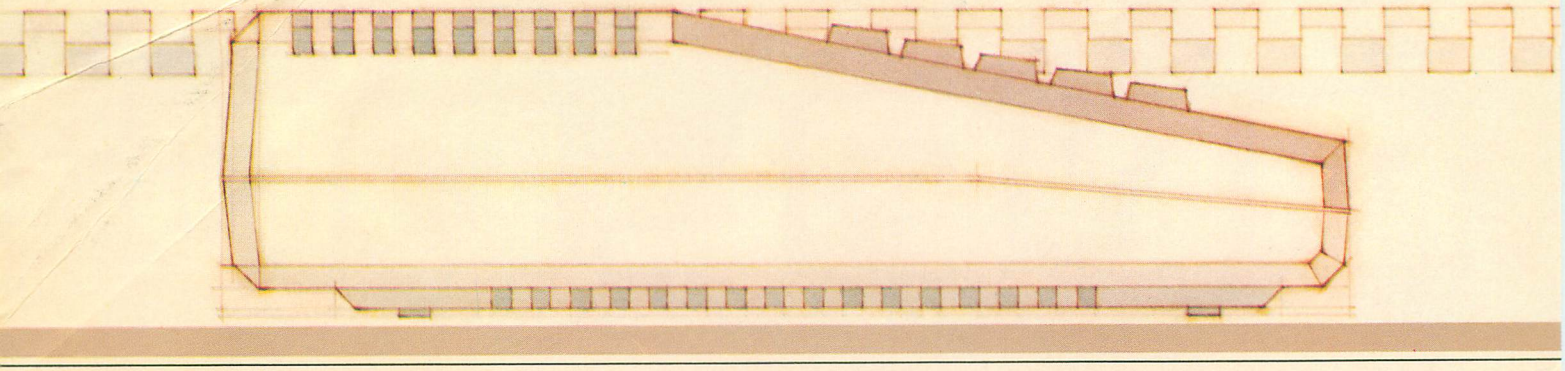
NO. 23 OCTOBER 1984 $3.00

THE MAGAZINE FOR ATARI® COMPUTER OWNERS

ANALOG COMPUTING™

FIRE BUG — *The Ultimate Magazine Game!*

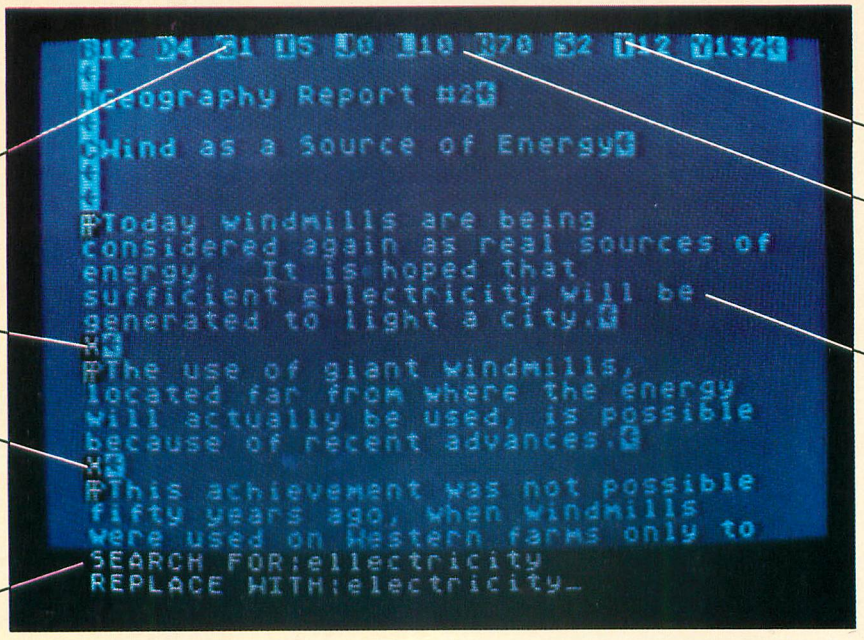


Prints with built-in format or lets you create your own: center, underline. Boldface, elongated, proportional and condensed print.

Inserts and deletes characters, lines or blocks of text.

Duplicates and moves blocks of text.

Searches for and/or replaces words or phrases, one at a time or all at once.



Changes line spacing, margins, print type, paragraph indents anywhere in the document.

Instantly reformats.

Enters text easily, with word wrap, one main menu and helpful prompts.

Creates multi-line headers and footers.

Has "print preview": you can check your copy before it's printed.

Chains and merges files.

Corrects spelling (with extra software).

Numbers pages and creates section numbers automatically.

Prints double columns, form letters, multiple copies of a page or document, at the touch of a key.

You can't find a friendlier, more powerful word processor at twice the price. New AtariWriter.™ Under $100.

Now you can do multi-featured word processing at home, simply. At a family budget price.

Our ROM-based cartridge technology means you can use new AtariWriter on any ATARI® Home Computer (even 16K) for personal and business correspondence, term papers, committee reports, mailings, etc.

It also lets you choose between cassette and disk storage systems.

One very special AtariWriter feature: you can correct as you write, without switching back and forth between Create and Edit modes.

And our memory buffer offers an "undo" command to let you change your mind, and restore text you've just deleted.

Check into our remarkable AtariWriter, and our choice of letter quality and dot matrix printers, also reasonably priced, at Atari dealers. Call 800-538-8543 for dealer nearest you. In California, call 800-672-1404.

You'll do more with Atari Home Computers.

 A Warner Communications Company

CIRCLE #101 ON READER SERVICE CARD

NO. 23

OCTOBER 1984

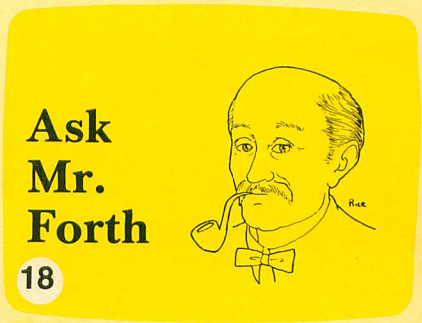
THE MAGAZINE FOR ATARI COMPUTER OWNERS

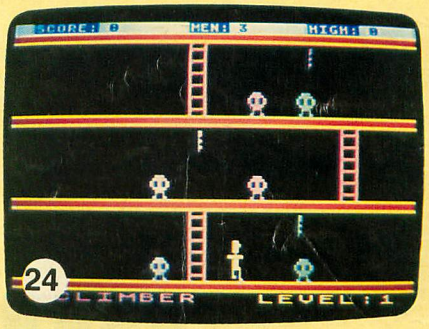
ANALOG COMPUTING

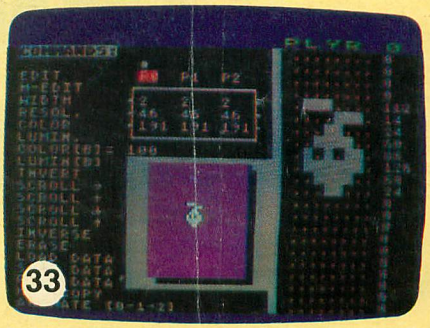
FEATURES

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.

OUR STORIES LACK IMAGINATION.

Because Infocom's interactive fiction is designed to run on *your* imagination.

That's precisely why there's nothing more interesting, challenging or interactive than an Infocom disk—but only after you've put it in your computer.

Once it's in, you experience something akin to waking up inside a novel. You find yourself at the center of an exciting plot that continually challenges you with surprising twists, unique characters (many of whom possess extraordinarily developed personalities) and original, logical, often hilarious puzzles. Communication is carried on in the same way as it is in a novel—in prose. And interaction is easy—you type in full English sentences.

But there is this key difference between our tales and conventional novels: Infocom's interactive fiction is active, never passive. The course of events is shaped by what you choose to do. And you enjoy enormous freedom in your choice of actions—you have hundreds, even thousands of alternatives at every step. In fact, an Infocom interactive story is roughly the length of a short novel in content, but because you're actively engaged in the plot, your adventure can last for weeks and months.

Find out what it's like to get inside a story. Get one from Infocom. Because with Infocom's interactive fiction, there's room for you on every disk.

INFOCOM™

For your: Apple II, Macintosh, Atari, Commodore 64, CP/M 8", DECmate, DEC Rainbow, DEC RT-11, HP 150 & 110, IBM PC* & PCjr,* KAYPRO II, MS-DOS 2.0,* NEC APC, NEC PC-8000, Osborne, TI Professional, TI 99/4A, Tandy 2000, TRS-80 Color Computer, TRS-80 Models I & III.

*Use the IBM PC version for your Compaq and the MS-DOS 2.0 version for your Wang, Mindset, Data General System 10, GRiD and many others.

CIRCLE #102 ON READER SERVICE CARD

ANALOG COMPUTING
STAFF

Editors/Publishers
MICHAEL DES CHENES
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

Contributing Artists
GARY LIPPINCOTT
DAVID NOZZOLILLO
LINDA RICE

Technical Division
CHARLES BACHAND
TOM HUDSON
KYLE PEACOCK

Advertising Manager
MICHAEL DES CHENES

Distribution
PATRICK J. KELLEY

Production/Distribution
LORELL PRESS, INC.

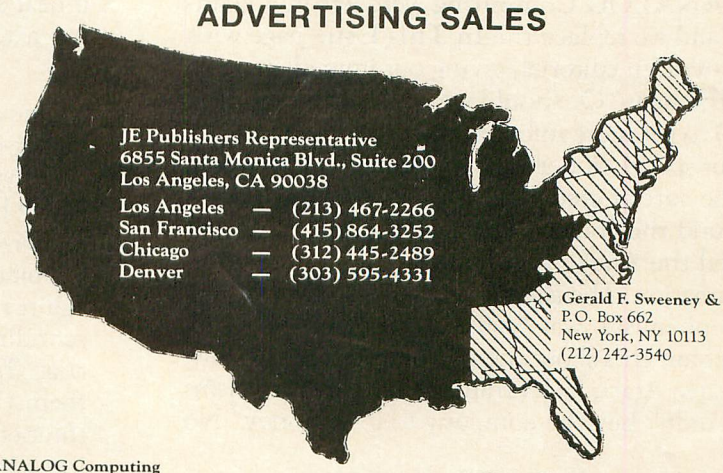
Contributors
PHILIP ALTMAN
KENNETH AMIDON
DAVID BOHLKE
BOB CURTIN
R.T. DOLBEARE
JOHN HANKE
STEPHEN JAMES
SCOTT SHECK
DAVID SHEN
RUTH ANN STONE
WAYNE UNDERWOOD

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 Des Chenes
National Advertising
(617) 892-9230

Address all advertising materials to:
Michael Des Chenes — 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.

EDITORIAL

by Jon A. Bell

Just as we went to press with our last issue (22), we received the news. Warner Communications had sold its loss-plagued Atari division to Jack Tramiel, former head of Commodore. Thus ended months of half-baked speculation and rampant rumors about the fate of Atari, once the fastest growing company in history. The anxiety and nail-biting of software producers, hardware producers and, most importantly, the readers of this magazine was replaced with. . . relief? . . . anticipation?

We at **ANALOG Computing** were left in a quandary. Should we replace the **In This Issue** page with a quickly-written editorial, giving our immediate reactions to the news? Or should we wait until next issue (this one) to get more information about the takeover, so that our speculation about the future of Atari would have some sort of credence? We placed a few phone calls around the country to gauge the reactions and monitored the Atari SIG on Compuserve to get further information. The general response, to use a White House euphemism, was "cautiously optimistic." Off-the-cuff comments about the sale ranged from, "if anyone can save Atari, Jack (Tramiel) can," to witticisms like, "he didn't buy the company to lose money." No kidding.

We decided to wait. Now, we feel that our readers want to know two things: first, a summary of what we have heard about Atari—the direction Tramiel intends for the company (either by his own admission or from the speculations of market analysts); and second, how these developments affect **ANALOG Computing** and our editorial policy.

What "they" are saying.

There are several schools of thought on where Tramiel may be taking Atari. One (strong) belief is that he will discontinue the non-profitable 600XL and flood the market with 800XLs, backing them up with a new media blitz. This would generate fast cash for new Atari projects, as well as knocking some of the stuffing out of Commodore in the Christmas sales competition. (At the time of this writing, August 1, we have heard reports of 800XLs being sold on the West Coast for $189 retail, with prices expected to go even lower before Christmas.) Prices for Atari software are also expected to be slashed dramatically.

It will be interesting to see how Jack Tramiel's old company fares against these tactics. Along with Texas Instruments, Commodore's participation in the bloody price wars of 1982 and 1983 severely damaged Atari. With a price reduction, the Atari 800XL will deal its old nemesis, the Commodore 64, a serious blow.

After cleaning out existing stocks of Atari hardware and software, then what? It's still too early to say whether Tramiel will use Christmas to get rid of old Atari products—to start afresh with new machines—or if he'll continue to manufacture the 800XL, maintaining a toehold in the low-end market.

Another possibility.

One train of thought is that Tramiel may tackle the higher-end computer market—Apple and IBM. Given this hypothesis, the next question is, "with what?" Many have assumed that Tramiel had a form of new, higher-end computer under development when he left Commodore. (The fact that Commodore has filed suit against several former employees for supposedly absconding with trade secrets doesn't necessarily mean that Tramiel's "secret" computer was taken from them.) What could this computer be? There are strong rumors in the industry that Atari has been developing a business computer based around the Motorola 68000 chip, like Apple's famed Macintosh. With color graphics and an un-Macintosh price ($1000-$1500), Tramiel would have a winner on his hands.

Boiling it down.

Again, this is all speculation. We'd advise our readers to take it with *several* grains of salt. Second guessing the computer industry is a task that few people would willingly—or could afford to—place bets on.

Where we stand.

One thing that our readers can count on is this: **ANALOG Computing** is *The Magazine for Atari Computer Owners*. Not Commodore, Apple or IBM—and that's the way it's going to stay. If Atari releases new computer systems, then they'll be covered *in* these pages with the technical savvy that has become our standard. We'll always be first in articles, product reviews and programs. That's what our readers expect.

Dedication is in what you do. . . not in what you say. □

READER COMMENT

AtariWriter for utilities.

I have been an Atari user for several years and an avid follower of **ANALOG** magazine during that time. Having just purchased a disk drive and printer, I was in the market for a word processor. After talking to other users and from reviews I've read, I decided on **AtariWriter**. I spent a little time experimenting with it and found it easy to use and everything a word processor should be.

Then I read in its handbook that the **AtariWriter** was designed to be able to load and process text generated on almost any other word processor. Knowing how programs are LISTed to disk or cassette, I stuck in my utility's disk and entered a program that I had LISTed to the disk. I entered the edit mode, and my listing looked normal. Okay, it handles a listed program as if it were text. Therefore, could I not use the power of the **AtariWriter** to edit this file? The program I had loaded was **BUNCRUSH (ANALOG** issue 7), which I had modified for screen use, as I did not have a printer at the time I started using this utility.

Well, to get to the meat of this (new word processors tend to make one long winded), I entered the search option and searched for any PRINT commands. It found the first, prompted for a replacement string, then asked if I wanted it to change all entries. Having used the **AtariWriter** to replace all PRINTs with LPRINTs, I saved it to disk (**AtariWriter** save, not DOS), installed my BASIC cartridge, and loaded a game program. At that point, I ENTERed **BUNCRUSH**. TEM, gave it a G.32500, and the printer came to life. Everything I had been getting on the screen was now coming up on the printer. It has also worked very well with GOTOs, GOSUBs and several other BASIC commands.

Was this too good to be true? Will it work on other programs? Within certain limitations, it works very well on most programs. In a matter of minutes I have converted several of my high text output programs from screen to printer.

Having been in the process of doing this manually since I bought my printer—and spending a considerable amount of time searching a program listing for PRINT and ? commands—this is so easy it's unreal! Atari made a more powerful word/program processor than even they imagined.

Miles H. Bosworth
Asheville, NC 28805

IBM — not for hackers.

I know that there are probably many Atari computer hackers who think that they have outgrown the Atari and should move up to a "real" computer, such as the IBM. Well, I have access to both an Atari and an IBM and, after using both, I think you should reconsider. Read over the following list and see what you think.

(1) *The IBM is extremely expensive.* The basic system — without a disk drive and with a monochrome monitor is over $2000. Also, believe it or not, you must buy DOS separately, for about $100! The whole computer is useless without it!

(2) *Graphics extra.* Wonderful graphics capabilities are built in on the Atari, but to get them on the IBM, you must buy a graphics card for around $500.

(3) *Unfriendly DOS.* IBM DOS doesn't even have a menu of commands that can be called up! You must keep the manual handy at all times.

(4) *No syntax checking in BASIC.* The computer will accept anything as long as it begins with a line number! You won't find out about your errors until you run the program.

(5) *Limited string capability.* Your strings can be DIMensioned to any length, but the string functions, such as MID$, only work on strings up to 255 characters.

(6) *The IBM is a memory hog.* I had the opportunity to use the IBM Pascal compiler. After typing in a short ten-statement program, I attempted to compile it. I re-

ceived the message "compiler out of memory." At the time, I had 128K RAM and two 320K double-sided disk drives. The program had filled an entire disk! Also, why does a mediocre word processor on the IBM take 128K, when a great one on the Atari only needs 16K? Good question!

Maybe now you'll look at your good ole' Atari in a new light. The IBM may be a great business machine, but it is far from a hacker's dream.

Sincerely,
James Hague
Randolph, NJ

BBS news.

We've just recently started a new BBS in north Idaho. We're calling it I.-P.A.C.E., and featuring Atari computer downloads, plus other useful items for computer hackers.

We are currently using a highly modified version of AMIS BBS software that is being enhanced and added to almost every day, to make it more user-friendly. We are now in 24-hour operation, requiring no passwords, and with no set time limits as of this writing. Our BBS phone number is (208) 772-5922.

Thank you,
The Sysops
Robert P. Marshall

While typing in Kyle Peacock's **Bacterion!**, I came up with a great idea. Why not have a national BBS (possibly with a toll-free number) for subscribers, from which they could download various programs from your magazine? While I am not a subscriber, you can bet I'd be one if such a service existed.

The thing I like about your magazine over other, similar ones is the way you list the programs as they would appear on the computer's screen. This feature has saved me a lot of frustration.

In your article on telecommunication, you only had one BBS listing in the 713 area code region. There are many good ones around here:

T.A.P.S. 713-467-0792
ACOM. 713-530-0164
H.A.C.E. 713-467-0792
(Houston Atari
Computer Enthusiasts)

Sincerely,
Paul Mitchum

We've received an announcement for a new BBS in Fayetteville, NC. Here it is:

The Soldier City BBS
P.O. Box 70
Fayetteville, NC 28302
Fred B. Deem, III — Sysop
919-323-3934

The Soldier City BBS serves the following Atari system configurations: Atari 800 (48K), Atari 850, Percom RFD44-S2 DDS Drive System, Atari 1025, and Hayes 1200 Smartmodem. The BBS is in 24-hour per day operation.

(continued on page 11)

Griffin's Lair

Educational Programs Review

by Braden E. Griffin, M.D.

Apologia Analogum: I must apologize to any of you expecting to see a review of Infocom's **Seastalker** (probably, mostly, the staff at Infocom). I was not aware of its proposed inclusion in this month's issue until I picked up issue 22—the same day I dropped this one off at the offices of **ANALOG**! I did promise to review this item soon after receiving it, but not *this* soon. When my 14-year-old son and 12-year-old daughter both made their respective all-star teams this summer, I lost my two crack game-testers to two-a-day practices. I promise (*never* promise, Griffin) that, in next month's column, I will take a look at **Seastalker**, as well as some other educational-adventure games.

Onward.

In recent years, our school systems have placed increased emphasis on science and related fields. It is unfortunate that this is often accomplished by limiting our children's exposure to art, music and the like, but that's the way it is. . .Sounds like the perfect introduction to a series of reviews pertaining to those areas that have been shortchanged and how home computers can assist in filling this void.

Wrong again, Renaissance man. Sure, it would be great if everyone were like Leonardo, *but they ain't*. It appears that, for many of our youngsters, the most predictable opportunities (read: employment, food, shelter!) are in science-related fields. Do not take this wrongly. Gifted individuals should always be encouraged to develop their unique skills. Life would, indeed, be bland, if it were not for those who show us the beauty of Nature and Man. But, if one is not strong in the "arts," then a solid foundation in the "sciences" may come in handy. Of course, some ambivalent individuals can't decide *which* way to go and often end up practicing the "art" of medicine. (And I'm going to *keep* practicing, until I get it right.) Most physicians, psychiatrists excluded, consider themselves scientists, until they are asked to provide a scientific explanation for much of what they do. Then they'll quickly fall back on that reliable old saw, "Medicine is an art, not a science." Sure it is.

Science is fun and exciting. Science is also tedious and exacting. Science is not just learning about what others have discovered as to the nature of things. It involves observation, study and experimentation in a systemized manner to provide this knowledge. Trial and error, deductive reasoning, and discovering patterns and relationships are all part of the "scientific method."

The mental discipline developed in the study of science is important in all aspects of our lives. The two programs reviewed this month will not only encourage the development of this discipline, they will

also help to stimulate interest in science itself. We should never discourage the natural curiosity of childhood, nor should we permit our children to take things for granted. Being able to substantiate the scientific principles of nature with well-founded data is a remarkable feeling. Remember, even a stopped clock is right twice a day (from Marie von Ebner-Eschenbach, 1905).

THE INCREDIBLE LABORATORY
SUNBURST COMMUNICATIONS, INC.
39 Washington Avenue
Pleasantville, NY 10570
48K Disk $49.00

Strictly speaking, **The Incredible Laboratory** is not really a science program. However, it does use a scientific theme to promote the development of problem-solving skills important in science and other areas of study.

Donning the cloak of a scientist, the player creates a monster by combining a variety of chemicals. As each chemical is selected, it is added to a bubbling beaker in the laboratory. Each selection is responsible for a particularly weird body part. Once a sufficient number of chemicals have been combined, a lab burner heats up the contents of the beaker. Upon vaporization of this concoction, the monster thus created appears on the screen. The object is to figure out which chemicals, or combinations of chemicals, are responsible for that distinct creation.

Three gradations of expertise—novice, apprentice and scientist—provide a number of variations on this basic theme. Additionally, each of these offers two methods of interaction. In the PLAY mode, one has the opportunity to experiment and discover just which chemicals relate to which body part. Using the information gathered during this phase, one may then enter the CHALLENGE mode, where this knowledge is pitted against an opponent.

Here, the players alternate in selecting a chemical to be added to the mixture. Once the necessary ingredients are present and have been vaporized, three dissimilar monsters appear on the screen, only one of which is the actual result of that specific combination of chemicals. After the players have chosen the monster each thinks is the one just created, the two phonies melt from the screen. The winner is the player who has made the correct choice.

The three levels of rank provide an excellent progression as skills are developed. In the novice level, a list of six chemicals is displayed on the screen, each chemical being responsible for an unknown but specific body part. One may select any or all of these to go into the beaker. If all of the chemicals are not used, the monster automatically adds as many others as are necessary to furnish all six body parts.

For example, the "magic powder" will always produce the same kind of eyes, while the "red dust" has a specific effect on another body segment. If a second monster is created with five of the six original ingredients, leaving out the red dust, a similar mutant will result, only this time the three heads of the earlier version have been replaced with a Medusa-like head. Having made this observation, one determines that red dust is the chemical which produces three heads. Another strategy would be to change all the ingredients save one, then deduce its unique effect by noticing which feature remained unchanged.

The Incredible Laboratory.

Keen observation and keeping a list of the results of one's experimentation help determine which chemicals give rise to distinct body components. These are similar to the problem-solving skills involved in the classic whodunit game, Clue. Each chemical produces exactly the same body feature every time one plays. If the "goose grease" produces tennis shoes, it will always produce tennis shoes. Other chemicals will be specific for other kinds of shoes. In the CHALLENGE mode, the monster is created from a list of available chemicals. This tests one's ability to discern the composite monster, based on the information acquired in the PLAY mode.

As an apprentice, the player encounters two additional skill levels. In the first, groups of three chemicals, each producing the same body part in a different form, are listed. One chemical from each group is selected until all six body parts (head, eyes, arms, torso, legs and feet) are represented. Strategies similar to those in the novice level are used. The PLAY and CHALLENGE modes resemble those above.

Level 2 offers a bit more of a challenge. Two chemicals may be selected from each group of three, and these combinations produce their own distinct monster parts, yielding a total of six different configurations from each group. At this rank, the individual chemicals and mixtures again produce the same result each time the game is played. The CHALLENGE

round presents a list of nine chemicals from which to choose, permitting one to use some of the mixtures in creating the monster.

Once one becomes a scientist, the skills practiced up to this point are really called into play. The two levels here are similar in format to those of the apprentice level, with one notable exception. Each time one plays scientist, the chemicals produce different results. A chemical previously responsible for winged arms may now produce high-heeled shoes. This means starting from scratch each game.

The same chemicals, or combinations of the same, are used in the CHALLENGE phase, once they have been successfully mastered in the PLAY mode. A number of variations in the playing format are suggested, from using a timer to limiting the number of PLAY experiments available. One could even have contests based on the creation of a particular kind of monster with similar characteristics (e.g., color specific or birdlike).

Success!

The Incredible Laboratory is a well designed game. The graphics are superb, providing a wide variety of colorful and hideously funny monsters. It plays fairly quickly and is quite user-friendly, controlled almost entirely with a joystick. Designed for ages eleven to adult, many children slightly younger would have little difficulty with this program. Even very young children will enjoy the creation of these miscreants.

The stimulation and development of problem solving skills, with an emphasis on the organization of information, make this a truly beneficial educational experience. Whether solving the mysteries of the Universe or dealing with the problems we have created for it, the fundamental approach to understanding is similar. **The Incredible Laboratory** will help establish a solid foundation on which to build.

ATARILAB STARTER SET
with TEMPERATURE MODULE
Atari Learning Systems
ATARI, INC.

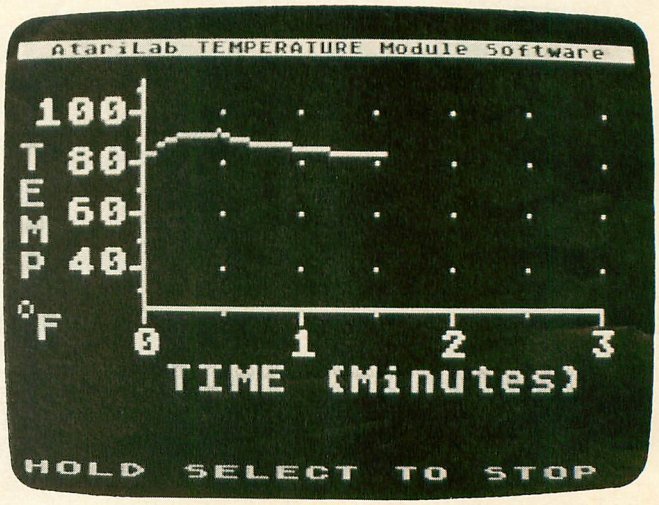
Sunnyvale, CA 94086
16K Cartridge $89.95
(Disk drive or printer optional)

The first of a proposed series of computer programs making up the **AtariLab Science Series**, this starter set includes the **AtariLab Interface** to be used with the different modules as they become available. Atari has begun this series with the **Temperature Module**.

Modules to come will contain special sensors and equipment enabling one to set up experiments dealing with light, sound, heart rate and, potentially, many others. The **Temperature Module** is composed of a 16K ROM cartridge, a temperature sensor, a thermometer and an instruction manual. This equipment, plus the interface, allows one to construct a portable laboratory station with relative ease.

The interface.

The **AtariLab Interface** is fundamental to the lab. Although it is plugged into the #2 slot for use with the **Temperature Module**, it can be inserted into any of the controller jacks of the computer. It contains eight phonojack inputs which may be used in a variety of ways.



Temperature Module.

The top two inputs are the analog inputs. Any sensor which has a resistance to the flow of electrical current similar to an Atari paddle can be connected to these (like the temperature sensor included in this set, light sensors and certain microphones). The ROM cartridge is programmed to calculate the quantity measured and translate it into meaningful information. Details on programming one's own experiments using BASIC, Logo or other computer languages are included in the manual.

The two binary inputs of the interface enable one to record information in an "on" or "off" mode (e.g., when the red fire button is pressed on a game paddle). These inputs will be used as part of a device called a "photogate" with the **Light Sensor Module**. This gives one the ability to measure the speed of moving objects, or even to read the bar code on supermarket items.

The third row of inputs are those used normally when a joystick is moved up and down. Household appliances could be turned on or off, or a small robot could be controlled using these inputs.

The final two slots are power outputs which allow users to share the +5-volt power supply with the computer and operate small devices. I am certain that there are many computer enthusiasts who will be able to perform a wide variety of projects with the interface alone.

(continued on page 92)

Reader Comment

(continued from page 7)

I'm trying to find a way to get a cursor that not only blinks but is reduced to a thin underline instead of a full block. Can you help me? A machine language subroutine for page 6 would be ideal. I enjoy your magazine a great deal —and have found it a great help, as well as a lot of fun.

Sincerely,
Patrick McShane
Nampa, ID

No problem! Check out the ***No Frills Alternate Cursor*** *in this issue.*

—TH

Send letters to:

Reader Comment

P.O. Box 23
Worcester, MA 01603

NEW PRODUCTS

by Lee Pappas

THE FIRST STARFIGHTER

Suncom now offers a high reliability joystick using a new technology which eliminates the bulk of moving parts found in most controllers. The **Starfighter** has two switches not found on other sticks—a hi/lo sensitivity control which will change the stick's response curve (speed) and a throw switch which allows the player to select either short or long movement of the joystick.

The **Starfighter** is also designed for left- and right-handed players, with an auxiliary fire button for programs requiring a second button function. The joystick is self-centering and comes with a two-year warranty (upgradable to three-year for a small additional charge).

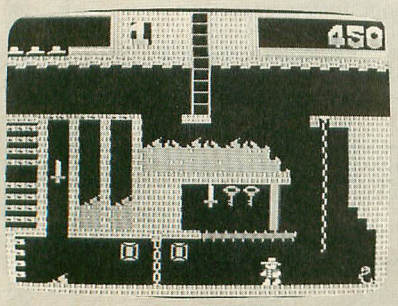
Available at $49.95 from Suncom, 260 Holbrook Drive, Wheeling, IL 60090 — (312) 459-8000.

PARKER BROTHERS' NEWEST CHALLENGE

Montezuma's Revenge puts you in the shoes of Panama Joe, in search of fantastic treasure. Guiding "Joe" through this graphic adventure pits you against laser gates, fiery pits, cobras, spiders, skulls and other assorted obstacles, including locked doors to which you must find the keys.

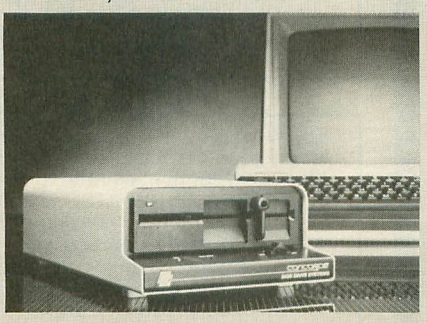
Finding magic amulets, swords, jewels and torches allows you to proceed to the pitch black lower levels, where avoiding the disappearing floors becomes even more difficult.

Available on a dual-sided disk (Atari/Commodore 64). Parker Brothers, Beverly, MA 01915.

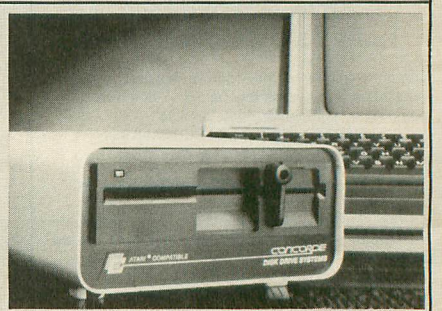


NEW ATARI DRIVE

The **Concorde C-221M** single-sided disk drive is a fairly low-priced unit, featuring the master/slave concept and running 5¼" floppy disks. The addition of up to three lower-cost slave units can bring your Atari computer up to 704KB of on-line disk storage. The **C-221M** (master) or **C-221S** (slave) provides 88K single density storage, with 176K in double density mode.



The **C-222M** or **C-222S** models are double-sided master and slave drives with 176K single density storage or 352K double density. All of these sleek-looking drives come with a one-year, over-the-counter warranty for exchange and Atari DOS. Additional features include direct drive motor, full 48-hour testing before leaving the factory and optical track-zero sensing. The master/slave concept requires purchasing one master unit to run additional slave drives.

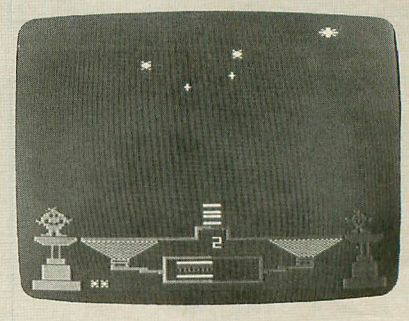


C-221M - $369, **C-221S** - $269, **C-222M** - $459, and **C-222S** - $349 all from Concorde Peripheral Systems, Inc., 23152 Verdugo Drive, Leguna Hills, CA 92653 — (714) 859-2850.

PEANUT BUTTER PANIC

Adding to its impressive array of educational Atari products, CBS Software now offers **Peanut Butter Panic**, developed with Children's Television Workshop.

This cartridge- or cassette-based program is an arcade game oriented toward children, where the object is to cooperate with others to achieve a common goal. While there are no winners—or losers—in the game, the players have fun sharing information and peanut butter sandwiches. Includes manual with activities and handy setup card.



Cartridge ($36.95) or cassette ($26.95) available from CBS Software, One Fawcett Place, Greenwich, CT 06386 — (203) 622-2525.

SIERRA'S NEW LINE

Sierra On-Line's recent change of name (to "Sierra") brings with it many new products. **Homeword** is one of the top-selling word processors for the Atari, now supported by **Homeword Speller**, a 28,000+ word dictionary of the most commonly used and misspelled words—with room for another 2,500 words of the user's choice.

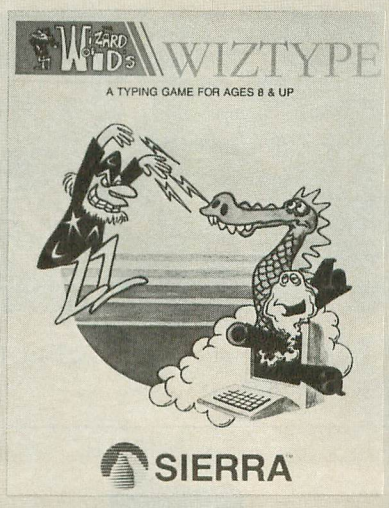
Homeword Filer integrates with the former two programs to form a complete home/family record management system. Pictures ("icons") replace endless text menus, making the program easy to use. Future Sierra plans call for **Homeword Typer**, **Tax** and **Gardener** additions to the current line.

Adding to its famous line of entertainment software, Sierra fills a long-standing gap with **B.C.'s Grog's Revenge**. This cartridge-based game presents the same style graphics used in **B.C.'s Quest for Tires**, except now Thor must stonewheel his way up a mountain and collect points along the way. The game is for one or two players.

Sierra has also announced a series of educational products, including **Wizmath**, a game for ages 8 and up that assists in the development of math skills. This program has a special scoring system that can be controlled according to ability, making it possible for a 12-year-old to compete with an adult.

Wiztype introduces basic typing and keyboard skills, as players learn to type letters and words. Spirit, a Wizard of Id character, creates the problems for the user to solve. Colorful graphics enhance the game—like the Wizard of Id zapping Spirit with a lightning bolt on correct responses; or, on slow typing, Spirit turns into a dragon and fries the Wizard with his dragon-breath. **Wiztricity** is a future release, a hi-res learning game teaching electricity.

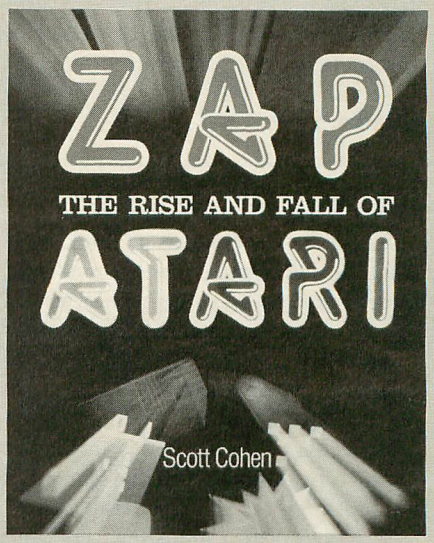
Homeword Speller lists for $49.95; **Homeword Filer**, $69.95. The two are also available in packages with **Homeword** and **Speller** for $99.95, or all three for $149.95.



B.C.'s Grog's Revenge is listed at $39.95, and the educational products on disk are $34.95 (or on cartridge $39.95). You can get them from Sierra On-Line, Inc., Coarsegold, CA 93614 — (209) 683-6858.

ZAP!

Scott Cohen's recent book, *Zap!—The Rise and Fall of Atari*, plots one of America's most famous companies from their humble beginning to recent troubled times. A book you'll probably want to read through in one night, it provides a fascinating history of Atari and Silicon Valley.



Sections cover the start of Silicon Valley in 1939 through 1972 (the founding of Atari) to 1976 (the Atari sellout to Warner), and all the way to late 1983. How Atari went from a five-hundred dollar company to a billion dollar business is fully profiled in this book.

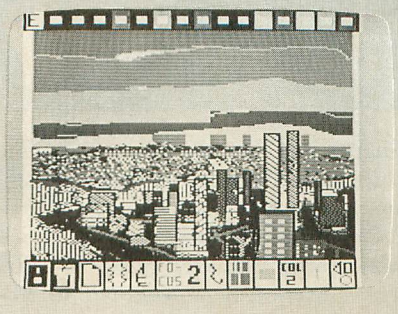
Also covered are the key people involved in the big decision making, those who were prominent in major products getting to market and the employees and executives who stirred up controversy and corruption.

Of course, Atari's latest and, possibly, most interesting chapter has yet to be written, with the Tramiel takeover on July 2, 1984.

Priced at $14.95 from McGraw-Hill Book Co., 1221 Avenue of the Americas, New York, NY 10020.

GRAPHICS PACKAGE USES LIGHT PEN

Peripheral Vision is a new, advanced graphics package designed to be used in conjunction with the **Edumate Light Pen**, both produced by Futurehouse. The former is a drawing program that features the ability to draw in up to eight colors and various textures, fill in, zoom, dump to a printer and accomplish many artful tasks.



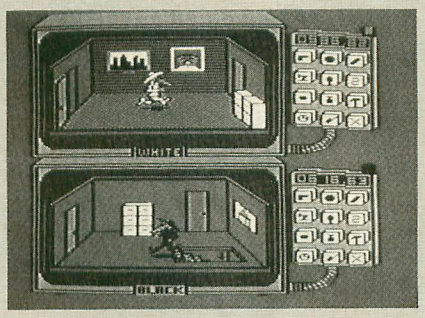
A row of "icons" at the bottom of the screen permits easy access to these functions, and a color bar at the screen top allows color selection at your whim. The manual describes how to access all of the program's capabilities, and a section for programmers is also included.

Peripheral Vision lists for $39.95 and the **Light Pen** for $34.95, or both together for $59.95—Futurehouse, Inc., P.O. Box 3470, Chapel Hill, NC 27514 — (919) 967-0861.

NOW PLAY "SPY vs. SPY"

First Star Software's recent affiliation with Warner Software brings us a new game based on *Mad Magazine*'s **Spy vs. Spy** characters. The game pivots around the zany tricks, espionage and competitiveness inherent in the *Mad* strip, so it should be perfect in a computer format.

Players take on the role of either spy and go from room to room attempting to "bomb" the other player (to the beat of crazy music).



From First Star Software, 22 East 41st St., New York, NY 10017.

"MICRO" INTERFACE

A new, low cost printer interface is available from Microbits, replacing the necessity of the Atari 850-type module. **MicroPrint** works with all software and includes a four foot cable which plugs into the serial port. The Centronics plug works with a lot of popular printers, like the Epson, C.ITOH Prowriter and NEC.

List price $79.95, from Microbits Peripheral Products, 225 Third Ave. SW., Albany, OR 97321 — (503) 967-9075.

STICKYBEAR SOFTWARE

Weekly Reader presents its **Stickybear** line of learning games for children—and several designed for the whole family. The star of these programs is none other than Stickybear himself, a full color, animated character developed just for computers.

Stickybear Numbers features over 250 picture combinations in a game that teaches numbers and counting.

In **Stickybear Opposites**, children learn the differences between up and down, in front and behind or empty and full. Kids see Stickybear drift in a hot air balloon, peddle a unicycle and watch beans grow.

Children learn their shapes in **Stickybear Shapes**, which consists of three animated activities. The players are asked to name a shape, pick a shape or find a shape. For a correct answer, the child is rewarded with cartoon animation and music.

Stickybear ABC puts colorful animated pictures on the screen accompanied by music to make learning the alphabet fun.

Stickybear Bop for kids and adults is a shooting gallery made up of planets, ducks, Stickybears, juggling Stickybears and hot air balloons that drop sandbags on you.

With **Stickybear Basketbounce**, you catch twirling, bouncing, falling bricks, donuts or stars—before you run out of baskets, get crowned on the head or are tripped by moving obstacles.

All of these programs are aimed at the 3- to 6-year-old crowd, except for the latter two family games, which are 5 to 99 and 6 to 99, respectively. All come on floppy disk with a user's guide and feature colorful packaging and various novelties, which may consist of posters, mobiles, "pop up" games, stickers or small, hardcover books.

If you want more information, contact Weekly Reader Family Software, 245 Long Hill Rd., Middletown, CT 06457 — (203) 347-7251.

SHAPES AND SOUNDS

A nicely-packaged book and disk combination for Atari 400, 800 and XL computers with a minimum of 32K and one disk drive. Written by Herb Moore, a musician, composer and writer, who also co-authored the book *Atari Sound and Graphics*.

Shapes and Sounds for the Atari covers sound effects, changing colors, graphic effects and other graphics/sound combinations, and includes chapters on how to integrate those utilities into your own programs.

Along with the two-disk set, programs are listed in the accompanying book.

Shapes and Sounds, which is designed with both the beginner and intermediate in mind, lists for $45.00.

Get it from Wiley Professional Software, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158 — (201) 469-4400.

JUST IN . . .

Lifespan synthesizes art, music and action in a series of games representing various stages of human development. **SETI (Search for Extra Terrestrial Intelligence)** is a program where you must search for, locate and decipher an alien message from space—all under a time consideration. **Twisted** is a parody of the whole text adventure genre, where the snide responses abound. $39.95 each, 48K disk from Trapeze, Inc., 3727 Buchanan St., San Francisco, CA 94123 — (415) 922-6606.

A No-Frills Alternate Cursor

16K Cassette or Disk

by Tom Hudson

Here at **ANALOG**, we aim to please. When I received a postcard from Patrick McShane (see this issue's **Reader Comment** section), I decided to tackle the challenge of writing an alternate cursor handler.

