers, Disassemble memory, Search memory, Hex/Char modes, Single Step execution, JSR or GOTO address, Push/Pull stack, Printer dump, etc. After interrupting a program with OMNIMON!, many times it is possible to return to the program as if you had never left it (e.g., BASIC, DOS, etc.). Instructions are provided for the addition of a simple toggle switch to make OMNIMON! invisible, thus making it compatible with all software. An external cable is now provided to eliminate the need to solder directly on the board.

New 8K OMNIMON! Upgrade

This enhancement, which is available to all OMNIMON! users, includes a substantial number of features not available in the standard version. The 8K OMNI resides in an 8K ROM which has been modified by the addition of a switch for selecting either of two 4K banks. The additional features include Hex Conversion and Hex Arithmetic, Block Move, a Relocater, and a Line Assembler. A Binary Load command allows you to load any binary load file without DOS and doubles as a disk directory command which prints out the start sector of each file. Lockup recovery allows you to recover from system lockup, meaning that when your computer freezes, you can usually salvage the program or text file in memory by popping into 8K OMNI and dumping memory to disk. Advanced users will like the user extendibility feature which allows them to make use of the interface routines of 8K OMNI in their own software. One of the most exciting features of the 8K OMNI is the resident Ramdisk handlers. They allow AXLON Ramdisk owners to use this powerful device with any DOS which uses standard SIO calls and even with boot programs like word processors and games which access the disk a lot. Several additional features make this version very valuable for advanced programmers, but if you have a Ramdisk, 8K OMNI is a MUST!

New OMNIVIEW 80 Column Upgrade

Did you know that for most applications you do not need an expensive, slot consuming 80 column board to enjoy the power of 80 columns? Would you 400 owners like the convenience of 80 columns? OMNIVIEW takes advantage of the high resolution graphics mode built into the ATARI to generate an 80 column screen editor essentially identical to the ATARI screen editor (E:, S:). Thus, you can use OMNIVIEW in any environment where you would normally use the 40 column "E:" (e.g., BASIC, Assembler/Editor, etc.). The 80 column "E:" of OMNIVIEW has been optimized for speed so that it is not significantly slower than 40 column "E:". In addition, the character font was specially designed to be legible on an ordinary TV set! A monitor is recommended, but not really necessary for casual 80 column operation. The Bit-3 version of LJK's 80 column Letter Perfect has been modified to support OMNIVIEW and other programs are sure to follow. Lastly, the Ramdisk handlers described under 8K OMNI are also incorporated in OMNIVIEW.

New RAMROD-XL

800XL owners are now able to equip their computers with OMNIMON and OMNIVIEW. In addition, the Newell enhanced operating system and Fastchip floating point package will be included at no extra charge. This will essentially turn your 800XL back into a 400/800 compatible machine and allow it to run most of the software which the XL-OS will not. A switch will allow you to select the XL-OS when needed.

	OMNIMON Piggy-back 400/800	Ramrod OS Board 800	Upgrades for OMNIMON or Ramrod 8K OMNI	8K VIEW	4K VIEW	Ramrod-XL Piggy-back 800XL	Addon for Ramrod-XL VIEWXL
Enhanced OS w/Fast Cursor		+				+	+
Includes FASTCHIP FP						+	+
80 Columns Emulation				+	+		+
AXLON Ramdisk Handlers			+	+	+		
OMNIMON Features:							
A:Alter Memory	+	+	+	+		+	
B:Boot (Ram) disk			+	+	+		
C:CPU Registers	+	+	+	+		+	
D:Display Memory	+	+	+	+		+	
E:Single Step Execution	+	+	+	+		+	
F:Fill Program Buffer			+			+	
G:Binary Load / Directory			+			+	
H:Hex Conversion			+	+		+	
H:Hex Arithmetic			+			+	
I:Install Ramdisk Handlers			+	+	+		
J:Jump Subroutine (JSR)	+	+	+	+		+	
L:Drive Selection/Control	+	+	+	+		+	
M:Move Block of Memory			+			+	
N:Relocate 6502 Code			+			+	
O:Operate from Program Buffer			+			+	
P:Printer Control	+	+	+	+		+	
R:Read Sector(s) from Disk	+	+	+	+		+	
S:Search Memory for Sequence	+	+	+	+		+	
T:Toggle Hex/Char Display Mode	+	+	+	+		+	
U:User's Custom Command			+				
V:Verify 2 Blocks of Memory			+	+		+	
W:Write Sector(s) to Disk	+	+	+	+		+	
X:Disassemble Memory	+*	+	+	+		+	
Y:Line Assembler			+			+	
Z:Exit Monitor			+				
Lockup Recovery			+	+		+	
Redirection of Printer I/O			+	+		+	
Talk to Happy Ram Buffer			+	+		+	

Pricing

Hardware:	Standard OMNIMON! Piggyback Board	$99.95
	RAMROD-XL with OMNIMON-XL (800XL)	$119.95
	OMNIVIEW-XL Add-on (RAMROD-XL)	$59.95
Enhancements:	(subtract $5.00 if ordered with board)	
	8K OMNIMON Enhancement	$45.00
	8K OMNIVIEW Enhancement	$45.00
	4K OMNIVIEW Enhancement	$30.00
	Letter Perfect or Data Perfect	$75.00
	AXLON 128K Ramdisk	$260.00
	BASIC-XL, MAC65, ACTION!	$75.00

Newell RAMROD OS Board

This is a new operating system board which replaces the existing OS board. It allows you to use EPROMs in place of the ATARI OS ROMs and comes with an enhanced OS which includes additional graphics modes and a fast cursor. It also has a socket which will accept any version of OMNIMON and thus is an alternative to the OMNIMON! piggyback board. For the 800 only.

RAMROD OS Board with Standard OMNIMON	$149.95
RAMROD OS Board with 8K OMNIMON or 8K OMNIVIEW	$189.95
Same as above with Fastchip Floating Point Package	$209.95
RAMROD OS Board with 8K OMNIMON and 4K OMNIVIEW	$209.95
Fastchip Floating Point Package by itself	$29.95

How To Order

Add $1.50 shipping ($3.00 for 2-day delivery). Add $1.65 for C.O.D. We accept checks, money orders, or credit cards (Visa or MC). COD must be payed with cash or MO. Our toll free order desk is (800) 227-3800 ext. 561. Call this number only if you know **exactly** what you want and be explicit! They can answer no questions. For faster service or if you have questions please call or write:

CDY Consulting
421 Hanbee
Richardson, TX 75080
(214) 235-2146

DEALER INQUIRIES SOLICITED

Bopotron Construction Set

16K Cassette or Disk

by Kyle Peacock

This article and program listing are to be used in conjunction with **Bopotron** on page 33. The **Construction Set** will allow you to enter, save and edit the levels you've created and incorporate them into the game. The manner in which you create your own levels may seem overwhelming at first, but — with a little patience — you and your friends will be bopping your brains out.

First, let's examine the graphics mode 0 graph paper found on page 58. You are free to photocopy this page for the creation of **Bopotron** levels. Any other photocopying of the magazine infringes on copyright laws, which could put you in a lot of hot water. The remainder of this article will be based on the numbers found at the top and leftmost side of the graph. These will be called the X and Y coordinates, respectively. Notice how rows 0 through 3 and column 39 have been shaded off. These areas are unavailable for use by the **Bopotron** designer. The program doesn't check to see if you're using these areas or not, so it's totally the designer's responsibility.

Initially, you must specify where the **Construction Set** is to put the level specific output it will generate. There are four options available. They are: screen (E:), printer (P:), disk drive (D:FILENAME.EXT) or cassette (C:). The characters in parentheses represent the required input for the particular device.

Next, specify the level number you are designing. There are six steps to designing your own **Bopotron** levels. They are: power-up, girder placement, ladder placement, power pack placement, power unit placement and platform programming. Each of these steps will be explained in detail along with specific examples from the graph of level 1 on page 66.

Power-up.

This step involves setting the initial conditions for a particular level. You must first specify the **Bopotron**'s starting X and Y coordinates. Given this X and Y, the **Bopotron**'s left footpad will appear on that coordinate. An initial and maximum power setting must then be input. When the level begins, the **Bopotron** will have an initial internal power level. Charging at a power pack will allow the **Bopotron** to accumulate energy up to the maximum setting. Next, the exit girder position must be specified. Bopping onto the exit girder, after energizing all of the power units, will allow you to proceed to the next level.

Girder placement.

This step involves the placement of all girders for a particular level. First, input the number of girder segments present for the level being designed. A girder segment is of variable length but must be contiguous. Level 1 has nine girder segments.

The format for girder placement is as follows: starting X coordinate, ending X coordinate and a Y coordinate. The starting X must be smaller than the ending X. For example, level 1 has a girder segment from X position 0 to 8 at a Y position of 4. Using the **Construction Set**, you would type:

```
GIRDER 1 POSITION :0,8,4
```

The remainder of the girder segments would follow the same format.

Ladder placement.

This segment involves the placement of ladders onto a particular level. First, input the number of ladder segments present on the level being designed. A ladder segment is of variable length but must be contiguous. Ladders consist of three horizontal characters side by side. For this reason, the maximum X coordinate for any ladder would be 36 (since column 39 is unavailable for use). Level 1 has four ladder segments.

The format for ladder placement is as follows: starting Y coordinate, ending Y coordinate and a leftmost X coordinate. The starting Y must be smaller than the ending Y. For example, level 1 has a ladder segment at Y position 4 to 9, at an X position of 27. Using the **Construction Set**, you would type:

```
LADDER 1 POSITION :4,9,27
```

The remainder of the ladder segments would follow the same format.

Power pack placement.

This step involves the placement of all power packs for a particular level. First, input the number of power packs present for the level being designed. A power pack consists of four characters arranged as a 2×2 square. Level 1 has one power pack.

The format for power pack placement is as follows: lower left X coordinate, lower left Y coordinate. Given this X and Y designation, the lower left character of the 2×2 square will appear at this position. For example, level 1 has a power pack at an X position of 0 and a Y position of 23. Using the **Construction Set**, you would type:

```
POWER PACK 1 POSITION :0,23
```

The remainder of the power packs would follow the same format.

Power unit placement.

This step involves the placement of all power units for a particular level. First, input the number of power units present for the level being designed. A power unit consists of four characters arranged as a 2×2 square. Level 1 has 2 power units.

The format for power unit placement is as follows: lower left X coordinate, lower left Y coordinate. Given this X and Y designation, the lower left character of the 2×2 square will appear at this position. For example, level 1 has a power unit at an X position of 3 and a Y position of 9. Using the **Construction Set**, you would type:

```
POWER UNIT 1 POSITION :3,9
```

The remainder of the power units would follow the same format.

Platform programming.

This step involves the programming of the multi-vector maintenance platforms. Any level can have up to two platforms. Each platform can have up to five vectors, for a total of ten pre-programmed vectors. Those of you unfamiliar with vectors are encouraged to read Tom Hudson's **BASIC Training** article in issue 18 (page 69). The platforms follow the simple X-Y matching algorithm found there.

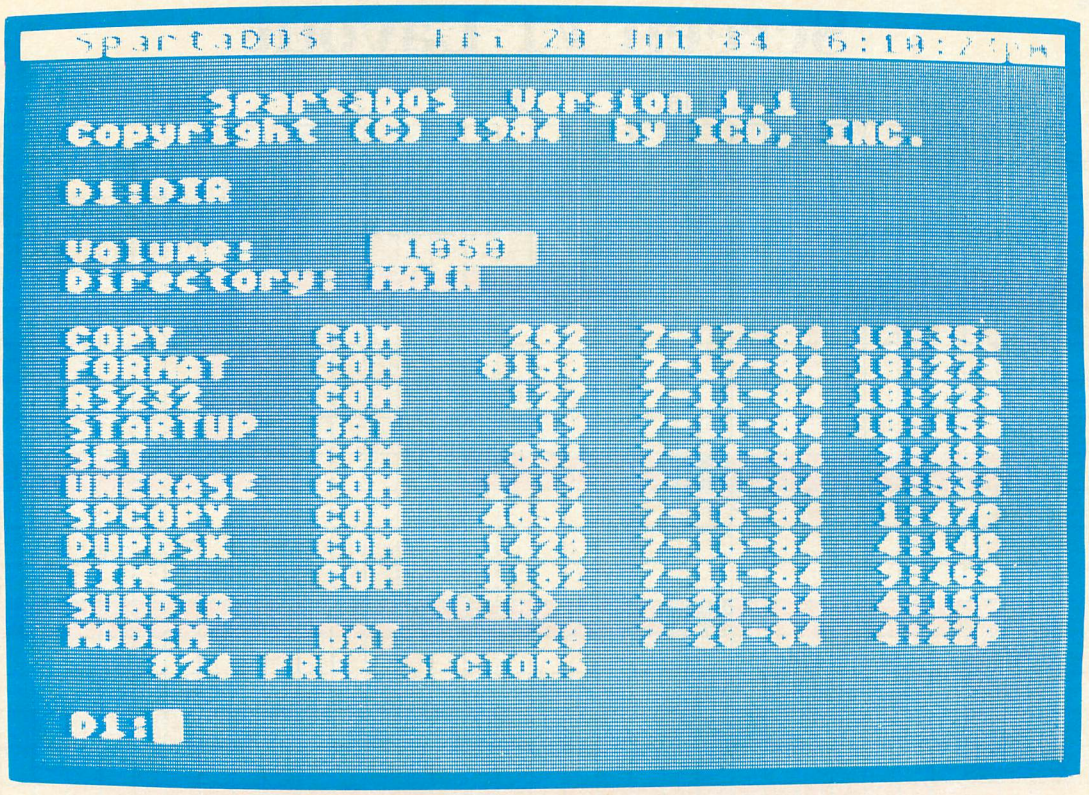
Platform programming is accomplished by first specifying the number of platforms present for the level being designed. Next, the number of vectors for each platform is specified. Finally, the coordinates of each vector are input. The format for platform vectoring is as follows: origin X, origin Y, destination X, destination Y, speed.

Each platform takes up two characters horizontally. The vector coordinates designate where the left side of the platform will be positioned. As mentioned earlier, simple X-Y matching is incorporated, so the origin and destination can be located anywhere on the graph.

The speed parameter mentioned above designates how often the platform moves. This value is in jiffies (1/60 second). A value of 1 will cause the platform to

(continued on page 67)

. . . Until now, there was no reason to buy another DOS for your Atari® Computer



Introducing SpartaDOS™ $39.95

a resident DOS that supports all drives, all densities, 5¼ and 8 inch, single or double sided, time/date stamping of all files, unlimited multiple directories, user created batch files, automatic (intelligent) format selection, works with **all** Atari compatible drives including the ATR8000® .

. . . Finally, true double density for the Atari 1050 and it's affordable!

Introducing the US Doubler™ . . . $69.95

turns your Atari 1050 into the drive Atari **should** have made. Fully compatible with all existing software. When used with included SpartaDOS, UltraSpeed™ I/O reads and writes an amazing **3 times faster** than your present 1050 - plus in single density, density and one-half (Atari double), and true double density! Installation required.

other fine products from ICD include: The Chip with Archiver I™ . . $99.95, Happy Archiver™ . . $39.95, and coming soon Archiver II™.

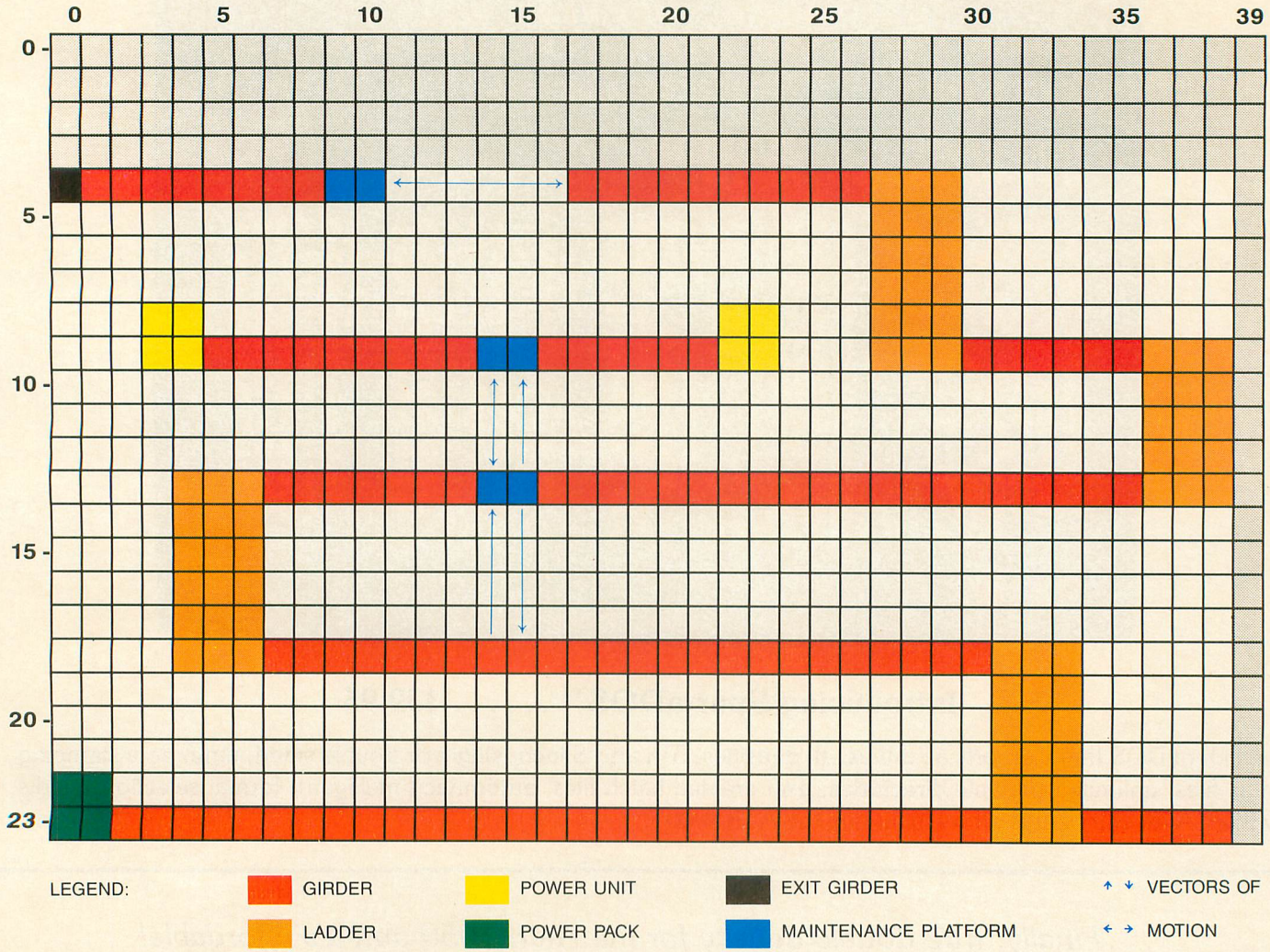
ICD, Inc.
828 Green Meadow Avenue
Rockford, IL 61107
(815) 229-2999 (1-9 pm CST)

Distributors now in:
United Kingdom and Canada

order direct or call for the dealer or distributor nearest you.

Bopotron Construction Set

Screen #1 Layout



BOPOTRON CONSTRUCTION SET

POWER-UP

LEVEL #1
BOPOTRON LOCATION (X,Y) :0,4
INITIAL POWER :500
MAXIMUM POWER :500
EXIT GIRDER POSITION (X,Y) :0,4

GIRDER PLACEMENT

GIRDER 1 POSITION :0,8,4
GIRDER 2 POSITION :17,26,4
GIRDER 3 POSITION :3,13,9
GIRDER 4 POSITION :16,23,9
GIRDER 5 POSITION :30,35,9
GIRDER 6 POSITION :7,13,13
GIRDER 7 POSITION :16,35,13
GIRDER 8 POSITION :7,30,18
GIRDER 9 POSITION :0,38,23

LADDER PLACEMENT

LADDER 1 POSITION :4,9,27
LADDER 2 POSITION :9,13,36
LADDER 3 POSITION :13,18,4
LADDER 4 POSITION :18,23,31

POWER PACK PLACEMENT

POWER PACK 1 POSITION :0,23

POWER UNIT PLACEMENT

POWER UNIT 1 POSITION :3,9
POWER UNIT 2 POSITION :23,9

PLATFORM PROGRAMMING

PLATFORM :1

NUMBER OF VECTORS: 3

VECTOR 1 TRAJECTORY :14,17,14,9,2
VECTOR 2 TRAJECTORY :14,9,14,13,1
VECTOR 3 TRAJECTORY :14,13,14,17,1

PLATFORM :2

NUMBER OF VECTORS: 2

VECTOR 1 TRAJECTORY :9,4,15,4,1
VECTOR 2 TRAJECTORY :15,4,9,4,3

advance once every 60th second. A value of sixty will cause it to move once a second. A value of zero will probably foul things up.

Once you've input all the level-specific information, the program will generate the DATA statements for that level on the output device specified. To incorporate your new level into the game, simply LOAD **Bopotron** into memory as usual. Next, if you're using a cassette or disk drive, load your new level into memory, using the ENTER command. Finally, increment the MAXLEVEL variable on Line 160 and SAVE the game again. Those with 16K should only specify the level they're designing as being from 1 to 5. Memory limitations allow for a maximum of five levels on 16K machines.

Editing levels.

Editing a level is accomplished by skipping over certain sections of the **Construction Set** and only entering values for updated data. Assuming the use of disk, this newly-edited data *must* have a different filename than the original level. For example, let's say that you designed a level where everything was working properly except for a misplaced ladder. When asked for the **Bopotron** starting position during the power-up section, enter 0,0. This will cause you to advance to the girder placement section. Enter 0 for the number of girders. This will advance you to the ladder placement section. Now, re-enter *all* of the information regarding ladders for that level. Next, enter 0 for the number of power packs, power units and platforms in the corresponding sections. The **Construction Set** will then generate output for the ladder placement section only. Once this is written to the specified device, assuming cassette or disk, type *NEW*. Now load the old level data into memory, using the ENTER command. Next, ENTER the newly edited data into memory. Finally, save the entire block of data to the output device, using the LIST command. Now your level is complete and ready to be loaded.

Become a Bopotron Brigadier.

Those of you with a little imagination and ambition are encouraged to put on your brain bubbles. Future issues of **ANALOG Computing** will publish additional levels for **Bopotron** from our readers. If you think you can bop with the best, send in the output generated by the **Construction Set** on either disk, cassette or printout. No hand-written output, please. Cassettes and disks will be returned if accompanied by a self-addressed, stamped envelope. All levels must be completable, and the designer's name should be specified. Send your output to:

Bopotron's Boogie Brigade (BBB)
ANALOG Computing
P.O. Box 23
Worcester, Ma 01603

Who knows? Maybe your name will appear in the pages of **ANALOG Computing**. □

```
100 REM ******************************
110 REM *                            *
120 REM * BOPOTRON CONSTRUCTION SET *
130 REM *       BY KYLE PEACOCK      *
140 REM *     ANALOG  COMPUTING      *
150 REM *                            *
160 REM ******************************
170 REM
180 TRAP 890
190 DIM FILE$(15),D$(6),X$(10),G(150),
L(150),B(150),P(150),V(150),X(150)
200 REM *** OUTPUT DEVICE
210 D$=" DATA ":X$="C:D:P:E:S:"
220 GOSUB 870:? "DEVICE FILENAME IS ";
:INPUT #16;FILE$
230 FOR X=1 TO 9:IF X$(X,X+1)=FILE$(1,
2) THEN 260
240 NEXT X:GOTO 890
250 REM *** LEVEL
260 GOSUB 870:POSITION 16,1:? "POWER-U
P":? :? "LEVEL # ";:INPUT #16;LVL
270 REM *** BOPOTRON LOCATION
280 ? :? "BOPOTRON LOCATION (X,Y) :";:
INPUT #16;BOPX,BOPY:IF BOPX=K0 AND BOP
Y=K0 THEN 310
290 ? :? "INITIAL POWER :";:INPUT #16;
PWR:? :? "MAXIMUM POWER :";:INPUT #16;
MAXPWR
300 ? :? "EXIT GIRDER POSITION (X,Y) :
";:INPUT #16;XITX,XITY
310 REM *** GIRDER DRAW
320 GOSUB 870:POSITION 12,1:? "GIRDER
PLACEMENT":?
330 ? "NUMBER OF GIRDERS :";:INPUT #16
;GIRAMT:IF GIRAMT>50 THEN 330
340 IF GIRAMT<=0 THEN 380
350 ? :? "FORMAT: STARTING X, ENDING X
, Y":?
360 ? :FOR X=1 TO GIRAMT:XX=(X-1)*3:?
"GIRDER ";X;" POSITION :";:INPUT #16;A
,B,C:G(XX+1)=A:G(XX+2)=B:G(XX+3)=C
370 NEXT X
380 REM *** LADDER DRAW
390 GOSUB 870:POSITION 12,1:? "LADDER
PLACEMENT":?
400 ? "NUMBER OF LADDERS :";:INPUT #16
;LADAMT:IF LADAMT>50 THEN 360
410 IF LADAMT<=0 THEN 450
420 ? :? "FORMAT: STARTING Y, ENDING Y
, X":?
430 ? :FOR X=1 TO LADAMT:XX=(X-1)*3:?
"LADDER ";X;" POSITION :";:INPUT #16;A
,B,C:L(XX+1)=A:L(XX+2)=B:L(XX+3)=C
440 NEXT X
450 REM *** POWER PACK DRAW
460 GOSUB 870:POSITION 10,1:? "POWER P
ACK PLACEMENT":?
470 ? "NUMBER OF POWER PACKS :";:INPUT
 #16;BATAMT:IF BATAMT>50 THEN 32767
480 IF BATAMT<=0 THEN 510
490 ? :? "FORMAT: LOWER LEFT X, LOWER
LEFT Y":?
500 ? :FOR X=1 TO BATAMT:XX=X*2-1:? "P
OWER PACK ";X;" POSITION :";:INPUT #16
;A,B:B(XX)=A:B(XX+1)=B:NEXT X
510 REM *** POWER UNIT DRAW
520 GOSUB 870:POSITION 10,1:? "POWER U
NIT PLACEMENT":?
530 ? "NUMBER OF POWER UNITS :";:INPUT
 #16;PWRAMT:IF PWRAMT>50 THEN 530
540 IF PWRAMT<=0 THEN 570
550 ? :? "FORMAT: LOWER LEFT X, LOWER
LEFT Y":?
560 ? :FOR X=1 TO PWRAMT:XX=X*2-1:? "P
OWER UNIT ";X;" POSITION :";:INPUT #16
;A,B:P(XX)=A:P(XX+1)=B:NEXT X
570 REM *** PLATFORM PROGRAMMING
580 GOSUB 870:POSITION 10,1:? "PLATFOR
M PROGRAMMING":?
590 V(1)=0:PLUS=1:? "NUMBER OF PLATFOR
MS :";:INPUT #16;PLRAMT:IF PLRAMT<0 OR
 PLRAMT>2 THEN 590
600 IF PLRAMT=0 THEN 690
610 V(PLUS)=PLRAMT:PLUS=PLUS+1
620 FOR X=1 TO PLRAMT:? :? "PLATFORM :
";X
630 ? :? "NUMBER OF VECTORS :";:INPUT
#16;VEC:IF VEC<1 OR VEC>5 THEN 630
```

```
640 V(PLUS)=VEC:PLUS=PLUS+1
650 ? :? "FORMAT: ORIGIN X, ORIGIN Y,"
:? "        DESTINATION X, DESTINATION
 Y,";? "              SPEED"
660 ? :FOR Y=1 TO VEC:? "VECTOR ";Y;"
TRAJECTORY :";:INPUT #16;A,B,C,D,E
670 V(PLUS)=A:V(PLUS+1)=B:V(PLUS+2)=C:
V(PLUS+3)=D:V(PLUS+4)=E:PLUS=PLUS+5
680 NEXT Y:NEXT X:HOLD=PLUS-1
690 REM *** DISPLAY OUTPUT
700 GOSUB 870:? "GENERATING OUTPUT, PL
EASE WAIT...";:OPEN #1,8,0,FILE$
710 IF BOPX=K0 AND BOPY=K0 THEN 740
720 PRINT #1;2080+(20*LVL);D$;BOPX;","
;BOPY;",";PWR;",";MAXPWR
730 PRINT #1;3080+(20*LVL);D$;XITX;","
;XITY
740 OFF=4080:AMOUNT=GIRAMT:MAX=GIRAMT*
3:FOR X=1 TO 150:X(X)=G(X):NEXT X:GOSU
B 800
750 OFF=5080:AMOUNT=LADAMT:MAX=LADAMT*
3:FOR X=1 TO 150:X(X)=L(X):NEXT X:GOSU
B 800
760 OFF=6080:AMOUNT=BATAMT:MAX=BATAMT*
2:FOR X=1 TO 150:X(X)=B(X):NEXT X:GOSU
B 800
770 OFF=7080:AMOUNT=PWRAMT:MAX=PWRAMT*
2:FOR X=1 TO 150:X(X)=P(X):NEXT X:GOSU
B 800
780 OFF=8080:AMOUNT=V(1):MAX=HOLD:FOR
X=1 TO 150:X(X)=V(X):NEXT X:GOSUB 800
790 END
800 COUNT=0+1*(OFF=8080):IF AMOUNT=0 T
HEN 860
810 PLUS=0:GOSUB 880:PRINT #1;AMOUNT;"
,";
820 COUNT=COUNT+1:PRINT #1;X(COUNT);
830 IF COUNT/25=INT(COUNT/25) THEN PLU
S=PLUS+1:IF COUNT<>GIRAMT*MAX THEN PRI
NT #1;"":GOSUB 880:GOTO 820
840 IF COUNT=MAX THEN PRINT #1;"":GOTO
 860
850 PRINT #1;",";:GOTO 820
860 RETURN
870 ? "K":POSITION 8,0:? "BOPOTRON CON
STRUCTION SET":? :RETURN
880 PRINT #1;OFF+(20*LVL)+PLUS;" DATA
";:RETURN
890 ? "K":POSITION 6,1:? "ABNORMAL PRO
GRAM TERMINATION"
```

●

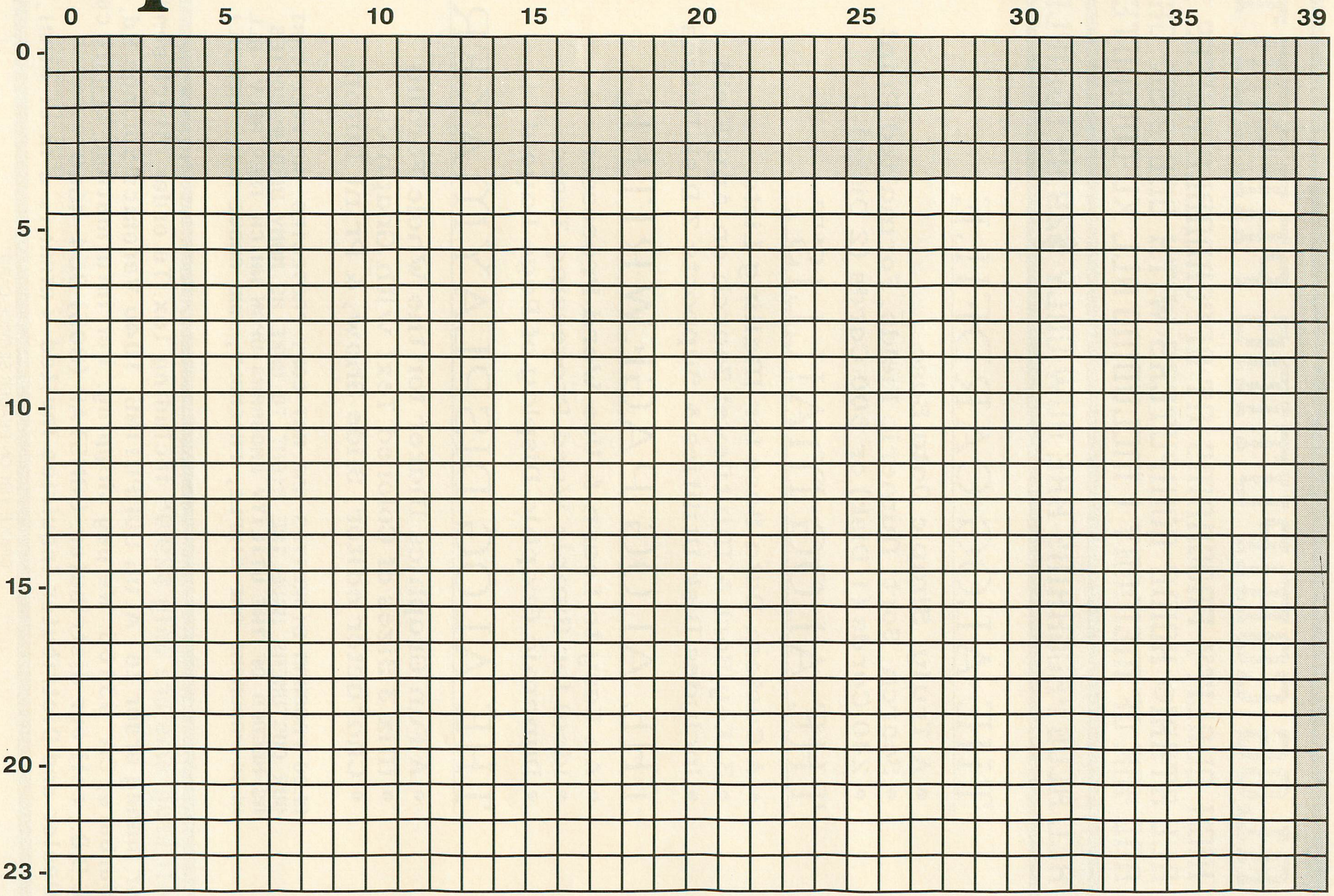
CHECKSUM DATA.

(see page 90)

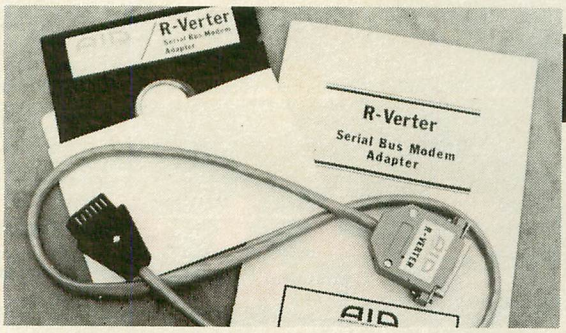
```
100 DATA 778,241,359,359,519,253,796,9
5,755,634,599,974,123,903,891,8279
250 DATA 186,853,49,499,922,966,974,94
1,725,393,30,544,783,967,941,9773
400 DATA 619,365,25,511,776,616,546,49
2,387,257,385,679,603,161,443,6865
550 DATA 247,604,489,662,415,297,442,1
48,209,982,265,117,3,191,563,5634
700 DATA 163,560,614,547,828,768,791,9
35,907,65,837,385,32,635,503,8570
850 DATA 126,613,8,199,757,1703
```

●

Bopotron Construction Set



Make Your Best Connection



R-Verter™
SERIAL BUS MODEM ADAPTOR

WITH • Smart Terminal Program-with X Modem Protocol • "R:" Handler -use with Basic, ACTION!, etc. • **AVAILABLE SOON** - Advanced software disk with 80 Column Terminal Program.

Available in Four Models to match Your Modem - compatible with most RS-232 devices.

*Works with — Atari 400™; 600XL™; 800™; 800XL™ ** $49.95

INTERFAST-I™
BUFFERED PRINTER INTERFACE

Can You Print This?

SCRIPT SCRIPT script script SCRIPT
INVERSE inverse INVERSE inverse

YOU CAN WITH AN INTERFAST-I™ $129.95

* *trademark of Atari, Inc.*
ACTION! is a trademark of Action Computer Services
R-Verter and INTERFAST-I are trademarks of Advanced Interface Devices, Inc.

AID
ADVANCED INTERFACE DEVICES

MC — VISA — C.O.D.
P.O. Box 2188
Melbourne, FL 32902
(305) 242-2772

CIRCLE #132 ON READER SERVICE CARD

INTRODUCING
the NEW ASTRA 2001

Double Density Double Disc Drive

- ☐ More Reliable
- ☐ Quieter Operation
- ☐ Faster Read Write
- ☐ More Capacity (360K Bytes)
- ☐ Easier Data Read

ALL THIS WRAPPED UP IN ONE ATTRACTIVE UNIT

ASTRA SYSTEMS INC.
2500 S. FAIRVIEW, UNIT L SANTA ANA, CA 92704
Call **(213) 804-1475** for your nearest dealer.

CIRCLE #133 ON READER SERVICE CARD

RACE IN SPACE

16K Cassette or 24K Disk

by Charles Bachand

This month's assembly language game, **Race in Space**, is a two-player game with options that allow 128 different variations.

You and a friend (or enemy) must race your spaceships through a densely-packed asteroid field. If your ship makes contact with any other object, it will be damaged and must return to its launch point. **Race in Space** is a semi-violent game, where no one gets killed.

Typing it in.

Before typing anything, look at the listings accompanying this article.

Listing 1 is the BASIC data and data checking routine. This listing is used to create both cassette and disk versions of **Race in Space**. The data statements are listed in hexadecimal (base 16), so the program will fit in 16K cassette systems.

Listing 2 is the assembly language source code for the game of **Race in Space**, created with the OSS MAC/65 assembler. You *don't* have to type this listing to play the game! It is included for those readers interested in assembly language.

Follow the instructions below to make either a cassette or disk version of **Race in Space**.

Cassette instructions.

1. Type Listing 1 into your computer using the BASIC cartridge and verify your typing with **Unicheck** (see page 90).

2. Type *RUN* and press RETURN. The program will begin and ask:

```
MAKE CASSETTE (0) OR DISK (1)?
```

Type *0* and press RETURN. The program will begin checking the DATA statements, printing the line number of each as it goes. It will alert you if it finds any problems. Fix any incorrect lines and re-RUN the program, if necessary, until all errors are eliminated.

3. When all of your DATA lines are correct, the computer will beep twice and prompt you to *READY CASSETTE AND PRESS RETURN.* Now, insert a blank cassette in your recorder, press the RECORD and PLAY buttons simultaneously and hit RETURN. The message *WRITING FILE* will appear, and the program will create a machine language boot tape version of **Race in Space**, printing each DATA line number as it goes. When the *READY* prompt appears, the game is recorded and ready to play. CSAVE the BASIC program onto a separate tape before continuing.

4. To play the game, rewind the tape created by the BASIC program to the beginning. Turn your computer OFF and remove all cartridges. Press the PLAY button on your recorder and turn ON your computer while holding down your START key. If you have a 600 or 800XL computer, you must hold the START and OPTION keys when you turn on the power. The computer will "beep" once. Hit the RETURN key, and **Race in Space** will load and run automatically.

Disk instructions.

1. Type Listing 1 into your computer, using the BASIC cartridge and verify your typing with **Unicheck** (see page 90).

2. Type *RUN* and press RETURN. The program will ask:

```
MAKE CASSETTE (0) OR DISK (1)?
```

Type *1* and press RETURN. The program will begin checking the DATA lines, printing the line number of each statement as it goes. It will alert you if it finds any problems. Fix incorrect lines and re-RUN the program, if necessary, until all errors are eliminated.

3. When all DATA lines are correct, you will be prompted to *INSERT DISK WITH DOS, PRESS RETURN.* Put a disk containing DOS 2.0S into drive #1 and press RETURN. The message *WRITING FILE* will appear, and the program will create an AUTORUN.SYS file on the disk, displaying each DATA line number as it goes. When the READY prompt appears, the game is ready to play. Be sure the BASIC program is SAVEd before continuing.

4. To play the game, insert the disk containing the AUTORUN.SYS file into drive #1. Turn your computer OFF, remove all cartridges and turn the computer back ON. **Race in Space** will load and run automatically.

Playing the game.

Race in Space requires the use of two joysticks plugged into ports 1 and 2. When the game starts, you are presented with a scrolling title screen. To enter the game options screen from either the intro screen or the game play screen, simply press OPTION.

Race in Space.

When in the options menu, use the OPTION key to select which option to change, and then use the SELECT key to change it. Use the START key to begin game play. While in the options menu, you may also press one of the number keys from 1 to 9 to change the time limit for the game from the default of three minutes. All selected options will remain in effect for subsequent game plays.

Game options.

Race in Space has five options that can be changed within the options menu to suit your own personal tastes. These options have been broken down for you here:

1. **Trigger** — NO EFFECT renders the trigger button completely useless.

SHIELDS will put up a protective barrier around your ship that will allow objects to pass through it without harm. Unfortunately, all the power from your engines will be used to run the shields, thus leaving you dead in space.

MISSILES allow both you and your opponent to fight it out in outer space using mega-menace Boomerang missiles, which automatically lock onto your enemy's coordinates when fired. To save yourself, you must move your ship out of the path of the oncoming missile. Only one missile per ship is allowed on the screen at any one time. You may fire a missile when the top of your ship has a "knob" on it (this is your missile). Every time your missile hits your opponent's ship, or is absorbed by a comet (see COMETS), you will receive one point.

WARP DRIVE allows your ship to travel at twice its normal cruising speed, using its wave-motion engine. At this speed, you must be twice as careful.

2. **Density** — This varies the number of asteroids present on the game screen. Choosing from STANDARD, DOUBLE, TRIPLE or SUPER will increase the number of asteroids, thereby increasing their density on the playfield and the game difficulty.

3. **Comets** — Comets can either be ENABLED or DISABLED and roar across the screen at random intervals. You can hear a comet as it approaches, and as it travels across the screen.

4. **Universe** — You will normally fly through a POSITIVE universe, one that is very stable. This is the default setting and the one that you will, most likely, use. Adventurous people can explore the dangers of a NEGATIVE universe, where space itself will flash at random intervals.

Every time your ship reaches the top of the screen, a new one will reappear at the bottom—at your launch site—and you will be awarded one point. The winner is simply the player with the most points at the end of the game. **Race in Space** will end when either player reaches 99 points, or you run out of time. Good luck; you'll need it! □

Listing 1.
BASIC listing.

```
10 REM *** RACE IN SPACE ***
20 TRAP 20:? "MAKE CASSETTE (0), OR DI
SK (1)";:INPUT DSK:IF DSK>1 THEN 20
30 TRAP 40000:DATA 0,1,2,3,4,5,6,7,8,9
,0,0,0,0,0,0,0,10,11,12,13,14,15
40 DIM DAT$(91),HEX(22):FOR X=0 TO 22:
READ N:HEX(X)=N:NEXT X:LINE=990:RESTOR
E 1000:TRAP 120:? "CHECKING DATA"
50 LINE=LINE+10:? "LINE:";LINE:READ DA
T$:IF LEN(DAT$)<>90 THEN 220
60 DATLIN=PEEK(183)+PEEK(184)*256:IF D
ATLIN<>LINE THEN ? "LINE ";LINE;" MISS
ING!":END
```

```
70 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)
80 IF PASS=2 THEN PUT #1,BYTE:NEXT X:R
EAD CHKSUM:GOTO 50
90 TOTAL=TOTAL+BYTE:IF TOTAL>999 THEN
TOTAL=TOTAL-1000
100 NEXT X:READ CHKSUM:IF TOTAL=CHKSUM
 THEN 50
110 GOTO 220
120 IF PEEK(195)<>6 THEN 220
130 IF PASS=0 THEN 170
140 IF  NOT DSK THEN 160
150 PUT #1,224:PUT #1,2:PUT #1,225:PUT
 #1,2:PUT #1,187:PUT #1,47:CLOSE #1:EN
D
160 FOR X=1 TO 89:PUT #1,0:NEXT X:CLOS
E #1:END
170 IF  NOT DSK THEN 200
180 ? "INSERT DISK WITH DOS, PRESS RET
URN";:DIM IN$(1):INPUT IN$:OPEN #1,8,0
,"D:AUTORUN.SYS"
190 PUT #1,255:PUT #1,255:PUT #1,0:PUT
 #1,32:PUT #1,254:PUT #1,47:GOTO 210
200 ? "READY CASSETTE AND PRESS RETURN
";:OPEN #1,8,128,"C:":RESTORE 230:FOR
X=1 TO 40:READ N:PUT #1,N:NEXT X
210 ? :? "WRITING FILE":PASS=2:LINE=99
0:RESTORE 1000:TRAP 120:GOTO 50
220 ? "BAD DATA: LINE ";LINE:END
230 DATA 0,33,216,31,255,31,169,0,141,
47,2,169,60,141,2,211,169,0,141,231,2,
133,14,169,56,141,232,2
240 DATA 133,15,169,187,133,10,169,47,
133,11,24,96
1000 DATA 00000000000000000000000007B32
3232000000060A0A020007E42427E42420000
7E404040407E00007C464646,79
1010 DATA 467C00007E407C40407E00007E40
7C4040400007E40404E427E000042427E4242
4200007E181818187E000040,739
1020 DATA 404040407E00007E424242424200
007E42427E404000007E42427E4C4600007E18
1818181800004242425A5A7E,369
1030 DATA 000042427E181818007E81BDA1A1
BD817E007E424242427E0000080808080800
007E027E40407E00007E027E,235
1040 DATA 02027E000042427E02020200007E
407E02027E00007E407E42427E00007E020202
020200007E427E42427E0000,361
1050 DATA 7E427E02027E000000000000000000
0018245A0000000000000183C3C1800000000
003C7E7E7E3C000000007EFF,6
1060 DATA FFFF00000000003C7E7E00000000
0000183C0000000000000018000000000000000
0000000000000000000000000,936
1070 DATA 0000000003C000000000000FF
FFFFF7E000000007E7E3C000000003C180000
0000000018000000000000000,307
1080 DATA 00181818183C7E5A1818183C7EFF
DB0000000000000000000042424242427E0000
0018180018180000424242242,227
1090 DATA 241800007E5A5A42424200004640
78584C4600FFFFFFFFFFFFFFFF7E4242004242
7E0002020200020202003E02,923
1100 DATA 023C40407C003E02023C02023E00
4242423C020202007C40403C02023E007C4040
3C42427E003E020200020202,713
1110 DATA 007E42423C42427E007E42423C02
023E007067002027272727272727272727272
2707C1001270707046911307,222
1120 DATA 462F1446B0130606462F1446DF13
463E14462F1446EF13467A14462F1446FF1346
B614462F14460F1446D41446,281
1130 DATA 2F14461F1446F214462F14469113
4115127070704F0020000F000F000F000F000F
000F000F000F000F000F000F,888
1140 DATA 000F000F000F000F000F000F000F
000F000F000F000F000F000F000F000F000F00
0F000F000F000F000F000F00,218
1150 DATA 0F000F000F000F000F000F000F00
0F000F000F000F000F000F000F000F000F000F
000F000F000F000F000F000F,563
1160 DATA 000F000F000F000F000F000F000F
000F000F000F000F000F000F000F000F000F00
0F000F000F7030C700083042,329
1170 DATA 1115415A12187E7EFFFF7E7E1800
0000555500000015030C030B1308FF24330308
03150A0C06FF444D4E465846,960
1180 DATA 4C4F58FF618E83848680BA8C8098
8D8384860102FF83504E4A4F4F464C405B51FF
A0C4C9C3CECBC6D8C0C0DBC3,107
1190 DATA C4C9C3CCC5FFC51200141C1B17FF
E6AD8080ADFFF66E40406EFF00000000000000
0000CDCEC6D8D8C0D3CDCFCA,917
1200 DATA D3CCC0C0000000000000000000075
75757575757575757575757575757500130D
0F0A130C18C0C0CFF1968075,44
1210 DATA 00534D4F4A534C71408B86878F80
7575005846 4B46444F71CECAC8C9CFC0757500
584F434E4F7100000D0B0311,159
1220 DATA 0075000F0E0A0808060E31000000
0000757500085868C988A8F91B1808000000075
75009889 8A8D98B100000000,183
1230 DATA 0000007575008493B3868F98B100
0000000007575008B08C8AB2868E9886B10000
0000757500000000000000,320
1240 DATA 0000000000007500000000CCD3C0
C6C7C7C6C4CFC07500000000000D8C9CAC6CB
C5D8C07500000000F3CAD8,953
1250 DATA D8CACBC6D8C07500000D0C3CECD
C0C5CECAF2C6C07500000000000D8CFC3CCC5C3
CEC5C07500000000000000C5,780
1260 DATA D3F0DBCBC6C07500000000000000
CFCECACDCBC6C07500000000000000000D8F0CD
C6CEC07500000000000CED3,481
1270 DATA C4F4C6CFD8C07500000000000D8
C3F0C4C6CED8C07500000000C5CAD8C3DBCB
C6C5C07500000000000C6CC,965
1280 DATA C3DBCBC6C5C07500000000000CDD3
D8CACFCAF2C6C07500000000000CCC6C8C3CFCA
F2C600750000000004130D11,94
1290 DATA 0E0A08090F001200030C030B1308
003303080315 0A0C0600141C1B170000000000
001313000000F9F1F6F6E4F6,930
1300 DATA 0000001313000003000000001F1E
1A181D1B333530FCF3030003010101030003
020302020302000000000E0D,842
1310 DATA 34123D1246124F122B123E144D14
5C146B147A1489149814A714B614C514D414E3
14F2140115030301010004,508
1320 DATA 040202001 83C3C185A7E5A420018
7EDB7E000000000B66DDBB6102001025 8A003
115030C434003C7E7EFFFFFF,15
1330 DATA FFFFFF7E7E3C00C646F666266CA2
51404848E683A5140A29CE8DC2026840AD1FD0
A84582528482C904608D02,265
1340 DATA D2A9A48D03D2A0C0CAD0FD864D88
D0F88C03D260A200A9A48D01D28D03D2A912C5
83D0FC20EB15D00568684CA4,554
1350 DATA 17A9008583E88A8D03D029FC8D02
D04A4A4AA8B971139939218A0A0A0A0A0A49F0
8D00D21869028D02D28AD0C4,204
1360 DATA 8D03D2858560A207A9009D00D0CA
10FA60A9008D01D28D03D28D05D28D07D260A9
3C8D02D3A92B8580A9128581,377
1370 DATA A9008D08D2A2559582CA10FBA929
8D2F02A9038D0FD2A9398D07D4A9028D1DD0A9
DE8D0002A9158D0102A9C08D,343
1380 DATA 0ED4A9108DF402A9C68DC502A994
8DC602A9048DC302A9018D0AD08D0BD08D6F02
205A1685A18586206516AA9D,347
1390 DATA 00209D00219D00229D003BCAD0F1
A207BD0A139D283BBD02139DA83BCA10F1E8BD
1213F010A8E8BD1213C9FFF0,359
1400 DATA F1990021C84C0117A9008D3002A9
128D3102A9208D0312A9008D0212854D8585A9
102483F0FC20EB15D0034CA4,577
1410 DATA 17A9008583E684A584186D021245
868D00D2A584290769DD8D06228D0922690088D
16228D1922A584C910D03BA9,895
1420 DATA 0085848D05D4E685A58645850980
8D01D2AD0212186910 8D0212901EEE0312A922
CD0312D0034C8D16200E16A9,919
1430 DATA 208D00D2A98F8D01D2A90F8586A5
848D05D44C2817A583C583F0FCA9218D2F02A9
158D3002A9128D310220281F,582
1440 DATA 205A16854D8D1DD020651685A085
A1A9948DC602A9048592A692BDD4158DC502BD
D91520F815C69210EEADFC02,993
1450 DATA 293FA209DD5115F005CAD0F8F01D
8A8D4D151869938DAE138A69F68D4015A9FF8D
FC028A0A0A493F20F81520EB,146
```

```
1460 DATA 154A90034CA01D4A9024A93020F8
15A687F689B5893D99159589058E0AAAA000BD
7D159180C8BD7E1591804CE6,158
1470 DATA 174A909FA91020F815A2018688A5
870A0A0A0AAAA00DA9805DE0139DE013E88810
F4A588D0034CE617E687A587,327
1480 DATA 0AAABD71158580BD72158581A687
BD9E15658E858EA200A587C905D0C18687868E
4C4E18A9008593A8A92C8594,767
1490 DATA A906859160A902858FEC0AD22690
0690069006 90C68FD0F1A5909193602094180 6
91A68A20A218C8D0FAE694C6,449
1500 DATA 91D0F4C694A90685918 8B1934A91
9398D0F7C694C691D0F160A00A99000A99800A
8810F7BDBE1595D49D000D08A,799
1510 DATA 48BDC015AAFE0008BD0008C91D90
08A9139D0008FEFF0768AABDC2158D04D2A9AE
8D05D2A90385B5D6AED004A9,140
1520 DATA 0085ADA9FF8D1ED0608A48A5836A
B02EC6B41004A90285B4A6B4BDB6158DAF1529
3C8DA515BDB715293C8DA615,386
1530 DATA A583290EAABD5D158D6E15BD5E15
8D6F1568AA60CE1808105BA9058D1808A5B5F0
05C6B54C7B198D05D2CE0C08,897
1540 DATA CE17081042A9FF8D0C08A9098D17
08A5B5D00AA9A48D05D2A9408D04D2CE0A08CE
16081022A9FF8D0A08A9098D,404
1550 DATA 1608CE0908CE150810 10A9FB8D09
08A9058D1508CE0708CE140808A5B505CB05CC
D01BA5A825A9F015A5B705B8,644
1560 DATA F00FA9288D05D2A5834A29070908
8D04D22860AD09D02901D010AD04D029040D0C
D0F00AA5B0C901F004A90085,48
1570 DATA A8AD08D02902D010AD05D029040D
0DD0F00AA5B1C901F004A90085A960B5B7D0FB
B5B0C902F00395B260D5B2F0,822
1580 DATA EE95B295B718B5D4690285CF95C7
B5D685D0E6D095BFF6BF8AF00BA5D4690285CD
A5D64C601AA5D5690285CDA5,658
```

CONTEST!

Here's a little contest that should keep all the code-crackers out there occupied.

The numbers below, when decoded, are a message in standard Atari ASCII. The numbers are in the proper sequence, and have been encrypted using a simple algorithm.

```
145 211 145 185 255 186 112 88
183 174 224 34 145 126 226 178
51 207 191 129 188 234 4 191
199 175 178 243 197 16 118 43
210 198 166 241 237 194 211 94
213 171 252 246 233 178 12 218
210 203 172 129 133 219 23 186
206 170 203 141 126 246 117 203
190 250 212 206 22 160 197 161
182 183 246 20 53 141
```

Decode the message, if you can, and send your solution to:

Code-Cracker Contest
c/o ANALOG Computing
P.O. Box 23
Worcester, MA 01603

The first five entries we receive with the correct solution will win a *free one-year cassette or disk subscription.* All entries must be postmarked before January 1, 1985.

```
1590 DATA D785CEE6CEA90195C195C938A5CE
E5D095BBB00AA9FF95C155BB95BBF6BB38A5CD
E5CF95C5B00AA9FF95C955C5,153
1600 DATA 95C5F6C5A90095C395B9B5C595BD
A90F95CBB5BBD5BD900695BD4A95C360B5BD4A
95B9604820EB19202D19E683,514
1610 DATA 18A5AC693085AC8D02D2A5ADD003
4C611CA9088D17D08A48A20FA5A8D0038E7802
A5A9D0038E7902AD78022D79,451
1620 DATA 02C90FA981B00CA5B005B1C903A9
88B002A9848D07D2A918B002A9208D06D2A200
86B6E8AD08D085D2AD09D085,918
1630 DATA D3A5B005B1C901D00FAD0AD2291F
09208D06D2A9A68D07D220E11CB5A8F034B5B0
C901F02E20221AA5836A9006,48
1640 DATA B5B0C903D0205E7802B002D6D65E
7802B002F6D6A58BF00E5E7802B002D6D45E78
02B002F6D4B5D4C930B002A9,744
1650 DATA 30C9C89002A9C795D49D00D08492
B5A83024A5836AB01D0A29F69DC002A5B5D013
A583290709C88D05D2AD0AD2,274
1660 DATA 291F09208D04D2F6D6B5D6D00320
E318C9C0902BB5A8D025B5D4DDBE15F00AF6D4
901AD6D4D6D4D014A9FF95A8,789
1670 DATA 8D1ED0BDC4159DC002A5B5D0038D
05D2A9BF95D6E0016A859B9007A5B61DBA1585
B68A4818B5A8F019AD0AD229,511
1680 DATA 0F1DC4159DC002B5B0C90138F007
18BDC4159DC002A009A209A58BF002A212BDA3
1590032D0AD2919BCA8810F2,513
1690 DATA 68AAB5B7F006A001A900919BA492
CA30034C2B1BA5B68D1CD0A5ADF01320651910
1CA9F68D07088D09088D0A08,147
1700 DATA 8D0C08A979854DA90085AD8D05D2
8D07D268AAA5A42583F0026840A5AAF0FA9848
8A48A5A225A7D009A5A51865,640
1710 DATA A6297F85A5A00DA6A5AD0AD239C6
159D000BAD0AD239C6159D800BE88A297FAA88
10E6AD0AD229F009048DC202,692
1720 DATA AD0AD229F009088DC30218A5A265
A385A28D02D08D03D085AAE6ABA5ADF011A5AB
C97DB00E4A4A8D00D2C91890,353
1730 DATA 05499F8D01D268AA68A86840BD84
024AB002A5B935A8D5B0F0038D1ED095B0B5B7
F006B5BFD00395B76018B5BB,103
1740 DATA 75B995B9D5BD900BF5BD95B918B5
C175BF95BF18B5C575C395C3D5BD900BF5BD95
C318B5C975C795C7B5CBF012,215
1750 DATA 4A09888D05D2B5CB490F0A0A0A8D
04D2D6CBB5C79D04D0B5BF4A900948A5B61DBC
1585B6680980859DB5D23D71,480
1760 DATA 15D022B5C7F01E9848A003B19D3D
5B1585D1B96D15E001D0020A0A05D1919D8810
E968A860A90095B79848A003,556
1770 DATA B19D3D5B15919D8810F6B5D23D71
15F00320F61868A8602065168D1ED0A9FF85A8
85A9A9408D0ED4A90C8591A9,51
1780 DATA 008593A9208594A909859EA9218D
6F028D06D2A9018D0AD08D0BD085ADA96385AE
85AFA218BD39159D0008CA10,9
1790 DATA F7A90A859C20281FA9058D0CD0A9
0085A285A085AA854D85B585B785B8A89193C8
D0FBE694C691D0F5A9BF85D6,579
1800 DATA 85D7989980099000A99000BC8D0
F420BA18A201BDBE159D00D095D4CA10F5E88E
C6028690A92E8D2F028DC502,939
1810 DATA A9088D07D4A9038D1DD0A9C48DC0
02A9348DC102A9B08D0002A91A8D0102A95A8D
3002A9128D3102A9C08D0ED4,667
1820 DATA 2094186690900 7B1930980189002
B1936A9193C8D0F8E694C691D0F2A906859 1A9
008593A937859488B1932A91,502
1830 DATA 9398D0F7C694C691D0F126908595
8597859 3A9328596A92C8598A9208594A9EC85
99A91F859AA24CA013B19511,842
1840 DATA 9791939199881 0F5A59518691485
9585979004E696E69818A59369288593900 2E6
9418A5996928859999002E69A,507
1850 DATA CAD0CB20EB1590034CA4176A9003
4CA01DA58DF04EA5A0F033202D1FC69FA5ADF0
05A59F4A09A08D03D2A59FF0,109
1860 DATA 300A0A0A49FF85ACD02CA5A16A90
0FA204A90F5DC4029DC402CA10F5C6A160AD0A
D2293FD010202D1FA91F859F,898
1870 DATA 85A0D00585A08D03D2A58CF004A5
AAF0034C6A1EAD0AD2293FD0F685A4AA9D000B
CAD0FA8D02D08D03D085ABAD,816
```

```
1880 DATA 0AD22903C903B0F785A6C6A6AD0A
D22903AAA91F4ACA10FC85A7A5ADF002A9888D
01D2AD0AD2293F691085A5A2,336
1890 DATA FE86AAAD0AD2102A20286A3AD0A
D210A6E6A44C6A1EA9208596A9108598AAA900
85958597A8B19519788D0F9,189
1900 DATA E696E698CAD0F24C741600000000
0000000000000000000000000000000000000000
00000000000000000000000,817
```

●

CHECKSUM DATA.