As the title implies, this cursor handler is a kind of "bare bones" program, written in about one hour, while Kyle Peacock and I were collaborating on a new game.

What it will do.

The short BASIC program in Listing 1 will install an alternate cursor in your computer. The normal "block" cursor will be replaced by a blinking underscore character. Type in the program, verify your typing with **C:CHECK** or **D:CHECK2** and SAVE the program to tape or disk *before* running it. This is necessary because the program erases itself from memory, and, if you don't save it, you'll have to retype it.

After you're sure the program's entered correctly, RUN it. After a couple of seconds, you'll see the message:

```
PRESS RESET TO INSTALL CURSOR
```

Press SYSTEM RESET, and you should see the new, improved cursor. Type LIST and press RETURN. You'll see that the BASIC program has erased itself from memory, and the cursor acts just like the normal one.

The alternate cursor will keep operating, even after SYSTEM RESET is pressed, so you don't have to worry about blowing it away if you panic and hit RESET by mistake.

You can set the "on" and "off" colors of the cursor to suit your needs. The cursor currently is white when on and black when off. To set the desired "on" color, change the 15 in Line 1030 to the desired value. Changing the 0 in Line 1040 to another value will determine the "off" color.

What it won't do.

Since the cursor routine is designed to fit in page 1 of computer memory (which is also the 6502 processor's stack), this program is only capable of operating in graphics mode 0. Don't try using it in other modes unless you feel like modifying the assembly code. (I *said* it was a no-frills program).

Do not run the BASIC program again after installing the cursor! Don't even *think* of it. If you do, the system will crash as soon as you press SYSTEM RESET. If you want to change the default cursor colors after installing the cursor, turn the system OFF and ON again before doing so.

If you want to change the cursor color while a program is running, that's fine. Just POKE 354 with the "on" color, POKE 358 with the "off" color, and everything will be dandy.

Don't use page 6 for anything. This entire block of memory is needed for the cursor's graphics area. Anything you place in this memory will be instantly erased.

How it works.

The **No-Frills Alternate Cursor** first turns off the system cursor with the CRSINH (cursor inhibit) flag. It then defines player/missile memory starting at address 0000. In this configuration, player 2 is in the memory range $0600-$06FF, or page 6.

A short deferred vertical blank routine sets player 2 to a simple underline character, reads the cursor position registers and places the cursor at the proper screen location.

That's it! I hope you'll find this alternate cursor an interesting change from the ole' Atari cursor. □

BASIC listing.

```
1 REM ********************************
2 REM *                              *
3 REM * NO-FRILLS ALTERNATE CURSOR *
4 REM *        BY TOM HUDSON         *
5 REM *      ANALOG COMPUTING        *
6 REM *                              *
7 REM ********************************
10 FOR X=256 TO 364:READ N:POKE X,N:NE
XT X:POKE 294,PEEK(12):POKE 295,PEEK(1
3):POKE 12,0:POKE 13,1
20 ? "{CLEAR}PRESS {INVERSE}RESET{INVERSE} TO INSTALL CURSOR":
NEW
1000 DATA 169,1,133,13,169,0,133,12,16
9,0,141,7,212,169,1,141,111,2,162,1,16
0,40,169,7,32
1010 DATA 92,228,169,58,141,47,2,169,2
,141,29,208,76,0,0,216,169,1,141,240,2
,169,58,141,47
1020 DATA 2,169,0,170,157,0,6,202,208,
250,165,85,10,10,24,105,48,141,2,208,1
65,84,10,10,10
1030 DATA 24,105,39,168,169,240,153,0,
6,238,109,1,173,109,1,74,74,74,41,1,20
8,4,169,15,208
1040 DATA 2,169,0,141,194,2,76,98,228
```

•

CHECKSUM DATA.

(see page 25)

```
1 DATA 507,939,3,936,237,947,519,241,3
58,0,313,210,311,220,5741
```

•

Assembly language listing.

```
;
;ALTERNATE CURSOR HANDLER
;
;BY TOM HUDSON
;
;ANALOG COMPUTING
;
DOSINI  =  $0C
PMBASE  =  $D407
HPOSP2  =  $D002
COLPM2  =  $02C2
CRSINH  =  $02F0
ROWCRS  =  $54
COLCRS  =  $55
SETVBV  =  $E45C
XITVBV  =  $E462
SDMCTL  =  $022F
GPRIOR  =  $026F
GRACTL  =  $D01D
;
;INITIALIZATION CODE
;
        *=  $0100
INIT    LDA # >INIT ;alter DOS init
        STA DOSINI+1 ;to point to
        LDA # <INIT ;our routine!
        STA DOSINI
        LDA #0      ;set up P/M
        STA PMBASE
        LDA #1      ;cursor priority
        STA GPRIOR
        LDX # >VBLANK ;point VBLANK
        LDY # <VBLANK ;to handler
        LDA #7
        JSR SETVBV
        LDA #$3A    ;turn on players
        STA SDMCTL
        LDA #$02
        STA GRACTL
ALLDUN  JMP $00     ;jump to init!
VBLANK  CLD         ;no decimal mode
        LDA #1      ;turn off cursor
        STA CRSINH
        LDA #$3A    ;turn on players
        STA SDMCTL
        LDA #0      ;erase player 2
        TAX
CP2     STA $0600,X
        DEX
        BNE CP2
        LDA COLCRS  ;get cursor column
        ASL A       ;*2
        ASL A       ;*4
        CLC         ;and add 48
        ADC #48
        STA HPOSP2  ;set horiz. pos!
        LDA ROWCRS  ;get cursor row
        ASL A       ;*2
        ASL A       ;*4
        ASL A       ;*8
        CLC         ;and add
        ADC #39     ;y-axis factor
        TAY
        LDA #$F0    ;put graphic
        STA $0600,Y ;on screen!
        INC COUNT   ;inc blink count
        LDA COUNT   ;get count
        LSR A       ;div by 8
        LSR A
        LSR A
        AND #1      ;blinking?
        BNE BLACK   ;yes!
        LDA #$0F    ;get white color
        BNE VBDONE  ;go store it!
BLACK   LDA #0      ;get black color
VBDONE  STA COLPM2  ;save color
        JMP XITVBV  ;voila!
COUNT   *=  *+1     ;blink counter
        .END
```

•

CASADAPTER
SAR-AN
12 Scamridge Curve
Williamsville, NY 14221
(716) 632-3441
$34.95

by Ruth Ann Stone

Like most Atari owners, when I originally purchased my computer I also bought the 410 Program Recorder for approximately $75.00. It wasn't long before it struck me that this was basically an ordinary cassette recorder, minus the built-in mike and earphone jacks, with the addition of a cable to connect it to the computer. This was great, because the recorder functions were controlled totally by the computer; I didn't have to worry about them. However, this recorder could not be used for other things, such as playing music, taping lectures at college, etc. Since I'm the type of person who likes one piece of equipment to perform as many functions as possible, I made a few enquiries of local Atari dealers—and came up with nothing. The 410 recorder was the only way to go (unless I wanted to invest in a disk drive), until the new 1010 recorder became available. The 1010, however, has the same limitations—it is strictly a computer program recorder, solely for use on Atari computers.

This didn't stop me, though, because I was sure that *someone* must have found a way to get around the problem. Then I saw the ad for the **Casadapter**. Sar-An's advertisement stated that this cassette interface would allow me to use any cassette recorder, or even my stereo, with all of the Atari computers. It would handle motor control, audio and data channels. All this for a mere $34.95! So, a friend and I sent out an order for two of them, post haste.

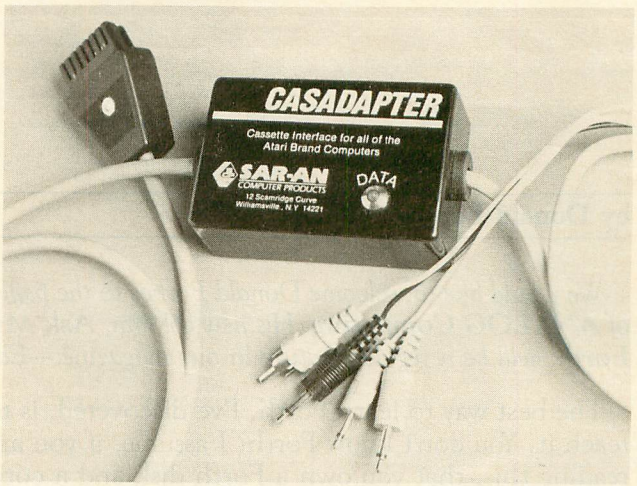
Beware the first version.

About a month later, the **Casadapter**s arrived. The packaging was impressive, and the adapter certainly *looked* official...but the only cable attached to it plugged into the computer. Lo and behold, I had to run out to my friendly neighborhood Radio Shack and purchase cables and adapters, in order to hook the **Casadapter** to either my friend's cassette or my stereo. This ran to just under $15.00 more.

Back home, after going through the directions for my fourth or fifth time, I discovered the greatest drawback to this hardware. The owner's manual was sadly and *unbelievably* inadequate for anyone who hasn't had extensive experience with stereo input/output jacks. The manual consisted of three 8½ × 11" pages, typed single-sided, double-spaced. The only real directions as to which stereo jacks hooked to which **Casadapter** jacks were on the first page...and consisted of a total of twelve lines.

After devoting an entire afternoon to this project, I did figure out how to hook the adapter to my stereo—and to my friend's portable cassette recorder—but I couldn't divine how to attach either one to the motor control jack. By this time, I was thoroughly frustrated with the adapters, so I packaged them up with a letter explaining my complaints and returned them to Sar-An the following Monday morning.

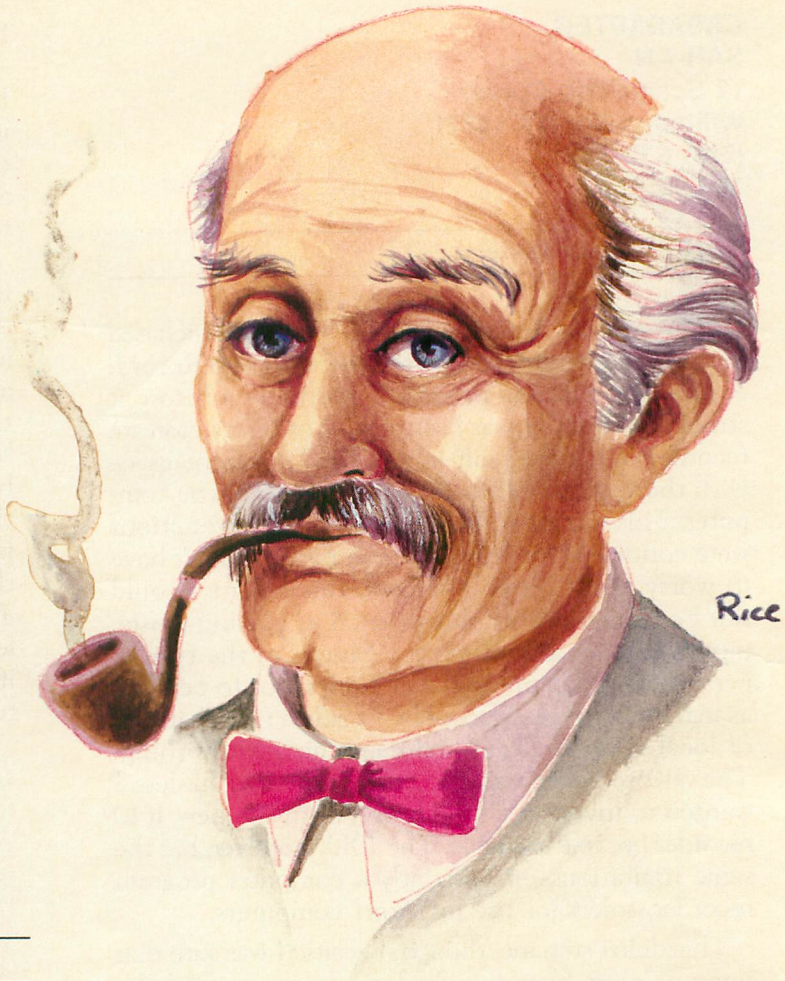
Less than a week later, I received a telephone call from Sar-An. It seems that they were unprepared for the mass response they had received to the original advertisements. Therefore, the **Casadapter**s my friend and I had received were not really ready for production; the newer versions do come complete with cables and the proper size jacks, so that they can be hooked directly to any recorder. Sar-An explained that the motor control should be hooked to a jack which is often labeled "remote" on recorders. Why didn't they say so in the owner's manual, I wanted to know. The reply was that they hadn't had the chance to perfect the manual yet, either. Therefore, I agreed that, if they would send us the latest version, we would try again.



The newer version of **Casadapter** does work very well. It hooks directly to any cassette recorder which can be placed within three feet or so of the computer. Unfortunately, my portable cassette recorder did not come equipped with a remote control jack, so I had to go out and get one that did. In so doing, I wasn't able to locate a portable recorder which could handle the audio function (for which Sar-An does not provide a cable) as well as motor control. This means that I cannot use any of the instructional tapes. They use the audio channel for the instructor's voice, which is meant to run along with the program (for example, **Invitation to Programming 1** through 3, or **Conversational Languages**). It should also be noted that the fast forward and review functions cannot be used unless the remote jack is first unplugged, which can be a bit annoying after several hours of a programming/saving/loading process.

(continued on page 20)

Ask Mr. Forth



by Donald Forbes

We would like to welcome Donald Forbes to the pages of ***ANALOG Computing****. His new column,* ***Ask Mr. Forth****, will be a regular feature in our magazine.—Ed.*

The best way to learn Forth, I've discovered, is to teach it. You don't know Forth? I assume, if you are reading this, that you own a Forth disk and a copy of Leo Brodie's tutorial book *Starting Forth*. That makes you special. There are more than 234,000,000 people in the United States and 60,000 copies of *Starting Forth* in print, so that 233,940,000 Americans don't own a copy. You can certainly tell them something.

"He who teaches learns twice" goes the old saying. "Expanding the radius of my knowledge," Einstein remarked, "extends the periphery of my ignorance." The French mathematician, Blaise Pascal, before him observed that, as the radius of his knowledge grew, the sphere of his ignorance increased. So. . .teaching, faster than anything else, will make you aware of all the gaps in your knowledge.

The demo.

To teach Forth you need, first of all, a demo. Like the job-hunter who needs a resume, a portfolio (samples of previous work) and interview savvy, the Forth teacher needs a good demonstration disk that will show off the features of Forth to good advantage.

Once you have a good demo, you will want to show it off. Atari is still tops in good computer graphics, as a visit to any of the national computer shows will prove. Game designer Chris Crawford, in his trademark collarless blue shirt, told 200 members of our user group this Memorial Day weekend that "you must remember, the Atari 800 is still the top machine in its class," and they responded to his talk with a standing ovation.

You can show the demo to your friends. Your local user group may provide an attentive audience among those who have heard of Forth but were never shown how it worked. A good demo is like money in the bank—it is good to know that it is there, and you will certainly be glad to have it at hand when the occasion arises.

The demo can be a collection of anything that will best display Forth's many advantages. Moire patterns, Fibonnacci series, prime number benchmarks, Brodie's letter F and Forth translations of common BASIC programs are all worthy candidates.

First, we need some good display screens. But which Forth? Forth is designed to be the universal operating system and language, created by programmers for programmers and transportable across all the micros.

There are at least six commercial implementations available for the Atari, of which James Albanese's 1980

QS Forth from Quality Software was the first. The most complete is **valFORTH**, which has established itself as the *de facto* standard for the Atari community. My favorite for tutorials to a wide audience is **Team Atari Forth**, the public domain, free Forth developed in the San Francisco Bay area by Steve Calfee, Harald Striepe, Peter Lipson, Robin Ziegler and others.

Our demo should work with all of them. This presents no difficulty, if we put the text on screen letter by letter. In BASIC, we could display an *A* with:

```
10 OPEN #1,8,0,"E:"
20 GRAPHICS 2
30 PUT #1,65:REM ATASCII A
```

In our Forth versions, we will need the ATASCII equivalent of each alphabetic character. This code will put the numbers on the screen:

```
: BLANK 32 EMIT ;
: ATASCII 91 65 DO I EMIT
  BLANK I . BLANK LOOP ;
```

QS Forth (after *1 LOAD LOAD-ED LOAD-IO*) will put an *A* on the screen with *2 GR. 65 6 PUT.* Rather than type *6 PUT* after each letter, we can define a single non-Forth character (%) to do it for us (after *EDITOR 1 CLEAR 1 LIST 1 L*) with *: % 6 PUT ; .*

Calfee's Forth requires *HEX 30 LOAD DECIMAL* to load the utilities and (if *34* is an empty screen) *34 WIPE 34 LIST 34 UE* for our edited text. The screen must be opened for output with *83 PAD C!* (where *83* is an ATASCII *S*) and *PAD 8 0 3 OPEN*. With *2 GR. 65 PUT*, we can now place an *A* at the top of the screen. We can use *: % PUT ;* to avoid repeated *PUT*'s.

The **valFORTH** 1.1 disk should be loaded with the printer, assembler, color, graphics, editor and operating system words, requiring *38LOAD 76 LOAD 100 LOAD 104LOAD 140LOAD 162LOAD*. Our **valFORTH** also requires an initialization with ATASCII *S PAD C!* and *PAD80 3OPEN*. Then *2 GR. 65 3 PUT DROP* will place an *A* at the top of the screen. We can abbreviate this with *: % 3 PUT DROP ; .*

What do we use for the demo text? Remember that we have twenty columns and ten lines in the *2 GR.* mode (or twenty lines if you prefer *1 GR.*) A pad of square ruled graph paper may come in handy. Here is one choice out of many:

```
: DEMOTEXT 7 0 POS.
  71 % 76 % 79 % 66 % 65 % 76 %
  3 2 POS.  84 % 72 % 69 % 82 %
  77 % 79 % 78 % 86 % 67 % 76 %
  69 % 65 % 82 %
  8 4 POS.  87 % 65 % 82 %
  9 6 POS.  66 % 89 %
  3 8 POS.  68 % 79 % 78 % 65 %
  76 % 68 % 32 % 70 % 79 % 82 %
  66 % 69 % 83 % ;
```

If this seems tedious, remember that it gets the job done, that it will work with the Forth that you have, and it's easy to modify. Furthermore, we now know how to translate a BASIC *PUT* statement into a Forth *PUT* statement.

What we have accomplished to date represents a significant first step: we can now place any text on the screen in graphics mode one or two.

To add some excitement, let us begin by cycling all the colors through the border and background color registers. This code will work:

```
: DELAY 2000 0 DO LOOP ;
: COLORS 255 0 DO I 712 C!
  I 710 C! DELAY LOOP ;
```

Now we can put it all together:

```
Scr # 1
  0 ( demo 1                  sf 840531 )
  1   DECIMAL
  2 ( QS Forth code )
  3 : INITIALIZE ;
  4 : % 6 PUT ;
  5 ( Team Atari Forth code )
  6 : INITIALIZE 83 PAD C!
  7   PAD 8 0 3 OPEN ;
  8 : % PUT ;
  9 ( valFORTH code )
 10 : INITIALIZE ASCII S PAD C! ;
 11   PAD 8 0 3 OPEN ;
 12 : % 3 PUT DROP ;
 13
 14 : DELAY 2000 0 DO LOOP ;
 15
```

```
Scr # 2
  0 ( demo 2                  sf 840531 )
  1 : DEMOTEXT 7 0 POS.
  2   71 % 76 % 79 % 66 % 65 % 76 %
  3   3 2 POS.  84 % 72 % 69 % 82 %
  4   77 % 79 % 78 % 86 % 67 % 76 %
  5   69 % 65 % 82 %
  6   8 4 POS.  87 % 65 % 82 %
  7   9 6 POS.  66 % 89 %
  8   3 8 POS.  68 % 79 % 78 % 65 %
  9   76 % 68 % 32 % 70 % 79 % 82 %
 10   66 % 69 % 83 % ;
 11 : COLORS 255 0 DO I 712 C!
 12   I 710 C! DELAY LOOP ;
 13 : DEMO INITIALIZE 2 GR.
 14   DEMOTEXT COLORS 0 GR. ;
 15
```

In BASIC, we could have created our text screen with:

```
10 GRAPHICS 2
20 ? #6;"       GLOBAL"
30 ? #6;"  THERMONUCLEAR"
40 ? #6;"        WAR"
50 ? #6;"         BY"
60 ? #6;"  DONALD FORBES"
80 GOTO 80
```

The translation into Forth would have been easy with either **valFORTH** or **QS Forth**, because they both have a special word (*G"* and *GR."* , respectively) that is missing in **Team Atari Forth**. This way is less instructive, but much easier and faster:

```
: DEMOTWO 2 GR.
  7 0 POS. G" GLOBAL"
  3 2 POS. G" THERMONUCLEAR"
  8 4 POS. G" WAR"
  9 6 POS. G" BY"
  3 8 POS. G" DONALD FORBES" ;
```

However, it is useful to be aware of both methods.

If we write:

```
: FOREVER BEGIN DEMO 0 UNTIL ;
```

the demo will run forever. Our demo is now off to a good start. But what do we put in it? Sound, perhaps? You undoubtedly have some ideas of your own. Certainly it should be something to show off the many unique benefits of Forth. Well, that belongs with next month's story.

As the Marquise du Deffand wrote on July 7, 1763 to the famous illegitimate mathematician, Jean le Rond d'Alembert: "The distance doesn't matter; it is only the first step that is difficult." □

Send letters to:

Ask Mr. Forth

P.O. Box 23

Worcester, MA 01603

Casadapter Review

(continued from page 17)

To date, I have been unsuccessful in satisfactorily interfacing **Casadapter** with my stereo, which could probably handle all the advertised functions, including the audio channel (but would still require that I pull out the remote jack, in order to use the fast forward or rewind).

The verdict.

Overall, the latest version of the **Casadapter** has a great deal of potential. It does allow the use of a regular cassette recorder for saving and/or loading programs, thus eliminating the need for a single-use piece of equipment. However, in my opinion, the manual falls far short of the needs of a novice, or even an individual with a limited amount of experience in hooking up electronic equipment. I would suggest that Sar-An describe other common labels for the input/output jacks, in the event that a user's stereo or cassette is labeled differently (for instance, as mentioned earlier, the motor jack is often labeled "remote"). I would give the **Casadapter** a grade of nine (on a scale of one to ten), if the manual were *vastly* improved, and if Sar-An were to include a cable for use on the audio jack. □

Attention Programmers!

ANALOG Computing is interested in programs, articles, and software review submissions dealing with the Atari home computers. If you feel that you can write as well as you can program, then submit those articles and reviews that have been floating around in your head, awaiting publication. This is your opportunity to share your knowledge with the growing family of Atari computer owners.

ANALOG pays between $30.00-$360.00 for all articles. All submissions for publication must be typed, upper and lower case with double spacing. Program listings should be provided in printed form, and on cassette or disk. By submitting articles to **ANALOG Computing**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ANALOG**. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply a self-addressed, stamped envelope. All submissions of any kind must be accompanied by the author's full address and telephone number.

Send programs to:

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

ALOG COMPUTING PRESENTS

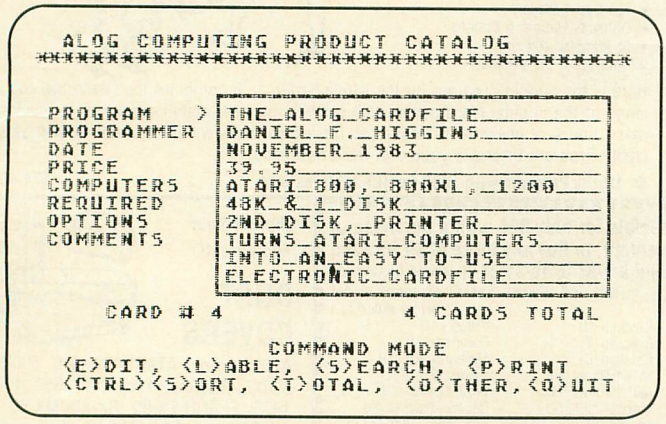
HIGH QUALITY PROGRAMS AT AFFORDABLE PRICES FOR ALL ATARI COMPUTERS WITH 48K AND 1 DISK DRIVE.

SUMMER SPECIAL — BUY TWO & GET ONE FREE!*

*** WHEN ORDERING DIRECT FROM ALOG**

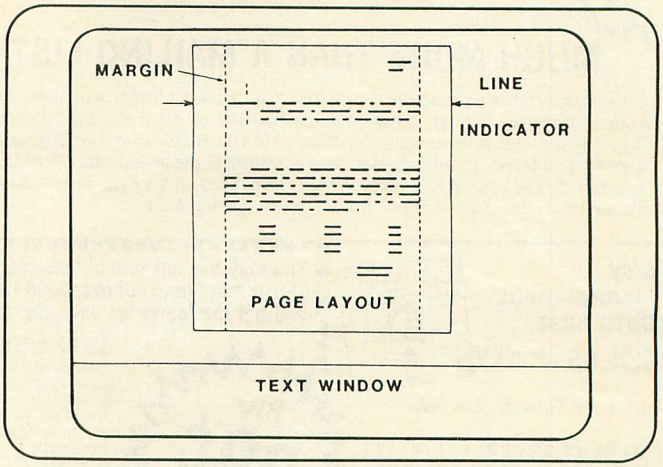
★★★★NEW!★★★★
THE ALOG CARDFILE

TURNS YOUR ATARI INTO AN ELECTRONIC CARD FILE. SEARCH & SORT ON ANY LINE. FORMATTED PRINT AND NUMERIC FIELDS MAKE THE CARDFILE IDEAL FOR YOUR EXPENSE RECORDS, INVENTORIES, AND RECEIPES.



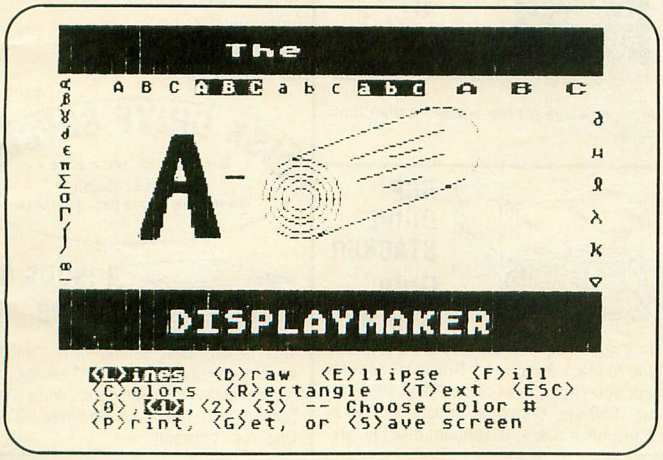
THE ALOG PAGEWRITER

TURNS YOUR ATARI AND 80 COLUMN PRINTER INTO AN EASY-TO-USE ELECTRONIC TYPEWRITER. THE ENTIRE PAGE LAYOUT IS DISPLAYED GRAPHICALLY WHILE YOU TYPE. IDEAL FOR SIMPLE WORD PROCESSING TASKS.



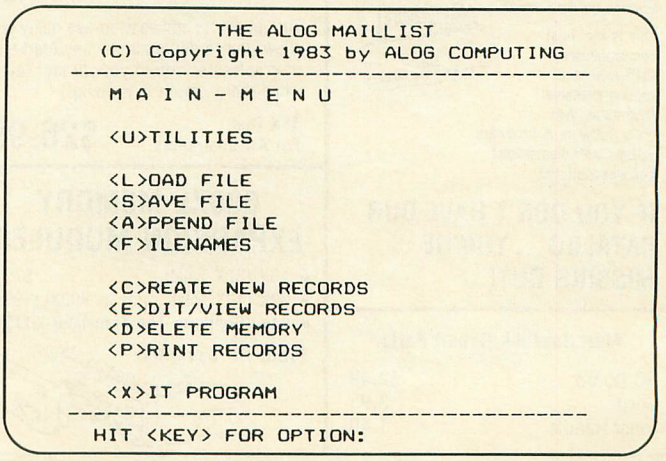
THE ALOG DISPLAYMAKER

MIXES COLOR GRAPHICS AND THREE TEXT SIZES ON THE SAME SCREEN. BUILT IN CHARACTER EDITOR AND SHAPE DRAWING ROUTINES. GRAPHICS DUMP TO MANY PRINTERS. GREAT FOR MAKING GRAPHS, CHARTS, AND HAVING FUN.



THE ALOG MAILLIST

IS AN EASY-TO-USE, MENU-DRIVEN DATA BASE PROGRAM DESIGNED FOR MAILING LISTS AND PRINTING LABELS. FAST 3 KEY SORT AND MULTI-LEVEL SEARCH ON ANY LINES. PRINT FORM LABELS 1 TO 2 INCHES HIGH.



All programs are only $39.95 each including tax. To order direct send check or money order to **Alog Computing, 1040 Veronica Springs Road, Santa Barbara, CA 93105.** We pay shipping. For information, credit card orders or C.O.D., call our distributor: **COMSTAR (805) 964-4660.**

Dealer and distributor inquiries invited.

(ATARI is a trademark of Atari, Inc.)

CIRCLE #111 ON READER SERVICE CARD

FOR BUSINESS OR PLEASURE . . .

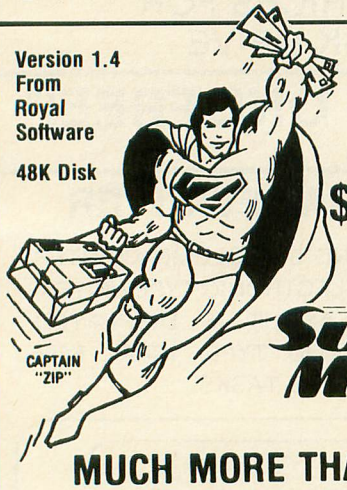
ATARI®

COMPUTER PALACE

WE KNOW ATARI!

Version 1.4 From Royal Software

48K Disk



Only $49.95

Now with built-in Printer Driver for compatibility with any printer.

Single and double density versions available, specify when ordering.

Specify Drive Model and Density When Ordering

Super Mailer PLUS

MUCH MORE THAN A MAILING LIST!

Super Mailer Plus is a versatile mini-data base that is ideal for business or home. Its many user-friendly features include: Multiple Searches on Any Field, Special Code Field, Single or Double Density (specify), Store Up to 1100 Records Per Disk (DD), Supports Up to 4 Drives, Create Sub-Files, Auto or Manual Delete of Duplicates, Print Labels 1, 2, or 3 Across, Print File Copy in Any Format, Sorts on Any Field, Built-In State Abbreviation Table, Built-In Printer Driver For Easy Set Up, More . . .

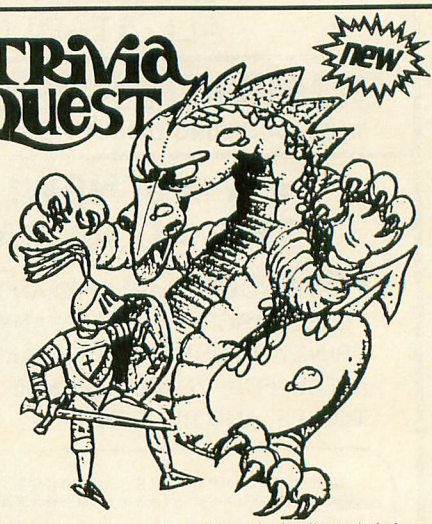
Outsmart Your Friends . . .

Outwit The Dragon . . .

Join The Quest!

Trivia Quest

new



From Royal Software

$39.95

(Program covers 4 Disk Sides)

- Over 2000 Trivia Questions
- For 1 to 4 Players
- Difficulty Handicap Options
- Challenging Action
- Survival of the Smartest!

Here is the perfect program for the whole family . . . It combines the intellectual challenge of the popular trivia games with exciting, playable action sequences to provide many hours of enjoyment. Additional Trivia Data Disks will be available as well as a Utility Program to create your own trivia questions.

● **Utility Program with 1000 Additional Questions** **$24.95**

Disk Management Data Base

$24.95

From Royal Software, 32K Disk

Organize and categorize your disk files with this easy to use database. It features super-fast Searches and Sorts, and will print File Copies and Disk Labels. The program supports 2 Drives. The more you use your computer . . . the more you will enjoy the DISKEEPER!

You may buy any one of these $9.95 specials for only 99¢ with a purchase of $50 or more from our over 3000-item inventory, or buy any three at $9.95 each and get the fourth for only 99¢. Offer Valid 9/1/84 to 10/31/84 (Limited Quantities, give 2nd choices. Specify computer model when ordering.)

BIG 99¢ SALE

$9.95 SPECIALS

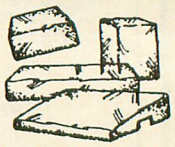
Protector II (T)	Chicken (D)	Preppie (T)
Claim Jumper (D)	Chicken (T)	Pathfinder (D)
Shamus (D)	Chicken (C)	Meteor Storm (D)
Thrax Lair (T)	Match Racer (D)	Cyborg (D)
Pharaoh's Pyramid (D)	Match Racer (T)	Wizard of Wor (D)
Frogger (D)	Moonbase 10 (T)	Deluxe Invaders (D)
Frogger (T)	Boulders & Bombs (C)	Vocabulary Bldr 1 (T)
Repton (D)	Mountain King (C)	Vocabulary Bldr. 2 (D)
Wayout (D)	Kid Grid (T)	Sammy the Seaserp't (T)

MISCELLANEOUS SPECIALS

32K Axlon RAM	$44.90	Deluxe Invaders (C)	17.95
Wizard of Wor (C)	17.95	Pool 400 (C)	14.95
		Frogger (C)	14.95

Protect Your Equipment!

DUST COVERS



Available For: ATARI 400, 800, 600XL, 800XL, 1200XL, 410, 810, 1050, 1025, 1027, EPSON FX-80, RX-80, MX-80, GEMINI 10X, PROWRITER 8510, PERCOM, RANA 1000, INDUS GT.

Only **$8.95** EACH

Additional Covers Ordered at Same Time ONLY **$7.95** each

FREE CATALOG

with any order . . . or send $1 (refundable with first purchase).

This is the most comprehensive Atari reference catalog available! It contains over 3000 software & hardware listings with illustrations and descriptions!

IF YOU DON'T HAVE OUR CATALOG . . . YOU'RE MISSING OUT!

The Ultimate Screen Dump Program

This powerful and easy-to-use utility will allow you to dump almost any Atari text or graphics screen to your printer (even while the program is running!)

48K Disk For All Computers **$26.95**

EXODUS: ULTIMA III

48K Disk $39.95

A Super Value for the newest in the Ultima Series.

DRIVES—DRIVES—DRIVES

TRAK AT-D2 (DD, Ptr., Port)	$398
TRAK AT-D4 (DD/DS, Ptr. Port)	529
TRAK 352 (DD/DS)	329
TRAK CHAMP (DD)	339
TRAK AT-S1 (DD Slave)	289
ATARI 1050 (1½ D)	Call
RANA 1000 (DD)	349
PERCOM 88-S1PD (DD, Ptr. Port)	369
RCP 810 Drive (SSSD)	285

PLUS FRT.

DISK DRIVE SPECIAL

Buy any drive and receive a $50 Gift Certificate (Redeemable after Dec. 25, 1984)

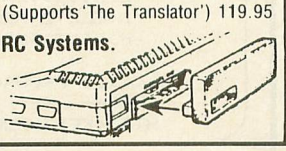
Atari Joystick Repair Parts

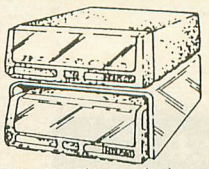
PC Board	$2.49
Cord	2.95
Inner Handle	1.49

600XL MEMORY EXPANSION MODULES

- 16K (32K Total) $79.95
- 32K (48K Total, same as 800XL) 99.95
- 64K (Supports 'The Translator') 119.95

From RC Systems.





RCP DRIVE STACKER

Only $19.95

This attractive smoked-acrylic stand allows you to stack 2 Indus GT Drives for convenient operation (also fits the RANA 1000, Atari 1050 and Concorde drives), or use as stand for modem, telephone, disk file, etc.

INDUS GT

$398 + FRT

This is our best selling drive. It offers double density, is quiet and reliable, and comes with Free Software (Word Proc., Spreadsheet, Database). Includes DOS-XL. One-Year Warranty.

COMPUTER PALACE

OPEN M-F, 9-6 Sat. 10-4 (Pacific Time)

2160 W. 11th Avenue Eugene, Oregon 97402

MasterCard VISA AMERICAN EXPRESS

USE YOUR CREDIT CARD & CALL

Toll Free 1-800-452-8013

★ ORDERS ONLY, PLEASE ★

There's never a penalty for using your credit card!

For Information, Call (503) 683-5361

SHIPPING INFO: Minimum $2.90 Ground, $4.75 Air. Actual Cost depends on weight. Call (503) 683-5361 for information. **WARRANTY INFO:** Everything that we sell is warrantied by the manufacturer. If any item purchased from us fails to perform properly when you receive it, call us at (503) 683-5361 so that we can assist you. No returned merchandise accepted without authorization. Defective software will be replaced with another copy of the same program, otherwise, no software is returnable.

CIRCLE #112 ON READER SERVICE CARD

Atari 800XL . . . . . Call
1010 Program Recorder 77.00
1027 Letter Quality Printer . . . . Call
1050 Disk Drive . . . . . Call
Atari 850 Interface . . . . . Call
Atari Light Pen 82.95
Numerical Keypad 104.95

PRINTERS

Alphacom 40C w/Interface . . . 99.95
Alphacom 80C w/Interface . . 189.95
Axiom AT-550 . 329.00
Epson . . . . . Call
Gemini 10X . . 269.00
Okidata . . . . . Call
Prowriter I . . . . . Call
Riteman . . . . . Call

KOALA

Koala Pad-D 69.95
Koala Pad-Cart 74.95

ATARI

ATARI is a trademark of ATARI, INC.

DISK DRIVES

Indus GT . . . . . Call
Percom . . . . . Call
Rana 1000 . . . . . Call
Trak . . . . . Call

MONITORS

AMDEK Color 1+ . . . . Call
USI 12" Amber 149.00
USI 12" Green 139.00
USI 14" Color 269.00
BMC 12" Color 229.00
BMC 12" Amber 99.00
BMC 12" Green 95.00
Phillips 12" Green w/Sound 129.00
Phillips 12" Amber w/Sound 139.00

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
Novation . . . . . Call
R-Verter Modem Adaptor 39.95

INTERFACES

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

MOSAIC

48K RAM 94.00
64K RAM/400 . . . 149.00
64K RAM/800 + Cable Kit #1 . . . 169.00
Handyman-D 23.95
Cable Kit #2 . . . . . 14.95
Superdrive-D 23.95

Triangle Replacement Keyboard for 400 . 54.95

Alien Voice Box II . 99.95

Expansion Memory for 600XL . . . . . Call

Back Chair . . . 79.95
(Ergonomic Computer Chair)

☆ *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

ENTERTAINMENT

Caverns of Mars-Cart 32.95
Donkey Kong-Cart 37.95
Donkey Kong Jr.-Cart 39.95
Eastern Front-Cart 32.95
Joust-Cart 39.95
Jungle Hunt-Cart 35.95
Mario Bros.-Cart 35.95

ENTERTAINMENT (cont'd.)

Ms. Pac-Man-Cart 39.95
Pengo-Cart 35.95
Pole Position-Cart 39.95
Qix-Cart 32.95
Realsports Football-Cart 35.95
Robotron: 2084-Cart 35.95
Star Raiders-Cart 32.95
Tennis-Cart 35.95

HOME MANAGEMENT

Atariwriter-Cart 74.95
Family Finances-D 37.95
Home Filing Manager-D 37.95
Mailing List-T 19.95
Syncalc-D 74.95
Synfile-D 74.95
Syntrend-D 74.95

LEARNING

Logo-Cart 79.95
My First Alphabet-D 26.95

PERSONAL DEVELOPMENT

Atarimusic I or II-D 29.95
Conversational French-T 44.95
Conversational German-T 44.95
Conversational Italian-T 44.95

PERSONAL DEVEL. (cont'd.)

Conversational Spanish-T 44.95
Paint-D 33.95

PROGRAMMING

Microsoft Basic II-Cart 67.95
Atari Macro-Assembler-Cart . . 67.95
Atari Basic-Cart 41.95
Assembler Editor-Cart 46.95

BOOKS

Analog Compendium 9.95
Atari Assembler 14.95
Basic Reference Manual 9.95
Book of Atari Software 84 . . . . 17.95
Compute's 1st Book of Atari . . 12.95
Compute's 1st Book/Games . . 12.95
Compute's 1st Book/Graphics . . 12.95
Compute's 2nd Book of Atari . . 12.95
Compute's 2nd Book/Graphics . 12.95
Compute's 3rd Book of Atari . . 12.95
Compute's Basic Source Book . 12.95
Compute's Machine Lang/Beg . 14.95
Inside Atari DOS 19.95
Mapping the Atari 14.95
Technical Reference Notes 27.95
Your Atari Computer 16.95

BRODERBUND

AE-D 23.95
Arcade Machine-D 41.95
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

DATASOFT

Basic Compiler-D 55.95
Bruce Lee-D/T 23.95
Dallas-D 23.95
Heathcliff-D/T 23.95
Letter Wizard . . . . . Call
Micropainter-D 23.95
Money Wizard . . . . . Call
Nibbler-D/T 20.95
Spell Wizard-D 34.95
Teletalk-D 34.95
Zaxxon-D/T 27.95

EPYX

Dragons/Pern-D/T 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-Cart 27.95
Puzzlemania . . . . . Call
Summer Games . . . . . Call
Temple of Apshai-D/T 27.95

INFOCOM

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

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
NATO Commander-D/T 23.95
Solo Flight-D/T 23.95
Spitfire Ace-D/T 23.95
Wingman-D/T 23.95

Hundreds of items available for the Atari . . . please call.

MISCELLANEOUS ATARI

Omnimon 82.95
Pitfall-Cart 31.95
River Raid-Cart 31.95
Diskey-D 34.95
Ultra Disassembler-D 34.95
Zombies-D/T 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
James Bond-Cart 34.95
Popeye-Cart 34.95
Q-Bert-Cart 34.95
Star Wars-Cart 34.95
Return of Heracles-D 26.50
Master Type-D/Cart 27.95
Songwriter-D 27.95
Flight Simulator II-D 37.95
Chatterbee-D 27.95
S.A.M.-D 41.95
Castle Wolfenstein-D 20.95
Compuserve Starter Kit 34.95
Home Accountant-D 52.95
Ken Uston's Blackjack-D 49.95
Megafont-D 19.95
Monkey Wrench II-Cart 49.95
Movie Maker-D 41.95
Pogo Joe-D 20.95
Spelunker-D 27.95
Ultima III-D 41.95
Jupiter Mission-D 34.95
Dreadnaught Factor-Cart 31.95

MISC. ATARI (cont'd.)

Keystone Kapers-Cart 31.95
Antica 4-D 37.95
Boulder Dash-D/T 20.95
Scraper Caper-Cart 34.95
Miner 2049'er-Cart 34.95

OPTIMIZED SYSTEMS

Action-Cart 74.95
Basic XL-Cart 74.95
MAC/65-Cart 74.95
DOS XL-D 27.95
Action Aid-D 27.95

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 Creature-Cart 27.95
Aerobics-D 30.95
Ageian Voyage-Cart 27.95
Alphabet Zoo-Cart 23.95
Cosmic Life-Cart 23.95
Delta Drawing-Cart 27.95
Grandma's House-D 23.95
Jukebox-Cart 27.95
Kids on Keys-Cart 23.95

SPINNAKER (cont'd.)

Kindercomp-Cart 20.95
Ranch-Cart 27.95
Search/Amazing Thing-D 27.95
Snooper -1 or -2-D 30.95
Trains-D 27.95
Alf in the Color Caves-Cart . . . 27.95
Bubble Burst-Cart 27.95
Kidwriter-D 23.95

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

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 . . . Call
Shamus Case II-D/T 23.95
Zepplin-D/T 23.95

To Order Call Toll Free
800-558-0003

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

D-Disk T-Cassette
Cart-Cartridge

New Lower Prices

Computability

NO SURCHARGE FOR MASTERCARD OR VISA

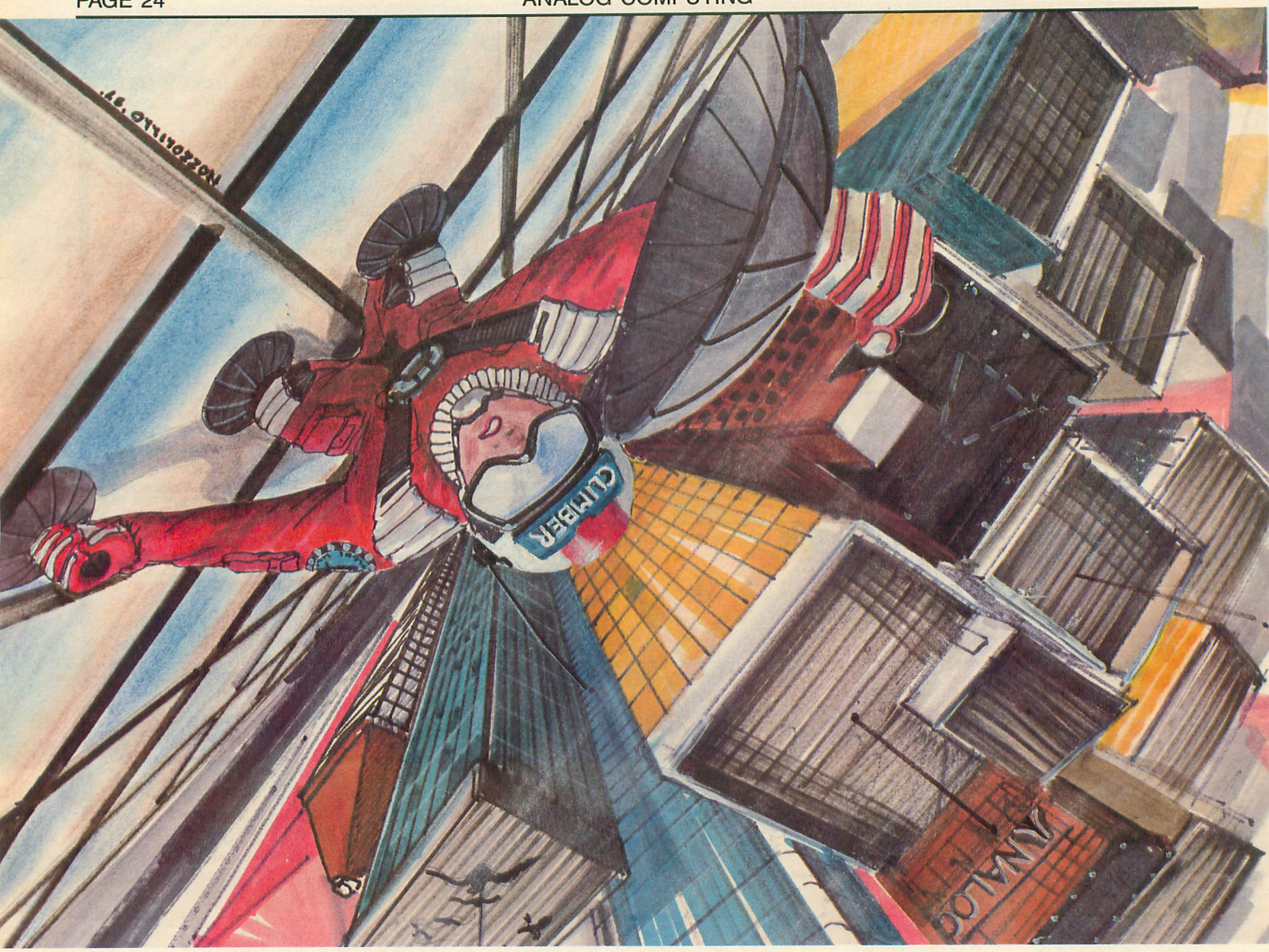
New Products

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 #113 ON READER SERVICE CARD



Climber

16K Cassette or 24K Disk

by John Hanke

Climber is an arcade game for one player. It requires 16K bytes of memory and a joystick plugged into port 0.

The object of the game is to guide your on-screen counterpart to the top of a fifteen-story building. The building is divided into five screens of three platforms each. A ladder joins each platform to the next. In order to climb the ladder and advance to the next platform, you must first acquire the "key." To get the key, simply maneuver under it and press the "fire" button. Then move to the ladder and push the joystick "up." You, unfortunately, are not the only inhabitant of the building. . .the munchkins are out there! The touch of these genetically-mutated beasts is deadly. They blindly roam the building in search of intruders. The only way to maneuver around the building is to master the art of jumping over them. Jumping is accomplished by pressing the "fire" button. You must be moving towards the munchkin, and the jump must be carefully timed in order to clear it. On levels three and upward, yet another factor hampers your ascent. . . sections of the platform have

decayed, leaving potentially deadly holes. These, too, must be jumped over. You are allotted three lives to begin with and awarded a bonus life every third screen. Watch out — the munchkins can be very tricky; you may find it beneficial to stay at the left edge of the screen and watch the patterns they follow, in order to play your moves. Good luck!

My thanks go to Tom Hudson for his excellent P/M mover subroutine (**ANALOG** issue 11). Without it, this program could not have been written. The munchkins are interesting examples of BASIC's limitations when writing an arcade-style game. Players 1-3 each contain two definitions. (Remember, a player is taller than the screen.) Therefore, two munchkins always share the same horizontal position, since they are, in fact, the same player. This provides two pursuers on each of three platforms, for a total of six. Using this method allows the program to simulate control of six objects with the same speed it would normally take to control three. Display list interrupts could have been used to change the players' horizontal position, but this would have bogged BASIC down with more numbers. Another step taken to conserve speed was to make the munchkins unintelligent. They move in a predictable pattern stored in three arrays. This requires minimal computation time in the game loop. REMs were not included in the program to conserve memory, so a brief summary of the main program segments and variables follows. □

Line 5 — Branch to initialization.

Lines 10 - 90 — This is the main game loop. Read stick, move players and check collisions.

Lines 5000 - 5040 — This is the jumping routine. Most functions of the main loop (check collisions, move munchkins, etc.) are duplicated here to increase speed.

Lines 6000 - 6070 — This is the ladder-climbing routine. Most functions of main loop duplicated here, also.

Lines 7000 - 7010 — Routine to handle falling off platform.

Lines 7500 - 7530 — This section handles special effects and branches to reset screen when player is killed.

Lines 8000 - 8010 — Control completion of a level and advance to the next.

Lines 10000 - 10030 — Begin initialization. Reserve memory for P/M tables and character set. Displays title screen.

Line 10120 — Reads in Tom Hudson's P/M mover routine.

Lines 10130 - 10185 — Read player data into strings.

Lines 10186 - 10188 — Poke new character definitions into memory.

Lines 10220 - 10225 — Initialize display list.

Lines 10250 - 10270 — Poke missile data (keys) into P/M memory.

Line 10280 — Sets up counters for a new game. Change LEVEL here to start on a level higher than 1. Change L to increase number of lives.

Lines 10290 - 10320 — Put proper munchkin for each level into working strings.

Line 10400 — Sets P/M colors.

Lines 10410 - 10570 — Print playfield and randomly place ladders and keys.

Lines 10580 - 10635 — Randomly place holes in platforms on higher levels. Check for interference with ladders.

Lines 10640 - 10660 — Read munchkin movement data and starting positions into AX1, AX2 and AX3 arrays.

Lines 10670 - 10680 — Flash player 0 onto screen and check for game in progress.

Lines 11000 - 11020 — Scroll intro message onto screen and check for trigger.

Lines 20000 - 20140 — Player shape data.

Lines 20150 - 20170 — Ladder and platform character definitions.

Lines 22000 - 22020 — Missile (key) data.

Lines 25000 - 25040 — Movement data for munchkins.

Variable table.

Variable — Usage

S — Used to read joystick.

X, Y — Player 0's horizontal and vertical position.

C1,M — Counters used for munchkin movement.

X1, X2, X3 — Munchkin in horizontal positions.

AX1(), AX2(), AX3() — Values used to increment munchkin in horizontal position.

LEVEL — Current difficulty level.

L — Lives remaining.

K — Key flag. 1 if player has key; 0 if not.

Z — Current platform.

G — Game in progress flag.

CBASE, CB — Character set address in pages and full.

PMBASE,PMB — P/M memory address in pages and full.

MOVE — Address of P/M mover routine.

F() — Addresses of player data strings MA$-MF$.

P3, P4, P5 — Addresses of munchkin data strings PA$-PC$.

LP() — Position of ladders.

B,D,C,R — General purpose variables.

MA$, MB$, MC$, MD$, ME$, MF$ — Definitions for player 0.

P3$, P4$, P5$ — Definitions for munchkins.

PA$, PB$, PC$ — Definitions for munchkins (temporary).

MOVE$ — Holds Tom Hudson's P/M mover routine.

BASIC listing.

```
5 GOTO 10000
10 S=STICK(0):IF S<>15 THEN S1=S
15 X=X+((S=7)-(S=11))*2:B=B(1:A=USR(MO
VE,0,PMB,F(B+1+(S1=11)*2),X,Y,15)
20 SOUND 0,0,0,15:SOUND 0,0,0,0:IF PEE
K(53252)=0 THEN 7000
30 POKE 53278,0:IF STRIG(0)=0 THEN GOS
UB 5000
40 IF S=14 AND (X-48)/4=LP(Z) AND K=1
THEN K=0:GOSUB 6000
50 C1=C1+1:IF C1>39 THEN C1=0:M=M+1:IF
 M>4 THEN M=1
60 X1=X1+AX1(M):X2=X2+AX2(M):X3=X3+AX3
(M):POKE 53249,X1:POKE 53250,X2:POKE 5
3251,X3
70 IF PEEK(53260)>0 THEN 7500
90 GOTO 10
5000 I=((S=7)-(S=11))*2
5005 FOR C=1 TO 8:Y=Y-((C<3)-(C>6))*5:
X=X+I:A=USR(MOVE,0,PMB,F(B+1+(S=11)*2)
,X,Y,14)
5010 C1=C1+1:IF C1>39 THEN C1=0:M=M+1:
IF M>4 THEN M=1
5020 X1=X1+AX1(M):X2=X2+AX2(M):X3=X3+A
X3(M):POKE 53249,X1:POKE 53250,X2:POKE
 53251,X3
5025 IF PEEK(53256+Z)>0 AND K=0 THEN 5
OUND 0,40,10,6:POKE 53252+Z,0:K=1:SCRE
=SCRE+500
5030 SOUND 0,Y,10,6*(C<4):IF PEEK(5326
0)>0 THEN POP :GOTO 7500
5040 NEXT C:RETURN
6000 POKE 77,0:Y1=Y:IF Z=1 THEN POP :G
OTO 8000
6010 S=STICK(0):Y1=Y1+((S=13 AND Y1<Y-
1)-(S=14))*2:A=USR(MOVE,0,PMB,F(B+5),X
,Y1,14)
6020 C1=C1+1:IF C1>39 THEN C1=0:M=M+1:
IF M>4 THEN M=1
6030 SOUND 0,0,0,0:X1=X1+AX1(M):X2=X2+
AX2(M):X3=X3+AX3(M):POKE 53249,X1:POKE
 53250,X2:POKE 53251,X3
6040 IF PEEK(53260)>0 THEN POP :GOTO 7
500
6050 IF Y1=Y-28 THEN Y=Y1:Z=Z-1:RETURN
6060 IF S=13 OR S=14 THEN B=B(1:SOUND
0,Y1*2,10,6:SOUND 0,0,0,0
6070 GOTO 6010
7000 FOR C=Y TO 128:B=B(1:SOUND 0,C,10
,6:A=USR(MOVE,0,PMB,F(B+5),X,C,14):NEX
T C
7010 FOR C=255 TO 0 STEP -10:SOUND 0,C
,4,6:SOUND 1,255-C,6,6:NEXT C:SOUND 0,
0,0,0:SOUND 1,0,0,0
7500 FOR C=0 TO 255 STEP 5:POKE 704,C:
SOUND 0,100,12,6:SOUND 1,99,12,B*6:SOU
ND 2,C,2,6:B=B(1:NEXT C
7510 POKE 53248,0:POKE 53278,0:L=L-1:S
OUND 0,0,0,0:SOUND 1,0,0,0:SOUND 2,0,0
,0:IF L<>0 THEN 8010
7520 POSITION 2,23:? "   GaME oVEr
 ";:POSITION 9,0:? SCRE;:IF SCRE>HIGH
THEN HIGH=SCRE
```

```
7530 FOR C=1 TO 300:NEXT C:GOTO 10280
8000 FOR C=Y TO Y-12 STEP -2:SOUND 0,0
,0,0:A=USR(MOVE,0,PMB,F(B+5),X,C,14):B
=B(1:SOUND 0,Y*2,10,8:NEXT C:Y=C
8005 SOUND 0,0,0,0:SCRE=SCRE+1000:LEVE
L=LEVEL+1-(LEVEL)4)*5:IF LEVEL/3=INT(L
EVEL/3) THEN L=L+1
8010 POKE 53248,0:POKE 53249,0:POKE 53
250,0:POKE 53251,0:GOTO 10290
10000 CBASE=PEEK(106)-8:PMBASE=CBASE-4
:POKE 106,PMBASE:PMB=PMBASE*256:CB=CBA
SE*256:POKE 54279,PMBASE:GRAPHICS 0
10010 GRAPHICS 2+16:POSITION 6,1:? #6;
"CLIMBER":POSITION 3,4:? #6;"BY JOHN
HANKE"
10020 POSITION 0,5:? #6;"FOR ANALOG CO
MPUTING":POSITION 4,9:? #6;"initializi
ng"
10030 FOR C=0 TO 511:POKE CB+C,PEEK(57
344+C):NEXT C
10110 DIM MA$(15),MB$(15),MC$(15),MD$(
15),ME$(14),MF$(14),P3$(8),P4$(8),P5$(
8),PA$(8),PB$(8),PC$(8)
10115 DIM AX1(4),AX2(4),AX3(4),F(6),LA
D$(40),LP(4),M5$(120),PMOV$(100)
10120 FOR C=1 TO 100:READ D:PMOV$(C,C)
=CHR$(D):NEXT C
10130 FOR C=1 TO 15:READ D:MA$(C,C)=CH
R$(D):NEXT C
10140 FOR C=1 TO 15:READ D:MB$(C,C)=CH
R$(D):NEXT C
10150 FOR C=1 TO 15:READ D:MC$(C,C)=CH
R$(D):NEXT C
10160 FOR C=1 TO 15:READ D:MD$(C,C)=CH
R$(D):NEXT C
10162 FOR C=1 TO 14:READ D:ME$(C,C)=CH
R$(D):NEXT C
10166 FOR C=1 TO 14:READ D:MF$(C,C)=CH
R$(D):NEXT C
10170 FOR C=1 TO 8:READ D:P3$(C,C)=CHR
$(D):NEXT C
10180 FOR C=1 TO 8:READ D:P4$(C,C)=CHR
$(D):NEXT C
10185 FOR C=1 TO 8:READ D:P5$(C,C)=CHR
$(D):NEXT C
10186 FOR C=0 TO 7:READ D:POKE 64*8+C+
CB,D:NEXT C
10187 FOR C=0 TO 7:READ D:POKE 65*8+C+
CB,D:NEXT C
10188 FOR C=0 TO 7:READ D:POKE 66*8+C+
CB,D:NEXT C
10190 FOR C=21 TO 25:READ D:LAD$(C,C)=
CHR$(D):NEXT C:FOR C=1 TO 16 STEP 5:LA
D$(C,C+4)=LAD$(21,25):NEXT C
10200 LAD$(26,27)=LAD$(21,22)
10210 M5$="
  -BY JOHN HANKE      PRESS fire  Climber
TO STAR
T                    "
10220 GRAPHICS 0:START=PEEK(560)+256*P
EEK(561):POKE START+28,6
10225 FOR C=START+6 TO START+27:POKE C
,4:NEXT C:POKE 756,CBASE:SETCOLOR 2,8,
10
10230 POKE 559,46:POKE 53277,3:POKE 53
275,2
10240 MOVE=ADR(PMOV$):F(1)=ADR(MA$):F(
2)=ADR(MB$):F(3)=ADR(MC$):F(4)=ADR(MD$
):F(5)=ADR(ME$):F(6)=ADR(MF$)
10250 FOR C=PMB+397+4*3 TO PMB+397+4*3
+7:READ D:POKE C,D:NEXT C
10260 FOR C=PMB+397+4*10 TO PMB+397+4*
10+7:READ D:POKE C,D:NEXT C
10270 FOR C=PMB+397+4*17 TO PMB+397+4*
17+7:READ D:POKE C,D:NEXT C
10280 LEVEL=1:SCRE=0:L=3:G=0
10290 IF LEVEL=1 THEN PA$=P3$:PB$=P3$:
PC$=P3$
10300 IF LEVEL=2 THEN PA$=P4$:PB$=P4$:
PC$=P4$
10310 IF LEVEL=3 THEN PA$=P3$:PB$=P4$:
PC$=P5$
10315 IF LEVEL)3 THEN PA$=P5$:PB$=P5$:
PC$=P5$
10320 P3=ADR(PA$):P4=ADR(PB$):P5=ADR(P
C$)
10330 A=USR(MOVE,1,PMB,P5,0,40,8):FOR
C=1 TO 8:POKE C+PMB+655+4*13,ASC(PB$(C
,C)):NEXT C
10340 A=USR(MOVE,2,PMB,P4,0,68,8):FOR
C=1 TO 8:POKE C+PMB+783+4*20,ASC(PA$(C
,C)):NEXT C
10350 A=USR(MOVE,3,PMB,P5,0,40,8):FOR
C=1 TO 8:POKE C+PMB+911+4*20,ASC(PA$(C
,C)):NEXT C
10400 POKE 705,102:POKE 706,120:POKE 7
07,200
10410 ? "K":POKE 752,1:POSITION 2,0:?
"SCORE: ";SCRE;:POSITION 17,0:? "MEN:
";L
10420 POSITION 29,0:? "HIGH: ";HIGH
10500 FOR C=1 TO 22 STEP 7:POSITION 0,
C:FOR D=1 TO 40:? CHR$(2);:NEXT D:NEXT
C
10505 POSITION 0,23:? "  Climber   LEV
EL:";LEVEL;:SETCOLOR 1,LEVEL*2+2,4
10510 R=INT(RND(0)*30+4):LP(1)=R:POSIT
ION R,2:? LAD$;
10520 FOR C=2 TO 3
10530 R=INT(RND(0)*30+4):IF ABS(R-(LP(
C-1)))(8 THEN 10530
10540 POSITION R,(C-1)*7+2:? LAD$;:LP(
C)=R:NEXT C
10550 FOR C=1 TO 3
10560 R=INT(RND(0)*30+4):IF ABS(R-LP(C
))(8 THEN 10560
10570 POKE 53252+C,48+R*4:NEXT C
10580 IF LEVEL(3 THEN 10640
10590 FOR C=4 TO 36 STEP 6:IF ABS(C-LP
(1))(4 OR ABS(C-LP(2))(4 OR RND(0))(LE
VEL-2)/10+0.5 THEN B=1
10600 IF B=0 THEN POSITION C,8:? "   "
;
10605 B=0:NEXT C
10610 IF LEVEL(4 THEN 10640
10620 FOR C=2 TO 3
10630 FOR C1=4 TO 36 STEP 6:IF ABS(C1-
LP(C))(4 OR ABS(C1-LP(C+(C=2)))(4 OR R
ND(0))(LEVEL-2)/10+0.5 THEN B=1
10635 IF B=0 THEN POSITION C1,C*7+1:?
"   ";
10637 B=0:NEXT C1:NEXT C
10640 RESTORE 25000+(LEVEL-1)*10:M=1:C
1=0:X=54:Y=90:Z=3:K=0:READ X1:X1=X1*4+
48:FOR C=1 TO 4:READ D:AX1(C)=D:NEXT C
10650 READ X2:X2=X2*4+48:FOR C=1 TO 4:
READ D:AX2(C)=D:NEXT C
10660 READ X3:X3=X3*4+48:FOR C=1 TO 4:
READ D:AX3(C)=D:NEXT C
10670 A=USR(MOVE,0,PMB,F(1),X,Y,15):FO
R C=0 TO 255 STEP 5:B=B(1:POKE 704,0:S
OUND 0,C,8,6*B:POKE 704,26:NEXT C
10680 SOUND 0,0,0,0:IF G)0 THEN 10
11000 FOR C=1 TO LEN(M5$)-20:POSITION
0,23:? M5$(C,C+20);
11010 IF STRIG(0)=0 THEN POSITION 0,23
:? "  Climber   LEVEL:";LEVEL;" ";:G=1
:C1=0:GOTO 10
11020 FOR C1=1 TO 10:NEXT C1:NEXT C:GO
TO 11000
19000 DATA 216,104,104,104,133,213,104
,24,105,2,133,206,104,133,205,104,133,
204,104,133,203,104,104,133,208
19010 DATA 104,104,133,209,104,104,24,
101,209,133,207,166,213,240,16,165,205
,24,105,128,133,205,165,206,105
19020 DATA 0,133,206,202,208,240,160,0
,162,0,196,209,144,19,196,207,176,15,1
32,212,138,168,177,203,164
19030 DATA 212,145,205,232,169,0,240,4
,169,0,145,205,200,192,128,208,224,166
,213,165,208,157,0,208,96
20000 DATA 56,60,56,56,16,254,56,56
20010 DATA 56,60,36,36,38,48,8
20020 DATA 56,60,56,56,16,254,56,56
20030 DATA 56,60,36,36,52,6,8
20040 DATA 28,60,28,28,8,127,28,28
20050 DATA 60,60,36,36,100,12,8
20060 DATA 28,60,28,28,8,127,28,28
20070 DATA 60,60,36,36,44,96,8
20080 DATA 92,92,93,93,73,127,28,28
20090 DATA 28,28,34,34,98,3
20100 DATA 29,29,93,93,73,127,28,28
20110 DATA 28,28,34,34,35,96
20120 DATA 60,126,219,219,126,60,36,10
2
```

```
20130 DATA 60,126,219,255,231,219,255,
170
20140 DATA 60,102,102,189,153,129,66,0
20150 DATA 240,240,240,255,255,240,240
,240
20160 DATA 15,15,15,255,255,15,15,15
20170 DATA 85,85,170,170,170,170,85,85
20180 DATA 128,129,29,126,126
22000 DATA 12,12,8,12,8,12,8,0
22010 DATA 48,48,32,48,32,48,32,0
22020 DATA 192,192,128,192,128,192,128
,0
25000 DATA 38,-2,1,-1,2,8,1,-1,1,-1,22
,-1,1,1,-1
25010 DATA 38,-2,-1,2,1,22,-1.5,1.5,1.
5,-1.5,38,-1,-2,1,2
25020 DATA 8,1,2,-1,-2,38,-2,-.5,2,.5,
38,-1,-1,2,0
25030 DATA 38,-1,-1,1,1,8,2,-1,1,-2,22
,-1,2,-2,1
25040 DATA 22,-1.5,-1,2,.5,22,1.5,-2,-
.5,1,22,-.5,1.5,1,-2
```

CHECKSUM DATA.

(see page 25)

```
5 DATA 619,84,347,101,602,55,265,176,4
02,617,555,40,288,563,417,5131
5030 DATA 397,859,961,885,291,12,965,1
71,496,730,985,366,828,282,835,9063
7530 DATA 830,303,207,993,349,116,535,
172,447,111,427,984,989,994,999,8456
10162 DATA 6,21,281,285,302,319,323,32
7,657,120,175,857,139,344,538,4694
10250 DATA 741,895,918,917,616,607,612
,633,962,432,447,433,101,325,580,9219
10500 DATA 662,620,731,251,611,369,255
,376,877,434,601,876,424,423,252,7762
10630 DATA 510,286,268,958,442,451,565
,281,963,882,938,750,786,524,304,8908
20000 DATA 746,321,750,118,881,563,885
,316,763,47,762,289,245,316,12,7014
20150 DATA 525,924,72,174,308,604,527,
956,882,53,971,84,6080
```

Mini comp

40K Disk

by David Bohlke

Minicomp is a compiler written in Atari BASIC that will translate a subset of Atari BASIC into 100% machine language. The **Minicomp** compiler, your BASIC source program, the Atari BASIC interpreter (cartridge or disk) and the compiled object code will all reside in memory at the same time. Hence, it's possible to write and debug your program in BASIC, then compile it without switching any disk or tape files. You will need 40K of memory to effectively utilize **Minicomp**.

Minicomp was designed originally as a graphics game compiler. It can compile just eleven different BASIC statements. But I think you'll find these statements, along with a working knowledge of the Atari Operating System, can be quite powerful in writing BASIC games that run with machine code speed. With PEEK and POKE you can utilize sound, color, joysticks, random, P/M graphics, display lists, timers, scrolling, data tables and other Operating System functions. In future articles, I will present and explain several full length games—written with only the eleven **Minicomp** statements—that you can compile, take apart, study and enhance.

Getting started.

First, you will need to key in the short program INIT96 (Listing 1). Be sure you SAVE this listing to disk or tape before you RUN it! The purpose of INIT96 is to move the low memory (LOMEM) pointer to page 96. After you RUN the program, you should have 15118 bytes of free memory remaining. The RAM below page 96 (24576) is now protected from BASIC and will be used to store the compiled object code. **Minicomp** will use page 48 (12288 dec—above DOS I) as a starting location for the object code, so our object code program can be around 12K in length (page 48 to page 96).

Now you can key in the BASIC code for **Minicomp** (Listing 2). This compiler program is about 5K in length and will reside, along with our BASIC source program, above the page 96 boundary. The **Minicomp** compiler, and future BASIC source programs, should be LISTed to disk or tape. This will make it possible to "chain" these programs together in memory later on. I won't attempt to describe **Minicomp**'s design at this time. If there is sufficient interest, though, I'll give a line-by-line explanation in a future article.

Using Minicomp.

It's time for a quick demonstration of **Minicomp**'s potential. During each session using **Minicomp**, it will be necessary to begin as follows: (1) Power up your Atari; (2) LOAD and RUN the INIT96 program to adjust LOMEM; (3) ENTER the **Minicomp** compiler; and (4) ENTER or type in your BASIC source program.

In this case, after you LOAD and RUN the INIT96 program and ENTER the **Minicomp** compiler, you may type in the Speed Demo (Listing 3) program. Now, type *RUN* and press RETURN to get an idea of the speed of BASIC. This Speed Demo program will display all 255 character codes on a GR.0 screen. You can press BREAK at any time to exit this run.

To invoke **Minicomp** and compile this short program, type *GOTO 1000* (RETURN). **Minicomp** will compile every line number from 1 to 999 into 100% 6502 machine code. During the first pass of the compiler process, **Minicomp**'s top line should read:

```
4 DATA 112,696,888
```

The first number is the current location in the object code buffer, beginning at 12277. At the end of compile, it will be the highest memory location used by the object code buffer. The second number is the current line number being compiled, and the third number is the number of lines compiled thus far. Lines 2-24 on the screen denote the LSB/MSB decimal for the program line numbers, paired with each line's number location in the object code buffer. These values are used during the second pass of the compiler to adjust the GOTO and GOSUB locations. During this second pass, the word *JUMPS* will be displayed in the upper right corner of the screen, and the compiler's location in the object code buffer will print to the left of this message. **Minicomp** does a minimal amount of error checking, so be sure to use just the eleven commands in their restricted format when writing your own programs to be compiled. If **Minicomp** does catch an error, the word *ERROR* will display in the upper right hand corner of the screen, and the compile process will terminate.

Once the compile process is finished, the prompt *RETURN* will be printed in the upper right of the screen. To execute your machine language code, just press the RETURN key. I hope you're impressed with the speed of the compiled code—because that's what **Minicomp** is all about. Control will return to the BASIC mode at the end of the run, or you can press RESET at any time to exit. The compiled code can be RUN additional times by entering *GOTO 2000* then pressing RETURN twice.

The Speed Demo program listing is fairly easy to follow. Lines 10 and 20 find the beginning of display memory [simulates PEEK(89) *256+ PEEK(88)] and put this value into variable DS. Since **Minicomp** doesn't support multiplication, we need the short routine at Lines 900-904 to multiply the variables A and B, with the product being placed in C. Lines 100-130 will fill the screen with each of the 255 character codes in succession. The variable P will increment from 1 to 255. **Minicomp** cannot compile any print statements, so everything to be displayed must be POKEd to the screen memory. The variable A begins at DS and increments until it reaches the end of the GR.0 display memory, which is DS+960 (variable E). Notice how the **Minicomp** statement set is used strictly in its prescribed format. This requires some extra BASIC source program coding, but I hope you find the compiled machine language speed worth the effort.

Disk users can save the object file on disk by entering *DOS* and using the K command. I recommend using DOS I, as DOS II will overlay the object code area in larger applications. Make a note of the end of the object code buffer during compile. For the Speed Demo, this should be around 12700. Now, when you use the K command in DOS, the beginning (12288) and end (12700) of the object code buffer will need to be converted to hex. This object code buffer can then be appended with the starting address (12288), so the file will run after you load it using the L command. An alternative would be to load the object file, using the L command, then use the M command in DOS with a hex starting address of 3000 (12288 dec) to execute the object code.

Minicomp statement set

A=ccc	A=B
END	A=B+C
A=B-C	IF A=B THEN nnn
IF A<B THEN nnn	GOTO nnn
GOSUB nnn,RETURN	
A=PEEK(B)	POKE A,B

When you enter your BASIC code, you can use only the eleven statements given above and described below. These statements will function the same in their compiled form as they do in BASIC. The use of other statements may cause a compiler error message and will not allow **Minicomp** to compile accurately.

It is an expedient practice to write and debug your BASIC program and LIST it to disk or tape before you compile it.

In the following examples, A, B and C represent legal variable names. You can use up to seventy different variables in your program. Your entire BASIC source program must use line numbers 1-999. These are denoted by *nnn* in the examples below. Because of the conversion from BCD to LSB,MSB number representation, it is a good idea to avoid negative numbers. Positive constant integers will be referred to as *ccc*. Your BASIC source program can have up to 230 lines of code.

A=ccc or A=B

The left member of the assignment must be a variable name, and the right side can be a constant (positive integer) or a variable name. This is the only statement where constants can be compiled by **Minicomp**.

END

This statement will compile into a 6502 RTS

command. When you execute your compiled code, the END statement will return the computer to your control. If your machine code has an endless loop, you can return to BASIC by pressing RESET. It is always a good idea to have an END statement at Line 999, which is the last line **Minicomp** will compile.

A=B+C or A=B−C

Only addition and subtraction are supported, with just one operation in each statement. The operators must be variables, as constants will not compile. So, to add 8 and 25, you need to use: B=8: C=25: A=B+C. See the Speed Demo for a multiplication routine.

IF A=B THEN nnn or IF A<B THEN nnn

Only variables can be used in the comparisons —no constants. "Greater than" isn't supported, but you only need to switch the variables around to achieve it. The branch location must be a line number (1-999) and not a variable.

GOTO nnn or GOSUB nnn, RETURN

As with the compare statement, nnn must be a line number (1-999) and not a variable name.

A=PEEK(B) or POKE A,B

In both examples, A and B must be variable names. The value to be POKEd must be in the range 0-255.

That's all for now, folks!

I hope the Speed Demo program has convinced you of **Minicomp**'s main attribute—speed. In future articles, I'll present full length, arcade-style games which were written with **Minicomp**'s statement set, so that you'll be able to key-in and compile them to machine language. □

Listing 1.

```
4 GRAPHICS 0:FOR A=1600 TO 1614:READ B
:POKE A,B:? "96↓";:NEXT A:C=USR(1600)
5 DATA 169,96,141,232,2,169,0,141,231,
2,133,8,76,0,160
```

•

Listing 2.

```
1000 CLR :GRAPHICS 0:K1300=1300:K1400=
1400:K7500=7500:K1450=1450:K8310=8310:
K8315=8315:K8320=8320:B=12288
1010 A=PEEK(136)+256*PEEK(137):C=0:D=1
640:L1=PEEK(88)+256*PEEK(89)+40:DIM R(
3):R(0)=L1:POKE 752,1
1100 K=PEEK(A):K1=PEEK(A+1):POKE L1+C,
K:POKE L1+C+1,K1:IF PEEK(L1+C-1)<>K1 T
HEN R(K1)=L1+C
1102 LN=K+K1*256:IF LN>999 THEN 1500
1105 POKE 85,0:? "MINICOMP ";B;": ";LN
;"/";INT(C/2)+1;" *";
1110 POKE L1+C+460,B-INT(B/256)*256:PO
KE L1+C+461,INT(B/256):C=C+2:A=A+3
1200 GOSUB K1300:IF E=20 THEN A=A+1:GO
TO 1200
1210 IF E=22 THEN A=A+1:GOTO 1100
1220 IF E=54 THEN GOSUB 3000
1230 IF E=21 OR E=36 THEN P=96:GOSUB K
1400
1240 IF E=7 THEN GOSUB 4000
1250 IF E=10 THEN GOSUB K1300:GOSUB K1
300:GOSUB 8500:GOSUB 8360
1260 IF E=12 THEN GOSUB K1300:GOSUB K1
300:GOSUB 8500:P=32:GOSUB K1400:GOSUB
8365
1270 IF E=31 THEN GOSUB 6000
1290 GOTO 1200
1300 A=A+1:E=PEEK(A):RETURN
1400 POKE B,P:B=B+1:RETURN
1450 W=F+F+D:Y=INT(W/256):X=W-Y*256:RE
TURN
1500 POKE 77,0:POKE 85,32:? "JUMPS ";
:FOR J=12288 TO B:Z=PEEK(J):IF Z<>76 A
ND Z<>32 THEN NEXT J:GOTO 1610
1540 K1=PEEK(J+2):IF K1>3 THEN NEXT J:
GOTO 1610
1550 K=PEEK(J+1):FOR I=R(K1) TO C+L1-1
 STEP 2:IF PEEK(I)<>K THEN NEXT I:NEXT
 J:GOTO 1610
1560 IF PEEK(I+1)<>K1 THEN NEXT I:NEXT
 J:GOTO 1610
1570 POKE 85,26:? J;:POKE J+1,PEEK(I+4
60):POKE J+2,PEEK(I+461):J=J+2:NEXT J
1610 POKE 752,0:POKE 85,32:? "RETURN"
;
1620 IF PEEK(764)<>12 THEN 1620
2120 POKE 12287,104:X=USR(12287):END
3000 GOSUB K1300:GOSUB K7500:GOSUB K13
00:IF E<>45 THEN 8020
3020 GOSUB K1300:IF E=14 THEN 3200
3030 IF E=70 THEN 3400
3050 G=F:GOSUB K7500:H=PEEK(A+1):IF H=
22 OR H=20 THEN 3300
3070 GOTO 3600
3200 GOSUB K1300:GOSUB 8500:GOSUB K145
0:Z=L:GOSUB 8300:GOSUB K8310
3230 Z=M:GOSUB 8300:X=X+1:GOTO K8310
3300 H=F:GOSUB K1450:GOSUB K8320:F=G:G
OSUB K1450:GOSUB K8310
3330 F=H:GOSUB K1450:X=X+1:GOSUB K8320
:F=G:GOSUB K1450:X=X+1:GOTO K8310
3400 G=F:A=A+1:GOSUB K1300:GOSUB K7500
:H=F:A=A+1
3410 P=162:GOSUB K1400:P=0:GOSUB K1400
:F=H:GOSUB 8600
3420 P=161:GOSUB K1400:P=203:GOSUB K14
00
3440 F=G:GOSUB K1450:GOSUB K8310:Z=0:G
OSUB 8300:X=X+1:GOTO K8310
3600 H=F:GOSUB K1300:Z=E:GOSUB K1300:G
OSUB K7500:I=F:IF Z=38 THEN 3700
3608 IF Z<>37 THEN 8020
3610 P=24:GOSUB K1400:F=I:GOSUB K1450:
GOSUB K8320:F=H:GOSUB K1450:P=109:GOSU
B K8315
3630 F=G:GOSUB K1450:GOSUB K8310:F=I:G
OSUB K1450:X=X+1:GOSUB K8320
3640 F=H:GOSUB K1450:X=X+1:P=109:GOSUB
 K8315:F=G:GOSUB K1450:X=X+1:GOTO K831
0
3700 P=56:GOSUB K1400:F=H:GOSUB K1450:
GOSUB K8320:F=I:GOSUB K1450:P=237:GOSU
B K8315
3720 F=G:GOSUB K1450:GOSUB K8310:F=H:G
OSUB K1450:X=X+1:GOSUB K8320
3730 F=I:GOSUB K1450:X=X+1:P=237:GOSUB
 K8315:F=G:GOSUB K1450:X=X+1:GOTO K831
0
4000 GOSUB K1300:GOSUB K7500:G=F:GOSUB
 K1300:T=E
4010 GOSUB K1300:GOSUB K7500:H=F:GOSUB
 K1300:IF E<>27 THEN 8020
4030 GOSUB K1300:GOSUB K1300:GOSUB 850
0:IF T=32 THEN 4200
4042 IF T<>34 THEN 8020
4050 J=208:K=J
4100 F=G:GOSUB K1450:X=X+1:GOSUB K8320
:F=H:GOSUB K1450:X=X+1:P=205:GOSUB K83
15:IF J=208 THEN 4120
4115 P=240:GOSUB K1400:P=5:GOSUB K1400
:P=144:GOSUB K1400:P=11:GOSUB K1400:P=
24:GOSUB K1400
4120 P=K:GOSUB K1400:P=11:GOSUB K1400:
F=G:GOSUB K1450:GOSUB K8320
```

```
4140 F=H:GOSUB K1450:P=205:GOSUB K8315
:P=J:GOSUB K1400:P=3:GOSUB K1400:GOTO
8360
4200 J=176:K=144:GOTO 4100
6000 GOSUB K1300:GOSUB K7500:G=F:A=A+1
:GOSUB K1300:GOSUB K7500:H=F
6010 P=162:GOSUB K1400:P=0:GOSUB K1400
:F=H:GOSUB K1450:GOSUB K8320
6030 P=168:GOSUB K1400:F=G:GOSUB 8600:
P=152:GOSUB K1400:P=129:GOSUB K1400:P=
203:GOTO K1400
7500 IF E>127 AND E<201 THEN F=E-127:R
ETURN
8020 POKE 85,28:? "ERROR ";PEEK(L1+C
-2)+PEEK(L1+C-1)*256:END
8300 P=169:GOSUB K1400:P=Z:GOTO K1400
8310 P=141
8315 GOSUB K1400:P=X:GOSUB K1400:P=Y:G
OTO K1400
8320 P=173:GOTO K8315
8360 P=76:GOSUB K1400
8365 P=L:GOSUB K1400:P=M:GOTO K1400
8500 A=A+1:GOSUB 8590:U=Z:A=A+1:GOSUB
8590:V=Z:A=A+1:GOSUB 8590:W=Z:Z=U:IF E
=65 THEN Z=U*100+V
8520 IF E=66 THEN Z=U*10000+V*100+W
8540 M=INT(Z/256):L=Z-M*256:RETURN
8590 X=INT(PEEK(A)/16):Y=PEEK(A)-X*16:
Z=X*10+Y:RETURN
8600 GOSUB K1450:GOSUB K8320:P=133:GOS
UB K1400:P=203:GOSUB K1400
8610 X=X+1:GOSUB K8320:P=133:GOSUB K14
00:P=204:GOTO K1400
```

●

CHECKSUM DATA.

(see page 25)

```
1000 DATA 970,451,35,371,792,458,450,1
6,710,172,335,916,984,712,713,8085
1300 DATA 880,664,837,52,812,277,336,9
54,603,881,613,939,275,861,252,9236
3070 DATA 719,377,742,794,98,51,498,18
,307,78,674,384,605,838,399,6582
3720 DATA 606,844,75,777,931,638,747,4
40,106,201,59,72,253,310,958,7017
7500 DATA 309,274,819,470,8,704,771,52
7,148,214,419,988,927,51,6629
```

●

Listing 3.

```
5 Z=0:W=1
10 C=89:B=PEEK(C):A=256:GOSUB 900
20 DS=C:C=88:B=PEEK(C):DS=DS+B
100 P=W:Q=255:E=960:E=E+DS
110 A=DS
120 POKE A,P:A=A+W:IF A<E THEN 120
130 P=P+W:IF P<Q THEN 110
140 END
900 C=Z:D=Z:IF B=Z THEN 904
902 C=C+A:D=D+W:IF D<B THEN 902
904 RETURN
999 END
```

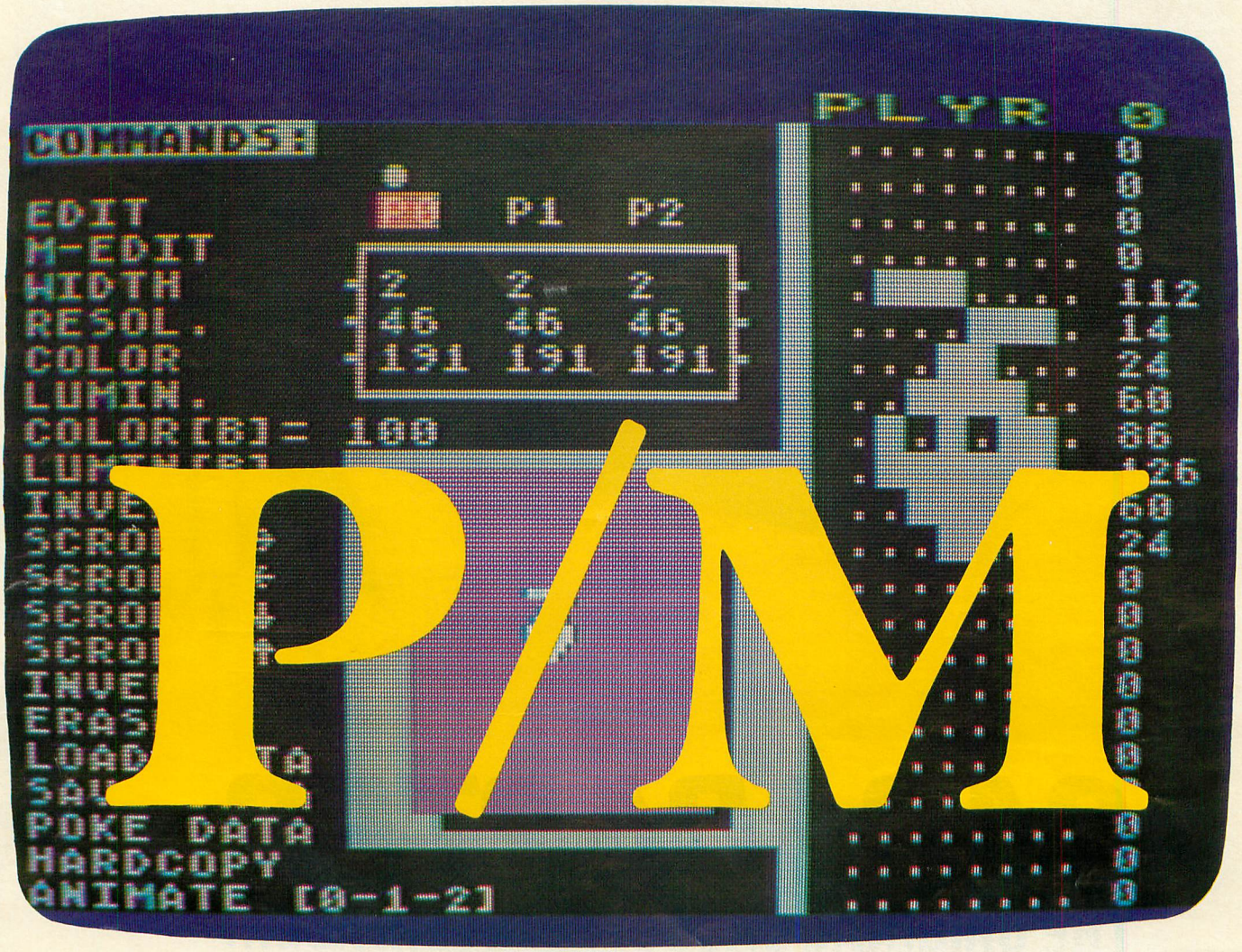
●

CHECKSUM DATA.