(see page 90)

```
10 DATA 551,351,496,811,423,729,200,60
3,555,573,694,613,29,205,227,7060
160 DATA 773,198,962,638,491,30,155,10
5,309,55,181,206,334,77,966,5480
1060 DATA 760,927,68,666,80,109,432,21
6,301,300,274,480,171,97,546,5427
1210 DATA 698,173,54,544,895,622,754,1
6,149,907,268,802,994,991,727,8594
1360 DATA 966,902,865,957,640,603,511,
762,944,24,809,859,907,684,806,11239
1510 DATA 46,984,937,909,847,4,921,240
,546,122,58,873,948,912,843,9190
1660 DATA 104,69,785,912,950,11,983,91
3,244,72,771,121,924,234,9,7102
1810 DATA 946,608,910,613,58,50,961,29
,909,773,5857
```

●

Listing 2.
Assembly listing.

```
;   +----------------+
;   | RACE IN SPACE  |
;   +----------------+
;
; Written by Charles Bachand
;
; ORIGIN=$1000
;
; Game Playfield Area
; -------------------
;
DRAM    =   ORIGIN+$1000
MRAM    =   DRAM+$1900
ASTR    =   PAGES*512+DRAM
ASTL    =   PAGES*256+ASTR
;
; Player/Missile Area
; -------------------
;
        *=  $0800
;
PM      *=  *+$0180 ;blank space
PMM     *=  *+$80   ;missiles
PM0     *=  *+$80   ;player 0
PM1     *=  *+$80   ;player 1
PM2     *=  *+$80   ;player 2
PM3     *=  *+$80   ;player 3
;
; System Equates
; --------------
;
TRAM    =   PM      ;PM blank space
CHBAS   =   $02F4   ;char base reg
CH      =   $02FC   ;last key press
VSCROL  =   $D405   ;vertical scroll
NMIEN   =   $D40E   ;interrupt ctrl
VDSLST  =   $0200   ;DLI vector
SDMCTL  =   $022F   ;DMA enable
SDLSTL  =   $0230   ;display list
COLOR0  =   $02C4   ;color reg 0
COLOR1  =   $02C5   ;color reg 1
COLOR2  =   $02C6   ;color reg 2
COLPF1  =   $D017   ;color playfld 1
DLI     =   $80     ;DLI flag
ATRACT  =   77      ;attract mode
RTCLOK  =   20      ;real time clock
PCOLR0  =   $02C0   ;player color 0
PCOLR1  =   $02C1   ;player color 1
PCOLR2  =   $02C2   ;player color 2
PCOLR3  =   $02C3   ;player color 3
GPRIOR  =   $026F   ;priority
SIZEP2  =   $D00A   ;size player 2
SIZEP3  =   $D00B   ;size player 3
HPOSP0  =   $D000   ;pos player 0
HPOSP2  =   $D002   ;pos player 2
HPOSP3  =   $D003   ;pos player 3
HPOSM0  =   $D004   ;pos missile 0
SIZEM   =   $D00C   ;missile size
VDELAY  =   $D01C   ;vertical delay
GRACTL  =   $D01D   ;graphics ctrl
CONSOL  =   $D01F   ;console keys
PMBASE  =   $D407   ;P/M base addr
AUDF1   =   $D200   ;audio freq 1
AUDF2   =   $D202   ;audio freq 2
AUDF3   =   $D204   ;audio freq 3
AUDF4   =   $D206   ;audio freq 4
AUDC1   =   $D201   ;audio volume 1
AUDC2   =   $D203   ;audio volume 2
AUDC3   =   $D205   ;audio volume 3
AUDC4   =   $D207   ;audio volume 4
AUDCTL  =   $D208   ;audio control
SKCTL   =   $D20F   ;serial ctrl
RANDOM  =   $D20A   ;random number
STRIG   =   $0284   ;joystick trig 0
STICK   =   $0278   ;joystick 0
PAGES   =   $06     ;scrn block size
P0PF    =   $D004   ;P0/PF collision
P1PF    =   $D005   ;P1/PF collision
M0PL    =   $D008   ;M0/PL collision
M1PL    =   $D009   ;M1/PL collision
P0PL    =   $D00C   ;P0/PL collision
P1PL    =   $D00D   ;P1/PL collision
HITCLR  =   $D01E   ;collision clear
PACTL   =   $D302   ;port A control
;
; Page Zero equates
; -----------------
;
        *=  $80
;
SELPNT  *=  *+2     ;menu select pnt
CONSAV  *=  *+1     ;console save
CLOCK   *=  *+1     ;local clock
SLINE   *=  *+1     ;scroll line cnt
VOLUME  *=  *+1     ;intro volume
DIRSW   *=  *+1     ;vol direction
OPTION  *=  *+1     ;OPTION key indx
OPTSW   *=  *+1     ;OPTION switch
TRIG    *=  *+1     ;Trigger option
DENS    *=  *+1     ;starfield dens
SHIP    *=  *+1     ;ship type
COME    *=  *+1     ;comet enabled
UNIV    *=  *+1     ;inverse univ
SOFSET  *=  *+1     ;SELECT offset
TEMP    *=  *+4     ;temp registers
GRPAGE  *=  *+2     ;screen pointer
GRP1    *=  *+2     ;asteroid left
GRP2    *=  *+2     ;asteroid right
GRP20P  *=  *+2     ;scrn pntr -20
GRPX    *=  *+2     ;player gr pntr
GRPM    *=  *+2     ;missile gr pntr
FLASHC  *=  *+1     ;univ flash cntr
FLASHF  *=  *+1     ;univ flash flag
REVF    *=  *+1     ;inv univ flag
HPOS    *=  *+1     ;comet H pos
HDIR    *=  *+1     ;comet H dir
HINC    *=  *+1     ;comet H speed
VPOS    *=  *+1     ;comet V pos
VDIR    *=  *+1     ;comet V dir
VINC    *=  *+1     ;comet V speed
DEAD    *=  *+2     ;dead ship flg
COMETF  *=  *+1     ;comet set flag
CSOUND  *=  *+1     ;comet snd cntr
UNIVS   *=  *+1     ;universe sound
ENDGAM  *=  *+1     ;game over flag
SCORES  *=  *+2     ;game scores
TRIGN   *=  *+2     ;processed trigs
TRIGS   *=  *+2     ;last triggers
ROTATE  *=  *+1     ;rotate index
SCLOCK  *=  *+1     ;score snd timer
VDEL    *=  *+1     ;vert delay
MISSLE  *=  *+2     ;shot flags
ROWAC   *=  *+2     ;shot row acc
DELTAR  *=  *+2     ;shot delta row
ENDPT   *=  *+2     ;shot end point
ROWCRS  *=  *+2     ;shot row cursor
ROWINC  *=  *+2     ;shot row inc
COLAC   *=  *+2     ;shot column acc
DELTAC  *=  *+2     ;delta column
COLCRS  *=  *+2     ;column cursor
COLINC  *=  *+2     ;shot column inc
COUNTR  *=  *+2     ;shot delta cnt
NEWCOL  *=  *+1     ;shot end column
NEWROW  *=  *+1     ;shot end row
OLDCOL  *=  *+1     ;shot old column
OLDROW  *=  *+1     ;shot old row
TEMPM   *=  *+1     ;shot temporary
M0PLS   *=  *+2     ;M0 col shadow
XPLR    *=  *+2     ;ship X coords
YPLR    *=  *+2     ;ship Y coords
;
        *=  ORIGIN
;
; Redefined Character Set Data
; ----------------------------
;
CHARS   .BYTE 0,0,0,0,0,0,0,0
        .BYTE 0,0,0,0,$7B,50,50,50
        .BYTE 0,0,0,0,$60,$A0,$A0,32
        .BYTE 0,$7E,66,66,$7E,66,66,0
        .BYTE 0,$7E,64,64,64,64,$7E,0
        .BYTE 0,$7C,70,70,70,70,$7C,0
        .BYTE 0,$7E,64,$7C,64,64,$7E,0
        .BYTE 0,$7E,64,$7C,64,64,64,0
        .BYTE 0,$7E,64,64,78,66,$7E,0
        .BYTE 0,66,66,$7E,66,66,66,0
        .BYTE 0,$7E,24,24,24,24,$7E,0
        .BYTE 0,64,64,64,64,64,$7E,0
        .BYTE 0,$7E,66,66,66,66,66,0
        .BYTE 0,$7E,66,66,$7E,64,64,0
        .BYTE 0,$7E,66,66,$7E,76,70,0
        .BYTE 0,$7E,24,24,24,24,24,0
        .BYTE 0,66,66,66,90,60,$7E,0
        .BYTE 0,66,66,$7E,24,24,24,0
        .BYTE $7E,$81,$BD,$A1
        .BYTE $A1,$BD,$81,$7E
        .BYTE 0,$7E,66,66,66,66,$7E,0
        .BYTE 0,0,0,0,0,0,0,0
        .BYTE 0,$7E,2,$7E,64,64,$7E,0
        .BYTE 0,$7E,2,$7E,2,2,$7E,0
        .BYTE 0,66,66,$7E,2,2,2,0
        .BYTE 0,$7E,64,$7E,2,2,$7E,0
        .BYTE 0,$7E,64,$7E,66,66,$7E,0
        .BYTE 0,$7E,2,2,2,2,2,0
        .BYTE 0,$7E,66,$7E,66,66,$7E,0
        .BYTE 0,$7E,66,$7E,2,2,$7E,0
        .BYTE 0,0,0,0,0,0,0,0
        .BYTE 24,36,90,0,0,0,0,0
        .BYTE 0,0,24,66,66,24,0,0
        .BYTE 0,0,0,0,$7E,$7E,$7E,60
        .BYTE 0,0,0,0,$7E,$FF,$FF,$FF
        .BYTE 0,0,0,0,0,66,$7E,$7E
        .BYTE 0,0,0,0,0,0,24,66
        .BYTE 0,0,0,0,0,0,0,24
        .BYTE 0,0,0,0,0,0,0,0
        .BYTE 0,0,0,0,0,0,0,0
        .BYTE 0,0,0,0,0,0,0,0
        .BYTE 66,66,66,66,66,66,66,66
        .BYTE $FF,$FF,$FF,$7E,0,0,0,0
        .BYTE $7E,$7E,66,0,0,0,0,0
        .BYTE 66,24,0,0,0,0,0,0
        .BYTE 24,0,0,0,0,0,0,0
        .BYTE 0,24,24,24,24,66,$7E,90
        .BYTE 24,24,24,66
        .BYTE $7E,$FF,$DB,0
        .BYTE 0,0,0,0,0,0,0,0
        .BYTE 0,66,66,66,66,66,$7E,0
        .BYTE 0,0,24,24,0,24,24,0
        .BYTE 0,66,66,66,66,36,24,0
```

```
          .BYTE 0,$7E,90,90,66,66,66,0
          .BYTE 0,70,76,$78,88,76,70,0
          .BYTE $FF,$FF,$FF,$FF
          .BYTE $FF,$FF,$FF,$FF
          .BYTE $7E,66,66,0,66,66,$7E,0
          .BYTE 2,2,2,0,2,2,2,0
          .BYTE 62,2,2,60,64,64,$7C,0
          .BYTE 62,2,2,60,2,2,62,0
          .BYTE 66,66,66,60,2,2,2,0
          .BYTE $7C,64,64,60,2,2,62,0
          .BYTE $7C,64,64,60,66,66,$7E,0
          .BYTE 62,2,2,0,2,2,2,0
          .BYTE $7E,66,66,60,66,66,$7E,0
          .BYTE $7E,66,66,60,2,2,62,0
;
; Introduction Display List
; -------------------------
;
DISPI     .BYTE $70,$67
DADR      .WORD DRAM
          .BYTE $27,$27,$27,$27
          .BYTE $27,$27,$27,$27
          .BYTE $27,$27,$27,$27
          .BYTE $27,$07,$41+DLI
          .WORD DISPI
;
; Game Options Display List
; -------------------------
;
DISPO     .BYTE $70,$70,$70,$46
          .WORD OPTMSG
          .BYTE $07,$46
          .WORD SB
          .BYTE $46
          .WORD OL1
          .BYTE $06,$06,$46
          .WORD SB
          .BYTE $46
          .WORD OL2
          .BYTE $46
DOT       .WORD TOPT
          .BYTE $46
          .WORD SB
          .BYTE $46
          .WORD OL3
          .BYTE $46
DOD       .WORD DOPT
          .BYTE $46
          .WORD SB
          .BYTE $46
          .WORD OL4
          .BYTE $46
DOS       .WORD SOPT
          .BYTE $46
          .WORD SB
          .BYTE $46
          .WORD OL5
          .BYTE $46
DOC       .WORD COPT
          .BYTE $46
          .WORD SB
          .BYTE $46
          .WORD OL6
          .BYTE $46
DOU       .WORD UOPT
          .BYTE $46
          .WORD SB
          .BYTE $46
          .WORD OPTMSG
          .BYTE $41
          .WORD DISPO
;
; Game Playfield Display List
; ---------------------------
;
DISPG     .BYTE $70,$70,$70,$4F
          .WORD DRAM
          .DBYTE 15,15,15,15,15,15,15,15
          .DBYTE 15,15,15,15,15,15,15,15
          .DBYTE 15,15,15,15,15,15,15,15
          .DBYTE 15,15,15,15,15,15,15,15
          .DBYTE 15,15,15,15,15,15,15,15
          .DBYTE 15,15,15,15,15,15,15,15
          .DBYTE 15,15,15,15,15,15,15,15
          .DBYTE 15,15,15,15,15,15,15,15
          .DBYTE 15,15,15,15,15,15,15,15
          .DBYTE 15,15,15,15,15
          .BYTE $70,$30,$47+DLI
          .WORD TRAM
          .BYTE $30,$42
          .WORD ANALOG
          .BYTE $41
          .WORD DISPG
;
; Flying saucer in Intro Data
; ---------------------------
;
SAUCER    .BYTE $18,$7E,$7E,$FF
          .BYTE $FF,$7E,$7E,$18
WINDOW    .BYTE $00,$00,$00,$55
          .BYTE $55,$00,$00,$00
;
; Redefined Char Set Equates
; --------------------------
;
TMC       =   1
AC        =   3
CC        =   4
DC        =   5
EC        =   6
FC        =   7
GC        =   8
HC        =   9
IC        =   10
LC        =   11
NC        =   12
PC        =   13
RC        =   14
TC        =   15
WC        =   16
YC        =   17
CRC       =   18
N0C       =   19
BC        =   N0C+8
QC        =   N0C
SC        =   QC+5
SMK       =   N0C+10
SMK2      =   SMK+8
SHC       =   SMK2+8
MT        =   SHC+2
UC        =   MT+1
COLC      =   UC+1
VC        =   COLC+1
MC        =   VC+1
KC        =   MC+1
BX        =   KC+$41
N7C       =   KC+2
;
; Introduction Title Data
; -----------------------
;
TITLE     .BYTE 21,AC,NC,AC,LC
          .BYTE QC,GC,$FF,3A,MC
          .BYTE AC,GC,AC,N0C+2,IC
          .BYTE NC,EC,$FF,68
          .BYTE +$40,PC,RC,EC,SC
          .BYTE +$40,EC,NC,TC,SC
          .BYTE $FF,97
          .BYTE +$80,RC,AC,CC,EC,0
          .BYTE +$80,IC,NC,0,SC,PC
          .BYTE +$80,AC,CC,EC
          .BYTE TMC,TMC+1,$FF,131
          .BYTE +$40,WC,RC,IC,TC,TC,EC
          .BYTE +$40,NC,0,BC,YC
          .BYTE $FF,160
          .BYTE +$C0,CC,HC,AC,RC,LC,EC
          .BYTE +$C0,BC,0,0,BC,AC,CC
          .BYTE +$C0,HC,AC,NC,DC
          .BYTE $FF,197,CRC,0
          .BYTE N0C+1,N0C+9,N0C+8
          .BYTE N0C+4,$FF,230
          .BYTE +$80,SHC,0,0,SHC
          .BYTE $FF,246
          .BYTE +$40,SHC+1,0,0,SHC+1
          .BYTE $FF
;
; PRESS OPTION Message Data
; -------------------------
;
PMSG      .BYTE 0,0,0,0,0,0,0,0,0,0
          .BYTE +$C0,PC,RC,EC,SC,SC
          .BYTE +$C0,0,QC,PC,TC,IC
          .BYTE +$C0,QC,NC,0,0
          .BYTE 0,0,0,0,0,0,0,0,0,0
;
; Options Screen Data
; -------------------
;
OPTMSG    .BYTE BX,BX,BX,BX,BX,BX,BX,BX
          .BYTE BX,BX,BX,BX,BX,BX,BX,BX
          .BYTE BX,0,QC,PC,TC,IC,QC,NC
          .BYTE SC
          .BYTE +$C0,0,0,TC,COLC
TIMOPT    .BYTE +$80,N0C+3,0
OL1       .BYTE BX,0
          .BYTE +$40,QC,PC,TC,IC
          .BYTE +$40,QC,NC,COLC,0
          .BYTE +$80,LC,EC,FC,TC,0
          .BYTE BX,BX,0
          .BYTE +$40,SC,EC,LC,EC,CC
          .BYTE +$40,TC,COLC
          .BYTE +$C0,RC,IC,SC,HC,TC,0
          .BYTE BX,BX,0
          .BYTE +$40,SC,TC,AC,RC,TC
          .BYTE +$40,COLC
          .BYTE 0,0,PC,LC,AC,YC,0
OL2       .BYTE BX,0,TC,RC,IC,SC,SC,EC
          .BYTE RC,COLC,0,0,0,0,0,BX
OL3       .BYTE BX,0
          .BYTE +$80,DC,EC,NC,SC,IC,TC
          .BYTE +$80,YC,COLC,0,0
          .BYTE 0,0,0,BX
OL4       .BYTE BX,0
          .BYTE +$80,SC,HC,IC,PC,SC
          .BYTE +$80,COLC
          .BYTE 0,0,0,0,0,0,0,BX
OL5       .BYTE BX,0
          .BYTE +$80,CC,QC,MC,EC,TC
          .BYTE +$80,SC,COLC
          .BYTE 0,0,0,0,0,0,0,BX
OL6       .BYTE BX,0
          .BYTE +$80,UC,NC,IC,VC
          .BYTE +$80,EC,RC,SC,EC,COLC
          .BYTE 0,0,0,0,BX
SB        .BYTE BX,0,0,0,0,0,0,0,0,0
TOPT      .BYTE BX,0,0,0,0,0,0
          .BYTE +$C0,NC,QC,0,EC,FC,FC
          .BYTE +$C0,EC,CC,TC,0
TOPT2     .BYTE BX,0,0,0,0,0,0
          .BYTE +$C0,SC,AC,IC,EC
          .BYTE +$C0,LC,DC,SC,0
TOPT3     .BYTE BX,0,0,0,0,0,0
          .BYTE +$C0,NC,IC,SC,SC
          .BYTE +$C0,IC,LC,EC,SC,0
TOPT4     .BYTE BX,0,0,0,0
          .BYTE +$C0,WC,AC,RC,PC,0
          .BYTE +$C0,DC,RC,IC,VC,EC,0
DOPT      .BYTE BX,0,0,0,0,0,0
          .BYTE +$C0,SC,TC,AC,NC
          .BYTE +$C0,DC,AC,RC,DC,0
DOPT2     .BYTE BX,0,0,0,0,0,0,0,0
          .BYTE +$C0,DC,QC,UC,SC,LC
          .BYTE +$C0,EC,0
DOPT3     .BYTE BX,0,0,0,0,0,0,0,0
          .BYTE +$C0,TC,RC,IC,PC,LC
          .BYTE +$C0,EC,0
DOPT4     .BYTE BX,0,0,0,0,0,0,0,0
          .BYTE +$C0,SC,QC,PC,EC,RC,0
SOPT      .BYTE BX,0,0,0,0,0,0,0,0
          .BYTE +$C0,RC,QC,CC,KC
          .BYTE +$C0,EC,TC,SC,0
SOPT2     .BYTE BX,0,0,0,0,0,0
          .BYTE +$C0,SC,AC,UC,CC,EC
          .BYTE +$C0,RC,SC,0
COPT      .BYTE BX,0,0,0,0,0,0
          .BYTE +$C0,DC,IC,SC,AC
          .BYTE +$C0,BC,LC,EC,DC,0
COPT2     .BYTE BX,0,0,0,0,0,0,0
          .BYTE +$C0,EC,NC,AC,BC
          .BYTE +$C0,LC,EC,DC,0
UOPT      .BYTE BX,0,0,0,0,0,0
          .BYTE +$C0,PC,QC,SC,IC
UOPT2     .BYTE +$C0,TC,IC,VC,EC,0
          .BYTE BX,0,0,0,0,0,0
          .BYTE +$C0,NC,EC,GC,AC,TC
          .BYTE +$C0,IC,VC,EC
          .BYTE 0,BX
ANALOG    .BYTE 0,0,0,0,CC,QC,PC,YC
          .BYTE RC,IC,BC,HC,TC,0
          .BYTE CRC,0,AC,NC,AC,LC
          .BYTE QC,BC,0,MC,AC,BC,AC
          .BYTE N0C+2,IC,NC,EC,0,N0C+1
          .BYTE N0C+9,N0C+8,N0C+4
          .BYTE 0,0,0,0
TRAMI     .BYTE 0,0,N0C,N0C,0,0,0
TRAMC     .BYTE +$C0,N7C+3,COLC,N7C
          .BYTE +$C0,N7C,SMK+7,N7C
          .BYTE 0,0,0,N0C,N0C,0,0
TRAMN     .BYTE 3,0,0,0,0
NUMMSK    .BYTE $1F,$1E,$1A,$18
          .BYTE $1D,$1B,$33,$35,$30
MISMSK    .BYTE $FC,$F3
MISIMA    .BYTE 3,0,3,1,1,1,3
          .BYTE 0,3,2,3,2,2,3,2
MIMAGE    .BYTE 0,0,0
MCMSK     .BYTE $0E,$0D
LADR      .WORD DOD,DOS,DOC,DOU,DOT
MADR      .WORD TOPT,TOPT2,TOPT3,TOPT4
          .WORD DOPT,DOPT2,DOPT3,DOPT4
          .WORD SOPT,SOPT2,COPT
          .WORD COPT2,UOPT,UOPT2
SELMSK    .BYTE 3,3,1,1
SELMS2    .BYTE 0,4,1,2,2
SHIPS     .BYTE 0,$18,$3C,$3C,$18
          .BYTE $5A,$7E,$5A,$42
          .BYTE 0,$18,$7E,$DB
          .BYTE $7E,0,0,0,0
ROTMSK    .BYTE $B6,$6D,$DB,$B6
VDMSK     .BYTE $10,$20,$01,$02
XSTRT     .BYTE 88,160
SCPNT     .BYTE 3,17
SCNOTE    .BYTE $50,$30
PCOLRS    .BYTE $C4,$34
COMETM    .BYTE 0,$3C,$7E,$7E
          .BYTE $FF,$FF,$FF,$FF,$FF
          .BYTE $FF,$7E,$7E,$3C,0
CE3K      .BYTE $C6,$46,$F6,$66,$26
CE3KN     .BYTE $6C,$A2,$51,$40,$48
;
; DLI Routine
; -----------
;
DLIVEC    PHA             ;save Acc
          INC CLOCK       ;increment clock
          LDA RTCLOK      ;system clock
          ASL A           ;times 2
          AND #$CE        ;mask color
          STA PCOLR2      ;saucer color
          PLA             ;restore Acc
          RTI             ;return
;
; Console Checker
; ---------------
;
CONC      LDA CONSOL      ;get console
          TAY             ;to Y register
          EOR CONSAV      ;only different
          AND CONSAV      ;pressed keys
          STY CONSAV      ;save current
          CMP #4          ;OPTION test
          RTS             ;return
;
; Beep Routine
; ------------
;
BUZZER    STA AUDF2       ;buzzer freq
          LDA #$A4        ;pure tone
          STA AUDC2       ;in tone 2
          LDY #$C0        ;beep counter
BZ        DEX             ;decrement X
          BNE BZ          ;zero yet? No.
          STX ATRACT      ;poke attract
          DEY             ;decrement Y
          BNE BZ          ;tone done? No.
          STY AUDC2       ;Yes. turn off
          RTS             ;return
;
; Flying Saucer Routine
; ---------------------
;
DELAY     LDX #0          ;get zero
          LDA #$A4        ;pure tone
          STA AUDC1       ;enable tone 1
          STA AUDC2       ;enable tone 2
DL1       LDA #$12        ;1/5 sec count
DL2       CMP CLOCK       ;compare clock
          BNE DL2         ;match? No.
          JSR CONC        ;check console
          BNE DL3         ;OPTION key? No.
          PLA             ;Yes. pull RTS
          PLA             ;addr from stack
          JMP STOPTS      ;options menu
DL3       LDA #0          ;get zero
          STA CLOCK       ;init clock
          INX             ;inc saucer pos
          TXA             ;move to Acc
          STA HPOSP3      ;move saucer
          AND #$FC        ;window movement
          STA HPOSP2      ;do windows
          LSR A           ;times 2
          LSR A           ;times 4
          LSR A           ;times 8
          TAY             ;index to displa
          LDA PMSG,Y      ;get text
          STA DRAM+313,Y  ;on screen
          TXA             ;saucer position
          ASL A           ;times 2
          ASL A           ;times 4
          ASL A           ;times 8
          ASL A           ;times 16
          ASL A           ;times 32 whew!
          EOR #$F0        ;inverse hi bits
          STA AUDF1       ;saucer sound
          CLC             ;clear carry
          ADC #2          ;weird sound
          STA AUDF2       ;with resonance
          TXA             ;saucer position
          BNE DL1         ;off screen? No.
          STA AUDC2       ;sound 2 off
          STA VOLUME      ;saucer volume
          RTS             ;return
;
; Move All P/M Off Screen
; -----------------------
;
CLRPM     LDX #7          ;8 objects 0-7
          LDA #0          ;zero position
CPM       STA HPOSP0,X    ;move P/M
          DEX             ;decrement count
          BPL CPM         ;done? No.
          RTS             ;return
;
; Turn Off All Sound
; ------------------
;
AUDOFF    LDA #0          ;get zero
          STA AUDC1       ;sound 1 off
          STA AUDC2       ;sound 2 off
          STA AUDC3       ;sound 3 off
          STA AUDC4       ;sound 4 off
          RTS             ;return
;
; Program Execution Starts Here
; -----------------------------
;
RIS       LDA #$3C        ;cassette off
          STA PACTL       ;poke hardware
          LDA # <DOT      ;density lo
          STA SELPNT      ;select point lo
          LDA # >DOT      ;density hi
          STA SELPNT+1    ;select pnt hi
          LDA #0          ;init sound
          STA AUDCTL      ;audio control
          LDX #YPLR+1-CONSAV
I1        STA CONSAV,X    ;zero flags
```

```
          DEX             ;decrement index
          BPL I1          ;done? No.
;
; System Reset Returns Here
; -------------------------
;
INTRO     LDA #$29        ;value to enable
          STA SDMCTL      ;narrow playfld
          LDA #3          ;value to turn
          STA SKCTL       ;off 2-tone mode
          LDA # >MRAM     ;high byte of
          STA PMBASE      ;P/M address
          LDA #2          ;value to
          STA GRACTL      ;enable players
          LDA # <DLIVEC   ;DLI addr lo
          STA VDSLST      ;DLI vector lo
          LDA # >DLIVEC   ;DLI addr hi
          STA VDSLST+1    ;DLI vector hi
          LDA #$C0        ;value to
          STA NMIEN       ;enable DLI's
          LDA # >CHARS    ;addr of
          STA CHBAS       ;character set
          LDA #$C6        ;value for
          STA COLOR1      ;medium green
          LDA #$94        ;value for
          STA COLOR2      ;dark blue
          LDA #$04        ;value for
          STA PCOLR3      ;dark grey
          LDA #1          ;value - player
          STA SIZEP2      ;double width
          STA SIZEP3      ;players set
          STA GPRIOR      ;PL has priority
          JSR CLRPM       ;remove PLayers
          STA REVF        ;rev screen flag
          STA DIRSW       ;sound direction
          JSR AUDOFF      ;sound off
          TAX             ;zero index
I2        STA DRAM,X      ;clear display
          STA DRAM+256,X  ;page 2
          STA DRAM+512,X  ;page 3
          STA MRAM+512,X  ;player 2+3
          DEX             ;decrement index
          BNE I2          ;done? No.
          LDX #7          ;move 8 bytes
I3        LDA WINDOW,X    ;saucer windows
          STA MRAM+$0228,X ;player 2
          LDA SAUCER,X    ;saucer ship
          STA MRAM+$02A8,X ;player 3
          DEX             ;decrement index
          BPL I3          ;saucer done? No.
;
; Print Text Onto Display
; -----------------------
;
PTITL     INX             ;increment index
          LDA TITLE,X     ;get text
          BEQ STITL       ;byte zero? Yes.
          TAY             ;display index
P2        INX             ;inc text index
          LDA TITLE,X     ;get text
          CMP #$FF        ;EOL flag
          BEQ PTITL       ;EOL? Yes.
          STA DRAM+256,Y  ;on screen
          INY             ;inc displa indx
          JMP P2          ;continue
;
; Scroll Text For Intro
; ---------------------
;
STITL     LDA # <DISPI    ;DL addr lo
          STA SDLSTL      ;DL pntr lo
          LDA # >DISPI    ;DL addr hi
          STA SDLSTL+1    ;DL pntr hi
          LDA # >DRAM     ;screen top hi
          STA DADR+1      ;DL LMS hi
          LDA #0          ;get zero
          STA DADR        ;DL LMS lo
          STA ATRACT      ;poke attract
          STA VOLUME      ;clear volume
S1        LDA #$10        ;16/60 sec count
S2        BIT CLOCK       ;check clock
          BEQ S2          ;time up? No.
          JSR CONC        ;console keys
          BNE S2A         ;OPTION key? No.
          JMP STOPTS      ;option menu
S2A       LDA #0          ;get zero
          STA CLOCK       ;reset clock
          INC SLINE       ;inc scroll cnt
          LDA SLINE       ;scroll count
          CLC             ;clear carry
          ADC DADR        ;screen addr lo
          EOR DIRSW       ;sound direction
          STA AUDF1       ;rocket sound
          LDA SLINE       ;scroll count
          AND #7          ;use only 0..7
          ADC #SMK+$C0    ;smoke offset
          STA DRAM+518    ;put below both
          STA DRAM+521    ;rocket ships
          ADC #8          ;both smk offset
          STA DRAM+534    ;put below top
          STA DRAM+537    ;smoke of ships
          LDA SLINE       ;scroll count
          CMP #16         ;check overflow?
          BNE S3          ;No. continue
          LDA #0          ;get zero
          STA SLINE       ;reset count
          STA VSCROL      ;vertical scroll
          INC VOLUME      ;raise volume
          LDA DIRSW       ;sound direction
          EOR VOLUME      ;EOR with volume
          ORA #$80        ;rough sounding
          STA AUDC1       ;rocket sound
          LDA DADR        ;DL LMS lo
          CLC             ;clear carry
          ADC #16         ;line width
          STA DADR        ;new DL LMS lo
          BCC S3          ;overflow? No.
          INC DADR+1      ;inc LMS hi
          LDA #2+ >DRAM   ;cmp scrn end
          CMP DADR+1      ;with LMS hi
          BNE S4          ;at end? No.
          JMP INTRO       ;Yes. do intro
S4        JSR DELAY       ;flying saucer
          LDA #$20        ;value for
          STA AUDF1       ;reset frequency
          LDA #$8F        ;full volume
          STA AUDC1       ;reset tone
          LDA #$0F        ;reverse sound
          STA DIRSW       ;sound direction
S3        LDA SLINE       ;scroll count
          STA VSCROL      ;vertical scroll
          JMP S1          ;continue
;
; Game Option Menu Routine
; ------------------------
;
STOPTS    LDA CLOCK       ;allow one VBLANK
STP       CMP CLOCK       ;period to go by
          BEQ STP         ;before start
          LDA #$21        ;value for
          STA SDMCTL      ;narrow playfld
          LDA # <DISPO    ;DL addr lo
          STA SDLSTL      ;displa list lo
          LDA # >DISPO    ;DL addr hi
          STA SDLSTL+1    ;displa list hi
          JSR SCRCLR      ;clr inv Univ
          JSR CLRPM       ;clear players
          STA ATRACT      ;poke attract
          STA GRACTL      ;no players
          JSR AUDOFF      ;sound off
          STA FLASHF      ;clr flash flag
          STA REVF        ;clr inv flag
          LDA #$94        ;color value for
          STA COLOR2      ;medium blue
          LDA #4          ;five notes 0-4
          STA TEMP+3      ;same index
STS       LDX TEMP+3      ;get index
          LDA CE3K,X      ;get color
          STA COLOR1      ;change color
          LDA CE3KN,X     ;get note
          JSR BUZZER      ;play note
          DEC TEMP+3      ;dec index
          BPL STS         ;done? No.
ST1       LDA CH          ;check keyboard
          AND #$3F        ;mask CTRL/SHIFT
          LDX #9          ;check for 0-9
ST1A      CMP NUMMSK-1,X  ;cmp keycode
          BEQ ST1B        ;numeric? Yes.
          DEX             ;decrement index
          BNE ST1A        ;done? No.
          BEQ ST1C        ;Yes. continue
ST1B      TXA             ;index is number
          STA TRAMN       ;store time
          CLC             ;clear carry
          ADC #N0C+$80    ;display offset
          STA TIMOPT      ;show time opt
          TXA             ;number again
          ADC #N7C+$C0    ;2nd char set
          STA TRAMC       ;game score line
          LDA #$FF        ;clear the
          STA CH          ;keyboard buffer
          TXA             ;number again
          ASL A           ;times 2
          ASL A           ;times 4
          EOR #$3F        ;inverse here to
          JSR BUZZER      ;number sound
ST1C      JSR CONC        ;check console
          LSR A           ;check START key
          BCC ST2         ;START key? No.
          JMP PLAYGM      ;Yes. play game
;
; SELECT Key Handler
; ------------------
;
ST2       LSR A           ;check SELECT
          BCC ST3         ;SELECT key? No.
          LDA #$30        ;Yes. do SELECT
          JSR BUZZER      ;key noise
          LDX OPTION      ;opt to change
          INC TRIG,X      ;inc option byte
          LDA TRIG,X      ;get option byte
          AND SELMSK,X    ;mask overflow
          STA TRIG,X      ;save opt byte
          ORA SOFSET      ;SELECT offset
          ASL A           ;times 2
          TAX             ;use as X index
          LDY #0          ;zero Y index
          LDA MADR,X      ;text addr lo
          STA (SELPNT),Y  ;LMS byte lo
          INY             ;inc Y index
          LDA MADR+1,X    ;text addr hi
          STA (SELPNT),Y  ;LMS byte hi
          JMP ST1         ;continue
;
; OPTION Key Handler
; ------------------
;
ST3       LSR A           ;check OPTION
          BCC ST1         ;OPTION key? No.
          LDA #$10        ;Yes. do OPTION
          JSR BUZZER      ;key sound
          LDX #1          ;reset text flag
ST4       STX OPTSW       ;option switch
          LDA OPTION      ;option counter
          ASL A           ;times 2
          ASL A           ;times 4
          ASL A           ;times 8
          ASL A           ;times 16
          TAX             ;use as index
          LDY #13         ;do 14 bytes
ST4A      LDA #$80        ;get sign bit
          EOR OL2+1,X     ;flip sign
          STA OL2+1,X     ;save byte
          INX             ;increment index
          DEY             ;decrement count
          BPL ST4A        ;done? No.
          LDA OPTSW       ;option switch
          BNE ST4B        ;flipping done?
          JMP ST1         ;Yes. continue
ST4B      INC OPTION      ;inc option cntr
          LDA OPTION      ;option counter
          ASL A           ;times 2
          TAX             ;use as index
          LDA LADR-2,X    ;DL LMS addr lo
          STA SELPNT      ;select pntr lo
          LDA LADR-1,X    ;DL LMS addr hi
          STA SELPNT+1    ;select pntr hi
          LDX OPTION      ;option counter
          LDA SELMS2,X    ;offset table
          ADC SOFSET      ;present offset
          STA SOFSET      ;new offset
          LDX #0          ;get zero
          LDA OPTION      ;option counter
          CMP #5          ;range: 0..4
          BNE ST4         ;overflow? No.
          STX OPTION      ;reset counter
          STX SOFSET      ;reset offset
          JMP ST4         ;continue
;
; Asteroid Field Initializer
; --------------------------
; Initialize pointers
; -------------------
;
LDGRRT    LDA # <ASTR     ;asteroid right
          STA GRPAGE      ;field addr lo
          TAY             ;use as index
          LDA # >ASTR     ;asteroid right
          STA GRPAGE+1    ;field addr hi
          LDA #PAGES      ;# of 256 byte
          STA TEMP+2      ;blocks to move
          RTS             ;return
;
; Generate Asteroid Field Bytes
; -----------------------------
;
RANWRD    LDA #2          ;# 4-bit nibbles
          STA TEMP        ;save counter
RANW0     CPX RANDOM      ;with density
          ROL TEMP+1      ;carry to bit 0
          ASL TEMP+1      ;to bit 1
          ASL TEMP+1      ;to bit 2
          ASL TEMP+1      ;to bit 3
          DEC TEMP        ;nibble count
          BNE RANW0       ;byte done? No.
          LDA TEMP+1      ;move byte to
          STA (GRPAGE),Y  ;graphic area
          RTS             ;return
;
; Fill Workspace With Graphics
; ----------------------------
;
RANFIL    JSR LDGRRT      ;init pointers
          ASL TEMP+2      ;blocks times 2
          LDX DENS        ;get density
RANF0     JSR RANWRD      ;make starfield
          INY             ;inc block index
          BNE RANF0       ;block done? No.
          INC GRPAGE+1    ;inc page pntr
          DEC TEMP+2      ;dec block cntr
          BNE RANF0       ;done? No.
;
; Shift ASTR Space To Right
; -------------------------
;
          DEC GRPAGE+1    ;dec page pntr
          LDA #PAGES      ;get block count
          STA TEMP+2      ;store count
RANF1     DEY             ;dec index
          LDA (GRPAGE),Y  ;asteroid byte
          LSR A           ;force to add
          STA (GRPAGE),Y  ;replace byte
          TYA             ;exam index
          BNE RANF1       ;index=0? No.
          DEC GRPAGE+1    ;back up pntr
          DEC TEMP+2      ;decrement count
          BNE RANF1       ;Done? No.
          RTS             ;return
;
; Ship Scoring Routine
; --------------------
;
SCORE     LDY #10         ;clr top 16 bytes
SC0       STA PM0,Y       ;of player 0
          STA PM1,Y       ;and player 1
          DEY             ;decrement index
          BPL SC0         ;done? No.
          LDA XSTRT,X     ;start position
          STA XPLR,X      ;to player pos
          STA HPOSP0,X    ;set hardware
SCOREM    TXA             ;save X to Acc
          PHA             ;push Acc
          LDA SCPNT,X     ;score Y pos
          TAX             ;use as index
          INC TRAM,X      ;increment score
          LDA TRAM,X      ;get score byte
          CMP #N0C+10     ;value over 9?
          BCC SC1         ;No. skip
          LDA #N0C        ;get zero char
          STA TRAM,X      ;to 1's position
          INC TRAM-1,X    ;inc 10's pos
SC1       PLA             ;pull Acc
          TAX             ;get X indx back
          LDA SCNOTE,X    ;score sound
          STA AUDF3       ;set frequency
          LDA #$AE        ;loud sound
          STA AUDC3       ;set volume
          LDA #3          ;value for
          STA SCLOCK      ;sound duration
          DEC SCORES,X    ;dec max score
          BNE SC2         ;from 99 to zero
          LDA #0          ;99 points so
          STA ENDGAM      ;end the game
SC2       LDA #$FF        ;value to return
          STA HITCLR      ;clear collision
          RTS             ;return
;
; Ship Graphics Rotation
; ----------------------
;
ROTOR     TXA             ;move X to Acc
          PHA             ;save X
          LDA CLOCK       ;get clock
          ROR A           ;test bit 0
          BCS RT2         ;bit=1? Yes.
          DEC ROTATE      ;rotation index
          BPL RT1         ;rotate<0? No.
          LDA #2          ;reset value for
          STA ROTATE      ;rotation index
RT1       LDX ROTATE      ;get index
          LDA ROTMSK,X    ;spinner graphic
          STA SHIPS+12    ;saucer section
          AND #$3C        ;mask for rocket
          STA SHIPS+2     ;rocket upper
          LDA ROTMSK+1,X  ;next seq
          AND #$3C        ;mask for rocket
          STA SHIPS+3     ;rocket lower
          LDA CLOCK       ;get clock again
          AND #$0E        ;missile mask
          TAX             ;use as index
          LDA MISIMA,X    ;missile image
          STA MIMAGE+1    ;image buffer
          LDA MISIMA+1,X  ;missile pic+1
          STA MIMAGE+2    ;image buffer+1
RT2       PLA             ;pull Acc
          TAX             ;restore X
          RTS             ;return
;
; Countdown Timer Handler
; -----------------------
;
TIMER     DEC TRAM+24     ;1/10 sec timer
          BPL TMX         ;time up? No.
          LDA #5          ;Yes. value to
          STA TRAM+24     ;reset timer
          LDA SCLOCK      ;tic sound clock
          BEQ RT3         ;tic done? Yes.
          DEC SCLOCK      ;dec tic clock
          JMP RT4         ;continue
;
RT3       STA AUDC3       ;tic sound off
RT4       DEC TRAM+12     ;1/10 sec displa
          DEC TRAM+23     ;1/10 sec cntr
          BPL TMX         ;1 sec done? No.
          LDA #N7C+$C9    ;value to reset
          STA TRAM+12     ;1/10 sec displa
          LDA #9          ;value to reset
          STA TRAM+23     ;1/10 sec cntr
          LDA SCLOCK      ;sound clock
          BNE RT5         ;in use? Yes.
          LDA #$A4        ;value for
          STA AUDC3       ;pure tone
          LDA #$40        ;value for
          STA AUDF3       ;medium freq
RT5       DEC TRAM+10     ;one's display
          DEC TRAM+22     ;one's counter
```

```
        BPL TMX      ;10 sec up? No.
        LDA #N7C+$C9 ;value to reset
        STA TRAM+10  ;one's display
        LDA #9       ;value to reset
        STA TRAM+22  ;one's counter
        DEC TRAM+9   ;ten's display
        DEC TRAM+21  ;ten's counter
        BPL TMX      ;minute up? No.
        LDA #N7C+$C5 ;value to reset
        STA TRAM+9   ;ten's display
        LDA #5       ;value to reset
        STA TRAM+21  ;ten's counter
        DEC TRAM+7   ;minute display
        DEC TRAM+20  ;minute counter
TMX     PHP          ;save flags
        LDA SCLOCK   ;if score sound
        ORA COUNTR   ;or missile 0/1
        ORA COUNTR+1 ;active?
        BNE TMXX     ;Yes. skip next
        LDA DEAD     ;either ship 0
        AND DEAD+1   ;or ship 1
        BEQ TMXX     ;dead? Yes.
        LDA MISSLE   ;OR flags for
        ORA MISSLE+1 ;projectiles
        BEQ TMXX     ;any active? No.
        LDA #$28     ;projectile snd
        STA AUDC3    ;to hardware
        LDA CLOCK    ;60 cycle clock
        LSR A        ;make 30 cycles
        AND #$07     ;only 0..7
        ORA #8       ;8..15
        STA AUDF3    ;boomerang sound
TMXX    PLP          ;restore flags
        RTS          ;return
;
; Collision Handler
; -----------------
;
SMASH   LDA M1PL     ;missile 1 to
        AND #1       ;player 0 ship
        BNE HITP0    ;collision? Yes.
        LDA P0PF     ;player 0 to
        AND #4       ;playfield 2 or
        ORA P0PL     ;to any player
        BEQ PLR1     ;collision? No.
        LDA TRIGN    ;trigger option
        CMP #1       ;shield in use?
        BEQ PLR1     ;Yes. ship safe
HITP0   LDA #0       ;value for ship
        STA DEAD     ;being shot down
PLR1    LDA M0PL     ;missile 0 to
        AND #2       ;player 1 ship
        BNE HITP1    ;collision? Yes.
        LDA P1PF     ;player 1 to
        AND #4       ;playfield 2 or
        ORA P1PL     ;to any player
        BEQ HITX     ;collision? No.
        LDA TRIGN+1  ;trigger option
        CMP #1       ;shield in use
        BEQ HITX     ;Yes. ship safe
HITP1   LDA #0       ;value for ship
        STA DEAD+1   ;being shot down
HITX    RTS          ;return
;
; Space Boomerang Handler
; -----------------------
;
MISFLY  LDA MISSLE,X ;missile status
        BNE HITX     ;active? Yes.
        LDA TRIGN,X  ;trigger value
        CMP #2       ;shots enabled
        BEQ MISF     ;Yes. continue
        STA TRIGS,X  ;put shadow
        RTS          ;return
;
; Set up Launch Coordinates
; -------------------------
;
MISF    CMP TRIGS,X  ;compare shadow
        BEQ HITX     ;same? Yes.
        STA TRIGS,X  ;put shadow
        STA MISSLE,X ;enable missile
        CLC          ;clear carry
        LDA XPLR,X   ;ship X coord
        ADC #2       ;get ship center
        STA OLDCOL   ;X plot coord
        STA COLCRS,X ;shot current X
        LDA YPLR,X   ;ship Y coord
        STA OLDROW   ;Y plot coord
        INC OLDROW   ;ship top
        STA ROWCRS,X ;shot current Y
        INC ROWCRS,X ;plot coord
        TXA          ;player index
        BEQ MIS1     ;player 0? Yes.
;
; Player 0 is target
; ------------------
;
        LDA XPLR     ;enemy X coord
        ADC #2       ;ship center
        STA NEWCOL   ;destination X
        LDA YPLR     ;enemy Y coord
        JMP MIS2     ;skip next
;
; Player 1 is target
; ------------------
;
MIS1    LDA XPLR+1   ;enemy X coord
        ADC #2       ;ship center
        STA NEWCOL   ;destination X
        LDA YPLR+1   ;enemy Y coord
MIS2    STA NEWROW   ;destination Y
        INC NEWROW   ;ship top
        LDA #1       ;init value for
        STA ROWINC,X ;Y increment
        STA COLINC,X ;X increment
        SEC          ;set carry
        LDA NEWROW   ;Y to coord
        SBC OLDROW   ;Y from coord
        STA DELTAR,X ;delta Y
        BCS MIS3     ;shot down? Yes.
        LDA #$FF     ;value = -1
        STA ROWINC,X ;move up
        EOR DELTAR,X ;get absolute
        STA DELTAR,X ;value for
        INC DELTAR,X ;delta Y
MIS3    SEC          ;set carry
        LDA NEWCOL   ;X to coord
        SBC OLDCOL   ;X from coord
        STA DELTAC,X ;delta X
        BCS MIS4     ;to right? Yes.
        LDA #$FF     ;value = -1
        STA COLINC,X ;move left
        EOR DELTAC,X ;get absolute
        STA DELTAC,X ;value for
        INC DELTAC,X ;delta X
MIS4    LDA #0       ;init value for
        STA COLAC,X  ;X accumulator
        STA ROWAC,X  ;Y accumulator
        LDA DELTAC,X ;get delta X
        STA ENDPT,X  ;line length
        LDA #$0F     ;init value for
        STA COUNTR,X ;draw iteration
        LDA DELTAR,X ;get delta Y
        CMP ENDPT,X  ;bigger than
        BCC MIS5     ;delta X? No.
        STA ENDPT,X  ;store new value
        LSR A        ;delta Y / 2
        STA COLAC,X  ;X coord Acc
        RTS          ;return
;
MIS5    LDA ENDPT,X  ;get delta X
        LSR A        ;divide by 2
        STA ROWAC,X  ;Y coord Acc
        RTS          ;return
;
; Game Interrupt Service Routine
; ------------------------------
;
GISR    PHA          ;save Acc
        JSR SMASH    ;test collisions
        JSR ROTOR    ;animate ships
        INC CLOCK    ;increment clock
        CLC          ;clear carry
        LDA UNIVS    ;universe sound
        ADC #$30     ;change sound
        STA UNIVS    ;replace sound
        STA AUDF2    ;make sound
        LDA ENDGAM   ;game end flag
        BNE GI7      ;game over? No.
        JMP GI99     ;end it now!
;
; Ship Noise Generation
; ---------------------
;
GI7     LDA #8       ;medium grey
        STA COLPF1   ;set color
        TXA          ;move X to Acc
        PHA          ;save X
        LDX #15      ;neutral stick
        LDA DEAD     ;player 0 status
        BNE GI70     ;dead? No.
        STX STICK    ;stick to center
GI70    LDA DEAD+1   ;player 1 status
        BNE GI71     ;dead? No.
        STX STICK+1  ;stick to center
GI71    LDA STICK    ;check if either
        AND STICK+1  ;stick pushed by
        CMP #15      ;comparing w/15
        LDA #$81     ;rumble noise
        BCS GI03     ;both 15? Yes.
        LDA TRIGN    ;trigger opt 0
        ORA TRIGN+1  ;trigger opt 1
        CMP #3       ;warp speed test
        LDA #$88     ;loud warp sound
        BCS GI03     ;warp? Yes.
        LDA #$84     ;med engine roar
GI03    STA AUDC4    ;make engine snd
        LDA #24      ;warp frequency
        BCS GI30     ;warp? Yes.
        LDA #32      ;normal rockets
GI30    STA AUDF4    ;set frequency
        LDX #8       ;init value for
        STX VDEL     ;vertical delay
        INX          ;make X = 1
        LDA M0PL     ;missile 0 to
        STA M0PL8    ;PL collisions
        LDA M0PL+1   ;missile 1 to
        STA M0PL8+1  ;PL collisions
        LDA TRIGN    ;trigger opt 0
        ORA TRIGN+1  ;trigger opt 1
        CMP #1       ;check shields
        BNE MOVE2    ;shields on? No.
        LDA RANDOM   ;random number
        AND #$1F     ;only 0..31
        ORA #$20     ;only 32..63
        STA AUDF4    ;shield sound
        LDA #$A4     ;value for
        STA AUDC4    ;shield volume
MOVE2   JSR TRIGR    ;read trigger
        LDA DEAD,X   ;player status
        BEQ MOVEX    ;dead? Yes.
        LDA TRIGN,X  ;trigger value
        CMP #1       ;shields in use?
        BEQ MOVEX    ;Yes. skip next
        JSR MISFLY   ;move missile
        LDA CLOCK    ;get clock value
        ROR A        ;bit 0 to carry
        BCC MOVE2X   ;even? Yes.
        LDA TRIGN,X  ;trigger value
        CMP #3       ;warp drive?
        BNE MOVEX    ;No. skip next
;
; Joystick Handler
; ----------------
;
MOVE2X  LSR STICK,X  ;check joystick
        BCS MOVED    ;stick up? No.
        DEC YPLR,X   ;move ship up
MOVED   LSR STICK,X  ;check joystick
        BCS MOVEL    ;stick down? No.
        INC YPLR,X   ;move ship down
MOVEL   LDA SHIP     ;get ship type
        BEQ MOVEX    ;U/D rocket? Yes.
        LSR STICK,X  ;check joystick
        BCS MOVER    ;stick left? No.
        DEC XPLR,X   ;move ship left
MOVER   LSR STICK,X  ;check joystick
        BCS MOVEX    ;stick right? No.
        INC XPLR,X   ;move ship right
MOVEX   LDA XPLR,X   ;ship X coord
        CMP #$30     ;past left
        BCS MOVEX3   ;border? No.
        LDA #$30     ;set to border
MOVEX3  CMP #$C0     ;past right
        BCC MOVEX4   ;border? No.
        LDA #$C7     ;set to border
MOVEX4  STA XPLR,X   ;save X coord
        STA HPOSP0,X ;position ship
        STY TEMP+3   ;save Y
        LDA DEAD,X   ;ship status
        BMI MOVX4    ;alive? Yes.
        LDA CLOCK    ;get clock value
        ROR A        ;test bit 0
        BCS MOV4     ;clock odd? Yes.
        ASL A        ;restore clock
        AND #$F6     ;max 7 brightness
        STA PCOLR0,X ;make ship color
        LDA SCLOCK   ;score snd count
        BNE MOV4     ;sound on? Yes.
        LDA CLOCK    ;get clock value
        AND #$07     ;make 0..7
        ORA #$C8     ;make $C8..$CF
        STA AUDC3    ;sound volume
        LDA RANDOM   ;random number
        AND #$1F     ;make 0..31
        ORA #$20     ;make 32..63
        STA AUDF3    ;weird sound
MOV4    INC YPLR,X   ;move ship down
MOVX4   LDA YPLR,X   ;ship Y coord
        BNE MOVEX6   ;screen top? No.
        JSR SCORE    ;increment score
MOVEX6  CMP #192     ;screen bottom?
        BCC MOVEX5   ;No. skip next
        LDA DEAD,X   ;ship status
        BNE MOVX6    ;ship dead? No.
        LDA XPLR,X   ;ship X coord
        CMP XSTRT,X  ;compare w/start
        BEQ MOVX6A   ;equal? Yes.
        INC XPLR,X   ;move ship right
        BCC MOVX6    ;ok? Yes. else
        DEC XPLR,X   ;move ship left
        DEC XPLR,X   ;move ship left
        BNE MOVX6    ;skip next
MOVX6A  LDA #$FF     ;value for
        STA DEAD,X   ;working ship
        STA HITCLR   ;clr collisions
        LDA PCOLRS,X ;value for
        STA PCOLR0,X ;ship color
        LDA SCLOCK   ;score snd clock
        BNE MOVX6    ;sound on? Yes.
        STA AUDC3    ;zero sound
MOVX6   LDA #191     ;value for
MOVEX5  STA YPLR,X   ;init Y coord
        CPX #1       ;ship 1?
        ROR A        ;carry=1 if Yes.
        STA GRPX     ;PM graphic pntr
        BCC MOVEX7   ;coord even? Yes.
        LDA VDEL     ;vertical delay
        ORA VDMSK,X  ;delay mask
        STA VDEL     ;new V.delay
MOVEX7  TXA          ;move X to Acc
        PHA          ;save X
        CLC          ;clear carry
        LDA DEAD,X   ;ship status
        BEQ MOVX7A   ;ship dead? Yes.
        LDA RANDOM   ;random number
        AND #$0F     ;make 0..15
        ORA PCOLRS,X ;add ship color
        STA PCOLR0,X ;new ship color
        LDA TRIGN,X  ;check trigger
        CMP #1       ;for shields
        SEC          ;set carry
        BEQ MOVX7A   ;shields? Yes.
        CLC          ;clear carry
        LDA PCOLRS,X ;ship color
        STA PCOLR0,X ;set color
MOVX7A  LDY #9       ;do 10 bytes
        LDX #9       ;rocket offset
        LDA SHIP     ;get ship type
        BEQ MOVEXD   ;rocket? Yes.
        LDX #18      ;saucer offset
MOVEXD  LDA SHIPS,X  ;ship graphic
        BCC MVXD     ;shields on? No.
        AND RANDOM   ;shield effect
MVXD    STA (GRPX),Y ;put ship
        DEX          ;graphic index
        DEY          ;PM index
        BPL MOVEXD   ;done? No.
        PLA          ;pull X
        TAX          ;move Acc to X
        LDA MISSLE,X ;missile status
        BEQ MVXDM    ;active? No.
        LDY #1       ;missile launcher
        LDA #0       ;no shot graphic
        STA (GRPX),Y ;put ship
MVXDM   LDY TEMP+3   ;restore Y index
        DEX          ;next ship
        BMI MOVEX0   ;done? Yes.
        JMP MOVE2    ;No. continue
MOVEX0  LDA VDEL     ;V.delay shadow
        STA VDELAY   ;vertical delay
        LDA ENDGAM   ;game status
        BEQ MOVXE2   ;game over? Yes.
        JSR TIMER    ;decrement time
        BPL GI9      ;time up? No.
        LDA #N7C+$C0 ;zero graphic
        STA TRAM+7   ;minutes display
        STA TRAM+9   ;10 sec display
        STA TRAM+10  ;seconds display
        STA TRAM+12  ;.1 sec display
MOVXE2  LDA #121     ;goto attract
        STA ATRACT   ;in 30 seconds
        LDA #0       ;get zero
        STA ENDGAM   ;end game
        STA AUDC3    ;sound off
        STA AUDC4    ;sound off
;
; Comet Mover Routine
; -------------------
;
GI9     PLA          ;pull X
        TAX          ;move Acc to X
GI99    LDA HINC     ;comet H speed
        AND CLOCK    ;mask with clock
        BEQ GI91     ;move comet? Yes.
GI90    PLA          ;restore Acc
        RTI          ;return
GI91    LDA COMETF   ;comet status
        BEQ GI90     ;active? No.
        TYA          ;move Y to Acc
        PHA          ;save Y
        TXA          ;move X to Acc
        PHA          ;save X
        LDA HPOS     ;comet H coord
        AND VINC     ;and w/V speed
        BNE GI5      ;move vert? No.
        LDA VPOS     ;comet V coord
        CLC          ;clear carry
        ADC VDIR     ;add V direction
        AND #$7F     ;make 0..127 only
        STA VPOS     ;replace V coord
;
; Draw Comet Graphics
; -------------------
;
GI5     LDY #13      ;do 14 bytes
        LDX VPOS     ;comet V coord
GI0     LDA RANDOM   ;random number
        AND COMETM,Y ;mask w/comet
        STA PM2,X    ;put player 2
        LDA RANDOM   ;random number
        AND COMETM,Y ;do same thing
        STA PM3,X    ;put player 3
        INX          ;inc player addr
        TXA          ;move X to Acc
        AND #$7F     ;make 0..127
        TAX          ;replace X
        DEY          ;dec byte count
        BPL GI0      ;pic done? No.
        LDA RANDOM   ;random number
        AND #$F0     ;color only
        ORA #$04     ;brightness 4
        STA PCOLR2   ;player 2 color
        LDA RANDOM   ;random number
        AND #$F0     ;color only
```

```
        ORA #$08    ;brightness 8
        STA PCOLR3  ;player 3 color
        CLC         ;clear carry
        LDA HPOS    ;comet H coord
        ADC HDIR    ;add H direction
        STA HPOS    ;new H coord
        STA HPOSP2  ;player 2 H pos
        STA HPOSP3  ;player 3 H pos
        STA COMETF  ;non 0 = enabled
;
; Generate Comet Sound
; --------------------
;
        INC C$OUND  ;comet snd freq
        LDA ENDGAM  ;game status
        BEQ GI8     ;game over? Yes.
        LDA C$OUND  ;comet snd freq
        CMP #125    ;value < 125
        BCS GI2     ;No. skip next
        LSR A       ;freq/2
        LSR A       ;freq/4
        STA AUDF1   ;set frequency
        CMP #$18    ;past peak?
        BCC GI2     ;No. skip next
        EOR #$9F    ;reverse volume
GI8     STA AUDC1   ;set volume
GI2     PLA         ;pull X
        TAX         ;move Acc to X
        PLA         ;pull Y
        TAY         ;move Acc to Y
        PLA         ;pull Acc
        RTI         ;return
;
; Joystick Trigger Processor
; --------------------------
;
TRIGR   LDA STRIG,X ;get trigger
        LSR A       ;move to carry
        BCS TRIGX   ;pressed? No.
        LDA TRIG    ;trigger option
TRIGX   AND DEAD,X  ;ship status
        CMP TRIGN,X ;same as last?
        BEQ TRIG2   ;Yes. skip next
        STA HITCLR  ;clr collisions
TRIG2   STA TRIGN,X ;trig option
;
; Missile Mover Routine
; ---------------------
;
        LDA MISSLE,X ;missile status
        BEQ MMX     ;fired? No.
        LDA ROWCRS,X ;missile Y coord
        BNE MM0     ;zero? No.
        STA MISSLE,X ;disable missile
MMX     RTS         ;return
;
MM0     CLC         ;clear carry
        LDA DELTAR,X ;add delta Y
        ADC ROWAC,X ;to Y Acc
        STA ROWAC,X ;new Y Acc
        CMP ENDPT,X ;cmp w/endpoint
        BCC MM1     ;change Y? No.
        SBC ENDPT,X ;sub endpoint
        STA ROWAC,X ;new Y Acc value
        CLC         ;clear carry
        LDA ROWINC,X ;add shot Y inc
        ADC ROWCRS,X ;to Y coord
        STA ROWCRS,X ;new Y coord
MM1     CLC         ;clear carry
        LDA DELTAC,X ;add delta X
        ADC COLAC,X ;to X Acc
        STA COLAC,X ;new X Acc
        CMP ENDPT,X ;cmp w/endpoint
        BCC MMP     ;change X? No.
        SBC ENDPT,X ;sub endpoint
        STA COLAC,X ;new X Acc value
        CLC         ;clear carry
        LDA COLINC,X ;add shot X inc
        ADC COLCRS,X ;to X coord
        STA COLCRS,X ;new X coord
MMP     LDA COUNTR,X ;iteration cnt
        BEQ MMP1    ;cnt done? Yes.
        LSR A       ;cnt div 2
        ORA #$88    ;weird sound
        STA AUDC3   ;make sound FX
        LDA COUNTR,X ;get cnt again
        EOR #$0F    ;very weird FX
        ASL A       ;times 2
        ASL A       ;times 4
        ASL A       ;times 8
        STA AUDF3   ;set frequency
        DEC COUNTR,X ;decrement cnt
MMP1    LDA COLCRS,X ;shot X coord
        STA HPOSM0,X ;PM horiz pos
        LDA ROWCRS,X ;shot Y coord
        LSR A       ;2 line res
        BCC MM00    ;even line? Yes.
        PHA         ;save PM Y coord
        LDA VDEL    ;VDELAY shadow
        ORA VDMSK+2,X ;add scan line
        STA VDEL    ;new shadow val
        PLA         ;restore Y coord
MM00    ORA # <PMM  ;missiles start
        STA GRPM    ;set pointer lo
        LDA M0PL8,X ;test if shot hit
        AND MCMSK,X ;comet or other
        BNE MME     ;ship? Yes.
        LDA COLCRS,X ;shot Y coord
        BEQ MME     ;off scrn? Yes.
        TYA         ;move Y to Acc
        PHA         ;save Y
        LDY #3      ;4 pic bytes
MM01    LDA (GRPM),Y ;current pic
        AND MISMSK,X ;erase old pic
        STA TEMPM   ;save temp
        LDA MIMAGE,Y ;new pic data
        CPX #1      ;player 1 shot?
        BNE MM03    ;No. skip next
        ASL A       ;shift byte two
        ASL A       ;bits to left
MM03    ORA TEMPM   ;add saved data
        STA (GRPM),Y ;new shot pic
        DEY         ;dec pic index
        BPL MM01    ;pic done? No.
        PLA         ;restore Y
        TAY         ;move Acc to Y
        RTS         ;return
;
; Erase Current Shot
; ------------------
;
MME     LDA #0      ;get zero value
        STA MISSLE,X ;kill shot
        TYA         ;move Y to Acc
        PHA         ;save Y
        LDY #3      ;do 4 bytes
MME2    LDA (GRPM),Y ;old shot pic
        AND MISMSK,X ;erase shot
        STA (GRPM),Y ;replace pic
        DEY         ;next byte
        BPL MME2    ;done? No.
        LDA M0PL8,X ;cmp collisions
        AND MCMSK,X ;w/all but own
        BEQ MME3    ;score point? No.
        JSR SCOREM  ;inc score
MME3    PLA         ;pull Y
        TAY         ;move Acc to Y
        RTS         ;return
;
; Initialize Playfield
; --------------------
;
PLAYGM  JSR AUDOFF  ;turn snd off
        STA HITCLR  ;clr collisions
        LDA #$FF    ;value for
        STA DEAD    ;ship 1 alive
        STA DEAD+1  ;ship 2 alive
        LDA #$40    ;value to
        STA NMIEN   ;enable VBI's
        LDA #PAGES*2 ;pages to zero
        STA TEMP+2  ;save in counter
        LDA # <DRAM ;display addr lo
        STA GRPAGE  ;display pntr lo
        LDA # >DRAM ;display addr hi
        STA GRPAGE+1 ;displa pntr hi
        LDA # >PMM  ;PM addr hi
        STA GRPM+1  ;PM pointer hi
        LDA #$21    ;value for
        STA GPRIOR  ;mulit-color PL
        STA AUDF4   ;engine sound
        LDA #1      ;value for
        STA SIZEP2  ;double width
        STA SIZEP3  ;comets enabled
        STA ENDGAM  ;game on
        LDA #99     ;max score value
        STA SCORE8  ;player 1
        STA SCORE8+1 ;player 2
        LDX #24     ;move 25 bytes
PLA3    LDA TRAMI,X ;from TRAMI
        STA TRAM,X  ;to TRAM
        DEX         ;dec index
        BPL PLA3    ;done? No.
        LDA # >PM0  ;player addr hi
        STA GRPX+1  ;pntr hi byte
        JSR SCRCLR  ;clear inverse
        LDA #$05    ;value for
        STA SIZEM   ;missile sizes
        LDA #0      ;get zero
        STA HPOS    ;comet X coord
        STA FLASHF  ;inverse flag
        STA COMETF  ;comet flag
        STA ATRACT  ;attract mode
        STA SCLOCK  ;score sound
        STA MISSLE  ;missile 0 flag
        STA MISSLE+1 ;missile 1 flag
        TAY         ;zero Y index
PLA0    STA (GRPAGE),Y ;scrn byte=0
        INY         ;inc Y index
        BNE PLA0    ;page done? No.
        INC GRPAGE+1 ;do next page
        DEC TEMP+2  ;page count
        BNE PLA0    ;all done? No.
        LDA #191    ;init Y coord of
        STA YPLR    ;ship 1
        STA YPLR+1  ;ship 2
        TYA         ;get zero
PLA1    STA PMM,Y   ;zero missiles
        STA PM0,Y   ;player 0+1
        STA PM2,Y   ;player 2+3
        INY         ;next byte
        BNE PLA1    ;page done? No.
        JSR RANFIL  ;put asteroids
        LDX #1      ;do 2 players
PLA11   LDA XSTRT,X ;ship X start
        STA HPOSP0,X ;set hardware
        STA XPLR,X  ;and shadow
        DEX         ;next ship
        BPL PLA11   ;done? No.
        INX         ;make it a one
        STX COLOR2  ;black
        STX TEMP+1  ;zero temporary
        LDA #$2E    ;value for
        STA SDMCTL  ;P/M enable
        STA COLOR1  ;bright orange
        LDA # >PM   ;set up P/M
        STA PMBASE  ;base address
        LDA #3      ;value to
        STA GRACTL  ;enable P/M
        LDA #$C4    ;green for
        STA PCOLR0  ;first ship
        LDA #$34    ;red/orange for
        STA PCOLR1  ;second ship
        LDA # <GISR ;DLI addr lo
        STA VDSLST  ;DLI pntr lo
        LDA # >GISR ;DLI addr hi
        STA VDSLST+1 ;DLI pntr hi
        LDA # <DISP0 ;DL addr lo
        STA SDLSTL  ;DL pntr lo
        LDA # >DISP0 ;DL addr hi
        STA SDLSTL+1 ;DL pntr hi
        LDA #$C0    ;value to
        STA NMIEN   ;enable DLI's
;
; Game Loops Here, Loops Here...
; ------------------------------
;
ASHIFT  JSR LD0RRT  ;set up pointers
        ROR TEMP+1  ;get saved carry
        BCC ART0    ;bit #1=0? Yes.
        LDA (GRPAGE),Y ;get byte
        ORA #$80    ;set high bit
        CLC         ;clear carry
        BCC ART2    ;skip next
ART0    LDA (GRPAGE),Y ;get byte
ART2    ROR A       ;rotate right
        STA (GRPAGE),Y ;replace byte
        INY         ;next byte
        BNE ART0    ;page done? No.
        INC GRPAGE+1 ;inc page addr
        DEC TEMP+2  ;dec page count
        BNE ART0    ;pages done? No.
        LDA #PAGES  ;pages to rotate
        STA TEMP+2  ;put someplace
```

```
                LDA # <ABTL     ;area to shift
                STA GRPAGE      ;gr pntr lo
                LDA #PAGES-1+ >ABTL ;hi addr
                STA GRPAGE+1    ;gr pntr hi
ALF1            DEY             ;dec Y index
                LDA (GRPAGE),Y  ;get graphic
                ROL A           ;move to left
                STA (GRPAGE),Y  ;replace byte
                TYA             ;test Y index
                BNE ALF1        ;Zero? No.
                DEC GRPAGE+1    ;dec hi addr
                DEC TEMP+2      ;dec page count
                BNE ALF1        ;pages done? No.
                ROL TEMP+1      ;save carry bit
                STA GRP1        ;zero low bytes
                STA GRP2        ;of these three
                STA GRPAGE      ;page zero pntrs
                LDA # >ABTL     ;scrn lf addr hi
                STA GRP1+1      ;scrn fl pntr hi
                LDA # >ABTR     ;scrn rt addr hi
                STA GRP2+1      ;scrn rt pntr hi
                LDA # >DRAM     ;display addr hi
                STA GRPAGE+1    ;displa pntr hi
                LDA # <DRAM-20  ;w/offset lo
                STA GRP20P      ;offset pntr lo
                LDA # >DRAM-20  ;w/offset hi
                STA GRP20P+1    ;offset pntr hi
                LDX #76         ;76 scan lines
ORA0            LDY #19         ;do 20 bytes
ORA1            LDA (GRP1),Y    ;to left field
                ORA (GRP2),Y    ;to right field
                STA (GRPAGE),Y  ;lf half scrn
                STA (GRP20P),Y  ;rt half scrn
                DEY             ;dec index
                BPL ORA1        ;line done? No.
                LDA GRP1        ;lf field lo
                CLC             ;clear carry
                ADC #20         ;20 byte offset
                STA GRP1        ;do next line of
                STA GRP2        ;asteroid fields
                BCC ORA2        ;overflow? No.
                INC GRP1+1      ;inc left, right
                INC GRP2+1      ;asteroid pntrs
ORA2            CLC             ;clear carry
                LDA GRPAGE      ;graphic addr lo
                ADC #40         ;line offset
                STA GRPAGE      ;new lo addr
                BCC ORA3        ;overflow? No.
                INC GRPAGE+1    ;inc hi addr
ORA3            CLC             ;clear carry
                LDA GRP20P      ;gr addr2 lo
                ADC #40         ;line offset
                STA GRP20P      ;new lo addr2
                BCC ORA4        ;overflow? No.
                INC GRP20P+1    ;inc high addr2
ORA4            DEX             ;decrement index
                BNE ORA0        ;done? No.
                JSR CONC        ;check console
                BCC FLASH       ;OPTION key? No.
                JMP STOPTS      ;run option menu
;
; Inverse Universe Handler
; ------------------------
;
FLASH       ROR A           ;START -> carry
            BCC FL0         ;START pressed?
            JMP PLAYGM      ;Yes. restart

FL0         LDA UNIV        ;univ flag
            BEQ COMET       ;inverse? No.
            LDA FLASHF      ;flash flag
            BEQ TRYFLA      ;flash on? No.
            JSR REVSCR      ;flash screen
            DEC FLASHC      ;decrement count
            LDA ENDGAM      ;game over flag
            BEQ FL1         ;game over? Yes.
            LDA FLASHC      ;flash count
            LSR A           ;make volume
            ORA #$A0        ;make pure tone
FL1         STA AUDC2       ;universe sound
            LDA FLASHC      ;flash count
            BEQ FLAEND      ;done? Yes.
            ASL A           ;times 2
            ASL A           ;times 4
            ASL A           ;times 8
            EOR #$FF        ;flip bits
            STA UNIVB       ;save univ sound
            BNE COMET       ;continue
;
SCRCLR      LDA REVF        ;reverse flag
            ROR A           ;check bit 0
            BCC SC9         ;zero? Yes.
REVSCR      LDX #4          ;do colors 0..4
REV0        LDA $$0F        ;flip brightness
            EOR COLOR0,X    ;of colors
            STA COLOR0,X    ;replace values
            DEX             ;decrement index
            BPL REV0        ;done? No.
            DEC REVF        ;clear flag
SC9         RTS             ;return
;
TRYFLA      LDA RANDOM      ;random number
            AND #$3F        ;make 0..63
            BNE COMET       ;do flash? No.
            JSR REVSCR      ;Yes. flash scrn
            LDA #$1F        ;value for
            STA FLASHC      ;flash count
            STA FLASHF      ;flash flag
            BNE COMET       ;continue
;
FLAEND      STA FLASHF      ;flash univ off
            STA AUDC2       ;flash sound off
;
; Comet Mover
; -----------
;
COMET       LDA COME        ;comet flag
            BEQ COMETX      ;comets? No.
            LDA COMETF      ;comet on flag
            BEQ TRYCOM      ;on? No.
COMETX      JMP ASHIFT      ;continue
;
TRYCOM      LDA RANDOM      ;random number
            AND #$3F        ;make 0..63
            BNE COMETX      ;enable? No.
            STA HINC        ;comet H speed=0
            TAX             ;initialize X=0
TRY0        STA PM2,X       ;clr players 2+3
            DEX             ;decrement index
            BNE TRY0        ;done? No.
            STA HPOSP2      ;zero players 23
            STA HPOSP3      ;horizontal pos
            STA CSOUND      ;zero comet snd
TRY5        LDA RANDOM      ;random number
            AND #3          ;make 0..3
            CMP #3          ;allow only 0..2
            BCS TRY5        ;3? Yes, repeat
            STA VDIR        ;save 0..2
            DEC VDIR        ;make -1..1
            LDA RANDOM      ;random number
            AND #3          ;make 0..3
            TAX             ;use as index
            LDA #$1F        ;init increment
TRY6        LSR A           ;divide by 2
            DEX             ;decrement index
            BPL TRY6        ;done? No.
            STA VINC        ;vert increment
            LDA ENDGAM      ;end game flag
            BEQ TRY7        ;game over? Yes.
            LDA #$88        ;No. make sound
TRY7        STA AUDC1       ;comet sound
            LDA RANDOM      ;random number
            AND #$3F        ;make 0..63
            ADC #16         ;sorta 16..79
            STA VPOS        ;vert start pos
            LDX #$FE        ;value -2
            STX COMETF      ;comet on flag
            LDA RANDOM      ;random number
            BPL TRY1        ;flip a coin!
            LDX #$02        ;value +2
TRY1        STX HDIR        ;horiz direction
            LDA RANDOM      ;random number
            BPL COMETX      ;another coin!
            INC HINC        ;change speed
            JMP ASHIFT      ;continue
;
; Relocate program
; ----------------
;
RELOC       LDA #$20        ;$2000 load addr
            STA GRP1+1      ;from pntr hi
            LDA #$10        ;to $1000 addr
            STA GRP2+1      ;to pntr hi
            TAX             ;move 16 pages
            LDA #0          ;value for
            STA GRP1        ;from pntr lo
            STA GRP2        ;to pntr lo
            TAY             ;zero index
REL0        LDA (GRP1),Y    ;from byte
            STA (GRP2),Y    ;to byte
            DEY             ;dec index
            BNE REL0        ;page done? No.
            INC GRP1+1      ;inc from hi
            INC GRP2+1      ;inc to hi
            DEX             ;page count
            BNE REL0        ;done? No.
            JMP RIS         ;run program
;
            *=  $02E0       ;RUN address
            .WORD RELOC+$1000
;
            .END
```