(see page 25)

```
5 DATA 332,317,3,356,153,421,401,38,70
9,17,601,78,3426
```

●



Creator/Animator

16K Cassette or 24K Disk

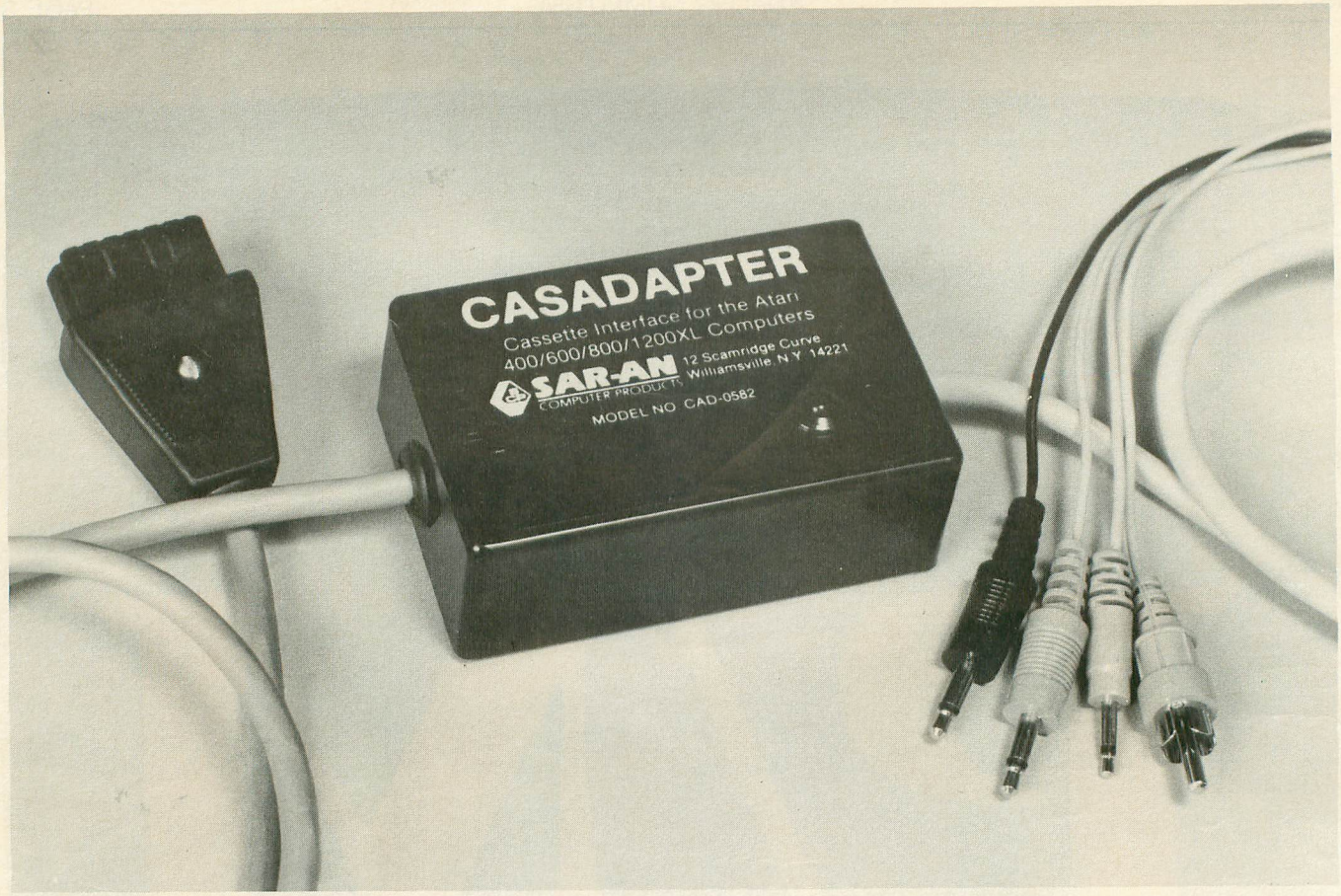
by Scott Sheck

Using player/missile graphics in your game can add extra smooth action and color, giving it a more professional look, but first you must be able to implement it. This can easily be accomplished using a pre-written P/M subroutine, namely Tom Hudson's P/M mover machine language subroutine (**ANALOG** issue 10). Everything you need to quickly move players around on-screen is contained in this subroutine! However, I found that you'll need to design your own player's shape, size and color. I don't like the idea of using graph paper and colored pencils, and, besides, it's really hard to get a feel for how the player will actually look on the screen. That's why I present to you my **Player Creator & Animator.**

How to use it.

After typing in the program, save it before running **PC&A**. Since the program contains some machine language routines, any mistake in typing these routines can result in your computer's locking up (which does no damage to the computer, just your schedule).

When you get the program running, you should see a flashing cursor on a grid of dots. This grid is where you will be designing your player shapes. The joystick is used to move the cursor around, and the fire button, when pressed, will plot a white square both on the grid and at the middle of the screen, where the player's actual size, shape and color is displayed. To erase a white square, press the fire button again. To



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 #116 ON READER SERVICE CARD

draw a continuous line, hold the fire button down, then move the joystick.

By looking either at the green letters above the grid or at the position of the orange square at the center of the screen, you can always tell which player is currently being edited. Up to three different players can be edited at a time.

You'll notice that the attributes of each player are displayed in the center white box. To change any of these attributes, move the cursor off the left or right side of the grid. This will place the flashing cursor at the top of the command list.

The cursor can be moved back and forth freely between the command list and the grid. To choose a command, position the cursor over the desired command, then press the joystick's fire button. Depending on which command you choose, either a "busy" message will appear, telling you that your command is being processed, or a message prompting you for your input will be displayed. You can either respond to the message (usually using keyboard input) or cancel the command by moving your joystick in any direction.

Command descriptions.

EDIT: Selects a single player to be edited and displays it by itself. The flashing cursor is then positioned on the grid.

M-EDIT: Allows you to display one to three players together. Move the arrow over the player in the center of the screen by pushing the joystick right. To display a player, press the fire button after the arrow has been positioned. That player will then be displayed, and a small white circle will appear above that player's number, letting you know that it's being displayed below. That player will then be placed on the grid for you to edit.

WIDTH: Changes width of player being edited to normal (2), double (1) or quadruple (3) width.

RESOL: Switches all players between single-line (62) and double-line (96) resolution.

COLOR: Changes color of player being edited. Use the keyboard to enter a color number between 0 and 255, then press RETURN.

LUMIN: Changes luminance of player being edited. Notice color value gets changed.

COLOR[B]: Changes color of background. Use the keyboard to enter a color number between 0 and 255, then press RETURN.

LUMIN[B]: Changes luminance of the background. Notice color value gets changed.

INVERT: Flips player being edited upside-down.

SCROLL →: Scrolls player being edited right.

SCROLL ←: Scrolls player being edited left.

SCROLL ↓: Scrolls player being edited down.

SCROLL ↑: Scrolls player being edited up.

INVERSE: Reverses player image being edited as inverse video.

ERASE: Erases player being edited. Press any key to confirm. Move joystick to cancel.

LOAD DATA: Loads data of all three players from tape. It will replace any currently existing data with the data loaded in. You can be in either resolution mode when loading or saving data. Press the PLAY button, then any key on the keyboard to begin loading.

SAVE DATA: Saves data of all three players to tape. You can be in either resolution mode when saving or loading data. Press the PLAY and RECORD buttons, then any key on the keyboard to begin saving.

POKE DATA: Allows you to poke your own player data into the grid. Use the keyboard to enter a data number from 0 to 255. Move the joystick to exit.

HARDCOPY: Prints out information about all three players. Turn on your printer, then press any key on the keyboard to begin printing.

ANIMATE: Shows animation using all three players.

(continued on page 36)

Move the joystick:

RIGHT — To exit.

LEFT — To reorder the animation sequence. The sequence is displayed at the bottom of the screen.

DOWN — To slow down the animation.

UP — To speed up the animation.

If an error occurs during the program (which shouldn't happen), you can always recover by pressing SYSTEM RESET, then typing RUN. The only way the player images can be lost is by using the ERASE command or by turning off the computer.

Multi-color players.

When two players overlap, namely players 0 and 1 or 2 and 3, a third color can be generated in the overlapping region. This is achieved by turning on what's called the "multi-color enable bit." In my program, press the OPTION key to turn this enable on or off. Note that this works *only* when overlapping players 0 and 1 in my program. Since I use player 3 for the background, it is removed if you decide to turn on the multi-color enable, leaving you with only a black background.

A tip.

While writing this program, I discovered a neat little trick that has to do with using the INPUT command. As you know, executing an INPUT command halts the program until the user presses the RETURN key. Since I wanted the joystick to handle practically everything, I needed a way for the user to be able to abort an input prompt (such as asking for a color number), using the joystick. The solution I came up with can be seen in Lines 2637 and 2646-2649. If a key is pressed, then the INPUT command is executed. If the joystick is moved, then the INPUT command is *not* executed.

Now, go forth and create! □

BASIC listing.

```
1 REM **************************
2 REM * P/M CREATOR/ANIMATOR *
3 REM *    BY SCOTT SHECK    *
4 REM *   ANALOG COMPUTING   *
5 REM **************************
20 GOSUB 20000:GOTO 2100
149 REM EDIT
150 PLEDIT=(PLEDIT+C1)*(PLEDIT<C2)
152 FOR P=CZ TO C2:POKE 53248+P,CZ:NEX
T P:POSITION 12,C2:? "                ";
153 GOSUB 2635:COLOR 20:PLOT 12+PLEDIT
*C4,C2:POKE 53252,96+PLEDIT*16
155 GOSUB 2642:X=27:Y=C1
160 POKE 53248+PLEDIT,114:IF ASC(WIDTH
$(PLEDIT+C1))=C3 THEN POKE 53248+PLEDI
T,104
175 RETURN
199 REM M-EDIT
200 XP=12:GOSUB C2640:? " right - TO S
ELECT  button - TO DISPLAY"
202 COLOR 29:PLOT XP,C1
203 A=C2^C2:IF STRIG(CZ)=CZ THEN PLEDI
T=(XP-12)/C4:GOSUB 153:GOTO 210
205 IF STICK(CZ)<>7 THEN 203
210 COLOR 32:PLOT XP,C1:IF XP=20 THEN
RETURN
215 XP=XP+C4:GOTO 202
249 REM WIDTH
250 NUM=(NUM<C3)*ASC(WIDTH$(PLEDIT+C1)
)+C1
254 WIDTH$(PLEDIT+C1,PLEDIT+C1)=CHR$(N
UM)
255 Z=53256+PLEDIT:ROW=5:GOSUB 415:GOS
UB 160
260 RETURN
299 REM RESOL.
300 FOR A=CZ TO C2:T(A)=P(A):NEXT A
302 IF RES=46 THEN GOSUB 21220:GOTO 30
7
303 GOSUB 21250
307 POSITION 12,6:? RES,RES,RES
310 FOR A=CZ TO C2:Z=T(A):FOR M=C1 TO
22
315 POKE P(A)+M,PEEK(Z+M):POKE Z+M,CZ
320 NEXT M:NEXT A:GOSUB 2635
344 RETURN
349 REM COLOR
350 GOSUB 2637
351 IF NUM<256 THEN Z=704+PLEDIT:ROW=7
:GOSUB 415
370 RETURN
399 REM LUMIN
400 Z=704+PLEDIT:ROW=7:A=PEEK(Z)/15:IF
 A<>INT(A) OR A=CZ THEN NUM=PEEK(Z)+C1
:GOTO 415
410 NUM=PEEK(Z)-14
415 POKE Z,NUM:POSITION 13+PLEDIT*C4,R
OW:? "  ←←←";NUM:RETURN
449 REM COLOR(B)
450 GOSUB 2637:IF NUM<256 THEN POKE C7
07,NUM:GOSUB 515
460 RETURN
499 REM LUMIN(B)
500 A=PEEK(C707)/15:IF A<>INT(A) OR A=
CZ THEN POKE C707,PEEK(C707)+C1:GOTO 5
15
510 POKE C707,PEEK(C707)-14
515 POSITION 12,9:? "  ←←←";PEEK(C707)
;:RETURN
550 A=USR(ADR(M1$)):RETURN :REM INVERT
600 A=USR(ADR(M2$)):RETURN :REM RIGHT
650 A=USR(ADR(M3$)):RETURN :REM LEFT
700 A=USR(ADR(M4$)):RETURN :REM DOWN
750 A=USR(ADR(M5$)):RETURN :REM UP
800 A=USR(ADR(M6$)):RETURN :REM INVERS
E
850 GOSUB 2640:GOSUB 2645:A=USR(ADR(M7
$)):RETURN :REM ERASE
899 REM LOAD
900 GOSUB C2640:? "  PRESS play,":GOSU
B 2645:GOSUB C2640:? "  LOADING DATA..
."
904 REM DISK: 905 OPEN#C1,C4,CZ,"D:FIL
ENAME.EXT"
905 POKE 764,63:OPEN #C1,C4,CZ,"C:"
910 INPUT #C1,WIDTH$:GET #C1,Z:POKE C7
07,Z:GOSUB 515
913 FOR PLEDIT=CZ TO C2:GET #C1,NUM:GO
SUB 351:A=P(PLEDIT)
920 FOR NUM=C1 TO C22:GET #C1,Z:POKE A
+NUM,Z:NEXT NUM
925 NUM=ASC(WIDTH$(PLEDIT+C1)):GOSUB 2
55
930 NEXT PLEDIT
931 PLEDIT=CZ:GOSUB 152:CLOSE #C1
933 RETURN
949 REM SAVE
950 GOSUB C2640:? "  PRESS play & rec,
":GOSUB 2645:GOSUB C2640:? "  SAVING D
ATA..."
951 REM DISK: 955 OPEN#C1,8,CZ,"D:FILE
NAME.EXT":?#C1;WIDTH$:PUT#C1,PEEK(704)
955 POKE 764,63:OPEN #C1,8,CZ,"C:":PRI
NT #C1;WIDTH$:PUT #C1,PEEK(C707)
966 FOR I=CZ TO C2:PUT #C1,PEEK(704+I)
:A=P(I)
977 FOR NUM=C1 TO C22:PUT #C1,PEEK(A+N
UM):NEXT NUM
983 NEXT I:CLOSE #C1:RETURN
999 REM POKE DATA
1000 FOR COL=C1 TO C22
1005 TRAP 1005:GOSUB C2640:? "   data
";COL;"?←";:GOSUB 2646:INPUT NUM
```

```
1010 TEMP=P(PLEDIT)+COL:GOSUB 2600:POS
ITION 27,COL
1011 Z=128:FOR P=CZ TO 7:IF NUM<Z THEN
 ? ".";:GOTO 1013
1012 ? "■";:NUM=NUM-Z
1013 Z=Z/C2:NEXT P:GOSUB 2644:NEXT COL
1015 RETURN
1049 REM HARDCOPY
1050 GOSUB C2640:? "  READY PRINTER":G
OSUB 2645:CLOSE #C1:TRAP 1050:OPEN #C1
,8,0,"P:"
1056 ? #C1:? #C1;"BGRND COLOR=";PEEK(C
707)
1058 ? #C1;"POKE 559,";RES
1060 FOR Z=CZ TO C2:? #C1:? #C1;"PLAYE
R ";Z:? #C1;"POKE ";C707+Z;",";PEEK(C7
07+Z);":REM COLOR"
1065 ? #C1;"POKE ";53256+Z;",";ASC(WID
TH$(Z+C1));":REM WIDTH"
1067 ? #C1;"DATA ";:FOR P=P(Z) TO P(Z)
+C22:? #C1;PEEK(P);:IF P<>P(Z)+C22 THE
N ? #C1;",";:NEXT P
1075 NEXT Z:? #C1:CLOSE #C1
1095 RETURN
1099 REM ANIMATE
1100 GOSUB C2640:? "down-SLOW SEQ.-lef
t   up-FAST STOP-rt."
1105 M=CZ
1110 POKE 53248+ORD(M),114
1114 S=STICK(CZ):IF S=13 THEN D=D+C1
1116 IF S=7 THEN GOSUB 2636:GOSUB 2641
:GOTO 931
1118 IF S=11 THEN GOSUB 1140
1120 IF S=14 THEN D=(D-C1)*(D>CZ)
1124 FOR P=C1 TO D:NEXT P:POKE 53248+O
RD(M),CZ:M=(M+C1)*(M<C2)
1135 GOTO 1110
1140 FOR A=CZ TO C2
1141 GOSUB C2640:? " frame ";A+C1;" (0
-2)";:INPUT Z:IF Z>C2 THEN 1141
1142 ORD(A)=Z:POSITION 10+C2*A,C22:? Z
:NEXT A:POP :GOTO 1100
1999 REM JOYSTICK
2000 JX=X:JY=Y:LOCATE JX,JY,CH:I=128
2002 CH=CH+I:? "←";CHR$(CH);:I=-I
2003 IF PEEK(53279)<>3 THEN 2005
2004 TEMP=PEEK(208):POKE 208,(TEMP=0)*
32:POKE 53251,(TEMP<>CZ)*108:POKE 5325
5,(TEMP<>CZ)*100
2005 TR=STRIG(CZ):IF STICK(CZ)=15 AND
TR THEN A=C2^C2:GOTO 2002
2009 IF I<CZ AND (TR OR X=C1) THEN 200
2
2010 S=STICK(CZ):X=JX+(S=7)-(S=11):Y=J
Y+(S=13)-(S=14)
2015 RETURN
2099 REM CHART
2100 IF X>34 OR X<27 THEN X=C1:Y=C3:TN
=43:GOSUB 2644:GOSUB 2605
2110 IF Y=CZ THEN Y=C22
2115 IF Y=23 THEN Y=C1
2120 GOSUB 2000
2125 IF  NOT TR THEN GOSUB 2500
2130 GOTO 2100
2499 REM FIRE BUTTON PRESSED
2500 NUM=ASC(BIT$(35-JX)):COL=JY
2510 CH$="■":IF ASC(CH$)=CH-128*(I<CZ)
 THEN CH$=".":NUM=-NUM
2520 ? "←";CH$;:TEMP=P(PLEDIT)+COL:NUM
=NUM+PEEK(TEMP):GOSUB 2600
2530 IF STRIG(CZ)=CZ AND STICK(CZ)=15
THEN 2530
2531 GOSUB 2010
2540 RETURN
2599 REM UPDATE CHART & PLAYER
2600 POKE TEMP,NUM
2601 POSITION 37,COL:? "  ←←←";NUM;:RE
TURN
2605 REM COMMANDS
2606 GOSUB 2000:IF STRIG(CZ)=C1 THEN 2
614
2607 TN=20:GOSUB 2644:TRAP 931:IF JY=C
3 OR JY=6 OR JY>10 AND JY<18 THEN GOSU
B 2641
2610 GOSUB JY*50:IF JY>10 AND JY<18 TH
EN GOSUB 2642
2612 GOSUB 2636
2614 IF X<>C1 THEN X=27+7*(X<C1):TN=43
:GOSUB 2644:RETURN
2616 IF Y=C2 THEN Y=C22
2618 IF Y=23 THEN Y=C3
2619 GOTO 2605
2630 REM SUBROUTINES
2635 P=P(PLEDIT):POKE C1,INT(P/256):PO
KE CZ,P-256*INT(P/256):RETURN
2636 GOSUB C2640:POSITION 33,CZ:? "PLY
R ";PLEDIT;:RETURN
2637 GOSUB C2640:? "   COLOR(0-255)?←"
;:GOSUB 2646:INPUT NUM:RETURN
2640 POSITION CZ,CZ:? "
                                ":POSITION CZ,
CZ:RETURN
2641 POSITION 7,CZ:? "BUSY":RETURN
2642 A=USR(1600):Z=P(PLEDIT)
2643 FOR COL=C1 TO C22:SOUND CZ,11+COL
,10,11:NUM=PEEK(Z+COL):GOSUB 2601:NEXT
 COL:SOUND CZ,CZ,CZ,CZ:RETURN
2644 FOR Z=15 TO CZ STEP -C2:SOUND CZ,
TN,10,Z:NEXT Z:SOUND CZ,CZ,CZ,CZ:RETUR
N
2645 POSITION C22,CZ:? "PRESS any key"
2646 POKE 764,255
2647 IF STICK(CZ)<>15 THEN POP :GOTO 2
612
2648 IF PEEK(764)=255 THEN 2647
2649 RETURN
19999 REM TITLE
20000 CZ=0:C1=1:C2=2:C3=3:C4=4:C22=22:
C707=707:C2640=2640
20001 GRAPHICS 18:Z=PEEK(560)+PEEK(561
)*256:POKE Z+C4,252
20002 POKE 708,0
20003 POKE 201,10:? #6;;;;"       playe
r","creator & animator";;"       BY"
;;;"    SCOTT SHECK"
20004 REM DLI
20005 FOR X=1536 TO 1543:READ A:POKE X
,A:NEXT X:DATA 72,165,208,141,27,208,1
04,64
20006 POKE 623,16:POKE 512,CZ:POKE 513
,6
20007 REM SCROLL TITLE
20008 FOR X=252 TO 112 STEP -C1:SOUND
CZ,X+150,8,9:POKE Z+C4,X:NEXT X:SOUND
CZ,CZ,CZ,CZ
20009 REM ML CHARTMAKER
20010 FOR X=C1 TO 41:READ A:POKE 1599+
X,A:NEXT X
20011 DATA 104,160,22,132,84,162,27,13
4,85,164,84,177,0,133,2,162,8,134,4,38
,2,176,4,169,46
20012 DATA 208,2,169,160,32,164,246,19
8,4,208,239,198,84,208,221,96
20013 DIM ORD(2),BIT$(8),CH$(1),P(2),T
(2),WIDTH$(3),FILL$(19),M1$(39),M2$(12
),M3$(12),M4$(13),M5$(15),M6$(13)
20014 DIM M7$(11):FOR X=1 TO 39:READ N
:M1$(X)=CHR$(N):NEXT X:FOR X=1 TO 12:R
EAD N:M2$(X)=CHR$(N):NEXT X
20015 FOR X=1 TO 12:READ N:M3$(X)=CHR$
(N):NEXT X:FOR X=1 TO 13:READ N:M4$(X)
=CHR$(N):NEXT X
20016 FOR X=1 TO 15:READ N:M5$(X)=CHR$
(N):NEXT X:FOR X=1 TO 13:READ N:M6$(X)
=CHR$(N):NEXT X
20017 FOR X=1 TO 11:READ N:M7$(X)=CHR$
(N):NEXT X:FOR X=1 TO 18:READ N:FILL$(
X)=CHR$(N):NEXT X
20018 FOR X=1 TO 8:READ N:BIT$(X)=CHR$
(N):NEXT X
20019 REM DLIST
20020 GRAPHICS CZ:X=PEEK(560)+256*PEEK
(561)
20025 POKE X+C3,70:POKE X+6,6:POKE X+1
2,130:POKE 54286,192
20099 REM P/M
20100 X=PEEK(106)-16:POKE 54279,X:PMBA
SE=256*X
20105 GOSUB 21250:POKE 53277,C3
20110 FOR X=CZ TO C2:ORD(X)=X:POKE 532
56+X,CZ:WIDTH$(X+C1,X+C1)=CHR$(C2):POK
E 704+X,191:NEXT X
20115 POKE 53251,108:POKE C707,100:POK
E 53259,C3
```

```
20122 A=USR(ADR(FILL$),PMBASE+446,37,1
92):POKE 53255,100:POKE 53260,195
20124 A=USR(ADR(FILL$),PMBASE+1916,74,
255)
20130 A=USR(ADR(FILL$),PMBASE+416,C3,C
3):POKE 53252,96
20699 REM SCREEN SET-UP
20700 POKE 708,24:POKE 712,144:POKE 71
0,C2:POKE 752,C1:POKE 82,11:POKE 83,23
:POSITION 11,C2
20701 ? " P0  P1  P2
  2   2   2   46  46  46  191 191 191
"
20704 POKE 82,C1:POKE 83,9:POSITION C1
,C1
20705 ? "COMMANDS ↓EDIT     M-EDIT   H
IDTH    RESOL.    COLOR    LUMIN.   COL
OR[B]=LUMIN[B] INVERT   SCROLL ←→"
20710 ? "SCROLL ↑← SCROLL ↑↓ SCROLL ↑→
 INVERSE  ERASE    LOAD DATASAVE DATAP
OKE DATAHARDCOPY"
20713 POKE 83,39:? "ANIMATE [0-1-2]";
20720 COLOR 149:PLOT 13,10:DRAWTO 23,1
0:COLOR 21:PLOT 13,20:DRAWTO 24,20
20725 COLOR 160:PLOT 25,C22:DRAWTO 25,
C1:PLOT 11,10:DRAWTO 11,20:PLOT 24,20:
DRAWTO 24,10:PLOT 12,10:DRAWTO 12,20
20726 PLOT 13,20:PLOT 23,10:COLOR 153:
PLOT 23,11:DRAWTO 23,19:COLOR 140:PLOT
 23,20:GOSUB 515:GOSUB 931:POKE 201,C4
21000 GOSUB 2636:RETURN
21219 REM CHANGE RESOLUTION
21220 P=CZ:RES=62:P(CZ)=PMBASE+1164:P(
C1)=PMBASE+1420:P(C2)=PMBASE+1676:GOSU
B 21260:REM SINGLE 62
21241 Z=(P<>255)*192:A=USR(ADR(FILL$),
PMBASE+892,74,Z)
21244 A=USR(ADR(FILL$),PMBASE+832,6,C3
*(Z=192)):POKE 559,RES:RETURN
21250 P=255:RES=46:P(CZ)=PMBASE+582:P(
C1)=PMBASE+710:P(C2)=PMBASE+838:GOSUB
21241:REM DOUBLE-46
21260 A=USR(ADR(FILL$),PMBASE+958,37,P
):RETURN
21290 REM ML ROUTINE DATA
21300 DATA 104,216,169,1,133,2,169,22,
133,3,164,3,177,0,170,164,2,177,0,133,
4,138,145,0,164
21310 DATA 3,165,4,145,0,230,2,198,3,1
92,13,16,228,96,104,160,23,177,0,74,14
5,0,136,208,248
21320 DATA 96,104,160,23,177,0,10,145,
0,136,208,248,96,104,160,21,177,0,200,
145,0,136,136,16,247
21330 DATA 96,104,160,2,177,0,136,145,
0,200,200,192,24,208,245,96,104,160,22
,169,255,81,0,145,0
21340 DATA 136,208,247,96,104,169,0,16
0,23,145,0,136,208,251,96,104,104,133,
1,104,133,0,104,104,168
21350 DATA 104,104,145,0,136,16,251,96
,1,2,4,8,16,32,64,128
```

•

CHECKSUM DATA.
(see page 25)

```
1 DATA 3,101,486,851,11,95,654,785,451
,507,197,284,607,486,983,6501
202 DATA 150,80,712,193,962,224,571,45
5,765,601,486,396,6,20,406,6027
310 DATA 211,56,82,601,257,842,365,606
,282,236,509,85,295,39,605,5071
499 DATA 320,639,399,639,136,144,833,8
79,731,358,753,672,954,294,215,7966
910 DATA 696,622,12,560,348,513,609,70
2,567,782,952,630,496,105,658,8252
1000 DATA 82,848,656,981,797,959,790,5
82,548,102,60,276,354,436,527,7998
1095 DATA 798,257,178,295,14,618,720,7
24,53,474,715,621,498,521,674,7160
2000 DATA 930,923,618,104,63,820,244,7
92,388,923,924,2,928,237,708,8604
2499 DATA 101,251,451,545,829,946,800,
551,655,520,586,986,660,383,973,9237
2614 DATA 192,911,27,759,912,550,463,2
84,785,158,859,347,211,425,39,6922
2647 DATA 128,911,821,535,724,112,195,
504,747,559,97,371,898,626,129,7357
20011 DATA 495,242,985,860,556,573,632
,195,454,734,126,795,860,989,336,8832
20115 DATA 654,349,268,722,674,206,330
,184,869,796,777,153,837,876,321,8016
21219 DATA 214,278,795,569,941,90,535,
460,438,821,637,115,61,5954
```

•

Fire Bug

16K Cassette or 24K Disk

by Tom Hudson and Kyle Peacock

You say you want action? You say you want great sound effects, great graphics, good playability and, above all, fine scrolling? What's that? You say you want all that and more for your three dollars? No problem. Here's a little something Tom and I threw together in three weeks that should satisfy all your honking and tooting.

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 **Fire Bug**. 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 **Fire Bug**, 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 **Fire Bug**.

Cassette instructions.

1. Type Listing 1 into your computer using the BASIC cartridge and verify your typing with **C:CHECK** (see page 25).

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

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

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 Assit (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,

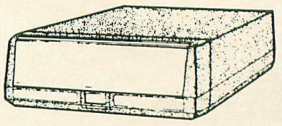
Add $2.50 shipping per software order in continental U.S. Add $5.00 shipping per software order for AK, HI, FPO-APO. Add $10.00 or 15% (whichever is greater) per software order for non-U.S. Call for cost of hardware shipping. Calif. residents add 6½% sales tax. Cashiers checks or money orders filled 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

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 **Fire Bug**, 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 the 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 **Fire Bug** will load and run automatically.

Disk instructions.

1. Type Listing 1 into your computer, using the BASIC cartridge and verify your typing with **D:CHECK2** (see page 25).

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. **Fire Bug** will load and run automatically.

Playing the game.

Fire Bug is a one-player game that requires a joystick in port one. To begin the game, press the START key while the opening credits are being displayed. (Tom and I had a hard time deciding whose name would go first.) You can select which level you wish to start on by pressing the corresponding number, 1 through 9. Pausing of the game is accomplished by pressing the spacebar. Hitting the spacebar a second time will resume the game.

Many years ago, a species of insect inhabited the subterranean chambers of Earth. Here, the "fire" species flourished for some four million years. Every two thousand years, the entire population expired, with the exception of one female. This female **Fire Bug** served as guardian over the dispersed fire nests of the underground. For eighty years, she defended these nests and their immature eggs from natural predators. Capable of emitting small, lethal "sparks," she combed Earth's interior with an ever-watchful eye. Such an instinct kept her species alive for aeons before man emerged on the surface of the world.

Now the **Fire Bug** is threatened with extinction. Man's nuclear wastes have infested the soil and mutated the predators of the **Fire Bug** into hideous,

(continued on next page)

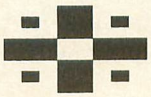
unstoppable bug-eating machines. These new super predators have the capacity to reproduce at astounding rates. Should they happen upon a **Fire Bug** nest, escape for the fire eggs is doubtful, if not impossible.

Firebug.

As fate would have it, you are the last remaining female **Fire Bug**. You must defend the nests and pave the way for your species' next generation. To further outline your eighty-year objective, the following illustrations will help you distinguish friend from foe.

This is your **Fire Bug**. It has the capability to burrow tunnels and fire sparks. Burrowing through the soil requires greater effort, thus your **Fire Bug** will travel more slowly through soil than through already existing tunnels. Coming in contact with any of the enemy predators will lead to instant vaporization. Use your **Fire Bug** carefully; you only have five lives!

This is a spark. It will destroy a predator or create a partial tunnel. They have a limited range, which is usually just beyond the screen's edge. The small bar graph at the bottom of the screen dictates how many sparks remain to you. Sparks regenerate one every two seconds while the **Fire Bug** is burrowing. The regeneration rate is one every four seconds while standing still or traveling through an already existing tunnel.



This is a fire egg. Initially, several eggs will be grouped in a small fire nest. Fire eggs do not have the capability to create their own tunnels; they will, however, travel through already existing ones. These eggs make a delightful snack for a lucky predator. The innocent "tweets" of a perishing egg would bring tears to any **Fire Bug**'s eye. Should all the fire eggs be destroyed or eaten, the fire species will terminate, and the game will end.

This is a proximity bomb, a product of man's nuclear waste. It is highly unstable. Should your **Fire Bug** tunnel next to or shoot past a proximity bomb, the device will detonate shortly thereafter. Being caught in the explosion radius is just as lethal as being chewed by a predator. Caution should be exercised with this nuclear trash, as closely grouped bombs may set off a chain reaction.

New Jersey BBS UPDATE

To all of you dedicated New Jersey users who've been patiently awaiting the correct phone number for the Jersey Atari Computer Group (issue 19's listing was for the wrong number; in issue 21's **Reader Comment**, we informed you that the number given in that earlier listing was a store, not a BBS), here it is:

(201) 549-7591

The New Jersey group is planning to feature a 25 megabyte hard disk on-line—to serve *all* of you.

(continued on page 44)

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!

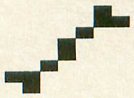
The MMG BASIC COMPILER comes with both single and double density versions on the same disk, and is available from your local computer store, or send $99.95 plus $3.00 for shipping and handling to:

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 #120 ON READER SERVICE CARD

This is a super predator, class one. They have the capability to create their own tunnels. They thrive on fire eggs, as well as **Fire Bugs**. They can be destroyed by a spark or the debris of a detonating proximity bomb. During mating season, should two of these bugs come in contact with one another, they will create four class two predators. Mating season occurs once every one hundred seconds.



This is a super predator, class two. They follow the same rules and restrictions as their class one counterparts. Should two class two bugs come in contact with one another, they will create four class one predators. The gurgling sound of mating predators strikes fear into a **Fire Bug**'s heart.

This month's public domain assembly language game should keep you busy for thirty days. Next month, Tom and I will be taking the BASIC spotlight with **BASIC Training** and **Bopotron!** Don't miss it. Aloha! □

BASIC listing.

```
10 REM *** FIREBUG ***
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$(X+1,X+1))-48:BYTE=H
EX(D1)*16+HEX(D2)
80 IF PASS=2 THEN PUT #1,BYTE:NEXT X:R
EAD CHKSUM:GOTO 50
90 TOTAL=TOTAL+BYTE:IF TOTAL>999 THEN
TOTAL=TOTAL-1000
100 NEXT X:READ CHKSUM:IF TOTAL=CHKSUM
 THEN 50
110 GOTO 220
120 IF PEEK(195)<>6 THEN 220
130 IF PASS=0 THEN 170
140 IF  NOT DSK THEN 160
150 PUT #1,224:PUT #1,2:PUT #1,225:PUT
 #1,2:PUT #1,0:PUT #1,36:CLOSE #1:END
160 FOR X=1 TO 92:PUT #1,0:NEXT X:CLOS
E #1:END
170 IF  NOT DSK THEN 200
180 ? "INSERT DISK WITH DOS, PRESS RET
URN";:DIM IN$(1):INPUT IN$:OPEN #1,8,0
,"D:AUTORUN.SYS"
190 PUT #1,255:PUT #1,255:PUT #1,0:PUT
 #1,36:PUT #1,251:PUT #1,54: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 0,39,216,7,255,7,169,0,141,47
,2,169,60,141,2,211,169,0,141,231,2,13
3,14,169,56,141,232,2
240 DATA 133,15,169,36,133,10,169,8,13
3,11,24,96
1000 DATA A9248589A9088581A90085888580
A000B188918088D0F9E681E689A589C938D0ED
4C2408A9228D2F02A90485C7,700
1010 DATA A90185BBA90185CD8D06D08D07D0
85BF201013A906A20BA0AC205CE4A907A20CA0
68205CE4A9C08D0ED4AD1FD0,938
1020 DATA C907F0068D69094C5708AD6909C9
06F02CAD0FD22904F0E5ADFC02A208DD6009F0
05CA10F830D686BBE6BB86BD,465
1030 DATA E6BD20D21A20E112A9FF8DFC024C
5708A9008D6909204910A90085BF85CDAD1FD0
C907F0034C3108ADFC02C921,450
1040 DATA D026AD0FD22904F01FA90185BF8D
FC02201013ADFC02C921D0F9AD0FD22904F0F9
A90085BF8DFC02A5DFD007A9,76
1050 DATA 0185DF206519A5E0D007A90385E0
202318A5D9D00F20380BA5A2F004A90BD002A9
0885D9A5D8F010A5CAD00CA9,400
1060 DATA 3C85CAC6C9D004A90085D8A5BEF0
034CA508A5BC4AC9109002A90FAABD500985BE
20880920161AA5E4D014E6BC,812
1070 DATA F8A5BD18690185BDD820D21A2004
134C9F084CA508141312111100F0E0D0C0B0A09
080706051F1E1A181D1B3335,368
1080 DATA 3000A598859 0A598591201C19C6
E3300920DD1A2004134C9F08200413 4C3108A9
0085E4A21D86D5BD1E05D003,441
1090 DATA 4C310ABD5A05858EBD7805858FA0
00B18EDD0005F018A9009D1E05A90C38FD0005
29060A0A0A859F20951A4C31,251
1100 DATA 0AA9008D69138D6B13A5D8D008A9
01BC00599611 3A90385D6A4D6BD5A05187959
13858EBD7805795D13858FA0,883
1110 DATA 00B18E297F85D3A8B96113F012B9
7413C9FFD00BA5D69D3C05206C0A4C310AC6D6
10C9A90485D2BD3C051005AD,915
1120 DATA 0AD229039D3C0520410A1009C6D2
10EAA9FF9D3C05A6D5BD1E05F002E6E4CA3003
4C8E0960A8BD5A0518795913,421
1130 DATA 858EBD7805795D13858FA000B18E
297F85D3A8B96113D003A9FF60B97413CD0AD2
90F5BD5A058588BD78058589,981
1140 DATA A900A89188C8918888BD0005918E
C8186901918EA58E9D5A05A58F9D7805A5D3DD
0005F0034C2D0BA9009D1E05,801
1150 DATA BD0005C908D008A90785A3A90AD0
06A90785A4A90885D7A90385D6A4D6A58E1879
59138588A58F795D13858 9A0,980
1160 DATA 00B188297FC901F04AC906F054C9
08F042C90AF03EC90CF048C90FF036A21DBD1E
05F00ECA10F8A90185D8A964,494
1170 DATA 85C94C2A0BA9019D1E05A9FF9D3C
05A5889D5A05A5899D7805A5D79D0005918818
6901C89188C6D630034CBE0A,275
1180 DATA A90060C906D0F9686868684C6A09
A598858 8A5998589A000B188C906F002D030AE
7802BC8713100160A5981879,376
1190 DATA 5913858EA599795D13858FA000B1
8EC901F0E8C90CF0E4C90FB0E0C9069009C90E
F00568684C6A09C900F00AC9,907
1200 DATA 0EF006A90185A2D004A90085A2A9
0E9188C8918888A906918EC8186901918EA58E
8598A58F859960D8A207AD0A,318
1210 DATA D229559D8804CA10F5A5CD0AAABD
850C8D3002BD860C8D3102A9918D0002A90C8D
0102CE350E102DAD0AD2293F,629
1220 DATA 090F8D350EEE340EAD340E29010A
0AAABD890C8D990EBD8A0C8D9A0EBD8B0C8DA1
0EBD8C0C8DA20EA5BFD05885,29
1230 DATA 4DCE320E1030AD7802C90FD004A2
00F002A6A2BD2F0E8D320EEE330EAD330E2901
F013A6B9BDE00EC914D007E0,630
1240 DATA 1DF006E6B9E8DEE00EA5CAF002C6
CAA5E0F002C6E0A5DFF002C6DFA5BEF002C6BE
A5D9F002C6D920751420940F,513
1250 DATA 4C5FE4D8207215A5BFD01220DF0C
20080D204D0D20720D20910D20BE0D4C62E436
0E8A0E440F6C0F6C0F440F48,9
```

```
1260 DATA 8A489848A904A23FA4CE8D0AD48E
00D48C1AD08D09D4A9B88D0002A90C8D010268
A868AA684048ADC8028D0AD4,9
1270 DATA 8D1AD0AD2F028D00D4A9D48D0002
A90C8D0102684048A9E08D0AD48D09D46840A5
A2F013CE310E101FA9038D31,598
1280 DATA 0EA6BABD7802C90FD007A9008D01
D2F00AA9468D01D2A9DC8D00D260A6A31007A9
008D03D2F00DBD3D0D8D02D2,648
1290 DATA A9A48D03D2C6A3A6A4100BA5A310
14A9008D03D2F00DBD450D8D02D2A9A48D03D2
C6A460F0A050000F5FAFFFD2,192
1300 DATA 8C46002D73B9FFA6C81009A5A310
04A5A430CF60BD680D8D02D2A9A68D03D2C6C8
60000810182028 2F373F47A6,785
1310 DATA A51006A9008D05D260BD8A0D8D04
D2A9AA8D05D2C6A5601239618 8B0D7FFA6A610
05A5A530DD60BDA80D8D04D2,169
1320 DATA A98C8D05D2C6A660F4E8DCD1C5BA
AEA2978B8074685D51463A2E23170C00A6DE10
11A93020E50DA5A61007A5A5,570
1330 DATA 10034C760D60BDEC0D8D04D2A98A
8D05D2C6DEAD0AD229708D360E8D370E6000FD
F9F5F1EDEAE6E2DEDAD6D3CF,816
1340 DATA CBC7C3BFBCB8B4B0ACA8A5A19D99
95918E8A86827E7B77736F6B6764605C585450
4D4945413D3936322E2A2622,156
1350 DATA 1F1B17130F0B0804783C00000000
0030303030303 0C61C0F44B00E740000740000
7400007400007400007400000,957
1360 DATA 7400007400007400007400007400
0074000074000074000074000074000074000
7400007400005400 0C4B00E,935
1370 DATA 00C4E00E20641360E7070707070
471C0F707046300F70476C0F7046580F704744
0F707070468 00F7046080F41,247
1380 DATA 8A0E0101010101010101010101
0101010101010101010101010101010101010
0101010101010101010101,442
1390 DATA 0101010101000000000000000000
0000000000000000000000000000000000000
0000000000000000000000,447
1400 DATA 4040404040406C766C5A808080
A2B5A7B39A80000000000006669726500627 5
6700000000000000000000,414
1410 DATA 3732293434252E00223900000000
0000000000B4AFAD00A8B5A4B3AFAE000000
0000000000000000000006,413
1420 DATA 0000000000000000000000000000
EBF9ECE500F0E5E1E3EFE3EB0000000000061
6E616C6F6700636F6D707574,274
1430 DATA 696E670000A00DA9138DC60FA5C0
0AAABDC80FC9FFD002A200BDC70F1865C19936
0EBDC80F690099370EE8E898,365
1440 DATA 186903A8CEC60F10DA6000002080
2000218021002280220023802300248024002 5
8025002680260027802700 28,46
1450 DATA 8028002980290 02A802A002B802B
002C802C002D802D002E802E002F802F003080
3000318031003280320033 80,583
1460 DATA 3300348034003580350036803600
3780370038803800398039003A803A003B803B
003C803C003D803D003E803E,296
1470 DATA 003F803FFFFFA9048589A20D86DD
A6DDBDC51385888A0A0A0AAAA000BDD3139188
E8C8C008D0F5C6DD10E3A900,208
1480 DATA 85BA85A185A085C3A202959A959D
CA10F9A90185C485D0A95485C5A92B85C6A903
85CF85D1A20486E3A5BB85BC,702
1490 DATA 85BDBD261395E8CA10F8A93E8D2F
02A9008D07D4A9038D1DD0A9218D6F02A9F08D
0CD0A9288D06D0A9CC8D07D0,136
1500 DATA A900AA9D0003E0C9B00BE0309007
A9F09D0003A900CAD0EB201013A20186BF8EFC
02CACA86A386A486A586A686,736
1510 DATA DE86C8A5C385C1A5C485C0A90085
A7A203A9FF95B1CA10F9A213A9149DEA0ECA10
FAA91D85B9A2008A0AA8B9C7,136
1520 DATA 0F8580B9C80F8581C9FFF013A07F
A904204314186902918088 10F3E84C1F11A200
A000BD3513996F14E8C8C006,20
1530 DATA D0F4AD6F1485CBAD701485CC2057
14A00098918 0AD6F14CD7114D008AC7014CC72
14F014186D73148D6F14AD70,63
1540 DATA 14186D74148D70144C5311E02490
B8A6BCE00B9002A20ABD2A1385DCAAA93A2043
14186903 0AA8B9C70F85809D,542
1550 DATA 9605B9C80F85819DFA05A9732043
14186906090 1A8B180F0D7C90FF0D3A90F9180
186901C8918088981 87D9605,566
1560 DATA 9D9605A9FF9D5E06CA10B8A200BD
C80FC9FFF0158581BDC70F8580A0019898180A0
7F9180E8E84CE511CACABDC7,894
1570 DATA 0F8580BDC80F8581ADC70F8586AD
C80F8587A07FA9019180918 68810F920A71A20
D21A20DD1AA21DA9009D1E05,880
1580 DATA CA10FAA203A9019D1E05A9FF9D3C
05BD97139D7805858FBD9B139D5A05858EA908
9D0005A000918EC818690191,589
1590 DATA 80CA10D4A90085D8A5BC38E9014A
4A4A85E5A90338E5E585E7AABDC11385E585D4
85E6A6E7BDB9138588BDBD13,39
1600 DATA 8589A20CA588187D9F13858EA4E6
99C206A5897DAC13858F99F606A901992A07A0
00A90C918EC8186901918EC6,247
1610 DATA E6CA10D3C6E710C1A9208599858F
A9838598858EA906A000918EC8186901918E20
E112A90085BF60AD0AD229F0,944
1620 DATA 090285CEA204AD0AD229F01D2113
9DC402CA10F2ADC5028DC2028DC30260A51418
6946C514D0FC4CE2102065E4,913
1630 DATA A207A9009D00D2CA10FA8D08D260
080A08040000040506070913 1D27313B454F59
633F0C3F1F0001400C401F00,474
1640 DATA 013F213F34000140214034000129
203E2001004120562001008002 80FEFF0000FF
0100010101010100000000 00,313
1650 DATA 0100010000000E00040404040FF
00FF00FF00FF00FF00000000FFFFFFFFFFFFFF
01FFFFFF03FF0200FF2E3030,278
1660 DATA 31BF3945BF008080FE02827E7E82
04FC00000000FFFF00FF00FF0000FFFF011F5F
1F5F283838280C1926330008,224
1670 DATA 1018202870788 0A0A8B0B8C00000
0000000000AAAAFFFFFFFFAAAA1001000440
0001400010014004001 10040,592
1680 DATA 0100044001001010010040011000
0400000000000000010401020B2F2F0A4010
0580E0F8F8A0FF6666666666,761
1690 DATA 66FFFF646464646464FFFF606060
606060FFFF4040404040 40FFFF000000000000
FFF0108D5614AD0AD2CD5614,925
1700 DATA 90054A4A4C4B1460008E5614A5CC
0AAABDC70F1865CB8580BDC80F8581AE561460
0000000000000A599CD590EB0,122
1710 DATA 04A900F007CD680E900AA9018D6E
15A2048E7015AD7015F041AD6E15D018C6A110
38A5C0C5C4D005E6A14CD014,138
1720 DATA A90785A1C6C04CCA14E6A1A5A1C9
08901CA5C0C5C6D005C6A14CD014A90085A1E6
C0CE70154CD514A9008D7015,998
1730 DATA A5A18D05D4AD580E297F8D6D15A5
98297F38ED6D15C91E9004A900F006C90FB00A
A9018D6F15A90C8D7115AD71,685
1740 DATA 15F041AD6F15F01CE6A0A5A0C904
9034A5C1C5C3D005C6A04C4115A90085A0C6C1
4C3B15C6A0101CA5C1C5C5D0,554
1750 DATA 05E6A04C4115A90385A0E6C1CE71
154C4615A9008D7115A5A08D04D4A6BABC7802
C00FF00ACC6215F005B95E15,619
1760 DATA 85E88C6215600003010100020002
0003000000000000A5C78DDD15AEDD15B4E8B5
EDD9E1159005B9E01595EDD9,989
1770 DATA E01590F60AA8B9F315858AB9F415
858BADDD150AA8B9E915858CB9EA15858DA00F
B18A918C8810F9CEDD1510C1,96
1780 DATA CEDE151010A5CF8DDE15A6BABD78
02C90FF002E6EDCEDF15100CA5D18DDF15A6C7
F6EDCAD0FB6000030300040 8,781
1790 DATA 0C10151F252830044004500460 04
900443165316631653167316831693168316A3
16B316C316B316D316E316F3,951
1800 DATA 16E3160317131723173317431753
176317731783179317A317931783177317631 7
B317C317D317E317D317C317,328
1810 DATA F3170318131800C0209C235CA3502
308060835CA35C8003020902F50AF502C08060
805FA05F800C02090235CA35,942
1820 DATA C2308060805CA35C830C33199999
19330CC000239898230 0C03333199999193333
0000209B9B200000CC331999,621
1830 DATA 991933CC000023989823000003 00
C82626C8000330CC64666664CC30000008E6E6
080000CCCC64666664CCCC00,514
```

```
1840 DATA 00C82626C8000033CC64666664CC
330235CA35C209020C805CA35C8360803002F5
0AF5020902038O5FA05F8060,529
1850 DATA 80C0C235CA35020902OC835CA35C
80608030300C02090932C00000038C60608030
0C00F00E09090E30300C0CB0,681
1860 DATA 6060B00F000000F20909020C0C30
308060608F000003000239C902030000C08063
6C8000C00C03020939C20000,301
1870 DATA 0000836C6080C030000000010108
A020080A2040400000000000010128080000
2028404000000000000109,644
1880 DATA 020000000080604000000000002
0901000000000004060800000000828010100
0000000004040282000020A0,676
1890 DATA 0801010000000000004040200A08
000000ABAB000000000000EAEA000000000000E
28280E00000000B02828B000,214
1900 DATA 000032020808023200008C802020
808C00C2020200000202C28380800000808083
0000000202000000000000080,236
1910 DATA 8000000000001040401000000000
4010104000000003003030000300000C0000C0C
00C0002029184C7E18C6D0D0,981
1920 DATA 50A90385D0A6BABD10D0D045B5A7
C905B03FADEA0EC918F038A203B5B1C9FFF006
CA10F74C7D18E6A7A59895A9,298
1930 DATA A59995ADA5E895B1A91495B5A906
85A5A6B9FEE00EBDE00EC918D008A5B9C90AF0
02C6B960A203B5B1C9FFD003,836
1940 DATA 4C0419B5A98584B5AD8585A000B1
84C912F004C911D007A90E9184C89184B5B110
09A9FF95B1C6A74C0419A8B5,430
1950 DATA A91879141985B4B5AD7918198585
A000B184297FF026C90EF02248A98095B168C9
01F02DC90FF029C90CF02520,553
1960 DATA 0B19A9119184C89184A91585A6D0
15A9129184C8186901918420OB19D6B51004A9
01D0C9CA30034C801860A584,254
1970 DATA 95A9A58595AD608002FE80FF00FF
00A90085DAA94285DEA91185DBA5DAC90D9007
38E90DA00084DBAAA590187D,147
1980 DATA 9F138592A5917DAC13859 3A000A5
DB9192C89192205A19E6DAA5DAC91AD0CB60A2
03A0FF88D0FDCAD0F860A6DC,865
1990 DATA BD5E06C9FFD04CBD96058594BDFA
058595A000B194C90FF016C90CD007A9809D5E
06D02FA90F9194A910C89194,600
2000 DATA D01FA003A59418795913859 6A595
795D13859784C2A000B196C912F004C906D00A
A93C9D5E06CA10AA3008A4C2,671
2010 DATA 8830F64C9819A6DCBD5E063044BD
96058590BDFA058591A000B190D004A90FD004
A9805190190C81869019190,7
2020 DATA A9008D06D2A9888D07D2DE5E06D0
15A9008D07D2A9809D5E0686DD84C2201C19A6
DDA4C2CA10B460A6E586E6BD,781
2030 DATA 2A07D0034C721ABDC2068588BDF6
068589A000B188C90CF015C6D4300CA9009D2A
07A90985C84C721A68684C82,559
2040 DATA 09A90385D2AD0AD22903A8BDC206
18795913858EBDF606795D13858FA000B18EF0
0CC90EF008C6D210DBCA10A3,773
2050 DATA 60A90E9188C8918888A90C918EC8
186901918EA58E9DC206A58F9DF6064C721AA0
00F818A202B59A759D959A94,288
2060 DATA 9DCA10F5D8A91085E1A200A000B9
9A0020BD1AE8E8C8C003D0F36085E2290F05E1
9D090FA5E24A4A4A4A05E19D,881
2070 DATA 080F60A05084E1A5BDA20B4CBD1A
A5E3300509908D1B0F60000000000000000000
00000000000000000000●●000,292
```

●

CHECKSUM DATA.

(see page 25)

```
10 DATA 904,351,496,811,423,729,200,60
3,555,573,694,613,29,205,226,7412
160 DATA 754,198,962,637,491,30,155,15
6,198,665,50,84,24,201,781,5386
1060 DATA 982,627,576,799,844,892,794,
971,870,802,2,928,665,988,837,11577
1210 DATA 44,455,968,158,858,67,818,99
1,6,791,966,30,392,960,51,7555
1360 DATA 804,302,501,337,960,2,386,50
,131,154,189,177,25,190,918,5126
1510 DATA 144,841,996,731,735,158,998,
910,393,924,956,737,618,884,920,10945
1660 DATA 802,311,958,773,776,820,40,6
1,884,933,633,390,287,166,274,8108
1810 DATA 430,374,441,468,337,83,694,5
58,901,761,866,200,129,795,841,7878
1960 DATA 515,97,9,144,727,569,991,838
,937,5,906,192,5930
```

●

Assembly language listing.

```
          .OPT NO LIST
;
; ************************
; *      FIREBUG         *
; *    by Kyle Peacock   *
; *      & Tom Hudson    *
; *   ANALOG Computing   *
; *  All Rights Reserved *
; ************************
;
;  Redefined Characters by
;    Jon Bell -- 6/26/84
;
; CONSTANTS
;
GAMESET  =   $0400    ;CHARACTER SET
PLAYBS   =   $00      ;P/M BASE ADDR.
SCREENBASE = $2000    ;BOARD RAM.
MISS     =   PLAYBS+768 ;MISSILES
BOMB     =   15
;
; ZERO PAGE USAGE
;
         *=   $80
LO       *=   *+2      ;2-BYTE POINTERS
BMB      *=   *+2
ZLO      *=   *+2      ;Z-PAGE POINTER.
LOW      *=   *+2      ;Z-PAGE POINTER.
HLO      *=   *+2
DRWLO    *=   *+2      ;CHAR DATA PNTR.
PLLO     *=   *+2      ;CHAR WORK PNTR.
ADL      *=   *+2
ELO      *=   *+2
ELO2     *=   *+2
BLO      *=   *+2
BLO2     *=   *+2
BUGL     *=   *+2      ;FIREBUG X & Y.
SCORE    *=   *+3      ;BCD SCORE VALUE
SCOADD   *=   *+3      ;SCORE ADD VALUE
HBIT     *=   *+1      ;HSCROL RAM COPY
VBIT     *=   *+1      ;VSCROL RAM COPY
DIGGIN   *=   *+1      ;DIGGING FLAG
CHANGE1  *=   *+1      ;MATING SOUND 1.
CHANGE2  *=   *+1      ;MATING SOUND 2.
BULLNOISE *=  *+1      ;BULLET NOISE
BOOMNOISE *=  *+1      ;BULLET NOISE
NOBULL   *=   *+2      ;SPARKS FIRED.
BULLX    *=   *+4      ;SPARK X-COORD.
BULLY    *=   *+4      ;SPARK Y-COORD.
BULLDIR  *=   *+4      ;SPARK DIRECT.
BULLTIME *=   *+4      ;SPARK LIFE
BULLPNT  *=   *+1      ;BAR LINE PNTR.
PLAYNO   *=   *+1      ;PLAYER UP.
STARTLEVEL *= *+1      ;STARTING LVL.
LEVEL    *=   *+1      ;CURRENT LEVEL
BCDLVL   *=   *+1      ;BCD LEVEL #
TIMER    *=   *+1      ;GENERAL TIMER
VSTOP    *=   *+1      ;HALT FLAG.
YPOINT   *=   *+1      ;BOARD Y-COORD.
XPOINT   *=   *+1      ;BOARD X-COORD.
YTEMP    *=   *+1      ;Y-REG TEMP HOLD
MINX     *=   *+1      ;MIN X VALUE
MINY     *=   *+1      ;MIN Y VALUE
MAXX     *=   *+1      ;MAX X VALUE
MAXY     *=   *+1      ;MAX Y VALUE
MAXBUGS  *=   *+1      ;# OF CHARs
EATNOISE *=   *+1      ;EATING SOUND
GENTIM   *=   *+1      ;MATING TIMER
GENCTR   *=   *+1      ;MATING COUNTER
XCOORD   *=   *+1      ;X COORDINATE
YCOORD   *=   *+1      ;Y COORDINATE
LISTPOINT *=  *+1      ;D-LIST PNTR.
NEWBACK  *=   *+1      ;BOARD B-COLOR.
MOVETIME *=   *+1      ;MOVEMENT TIMER.
FIRETIME *=   *+1      ;FIRING TIMER.
PHASETIME *=  *+1      ;PHASE TIMER.
ETRY     *=   *+1
OLDVAL   *=   *+1      ;OLD CHAR
EGGS     *=   *+1      ;BABY COUNT
ENIX     *=   *+1      ;ENEMY INDEX
GNEWIX   *=   *+1      ;GENERATION INDX
NEWTYP   *=   *+1      ;NEW ENEMY TYPE
GENFLG   *=   *+1      ;MATING FLAG
BUGTIM   *=   *+1      ;BUG TIMER
EXPPH    *=   *+1      ;EXPLOSION PHASE
EXPCHR   *=   *+1      ;EXPL. CHARACTER
BOMBCT   *=   *+1      ;# OF BOMBS
XTEMP    *=   *+1      ;X SAVE AREA
EXNOISE  *=   *+1      ;EXPLOSION SOUND
ARMTIM   *=   *+1      ;BOMB ARM TIMER
BULTIM   *=   *+1      ;BULLET TIMER
SHCOLR   *=   *+1      ;TEXT COLOR
SHOBYT   *=   *+1      ;TEXT CHAR
LIVES    *=   *+1      ;# OF BUGS LEFT
ECOUNT   *=   *+1      ;# OF BABIES
EGGTOT   *=   *+1      ;# OF EGGS
EGGIX    *=   *+1      ;BABY INDEX
EGROUP   *=   *+1      ;BABY GROUP #
TYPE     *=   *+5      ;CHAR. TYPE.
PHASE    *=   *+5      ;CHAR. PHASE.
;
; RAM USAGE
;
         *=   $0500
ETYP     *=   *+30     ;ENEMY TYPE
EACT     *=   *+30     ;ENEMY ACTIVE
```

```
EDIR    *=  *+30    ;ENEMY DIRECTION
EADL    *=  *+30    ;ENEMY ADDR LO
EADH    *=  *+30    ;ENEMY ADDR HI
BOMBL   *=  *+100   ;BOMB ADDR LO
BOMBH   *=  *+100   ;BOMB ADDR HI
ARMED   *=  *+100   ;BOMB ARMED FLAG
EBGL    *=  *+52    ;BABY ADDR LO
EBGH    *=  *+52    ;BABY ADDR HI
EBGACT  *=  *+52    ;BABY ACTIVE
; **********************
; * MASTER EQUATE FILE *
; **********************
;
; I/O Routines
;
DSKINV  =   $E453
SIOINV  =   $E465
;
; Operating System Equates
;
CASINI  =   $02
DOSVEC  =   $0A
DOSINI  =   $0C
ATRACT  =   $4D
POKMSK  =   $10
BRKKEY  =   $11
RTCLOK  =   $12
VDSLST  =   $0200
VIMIRQ  =   $0216
CDTMV1  =   $0218
CDTMV2  =   $021A
CDTMV3  =   $021C
CDTMV4  =   $021E
CDTMV5  =   $0220
CDTMA1  =   $0226
CDTMA2  =   $0228
CDTMF3  =   $022A
CDTMF4  =   $022C
CDTMF5  =   $022E
SRTIMR  =   $022B
SDMCTL  =   $022F
SDLSTL  =   $0230
SDLSTH  =   $0231
SSKCTL  =   $0232
LPENH   =   $0234
LPENV   =   $0235
GPRIOR  =   $026F
PADDL0  =   $0270
STICK0  =   $0278
STICK1  =   $0279
STRIG0  =   $0284
SHFLK   =   $02BE
PCOLR0  =   $02C0
COLOR0  =   $02C4
COLOR1  =   $02C5
COLOR2  =   $02C6
COLOR3  =   $02C7
COLOR4  =   $02C8
MEMLO   =   $02E7
CHART   =   $02F3
CHBAS   =   $02F4
CH      =   $02FC
DDEVIC  =   $0300
DUNIT   =   $0301
DCOMND  =   $0302
DSTATS  =   $0303
DBUFLO  =   $0304
DBUFHI  =   $0305
DTIMLO  =   $0306
DBYTLO  =   $0308
DBYTHI  =   $0309
DAUX1   =   $030A
DAUX2   =   $030B
ICCOM   =   $0342
ICBAL   =   $0344
ICBAH   =   $0345
ICBLL   =   $0348
ICBLH   =   $0349
ICAX1   =   $034A
ICAX2   =   $034B
CIOV    =   $E456
SIOV    =   $E465
SETVBV  =   $E45C
SYSVBV  =   $E45F
XITVBV  =   $E462
OSRETN  =   $E474
CLICK   =   $FCD8
;
; Hardware Registers
;
HPOSP0  =   $D000
HPOSP1  =   $D001
HPOSP2  =   $D002
HPOSP3  =   $D003
M0PF    =   $D000
M1PF    =   $D001
M2PF    =   $D002
M3PF    =   $D003
HPOSM0  =   $D004
HPOSM1  =   $D005
HPOSM2  =   $D006
HPOSM3  =   $D007
P0PF    =   $D004
P1PF    =   $D005
P2PF    =   $D006
P3PF    =   $D007
M0PL    =   $D008
M1PL    =   $D009
M2PL    =   $D00A
M3PL    =   $D00B
P0PL    =   $D00C
P1PL    =   $D00D
P2PL    =   $D00E
P3PL    =   $D00F
SIZEP0  =   $D008
SIZEP1  =   $D009
SIZEP2  =   $D00A
SIZEP3  =   $D00B
SIZEM   =   $D00C
GRAFP0  =   $D00D
GRAFP1  =   $D00E
GRAFP2  =   $D00F
GRAFP3  =   $D010
TRIG0   =   $D010
GRAFM   =   $D011
COLPM0  =   $D012
COLPM1  =   $D013
COLPM2  =   $D014
COLPM3  =   $D015
COLPF0  =   $D016
COLPF1  =   $D017
COLPF2  =   $D018
COLPF3  =   $D019
COLBK   =   $D01A
PRIOR   =   $D01B
VDELAY  =   $D01C
GRACTL  =   $D01D
HITCLR  =   $D01E
CONSOL  =   $D01F
POT0    =   $D200
AUDF1   =   $D200
AUDF2   =   $D202
AUDF3   =   $D204
AUDF4   =   $D206
AUDC1   =   $D201
AUUDC2  =   $D203
AUDC3   =   $D205
AUDC4   =   $D207
AUDCTL  =   $D208
ALLPOT  =   $D208
KBCODE  =   $D209
STIMER  =   $D209
RANDOM  =   $D20A
SERIN   =   $D20A
SKREST  =   $D20A
POT80   =   $D20B
SEROUT  =   $D20C
IRQEN   =   $D20E
IRQST   =   $D20E
SKCTL   =   $D20F
SKSTAT  =   $D20F
PORTA   =   $D300
DMACTL  =   $D400
CHACTL  =   $D401
DLISTL  =   $D402
DLISTH  =   $D403
HSCROL  =   $D404
VSCROL  =   $D405
PMBASE  =   $D407
CHBASE  =   $D409
WSYNC   =   $D40A
VCOUNT  =   $D40B
PENH    =   $D40C
PENV    =   $D40D
NMIEN   =   $D40E
NMIRES  =   $D40F
NMIST   =   $D40F
;
        *=  $0800   ;prog start
;
; ****************************
; *    FIREBUG IGNITION      *
; * Tom Hudson & Kyle Peacock *
; ****************************
;
        .LOCAL
;
BICFLICK
        LDA #>24    ;RELOCATE PROG
        STA HLO+1   ;TO $0800
        LDA #>08    ;FROM $2400
        STA LO+1    ;(DISK ONLY)
        LDA #0
        STA HLO
        STA LO
RELOC1  LDY #0
RELOC2  LDA (HLO),Y
        STA (LO),Y
        DEY
        BNE RELOC2
        INC LO+1
        INC HLO+1
        LDA HLO+1
        CMP #>38
        BNE RELOC1
        JMP DIGIN
DIGIN
        LDA #>22    ;TURN ON...
        STA SDMCTL  ;SCREEN
        LDA #>04    ;NUMBER OF CHARs
        STA MAXBUGS ;TO MANIPULATE.
        LDA #>01    ;STARTING LEVEL
        STA STARTLEVEL ;IS ONE.
IGNITE
        LDA #>01    ;SET UP.
        STA LISTPOINT ;DISPLAY LIST.
        STA HPOSM2  ;PLACE BORDERS
        STA HPOSM3  ;OFFSCREEN.
        STA VSTOP   ;HALT VBLANK.
        JSR SOUNDOFF ;SOUNDS OFF.
        LDA #>06    ;SET ACC.
        LDX #VBLANK/256 ;HI ADDR.
        LDY #VBLANK&255 ;LO ADDR.
        JSR SETVBV  ;VBLANK INIT.
        LDA #>07    ;SET ACC.
        LDX #DBLANK/256 ;HI ADDR
        LDY #DBLANK&255 ;LO ADDR.
        JSR SETVBV  ;DBLANK INIT.
        LDA #>C0    ;SET ACC.
        STA NMIEN   ;ENABLE DLIs
?NOPRESS
        LDA CONSOL  ;GET CONSOLE.
        CMP #7      ;UNTOUCHED?
        BEQ ?PRESS  ;IF =0 BRANCH!
        STA ?PREV   ;SAVE CONSOLE.
        JMP ?NOPRESS ;JUMP ON IT!
?PRESS
        LDA ?PREV   ;PREV. CONSOLE.
        CMP #6      ;START PRESSED?
        BEQ ?VAROOM ;IF =6 BRANCH!
        LDA SKSTAT  ;KEYBOARD KEY
        AND #>04    ;STILL PRESSED?
        BEQ ?NOPRESS ;IF =0 BRANCH!
        LDA CH      ;GET CHARACTER.
        LDX #>08    ;SET INDEX-X.
?NUMCOMP
        CMP ?NUMBASE,X ;IS CHAR A #?
        BEQ ?GOTANUM ;YES! BRANCH!
        DEX         ;NO. RESET PNTR.
        BPL ?NUMCOMP ;HANDLE ALL.
        BMI ?NOPRESS ;NO #s PRESSED.
?GOTANUM
        STX STARTLEVEL ;SAVE START-
        INC STARTLEVEL ;ING LEVEL #.
        STX BCDLVL
        INC BCDLVL
        JSR SHOLVL
        JSR SETCOLORS
        LDA #>FF    ;RESET KEYBOARD
        STA CH      ;CHARACTER.
        JMP ?NOPRESS ;JUMP ON IT.
?VAROOM
        LDA #>00
        STA ?PREV   ;RESET PREVIOUS.
        JSR INIT    ;INITIALIZE
NEWLVL
        LDA #>00    ;CLEAR.
        STA VSTOP   ;START VBLANK.
        STA LISTPOINT ;SET DISPLAY.
MAINLN
;
; GAME PAUSED?
;
        LDA CONSOL  ;CONSOLE BUTTON.
        CMP #>07    ;UNTOUCHED?
        BEQ ?PCHECK ;YES. BRANCH!
        JMP IGNITE  ;RESTART ALL!!!
?PCHECK
        LDA CH      ;IS KEYBOARD
        CMP #>21    ;SPACEBAR?
        BNE ?PAUSE  ;IF NOT, BRANCH!
        LDA SKSTAT  ;STILL HOLDING
        AND #>04    ;DOWN KEY?
        BEQ ?PAUSE  ;IF YES, BRANCH!
        LDA #>01
        STA VSTOP   ;HALT VBLANK.
        STA CH      ;CLEAR CHAR.
        JSR SOUNDOFF ;SOUNDS OFF.
?WAIT
        LDA CH      ;IF SPACEBAR
        CMP #>21    ;PRESSED?
        BNE ?WAIT   ;NO. BRANCH!
?WAITMORE
        LDA SKSTAT  ;IS SPACEBAR
        AND #>04    ;STILL HELD?
        BEQ ?WAITMORE ;YES, BRANCH!
        LDA #>00    ;CLEAR
        STA VSTOP   ;START VBLANK.
        STA CH      ;RESET CHAR.
?PAUSE
        LDA ARMTIM  ;TIMER ELAPSED?
        BNE CHKBUL  ;NO! BRANCH!
        LDA #1      ;RESET TIMER.
        STA ARMTIM
        JSR ARMBOM  ;SUBROUTINE.
CHKBUL
        LDA BULTIM  ;TIMER ELAPSED?
        BNE CHKBUG  ;NO! BRANCH!
        LDA #3      ;RESET TIMER.
        STA BULTIM
        JSR SPARK   ;SUBROUTINE.
CHKBUG  LDA BUGTIM  ;TIMER UP?
        BNE NOFBMV  ;NO! BRANCH!
        JSR MOVBUG  ;SUBROUTINE.
        LDA DIGGIN  ;RESET TIMER
        BEQ ?EIGHT
        LDA #11     ;DIGGING.
        BNE ?BOTSPD
?EIGHT
?BOTSPD LDA #8      ;NOT DIGGING.
        STA BUGTIM  ;SAVE TIMER.
NOFBMV
        LDA GENFLG  ;MATING SEASON?
        BEQ NOGTST  ;YES!
        LDA GENCTR  ;SECOND ELAPSED?
        BNE NOGTST  ;NO!
        LDA #60     ;SET TIMER TO
        STA GENCTR  ;1 SECOND
        DEC GENTIM  ;1 LESS SECOND
        BNE NOGTST  ;NOT ZERO!
        LDA #0      ;TIME TO START
        STA GENFLG  ;MATING SEASON!
NOGTST
        LDA TIMER   ;TIMER ELAPSED?
        BEQ DOENMY  ;YES, DO ENEMIES
        JMP MAINLN  ;NO! BRANCH!
DOENMY  LDA LEVEL   ;RESET ENEMY
        LSR A       ;BUG MOVEMENT
        CMP #16     ;TIMER ACCORDING
        BCC ?BOTNEW ;TO LEVEL OF
        LDA #15     ;GAME.
?BOTNEW
        TAX         ;[LEVEL 30 MAX.]
        LDA ENSPEED,X
        STA TIMER   ;SAVE TIMER.
        JSR ENMOVE  ;MOVE ENEMIES
        JSR BMOVE   ;MOVE BABIES
        LDA ECOUNT  ;ANY ENEMIES?
        BNE GAMECONT ;YES!
        INC LEVEL   ;NO ENEMIES,
        SED         ;ADVANCE LEVEL
        LDA BCDLVL
        CLC
        ADC #1
        STA BCDLVL
        CLD         ;HERE TOM?!?
        JSR SHOLVL  ;SHOW NEW LEVEL
        JSR INIT3   ;RE-INIT
        JMP NEWLVL  ;AND PLAY!
GAMECONT
        JMP MAINLN  ;LOOP BACK!
ENSPEED
        .BYTE 20,19,18,17,16,15
        .BYTE 14,13,12,11,10,9
        .BYTE 8,7,6,5
?NUMBASE
        .BYTE $1F,$1E,$1A,$18,$1D
        .BYTE $1B,$33,$35,$30
?PREV   .BYTE 0
;
; ****************************
; * FIREBUG DEATH/ENEMY MOVER *
; *       BY Tom Hudson      *
; ****************************
;
;FIREBUG DEATH
;
DEAD    LDA BUGL    ;SET EXPLOSION
        STA ELO     ;POSITION = TO
        LDA BUGL+1  ;BUG'S LAST
        STA ELO+1   ;POSITION
        JSR EXPLOD  ;BOOM!!!
        DEC LIVES   ;1 LESS LIFE
        BMI GAMOVR  ;GAME OVER IF <0
        JSR SHOLIV  ;SHOW LIVES
        JSR INIT3   ;RE-INIT LEVEL
        JMP NEWLVL  ;AND RESTART!
;
;GAME OVER!
;
GAMOVR
        JSR INIT3   ;RE-INIT
        JMP IGNITE  ;AND SHOW TITLE
;
;ENEMY MOVER
;
ENMOVE  LDA #0      ;ZERO...
        STA ECOUNT  ;ENEMY COUNT
        LDX #29     ;30 ENEMIES MAX
ENLP    STX ENIX    ;SAVE INDEX
        LDA EACT,X  ;ACTIVE?
        BNE ACTIVE  ;YES!
        JMP NXTEN   ;NO, NEXT ENEMY
ACTIVE  LDA EADL,X  ;GET...
        STA ADL     ;ENEMY'S...
        LDA EADH,X  ;ADDRESS...
        STA ADL+1   ;IN POINTER
        LDY #0      ;GET THE CHAR...
        LDA (ADL),Y ;FROM DISPLAY
        CMP ETYP,X  ;ENEMY OK?
        BEQ ENMYOK  ;YES!
        LDA #0      ;UH-OH, HE'S NO
        STA EACT,X  ;LONGER ACTIVE!
        LDA #12     ;GET ENEMY'S
        SEC         ;SCORE VALUE
```