Some program listings reproduced in **ANALOG** may contain "strange" characters not shown on the Atari keyboard. These are special characters which use the CTRL, ESC and "ATARI LOGO" (INVERSE) keys. Shown below is a list of these characters and the keystrokes used to get them. □

♥ --- CTRL ,
├ --- CTRL A
| --- CTRL B
┘ --- CTRL C
┤ --- CTRL D
┐ --- CTRL E
╱ --- CTRL F
╲ --- CTRL G
◢ --- CTRL H
▝ --- CTRL I
◣ --- CTRL J
▘ --- CTRL K
▖ --- CTRL L
▔ --- CTRL M
▁ --- CTRL N
▗ --- CTRL O
┼ --- CTRL P
┌ --- CTRL Q
─ --- CTRL R
┼ --- CTRL S
● --- CTRL T
▄ --- CTRL U
▏ --- CTRL V
┬ --- CTRL W
┴ --- CTRL X
▌ --- CTRL Y
└ --- CTRL Z
ESC ESC
↑ --- ESC CTRL UP-ARROW
↓ --- ESC CTRL DOWN-ARROW
← --- ESC CTRL LEFT-ARROW
→ --- ESC CTRL RIGHT-ARROW
♦ --- CTRL .
♠ --- CTRL ;
↰ --- ESC SHIFT CLEAR
↰ --- ESC BACK S
↦ --- ESC TAB
▒ --- INVERSE CTRL ,
▙ --- INVERSE CTRL A
▉ --- INVERSE CTRL B
▟ --- INVERSE CTRL C
▙ --- INVERSE CTRL D
▛ --- INVERSE CTRL E
▚ --- INVERSE CTRL F
▜ --- INVERSE CTRL G
▛ --- INVERSE CTRL H
▛ --- INVERSE CTRL I
▜ --- INVERSE CTRL J
▙ --- INVERSE CTRL K
▟ --- INVERSE CTRL L
■ --- INVERSE CTRL M
■ --- INVERSE CTRL N
▜ --- INVERSE CTRL O
▛ --- INVERSE CTRL P
▛ --- INVERSE CTRL Q
▀ --- INVERSE CTRL R
▀ --- INVERSE CTRL S
▒ --- INVERSE CTRL T
▀ --- INVERSE CTRL U
▌ --- INVERSE CTRL V
▀ --- INVERSE CTRL W
▀ --- INVERSE CTRL X
▌ --- INVERSE CTRL Y
▙ --- INVERSE CTRL Z
▒ --- ESC DELETE
▒ --- ESC INSERT
▒ --- ESC CTRL TAB (CLR)
▒ --- ESC SHIFT TAB (SET)
▉ --- INVERSE SPACE
▁ --- INVERSE _
▒ --- INVERSE CTRL .
▒ --- INVERSE CTRL ;
▌ --- INVERSE |
▒ --- ESC CTRL 2
▒ --- ESC CTRL BACK S
▒ --- ESC CTRL INSERT

ARCHON II: ADEPT
by Anne Westfall, Jon Freeman and Paul Reiche, III
ELECTRONIC ARTS
2755 Campus Drive
San Mateo, CA 94403
48K Disk $39.95

by Patrick J. Kelley

When the original **Archon** first made its debut in early 1983, I was convinced that I had found the ideal computer game. **Archon** combined chess strategy with hard-driving action. It offered an exciting challenge to jaded video-jocks who wanted a little thought with their shoot-'em-ups, and vice versa.

Only a pile of battered joysticks survive, mute witnesses to my devotion to **Archon**, along with a few fond memories of pitched battles shared with fellow devotee Tom Hudson. My feelings for this game run deep, so it was with mixed emotions that I received the news that Electronic Arts was considering a "sequel" of sorts, **Archon II: Adept**. I awaited the arrival of the game here at **ANALOG Computing**, joystick in hand and breath bated.

The age of Archon.

In **Archon II**, the lines of battle are drawn: the minions of Chaos face off against the forces of Order for the control of the material world. The battle takes place in each of the four elements—air, earth, fire and water. The principle combatants are mages (or sorcerers of each Demesne or Dimension) and their Demons or Elementals, summoned by each to slug it out. Only when all six power points on the board are secured—or your opponent runs out of energy—is the game over.

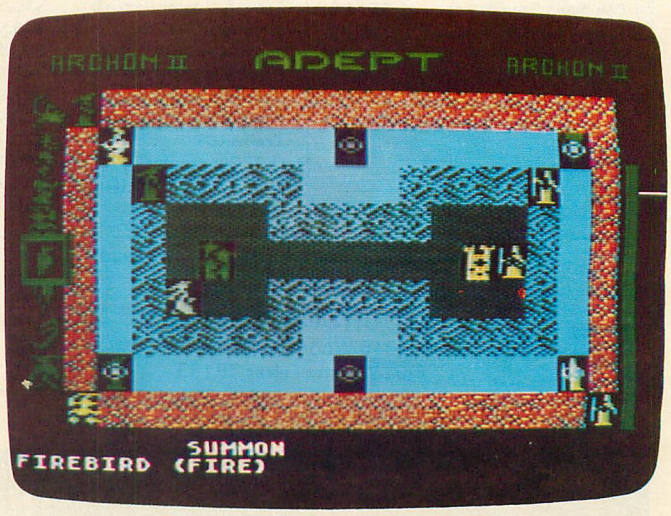
You also have a trump card: the doomsday spell of Apocalypse, where a single sorcerer meets his opposite number in a fireball-throwing fight to the finish. The real emphasis of **Archon II** is power, magical power. And herein lies this new version's major weakness.

Power and prey.

In **Archon II**, your primary obstacle to overcome when doing combat is your expenditure of magical energy. At the beginning of each game, both sides have a limit of energy to expend in combat, be it in spells to hobble your enemy or in power used to summon elemental "troops" into combat to remove an opponent.

Power can be maintained only by careful budgeting or successful occupation of your dimensional power points. Thus, you're not battling individual pieces as you were in the original **Archon**, where victory depended on your skills in attrition against your opponent's pieces. Instead, you are battling "power," a concept I feel falls flat when compared to the original. Surgically eliminating pieces cannot give you the same gain as in the earlier version, let alone the satisfaction of splattering an enemy to the four winds (or, in this case, elements).

In this respect, **Archon II** is more strategy oriented . . . and slower. Attrition here can be brought about by forcing your opponent to deplete his pool of power, replacing pieces or casting spells. Your advantage is to grab power points as soon as possible, using as little energy as possible to dig in. Pitched battles are still possible using this strategy, but usually they don't come until later in the game.



Archon II: Adept.

However, if things are going badly for you in the strategy phase of the game, there is still the Apocalypse spell waiting, either to save or damn you. In calling for Apocalypse, you and your opposite decide the outcome with well-placed shots. The two solitary pieces square off on a symbolically fiery battleground and have it out. The object of the Apocalypse is to hit the other guy hard, fast and continuously—until he ceases to be, buys the farm, checks out, goes belly up, etc. In other words, if you can outgun your adversary and withstand his assaults on you, you emerge as **Archon** Master, ruler of all you survey.

All's fair?

In discussing the Apocalypse segment of **Archon II: Adept**, I am bound by my duty as fair and impartial reviewer to comment on a built-in "cheat" that favors the computer when you take up arms against it. In **Archon II**, not only is the computer a more ruthless player than in **Archon**, but it has gained the ability to steer its shots against you when it chooses an Adept as a playing piece. Although you, too, have this ability, virtually no amount of practice will give you the lightning-fast response time that the computer has. This, I feel, is stacking the deck *too* much in favor of the machine-enemy.