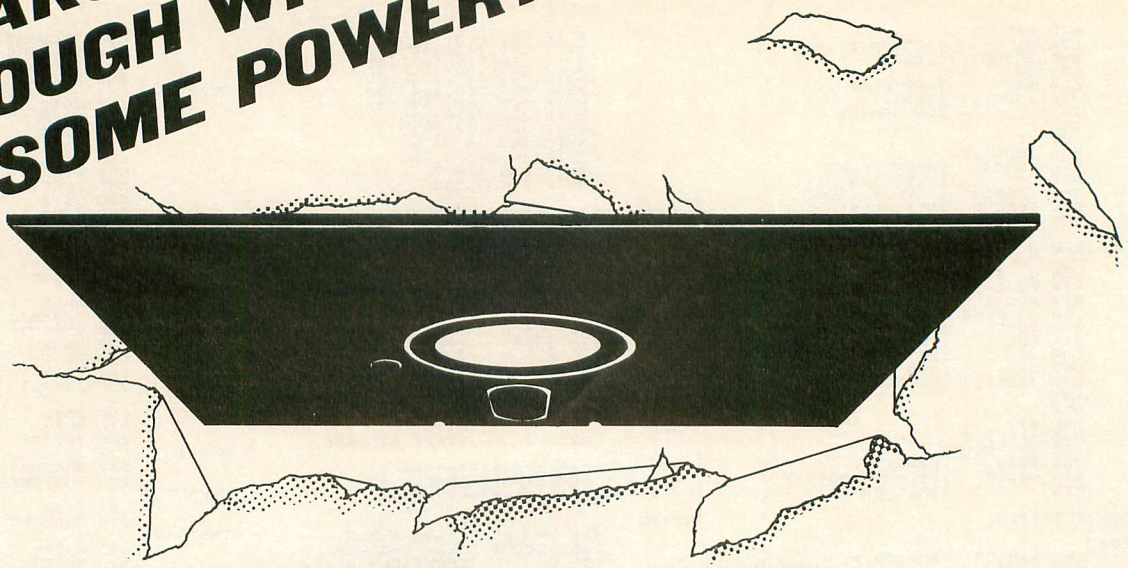
```
        SBC ETYP,X
        AND #$06
        ASL A
        ASL A
        ASL A           ;AND PUT IN...
        STA SCOADD+2    ;ADD VALUE
        JSR ADDSCO      ;ADD TO SCORE
        JMP NXTEN       ;NEXT ENEMY!
ENMYOK  LDA #0          ;RESET MOVEMENT
        STA OK+8        ;TO CHARS 8
        STA OK+10       ;AND 10
        LDA GENFLG      ;MATING SEASON?
        BNE GENDUN      ;NO!
        LDA #1          ;SET...
        LDY ETYP,X      ;ENEMY TYPE'S...
        STA OK,Y        ;OK FLAG
;
;PRIORITY MOVE CHECK
;
GENDUN  LDA #3          ;TRY 4 DIRS.
        STA GNEWIX      ;SAVE IN INDEX
PRILP   LDY GNEWIX      ;GET INDEX
        LDA EADL,X      ;ADD...
        CLC             ;DIRECTION...
        ADC DIRADL,Y    ;OFFSET...
        STA ADL         ;TO...
        LDA EADH,X      ;ENEMY'S...
        ADC DIRADH,Y    ;ADDRESS
        STA ADL+1
        LDY #0          ;LOAD THE BYTE..
        LDA (ADL),Y     ;IN THAT DIR.
        AND #$7F        ;MASK HI BIT
        STA OLDVAL      ;SAVE OLD VALUE
        TAY             ;CHAR # IN Y
        LDA OK,Y        ;CAN WE GO THERE?
        BEQ NOPRI       ;NO!
        LDA CHANCE,Y    ;GET CHANCE
        CMP #$FF        ;100% CHANCE?
        BNE NOPRI       ;NO!
        LDA GNEWIX      ;GET DIRECTION,
        STA EDIR,X      ;SAVE DIRECTION,
        JSR EXMOVE      ;MOVE ENEMY!
        JMP NXTEN       ;NEXT ENEMY
NUPRI   DEC GNEWIX      ;NEXT DIRECTION
        BPL PRILP       ;KEEP CHECKIN'
;
;NO PRIORITY MOVE, SO LET'S TRY
;A RANDOM DIRECTION!
;
ETRYLP  LDA #4          ;TRY 5 DIRS.
        STA ETRY        ;SAVE INDEX
GETDIR  LDA EDIR,X      ;GET OLD DIR.
        BPL SETDIR      ;IT'S MOVING!
        LDA RANDOM      ;RANDOMIZE A
        AND #3          ;DIRECTION
SETDIR  STA EDIR,X      ;SAVE DIR.
        JSR DIRCHK      ;IS THAT DIR OK?
        BPL NXTEN       ;YES!
        DEC ETRY        ;NEXT DIR.
        BPL GETDIR      ;AND LOOP BACK
        LDA #$FF        ;DIDN'T MOVE!
        STA EDIR,X
NXTEN   LDX ENIX        ;GET ENEMY INDEX
        LDA EACT,X      ;ACTIVE?
        BEQ EDEAD       ;NO!
        INC ECOUNT      ;1 MORE ENEMY
EDEAD   DEX             ;NEXT ENEMY
        BMI ENDONE      ;IF <0, FINISHED
        JMP ENLP        ;LOOP BACK.
ENDONE  RTS             ;ALL DONE!
;
;DIRECTION CHECK SUBROUTINE
;
DIRCHK  TAY             ;MOVE DIR. TO Y
        LDA EADL,X      ;NOW ADD...
        CLC             ;DIRECTION...
        ADC DIRADL,Y    ;OFFSET...
        STA ADL         ;TO ENEMY'S...
        LDA EADH,X      ;POSITION
        ADC DIRADH,Y
        STA ADL+1
        LDY #0          ;GET THE BYTE...
        LDA (ADL),Y     ;FROM DISPLAY
        AND #$7F        ;MASK HI BIT
        STA OLDVAL      ;SAVE OLD CHAR
        TAY             ;GET CHAR IN Y
        LDA OK,Y        ;CAN ENEMY MOVE?
        BNE OKMOVE      ;YES!
NOMOVE  LDA #$FF        ;NO!
        RTS             ;RETURN
OKMOVE  LDA CHANCE,Y    ;GET CHANCE
        CMP RANDOM      ;MOVE OK?
        BCC NOMOVE      ;NO!
EXMOVE  LDA EADL,X      ;GET ENEMY'S
        STA HLO         ;OLD ADDRESS
        LDA EADH,X
        STA HLO+1
        LDA #0          ;ERASE OLD...
        TAY             ;ENEMY IMAGE
        STA (HLO),Y
        INY
        STA (HLO),Y
        DEY
        LDA ETYP,X      ;GET ENEMY CHAR,
        STA (ADL),Y     ;PUT ON SCREEN
        INY
        CLC
        ADC #1
        STA (ADL),Y
        LDA ADL         ;UPDATE ENEMY...
        STA EADL,X      ;ADDRESS TABLE
        LDA ADL+1
        STA EADH,X
        LDA OLDVAL      ;GET OLD CHAR
        CMP ETYP,X      ;SAME AS ENEMY?
        BEQ GENIT       ;YES, GENERATE!!
        JMP NOGEN       ;NO, DON'T GEN.
GENIT   LDA #0          ;ENEMY NOT...
        STA EACT,X      ;ACTIVE ANYMORE
;
;THIS CODE GENERATES A NEW
;GROUP OF ENEMY OBJECTS
;
        LDA ETYP,X      ;GET ENEMY TYPE
        CMP #8          ;TYPE 8?
        BNE TYP10       ;NO, TYPE 10!
        LDA #7          ;TURN ON...
        STA CHANGE1     ;CHANGE SOUND 1
        LDA #10         ;GEN TYPE 10
        BNE GNEW        ;GENERATE 'EM!
TYP10   LDA #7          ;TURN ON...
        STA CHANGE2     ;CHANGE SOUND 2
        LDA #8          ;GEN TYPE 8
GNEW    STA NEWTYP      ;SAVE NEW TYPE
        LDA #3          ;GEN. 4 ENEMIES
        STA GNEWIX      ;SAVE IN INDEX
GNEWLP  LDY GNEWIX      ;GET INDEX
        LDA ADL         ;GET ENEMY ADDR,
        CLC             ;ADD...
        ADC DIRADL,Y    ;DIRECTION...
        STA HLO         ;ADDRESS...
        LDA ADL+1       ;OFFSET
        ADC DIRADH,Y
        STA HLO+1
        LDY #0          ;GET THE CHAR...
        LDA (HLO),Y     ;FROM DISPLAY
        AND #$7F        ;MASK HI BIT
        CMP #1          ;BORDER?
        BEQ NONEW       ;CAN'T GEN IT!
        CMP #6          ;FIREBUG?
        BEQ KILL        ;KILL HIM!
        CMP #8          ;ENEMY TYPE 8?
        BEQ NONEW       ;CAN'T GEN!
        CMP #10         ;ENEMY TYPE 10?
        BEQ NONEW       ;CAN'T GEN!
        CMP #12         ;BABY BUG?
        BEQ KILL        ;KILL IT!
        CMP #15         ;BOMB?
        BEQ NONEW       ;CAN'T GENERATE
        LDX #29         ;SCAN FOR...
SCANEW  LDA EACT,X      ;INACTIVE ENEMY
        BEQ GOTNEW      ;GOT ONE!
        DEX             ;TRY NEXT SLOT
        BPL SCANEW      ;KEEP TRYING
        LDA #1          ;NONE AVAILABLE!
        STA GENFLG      ;MATING OVER
        LDA #100        ;WAIT 100...
        STA GENTIM      ;SECONDS.
        JMP MOVEOK      ;PROCEED
GOTNEW  LDA #1          ;SET ENEMY AS...
        STA EACT,X      ;ACTIVE
        LDA #$FF        ;SET UP FOR...
        STA EDIR,X      ;NO DIRECTION
        LDA HLO         ;SAVE ENEMY...
        STA EADL,X      ;ADDRESS
        LDA HLO+1
        STA EADH,X
        LDA NEWTYP      ;GET NEW TYPE
        STA ETYP,X      ;SAVE IN TABLE
        STA (HLO),Y     ;PUT ENEMY...
        CLC             ;ON SCREEN!
        ADC #1
        INY
        STA (HLO),Y
NONEW   DEC GNEWIX      ;NEXT GEN.
        BMI MOVEOK      ;ALL DONE!
        JMP GNEWLP      ;LOOP IF MORE
MOVEOK  LDA #0          ;STATUS=0 (OK)
        RTS             ;RETURN!
NOGEN   CMP #6          ;HIT FIREBUG?
        BNE MOVEOK      ;NO, MOVE OK.
KILL    PLA             ;REMOVE...
        PLA             ;OLD...
        PLA             ;STACK...
        PLA             ;ENTRIES,
        JMP DEAD        ;KILL BUG!!!
;
; ***************************
; * FIREBUG MOVEMENT ROUTINE *
; *    BY Tom Hudson        *
; ***************************
;
MOVBUG  LDA BUGL        ;GET AND SAVE...
        STA HLO         ;FIREBUG ADDRESS
        LDA BUGL+1
        STA HLO+1
        LDY #0          ;GET THAT...
        LDA (HLO),Y     ;DISPLAY BYTE
        CMP #6          ;BUG OK?
        BEQ BUGLIV      ;YUP!
        BNE KILBUG      ;UH-OH! KILL HIM
BUGLIV
        LDX STICK0      ;GET STICK
        LDY SDIR,X      ;AND DIRECTION
        BPL TRYBMV      ;TRY TO MOVE
NOBMV   RTS             ;NO MOVE!
TRYBMV  LDA BUGL        ;ADD...
        CLC             ;DIRECTION...
        ADC DIRADL,Y    ;OFFSET...
        STA ADL         ;TO...
        LDA BUGL+1      ;BUG'S...
        ADC DIRADH,Y    ;ADDRESS
        STA ADL+1
        LDY #0          ;GET THAT...
        LDA (ADL),Y     ;DISPLAY BYTE
        CMP #1          ;BORDER?
        BEQ NOBMV       ;CAN'T GO THERE!
        CMP #12         ;BABY?
        BEQ NOBMV       ;CAN'T GO THERE!
        CMP #15         ;BOMB?
        BCS NOBMV       ;CAN'T GO THERE!
        CMP #6          ;DIRT/TRAIL?
        BCC OKBMV       ;GO AHEAD!
        CMP #14         ;BUG TRAIL?
        BEQ OKBMV       ;GO AHEAD!
KILBUG  PLA             ;MUST BE ENEMY!
        PLA             ;CLEAR STACK,
        JMP DEAD        ;KILL BUG!
OKBMV   CMP #0          ;ENEMY TRAIL?
        BEQ NOTDIG      ;NO DIGGING!
        CMP #14         ;BUG TRAIL?
        BEQ NOTDIG      ;NO DIGGING!
        LDA #1          ;SET DIGGING...
        STA DIGGIN      ;SOUND FLAG
        BNE DRWBUG      ;AND DRAW BUG
NOTDIG  LDA #0          ;TURN OFF...
        STA DIGGIN      ;DIGGING SOUND
DRWBUG  LDA #14         ;ERASE BUG...
        STA (HLO),Y     ;WITH BUG TRAIL
        INY
        STA (HLO),Y
        DEY
        LDA #6          ;DRAW BUG...
        STA (ADL),Y     ;IN NEW...
        INY             ;LOCATION
        CLC
        ADC #1
        STA (ADL),Y
        LDA ADL         ;UPDATE BUG'S
        STA BUGL        ;ADDRESS
        LDA ADL+1
        STA BUGL+1
        RTS             ;ALL DONE!
; ****************************
; * FIREBUG INTERRUPTS & SOUND *
; *      BY Kyle Peacock     *
; ****************************
;
        .LOCAL
;
VBLANK
        CLD             ;CHILL DECIMAL.
;
; RANDOMIZE DETONATION
;
        LDX #7          ;CHANGE 8 BYTES.
?JUNKIT
        LDA RANDOM      ;RANDOM #
        AND #$55        ;MASK FOR PF#0.
        STA GAMESET+[8*17],X ;SAVE.
        DEX             ;HANDLE NEXT.
        BPL ?JUNKIT     ;NOT DONE-BRANCH
;
; SET UP CORRECT DISPLAY LIST
;
        LDA LISTPOINT   ;D-LIST PNTR.
        ASL A           ;MULTIPLY BY 2.
        TAX             ;MOVE A TO X.
        LDA ?DLISTBASE,X ;LET'S
        STA SDLSTL      ;PAINT THE
        LDA ?DLISTBASE+1,X ;GAME-
        STA SDLSTL+1    ;GREEN, & EAT
        LDA #GAMEDLI&255 ;FRANKS &
        STA VDSLST      ;BEANS.
        LDA #GAMEDLI/256 ;C.B.I. IS
        STA VDSLST+1    ;ON THE MOVE.
;
; CHANGE TITLE SCREEN
;
        DEC ?SWITCHTIME ;DEC TIMER.
        BPL ?SWITCHDONE ;>0 BRANCH!
        LDA RANDOM      ;RANDOM #
        AND #$3F        ;MASK BITS 7 & 6
        ORA #$0F        ;ADD BITS 0-3.
        STA ?SWITCHTIME ;SAVE TIMER.
        INC ?FLOPFLIP   ;INC NAME
        LDA ?FLOPFLIP   ;TOGGLE FLAG.
        AND #$01        ;EXAMINE BIT 0.
        ASL A           ;MULTIPLY BY 2.
        ASL A           ;MULTIPLY BY 4.
        TAX             ;MOVE A TO X.
        LDA ?SWITCHBASE,X ;GET ADDR
        STA SWITCH1     ;OF NAME DATA
        LDA ?SWITCHBASE+1,X ;AND
        STA SWITCH1+1   ;PLACE INTO
        LDA ?SWITCHBASE+2,X ;PROPER
        STA SWITCH2     ;AREA OF DISPLAY
        LDA ?SWITCHBASE+3,X ;LIST.
        STA SWITCH2+1
?SWITCHDONE
;
; HALT ATTRACT MODE?
;
        LDA VSTOP       ;VBLANK HALTED?
        BNE ?VDONE      ;YES!
        STA ATRACT      ;ATTRACT MODE.
;
; SPARK REGENERATION
;
        DEC ?REGEN      ;TIME UP?
        BPL ?REGEND     ;IF >0 BRANCH!
        LDA STICK0      ;GET JOYSTICK.
        CMP #15         ;CENTERED?
        BNE ?GENSET1    ;NO! BRANCH!
        LDX #$00        ;CLEAR INDEX.
        BEQ ?GENSET2    ;BRANCH!
?GENSET1
        LDX DIGGIN      ;BUG IN DIRT?
?GENSET2
        LDA ?GENSET,X   ;REGENERATION RATE
        STA ?REGEN      ;SAVE TIMER.
        INC ?FLIPFLOP   ;INC TOGGLE
        LDA ?FLIPFLOP   ;FLAG.
        AND #$01        ;EXAMINE BIT 0.
        BEQ ?REGEND     ;IF EVEN-BRANCH!
        LDX BULLPNT     ;GET POINTER.
        LDA BULLETLINE,X ;DETERMINE
        CMP #20         ;IF WE SHOULD
        BNE ?BULLSET    ;MODIFY THIS
        CPX #29         ;CHARACTER OR
        BEQ ?REGEND     ;MOVE TO NEXT.
        INC BULLPNT     ;MOVE TO NEXT
        INX             ;CHARACTER.
?BULLSET
        DEC BULLETLINE,X ;MODIFY.
?REGEND
;
; DECREMENT COUNT DOWN TIMERS
;
        LDA GENCTR
        BEQ ?NOGENT
        DEC GENCTR
?NOGENT
        LDA BULTIM      ;TIMER = 0?
        BEQ ?NOBUL      ;YES. BRANCH!
        DEC BULTIM      ;NO. DECREMENT.
?NOBUL
        LDA ARMTIM      ;TIMER = 0?
        BEQ ?NOAT       ;YES. BRANCH!
        DEC ARMTIM      ;NO. DECREMENT.
?NOAT
        LDA TIMER       ;TIMER = 0?
        BEQ ?NOTIME     ;YES. BRANCH!
        DEC TIMER       ;NO. DECREMENT.
?NOTIME
        LDA BUGTIM      ;TIMER = 0?
        BEQ ?NOBTIM     ;YES. BRANCH!
        DEC BUGTIM      ;NO. DECREMENT.
?NOBTIM
;
; FINE SCROLLING (REVISITED)
;
        JSR SCROLL
?VDONE
;
; UPDATE LMS POINTERS
;
        JSR SHIFTER
;
; VERTICAL BLANK DONE
;
        JMP SYSVBV
;
; ****************************
; * DEFERRED VERTICAL BLANK *
; ****************************
;
DBLANK
        CLD             ;CHILL DECIMAL
;
; UPDATE CHARACTER SET
;
        JSR DRAW
        LDA VSTOP       ;DBLANK HALTED?
        BNE ?DDONE      ;IF <>0 BRANCH!
;
; DIGGING SOUND
;
        JSR DIGSOUND
;
; BUG TRANSFORMATION SOUND
;
        JSR TRANSFORM
;
; FIRE EGG EATEN SOUND
;
        JSR CHOMP
;
```

```
; SPARK FIRING SOUND
;
        JSR POPGUN
;
; DETONATION SOUND
;
        JSR KABOOM
;
; BOMB DETONATION SOUND
;
        JSR BOOMBOX
;
; DEF. VERTICAL BLANK DONE
;
?DDONE
        JMP XITVBV
;
; POINTERS' DATABASES
;
?DLISTBASE
        .WORD GAMELIST
        .WORD TITLELIST
?SWITCHBASE
        .WORD TITLE2
        .WORD TITLE4
        .WORD TITLE4
        .WORD TITLE2
;
; ****************
; * FIREBUG DLI= *
; ****************
;
GAMEDLI1
        PHA             ;SAVE ACC.
        TXA             ;MOVE X TO A.
        PHA             ;SAVE ACC.
        TYA             ;MOVE Y TO A.
        PHA             ;SAVE ACC.
        LDA #GAMESET/256 ;BUG CHR=
        LDX ##3F        ;DMA CONTROL.
        LDY NEWBACK     ;BACKGROUND.
        STA WSYNC       ;SAVE IT.
        STX DMACTL      ;SAVE IT.
        STY COLBK       ;SAVE IT.
        STA CHBASE      ;SAVE IT.
        LDA #GAMEDLI2&255 ;LOAD UP
        STA VDSLST      ;FOR NEXT DLI.
        LDA #GAMEDLI2/256 ;SAVE NEXT
        STA VDSLST+1    ;DLI ADDRESS.
        PLA             ;RESTORE ACC.
        TAY             ;MOVE A TO Y.
        PLA             ;RESTORE ACC.
        TAX             ;MOVE A TO X.
        PLA             ;RESTORE ACC.
        RTI             ;BOGGIE BABY...
GAMEDLI2
        PHA             ;SAVE ACC.
        LDA COLOR4      ;BACKGROUND.
        STA WSYNC       ;SAVE IT.
        STA COLBK       ;SAVE IT.
        LDA SDMCTL      ;OLD DMA CONTROL
        STA DMACTL      ;IS RESTORED.
        LDA #GAMEDLI3&255 ;SET UP
        STA VDSLST      ;FOR NEXT
        LDA #GAMEDLI3/256 ;DLI ADDR.
        STA VDSLST+1    ;SAVE IT.
        PLA             ;RESTORE ACC.
        RTI             ;BOGGIE BABY...
GAMEDLI3
        PHA             ;SAVE ACC.
        LDA ##E0        ;OLD CHR SET.
        STA WSYNC       ;SAVE IT.
        STA CHBASE      ;SAVE IT.
        PLA             ;RESTORE ACC.
        RTI             ;BOGGIE BABY...
;
; DIGGING SOUND ROUTINE
;
DIGSOUND
        LDA DIGGIN      ;BUG IN DIRT?
        BEQ ?DIGOFF     ;IF = 0 BRANCH!
        DEC ?DIGTIME    ;DEC TIMER.
        BPL ?DIGDONE    ;IF >0 BRANCH!
        LDA ##03        ;RESET DIGGING
        STA ?DIGTIME    ;TIMER.
        LDX PLAYNO      ;PLAYER NUMBER.
        LDA STICK0,X    ;READ JOYSTICK.
        CMP #15         ;CENTERED?
        BNE ?DIGON      ;<> 15 - BRANCH!
?DIGOFF
        LDA ##00        ;CLEAR.
        STA AUDC1       ;SOUND CHANNEL.
        BEQ ?DIGDONE    ;BRANCH!
?DIGON
        LDA ##46        ;SET UP
        STA AUDC1       ;SOUND CHANNEL 1
        LDA #220        ;SET UP
        STA AUDF1       ;SOUND FREQ 1.
?DIGDONE
        RTS             ;BEAM ME UP...
;
; BUG TRANSFORMATION SOUND
;
TRANSFORM
        LDX CHANGE1     ;1st SOUND ON?
        BPL TRAN1       ;IF >0 BRANCH!
        LDA ##00        ;SOUND OFF.
        STA AUDC2       ;CHANNEL 2.
        BEQ TRAN2       ;BRANCH!
TRAN1
        LDA ?CHANGE1,X  ;GET CORRECT
        STA AUDF2       ;FREQ & SAVE.
        LDA ##A4        ;SET UP CORRECT
        STA AUDC2       ;SOUND CHANNEL.
        DEC CHANGE1     ;DEC POINTER.
TRAN2
        LDX CHANGE2     ;2nd SOUND ON?
        BPL TRAN3       ;IF >0 BRANCH!
        LDA CHANGE1     ;SOUND 1 OFF?
        BPL TRAN4       ;IF >0 BRANCH!
?DEACT2
        LDA ##00        ;SOUND OFF.
        STA AUDC2       ;CHANNEL 2.
        BEQ TRAN4       ;BRANCH!
TRAN3
        LDA ?CHANGE2,X  ;GET CORRECT
        STA AUDF2       ;FREQ & SAVE.
        LDA ##A4        ;SET UP CORRECT
        STA AUDC2       ;SOUND CHANNEL.
        DEC CHANGE2     ;DEC POINTER.
TRAN4
        RTS             ;BAMF!
?CHANGE1
        .BYTE 240,160,80,0
        .BYTE 15,95,175,255
?CHANGE2
        .BYTE 210,140,70,0
        .BYTE 45,115,185,255
;
; FIRE EGG EATEN SOUND
;
CHOMP
        LDX EATNOISE    ;SOUND ON?
        BPL ?CHOMPON    ;IF >0 BRANCH!
        LDA CHANGE1     ;CHANGE 1 ON?
        BPL ?CHOMPOFF   ;IF >0 BRANCH!
        LDA CHANGE2     ;CHANGE 2 ON?
        BMI ?DEACT2     ;IF >0 BRANCH!
?CHOMPOFF
        RTS             ;BOGGIE...
?CHOMPON
        LDA ?CHOMPF,X   ;GET SOUND
        STA AUDF2       ;FREQ & SAVE.
        LDA ##A6        ;SET CORRECT
        STA AUDC2       ;SOUND CHANNEL.
        DEC EATNOISE    ;DEC POINTER.
        RTS             ;LATER Y'ALL!
?CHOMPF
        .BYTE 0,8,16,24,32
        .BYTE 40,47,55,63,71
;
; SPARK FIRING SOUND
;
POPGUN
        LDX BULLNOISE   ;SOUND ON?
        BPL ?SHOOTIT    ;IF >0 BRANCH!
?CHILLOUT
        LDA ##00        ;CLEAR SOUND
        STA AUDC3       ;CHANNEL 3.
        RTS             ;LATER...
?SHOOTIT
        LDA ?SHOOTF,X   ;GET CORRECT
        STA AUDF3       ;FREQ & SAVE.
        LDA ##AA        ;SET UP CORRECT
        STA AUDC3       ;SOUND CHANNEL.
        DEC BULLNOISE   ;DEC POINTER.
?SHOTIT
        RTS             ;LATER...
?SHOOTF
        .BYTE 18,57,97,136
        .BYTE 176,215,255
;
; DETONATION SOUND
;
KABOOM
        LDX BOOMNOISE   ;SOUND ON?
        BPL ?BOOMIT     ;IF >0 BRANCH!
        LDA BULLNOISE   ;BULLET ON?
        BMI ?CHILLOUT   ;IF <0 BRANCH!
        RTS             ;LATER...
?BOOMIT
        LDA ?BOOMF,X    ;GET CORRECT
        STA AUDF3       ;FREQ & SAVE.
        LDA ##8C        ;GET CORRECT
        STA AUDC3       ;SOUND CHANNEL.
        DEC BOOMNOISE   ;DEC POINTER.
?BOOMED
        RTS             ;LATER (MUCH!)
?BOOMF
        .BYTE 244,232,220,209
        .BYTE 197,186,174,162
        .BYTE 151,139,128,116
        .BYTE 104,93,81,70,58
        .BYTE 46,35,23,12,0
;
; BOMB DETONATION SOUND
;
BOOMBOX
        LDX EXNOISE     ;SOUND ON?
        BPL ?ZAPON      ;IF >0 BRANCH!
        LDA ##30        ;ADJUST DISPLAY
        JSR ?DSET       ;LIST.
        LDA BOOMNOISE   ;EXPLOSION?
        BPL ?ZAPOFF     ;IF >0 BRANCH!
        LDA BULLNOISE   ;BULLET?
        BPL ?ZAPOFF     ;IF >0 BRANCH!
        JMP ?CHILLOUT   ;CHILLY BE.
?ZAPOFF
        RTS             ;CHILL OUT.
?ZAPON
        LDA EXF,X       ;GET CORRECT
        STA AUDF3       ;FREQ & SAVE.
        LDA ##8A        ;GET CORRECT
        STA AUDC3       ;SOUND CHANNEL.
        DEC EXNOISE     ;DEC POINTER.
        LDA RANDOM      ;RANDOM #
        AND ##70        ;MASK 4,5,6.
?DSET
        STA GAMELIST    ;SAVE INTO DIS-
        STA GAMELIST+1  ;PLAY LIST.
        RTS             ;CHILLY BE Y'ALL
EXF
        .BYTE 0,253,249,245,241,237
        .BYTE 234,230,226,222,218
        .BYTE 214,211,207,203,199
        .BYTE 195,191,188,184,180
        .BYTE 176,172,168,165,161
        .BYTE 157,153,149,145,142
        .BYTE 138,134,130,126,123
        .BYTE 119,115,111,107,103
        .BYTE 100,96,92,88,84,80
        .BYTE 77,73,69,65,61,57,54
        .BYTE 50,46,42,38,34,31,27
        .BYTE 23,19,15,11,8,4
?GENSET
        .BYTE 120,60
?DIGTIME .BYTE 0
?REGEN  .BYTE 0
?FLIPFLOP .BYTE 0
?FLOPFLIP .BYTE 0
?SWITCHTIME .BYTE 0
; ************************
; * FIREBUG DISPLAY LISTS *
; *   BY Kyle Peacock    *
; ************************
;
        .LOCAL
;
; GAME BOARD DISPLAY LIST
;
GAMELIST
        .BYTE #30,#30,#30
        .BYTE #30,#30,#30
        .BYTE #C6
        .WORD STATUSLINE
        .BYTE #44
        .WORD BLANKLINE
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
DLISTSTART
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
DLISTEND
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #74,0,0
        .BYTE #C4
        .WORD BLANKLINE
        .BYTE #00,#C4
        .WORD BULLETLINE
        .BYTE #20,#06,#41
        .WORD GAMELIST
;
; TITLE SCREEN DISPLAY LIST
;
TITLELIST
        .BYTE #70,#70,#70,#70,#70
        .BYTE #47
        .WORD STATUSLINE
        .BYTE #70,#70,#46
        .WORD TITLE1
        .BYTE #70,#47
SWITCH1
        .WORD TITLE4
        .BYTE #70,#46
        .WORD TITLE3
        .BYTE #70,#47
SWITCH2
        .WORD TITLE2
        .BYTE #70,#70,#70,#46
        .WORD TITLE5
        .BYTE #70,#46
        .WORD BULLETLINE+40
        .BYTE #41
        .WORD TITLELIST
;
; BORDER LINE CHARACTERS
;
BLANKLINE
        .BYTE 1,1,1,1,1,1,1,1,1,1,1
        .BYTE 1,1,1,1,1,1,1,1,1,1,1
        .BYTE 1,1,1,1,1,1,1,1,1,1,1
        .BYTE 1,1,1,1,1,1,1,1,1,1,1
        .BYTE 1,1,1,1,1,1,1,1,1
;
; SPARK BAR GRAPH
;
BULLETLINE
        .BYTE 0,0,0,0,0,0,0,0,0,0,0
        .BYTE 0,0,0,0,0,0,0,0,0,0,0
        .BYTE 0,0,0,0,0,0,0,0,0,0,0
        .BYTE 0,0,0,0,0,0,0,0,0,0,0
;
; CREDITS & WHAT-NOT
;
SCOLIN
        .SBYTE +#40,"       LVL:"
        .SBYTE +#80,"   BUGS: "
STATUSLINE
        .SBYTE "      fire"
        .SBYTE " bug     "
TITLE1
        .SBYTE "     WRITT"
        .SBYTE "EN BY     "
TITLE2
        .SBYTE "     TOM H"
        .SBYTE "UDSON     "
TITLE3
        .SBYTE "          &"
        .SBYTE "          "
TITLE4
        .SBYTE "    kyle p"
        .SBYTE "eacock    "
TITLE5
        .SBYTE "  analog c"
        .SBYTE "omputing  "
; ************************
; * DISPLAY LIST MODIFIER *
; *   BY Kyle Peacock    *
; ************************
;
        .LOCAL
;
SHIFTER
        LDY #13         ;D-LIST INDEX.
        LDA #19         ;# OF LINES TO
        STA ?COUNT      ;MODIFY LMS=.
        LDA YPOINT      ;SET Y-COORD.
        ASL A           ;MULTIPLY BY 2.
        TAX             ;USE AS INDEX.
?SET
        LDA SCREENBYTES+1,X ;END OF
        CMP ##FF        ;SCREEN RAM?
        BNE ?PASS1      ;YES! BRANCH!
        LDX ##00        ;CLEAR X-REG.
?PASS1
        LDA SCREENBYTES,X ;SET
        CLC             ;SCREEN RAM &
        ADC XPOINT      ;ADD X-COORD.
        STA GAMELIST,Y  ;& SAVE.
        LDA SCREENBYTES+1,X ;ALSO
        ADC ##00        ;ADD TO HI-BYTE
        STA GAMELIST+1,Y ;& SAVE.
        INX             ;INCREMENT POIN-
        INX             ;TERS FOR NEXT
        TYA             ;LMS OPERAND
        CLC             ;TRANSFER.
        ADC ##03
        TAY
        DEC ?COUNT      ;ALL LINES DONE?
        BPL ?SET        ;NO! BRANCH!
        RTS             ;LATER Y'ALL!!!
?COUNT
        .BYTE 0         ;COUNTER
;
; ****************************
; * FIREBUG SCREEN MEMORY MAP *
; ****************************
;
SCREENBYTES
        .WORD SCREENBASE+[0*128]
        .WORD SCREENBASE+[1*128]
        .WORD SCREENBASE+[2*128]
        .WORD SCREENBASE+[3*128]
        .WORD SCREENBASE+[4*128]
        .WORD SCREENBASE+[5*128]
        .WORD SCREENBASE+[6*128]
        .WORD SCREENBASE+[7*128]
        .WORD SCREENBASE+[8*128]
        .WORD SCREENBASE+[9*128]
        .WORD SCREENBASE+[10*128]
        .WORD SCREENBASE+[11*128]
        .WORD SCREENBASE+[12*128]
        .WORD SCREENBASE+[13*128]
```

```
        .WORD SCREENBASE+[14*128]
        .WORD SCREENBASE+[15*128]
        .WORD SCREENBASE+[16*128]
        .WORD SCREENBASE+[17*128]
        .WORD SCREENBASE+[18*128]
        .WORD SCREENBASE+[19*128]
        .WORD SCREENBASE+[20*128]
        .WORD SCREENBASE+[21*128]
        .WORD SCREENBASE+[22*128]
        .WORD SCREENBASE+[23*128]
        .WORD SCREENBASE+[24*128]
        .WORD SCREENBASE+[25*128]
        .WORD SCREENBASE+[26*128]
        .WORD SCREENBASE+[27*128]
        .WORD SCREENBASE+[28*128]
        .WORD SCREENBASE+[29*128]
        .WORD SCREENBASE+[30*128]
        .WORD SCREENBASE+[31*128]
        .WORD SCREENBASE+[32*128]
        .WORD SCREENBASE+[33*128]
        .WORD SCREENBASE+[34*128]
        .WORD SCREENBASE+[35*128]
        .WORD SCREENBASE+[36*128]
        .WORD SCREENBASE+[37*128]
        .WORD SCREENBASE+[38*128]
        .WORD SCREENBASE+[39*128]
        .WORD SCREENBASE+[40*128]
        .WORD SCREENBASE+[41*128]
        .WORD SCREENBASE+[42*128]
        .WORD SCREENBASE+[43*128]
        .WORD SCREENBASE+[44*128]
        .WORD SCREENBASE+[45*128]
        .WORD SCREENBASE+[46*128]
        .WORD SCREENBASE+[47*128]
        .WORD SCREENBASE+[48*128]
        .WORD SCREENBASE+[49*128]
        .WORD SCREENBASE+[50*128]
        .WORD SCREENBASE+[51*128]
        .WORD SCREENBASE+[52*128]
        .WORD SCREENBASE+[53*128]
        .WORD SCREENBASE+[54*128]
        .WORD SCREENBASE+[55*128]
        .WORD SCREENBASE+[56*128]
        .WORD SCREENBASE+[57*128]
        .WORD SCREENBASE+[58*128]
        .WORD SCREENBASE+[59*128]
        .WORD SCREENBASE+[60*128]
        .WORD SCREENBASE+[61*128]
        .WORD SCREENBASE+[62*128]
        .WORD SCREENBASE+[63*128]
        .WORD $FFFF
; ****************************
; *  FIREBUS INITIALIZATION  *
; * Tom Hudson & Kyle Peacock *
; ****************************
;
; ************************
; * ONE SHOT INITIALIZER *
; ************************
;
        .LOCAL
;
INIT
;
;SET UP CHARACTERS
;
        LDA #$04    ;CHSET START
        STA HLO+1   ;PUT IN POINTER
        LDX #13     ;14 CHARS...
        STX XTEMP   ;TO LOAD
CHLOOP
        LDX XTEMP   ;GET CHAR #
        LDA CSTRT,X ;AND START ADDR
        STA HLO     ;PUT IN POINTER
        TXA         ;NOW...
        ASL A       ;MULT...
        ASL A       ;POINTER...
        ASL A       ;BY 8,
        TAX         ;USE AS INDEX
        LDY #0      ;START CHAR COPY
CHL2
        LDA CDATA,X ;GET CHAR DATA
        STA (HLO),Y ;MOVE TO CHSET
        INX         ;NEXT CHAR BYTE
        INY         ;NEXT SET BYTE
        CPY #8      ;DONE 8?
        BNE CHL2    ;NO, LOOP BACK
        DEC XTEMP   ;NEXT CHAR
        BPL CHLOOP  ;LOOP IF MORE
;
; GENERAL VARIABLE SETTING
;
        LDA #$00    ;CLEAR
        STA PLAYNO  ;PLAYER #.
        STA VBIT    ;RAM VSCROL
        STA HBIT    ;RAM HSCROL
        STA MINX    ;MIN. X-COORD.
        LDX #2
CSCLP   STA SCORE,X ;SCORES
        STA SCOADD,X ;& STUFF.
        DEX
        BPL CSCLP
        LDA #$01    ;SET W/ 1.
        STA MINY    ;MIN Y-COORD.
        STA FIRETIME ;FIRING.
        LDA #84
        STA MAXX    ;MAX X-COORD.
        LDA #43
        STA MAXY    ;MAX Y-COORD.
        LDA #$03    ;CHARACTER UP-
        STA MOVETIME ;DATA TIMER.
        STA PHASETIME
        LDX #$04    ;SET UP.
        STX LIVES   ;# OF LIVES
        LDA STARTLEVEL
        STA LEVEL   ;STARTING LEVEL
        STA BCDLVL  ;STARTING LEVEL
?TYPESET
        LDA ?TYPEBASE,X ;CHARACTER
        STA TYPE,X  ;TYPE SETTING.
        DEX         ;SET THEM ALL.
        BPL ?TYPESET
;
; PLAYER/MISSILE INITIALIZATION
;
        LDA #$3E
        STA SDMCTL  ;DMA.
        LDA #PLAYBS/256
        STA PMBASE  ;ADDRESS.
        LDA #$03    ;GRAPHICS
        STA GRACTL  ;CONTROL.
        LDA #$21    ;PRIORITY
        STA GPRIOR  ;REG.
        LDA #$F0    ;MISSILES 2 & 3
        STA SIZEM   ;ARE 4X SIZE.
        LDA #$28    ;MISSILE 2 X.
        STA HPOSM0+2
        LDA #$CC    ;MISSILE 3 X.
        STA HPOSM0+3
        LDA #$00    ;CLEAR OUT
        TAX
?PMSET
        STA MISS,X  ;MISSILES.
        CPX #$C9    ;INSTALL
        BCS ?PMSET2 ;PLAYFIELD
        CPX #$30    ;BORDER
        BCC ?PMSET2 ;INTO MISSILES
        LDA #$F0
        STA MISS,X  ;2 AND 3.
        LDA #$00
?PMSET2
        DEX         ;ALL DONE?
        BNE ?PMSET  ;NO! CONT.
;
; ************************
; * MULTIPLE INITIALIZER *
; ************************
;
INIT2
        JSR SOUNDOFF
        LDX #$01
        STX VSTOP   ;HALT VBLANK.
        STX CH      ;STUFF CHAR.
;
; GENERAL VARIABLE SETTING
;
        DEX
        DEX
        STX CHANGE1 ;TURN OFF
        STX CHANGE2 ;ALL SOUND
        STX BULLNOISE ;RAM VARIA-
        STX BOOMNOISE ;BLES.
        STX EXNOISE
        STX EATNOISE
        LDA MINX    ;SET LMS OPER-
        STA XPOINT  ;ANDS TO UPPER
        LDA MINY    ;LEFT OF BOARD.
        STA YPOINT
        LDA #$00    ;NO SPARKS ARE
        STA NOBULL  ;ACTIVE.
        LDX #$03
?SETBULL
        LDA #$FF    ;CLEAR SPARK
        STA BULLDIR,X ;WORK AREA.
        DEX
        BPL ?SETBULL
        LDX #19     ;SET SPARK LINE
        LDA #20     ;TO FULL
?SETBULL2 ;         CAPACITY.
        STA BULLETLINE+10,X
        DEX
        BPL ?SETBULL2
        LDA #29     ;SET BULLET
        STA BULLPNT ;POINTER.
;
; FILL PLAY AREA WITH DIRT
;
        LDX #$00    ;CLEAR X-REG.
?FILL1
        TXA         ;MOVE X TO A.
        ASL A       ;MULTIPLY BY 2.
        TAY         ;MOVE A TO Y.
        LDA SCREENBYTES,Y
        STA LO      ;GET SCREEN
        LDA SCREENBYTES+1,Y
        STA LO+1    ;ADDRESS.
        CMP #$FF    ;AT RAM END?
        BEQ ?FILL3  ;YES! BRANCH!
        LDY #127    ;SET FOR FILL.
?FILL2
        LDA #$04    ;GET RANDOM #
        JSR RANDO   ;(0 UP TO ACC-1)
        CLC         ;ADD TWO TO
        ADC #$02    ;VALUE IN ACC.
        STA (LO),Y  ;STORE IT.
        DEY         ;HANDLE NEXT.
        BPL ?FILL2  ;BRANCH!
        INX         ;NEXT RAM AREA.
        JMP ?FILL1  ;JUMP ON IT!
?FILL3
;
; PLACE PRE-PROGRAMMED TUNNELS
;
        LDX #$00    ;CLEAR X-REG.
?VECSET
        LDY #$00    ;CLEAR Y-REG.
?VECSET2
        LDA ?VECTORS,X ;GET VECTOR
        STA ?INITX,Y ;INITIAL X.
        INX         ;GET ADDITIONAL
        INY         ;VECTOR INFO.
        CPY #$06    ;LAST DATA BYTE?
        BNE ?VECSET2 ;NO! BRANCH!
?VECPLOT
        LDA ?INITX  ;GET INITIAL X.
        STA XCOORD  ;SAVE IT.
        LDA ?INITY  ;GET INITAIL Y.
        STA YCOORD  ;SAVE IT.
        JSR XYFIND  ;FIND RAM LOC.
        LDY #$00    ;CLEAR Y-REG.
        TYA         ;MOVE Y TO A.
        STA (LO),Y  ;STORE IN MEM.
        LDA ?INITX  ;IS X VECTOR AT
        CMP ?FINALX ;DESTINATION?
        BNE ?VECRESET ;NO! BRANCH!
        LDY ?INITY  ;IS Y VECTOR AT
        CPY ?FINALY ;DESTINATION?
        BEQ ?NEXTVEC ;YES! BRANCH!
?VECRESET
        CLC         ;ADD X-DELTA
        ADC ?DELTAX ;TO X VECTOR.
        STA ?INITX  ;SAVE IT.
        LDA ?INITY  ;ADD Y-DELTA
        CLC         ;TO Y VECTOR.
        ADC ?DELTAY
        STA ?INITY  ;SAVE IT.
        JMP ?VECPLOT ;PLOT NEXT.
?NEXTVEC
        CPX #[6*6]  ;6 VECTORS
        BCC ?VECSET ;NEXT VECTOR.
;
; PLACE PROXIMITY BOMBS
;
        LDX LEVEL   ;GET LEVEL #.
        CPX #11     ;IS IT >10 ?
        BCC ?NUMBMB ;NO! BRANCH!
        LDX #10     ;SET 10 AS MAX.
?NUMBMB
        LDA ?NUMPROX-1,X ;GET # OF
        STA BOMBCT  ;BOMBS FOR LEVEL
        TAX         ;& SAVE IN X.
?PLACEBOMBS
        LDA #58     ;GET RANDOM #
        JSR RANDO   ;(0 UP TO ACC-1)
        CLC         ;THEN ADD THREE.
        ADC #$03
        ASL A       ;MULTIPLY BY 2.
        TAY         ;MOVE A TO Y.
        LDA SCREENBYTES,Y
        STA LO      ;GET SCREEN RAM
        STA BOMBL,X
        LDA SCREENBYTES+1,Y
        STA LO+1    ;Y-COORDINATE.
        STA BOMBH,X
        LDA #115    ;GET RANDOM #.
        JSR RANDO   ;(0 TO ACC-1)
        CLC         ;NOW ADD SIX.
        ADC #$06
        ORA #$01    ;BIT 0 ON.
        TAY         ;MOVE TO Y.
        LDA (LO),Y  ;GET LOCATION
        BEQ ?PLACEBOMBS ;TUNNEL!
        CMP #BOMB
        BEQ ?PLACEBOMBS ;IT'S A BOMB!
        LDA #BOMB   ;BOMB CHARACTER.
        STA (LO),Y  ;STORE IN RAM.
        CLC         ;STORE NEIGH-
        ADC #1      ;BORING CHARA-
        INY         ;CTER IN ADJA-
        STA (LO),Y  ;CENT LOCATION.
        DEY         ;ADD CORRECT
        TYA         ;VALUE INTO
        CLC         ;BOMB DATABASE
        ADC BOMBL,X ;& SAVE BOMB'S
        STA BOMBL,X ;X & Y COORDS.
        LDA #$FF    ;RESET BOMB
        STA ARMED,X ;WITH 255.
        DEX         ;HANDLE NEXT.
        BPL ?PLACEBOMBS
;
; PLACE BORDERS
;
        LDX #$00    ;CLEAR X.
?BOARD1
        LDA SCREENBYTES+1,X ;AT END
        CMP #$FF    ;OF SCR BYTES?
        BEQ ?BOARD2 ;YES! BRANCH!
        STA LO+1    ;SAVE RAM LOC.
        LDA SCREENBYTES,X
        STA LO      ;BOTH LO & HI.
        LDY #$01    ;BORDER CHAR.
        TYA         ;MOVE Y TO A.
        STA (LO),Y  ;STORE IT.
        LDY #127
        STA (LO),Y  ;STORE IT.
        INX         ;MOVE TO NEXT
        INX         ;SCREEN RAM.
        JMP ?BOARD1 ;JUMP ON IT.
?BOARD2
        DEX         ;SCREEN RAM
        DEX         ;END.
        LDA SCREENBYTES,X
        STA LO      ;GET LAST 1/2
        LDA SCREENBYTES+1,X
        STA LO+1    ;& FIRST 1/2
        LDA SCREENBYTES
        STA LOW     ;PAGE & SAVE
        LDA SCREENBYTES+1
        STA LOW+1   ;THEIR LOCS.
        LDY #127    ;LOAD POINTER.
        LDA #$01    ;BORDER CHAR.
?BOARD3
        STA (LO),Y  ;SAVE INTO RAM
        STA (LOW),Y ;AT TOP & BOTTOM
        DEY         ;OF BOARD.
        BPL ?BOARD3 ;HANDLE ALL...
        JSR SHOSCO  ;SHOW SCORE
        JSR SHOLVL  ;SHOW LEVEL
        JSR SHOLIV  ;SHOW LIVES
;
;RESET ALL ENEMIES
;
        LDX #29     ;CLEAR ALL 30
        LDA #0      ;ZERO OUT...
CLEN    STA EACT,X  ;ACTIVE FLAG
        DEX         ;NEXT ENEMY
        BPL CLEN    ;LOOP UNTIL DONE
;
;NOW INITIALIZE 4 ENEMIES
;
        LDX #3      ;4 OF 'EM
IELP    LDA #1      ;SET...
        STA EACT,X  ;ACTIVE FLAG
        LDA #$FF    ;GIVE IT...
        STA EDIR,X  ;RANDOM DIRECTION
        LDA ENSTH,X ;GET ADDRESS...
        STA EADH,X  ;OF EACH...
        STA ADL+1   ;ENEMY...
        LDA ENSTL,X ;AND SAVE!
        STA EADL,X
        STA ADL
        LDA #8      ;SET UP AS...
        STA ETYP,X  ;TYPE 8
        LDY #0      ;AND DRAW...
        STA (ADL),Y ;ON DISPLAY
        INY
        CLC
        ADC #1
        STA (LO),Y
        DEX         ;NEXT ENEMY
        BPL IELP    ;LOOP IF MORE
        LDA #0      ;START UP IN...
        STA GENFLG  ;MATING SEASON!
;
;INIT E8BS
;
        LDA LEVEL   ;GET LEVEL
        SEC         ;AND DECREMENT
        SBC #1      ;BY 1.
        LSR A       ;DIVIDE BY 8,
        LSR A
        LSR A
        STA E8BTOT  ;SAVE A MINUTE,
        LDA #3      ;GET 3
        SEC         ;SUBTRACT TEMP
        SBC E8BTOT  ;FROM IT.
        STA E8GROUP ;SAVE # OF NESTS
        TAX         ;USE AS INDEX...
        LDA IE8BCT,X ;GET BABY COUNT
        STA E8BTOT  ;AND SAVE...
        STA E8BS    ;IN WORK...
        STA E8BIX   ;AREAS
IE8BL1  LDX E8GROUP ;GET NEXT COUNT
        LDA E8BSTL,X ;GET ADDRESS...
        STA HLO     ;OF NEST CENTER
        LDA E8BSTH,X
        STA HLO+1
        LDX #12     ;13 BABIES
IE8BL2  LDA HLO     ;GET NEST ADDR...
        CLC         ;AND ADD...
        ADC EADDL,X ;DIRECTION...
        STA ADL     ;OFFSET
        LDY E8BIX
        STA E8BL,Y
        LDA HLO+1
        ADC EADDH,X
        STA ADL+1
```

```
        STA E88H,Y
        LDA #1          ;ACTIVATE...
        STA E8BACT,Y    ;BABY
        LDY #0
        LDA #12         ;AND DRAW...
        STA (ADL),Y     ;BABY...
        INY             ;ON SCREEN!
        CLC
        ADC #1
        STA (ADL),Y
        DEC E88IX       ;NEXT INDEX
        DEX             ;NEXT BABY
        BPL IE88L2      ;LOOP IF MORE
        DEC E8ROUP      ;NEXT NEST
        BPL IE88L1      ;LOOP IF MORE
;
;PLACE FIREBUG
;
        LDA # >SCREENBASE+131
        STA BUGL+1
        STA ADL+1
        LDA # <SCREENBASE+131
        STA BUGL        ;SAVE FIREBUG'S
        STA ADL         ;INITIAL ADDRESS
        LDA #6          ;GET BUG CHAR.
        LDY #0          ;AND DRAW...
        STA (ADL),Y     ;ON SCREEN
        INY
        CLC
        ADC #1
        STA (ADL),Y
        JSR SETCOLORS
        LDA #$00        ;START...
        STA VSTOP       ;VBLANK AGAIN!
        RTS             ;INIT DONE!
;
; COLOR SELECTION
;
SETCOLORS
        LDA RANDOM      ;RANDOM #.
        AND #$F0        ;CLEAR LO-NIBBLE
        ORA #$02        ;BIT 1 ON.
        STA NEWBACK     ;SAVE IT.
        LDX #$04        ;HANDLE 4 REGs.
?SETCOLORS
        LDA RANDOM      ;RANDOM #.
        AND #$F0        ;CLEAR LO-NIB.
        ORA ?ORBASE,X   ;TURN ON BITS.
        STA COLOR0,X    ;SAVE IT.
        DEX             ;HANDLE ALL...
        BPL ?SETCOLORS
        LDA COLOR0+1    ;GET COLOR FOR
        STA PCOLR0+2    ;BORDER MISS-
        STA PCOLR0+3    ;ILES.
        RTS
;
; MOMENTARY PAUSE
;
INIT3
        LDA RTCLOK+2    ;SYSTEM CLOCK.
        CLC             ;ONE SECOND
        ADC #70         ;PAUSE...
?INIT3
        CMP RTCLOK+2    ;1 SEC UP?
        BNE ?INIT3      ;NO! BRANCH!
        JMP INIT2       ;YES! WHOOSH!!!
;
; SOUND REGISTER INITIALIZATION
;
SOUNDOFF
        JSR SIOINV      ;INIT SOUNDS.
        LDX #$07        ;HANDLE ALL.
        LDA #$00        ;STUFF ZERO.
?SOUNDOFF
        STA AUDF1,X     ;INTO REGISTER.
        DEX             ;ALL DONE?
        BPL ?SOUNDOFF   ;NO! BRANCH!
        STA AUDCTL      ;STUFF CONTROL.
        RTS             ;BEAM ME UP...
;
?ORBASE
        .BYTE $08,$0A,$08,$04,$00
?TYPEBASE
        .BYTE 0,4,5,6,7
?NUMPROX
        .BYTE 7,19,29,39,49
        .BYTE 59,69,79,89,99
?VECTORS
        .BYTE 63,12,63,31,0,1
        .BYTE 64,12,64,31,0,1
        .BYTE 63,33,63,32,0,1
        .BYTE 64,33,64,32,0,1
        .BYTE 41,32,62,32,1,0
        .BYTE 65,32,86,32,1,0
DIRADL  .BYTE 128,2,128,254
DIRADH  .BYTE $FF,0,0,$FF
OK      .BYTE 1,0,1,1,1,1,1,0
        .BYTE 0,0,0,0,1,0,1,0
        .BYTE 0,0,0
CHANCE  .BYTE 224,0,64,64,64
        .BYTE 64,255,0,255,0
        .BYTE 255,0,255,0,255,0
        .BYTE 0,0,0
SDIR    .BYTE $FF,$FF,$FF,$FF
        .BYTE $FF,$FF,$FF,1
        .BYTE $FF,$FF,$FF,3
        .BYTE $FF,2,0,$FF
ENSTH   .BYTE  >SCREENBASE+3775
        .BYTE  >SCREENBASE+4153
        .BYTE  >SCREENBASE+4165
        .BYTE  >SCREENBASE+4543
ENSTL   .BYTE  <SCREENBASE+3775
        .BYTE  <SCREENBASE+4153
        .BYTE  <SCREENBASE+4165
        .BYTE  <SCREENBASE+4543
EADDL   .BYTE 0,128,128,254,2,130
        .BYTE 126,126,130,4,252,0,0
EADDH   .BYTE 0,0,255,255,0,255
        .BYTE 0,255,0,0,255,255,1
E88STL  .BYTE  <SCREENBASE+2079
        .BYTE  <SCREENBASE+6239
        .BYTE  <SCREENBASE+6175
        .BYTE  <SCREENBASE+2143
E88STH  .BYTE  >SCREENBASE+2079
        .BYTE  >SCREENBASE+6239
        .BYTE  >SCREENBASE+6175
        .BYTE  >SCREENBASE+2143
IE88LT  .BYTE 12,25,38,51
CSTRT   .BYTE 0,8,16,24,32,40,112
        .BYTE 120,128,180,168,176
        .BYTE 184,192
CDATA   .BYTE 0,0,0,0,0,0,0,0
        .BYTE 170,170,255,255
        .BYTE 255,255,170,170
        .BYTE 16,1,0,4,64,0,1,64
        .BYTE 0,16,1,64,4,0,17,0
        .BYTE 64,1,0,4,64,1,0,16
        .BYTE 16,1,0,64,1,16,0,4
        .BYTE 0,0,0,0,0,0,0,0
        .BYTE 1,4,1,2,11,47,47,10
        .BYTE 64,16,5,128
        .BYTE 224,248,248,160
        .BYTE 255,102,102,102
        .BYTE 102,102,102,255
        .BYTE 255,100,100,100
        .BYTE 100,100,100,255
        .BYTE 255,96,96,96
        .BYTE 96,96,96,255
        .BYTE 255,64,64,64
        .BYTE 64,64,64,255
        .BYTE 255,0,0,0
        .BYTE 0,0,0,255
;
; ************************
; * RANDOM NUMBER GENERATOR *
; ************************
;
RANDO
        BEQ ?RET        ;IF = 0 BRANCH.
        STA ?HOLD       ;SAVE ACC.
        LDA RANDOM      ;GET RANDOM #.
?RAN1
        CMP ?HOLD       ;IS IT < ACC?
        BCC ?RET        ;YES! BRANCH!
        LSR A           ;NO. DIVIDE BY
        LSR A           ;TWO. TRY AGAIN.
        JMP ?RAN1       ;JUMP ON IT!
?RET
        RTS             ;BOOGIE...
?HOLD
        .BYTE 0         ;TEMP STORAGE.
;
; ************************
; * X & Y COORDINATE TRACER *
; ************************
;
XYFIND
        STX ?HOLD       ;SAVE X-REG.
        LDA YCOORD      ;GET Y-COORD.
        ASL A           ;MULTIPLY BY 2.
        TAX             ;MOVE A TO X.
        LDA SCREENBYTES,X ;GET RAM
        CLC             ;BYTE & ADD
        ADC XCOORD      ;X-COORD.
        STA LO          ;SAVE IT.
        LDA SCREENBYTES+1,X
        STA LO+1        ;GET HI BYTE.
        LDX ?HOLD       ;RESTORE X-REG.
        RTS             ;BAMF!!!
?INITX  .BYTE 0
?INITY  .BYTE 0
?FINALX .BYTE 0
?FINALY .BYTE 0
?DELTAX .BYTE 0
?DELTAY .BYTE 0
; ************************
; * FINE SCROLL CONTROL *
; *    BY Kyle Peacock    *
; *    [WHO ELSE?!?]      *
; ************************
;
        .LOCAL
;
SCROLL
        LDA BUGL+1      ;BUG AT SCREEN
        CMP DLISTSTART+2 ;TOP?
        BCS ?VTEST1     ;NO! BRANCH!
        LDA #$00        ;YES! CLEAR &
        BEQ ?VSET       ;BRANCH!
?VTEST1
        CMP DLISTEND+2  ;BUG AT BOTTOM?
        BCC ?VSCROLL    ;NO! BRANCH!
        LDA #$01        ;YES! SET.
?VSET
        STA ?VDIR       ;SAVE VERT.FLAG
        LDX #4          ;SET VERT SCROLL
        STX ?VCOUNT     ;DISTANCE.
?VSCROLL
        LDA ?VCOUNT     ;SCROLL NOW?
        BEQ ?VSTORE     ;IF NOT, BRANCH.
        LDA ?VDIR       ;GET DIRECTION
        BNE ?DOWN       ;TO SCROLL IN.
?UP
        DEC VBIT        ;DEC RAM VSCROL.
        BPL ?VSTORE     ;IF >0 BRANCH.
?UP1
        LDA YPOINT      ;GET Y-COORD.
        CMP MINY        ;AT BOARD TOP?
        BNE ?UP2        ;NO! BRANCH!
        INC VBIT        ;YES! HALT
        JMP ?VHALT      ;SCROLL. BRANCH.
?UP2
        LDA #$07        ;RESET RAM COPY
        STA VBIT        ;OF VSCROL.
        DEC YPOINT      ;DEC Y-COORD.
        JMP ?VDEC       ;JUMP ON IT!
?DOWN
        INC VBIT        ;INC RAM VSCROL
        LDA VBIT        ;IF <8 BRANCH.
        CMP #$08
        BCC ?VSTORE
        LDA YPOINT      ;GET Y-COORD.
        CMP MAXY        ;AT BOARD BOTTOM?
        BNE ?DOWN2      ;NO! BRANCH!
        DEC VBIT        ;YES! HALT
        JMP ?VHALT      ;SCROLL. BRANCH.
?DOWN2
        LDA #$00        ;RESET RAM COPY
        STA VBIT        ;OF VSCROL.
        INC YPOINT      ;INC Y-COORD.
?VDEC
        DEC ?VCOUNT     ;DEC VERT
        JMP ?VSTORE     ;SCROLL COUNT.
?VHALT
        LDA #$00        ;HALT VERT
        STA ?VCOUNT     ;SCROLL COUNT.
?VSTORE
        LDA VBIT        ;GET RAM COPY &
        STA VSCROL      ;SAVE IN OS.
        LDA DLISTSTART+1 ;TEST TO
        AND #$7F        ;SEE IF BUG IS
        STA ?HOLD       ;TOO FAR LEFT
        LDA BUGL        ;OR TOO FAR
        AND #$7F        ;RIGHT OF SCREEN
        SEC             ;CENTER.
        SBC ?HOLD
        CMP #30         ;VALUE < 30?
        BCC ?HTEST2     ;YES! BRANCH!
        LDA #$00        ;TOO FAR RIGHT.
        BEQ ?HSET       ;BRANCH!
?HTEST2
        CMP #15         ;VALUE > 14?
        BCS ?HSCROLL    ;YES! BRANCH!
        LDA #$01        ;TOO FAR LEFT.
?HSET
        STA ?HDIR       ;SAVE DIRECTION.
        LDA #12         ;SET HORT SCROLL
        STA ?HCOUNT     ;DISTANCE.
?HSCROLL
        LDA ?HCOUNT     ;HORT SCROLL?
        BEQ ?HSTORE     ;NO! BRANCH!
        LDA ?HDIR       ;YES! GET SCROLL
        BEQ ?RIGHT      ;DIRECTION.
?LEFT
        INC HBIT        ;INC RAM HSCROL.
        LDA HBIT        ;IS IT > 3?
        CMP #$04
        BCC ?HSTORE     ;NO! BRANCH!
        LDA XPOINT      ;AT LEFT EDGE?
        CMP MINX
        BNE ?LEFT2      ;NO! BRANCH!
        DEC HBIT        ;RESET HSCROL
        JMP ?HHALT      ;& HALT SCROLL.
?LEFT2
        LDA #$00        ;RESET HSCROL
        STA HBIT        ;RAM COPY.
        DEC XPOINT      ;DEC X-COORD.
        JMP ?HDEC       ;JUMP ON IT!
?RIGHT
        DEC HBIT        ;DEC RAM HSCROL
        BPL ?HSTORE     ;IF >0 BRANCH!
        LDA XPOINT      ;GET X-COORD.
        CMP MAXX        ;AT RIGHT EDGE?
        BNE ?RIGHT2     ;NO! BRANCH!
        INC HBIT        ;RESET HSCROL
        JMP ?HHALT      ;& HALT SCROLL.
?RIGHT2
        LDA #$03        ;RESET RAM
        STA HBIT        ;HSCROL &
        INC XPOINT      ;INC X-COORD.
?HDEC
        DEC ?HCOUNT     ;DEC HORT SCROLL
        JMP ?HSTORE     ;COUNT.
?HHALT
        LDA #$00        ;HALT HORT
        STA ?HCOUNT     ;SCROLL COUNT.
?HSTORE
        LDA HBIT        ;SAVE RAM HSCROL
        STA HSCROL      ;INTO OS.
        LDX PLAYNO      ;GET JOYSTICK
        LDY STICK0,X    ;POSITION.
        CPY #15         ;CENTERED?
        BEQ ?SCROLLDONE ;YES-BRANCH.
        CPY ?PREVIOUS   ;UNTOUCHED?
        BEQ ?SCROLLDONE ;YES-BRANCH.
        LDA ?TYPESET-5,Y ;UPDATE
        STA TYPE        ;BUG FACING.
?SCROLLDONE
        STY ?PREVIOUS   ;SAVE STICK.
        RTS             ;CHOW BABY...
?PREVIOUS
        .BYTE 0         ;PREVIOUS STICK
?TYPESET ;              BUG HEADINGS
        .BYTE 3,1,1,0,2,0,2,0,3,0
?HOLD
        .BYTE 0         ;TEMP STORAGE
?VDIR
        .BYTE 0         ;VERT DIRECTION
?HDIR
        .BYTE 0         ;HORT DIRECTION
?VCOUNT
        .BYTE 0         ;VERTICAL COUNT
?HCOUNT
        .BYTE 0         ;HORIZONTAL COUNT
; ************************
; * FIREBUG DRAW ROUTINE *
; *    BY  Kyle Peacock   *
; ************************
;
        .LOCAL
;
DRAW
        LDA MAXBUGS     ;# OF CHARS TO
        STA ?COUNT      ;MODIFY.
?DRAW1
        LDX ?COUNT      ;GET COUNTER.
        LDY TYPE,X      ;CHARACTER TYPE.
        LDA PHASE,X     ;TYPE PHASE.
        CMP ?INDEX+1,Y  ;PHASE EXCEEDED?
        BCC ?DRAW3      ;NO! BRANCH!
?DRAW2
        LDA ?INDEX,Y    ;RESET WITH
        STA PHASE,X     ;PROPER PHASE.
?DRAW3
        CMP ?INDEX,Y    ;CORRECT PHASE?
        BCC ?DRAW2      ;NO! BRANCH!
        ASL A           ;PHASE TIMES 2.
        TAY             ;MOVE A TO Y.
        LDA ?OFFSET,Y   ;ADDR OF CHAR
        STA DRWLO       ;DATA (LO BYTE)
        LDA ?OFFSET+1,Y ;HI BYTE OF
        STA DRWLO+1     ;CHAR DATA.
        LDA ?COUNT      ;GET COUNTER.
        ASL A           ;MULTIPLY BY 2.
        TAY             ;MOVE A TO Y.
        LDA ?PLAYBASE,Y ;DESTINATION
        STA PLLO        ;LO BYTE
        LDA ?PLAYBASE+1,Y ;&
        STA PLLO+1      ;HI BYTE.
        LDY #15         ;MOVE 16 BYTES.
?DRAW4
        LDA (DRWLO),Y   ;GET DATA.
        STA (PLLO),Y    ;SAVE DATA.
        DEY             ;NEXT BYTE.
        BPL ?DRAW4      ;ALL DONE?
        DEC ?COUNT      ;NEXT CHARACTER.
        BPL ?DRAW1      ;ALL DONE?
        DEC ?PHTIME     ;DEC TIMER.
        BPL ?DRAW7      ;IF >0 BRANCH!
        LDA MOVETIME    ;RESET PHASE
        STA ?PHTIME     ;TIMER.
        LDX PLAYNO      ;PLAYER NUMBER.
        LDA STICK0,X    ;GET JOYSTICK.
        CMP #15         ;CENTERED?
        BEQ ?DRAW7      ;YES! BRANCH!
        INC PHASE       ;INC FIREBUG PHASE.
?DRAW7
        DEC ?PHTIME+1   ;DEC TIMER.
        BPL ?DRAW9      ;IF >0 BRANCH.
        LDA PHASETIME   ;RESET TIMER.
        STA ?PHTIME+1   ;SAVE IT.
        LDX MAXBUGS     ;# OF CHARS
?DRAW8
        INC PHASE,X     ;UPDATE PHASES.
        DEX             ;HANDLE NEXT.
        BNE ?DRAW8      ;ALL DONE?
?DRAW9
        RTS             ;ZOOOOOOM!!!
?COUNT
?PHTIME
        .BYTE 0         ;COUNTER
        .BYTE 3,3       ;TIMERS
;
?INDEX  ;               CHARACTER INDEX
        .BYTE 0,4,8,12,16
        .BYTE 21,31,37,40
;
?PLAYBASE
```

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

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

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

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.

CIRCLE #121 ON READER SERVICE CARD

```
          .WORD [GAMESET+48]
          .WORD [GAMESET+64]
          .WORD [GAMESET+80]
          .WORD [GAMESET+96]
          .WORD [GAMESET+144]
;
?OFFSET
          .WORD FIREBUGUP1
          .WORD FIREBUGUP2
          .WORD FIREBUGUP3
          .WORD FIREBUGUP2
          .WORD FIREBUGRIGHT1
          .WORD FIREBUGRIGHT2
          .WORD FIREBUGRIGHT3
          .WORD FIREBUGRIGHT2
          .WORD FIREBUGLEFT1
          .WORD FIREBUGLEFT2
          .WORD FIREBUGLEFT3
          .WORD FIREBUGLEFT2
          .WORD FIREBUGDOWN1
          .WORD FIREBUGDOWN2
          .WORD FIREBUGDOWN3
          .WORD FIREBUGDOWN2
          .WORD ?BUG11
          .WORD ?BUG12
          .WORD ?BUG13
          .WORD ?BUG14
          .WORD ?BUG15
          .WORD ?BUG21
          .WORD ?BUG22
          .WORD ?BUG23
          .WORD ?BUG24
          .WORD ?BUG25
          .WORD ?BUG26
          .WORD ?BUG25
          .WORD ?BUG24
          .WORD ?BUG23
          .WORD ?BUG22
          .WORD ?E001
          .WORD ?E002
          .WORD ?E003
          .WORD ?E004
          .WORD ?E003
          .WORD ?E002
          .WORD BULLET1
          .WORD BULLET2
          .WORD BULLET3
FIREBUGUP1
          .BYTE 12,2,9,194,53,202,53,2
          .BYTE 48,128,96,131,92,163,92,128
FIREBUGUP2
          .BYTE 3,2,9,2,245,10,245,2
          .BYTE 192,128,96,128,95,160,95,128
FIREBUGUP3
          .BYTE 12,2,9,2,53,202,53,194
          .BYTE 48,128,96,128,92,163,92,131
FIREBUGRIGHT1
          .BYTE 12,51,25,153,153,25,51,12
          .BYTE 192,0,35,152,152,35,0,192
FIREBUGRIGHT2
          .BYTE 51,51,25,153,153,25,51,51
          .BYTE 0,0,32,155,155,32,0,0
FIREBUGRIGHT3
          .BYTE 204,51,25,153,153,25,51,204
          .BYTE 0,0,35,152,152,35,0,0
FIREBUGLEFT1
          .BYTE 3,0,200,38,38,200,0,3
          .BYTE 48,204,100,102,102,100,204,48
FIREBUGLEFT2
          .BYTE 0,0,8,230,230,8,0,0
          .BYTE 204,204,100,102,102,100,204,204
FIREBUGLEFT3
          .BYTE 0,0,200,38,38,200,0,0
          .BYTE 51,204,100,102,102,100,204,51
FIREBUGDOWN1
          .BYTE 2,53,202,53,194,9,2,12
          .BYTE 128,92,163,92,131,96,128,48
FIREBUGDOWN2
          .BYTE 2,245,10,245,2,9,2,3
          .BYTE 128,95,160,95,128,96,128,192
FIREBUGDOWN3
          .BYTE 194,53,202,53,2,9,2,12
          .BYTE 131,92,163,92,128,96,128,48
?BUG11
          .BYTE 48,12,2,9,9,50,192,0
          .BYTE 0,3,140,96,96,128,48,12
?BUG12
          .BYTE 0,240,14,9,9,14,48,48
          .BYTE 12,12,176,96,96,176,15,0
?BUG13
          .BYTE 0,0,242,9,9,2,12,12
          .BYTE 48,48,128,96,96,143,0,0
?BUG14
          .BYTE 3,0,2,57,201,2,3,0
          .BYTE 0,192,128,99,100,128,0,192
?BUG15
          .BYTE 12,3,2,9,57,194,0,0
          .BYTE 0,0,131,100,96,128,192,48
?BUG21
          .BYTE 0,0,0,1,1,8,160,32
          .BYTE 8,10,32,64,64,0,0,0
?BUG22
          .BYTE 0,0,0,1,1,40,8,0
          .BYTE 0,32,40,64,64,0,0,0
?BUG23
          .BYTE 0,0,0,1,9,2,0,0
          .BYTE 0,0,128,96,64,0,0,0
?BUG24
          .BYTE 0,0,2,9,1,0,0,0
          .BYTE 0,0,0,64,96,128,0,0
?BUG25
          .BYTE 0,8,40,1,1,0,0,0
          .BYTE 0,0,0,64,64,40,32,0
?BUG26
          .BYTE 32,160,8,1,1,0,0,0
          .BYTE 0,0,0,64,64,32,10,8
?E001
          .BYTE 0,0,0,171,171,0,0,0
          .BYTE 0,0,0,234,234,0,0,0
?E002
          .BYTE 0,0,14,40,40,14,0,0
          .BYTE 0,0,176,40,40,176,0,0
?E003
          .BYTE 0,50,2,8,8,2,50,0
          .BYTE 0,140,128,32,32,128,140,0
?E004
          .BYTE 194,2,2,0,0,2,2,194
          .BYTE 131,128,128,0,0,128,128,131
BULLET1
          .BYTE 0,0,0,2,2,0,0,0
          .BYTE 0,0,0,128,128,0,0,0
BULLET2
          .BYTE 0,0,1,4,4,1,0,0
          .BYTE 0,0,64,16,16,64,0,0
BULLET3
          .BYTE 0,3,0,48,48,0,3,0
          .BYTE 0,192,0,12,12,0,192,0
;
; ***************************
; * FIREBUG CHARACTER USAGE *
; ***************************
;
;   CHAR #  REDEFINED CHARACTER
;
;       0       BLANK
;       1       BORDER
;       2       DIRT
;       3       DIRT
;       4       DIRT
;       5       DIRT
;     6 & 7     PLAYER'S FIREBUG
;     8 & 9     ENEMY BUG #1
;    10 & 11    ENEMY BUG #2
;    12 & 13    BABY FIREBUG
;      14       FIREBUG TRAIL
;    15 & 16    PROXIMITY BOMB
;      17       EXPLOSION
;    18 & 19    FIREBUG SPARK
;20,21,22,23,24 SPARK BARLINE
; ***************************
; * FIREBUG SPARK HANDLER *
; *    BY Kyle Peacock    *
; ***************************
;
          .LOCAL
;
SPARK
          JSR STARTSPARK ;FIRING.
          JMP MOVESPARK ;MOVING.
STARTSPARK
          DEC FIRETIME ;DEC TIMER.
          BNE ?RETURN ;IF <>0 BRANCH!
          LDA #$03    ;RESET TIMER.
          STA FIRETIME ;SAVE IT.
          LDX PLAYNO  ;GET PLAYER #.
          LDA TRIGG,X ;GET TRIGGER.
          BNE ?RETURN ;IF <>0 BRANCH!
          LDA NOBULL,X ;FIRED BULLETS.
          CMP #$05    ;IF >=5
          BCS ?RETURN ;BRANCH!
          LDA BULLETLINE+10 ;CHECK BAR
          CMP #24     ;LINE.
          BEQ ?RETURN ;IF CHR=24 BRANCH!
          LDX #$03    ;SET UP X.
?SEARCH
          LDA BULLDIR,X ;THIS BULLET
          CMP #$FF    ;ACTIVE?
          BEQ ?FOUND  ;NO! BRANCH!
          DEX         ;YES, CHECK NEXT
          BPL ?SEARCH ;WORK AREA.
          JMP ?RETURN ;NONE AVAILABLE.
?FOUND
          INC NOBULL  ;INC BULLETS FIRED.
          LDA BUGL    ;FIREBUG'S LO
          STA BULLX,X ;ADDRESS.
          LDA BUGL+1  ;FIREBUG'S HI
          STA BULLY,X ;ADDRESS.
          LDA TYPE    ;GET FIREBUG'S
          STA BULLDIR,X ;DIRECTION.
          LDA #20     ;SET SPARK LIFE.
          STA BULLTIME,X ;SAVE IT.
          LDA #6      ;SET UP FIRING
          STA BULLNOISE ;SOUND EFFECT.
          LDX BULLPNT ;DEC BAR LINE OF
          INC BULLETLINE,X ;REMAINING
          LDA BULLETLINE,X ;SPARKS.
          CMP #24     ;[DON'T ALLOW
          BNE ?RETURN ;SPARK EJECTION
          LDA BULLPNT ;IF NO SPARKS
          CMP #10     ;ARE ON BAR
          BEQ ?RETURN ;LINE.]
          DEC BULLPNT ;DEC POINTER.
?RETURN
          RTS         ;GET OUT!!!
MOVESPARK
          LDX #$03    ;HANDLE ALL.
?SPARKY
          LDA BULLDIR,X ;SPARK DIRECT.
          CMP #$FF    ;IS IT $FF?
          BNE ?SPARKS ;NO! BRANCH!
          JMP ?NEXTBULL ;HANDLE NEXT.
?SPARKS
          LDA BULLX,X ;GET BULLET LO.
          STA ZLO     ;SAVE IT.
          LDA BULLY,X ;GET BULLET HI.
          STA ZLO+1   ;SAVE IT.
          LDY #$00    ;CLEAR Y-REG.
          LDA (ZLO),Y ;GET CHAR.
          CMP #18     ;BULLET?
          BEQ ?BLANK  ;YES. BRANCH!
          CMP #17     ;EXPLOSION?
          BNE ?NOBLANK ;YES! BRANCH!
?BLANK
          LDA #14     ;FIREBUG TRAIL.
          STA (ZLO),Y ;SAVE IT.
          INY         ;INC POINTER.
          STA (ZLO),Y ;SAVE IT.
?NOBLANK
          LDA BULLDIR,X ;SPARK DIRECT.
          BPL ?REDRAW ;IF >0 BRANCH!
          LDA #$FF    ;TURN OFF SPARK.
          STA BULLDIR,X ;SAVE IT.
          DEC NOBULL  ;DEC FIRED SPARKS.
          JMP ?NEXTBULL ;HANDLE NEXT.
?REDRAW
          TAY         ;MOVE A TO Y.
          LDA BULLX,X ;GET SPARK LO.
          CLC         ;CLEAR CARRY.
          ADC ?LOADD,Y ;DIRECTION ADD.
          STA ZLO     ;SAVE IT.
          LDA BULLY,X ;GET SPARK HI.
          ADC ?HIADD,Y ;DIRECTION ADD.
          STA ZLO+1   ;SAVE IT.
          LDY #$00    ;CLEAR Y-REG.
          LDA (ZLO),Y ;GET CHARACTER.
          AND #$7F    ;CLEAR BIT 7.
          BEQ ?80     ;IF =0 BRANCH!
          CMP #14     ;IS IT =14?
          BEQ ?80     ;YES! BRANCH!
?STOP
          PHA         ;SAVE CHARACTER.
          LDA #$80    ;BIT 7 ON.
          STA BULLDIR,X ;SAVE IT.
          PLA         ;RESTORE CHAR.
          CMP #1      ;BORDER?
          BEQ ?NEXTBULL ;YES! BRANCH!
          CMP #15     ;BOMB?
          BEQ ?NEXTBULL ;YES! BRANCH!
          CMP #12     ;BABY FIREBUG?
          BEQ ?NEXTBULL ;YES! BRANCH!
          JSR ?POSITION ;SAVE NEW X&Y.
          LDA #17     ;EXPLOSION.
          STA (ZLO),Y ;SAVE IT.
          INY         ;INC POINTER.
          STA (ZLO),Y ;SAVE IT.
          LDA #21     ;TURN ON DETO-
          STA BOOMNOISE ;NATION SOUND.
          BNE ?NEXTBULL ;BRANCH!
?80
          LDA #18     ;BULLET.
          STA (ZLO),Y ;SAVE IT.
          INY         ;INC POINTER.
          CLC         ;INC CHAR.
          ADC #$01
          STA (ZLO),Y ;SAVE IT.
          JSR ?POSITION ;SAVE NEW X&Y.
          DEC BULLTIME,X ;DEC LIFETIME.
          BPL ?NEXTBULL ;IF >0 BRANCH!
          LDA #$01    ;FORCED HALT.
          BNE ?STOP   ;BRANCH!
```

```
?NEXTBULL
          DEX             ;DEC POINTER.
          BMI ?RETURN2    ;IF <0 BRANCH!
          JMP ?SPARKY     ;JUMP ON IT.
?RETURN2
          RTS             ;LATER Y'ALL...
?POSITION
          LDA ZLO         ;GET NEW LO.
          STA BULLX,X     ;SAVE IT.
          LDA ZLO+1       ;GET NEW HI.
          STA BULLY,X     ;SAVE IT.
          RTS             ;PLEASE LEAVE!!!
;
; DIRECTIONAL ADD ONs
;
?LOADD
          .BYTE 128,2,254,128
?HIADD
          .BYTE 255,0,255,0
;
; **********************
; * EXPLOSION ROUTINE  *
; *    BY Tom Hudson   *
; **********************
;
EXPLOD    LDA #0          ;RESET...
          STA EXPPH       ;EXP. PHASE
          LDA #66         ;AND START...
          STA EXNOISE     ;EXP. SOUND
EXPDLP    LDA #17         ;SET UP...
          STA EXPCHR      ;EXPLOSION CHAR.
          LDA EXPPH       ;GET PHASE
          CMP #13         ;>12?
          BCC PH1         ;NO, PHASE 1
          SEC             ;ADJUST INDEX...
          SBC #13         ;FOR PHASE 2
          LDY #0          ;AND SET UP...
          STY EXPCHR      ;ERASE CHARACTER
PH1       TAX             ;PUT PHASE IN X
          LDA ELO         ;AND ADD...
          CLC             ;EXPLOSION...
          ADC EADDL,X     ;ADDRESS...
          STA ELO2        ;OFFSET...
          LDA ELO+1       ;FOR THE BYTE...
          ADC EADDH,X     ;TO BE...
          STA ELO2+1      ;CHANGED!
          LDY #0          ;NOW PUT...
          LDA EXPCHR      ;EXPLOSION CHAR.
          STA (ELO2),Y    ;ONTO DISPLAY
          INY
          STA (ELO2),Y
          JSR WAIT        ;WAIT A LITTLE
          INC EXPPH       ;NEXT PHASE
          LDA EXPPH       ;GET PHASE
          CMP #24         ;ALL DONE?
          BNE EXPDLP      ;NO, LOOP BACK!
          RTS             ;BYE!
;
;TTIME DELAY ROUTINE
;
WAIT      LDX #$03        ;THIS ROUTINE
XLOOP     LDY #$FF        ;WASTES A LITTLE
YLOOP     DEY             ;TIME SO THE
          BNE YLOOP       ;EXPLOSIONS
          DEX             ;CAN BE SEEN.
          BNE XLOOP
          RTS
;
; ****************************
; * PROXIMITY BOMB HANDLERS  *
; *      BY Tom Hudson       *
; ****************************
;
ARMBOM    LDX BOMBCT      ;GET # OF BOMBS
ARMLP     LDA ARMED,X     ;GET ARMED FLAG
          CMP #$FF        ;ARMED YET?
          BNE NXTBOM      ;YES!
          LDA BOMBL,X     ;GET BOMB...
          STA BLO         ;ADDRESS...
          LDA BOMBH,X     ;AND SAVE IT
          STA BLO+1
          LDY #0          ;GET CHAR...
          LDA (BLO),Y     ;FROM DISPLAY
          CMP #15         ;IS IT BOMB?
          BEQ DONARM      ;YUP, IT'S OK.
          CMP #12         ;IS IT BABY?
          BNE SETARM      ;NO, ARM IT!
          LDA #$80        ;THIS CORRECTS A
          STA ARMED,X     ;SMALL PROBLEM,
          BNE NXTBOM      ;DISABLING BOMB
SETARM    LDA #15         ;RESTORE...
          STA (BLO),Y     ;BOMB GRAPHIC...
          LDA #16         ;ON DISPLAY
          INY
          STA (BLO),Y
          BNE ARMIT       ;SET ARM TIMER

DONARM    LDY #3          ;SCAN 4 DIRS.
ARMSCAN   LDA BLO         ;ADD...
          CLC             ;DIRECTION...
          ADC DIRADL,Y    ;OFFSET...
          STA BLO2        ;TO BOMB...
          LDA BLO+1       ;ADDRESS
          ADC DIRADH,Y
          STA BLO2+1
          STY YTEMP       ;SAVE Y REG.
          LDY #0          ;GET DESIRED...
          LDA (BLO2),Y    ;DIRECTION BYTE
          CMP #18         ;SPARK?
          BEQ ARMIT       ;YES, ARM IT!
          CMP #6          ;FIREBUG?
          BNE NOARM       ;NO!
ARMIT     LDA #60         ;1 SECOND DELAY
          STA ARMED,X     ;ARM IT!
NXTBOM    DEX             ;NEXT BOMB
          BPL ARMLP       ;LOOP IF MORE
          BMI DECBT       ;NOW HANDLE TIME
NOARM     LDY YTEMP       ;RESTORE Y REG
          DEY             ;NEXT DIRECTION
          BMI NXTBOM      ;NO MORE DIRS!
          JMP ARMSCAN     ;LOOP IF MORE
;
;HANDLE BOMB TIMERS, DETONATE
;
DECBT     LDX BOMBCT      ;GET # OF BOMBS
DBLP      LDA ARMED,X     ;IS IT ARMED?
          BMI NOTARM      ;NOT ARMED!
          LDA BOMBL,X     ;LOAD POINTER...
          STA ELO         ;WITH BOMB'S...
          LDA BOMBH,X     ;ADDRESS
          STA ELO+1
          LDY #0          ;GET THE BYTE...
          LDA (ELO),Y     ;FROM DISPLAY
          BNE BLOW1       ;IT'S OK.
          LDA #15         ;RESTORE IMAGE
          BNE BLOW2
BLOW1
          LDA #$80        ;TOGGLE HI BIT
          EOR (ELO),Y     ;TO FLASH BOMB
BLOW2
          STA (ELO),Y     ;RE-DRAW BOMB
          INY
          CLC
          ADC #1
          STA (ELO),Y
          LDA #$00        ;SET FUSE...
          STA AUDF4       ;FREQUENCY
          LDA #$88        ;TURN ON...
          STA AUDC4       ;FUSE SOUND!
          DEC ARMED,X     ;DEC TIMER
          BNE NOTARM      ;NO EXPLOSION YET
          LDA #$00        ;TURN OFF...
          STA AUDC4       ;FUSE SOUND
          LDA #$80        ;MARK THE BOMB...
          STA ARMED,X     ;INACTIVE
          STX XTEMP       ;SAVE X REG.
          STY YTEMP       ;SAVE Y REG.
          JSR EXPLOD      ;BOOM!!!
          LDX XTEMP       ;RESTORE X
          LDY YTEMP       ;RESTORE Y
NOTARM    DEX             ;NEXT BOMB
          BPL DBLP        ;LOOP IF MORE
          RTS             ;BYE!
;
; **********************
; * BABY FIREBUG MOVER *
; *    BY Tom Hudson   *
; **********************
;
BMOVE     LDX E88TOT      ;GET BABY COUNT
E88LP     STX E88IX       ;SAVE INDEX
          LDA E88ACT,X    ;IS IT ACTIVE?
          BNE EACTIVE     ;YES!
          JMP NXTE88      ;NO, NEXT BABY.
EACTIVE   LDA E88L,X      ;GET...
          STA HLO         ;BABY ADDRESS...
          LDA E88H,X      ;AND SAVE IT
          STA HLO+1
          LDY #0          ;GET BABY'S...
          LDA (HLO),Y     ;SCREEN BYTE
          CMP #12         ;IS IT OK?
          BEQ E88OK       ;YES!
          DEC E88S        ;NO, KILL ONE
          BMI ALLDEAD     ;ALL DEAD!
          LDA #0          ;MARK BABY...
          STA E88ACT,X    ;INACTIVE
          LDA #9          ;START EATING...
          STA EATNOISE    ;SOUND
          JMP NXTE88      ;DO NEXT ONE
ALLDEAD   PLA             ;CLEAR...
          PLA             ;STACK.
          JMP GAMOVR      ;GAME OVER!!!
E88OK     LDA #3          ;TRY 4 DIRS.
          STA ETRY        ;SAVE INDEX
E88DLP    LDA RANDOM      ;GET RANDOM...
          AND #3          ;DIRECTION 0-3
          TAY             ;PUT IN Y REG.
          LDA E88L,X      ;GET...
          CLC             ;BABY ADDRESS...
          ADC DIRADL,Y    ;AND ADD...
          STA ADL         ;DIRECTION...
          LDA E88H,X      ;OFFSET
          ADC DIRADH,Y
          STA ADL+1
          LDY #0          ;GET CHARACTER
          LDA (ADL),Y     ;FROM SCREEN
          BEQ MOVE88      ;IT'S TUNNEL!
          CMP #14         ;BUG TUNNEL?
          BEQ MOVE88      ;YES!
NEDIR     DEC ETRY        ;NEXT DIRECTION
          BPL E88DLP      ;LOOP IF MORE
NXTE88    DEX             ;NEXT BABY
          BPL E88LP       ;LOOP IF MORE
          RTS             ;ALL DONE!
MOVE88    LDA #14         ;ERASE BABY...
          STA (HLO),Y     ;USING CHAR 14
          INY             ;(BUG TRAIL)
          STA (HLO),Y
          DEY
          LDA #12         ;DRAW BABY...
          STA (ADL),Y     ;IN NEW...
          INY             ;LOCATION
          CLC
          ADC #1
          STA (ADL),Y
          LDA ADL         ;UPDATE BABY'S
          STA E88L,X      ;ADDRESS
          LDA ADL+1
          STA E88H,X
          JMP NXTE88      ;AND LOOP!
ADD8CO    LDY #0          ;ZERO Y REG.
          SED             ;DECIMAL MODE
          CLC             ;CLEAR CARRY
          LDX #2          ;DO 3 DIGITS
A8CLP     LDA SCORE,X     ;GET DIGIT
          ADC SCOADD,X    ;ADD INCREMENT
          STA SCORE,X     ;SAVE DIGIT
          STY SCOADD,X    ;CLEAR INCREMENT
          DEX             ;NEXT DIGIT
          BPL A8CLP       ;LOOP IF MORE
          CLD             ;NO MORE DECIMAL
;
;SHOW SCORE
;
SHOSCO    LDA #$10        ;COLOR 0
          STA SHCOLR      ;SAVE IT
          LDX #0          ;DISPLAY INDEX
          LDY #0          ;DIGIT #
S8COLP    LDA SCORE,Y     ;GET DIGIT
          JSR SHOBCD      ;SHOW ON SCREEN
          INX             ;NEXT SCREEN POS
          INX             ;(INC TWICE)
          INY             ;NEXT DIGIT
          CPY #3          ;DONE ALL 3?
          BNE S8COLP      ;NO!
          RTS             ;ALL DONE!
;
;SHOW 2 BCD DIGITS
;
SHOBCD    STA SHOBYT      ;SAVE BCD BYTE
          AND #$0F        ;GET LOW DIGIT
          ORA SHCOLR      ;ADD COLOR
          STA SCOLIN+1,X  ;SHOW IT!
          LDA SHOBYT      ;GET BCD BYTE
          LSR A           ;GET HI DIGIT
          LSR A
          LSR A
          LSR A
          ORA SHCOLR      ;ADD COLOR
          STA SCOLIN,X    ;SHOW IT!
          RTS             ;ALL DONE!
;
;SHOW CURRENT LEVEL
;
SHOLVL    LDY #$50        ;COLOR 1
          STY SHCOLR      ;SAVE COLOR
          LDA BCDLVL      ;GET LEVEL #
          LDX #11         ;PUT IN 11TH BYTE
          JMP SHOBCD      ;SHOW IT!
;
;SHOW # OF LIVES
;
SHOLIV    LDA LIVES       ;GET # OF LIVES
          BMI ENDSHO      ;<0!!
          ORA #$90        ;ADD COLOR 3
          STA SCOLIN+19   ;SHOW IT!
ENDSHO
          RTS             ;ALL DONE!
```

FAMILY FINANCES
ATARI, INC.
Sunnyvale, CA 94086
32K Disk $49.95

by Bob Curtin

One of the really great things about dropping by the ANALOG offices is seeing all the new software products for Atari computers and getting the latest scuttlebutt about upcoming or existing hardware.

Sometimes it seems, though, that the less flashy—but certainly no less useful—software gets lost in the shuffle. (Picture your local plumber being trampled by a crowd stampeding to catch a glimpse of Michael Jackson.) Well, **Family Finances** is just that kind of software package—it's hardworking, incredibly useful, friendly and reliable, but lacking in spectacle and glitter.

Written by Jerry Falkenhan, **FF** has actually been around for over two years, but you've seen it in the familiar APX boxes, being sold as two separate programs. Atari, in one of their brighter moments, combined the two programs into one package, dressed it up in nifty Atari bookcase packaging and renamed it. I've been using the originals for two years, and this review is based on that experience. The few changes made to the programs (aside from the cosmetics) only serve to improve them; functionally, they remain unchanged.

Family Finances is, as before, broken up into two parts. The first disk contains the Family Cash Flow (FCF) program, and the second, the Family Budget (FB) program. Let's take a look at each in turn.

Where did it all go?

FCF is designed to provide a detailed reporting of where your bucks have gone and it's superb in that role. The program allows up to 100 individual expense entries in 13 categories and 20 income entries in 13 categories. All 26 categories are user-definable, and if you're lucky enough to have two disk drives, the storage capacity is doubled, allowing 200 entries in expense and 40 entries each month in the income mode. Even with one disk drive, I've never come close to the 100-entry limit in a single month.

The input consists of an entry number (which can come in handy as a reference number on receipts or checks), the date, a label or description up to fifteen characters in length and, of course, the amount of the income or expenditure. All data inputs are two-step routines, so that you can change your mind before committing the data to memory.

One of the things that make **Family Finances** such a joy to use is this amicability. You can almost fire it up and use it without referring to the documentation—which, by the way, is impeccably written, complete and put together in a neat little booklet that's just right for reference when you need it. Access to the various modes is accomplished through one-touch commands, and any invalid inputs are immediately trapped, flagged and looped back for another try.

The displays are clean and easy to read, with trailing zeros and right justification on all columns. Mr. Falkenhan has made clever use of the keyboard graphics symbols in his menu displays, and the dark text on light background makes the whole package easy on the eyes.

But the pleasure of this program is in the using. The different modes allow you to review your income or expenditures on several different levels. In addition to the screen displays, hard copies can be had in each of the different modes, including the detail of expenditures (which eats printing paper at a frightening rate, but these allow you to make comparisons of your data from month to month). FCF gives you a good, detailed recording of your expenditures and income—standing alone, it does a magnificent job in that respect. However, coupled with the analysis capability provided in the FB section, it becomes a powerful tool for helping with the household finances.

Budgeting made painless.

The second disk, FB, contains the means to set up and maintain a monthly budget, as well as to provide a thorough analysis of the data stored on your FCF disk. Again, the program is multi-leveled. Once you've projected your budget and filed it, **Family Finances** takes the data from the FCF disk and compares it to your budget figures.

Your entire budget—or any part of it, both income and expenditures—can be analyzed in a host of different ways, on a monthly or yearly basis, or for any period in between. Budgeted vs. actual spending categories in yearly and monthly modes, or single-category comparisons, can be conjured up at will and dumped to a printer, if you wish.

As with FCF, the options open to you are self explanatory, and there are prompts and screen information all along the way to help you out. Unlike a lot of programs—where, once you get a handle on it, the on-screen help turns into a hindrance—**Family Finances** does no such thing. It just keeps doing a superb job. . .quickly and efficiently.

An interesting aside: **FF** is written in Atari BASIC and is completely open to examination and modification. For those who are new to BASIC programming and want to see how the pros operate, you'll be able to do just that with this product.

The price makes this package a real value. Moreover, when used to even a fraction of its potential, **Family Finances** can save you scads of money, by tagging unwanted trends and unwise spending habits. I heartily recommend it to anyone looking for a fast, accurate and easy way of integrating their computer into the household financial chores. □

Another BASIC Bug

by R.T. Dolbeare

I recently spent several exasperating evenings trying to track down the source of an error in one of my BASIC programs. My problem was that the first character of one of my string variables was inexplicably altered during program execution. I was finally able to determine that the alteration was occurring immediately after a "GET" statement, but the reason for the change was certainly not apparent.

After more frustrating evenings, I finally found a previously-undocumented error in Atari BASIC. It seems that use of the "VAL" function and the GET statement in the same program causes the very undesirable string variable alteration described above.