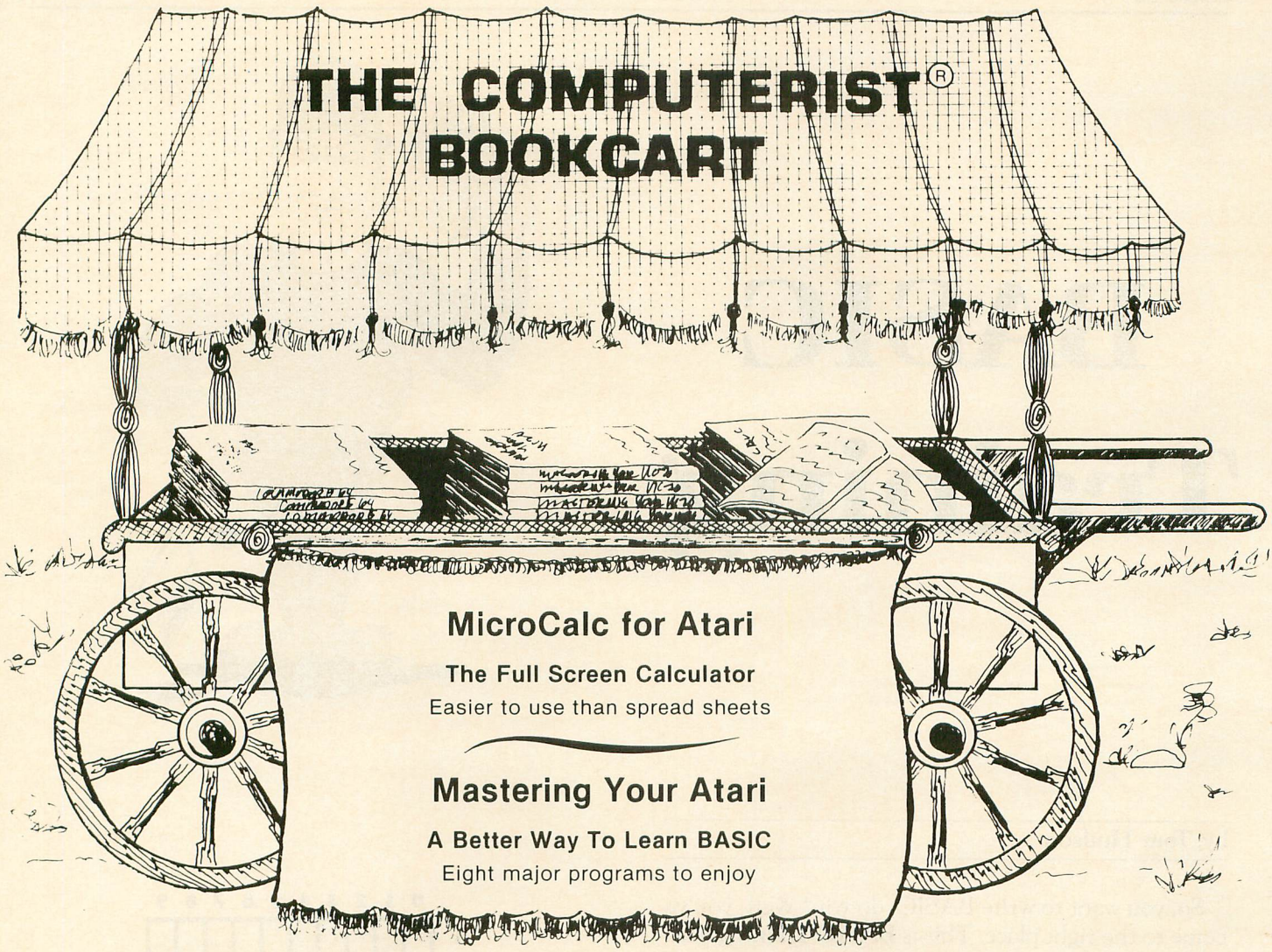
Let's face it, the computer can't *help* but be a better shot than you are. Knowing where the adversary is going to be next is a great advantage, and that lets the computer put one right smack dab into your man every time. No matter how you dodge, parry, run, hide or scream, the outcome will be the same: defeat for you; dancing Adept; START button depressed.

There is one pointer that I *can* give you. The computer's homing missiles or fireballs aren't functional on a battlefield where there is natural cover or a place to hide. Use this to your advantage to snipe at the computer, or as shelter from its deadly shots.

The ending is yours.

In the final accounting, **Archon II: Adept** is a fine game. With the exception of the two features that I found personally annoying, I cannot fault the game. Conceptually, it is still a winner, and the playability is superb. **Archon II** has great entertainment value, with the added plus of giving an old classic a new twist.

I think that, in time to come, players are bound to develop new strategies, fresh tactics and more efficient ways to give the computer its just desserts. But, in the meantime, I foresee many strained friendships, long marathon sessions and many a case of "joystick thumb." Electronic Arts, you've done it again. □



Mastering Your Atari

The 8 programs, "run-ready" on disk and explained in the 192 page book, each demonstrate important concepts of BASIC while providing useful, enjoyable software. Programs include:

- Player — compose songs from your keyboard, save, load and edit for perfect music
- MicroCalc — display calculation program that make even complex operations easy
- Master — a one or two person guessing game
- Clock — character graphics for a digital clock

Atari with disk & book ..... **just $19.95**

MicroCalc for Atari

This on-screen calculator comes with diskette and 48-page manual offering a wide variety of useful screens, and a great way to learn BASIC expressions if you don't already know them.

- Unlimited calculation length & complexity
- Screens can be linked and saved on disk/cassette
- Build a library of customized screens
- Provide formatted printer output

Diskette & 48-page manual ..... **just $29.95**

Look for us at the
International Software Show
Toronto, September 20-23

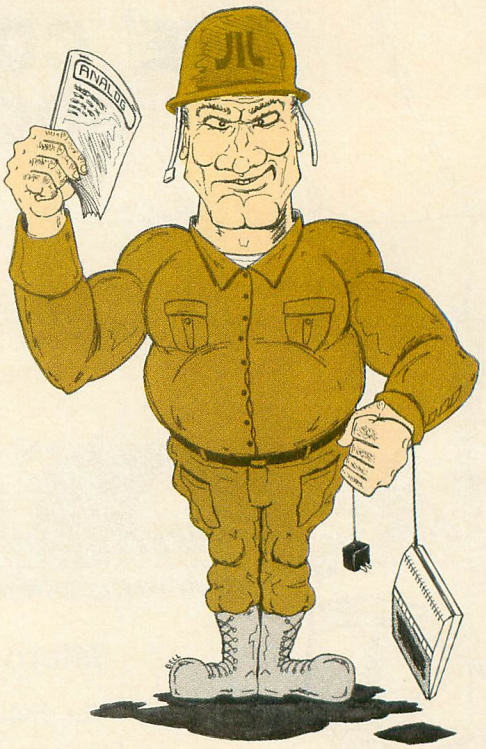
The Computerist Bookcart
P.O. Box 6502, Chelmsford, MA 01824
For faster service, phone: 617/256 – 3649.

For the Freshest Books, Buy Direct!

- No prehandled books with bent corners
- Books come direct to your door
- No time wasted searching store to store
- 24 hours from order receipt to shipment
- No shipping/handling charges
- No sales tax (except 5% MA res.)
- Check, MO, VISA/MC accepted (prepaid only)

CIRCLE #140 ON READER SERVICE CARD

BASIC Training



by Tom Hudson

So, you want to write BASIC, do you? Well, you've come to the right place. This is **BASIC Training** (also known as "My Game"), and we're in the process of writing a simple little game originally called **Battleship**.

What's in a name?

This game's been around in one form or another for years, all the way back to the forties, when it was played with paper and a pencil by two players.

But let's face it, this is the computer age, and, as long as we're going to have our Atari computers play this game, we might as well update the name. From now on, it will be known as **High Seas**.

As we all know, the name alone doesn't make a game fun or challenging. You've got to invest the *time* to make it fun. That's just what we're going to do, starting right now.

Last issue, we looked at the general flowchart for **High Seas**. This time, we're going to describe the major data elements used by the program.

What's where?

In order for the computer to be able to play **High Seas** (or any other game), we must represent the game board in memory, in a way that allows the computer to play.

The first data area we'll set up is that for the game boards. Figure 1 shows the **High Seas** game board layout.

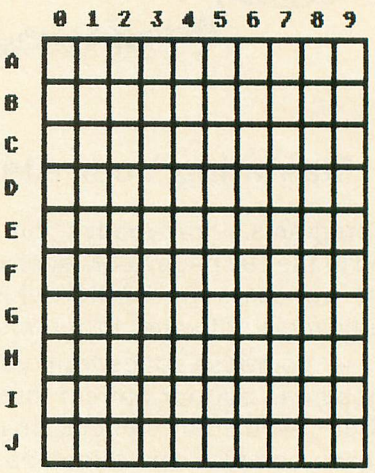


Figure 1.

In **High Seas**, there are actually two game boards, one for each player. In our application, one is for the computer, the other for the human player. Otherwise, the boards are identical.

If you've done any BASIC programming at all, you'll see that these game boards are ideally suited for computer representation. Each of the boards is simply a two-dimensional matrix with ten rows and ten columns. In Atari BASIC, we could define a game board with the statement *DIM BOARD(9,9)*. We only dimension the matrix with *nine* rows and columns, be-

cause the computer can reference the zero element, also. The only thing the computer has to do is change the A-J row labels to 0-9 internal matrix indexes, as in Figure 2.

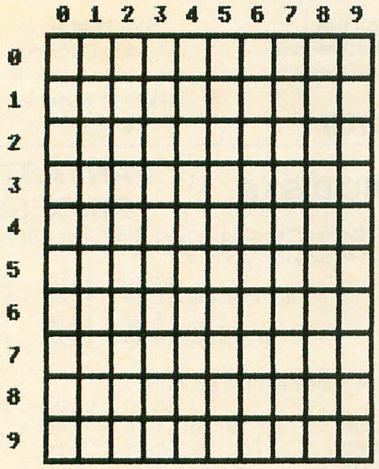


Figure 2.

In the game, we'll have two separate game boards, one for the computer's ships (we'll call it *CG0*) and one for the human player's ships (we'll call it *HG0*).

That was simple enough, right? Now, whenever we refer to the computer's board location "G4," the computer will think of it as *CG0(6,4)*.

Each of the game matrices (CG0, HG0) will contain codes indicating what type of ship is placed in each location of the board. More on that in a moment.

As the game proceeds, each player will take turns "shooting" at the other player's ships. It will be necessary to record which locations on the board have been shot at, in order to prevent shot duplication. Naturally, your first impulse is to set up another matrix for this purpose, but that's not going to work here. Let's see why not.

In a numeric matrix, like CG0, there are 100 separate elements, numbered from CG0(0,0) to CG0(9,9). Each element requires 6 bytes of memory, so the entire matrix requires 600 bytes.

High Seas requires five ten-by-ten matrices in order to work. I have already mentioned CG0 and HG0, the ship placement arrays. There are also two arrays for shot recording, and a fifth array for the computer's shooting routine. If each of these matrices is set up as a numeric matrix, together they will require 5*600 or 3000 bytes! Take it from me, there's no way **High Seas** would fit in a 16K system if the work areas were defined this way. There would simply be no room for the artificial intelligence routines.

How *are* we going to do it, then? We'll use *strings* for the shot recording matrices. If strings are used, these two data areas will only require 200 bytes, instead of 1200 bytes. This simple action alone saves 1000 bytes!

We'll call the shot recording areas *CG1$* (computer grid shots) and *HG1$* (human grid shots). Each of these strings is 100 bytes long, each byte representing a position on the grid. A zero in any position means that the location has never been shot at. A one indicates that a shot was taken, and it was a miss. A two indicates that a shot was taken, and it was a hit. Figure 3 shows how the grid positions are represented in computer memory.

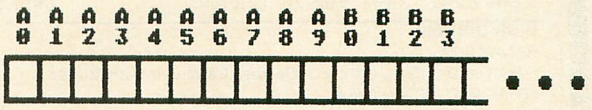


Figure 3.

In order to point to the proper position of CG1$ and HG1$, the program has to calculate a pointer from the two indexes of the game board. The equation needed to do this is:

```
POINTER = INDEX1*10 + INDEX2 + 1
```

For example, the computer's board position *B3*, held in CG0(1,3), would be calculated as:

```
1 * 10 + 3 + 1 = 14
```

After calculating the pointer, we would simply examine CG1$(14,14) and see what it contained. The matrix position (0,0) would result in a pointer value of 1, and the position (9,9) gives a pointer value of 100.

The calculation of string indexes takes somewhat longer than the equivalent matrix operations, but the memory savings are more important than speed. After all, you can make a game fast, but what good does that do if it won't fit in your computer?

The ships.

As mentioned last issue, **High Seas** is played with five types of ships. These are: destroyer (2 units long), submarine (3 units), cruiser (3 units), battleship (4 units), and aircraft carrier (5 units).

Now, computers are not very smart. Unless you set things up just right, they can't tell one ship from another. That's why we're going to assign each type of ship a specific numeric value. Figure 4 lists each ship and its number.

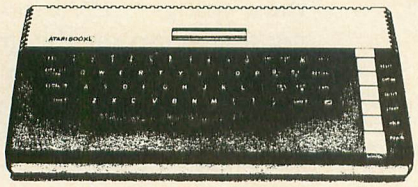
```
1 = DESTROYER
2 = SUBMARINE
3 = CRUISER
4 = BATTLESHIP
5 = A-CARRIER
```

Figure 4.

(continued on page 89)

Always the Lowest Prices on the Finest Quality at. . . COMPUTER CREATIONS

ATARI®



ATARI LEARNING PHONE SYSTEM

SYSTEM I	SYSTEM II
Atari 1030 Modem Atari Plato Cartridge	Atari Plato Cartridge Anchor Mark XII Modem (With R-Verter
$159	$349

ATARI HOME COMPUTER

ATARI 600™ Home Computer (16K RAM)	CALL FOR
ATARI 800XL™ Home Computer (64K RAM)	UPDATED PRICES

DISK DRIVES

RANA 1000
*TRAK ATD 2 (Single/Double Density/Parallel Int. 2K Buffer
*TRAK ATD 4 (Dual Drive)
TRAK CHAMP
TRAK (352)
*INDUS GT (Free Software)
ASTRA 1620
ATR-8000
HAPPY ENHANCEMENT for Atari 810 and 1050 Disk Drives
MY-DOS for ATR-8000

CALL FOR UPDATED PRICES

ATARI ADDITIONAL EQUIPMENT

ATARI 1010™ Program Recorder
ATARI 1020™ 40-Column Color Printer/Plotter
ATARI 1025™ 80-Column Printer
ATARI 1027™ Letter Quality Printer
ATARI 1030™ Direct Connect Modem
ATARI 1050™ Disk Drive
CX77 ATARI Touch Tablet™

CALL FOR UPDATED PRICES

*Screen Dump Program **FREE!!** with purchase of CX77

HARDWARE COVERS

ATARI 1010	6.95
ATARI 1200	6.95
ATARI 1025	6.95
ATARI 600XL	6.95
ATARI 1050	6.95
ATARI 800XL	6.95
ATARI 1027	6.95
GEMINI 10X	6.95
GEMINI 15X	7.49
POWERTYPE	6.95
DELTA-15	7.49
RANA 1000	6.95
INDUS	6.95
TRAK	6.95

INQUIRE FOR OTHERS

RAM (MEMORY) BOARDS

Microbits 64K (600XL) Expansion $89.95

DISKETTE/CARTRIDGE/CASSETTE FILES

Flip 'N' File 10	3.95
Flip 'N' File 15	6.95
Data Defender	16.99
Flip 'N' File/The Original	16.95
Disk Bank	3.99
Colored Library Case	2.99
Disk Bank/5 (holds 50)	12.99
Power Strip (6 outlet)	16.95
Lineguard Spike Suppressor	13.99
Disk Drive Cleaning Kit	11.99
Ring King Wallet (holds 10 disks)	4.99
Ring King Wallet (holds 20 disks)	7.99

AUTHORIZED SERVICE CENTER
ATARI ● TRAK
STAR MICRONICS (GEMINI)
Call for prices and services.

GENERIC DISKS AT FANTASTIC PRICES!!

Generic 100% Defect-Free/Guaranteed Mini-Floppy Disks
Diskettes (1 Box Minimum) - 10 Per Box

	SS/SD	SS/DD	DS/DD
1 or 2 Boxes	15.49	15.99	19.99
3 - 9 Boxes	14.49	14.99	18.99
10+ Boxes	13.99	14.49	17.99

Bulk Diskettes with Sleeves - Price per Disk

CALL*		SS/SD	SS/DD	DS/DD
DEALER	10 - 29	1.45	1.49	1.89
INQUIRIES	30 - 99	1.35	1.39	1.79
INVITED	100+	1.29	1.35	1.69

PRINTERS

*GEMINI 10X (80 Column)	269.00
*GEMINI 15X (136 Column)	399.00
*DELTA 10 (80 Column)	419.00
*RADIX 10 (80 Column)	639.00
*RADIX 15 (136 Column)	749.00
*POWERTYPE Daisywheel	379.00
EPSON RX-80 (80 Column)	339.00
EPSON RX-80 FT (80 Column)	469.00
EPSON FX-80 (80 Column)	499.00
EPSON FX-100 (136 Column)	749.00
OKIDATA 92P	449.00
EPSON RX 100 (136 Column)	439.00
TTX LETTER Quality Printer (includes Tractor Feed)	359.95

PRINTER INTERFACE CABLES

APE FACE with serial port	69.00
MPP-1150 Parallel Printer Interface	69.00

PRINTER RIBBONS

GEMINI Printers (Black/Blue/Red/Purple)	3.00
EPSON Printers	8.95

MONITORS

Sakata SC 100 Color Screen	239.00
Sakata Green Screen	99.00
Sakata Amber Screen	109.00
Monitor Cable	10.00
Sanyo 12" Green Screen	79.95
Sanyo 12" Amber Screen	79.95
Sanyo Color Screen	229.95
Sanyo 9" Green Screen	69.95

MODEMS

MPP-1000C Modem	114.95
Signalman Mark XII Modem	259.95
Mark X with R-Verter	159.95

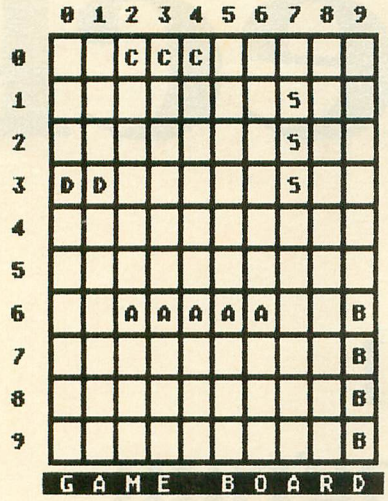
To order call TOLL FREE
1-800-824-7506
ORDER LINE ONLY

No extra charge for MasterCard or Visa

COMPUTER CREATIONS, Inc.
P.O. Bo 292467 - Dayton, Ohio 45429
For information call: (513) 294-2002 (Or to order in Ohio)

Order Lines Open - 8:30 a.m. to 8:00 p.m. Mon.-Fri.; 10:00 a.m. to 4:00 p.m. Sat. (Eastern Standard Time). FREE Catalog with each order. Minimum $10 per order. Prepaid Money Order - Free Shipping MasterCard - Visa - C.O.D. (Add $2.50). All orders add $3.50 shipping and handling in Continental United States. Actual freight will be charged outside U.S. to include Canada, Alaska, Hawaii, Puerto Rico. Price is subject to change without notice. Call for update prices and availability of product. Ohio residents add 6% sales tax. A.P.O. freight $10.00.

The ship numbers will be used in the CG0 and HG0 matrices. Each position in the matrix will contain a number reflecting the contents of that square on the game board. If the square is empty, the matrix location will contain a zero. If the square has part of a ship on it, the matrix location will contain the number of that ship. Figure 5 shows a typical computer ship setup and the contents of the CG0 array for that configuration.



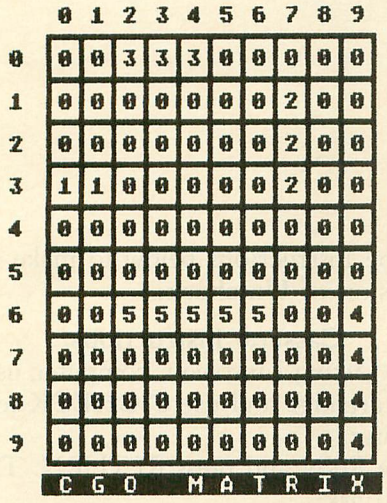


Figure 5.

As you can see, the two positions occupied by the destroyer are represented by two *1*s in the CG0 positions (3,0) and (3,1). All the other ships' locations are similarly marked. Empty squares are indicated by a *0*.

Numbers are fine for the computer—after all, it spends its whole life working with them! Humans, on the other hand, just don't feel comfortable with numbers alone. If we're told that "ship 4" has been sunk, that doesn't mean much to us. No, we'd rather see the name of the ship.

We'll use a large string, *SN$*, to hold the names of the ships in **High Seas**. By abbreviating the term *aircraft carrier* to *A-CARRIER*, all the ship names will fit in ten characters. We'll set up SN$ so that *DESTROYER* is in characters 1-10, *SUBMARINE* in characters 11-20, and so on.

When we want to recall the individual ship names, we will calculate a pointer based on the ship number, as follows:

```
POINTER = ( SHIP# * 10 ) - 9
```

We then retrieve SN$(POINTER,POINTER+9), and we have the ship name. For example, the name of "ship 1" can be found in SN$(1,10); "ship 5" has its name in SN$(41,50). Simple, right?

Of course, we'll have to initialize all our matrices at the beginning of our program, because Atari BASIC doesn't do this for us. The matrices will contain all sorts of random garbage, so we'll have to set them to zeroes with FOR-NEXT loops.

We'll also need to set up the strings we're going to use, such as SN$. The CG1$ and HG1$ strings, used for shot recording, have to be initialized to blanks, indicating that no shots have been taken.

All set?

Now that we've defined all of our data areas, we're ready to start coding, right? Wrong. We still have to flowchart the other major areas of the game, ship placement and shooting for both the human player and the computer. For next issue. . . the human's routines! □

Unicheck

The Universal Checksum Program

16K Cassette or Disk

by Tom Hudson

Many of our readers currently use the **D:CHECK2** and **C:CHECK** programs to find typing errors in the programs they enter from **ANALOG Computing**. Unfortunately, these checksum programs can be cumbersome to use. In an effort to simplify checking programs for typos, I have developed **Unicheck**.

This is a program which loads into your system at power-up time as a device, allowing you to generate a checksum table with a simple keyboard command. Your BASIC program stays in memory all the time, eliminating the annoying LIST and ENTER operations.

Typing it in.

Before typing anything, look at the listings accompanying this article.

Listing 1 is the BASIC data and data checking routine. This listing is used to create both cassette and disk versions of **Unicheck**. The data statements are listed in hexadecimal (base 16) to conserve memory.

Listing 2 is the assembly language source code for **Unicheck**, created with the OSS MAC/65 assembler. You *do not* have to type this listing to use **Unicheck**! It is included for those readers interested in assembly language.

Follow the instructions below to make a cassette or disk version of **Unicheck**.

Cassette instructions.

1. Type Listing 1 into your computer, using the BASIC cartridge and use **C:CHECK** to check your typing.

2. Type *RUN* and press RETURN. The program will ask:

```
MAKE CASSETTE (0) OR DISK (1)?
```

Type *0* and press RETURN. The program will begin checking the DATA statements, printing the line number of each as it goes. It will alert you if it finds any problems. Fix any incorrect lines and re-RUN the program, if necessary, until all errors are eliminated.

3. When all your DATA lines are correct, the computer will beep twice and prompt you to *READY CASSETTE AND PRESS RETURN*. Insert a blank cassette in your recorder, press the RECORD and P buttons simultaneously and hit RETURN. The message *WRITING FILE* will appear, and the program will create a boot tape

version of **Unicheck**, printing each DATA line number as it goes. When the *READY* prompt appears, **Unicheck** is ready to use. CSAVE the BASIC program onto a separate tape before continuing.

4. You will want to load **Unicheck** whenever you're entering programs from **ANALOG Computing**, so you can check them for accuracy. To do this, rewind the tape created by the program to the beginning. Turn your computer OFF. If you have a 400/800/1200XL computer, be sure the BASIC cartridge is inserted. Press the PLAY button on your recorder and turn ON your computer, while pressing the START button. The computer will beep once. Hit the RETURN key, and **Unicheck** will load into your computer. The *READY* prompt will appear, and you're ready to type in your program.

Disk instructions.

Type Listing 1 into your computer, using the BASIC cartridge. If you have **D:CHECK2**, use it to check your typing.

2. Type *RUN* and press RETURN. The program will ask:

```
MAKE CASSETTE (0) OR DISK (1)?
```

Type *1* and press RETURN. The program will begin checking the DATA lines, printing the line number of each statement as it goes. It will alert you if it finds any problems. Fix incorrect lines and re-RUN the program, if necessary, until all errors are eliminated.

3. When all DATA lines are correct, you will be prompted to *INSERT DISK WITH DOS, PRESS RETURN*. Put a disk with DOS 2.0S or DOS XL into drive 1 and press RETURN. The message *WRITING FILE* will appear, and the program will create an AUTORUN.SYS file on the disk, displaying each DATA line number as it goes. When the *READY* prompt appears, **Unicheck** is ready to use. Be sure the BASIC program is SAVEd before continuing.

4. You will want to load **Unicheck** whenever you're entering programs from **ANALOG Computing**, so you can check them for accuracy. To do this, place the disk containing the AUTORUN.SYS file in drive 1. Turn your computer OFF. If you have a 400/800/1200XL computer, be sure the BASIC cartridge is inserted. When you turn on your computer, **Unicheck** will load automatically. (Note: **Unicheck** will only work as an AUTORUN.SYS file. Do not try to load it with the Binary Load function.) The *READY* prompt will appear, and you're ready to type in your program.

(continued on next page)

Lyco Computer Marketing & Consultants

TO ORDER CALL US

TOLL FREE 800-233-8760
In PA 1 717-327-1824

ATARI
Computers for people.

COMPUTER CARE BIB

DISK DRIVE CLEANER	$12.75
COMPUTER CARE KIT	$19.75

NORTRONICS

DISK DRIVE CLEANER WITH SOFTWARE FOR IBM-PC, ATARI, VIC, APPLE, & TI	$29.75
DISK CLEANER REFILLS	$14.75
CASS DRIVE CLEANER	$9.95
MEDIA BULK ERASER	$46.75

MODEMS

ANCHOR MARK I	$45.75
ANCHOR MARK II	$45.95
HAYES SMART	$239.00
HAYES MICRO II	$309.00
MICROBIT 1000C	$129.00
CAT	$144.00
D-CAT	$155.00
J-CAT	$115.00
APPLE CAT II	$279.00
212 APPLE CAT	$589.00
MITEY MO	79.95

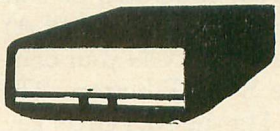
PRINTING PAPER

3000 SHEETS FANFOLD	$42.75
1000 SHEETS FANFOLD	$19.75
1000 SHEETS LETTER	$21.95
200 SHEETS LETTER	$8.99
150 RAG STATIONERY	$10.99
MAILING LABELS (1 in)	$9.75
14x11 1000 FANFOLD	$24.75

PERCOM

AT88S1 .... $249.00
AT88S1 PD $299.00

ADD-ON DRIVES	CALL
AT 88 DOUBLER	$119.00



INNOVATIVE CONCEPTS

FLIP-N-FILE 10	$3.75
FLIP-N-FILE 15	$8.95
FLIP-N-FILE 25	$18.95
FLIP-N-FILE 50	$17.75
FLIP-N-FILE (Rom Holder)	$17.75

DISK DRIVES

RANA 1000 .... $299
INDUS GT..... $325

CONCORD

ATARI 176K MASTER	$289.00
ATARI 348K MASTER	$369.00
ATARI ADD-ON DRIVES	$CALL

TRACK DRIVES

AT-D2	$329.00
AT-D2 TURBO PAK	$22.95
AT-DH	CALL
PRINTER CABLE	$22.95

SAVE ON THESE IN-STOCK PRINTERS

MANNESMANN

SPIRIT 80	$299.00
MTL-160L	$559.00
MTL-180L	$775.00

NEC

NEC8023	$369.00
NEC8025	$699.00

CITOH

GORILLA GX-100	$169.00
GORILLA SERIAL	$199.00
PROWRITER II	$575.00
8600	$899.00
SHEET FEEDER	$425.00
620	$929.00
630	$1699.00
8510 SP	$499.00
8510 SCP	$559.00
A10 LETTER QUAL	$499.00
PROWRITER 8510	$335.00
PRINTMASTER	$1199.00
STARWRITER	$929.00

STAR MICRONICS

GEMINI 15X	$379.00
DELTA 10	$449.00
DELTA 15	$525.00
RADIX 10	$575.00
RADIX 15	$675.00
POWERTYPE	$CALL
SWEET P (Model 100)	$549.00
STX80	$149.00

GEMINI 10X ..$259.00

PANASONIC

1090	$279.00
1091	$309.00
1092	$459.00

LEGEND

880	$259.00
1000	$279.00
1200	CALL
1500	CALL
1018	CALL

BLUE CHIP

M12010	$279.00
M12010 C-64	$279.00
D4015	$1399.00

EPSON

RX-80	$269.00
RX-80FT	$299.00
FX-80	$439.00
FX-100	$639.00

OKIDATA

80	$199.00
82A	$299.00
83A	$569.00
84	$959.00
92	$409.00

DUST COVERS

800	$3.99
400	$3.99
1200	$3.99
410	$3.99
810	$3.99
600 XL	$5.99
800XL	$5.99
1050	$5.99
1010	$5.99
PROWRITER	$5.99
GEMINI 10X	$5.99
PERCOM	$5.99
EPSON	$5.99
RANA	$5.99
VIC 64/20	$5.99
VIC 1541	$5.99

DISKETTES

SKC

SKC-SSSD	14.75
SKC-SSDD	17.75
SKC-DSDD	21.75

ELEPHANT

5¼" SSSD	$16.75
5¼" SSDD	$19.95
5¼" DSDD	$25.95

MAXELL

5¼" MD-1	$24.75
5¼" MD-2	$34.75

ATARI

1050 DRIVE	$299.00
1027 PRINTER	CALL
1025 PRINTER	$299.00
1020 PRINTER	$219.00
1010 RECORDER	$74.00
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
ZORK III	$34.75

BUSINESS

VISICALC	$159.75
LETTER PERFECT	$69.75
LETTER PERFECT R	$69.75
DATA PERFECT	$89.75
FILE MANAGER	$69.75
HOME FILE MGR	$69.75
BOOKKEEPER	$19.75
C.R.I.S.	$119.75
TEXT WIZZARD	$34.75
SPELL WIZZARD	$34.75

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

MONITORS

SAKATA COLOR	$249.00
TAXAN GREEN	$119.00
TAXAN AMBER	$129.00
TAXAN RGB COMPOSITE	$289.00
ZENITH AMBER	$105.00
ZENITH GREEN	$95.00
GORILLA GREEN	$88.00
GORILLA AMBER	$95.00
NEC JB1260	$99.00
NEC JB1205	$145.00
NEC JB1215 COLOR	$269.00
AMDEK GREEN	$145.00
AMDEK AMBER	$149.00
AMDEK COLOR 1	$289.00

VISA MasterCard

CUSTOMER SERVICE
1-717-327-1824

TO ORDER
CALL TOLL FREE
800-233-8760
or send order to
Lyco Computer
P.O. Box 5088
Jersey Shore, PA 17740

POLICY

Risk Free –

No deposit on C.O.D. orders; free shipping on prepaid cash orders within the continental U.S.

APO, FPO, and international orders add $5.00 plus 3% for priority mail service. PA residents add sales tax. Advertized prices show 4% discount offered for cash; add 4% for Master Card or Visa.

Checking your typing.

Once **Unicheck** is loaded into your computer, it's ready to check the typing of your programs—anytime you want. After the program is typed in, just enter:

```
LIST "U:"
```

This will print a checksum data on your screen. If you have a printer, you can enter:

```
LIST "U2:"
```

This will print the checksum data on your printer. Let's see how to interpret the checksum data. Figure 1 shows a typical checksum data table.

```
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
```

Figure 1.

Each line of the program being checked has its own checksum value. If any characters in the line are incorrect, the checksum for that line will differ from the corresponding magazine checksum. The checksum data is organized so that there are fifteen checksum values in each line, with the sixteenth 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 Figure 1, the first line checked in the first checksum line is 10. The checksum for Line 100 is 34. The checksum of the line after Line 10 is 455, and so on. The total of the checksums in the first group is 2957. 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 that the checksum data in Figure 1 was listed in the magazine, and you typed in the program and checked it with **Unicheck**. Figure 2 shows an example of what the **Unicheck** output may look like if you have typing errors.

```
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
```

Figure 2.

The first thing to do is look at the total of the values in the first line. If there are any mistyped lines, it is easiest to spot here. This value should be 2957, as shown in Figure 1. However, in the results in the **Unicheck** output, the total is 3108. This means that there is an error in the fifteen checksum values in this line. Comparing the individual **Unicheck** checksum values to the magazine values, we find that the seventh checksum is 244 in the **Unicheck** output, but should be 93. This means that the sixth line after Line 10 has an error that must be fixed. 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. The total of 2302 in our **Unicheck** output 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 fifteen lines. The line is checked the same way as the others. As you can see, the checksum line total should be 547, but is only 200 in the **Unicheck** output. Looking at the **Unicheck** 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 three errors in the program we checked. Two errors were caused by typos, and the third appeared because of a missing line.

After all errors have been noted, make the necessary changes, re-LIST the program to "U:" or "U2:" and compare the **Unicheck** output to the magazine checksum data again. Simply repeat this process until all errors are eliminated. When you're finished, you'll have an error-free program!

Some final notes.

I feel sure that users of **ANALOG Computing**'s **D:CHECK2** and **C:CHECK** will find **Unicheck** a much easier program to work with. There a few things to remember when using it, however.

Unicheck takes up about 400 bytes of memory. Some programs may be too large to load into memory with **Unicheck** present, and you'll get an ERROR-19. In these rare cases, you should use the less convenient **C:** or **D:CHECK** programs.

Don't worry about pressing RESET when **Unicheck** is loaded. It will remain safely installed until you turn your machine off (or type *DOS* in a disk system—see below).

For disk users, typing *DOS* will remove **Unicheck** from memory. This is a necessary precaution with DOS 2.0S. The first time you type *DOS*, the computer will perform a system reset and remove **Unicheck**. Typing *DOS* again will take you to the DOS menu, as usual. If you return to BASIC, however, **Unicheck** will no longer be present.

Unicheck only works with programs from issue 10 or later. If it is used with programs before issue 10, incorrect checksum values will result. □

(BASIC listing starts on page 86)

BASIC listing.

```
10 REM *** UNICHECK ***
20 TRAP 20:? "MAKE CASSETTE (0), OR DI
SK (1)";:INPUT DSK:IF DSK>1 THEN 20
30 TRAP 40000:DATA 0,1,2,3,4,5,6,7,8,9
,0,0,0,0,0,0,10,11,12,13,14,15
40 DIM DAT$(91),HEX(22):FOR X=0 TO 22:
READ N:HEX(X)=N:NEXT X:LINE=990:RESTOR
E 1000:TRAP 120:? "CHECKING DATA"
50 LINE=LINE+10:? "LINE:";LINE:READ DA
T$:IF LEN(DAT$)<>90 THEN 220
60 DATLIN=PEEK(183)+PEEK(184)*256:IF D
ATLIN<>LINE THEN ? "LINE ";LINE;" MISS
ING!":END
70 FOR X=1 TO 89 STEP 2:D1=ASC(DAT$(X,
X))-48:D2=ASC(DAT$(X+1,X+1))-48:BYTE=H
EX(D1)*16+HEX(D2)
80 IF PASS=2 THEN PUT #1,BYTE:NEXT X:R
EAD CHKSUM:GOTO 90
90 TOTAL=TOTAL+BYTE:IF TOTAL>999 THEN
TOTAL=TOTAL-1000
100 NEXT X:READ CHKSUM:IF TOTAL=CHKSUM
 THEN 50
110 GOTO 220
120 IF PEEK(195)<>6 THEN 220
130 IF PASS=0 THEN 170
140 IF  NOT DSK THEN 160
150 PUT #1,224:PUT #1,2:PUT #1,225:PUT
 #1,2:PUT #1,154:PUT #1,50:CLOSE #1:EN
D
160 FOR X=1 TO 66:PUT #1,0:NEXT X:CLOS
E #1:END
170 IF  NOT DSK THEN 200
180 ? "INSERT DISK WITH DOS, PRESS RET
URN";:DIM IN$(1):INPUT IN$:OPEN #1,8,0
,"D:AUTORUN.SYS"
190 PUT #1,255:PUT #1,255:PUT #1,0:PUT
 #1,48:PUT #1,176:PUT #1,51:GOTO 210
200 ? "READY CASSETTE AND PRESS RETURN
";:OPEN #1,8,128,"C:":RESTORE 230:FOR
X=1 TO 13:READ N:PUT #1,N:NEXT X
210 ? :? "WRITING FILE":PASS=2:LINE=99
0:RESTORE 1000:TRAP 120:GOTO 50
220 ? "BAD DATA: LINE ";LINE:END
230 DATA 0,8,243,47,158,50,169,60,141,
2,211,24,96
1000 DATA 000000000E0600000E060E064C3E
07A00160A20FB5209DD206CA10F8A9018DC206
60A20FBDD2069520CA10F860,623
1010 DATA A5CF8DCE06A5CE8DCF06A5CD8DD0
0620E206A99BAED106E002F014A20B8E4203A2
008E48038E49032056E48CC2,922
1020 DATA 0660AEB9069D0001EEB906AEB906
E026F004C99BD0EBA20BBDAD069D0003CA10F7
A9008DB9062059E48CC206A2,164
1030 DATA 27A9209D0001CA10FAA99B8D2601
60ADF006850AADF106850BADF206850CADF306
850D4C74E440015780000105,596
1040 DATA 0028004E0000204154414400000
00010A6401100000000100000000000000000
00000000000000000000000,157
1050 DATA 004C00003D5F735D019A01000000
00000000000000000000000000000000000020FF
FFD8ADEC068DE7028580ADED,971
1060 DATA 068DE8028581ADF406850AADF506
850BA000B91A03C900F005C8C8C8D0F4A90099
1B03A906991C03A955991A03,767
1070 DATA 4C00A00820120 6A9008DCD0685CD
85CE85CF85D085D1A9028DC1068DCC06A5218D
D106D00C08201206ADCC06D0,579
1080 DATA 03202D0620220628ACC206600885
D485D5201206A9008DBF068DC006A202D8A5D5
DDC306901D38FDC30685D5F8,303
1090 DATA ADBF06187DC6068DBF06ADC0067D
C9068DC0061890DBCA10D8AECC06F020A5D420
4106A5D4C920D015A9008DCC,716
1100 DATA 06A90485CCA6CCBDBA0620 4106C6
CC10F4AEC106F8ADBF061865D085D0A5D16DC0
06290F85D1CAD0ECEEC106AD,748
1110 DATA C106C904D005A9018DC106A5D4C9
9BF00820220628ACC20660A5D01865CD85CDA5
D165CE85CEA5CF690085CFD8,530
1120 DATA A9008DCE06A5D18DCF06A5D08DD0
0620E206A92C204106A90085D085D1A9028DC1
06EECD06ADCD06C90FD0B820,924
1130 DATA 2D06A90085CD85CE85CF8DCD06A9
018DCC06D0A3A90085CB85CCA4CCB9CE064A4A
4A4AD004A6CBF007E6CB0930,626
1140 DATA 204106A4CCB9CE06290FD008C002
F004A6CBF007E6CB0930204106E6CCA5CCC903
D0CA60A9018509A90685CEA9,921
1150 DATA 3085CCA90085CB85CD8D4402A8B1
CB91CD88D0F9ADE70285CDADE80285CEA93185
CCA201A000B1CB91CD88D0F9,542
1160 DATA E6CEE6CCCA10F0ADE70285CD186D
E5068D0006ADE80285CE69008D0106A5CD186D
E6068D0206A5CE69008D0306,642
1170 DATA A5CD186DE7068D0606A5CE69008D
0706A5CD186DE8068DE306A5CE6DE9068DE406
A5CD186DEA068DEC06A5CE6D,980
1180 DATA EB068DED06A5CD1869038DEE06A5
CE69008DEF06A509C901D030A002A50D8DF306
91CD88A50C8DF20691CDA50A,293
1190 DATA 8DF006A50B8DF106A5CD850CA5CE
850DA9968DF406A9068DF5066CEE06A0028409
A90691CD88A91191CDA5CD85,908
1200 DATA 02A5CE8503A9718DF406A9E48DF5
066CEE06000000000000000000000000000000
00000000000000000000000,231
```

•

CHECKSUM DATA.

(see page 90)

```
10 DATA 732,351,496,811,423,729,200,60
3,555,573,694,613,29,205,202,7216
160 DATA 760,198,962,645,494,30,155,40
7,655,57,955,761,507,330,782,7698
1070 DATA 842,694,121,189,39,986,181,9
79,141,22,300,108,149,971,5722
```

•

Assembly language listing.

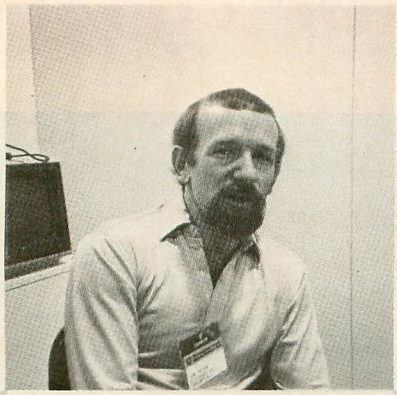
```
;          .OPT NO LIST
;UNICHECK
;Universal Checksum Program
;
;by Tom Hudson, July 1984
;ANALOG Computing
;
;page zero equates
;
LEAD0     =    $CB      ;leading 0 flag
PRTIX     =    $CC      ;# print index
TOTLO     =    $CD      ;BCD chksum total
TOTMD     =    $CE
TOTHI     =    $CF
CHKLO     =    $D0      ;BCD checksum
CHKHI     =    $D1
BYTE      =    $D4      ;incoming char
BYTE2     =    $D5      ;incoming char
;
;zero-page setup pointers
;
FROM      =    $CB
TO        =    $CD
;
;Program equates
;
CASINI    =    $02      ;cass init vector
BOOT?     =    $09      ;boot device
DOSVEC    =    $0A      ;DOS run vector
DOSINI    =    $0C      ;DOS init vector
ZIOCB     =    $20      ;zero page IOCB
ICDNOZ    =    $21      ;device # 1/2
LOMEM     =    $80      ;BASIC lo memory
PRTBUF    =    $0100    ;my print buffer
COLDST    =    $0244    ;cold start flag
MEMLO     =    $02E7    ;low memory ptr
DDEVIC    =    $0300    ;SIO device #
HATABS    =    $031A    ;handler tables
ICCOM     =    $0342    ;IOCB command
ICBLEN    =    $0348    ;CIO buf length
CIOV      =    $E456    ;CIO vector
SIOV      =    $E459    ;SIO vector
WARMSV    =    $E474    ;warm start
;
;This code is placed at $3000
;then moved to page 6 for easy
;subroutine access
;
          *=   $0600
;
;UNICHECK entry table
;
UHTBL     .WORD $00     ;open file
UC        .WORD $00     ;close file
          .WORD NOHAN-1
UW        .WORD $00     ;write file
          .WORD NOHAN-1
          .WORD NOHAN-1
          JMP OPEN      ;init vector
NOHAN     LDY #1        ;operation OK
MYRTS     RTS           ;return!
```

```
SAVE    LDX #15
SZIOCB  LDA ZIOCB,X ;save the
        STA ZIOBAK,X ;calling IOCB
        DEX         ;data
        BPL SZIOCB
        LDA #1      ;set up good...
        STA MYSTAT  ;return status
        RTS         ;and exit
RESTOR  LDX #15
RZIOCB  LDA ZIOBAK,X ;restore the
        STA ZIOCB,X ;calling IOCB
        DEX         ;data
        BPL RZIOCB
        RTS         ;and exit
;
;print checksum line total
;
PLTOTL  LDA TOTHI   ;set up chksum
        STA TOT0    ;print area with
        LDA TOTMD   ;total
        STA TOT1
        LDA TOTLO
        STA TOT2
        JSR JTOTL   ;print the amount
        LDA #155    ;get CR...
;
;print char in A reg
;
PBYTE   LDX OUTPUT  ;get output unit
        CPX #2      ;printer?
        BEQ PRINTR  ;yes!
        LDX #$0B    ;output mode
        STX ICCOM
        LDX #0      ;zero buffer
        STX ICBLEN  ;length (char in
        STX ICBLEN+1 ;accumulator)
        JSR CIOV    ;print to screen
        STY MYSTAT  ;save status
PRTXIT  RTS         ;and exit
PRINTR  LDX PBUFIX  ;get buf index
        STA PRTBUF,X ;put char in buf
        INC PBUFIX  ;next buf index
        LDX PBUFIX  ;get index
        CPX #38     ;end of buffer?
        BEQ PRNTIT  ;yes!
        CMP #$9B    ;carriage return?
        BNE PRTXIT  ;no!
PRNTIT  LDX #$0B    ;copy printer...
SPLP    LDA PCOMND,X ;commands...
        STA DDEVIC,X ;to SIO area
        DEX         ;more bytes?
        BPL SPLP    ;yes!
        LDA #0      ;zero out...
        STA PBUFIX  ;buffer index
        JSR SIOV    ;print the line!
        STY MYSTAT  ;save status
        LDX #39     ;now clear...
        LDA #32     ;print buffer
CPBUF   STA PRTBUF,X
        DEX
        BPL CPBUF
        LDA #$9B    ;set up CR
        STA PRTBUF+38 ;in buffer
        RTS         ;and exit!
MYDVEC  LDA DVSAVE  ;restore...
        STA DOSVEC  ;DOS vector
        LDA DVSAVE+1
        STA DOSVEC+1
        LDA DISAVE  ;restore...
        STA DOSINI  ;DOS init vector
        LDA DISAVE+1
        STA DOSINI+1
        JMP WARMSV  ;system reset!
;
;miscellaneous data
;
PCOMND  .BYTE $40   ;printer
        .BYTE $01   ;unit #1
        .BYTE $57   ;"W" = write
        .BYTE $80   ;output
        .WORD PRTBUF ;buffer address
        .WORD $05   ;timeout
        .WORD $28   ;buffer length
        .BYTE $4E   ;normal print
        .BYTE $00   ;unused
PBUFIX  .BYTE 0
PDATA   .BYTE " ATAD"
BCDBYT  .BYTE 0,0
X       .BYTE 0
MYSTAT  .BYTE 0
DECTBL  .BYTE 1,10,100
BCDADL  .BYTE $01,$10,$00
BCDADH  .BYTE $00,$00,$01
LFLAG   .BYTE 0
LCOUNT  .BYTE 0
TOT0    .BYTE 0
TOT1    .BYTE 0
TOT2    .BYTE 0
OUTPUT  .BYTE 0
ZIOBAK  .BYTE 0,0,0,0,0,0,0,0
        .BYTE 0,0,0,0,0,0,0,0
;
;JMP vector for subroutine
;
JTOTL   JMP $00     ;to PTOTAL
;
;Relocation factors
;
OPNDLO  .BYTE  <OPEN-START-1
CLODLO  .BYTE  <CLOSE-START-1
WRTDLO  .BYTE  <WRITE-START-1
TOTDLO  .BYTE  <PTOTAL-START
TOTDHI  .BYTE  >PTOTAL-START
CENDLO  .BYTE  <CODEND-START
CENDHI  .BYTE  >CODEND-START
NEWMLL  .BYTE 0     ;new MEMLO...
NEWMLH  .BYTE 0     ;address
JMPDVL  .BYTE 0     ;device load...
JMPDVH  .BYTE 0     ;address
DVSAVE  .BYTE 0,0   ;DOSVEC save
DISAVE  .BYTE 0,0   ;DOSINI save
MYDOS   .BYTE 0,0   ;my DOSVEC
FILLER  .BYTE 0,0,0,0,0,0,0,0
        .BYTE 0,0
;
;This section is placed at $3100
;then moved down to the old
;MEMLO location.
;
;install U: device handler
;
START   JSR $FFFF   ;init DOS
DEVICE  CLD         ;no decimal mode
        LDA NEWMLL  ;alter the...
        STA MEMLO   ;low memory...
        STA LOMEM   ;pointers
        LDA NEWMLH
        STA MEMLO+1
        STA LOMEM+1
        LDA MYDOS   ;re-point DOSVEC
        STA DOSVEC
        LDA MYDOS+1
        STA DOSVEC+1
        LDY #0      ;look for slot
UUINSLP  LDA HATABS,Y ;get entry
        CMP #0      ;open?
        BEQ GOTLOC  ;yes!
        INY         ;point to...
        INY         ;next...
        INY         ;entry
        BNE UINSLP  ;and keep looking
GOTLOC  LDA # <UHTBL ;set up...
        STA HATABS+1,Y ;handler...
        LDA # >UHTBL ;in the...
        STA HATABS+2,Y ;device table
        LDA #'U     ;call it U:
        STA HATABS,Y
        JMP $A000   ;go to BASIC!
;
;UNICHECK 'OPEN' code
;
OPEN    PHP
        JSR SAVE    ;save proc stat
        LDA #0      ;zero out...
        STA LCOUNT  ;line counter
        STA TOTLO   ;checksum total
        STA TOTMD
        STA TOTHI
        STA CHKLO   ;checksum
        STA CHKHI
        LDA #2      ;init...
        STA X       ;multiplier
        STA LFLAG   ;new line flag
        LDA ICDNOZ  ;get U1/U2 #
        STA OUTPUT  ;1=scrn 2=printer
        BNE ALLDUN  ;OPEN done!
;
;UNICHECK "CLOSE" code
;
CLOSE   PHP
        JSR SAVE    ;save proc stat
        LDA LFLAG   ;new line?
        BNE ALLDUN  ;yes!
        JSR PLTOTL  ;print last tot.
ALLDUN  JSR RESTOR  ;restore proc
        PLP
        LDY MYSTAT  ;set status
        RTS         ;all done!
;
;UNICHECK "WRITE" code
;
WRITE   PHP         ;save proc stat
        STA BYTE    ;save incoming
        STA BYTE2   ;byte in 2 places
        JSR SAVE    ;save IOCB stuff
        LDA #0      ;clear out...
        STA BCDBYT  ;BCD byte value
        STA BCDBYT+1
        LDX #2      ;this routine
BCDLP   CLD         ;converts the
        LDA BYTE2   ;binary byte
        CMP DECTBL,X ;value to BCD
        BCC NXTDIG  ;for easier
        SEC         ;handling
        SBC DECTBL,X ;the BCD value
        STA BYTE2   ;is in BCDBYT
        SED         ;and BCDBYT+1
        LDA BCDBYT
        CLC
        ADC BCDADL,X
        STA BCDBYT
        LDA BCDBYT+1
        ADC BCDADH,X
        STA BCDBYT+1
        CLC         ;force branch,
        BCC BCDLP   ;loop back!
NXTDIG  DEX         ;BDC conv done?
        BPL BCDLP   ;no, loop back!
        LDX LFLAG   ;new line?
        BEQ NOTNEW  ;no!
        LDA BYTE    ;print byte to
        JSR PBYTE   ;output device
        LDA BYTE    ;get byte again
        CMP #32     ;space?
        BNE NOTNEW  ;no!
        LDA #0      ;reset new line
        STA LFLAG   ;flag
        LDA #4      ;now print the
        STA PRTIX   ;word DATA to
PDATLP  LDX PRTIX   ;the output unit
        LDA PDATA,X ;get byte
        JSR PBYTE   ;print it!
        DEC PRTIX   ;more bytes?
        BPL PDATLP  ;yup!
NOTNEW  LDX X       ;get multiplier
        SED         ;go to decimal
CKADLP  LDA BCDBYT  ;add the BCD
        CLC         ;byte value to
        ADC CHKLO   ;the checksum
        STA CHKLO   ;amount
        LDA CHKHI   ;the AND #$0F
        ADC BCDBYT+1 ;is a modulo
        AND #$0F    ;1000 operation,
        STA CHKHI   ;so chksum < 1000
        DEX         ;add it again?
        BNE CKADLP  ;yes!
        INC X       ;inc multiplier
        LDA X       ;get mult value
        CMP #4      ;=4?
        BNE XOK     ;no, it's less.
        LDA #1      ;>3, reset it
        STA X       ;to 1
XOK     LDA BYTE    ;get byte again
        CMP #155    ;end of line?
        BEQ EOL     ;yes!
EXIT    JSR RESTOR  ;restore IOCB
        PLP         ;restore proc.
        LDY MYSTAT  ;set status
        RTS         ;exit!
;
;end of line, print checksum
;
EOL     LDA CHKLO   ;add checksum
        CLC         ;to checksum
        ADC TOTLO   ;total
        STA TOTLO
        LDA CHKHI
        ADC TOTMD
        STA TOTMD
        LDA TOTHI
        ADC #0
        STA TOTHI
        CLD         ;no more decimal
        LDA #0      ;put checksum in
        STA TOT0    ;print work area
        LDA CHKHI   ;(TOT0-TOT2)
        STA TOT1
        LDA CHKLO
```

```
        STA TOT2
        JSR JTOTL     ;print checksum
        LDA #$2C      ;get a comma
        JSR PBYTE     ;print it
        LDA #0        ;reset checksum
        STA CHKLO     ;to zero
        STA CHKHI
        LDA #2        ;reset multiplier
        STA X         ;to 2
        INC LCOUNT    ;1 more line
        LDA LCOUNT    ;get # of lines
        CMP #15       ;done 15?
        BNE EXIT      ;no, leave!
        JSR PLTOTL    ;print cksum tot
        LDA #0        ;zero out
        STA TOTLO     ;checksum total
        STA TOTMD
        STA TOTHI
        STA LCOUNT    ;and line count
        LDA #1        ;set lflag to 1
        STA LFLAG     ;(new cksum line)
        BNE EXIT      ;and exit!
;
;print checksum to screen,
;with zero-suppression
;
PTOTAL  LDA #0        ;reset leading 0
        STA LEAD0     ;indicator
        STA PRTIX     ;and print index
PTLP    LDY PRTIX     ;get index
        LDA TOT0,Y    ;and work byte
        LSR A         ;shift over
        LSR A         ;the upper
        LSR A         ;BCD digit
        LSR A
        BNE GOTP1     ;it's not zero
        LDX LEAD0     ;print leading 0?
        BEQ DIGIT2    ;no!
GOTP1   INC LEAD0     ;cancel 0 supp.
        ORA #$30      ;convert to ASCII
        JSR PBYTE     ;and print it!
DIGIT2  LDY PRTIX     ;get print index
        LDA TOT0,Y    ;get the byte
        AND #$0F      ;get low BCD
        BNE GOTP2     ;it's not zero
        CPY #2        ;last digit?
        BEQ GOTP2     ;yes, print it!
        LDX LEAD0     ;suppress zero?
        BEQ NXTPT     ;yes!
GOTP2   INC LEAD0     ;cancel 0 supp.
        ORA #$30      ;conv to ASCII
        JSR PBYTE     ;print it!
NXTPT   INC PRTIX     ;next byte
        LDA PRTIX     ;get pointer
        CMP #3        ;done all 3?
        BNE PTLP      ;no!
        RTS           ;yes, return!
;
;1-shot vector changer
;(erased after use)
;
```

```
CODEND  ;             new MEMLO here!
DISKIN  LDA #1        ;disk entry, set
        STA BOOT?     ;BOOT flag
CASSIN  LDA #$06      ;move code...
        STA TO+1      ;to $0600
        LDA #$30      ;from $3000
        STA FROM+1
        LDA #0
        STA FROM
        STA TO
        STA COLDST    ;clear cold start
        TAY           ;put 0 in Y reg
MOVE1   LDA (FROM),Y  ;move code!
        STA (TO),Y
        DEY
        BNE MOVE1
        LDA MEMLO     ;move rest...
        STA TO        ;of code...
        LDA MEMLO+1   ;to old MEMLO
        STA TO+1
        LDA #$31      ;from $3100
        STA FROM+1
        LDX #1        ;move 2 pages
MOVE21  LDY #0        ;move loop
MOVE22  LDA (FROM),Y
        STA (TO),Y
        DEY
        BNE MOVE22
        INC TO+1      ;next page
        INC FROM+1
        DEX           ;all done?
        BPL MOVE21    ;no!
        LDA MEMLO     ;ge old MEMLO,
        STA TO        ;save it,
        CLC           ;add OPEN offset
        ADC OPNDLO    ;and save in
        STA UHTBL     ;handler table
        LDA MEMLO+1   ;also hi byte
        STA TO+1
        ADC #0                        CASSET
        STA UHTBL+1
        LDA TO        ;now relocate
        CLC           ;CLOSE vector
        ADC CLODLO    ;and put in
        STA UC        ;handler table
        LDA TO+1
        ADC #0
        STA UC+1
        LDA TO        ;now relocate
        CLC           ;WRITE vector
        ADC WRTDLO    ;and put in
        STA UW        ;handler table
        LDA TO+1
        ADC #0
        STA UW+1
        LDA TO        ;now relocate
        CLC           ;total print
        ADC TOTDLO    ;address and
        STA JTOTL+1   ;put in JMP
        LDA TO+1      ;vector (JTOTL)
        ADC TOTDHI
```

```
        STA JTOTL+2
        LDA TO        ;now relocate
        CLC           ;code end
        ADC CENDLO    ;address and
        STA NEWMLL    ;save in the
        LDA TO+1      ;new low memory
        ADC CENDHI    ;hold area
        STA NEWMLH    ;(NEWMLL&H)
        LDA TO        ;now set up
        CLC           ;handler install
        ADC #3        ;vector and save
        STA JMPDVL    ;in JMPDVL&H
        LDA TO+1
        ADC #0
        STA JMPDVH
        LDA BOOT?     ;get boot flag
        CMP #1        ;disk?
        BNE CASSET    ;no!
        LDY #2        ;DOS loaded
        LDA DOSINI+1  ;get DOS init
        STA DISAVE+1  ;save it,
        STA (TO),Y    ;put in my JSR
        DEY           ;repeat for
        LDA DOSINI    ;hi byte
        STA DISAVE
        STA (TO),Y
        LDA DOSVEC    ;get old DOSVEC
        STA DVSAVE    ;and save it
        LDA DOSVEC+1
        STA DVSAVE+1
        LDA TO        ;get my start
        STA DOSINI    ;address and put
        LDA TO+1      ;in DOS init!
        STA DOSINI+1
        LDA # <MYDVEC ;replace DOSVEC
        STA MYDOS     ;with my routine
        LDA # >MYDVEC ;to remove
        STA MYDOS+1   ;the U: handler
        JMP (JMPDVL)  ;go install U:
CASSET  LDY #2        ;cass loaded
        STY BOOT?
        LDA # >MYRTS  ;get RTS address
        STA (TO),Y    ;& put in my JSR
        DEY
        LDA # <MYRTS
        STA (TO),Y
        LDA TO        ;get my start
        STA CASINI    ;address and put
        LDA TO+1      ;in cass init!
        STA CASINI+1
        LDA #$71      ;point DOSVEC...
        STA MYDOS     ;to memo pad
        LDA #$E4
        STA MYDOS+1
        JMP (JMPDVL)  ;go install U:
```



An Interview with Free Fall Associates: Jon Freeman and Anne Westfall

by Arthur Leyenberger

Anne Westfall and Jon Freeman are wedded partners. Not that it's uncommon for a married couple to consider themselves partners, but this partnership concerns computers, mainly software: video games.

Anne and Jon are principal members of Free Fall Associates, a game design and development group located in Portola Valley, California. You've probably heard of some of their games: **Archon**, **Murder on the Zinderneuf**, *and the new* **Archon II: Adept**, *all published by Electronic Arts (a review of the latter appears on page 75). Each of these games is imaginative, fun to play on several levels and well implemented. With this in mind, the following chat with Anne and Jon was, for me, both interesting and very enjoyable. I hope that you'll enjoy it, too.*

AL: Where did you get the idea for **Archon**?

JF: That came from a couple of different thoughts. For a long time I have wanted to do a fantasy chess game. This derived partly from a fantasy-oriented chess set I once saw. It had "Conan" pieces on one side and "Goblin" pieces on the other. The other source was a living chess game I once participated in.

People were dressed in armor and differently armed. They acted as the pieces, while two opponents called out their moves. When a piece moved into an occupied square, the two players had to fight it out for possession. This was done with fake swords, axes and clubs, and—since I was playing a pawn—all I had was a small shield and sword.

The first battle was a pawn-to-pawn battle, and I "killed" off the other guy. Another variable in the game was the fighting ability of each of the players. The two strongest players were a warlord (instead of a queen) and a knight. When I came up against this particular knight, I realized that, because of his ability and his equipment, I didn't stand a chance. So I thought that the least I could do was to "kill" him, in addition to him "killing" me. The battle was very short, and we ended up "killing" each other. This is where the notion of the "double-kill" in **Archon** came about.

AL: How long did it take to complete the **Archon** project?

JF: It took about six months, though

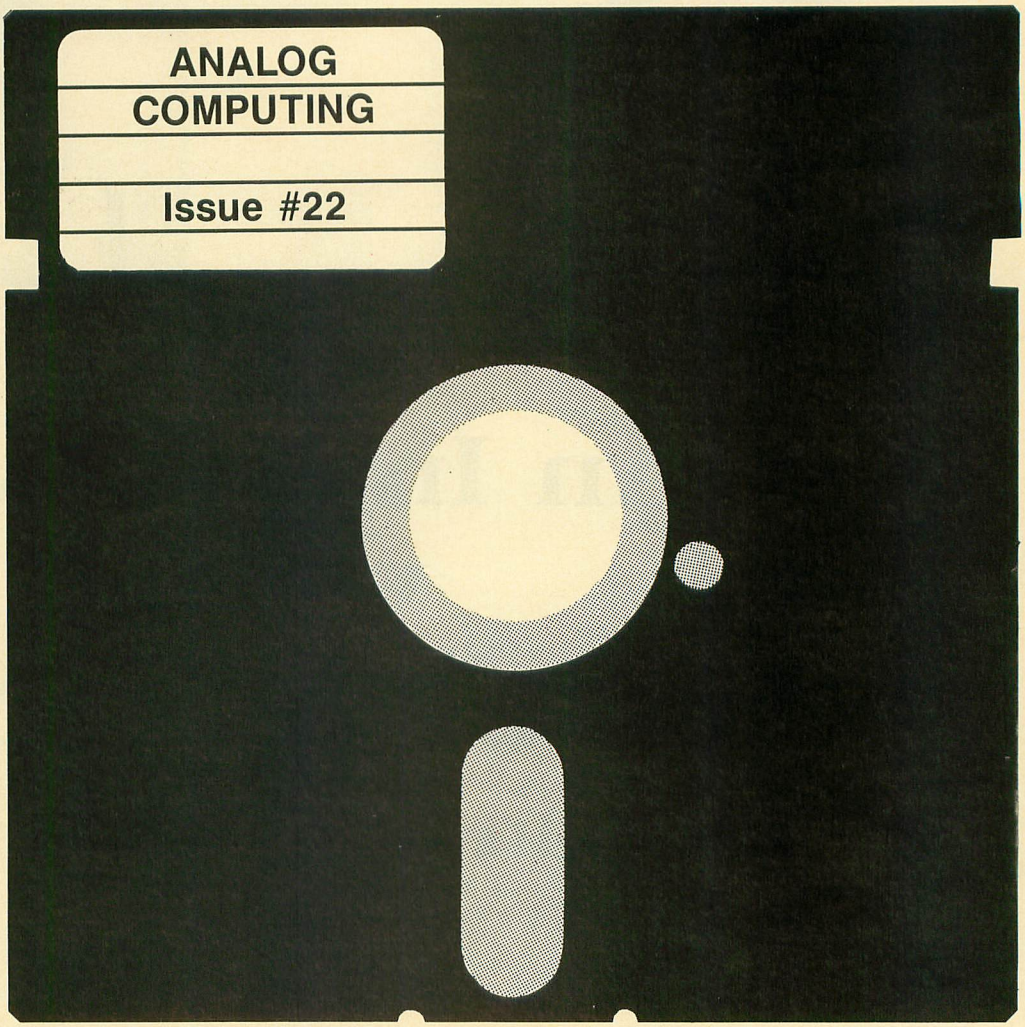
Do you subscribe to ANALOG Computing *on disk or cassette?*

. . . 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.

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 615
HOLMES, PA 19043

For Fastest Service,
Call Our **Toll-Free**
US Order Line
800-345-8112
In Pennsylvania call 800-662-2444

we were working on it night and day. Normally a game like this should take about nine months.

AW: We had a deadline and we were determined to make it.

AL: I understand that one of you is the programmer, and one of you is the designer. Who is which, and how does that work out when creating a game?

AW: I am the programmer, and Jon is the designer. It is a little hard to generalize, but usually Jon comes up with an idea that he likes, then he discusses it with me to see if I like it and if it is technically feasible.

JF: Regardless of how practical the concept is, if Anne doesn't like it, then it becomes work. We'd rather have fun while we are working. Many times we go back and forth and talk with Paul (Reiche) about the concept until we've come up with a preliminary design. We then produce an outline of the game, and Anne takes that and does a program design and the programming.

AW: That is really an oversimplification. I would love to have a script to work from, but sometimes the script is evolving while I am actually doing the programming. What we call a script is nothing like a Hollywood script. It is not that detailed.

AL: It seems that there would have to be a lot of cooperation and role-changing. If Jon did just the design and Anne just the programming, Jon would have nothing to do for most of the time during the game development.

JF: That's right, so the way that it ends up is that she does all of the programming and program design, but most of the graphics work is done by Paul and I—because she simply does not have the time. We end up doing a lot of the little things, like shape design and sound effects.

AW: They will experiment with different sounds until they have something they like. The data, such as frequency and duration, is given to me, and I come up with a sound program that will create what they want. Their routines are typically written in BASIC, because it is quickly coded and easily changed. I do the conversion to assembly language, which is quite a bit different.

It is really a dynamic process as we are going along, because they will present an idea of what they want, and I will manage to get it working so that they can see the results. It may not work the way they want it to. So we all sit back and decide what we really want, then come up with several alternatives. Some approaches may require more coding or be more appropriate for the particular game.

AL: In what ways did **Archon** finally turn out differently than what you first had envisioned?

AW: Only the tactical board turned out to be different.

JF: **Archon** turned out to be pretty much what we had wanted. We did a lot more adjusting with **Adept** than **Archon**. The look of the tactical board was a little nebulous at the beginning. We knew that we wanted a tactical board in which the players would fight, but we were also concerned that it be dynamic.

It is really a dynamic process . . . they will present an idea; I will manage to get it working so they can see the results. It may not work the way they want it to, so we all sit back and come up with alternatives.

We did not want people to just park the player and wait for the weaker piece or the piece that had to close. We ended up with the disappearing barriers so that nobody could just sit on the board. At one point, we were going to have stationary barriers and change the luminance of the background. It turned out to be a whole lot simpler to do it the other way around and change the colors of the barriers. Having disappearing barriers made the tactical board dynamic.

AW: Aside from that, it was mostly fine tuning, like adjusting the balance of the pieces. Unlike **Archon**, **Adept** changed considerably from start to finish. Almost the whole way the game is played—the strategy—is very different from what it was in the beginning. In fact, I don't remember what it was at first.

JF: It changed substantially. At the beginning, for instance, we started people out with a lot of energy, and it could not increase. We dismissed that idea pretty fast, because we felt that players wouldn't pay attention to their resources. They would spend, spend, spend, until they were almost out of energy, and only then start worrying about it. The energy you get from the different squares (the power points) was all the same, including the Void.

AW: Since the elemental power points would provide energy only when that element was active, every fourth turn, the Void was the only square that counted, since it stayed the same. Getting control of the Void was much more important than getting control of the corners, which wasn't what we had in mind.

What tended to happen was that, as it was so important, you would immediately put an Adept on the Void. Since it took two turns to get anybody else there, and you didn't want the other person getting that much energy, you'd immediately counterattack with an Adept. It ended up being a game of Adepts, attacking and counterattacking on the Void. The game would last five or six turns, and that would be it.

JF: We kicked around several possibilities, such as eliminating the Adepts or keeping them out of the Void, but we . . .wanted to give players as many options as possible to build their own armies and use whatever strategies they wanted. We did not want to prohibit Adepts from going into the Void, so we came up with two changes.

One was to reduce the energy you got from the Void, so that it was less than the corners. This brought the overall game back into balance. The second change we made was to strip the Adepts of their advantage while in the Void. Normally, the Adepts' strength is determined in part from how far they are from their home citadel. By making the Void squares absent of magic, we ended up making the Adepts weaker. Then a player will not be so eager to move an Adept onto a Void square. Also, without magic, the Adepts cannot heal themselves in the Void and must be removed to get stronger. This is like pulling a piece back in chess; you really lose two moves.

AL: In **Archon**, there is a tendency for the inexperienced player to simply battle it out, without regard to strategy. In what other ways would you characterize

the differences between the expert and inexperienced player?

JF: With **Adept**, we have not had a chance to watch players as much as we would like to. From what we have seen, new players seem to do a lot of attacking, much of it not making too much strategic sense. Attacking for attacking's sake is not a particularly good long-range strategy.

As you learn the game, you tend only to put pieces on corners and the squares around the Void, which are strategically important positions. Secondly, you tend to avoid combat and use spells a lot. You have to watch your resources, and spells are used to either keep you from having to go to battle or to set up battles in favorable circumstances, like the Weaken spell. You also become more conservative as you realize that you don't have unlimited energy.

AW: There are two ways to look at the pieces in **Adept**. Some pieces are best viewed as placeholders or defenders, who just occupy a power point. Then there are the pieces that are good at attacking, which should not be wasted, like some of the Demons. Those are put on the board initially and then used to attack piece after piece. I think that new players to **Adept** will be less likely to slug it out than they were with **Archon**. Once they have played **Adept** one or two times, they will see the value of strategy.

AL: Jon, **Adept** is obviously a sequel to **Archon**. This is something fairly new in the game world. But I see you cringe every time the word *sequel* is mentioned. Would you explain how **Adept** is or is not a sequel to **Archon**?

JF: It is a sequel, in the sense that it was an attempt to do a game that would feel—sensually and emotionally—like **Archon**. The planning and pacing is similar. There are times when you sit back and think for a while. These are followed by periods of very intense excitement and action. Then you're back to thinking again. The mechanics—such as the action board, strategy board, picking pieces and casting spells—are the same, so that someone who plays **Archon** can come in, pick up the game and start playing almost at once. So in these ways, it is a sequel. However, they will not be as good at **Adept** as they would have thought, because the pieces and strategy are so different. Also, the whole idea of resource or energy management is totally new.

AL: How do you see yourselves—as artists or as game designers—fitting into the overall cosmic scheme of things?

JF: On one level, I view myself as a game designer, and Anne sees herself as a program designer. But on another level, I think that entertainment and fun and games are an important thing, not just a fun thing.

AW: We were discussing this recently. There currently is kind of a slump in the software, and, as I look at it, the economy is doing fine. If you look back, when the economy was doing poorly and the whole world looked grim, the software business was doing great, because people need entertainment. Recreation is an important part of life. You *have* to have it.

If you look back, when the economy was doing poorly and the whole world looked grim, the software business was doing great, because people need entertainment. You have to have it.

JF: Doing games that people enjoy playing, that give them pleasure and exercise their mind, is a good thing. It is also the case that I have enough of the ex-teacher in me that there are things that I like to get across. There are certain kinds of philosophical, political or social things that would be nice to be able to influence people in. Some of it is subtle, and some of it is more overt.

The games we have been doing for Electronic Arts are not necessarily political in nature. When I wrote games for Epyx, the content was a lot more political and satirical. Many were libertarian, for instance. In our current games, there are both male and female characters. It is kind of a minor thing, and we are not really saying, "girls are okay, too," but the fact that there are male, female and neuter things in the games is important to us. That is why we put them there.

AW: "Artist" is a label that people seem to put on other people. It is not a self-imposed label. You don't walk around saying, "I have created a work of art." You don't know that, because art is a process of communication, and you do not know whether you've communicated with anybody. *They* do. If someone says that we have created a great game, then we have done a good job. We can't say that until we get feedback from people who play the games.

JF: Harlan Ellison once made a distinction between writers and authors that may be analogous. He said that real writers don't call themselves authors. That would be pretentious. An author is a person whose name is on the cover of the book. A writer is somebody who writes. If what we do can be considered a work of art, then by definition, we are artists. I don't think that is unreasonable.

AL: From what you have already said, you think that a person's beliefs come through in his or her work. To what extent do you think that a programmer's or designer's personality comes through?

JF: It is not nearly as simple as a one-to-one correspondence, but to a varying degree, it does. Things like humor or the type of violence in a game often reflect the personality of the people who created it.

AW: If you look at the difference between **M.U.L.E.** and **Seven Cities of Gold**, the personalities of Ozark's team comes through in different ways. In the first, their personalities come through rather directly. It is light, humorous—almost witty—and entertaining. Just look at the introduction screen and listen to that music. In **Seven Cities**, you don't necessarily see their personalities coming through visually until you play it. There is a signature, and when you are familiar with enough games from a particular group, you could line them up and say, "that was done by Free Fall, and that was done by Ozark."

AL: What would you ultimately like to achieve in your work, and what do you feel is the biggest constraint to doing that right now?

JF: There are three different things, two of which probably go together. We would like to be very successful commercially. . .to sell a lot of games to a lot of people. Partly because we would like to have a lot of money, but also to be able to continue what we are doing now.

I would also like to get recognition for doing a good job. That pretty much goes hand in hand with success. Apart from commercial success, I would like to do things that affect the way people think and how they look at the world.

AW: I would like to have people question things, rather than telling them that they should think this way or that . . . have them question the way they themselves think and how they view the world. To broaden people's horizons.

AL: What do you see as the main constraint to doing that?

JF: The main constraint with the latter part is imagination. I have to come up with game designs that manage to be fun and entertaining, and, at the same time, have a point. Doing that is not easy, because I don't want to do things that are heavy-handed. They have *got* to be fun. You want to charm people into thinking about things differently, rather than beating them over the head with it.

AL: Which game do you like better: **Archon** or **Adept**?

AW: It's hard to say. It's like having two children and asking which one you like better. But, having seen people playing **Archon** in tournaments, it became clear to me that I personally like a game with more strategy. Therefore, **Adept** would be my choice.

JF: I think I am still too close to them to have a favorite. If I actually had a clear favorite, in one place or another during the design I would have fallen down on the job. What I do is design games that I like to play and hope that other people will, too. I can't sell a game that only I like and, conversely, I can't work on a game that I don't like. That's too much like work.

AL: Can you talk about your next project?

JF: It's too soon to tell. We have a number of projects that we are considering, but none have reached the formal proposal stage. Part of it depends on machines. Whether we decide to do another Atari/Commodore game or maybe try something on the Apple MacIntosh, for example, is uncertain. The machines affect the type of game that you can do. For instance, you can't really do a four-player game for the Mac. It's difficult enough to do that type of game for any of these other machines. The Mac is going to probably be a one-person game.

AL: Do you think that there are too many limitations with the existing low-end computers, such as hardware, processor type and speed, or installed base?

JF: It seems to us that there are an infinite number of good games which could be done on any of the currently popular machines. I wouldn't want to try to do a game for the VIC-20. It would be too limiting, but the Atari, Commodore and Apple provide the means for a lot of different possibilities.

AW: The machines' capabilities are not a problem, but the futures of the machines may be. We really like the Atari and have liked it for a long time. But it is really a question of what's going to happen with Atari and the Atari user. That raises questions about how long it will be feasible to do games for the Atari market. I will always have my Atari, though. □

Talk to ANALOG Computing

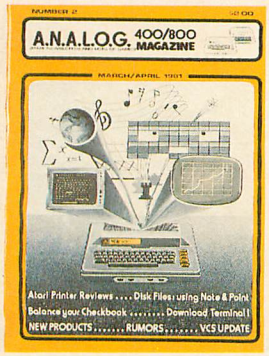
We're happy to announce that three members of our staff can now be regularly found on CompuServe. If you're a CompuServe member, you can contact Tom Hudson, Charles Bachand or Art Leyenberger by leaving a message on the Atari SIG, which can be accessed by typing GO PCS-132 at any menu page.

The Atari SIG has logged over 100,000 calls—with over 60,000 messages being posted! They have a staff of highly competent SYSOPs, headed up by Ron Luks, who are more than happy to help you. Their program database contains well over a megabyte (that's one million bytes, folks!) of Atari programs that can be downloaded into your computer.

So, if you need to get in touch with **ANALOG Computing**, you can now do it through CompuServe. Our user numbers are:

Tom Hudson 70775,424
Charles Bachand 73765,646
Art Leyenberger 71266,46

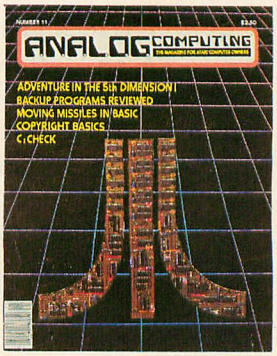
BACK ISSUES



ISSUE 2 • Wasting Arrays • Atari's CPU • Download Terminal • Converting BASIC • Disk Files

ISSUE 8 • GTIA Graphics • Audio in Your Programs • NOREM • Graphic Violence • Color Slot Machine

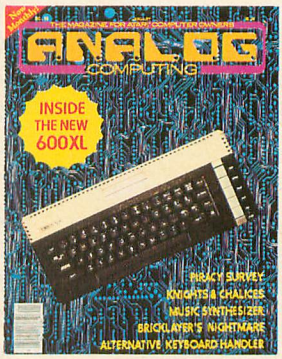
ISSUE 9 • Build Your Own 400 Keyboard • Harvey Wallbanger • Forth-DOS • Letter Writer



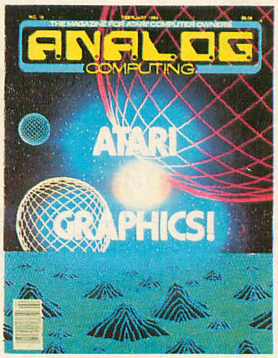
ISSUE 11 • Strings in BASIC • C:CHECK • Disk Cataloging Utility • Adventure in the Fifth Dimension • Moving Missiles in BASIC

ISSUE 13 • Fine Scrolling Part 1 • Roundup • Space Assault • Observational Astronomy • CIO Routines

ISSUE 14 • Fine Scrolling Part 2 • Disassembler in BASIC • Hexpad • Lumberjack • Retrofire!

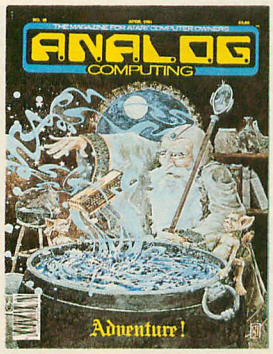


ISSUE 15 • Fine Scrolling Part 3 • Knights & Chalices • Music Synthesizer • Bricklayer's Nightmare • Keyboard Handler

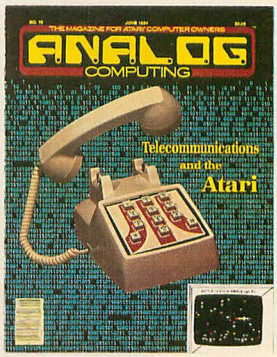


ISSUE 16 • Fine Scrolling Part 4 • Create-A-Font • Bar Chart Subroutine • Shooting Stars • 3-D Object Rotation

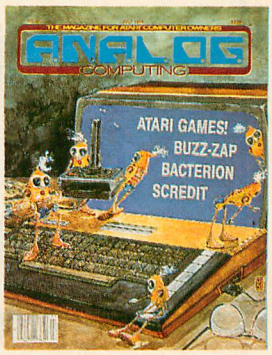
ISSUE 17 • Planetary Defense • Disk Miser • Live without DOS • Boot Camp • D:CHECK2



ISSUE 18 • Crash Dive • File'em • Munch'in Climb'in • H:BUG • Adventure Reviews



ISSUE 19 • Battle in the B-Ring • A Look at Modems • Bulletin Board Systems • Siege • Touch-Tone® Dialer



ISSUE 20 • Buzz-zap! • Scredit • AlterDOS • Bacterion! • ConTEXT

All back issues $4.00 each

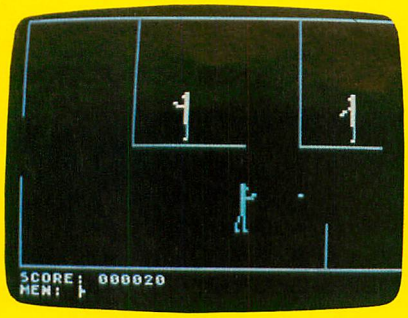
Send check or money order to:
ANALOG Computing Back Issues
P.O. Box 615, Holmes, PA 19043

Issues 2, 21, 22 and 23 are also still available.
Order now—limited quantities available.

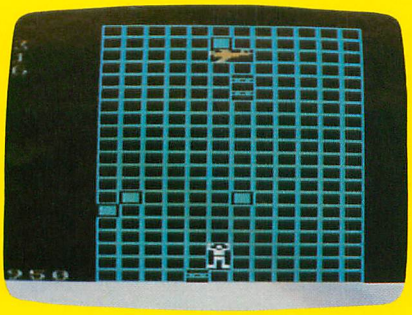
MasterCard and VISA orders call:
1-800-345-8112
in PA, 1-800-662-2444

INDEX TO ADVERTISERS

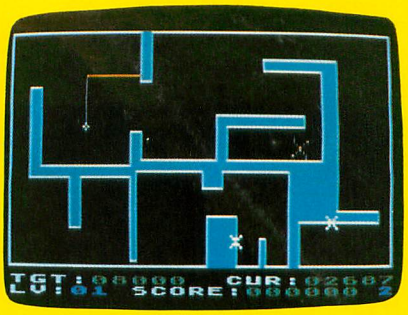
This index is an additional service. While every effort is made to provide a complete and accurate listing, the publisher cannot be responsible for inadvertent errors.



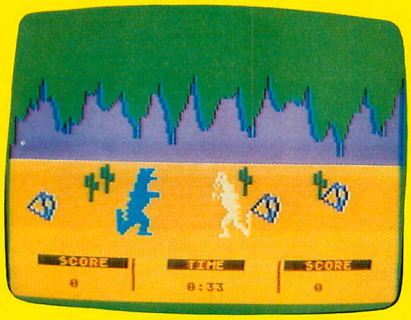
Maniac!



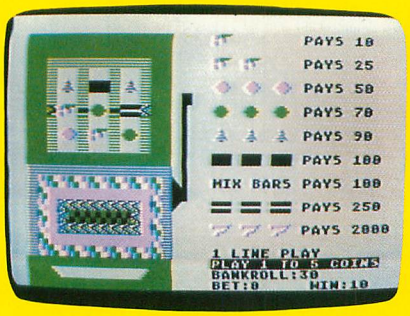
Stuntman



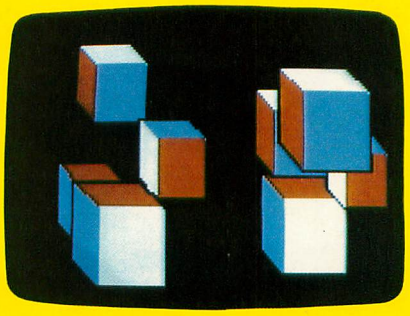
Fill 'Er Up



Dino Battle

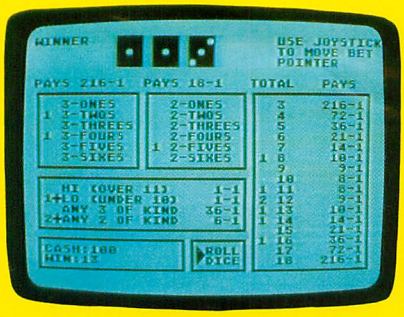


Color Slot Machine

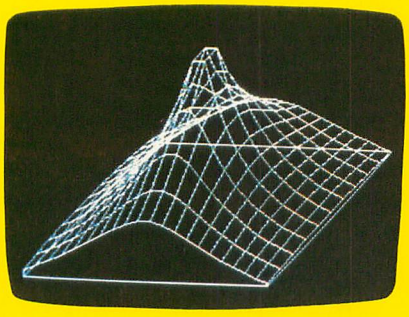


Cubes

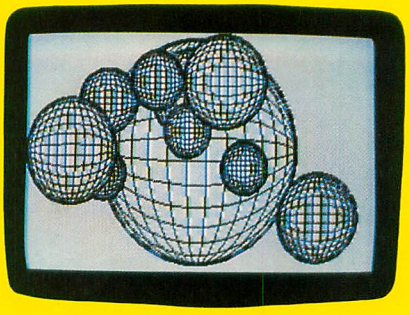
Where can you get all of these programs (and dozens more!) for only $14.95?



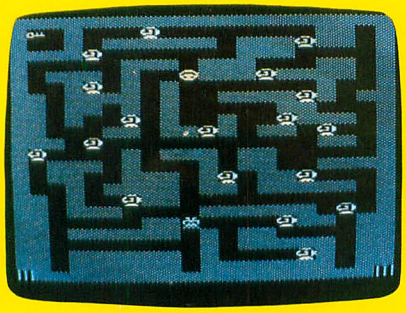
Triple Threat Dice



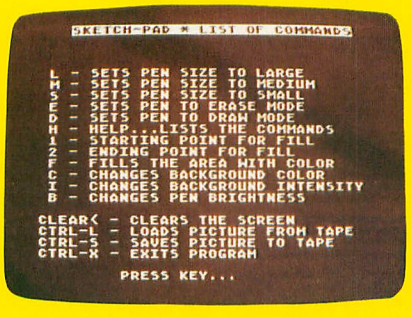
3-D Graphs



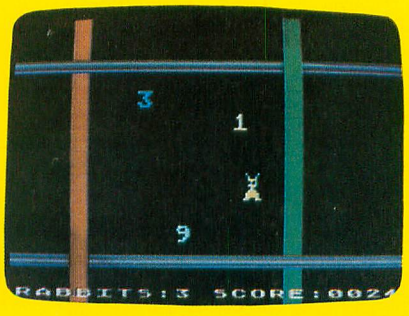
Sphere Demo



Leprechaun King



Sketch Pad



Harvey Wallbanger

From the editors of
A.N.A.L.O.G. Computing

$14.95

THE ANALOG COMPENDIUM

The best ATARI® Home Computer Programs from the first ten issues of A.N.A.L.O.G. Computing Magazine.

All Compendium programs are available on DISK.

The **ANALOG Compendium** is available at selected book and computer stores, or you can order it direct. Send a check or money order for $14.95 + $2 shipping and handling to: **ANALOG Compendium, P. O. Box 615, Holmes, PA 19043**.

Or you can order by phone with MasterCard or VISA. Call toll free: 1-800-345-8112 (in PA, call 1-800-662-2444). **For orders outside the U.S., add an additional $5 air mail, $2 surface.**