The obvious solution to the problem is not to use both functions in the same program. Certainly, the results of the VAL function can be simulated in other ways, so that one need never use that function. However, there is a way to use both functions in the same program—once the nature of the problem is understood. Suppose we have a program segment such as:

```
100 OPEN #1,4,0,"K:"
110 DIM S$(8)
120 S$="84-06-30":REM DATE YR-MO-DA
130 YEAR=VAL(S$):REM RETURNS 84
140 ? "BEFORE GET STATEMENT, S$=";S$
150 ? "HIT ANY KEY":GET #1,D
160 ? "AFTER GET STATEMENT, S$=";S$
170 END
```

Type in and run this simple program, and you will find that *S$* prints normally in Line 140, but that, at Line 160, the *8* in *S$* has been replaced by whatever character you've hit in response to Line 150. If you hit the RETURN key, then the *8* was replaced by a *RETURN*, which inserted a blank line in the output. A little experimentation will show that, if several string variables and VAL functions are used, only the string variable last used with the VAL function is affected by the subsequent GET statement. The seemingly obvious (but *still* improper) fix is to insert an extra line prior to the GET statement with a VAL function, such as:

```
135 X=VAL("1")
```

This seems to work fine the first time the program is run. However, list the program after execution has been completed and you will find that the program itself has been altered. The *1* in Line 135 will be replaced in the listing by whatever character you hit for the GET statement. If this is a non-numeric character, then *ERROR 18* (invalid string character) results the next time the program is run. The program listing really looks messy if you've entered a RETURN. *This has to be the worst of bugs—one which actually alters your program during execution.*

There *is* a way around this problem. . . by defining a dummy string variable as shown below:

```
135 DIM DUMMY$(1):DUMMY$="1":X=VAL(DUM
MY$)
```

Now, when the program is executed, the contents of DUMMY$ will indeed be altered, but the program listing itself will remain unchanged. As long as the actual contents of DUMMY$ are not important, then no problem results. □

Graphics 8 Character Generator

16K Cassette or Disk

by Tom Hudson

Here at **ANALOG**, we often receive requests for specific utility programs. Sometimes the requests are beyond the scope of a simple article, but often the staff has a solution right in hand.

Such was the case with John Chung's request for a subroutine which would print text in graphics mode 8, to allow mixing text with charts and other graphics. Luckily, I happened to have written just such a subroutine for my **Retrofire** game in **ANALOG** issue 14.

I've modified the routine so that it fits on page 6, can be called by BASIC with a simple USR call and prints text almost any way you could possibly want it. It'll even work with an alternate character set!

The listings.

Listing 1 is the BASIC code necessary to install the character generator into your computer's memory. Type in this code and verify your typing with **C:CHECK** or **D:CHECK2**. When you're sure your typing is correct, save the code to your storage device.

Whenever you want to use the character generator, the code in Listing 1 is all you need to get the routine ready to use.

Listing 2 is a short program which demonstrates the use of the character generator subroutine. After you've typed in this code and checked your typing, add it to Listing 1.

Using the subroutine.

When you have merged the two listings, RUN the program. After a few seconds, your screen should look like Figure 1.

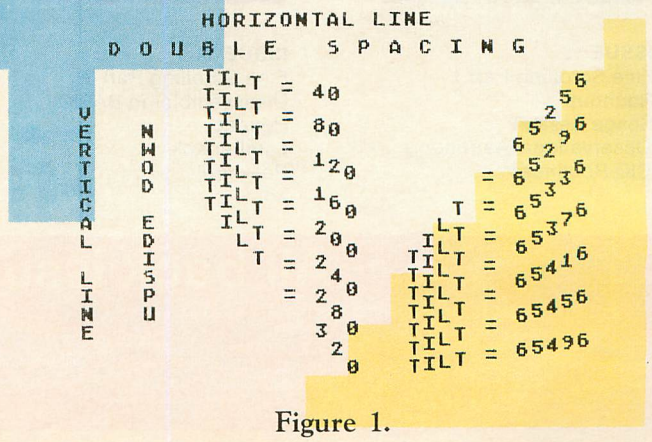


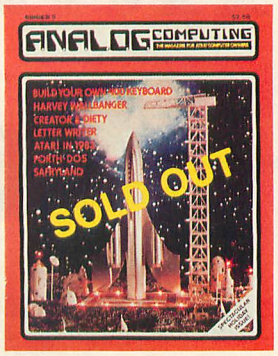
Figure 1.

BACK ISSUES

Catch up on what you've missed!

ISSUE 2
Wasting Arrays
Atari's CPU
Download Terminal
Converting BASIC
Disk Files

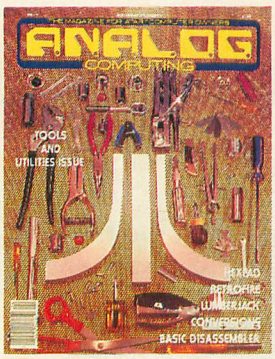
ISSUE 8
GTIA Graphics
Audio in Your Programs
NOREM
Graphic Violence
Color Slot Machine



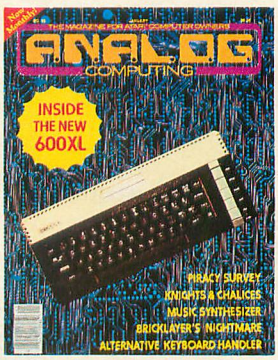
ISSUE 9
Build Your Own 400
Keyboard
Harvey Wallbanger
Forth-Dos
Letter Writer

ISSUE 11
Strings in BASIC
C:CHECK
Disk Cataloging Utility
Adventure in the
Fifth Dimension
Moving Missiles in BASIC

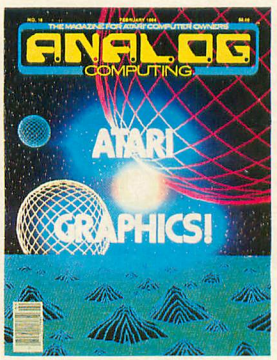
ISSUE 13
Fine Scrolling Part 1
Roundup
Space Assault
Observational Astronomy
CIO Routines



ISSUE 14
Fine Scrolling Part 2
Disassembler in BASIC
Hexpad
Lumberjack
Retrofire!



ISSUE 15
Fine Scrolling Part 3
Knights and Chalices
Music Synthesizer
Bricklayer's Nightmare
Alternative Keyboard
Handler



ISSUE 16
Fine Scrolling Part 4
Create-A-Font
Bar Chart Subroutine
Shooting Stars
3-D Object Rotation

All back issues $4.00 each

Send check or money order to:
ANALOG Back Issues
P.O. Box 615
Holmes, PA 19043

Issues 2, 11, 21 and 22 are also still available.
Order now—limited quantities available.

MasterCard and Visa orders call:
1-800-345-8112
in PA. 1-800-662-2444

The screen looks like a graphics 0 display, but there's no easy way you could generate the diagonal text on a graphics 0 screen. All that text is generated on a normal graphics 8 screen! If you don't believe this, add the following line to the program and re-RUN it.

```
685 COLOR 1:PLOT 0,0:DRAWTO 319,159
```

Now, when the program executes, it will draw a diagonal line across the screen. Now do you believe me? Let's see what makes this program tick.

Before we start looking at the beginning of the program let's look at the end of it, where the heart of the program lies.

Lines 700 - 730 are a short subroutine which calls the machine language subroutine and prints the text on the graphics 8 screen. The BASIC USR statement is used and is in the following form:

```
A=USR(1536,X,Y,ADR(A$),LEN(A$),TILT)
```

The first parameter in the USR parentheses is 1536. Do *not* change this number. It is the address of the machine language subroutine. If you look at Line 4000, you will see that the routine is placed in the address range 1536 - 1770.

The next parameter is the variable X. This number, ranging from 0 to 39, indicates the character's horizontal position on the screen. X coordinates larger than 39 will give unknown results, so be careful.

The third parameter is the variable Y. This number ranges from 0 to 184 and indicates the vertical position of the top of the character on the screen. This value corresponds with the graphics 8 Y-coordinate system. You can place a character anywhere on the screen vertically.

The fourth parameter, ADR(A$), is the address of the string containing the text we want to print. In this demo program, we're using the string A$ to hold the text. If you're always going to print the same message, you can imbed it in the USR call. For example, to always print the message "HELLO," simply replace ADR(A$) with:

```
ADR("HELLO")
```

The fifth parameter, LEN(A$), is the length of the string we're printing. If you know the length of the string is going to remain constant, you can simply replace LEN(A$) with the number. In the earlier example with the message HELLO, the length would always be 5. Be sure that the message never goes past the right side of the screen (X = 39), since the program was not designed to handle this.

The sixth parameter, TILT, tells the subroutine how to display the text. A TILT of 0 will product normal text. A TILT of 319 will print text vertically. If TILT is a multiple of 40, the text will be plotted diagonally. The demo program will demonstrate the use of the subroutine with different TILT values.

Now let's step through the rest of the program.

Line 170 GOSUBs to Line 4000, the character generator setup subroutine. The subroutine simply reads the machine language data and POKEs it into page 6. When using the character generator in your own programs, the GOSUB 4000 statement should be one of the first things you do. This step only needs to be performed once.

Line 180 sets up the graphics 8 screen. You can use the character generator with either a full-screen or split-screen graphics 8 mode.

Line 190 dimensions the string A$. We will use this string to hold the text we want to put on the screen. The character generator will accept any Atari character, even inverse and control characters. Simply place them in the string and call the routine.

Lines 200 - 260 show the parameters needed to print a normal horizontal text line. If you look at Figure 1, you'll see this line printed at the top. Note that for a horizontal line the TILT value is 0.

Lines 270 - 330 show how to double-space a text line. Simply set TILT to 1, and the text is spread out, with one space between each character. A TILT of 2 would place two spaces between each character, and so on.

Lines 340 - 400 print a message vertically. The only special action necessary here is to set TILT to 319. If you want to double-space vertically, set TILT to 639.

Lines 410 - 470 do something I've yet to find a practical use for, but it's interesting. By setting TILT to 65215, you can print the text vertically, *backwards*! Note that the Y position must be set to the top of the lowest character.

Lines 480 - 570 print a group of eight diagonal lines. They are printed on the screen with the corresponding TILT setting. As you can see, multiples of 40 produce nice diagonal messages.

Lines 580 - 680 print seven lines that are reverse-slope diagonals. This is achieved by subtracting the normal TILT value from 65536. If you're familiar with the binary number system, you'll realize that this produces a negative number.

Lines 700 - 730 call the machine language subroutine and actually print the text.

Your own applications.

What can you do with this character generator? Some obvious uses that come to mind are labeling graphs, charts, diagrams and other graphics 8 pictures. Figure 2 shows a couple of fancy things I did in about twenty minutes with the generator.

(continued on page 60)

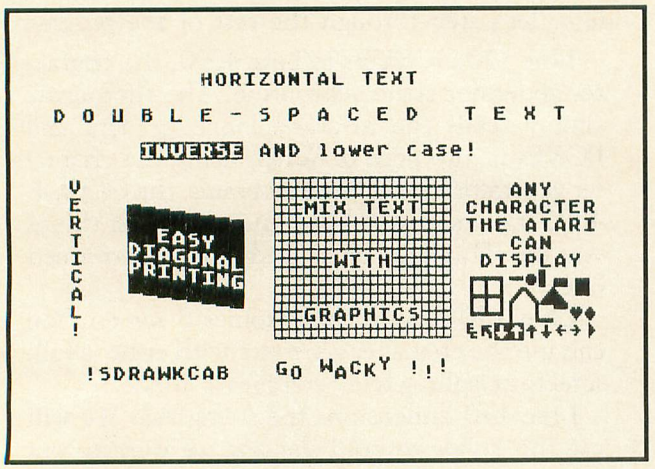


Figure 2.

You aren't limited to just the Atari characters, either. You can develop your own character set and plot those characters on the screen. All you have to do is place the character set in memory and POKE its base address into CHBAS (756 decimal). The character generator does the rest.

Finally, I'd like to thank John Chung of Alexandria, Virginia for prompting me to write the subroutine. The code itself had been sitting around for quite a while, but his letter actually made me sit down and turn it into a useful product. □

Listing 1.

```
4000 FOR X=1536 TO 1770:READ N:POKE X,
N:NEXT X:RETURN
4010 DATA 216,104,104,104,133,203,104,
104,133,204,169,0,133,205,6,204,38,205
,6,204,38,205,6,204,38
4020 DATA 205,165,204,24,101,88,133,20
6,165,205,101,89,133,207,6,204,38,205,
6,204,38,205,165,204,24
4030 DATA 101,206,133,206,165,205,101,
207,133,207,165,206,24,101,203,133,206
,141,240,6,165,207,105,0,133
4040 DATA 207,141,241,6,104,133,213,10
4,133,212,104,104,141,236,6,206,236,6,
104,141,239,6,104,141,238
4050 DATA 6,169,0,141,237,6,169,0,141,
235,6,172,237,6,177,212,16,5,206,235,6
,41,127,201,32
4060 DATA 176,5,24,105,64,16,7,201,96,
176,3,56,233,32,133,204,169,0,133,205,
133,208,6,204,38
4070 DATA 205,6,204,38,205,6,204,38,20
5,165,205,24,109,244,2,133,205,164,208
,177,204,77,235,6,172
4080 DATA 237,6,145,206,230,208,165,20
8,201,8,240,13,165,206,24,105,40,133,2
06,144,227,230,207,208,223
4090 DATA 238,237,6,206,236,6,48,26,17
3,240,6,24,109,238,6,133,206,141,240,6
,173,241,6,109,239
4100 DATA 6,133,207,141,241,6,76,106,6
,96
```

•

CHECKSUM DATA.
(see page 25)

```
4000 DATA 170,883,129,480,161,243,321,
97,388,873,386,4131
```

•

Listing 2.

```
100 REM ********************************
110 REM *   GR. 8 CHARACTER DEMO       *
120 REM *                              *
130 REM *      BY TOM HUDSON           *
140 REM *                              *
150 REM *     ANALOG COMPUTING         *
160 REM ********************************
170 GOSUB 4000:REM *** SET IT UP ***
180 GRAPHICS 8:SETCOLOR 2,0,0
190 DIM A$(20)
200 REM
210 REM LET'S DO A HORIZONTAL LINE!
220 REM
230 X=12:Y=0
240 A$="HORIZONTAL LINE"
250 TILT=0
260 GOSUB 700
270 REM
280 REM DOUBLE SPACING
290 REM
300 X=6:Y=12
310 A$="DOUBLE SPACING"
320 TILT=1
330 GOSUB 700
340 REM
350 REM NOW A VERTICAL LINE
360 REM
```

```
370 X=4:Y=40
380 A$="VERTICAL LINE"
390 TILT=319
400 GOSUB 700
410 REM
420 REM HOW ABOUT INVERTED VERTICAL?
430 REM
440 X=8:Y=128
450 A$="UPSIDE DOWN"
460 TILT=65215
470 GOSUB 700
480 REM
490 REM NOW SOME NORMAL DIAGONALS
500 REM
510 X=12:Y=24
520 A$="TILT =    "
530 FOR TILT=40 TO 320 STEP 40
540 A$(8)=STR$(TILT)
550 GOSUB 700
560 Y=Y+8
570 NEXT TILT
580 REM
590 REM AND SOME REVERSE DIAGONALS!
600 REM
610 X=25:Y=152
620 A$="TILT =      "
630 FOR T2=40 TO 280 STEP 40
640 TILT=65536-T2:REM *** REVERSE! ***
650 A$(8)=STR$(TILT)
660 GOSUB 700
670 Y=Y-8
680 NEXT T2
690 END
700 REM
710 REM NOW PLOT THE TEXT!
720 REM
730 A=USR(1536,X,Y,ADR(A$),LEN(A$),TIL
T):RETURN
```

CHECKSUM DATA.

(see page 25)

```
100 DATA 539,424,15,84,21,325,557,519,
842,589,76,617,82,715,378,5783
250 DATA 722,991,97,392,103,683,55,717
,984,90,742,96,707,109,791,7279
400 DATA 977,83,269,89,968,645,897,998
,104,772,82,976,369,758,570,8557
550 DATA 994,381,242,106,764,84,830,34
2,565,629,575,999,392,624,63,7590
700 DATA 86,546,92,113,837
```

●

Assembly language listing.

```
; GRAPHICS 8 CHARACTER GENERATOR
;
; BY TOM HUDSON
;
; ANALOG COMPUTING
;
XPOS    =   $CB     ;CHAR. X SAVE
LO      =   $CC     ;2-BYTE...
HI      =   $CD     ;POINTER
DESTLO  =   $CE     ;DESTINATION...
DESTHI  =   $CF     ;POINTER
COPCNT  =   $D0     ;BYTE COPY COUNT
COPIX   =   $D1     ;COPY INDEX
CHADLO  =   $D4     ;CHARACTER...
CHADHI  =   $D5     ;DATA POINTER
SAVMSC  =   $58     ;SCREEN ADDRESS
CHBAS   =   $02F4   ;CHARACTER SET
;
; LOCATE ROUTINE ON PAGE 6
;
        *=  $0600
SHONUM  CLD         ;NO DECIMAL MODE!
        PLA         ;DISCARD
        PLA         ;DISCARD HI
        PLA         ;PULL X POSITION
        STA XPOS
        PLA         ;DISCARD
```

```
          PLA             ;PULL Y POSITION
          STA LO          ;SET UP...
          LDA #0          ;2-BYTE...
          STA HI          ;ADDRESS POINTER
          ASL LO          ;*2
          ROL HI
          ASL LO          ;*4
          ROL HI
          ASL LO          ;*8
          ROL HI
          LDA LO          ;NOW ADD ADDRESS
          CLC             ;OF START OF
          ADC SAVMSC      ;SCREEN
          STA DESTLO
          LDA HI
          ADC SAVMSC+1
          STA DESTHI
          ASL LO          ;*16
          ROL HI
          ASL LO          ;*32
          ROL HI
          LDA LO          ;NOW ADD...
          CLC             ;PREVIOUS RESULT
          ADC DESTLO      ;TO GET
          STA DESTLO      ;SCREEN + Y*40
          LDA HI          ;(1ST BYTE OF
          ADC DESTHI      ;DESIRED LINE)
          STA DESTHI
          LDA DESTLO      ;NOW ADD X POS
          CLC             ;FOR PLOT BYTE!
          ADC XPOS
          STA DESTLO
          STA LOSAVE
          LDA DESTHI
          ADC #0
          STA DESTHI
          STA HISAVE
          PLA             ;PULL...
          STA CHADHI      ;STRING ADDRESS
          PLA             ;AND SAVE IT
          STA CHADLO      ;ON PAGE 0
          PLA             ;DISCARD
          PLA             ;PULL & SAVE...
          STA LENGTH      ;STRING LENGTH
          DEC LENGTH      ;MAKE 1 LESS!
          PLA             ;PULL & SAVE...
          STA ADDFCH      ;ADD FACTOR
          PLA
          STA ADDFCL
          LDA #0          ;ZERO OUT...
          STA CHRNUM      ;CHARACTER #
CLOOP     LDA #0          ;RESET INVERSE
          STA INVERSE
          LDY CHRNUM      ;GET CHAR INDEX
          LDA (CHADLO),Y  ;GET CHAR#
          BPL NOINV       ;NOT INVERSE
          DEC INVERSE     ;SET INVERSE FLAG
          AND #$7F        ;MAKE NOT-INVERSE
NOINV     CMP #32         ;CHAR < 32?
          BCS NOTL32      ;NO, CHECK MORE
          CLC             ;ADJUST NUMBER
          ADC #64         ;UP BY 64
          BPL GOTSCR
NOTL32    CMP #96         ;CHAR < 96?
          BCS GOTSCR      ;NO, IT'S OK
          SEC             ;ADJUST NUMBER
          SBC #32         ;DOWN 32
GOTSCR    STA LO          ;AND CALC...
          LDA #0          ;CHSET INDEX
          STA HI
          STA COPCNT
          ASL LO          ;*2
          ROL HI
          ASL LO          ;*4
          ROL HI
          ASL LO          ;*8
          ROL HI
          LDA HI          ;NOW ADD CHSET
          CLC             ;BASE ADDR
          ADC CHBAS       ;(CAN BE ANYWHERE)
          STA HI
COPNUM    LDY COPCNT      ;GET COPY COUNT
          LDA (LO),Y      ;AND CHAR BYTE
          EOR INVERSE     ;INVERSE IF REQ'D
          LDY CHRNUM      ;GET SCRN OFFSET
          STA (DESTLO),Y  ;PUT ON SCREEN!
          INC COPCNT      ;NEXT COPY BYTE
          LDA COPCNT      ;COPY DONE?
          CMP #8          ;(8 BYTES COPIED)
          BEQ NXTCHR      ;YES!
          LDA DESTLO      ;NO, ADD 40...
          CLC             ;TO POINT TO...
          ADC #40         ;NEXT SCR.8...
          STA DESTLO      ;SCAN LINE
          BCC COPNUM
          INC DESTHI
          BNE COPNUM
NXTCHR    INC CHRNUM      ;NEXT CHARACTER
          DEC LENGTH      ;MORE CHARACTERS?
          BMI FINISH      ;NO!
          LDA LOSAVE      ;OK, RESTORE...
          CLC             ;SCREEN POINTER
          ADC ADDFCL      ;WITH VERTICAL/
          STA DESTLO      ;HORIZONTAL ADD
          STA LOSAVE
          LDA HISAVE
          ADC ADDFCH
          STA DESTHI
          STA HISAVE
          JMP CLOOP       ;AND DO NEXT ONE!
FINISH    RTS             ;ADIOS!
;
; MISCELLANEOUS WORK AREAS
;
INVERSE   *=  *+1         ;INVERSE VIDEO FLAG
LENGTH    *=  *+1         ;STRING LENGTH
CHRNUM    *=  *+1         ;CHARACTER #
ADDFCL    *=  *+1         ;LINE ADD FACTOR
ADDFCH    *=  *+1
LOSAVE    *=  *+1         ;ADDRESS...
HISAVE    *=  *+1         ;HOLD AREA
          .END
```

The Only Really Usable Color Printing System For All Atari Computers.

And many more. See below.

Koala Pad, Movie Maker, B/Graph, Atari Touch Tablet, Micro Painter, Music Construction Set, Fun With Art, Graphic Master, G. Chip Cookbook, Logo, Graph It and Infocom Adventures (plus others — see your dealer).

A rainbow of color and black and white too. That's what you'll get from our GP-700AT — the only printer you'll ever need for your Atari computer.

First the Color: Ours is the ONLY full-color printer that's supplied with exclusive "Rainbow"* software to give color printouts from virtually all color graphics programs for your Atari computer. It connects directly to your Atari (no 850 interface required). Just plug it in and a whole wide world of printed color is instantly yours. Our "Rainbow" software converts your images from the Atari Touch Tablet, Koala Pad, Logo, and most other popular color graphics programs to fun color printouts on plain paper. We even include the "G. Chip Cookbook" free on disk so you can start color printing immediately.

Black and White Too: The GP-700AT is also a high quality dot matrix black and white printer. It has friction and tractor feed, makes multiple copies, and can do letters, invoices, labels, word processing and more.

The GP-700AT is priced at just $599 — available at leading department and computer stores, everywhere.

For the outlet nearest you, phone 818-365-9521.

*"Rainbow" software by Computer E.A.S.E.

AXIOM®

AXIOM CORPORATION

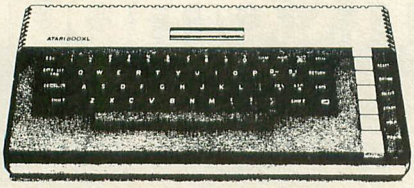
1014 Griswold Avenue • San Fernando, CA 91340
Telephone: (818) 365-9521 • TWX: 910-496-1746

GP-Series Printers Built for Lasting Quality by SEIKOSHA.

CIRCLE #125 ON READER SERVICE CARD

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

ATARI®



ATARI PLATO SYSTEMS

The Electronic Highway To The Universe
Over 6000 Educational Programs

SYSTEM I	SYSTEM II
Atari 1030 Modem Atari Plato Cartridge	Atari 850 Interface Module Atari Plato Cartridge Anchor Mark XII Modem
$159	**$549**

ATARI HOME COMPUTER

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

DISK DRIVES

RANA 1000	CALL FOR UPDATED PRICES
*TRAK ATD 2 (Single/Double Density/Parallel Int. 2K Buffer	
*TRAK ATD4 (Dual Drive)	
TRAK TURBO DOS	
*INDUS GT (Free Software)	
ASTRA 1620	
ATR - 8000	
HAPPY ENHANCEMENT for Atari 810 and 1050 Disk Drives	
MY-DOS for ATR-8000	

ATARI ADDITIONAL EQUIPMENT

ATARI 1010™ Program Recorder	CALL FOR UPDATED PRICES
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™	

*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	99.95

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

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

DEALER INQUIRIES INVITED

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)	555.00
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)	369.95

PRINTER INTERFACE CABLES

APE FACE	65.00
APE FACE with serial port	79.00
MPP-1150 Parallel Printer Interface	79.00

PRINTER RIBBONS

GEMINI Printers (Black/Blue/Red/Purple)	3.00
EPSON Printers	6.95

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	84.95
Sanyo 12" Amber Screen	89.95
Sanyo Color Screen	229.95

MODEMS

MPP-1000C Modem	129.00
Signalman Mark XII Modem	269.00

***DEALER INQUIRIES INVITED**

To order call **TOLL FREE**

1-800-824-7506

ORDER LINE ONLY

No extra charge for MasterCard or Visa

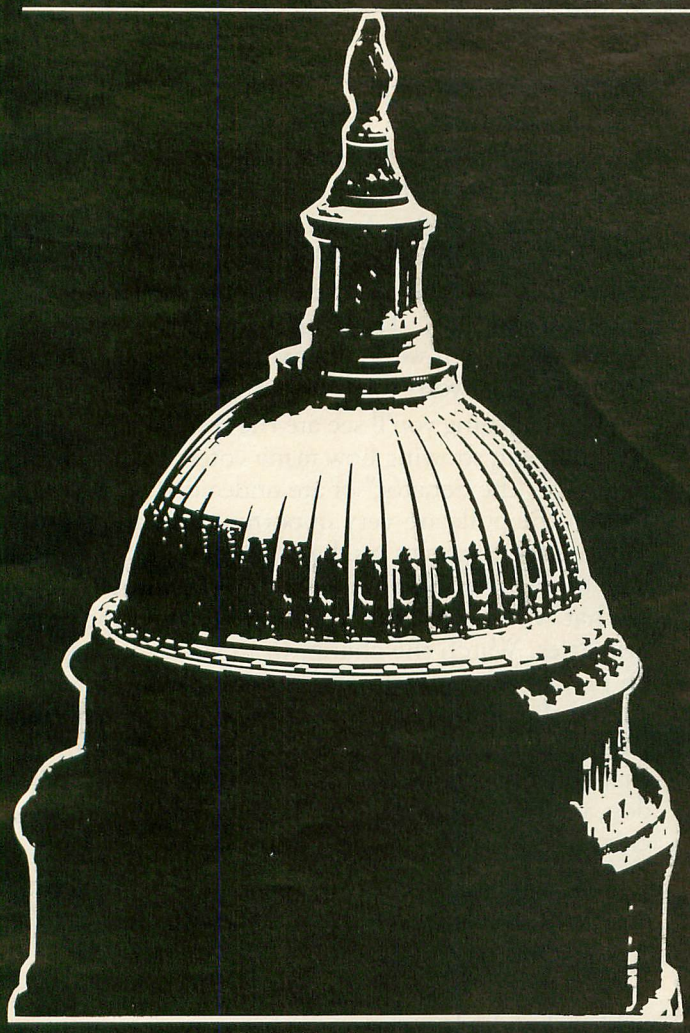
COMPUTER CREATIONS, Inc.

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.

CIRCLE #126 ON READER SERVICE CARD



DARK HORSE

A presidential campaign simulation

16K Cassette or 24K Disk

by Kenneth Amidon and Wayne Underwood

Dark Horse is a single-player simulation of a presidential primary campaign. It takes place in the near future, and you play the part of an underdog candidate. Your backers provide you with a substantial "war chest" and give you complete freedom to use the funds as you see fit. At the start of each of the next nine "weeks," you decide where and how to spend your funds, then watch the results come in. This is a game of strategy and planning, calculated to satisfy the lust for power in those of us whose purses (or ideals) make political office an unlikely goal.

The setup.

Type in the program exactly as it appears. Be careful with the data statements. Use **D:CHECK** or **C:CHECK** to check your work. Load the program and, when the READY prompt appears, turn the volume of your TV to a moderate level and type RUN.

Choosing your level of play.

After a brief introduction, screen number 1 will appear. You can change the level of play by pressing the SELECT key. Each level starts you off with a different amount of money in your campaign fund. Beginners receive $20 million; intermediate players get $17 million; and experts have a mere $14 million. We recommend playing at the beginner level, at least until you are familiar with how the program operates.

An *important* note: don't spend more money than you have. If your campaign funds drop to zero, your backers will abandon you—and you'll automatically lose!

Once you have determined the level you want to start at, by using the SELECT key, you should press START to begin the game. Note that the BREAK key has been shut off, so you cannot use it to end the game. Pressing SYSTEM RESET at any time will end the game and return you to BASIC. If you want to play again, simply type RUN followed by the RETURN key.

Weekly report and main menu.

After you have turned up the volume on your TV and pressed START, screen number 2 will be displayed. There are two important parts to this screen: the weekly report (top of screen) and the main menu (bottom of screen). Let's look at the main menu section first.

There are nine numbers, followed by a description of the screen you will see if you enter that number:

The first six numbers will provide you with in-

formation about each of the six regions that the nation has been divided into for purposes of this simulation.

Pressing the number *7* will bring you to the resource menu.

Number 8 can be used to return to the weekly report screen when viewing a region.

The number 9 should be pressed when you have finished all your resource allocations for the week and are ready to "get out the vote."

Note: all main menu choices (the numbers *1* through 9) are single-key input. You do not need to press RETURN. Any input errors will simply redisplay the main menu at the bottom of the screen.

Now, let's take a look at the weekly report section of screen number 2. This report provides you with basic information which you need to judge how well your campaign is progressing.

The first thing it tells you is the week number. Remember, there are nine weeks to the primary season, so budget wisely at the beginning. The week number is also important because different states within each region have primaries on different weeks. More about this under the discussion of regional reports.

Next, you will see the total campaign funds available to you. This figure is automatically reduced as you allocate funds to various resources. Don't overspend in the later weeks—if you spend all your money before the last week, you will lose!

Following the figure for campaign funds, the screen will display the states that have primaries that week. Below this, you will see the number of delegates you have won and lost. These figures will change at the beginning of each week, after you have allocated your resources and won—or lost—primaries.

The next thing you'll see are the results of the latest national poll, showing how many voters favor you (the player) or "the Senator," or are undecided. Your standings in the polls are very important and are a direct result of the resources you have used in the past weeks. The more favorable your standing, the more likely it is that voters will choose you in the upcoming primaries. Watch the change from week to week and be prepared to spend extra money if your standing should slip. Remember that the Senator, an old campaigner, has a solid block of supporters. It's unlikely that you will take many away from him. As the underdog, your goal is to sway the undecided voter to your camp.

Finally, if you successfully seek a debate with your opponent, the announcement of the upcoming debate will appear on this screen.

Regional reports.

Now, let's look at the regional reports (screen number 3). Press the number *1* key. This will bring you to the current report for the North Atlantic region.

The name and number of the region are displayed, along with the current week number. Then you'll see a list of the states in that region, the number of delegates each state has and the week number when each state holds its primary. The last column will show whether you won or lost a particular state after its primary is held.

Because resources are allocated on a regional basis, not state-by-state, it is important to know the total delegates available in a region for the current week. You may want to make a note of this figure before you begin allocating resources.

Once you have distributed resources, this regional report will also tell you what you have done in that particular locale.

All six regional reports have a similar structure. You should review all of them each week, *before* you begin designating resource use. And, a quick review before you end the week's orders will tell you if you've accomplished everything you wanted to that week.

When you are through reviewing the current regional reports, press the number 8 key to return to the weekly report. You can go directly to the resource menu by pressing number *7*, but you should note your delegate count and poll results before you start allocating your precious dollars.

Resource menu.

When you press the number 7 key from a main menu screen, you will see the resource menu, screen number 4. As with the main menu, you can go to any screen on the resource menu with a single-key input—you don't need to press the RETURN key. In this case, the key entries are the letters A through G. Any input errors during the resource allocation phase of the simulation will automatically return you to this screen.

You're now ready to decide how you will conduct your campaign for the current week. You will spend your funds on various resources in an attempt to increase your standing in the national poll and to win the delegates available in the current week. Let's look at each resource.

A — ALLOT COORDINATORS (screen 5). First, press A to get to the allot coordinators screen. In addition to telling you how much is left in your campaign fund, this screen tells you how many coordinators have been assigned to each region. The maximum number of coordinators is thirty-six, but you can assign as many of these as you want to a region.

Each coordinator you assign to a region will cost you $7500 per week. The "Current Coords Cost" line shows how much it will cost to maintain the current allocation of coordinators for the week. At the bottom of the screen, you are prompted to enter the number of the region you wish to change. Press the number, then the RETURN key.

Note that, unlike the single-key entries from the main and resource menus, making changes or allocating resources demands that you use the RETURN key. This procedure allows you to correct any mistakes (with the DELETE/BACK-SPACE key) before you press RETURN.

After entering the number for the region you want to change, you will be prompted to enter the number of coordinators you want in the region. This is an absolute number, not an addition to any existing coordinators. For example, if Region 1 displays three coordinators, and you want four in that region, you enter a *4* when prompted for the number of coordinators. The screen will change Region 1 from three to four coordinators. After you have assigned coordinators in the way you want, examine the "Current Coords Cost" line.

It's important to realize that the cost of coordinators is *not* deducted from your current campaign funds until the end of the week. That is, after you have ended the week's play by pressing 9. Therefore, be sure that the current coordinator cost is less than the available campaign funds, to avoid losing by overspending.

When you are satisfied with your coordinator distribution, you can return to the resource menu by simply pressing RETURN. This is true for all resource allocation screens. You return to the resource menu at any time by pressing RETURN with no other input.

B — BUY ADVERTISING (screen 6). When the resource menu is displayed, press *B* to go to the buy advertising screen. You will be prompted at the bottom of the screen to enter the region number where you want to advertise. Enter the number (*1* through 6) and press RETURN. The next prompt asks how much you want to spend. You *must* spend at least $100,000. Enter the amount without any spaces, commas or dollar signs. For example, $200,000 would be entered as *200000*. If you want to reduce an amount previously allocated, you may enter a negative amount by using the minus sign. Example: you have $300,000 allocated to Region 1 but would like to reduce this to $100,000. You should enter *−200000*.

The funds you spend on advertising will be automatically deducted from your available campaign funds. When you are through allocating

(continued on next page)

funds for advertising in all regions, press the RETURN key to return to the resource menu.

C — CANVAS/MASS MAILING (screen 7). From the resource menu, press C to reach the canvas/mass mailing screen. Unlike advertising, this "PR" costs a standard amount per region: $200,000. You cannot spend more or less, although you do not *have* to spend anything.

The prompt at the bottom of the screen asks you to input the region number where you want a canvas campaign this week. Enter a number (*1* through 6) and press RETURN. The screen will display the region number chosen and the cost.

It's important to remember that, once you have entered a region number and pressed RETURN, you cannot change your allocation. *Be sure* that you are allocating canvassing funds to the region you want before you press RETURN.

When you have made all the allocations you want, press the RETURN key without any other input to return to the resource menu.

D — DEBATE (screen 8). Press *D* from the resource menu, and the debate screen will display. You will be able to have only one debate during the nine-week primary season. Attempting to get the Senator to debate costs $100,000—for each attempt. If you're successful, and the Senator agrees to debate, it will cost you an additional $250,000.

Your success at getting a debate depends heavily on your standing in the national poll—the lower your percentage, the less likely the Senator is to agree. However, you can reach a point where your polls are so high that the Senator will agree because it will help *his* campaign—and will actually hurt yours.

In addition to the problem of timing in regard to your campaign success, you should keep in mind that a debate has a strong effect on your poll standings and chances to win delegates for several weeks. Try to get a debate when you have two or three weeks of important, large primaries coming up.

If you answer Y to the debate prompt, your campaign funds will be reduced by $100,000, and the resource menu will reappear. Any other response will simply return you to the resource menu.

E — ELECTION APPEARANCE (screen 9). Enter *E* and press RETURN from the resource menu to display the election appearance screen.

You may schedule personal appearances in only one region per week. Assume you make several stops in each state in the region. These appearances will cost you $200,000 for the week.

You choose the region in which you will appear by answering the prompt with a region number (*1* through 6) and pressing RETURN. Again, once you enter a number and RETURN, you are committed to the allocation. Make *sure* your entry is correct before you press RETURN. After selecting a region, you will be returned to the resource menu. If you do not want any personal appearances, simply press RETURN without any other input.

F — FUND RAISER (screen 10). Pressing *F* from the resource menu will display this screen. You may have one fund raiser each week, but it will cost you $50,000 to organize and conduct. How much money you make (or lose!) will depend on the past success of your campaign. The results of any fund raiser will be announced after the primaries for the week are over.

If you want a fund raiser, enter a Y and press the RETURN key. You will go back to the resource menu. Any other response will be interpreted as a "no" answer, and the resource menu will be displayed.

G — GOTO WEEKLY REPORT. Pressing G from the resource menu will display the weekly report screen. Use this command when you have finished all of your resource allocations.

Ending the week's orders.

After you've made all your resource allocations and reviewed the regional reports to make sure everything is the way you want it, you're ready to end the current week's play. You do this by pressing number 9 from either the weekly report or one of the regional reports. Screen number 11 will display.

This screen gives you another chance to return to your resources and make additional changes. If you are certain that every detail is as you want it, enter Y and press RETURN. *You are now committed to using your resources as planned.* Any response except Y will return you to the current weekly report.

The voting.

Once you have ended the week's orders, **Dark Horse** will display several screens that give you important information, while the program tabulates the results of the voting.

First, you'll see the ballot screen. This represents the voting that takes place. Next, you will be given a "News Watch." This will include any endorsements you receive and the major headline of the week. After this, the results of your fund raiser will be displayed. The figure shown is your net gain (or loss) after deducting the $50,000 cost of the fund raiser. Finally, the weekly report for the new week will appear. If you have won or lost at this point, the final spending screen will be displayed.

Strategy.

While winning delegates is your main objective, you cannot hope to win without improving your standing in the national poll. This suggests that you should not neglect those regions with small numbers of delegates during a particular week, since the poll is independent of the primaries themselves.

You can spend a lot of money on mass media advertising in each region. Doing so will increase your poll percentage more quickly than any other resource can. The problem is the expense, which will rapidly deplete your fund and make it difficult to last the full nine weeks. If you use this strategy, you must try to win *fast*.

Local, "grass roots" campaigning is represented by the canvas/mass mailing resource. While it is expensive when used in all six regions ($1.2 million per week), it is a very effective way to maintain and gradually increase your national poll standing. If you have to make a choice between this and another resource, lean toward the grass roots campaign.

A debate can be very beneficial to your campaign when your poll percentage is low, but your chances of getting the debate are less. When your poll standing is high, you can easily obtain a debate. . . but it could hurt your campaign. Seek a debate when you have moved up in the polls for two or three weeks, but not when your percentage is much higher than the Senator's.

Remember, you can try to get a debate more than once each week, but it will cost you $100,000 each time you try. Think of this as you're doubling or tripling your efforts to get a debate.

That's it.

Win or lose, when the simulation is through, you'll be shown how your funds were spent. If you lost, examine your spending pattern and make adjustments for the next "big race." Good luck! □

BASIC listing.

```
100 DIM S$(2),L$(37),PERSAP$(19),R$(1)
,I$(38),A$(57),C$(57),RG$(16)
105 DIM ST(50),CD(6),MD(6),MA(6),PA(6)
,I(3)
110 DIM M5(6):L$="**********************
**********":DIM RN(6)
111 N0=0:N1=1:N2=2:N4=4:N6=6:N500=500:
N520=520:N550=550:N600=600:N2000=2000:
N7000=7000
112 N300=300:N10165=10165:N3=3:N10=10:
N9300=9300
113 N5=5:N3100=3100:N100=100:N7100=710
0
114 N7=7:N8=8
121 PERSAP$="PERSONAL APPEARANCE"
131 FOR X=N1 TO N6:CD(X)=N0:MD(X)=N0:M
A(X)=N0:PA(X)=N0
135 M5(X)=N0:NEXT X
137 TC=36:DD=17
140 WM=N1:GOTO 10000
150 RESTORE 475:GOTO N300
```

```
151 RESTORE 476:GOSUB N300:POKE 752,N1
:RETURN
152 RESTORE 477:GOTO N300
153 SOUND N0,N100,N10,N10:FOR X=N1 TO
20:NEXT X:SOUND N0,N0,N0,N0:RETURN
300 DATA 5,243,5,243,5,217,5,193,5,243
,5,193,8,217,2,0,5,243,5,243,5,217,5,1
93,10,243,10,255
301 DATA 5,243,5,243,5,217,5,193,5,182
,5,193,5,217,5,243,5,255,5,162,5,182,5
,217,10,243,10,243,255
302 READ D:IF D=255 THEN RESTORE N300:
RETURN
304 READ Y:SOUND N0,Y,N10,N10:FOR P=N0
 TO D*DD:NEXT P:SOUND N0,N0,N0,N0:GOTO
 N300
411 DATA 4,243,4,243,4,217,4,193,4,243
,4,193,6,217,2,0,255
412 DATA 12,193,6,204,6,193,2,184,2,0,
6,184,4,193,12,184,255
413 DATA 4,121,4,121,4,121,4,162,4,144
,4,144,8,162,4,96,4,96,4,108,4,108,8,1
21,255
414 DATA 6,243,6,217,8,193,8,193,8,243
,6,217,6,193,6,182,6,193,6,182,6,144,8
,162,255
415 DATA 4,121,4,144,6,182,6,182,3,182
,3,162,3,144,3,136,6,121,6,121,6,121,8
,144,255
416 DATA 6,243,4,193,12,162,12,162,6,1
82,6,193,16,217,255
450 DATA 6,121,2,121,6,108,2,108,4,96,
2,81,2,96,4,121,2,0,2,81,6,121,2,121,6
,108,2,108,8,96,8,121
451 DATA 2,0,6,121,2,121,6,108,2,108,4
,96,2,81,2,96,4,121,24,0,255
452 DATA 2,72,8,0,4,108,2,91,8,96,8,12
1,255
475 DATA 8,121,8,121,4,136,4,144,8,144
,4,153,4,144,12,144,255
476 DATA 4,204,4,204,4,204,12,243,255
477 DATA 4,121,4,162,4,193,4,243,255
478 DATA 4,243,4,193,4,162,4,121,255
500 FOR WAIT=N1 TO N600:NEXT WAIT:RETU
RN
520 ? #N6;"funds: $";CF:RETURN
550 FOR X=N1 TO 8
553 READ S$,DELG,WEEK
555 IF WEEK=WM THEN RN(C)=N1
557 NEXT X
558 RETURN
560 TW=INT(RND(N0)*N100+N1):RETURN
600 POSITION N2,20:? L$:? "1-NO.ATLAN.
 4-MAS/DIX  7-RESOURCES"
605 ? "2-INDUS BLT 5-SUNBELT  8-WEEK R
EPORT"
610 ? "3-FARM BELT 6-CASCADES 9-END OR
DERS";
625 TRAP N600:POKE 764,255:OPEN #1,4,0
,"K:"
626 IF PEEK(764)=255 THEN 626
627 GET #1,IRR:R=IRR-48:CLOSE #N1:IF R
<N7 THEN GOSUB 700+R
628 IF R=N7 THEN RESTORE 478:GOSUB N30
0
630 ON R GOTO 1010,1020,1030,1040,1050
,1060,N7000,8000,9000
701 RG$="1-north atlantic":RETURN
702 RG$="2-industry belt":RETURN
703 RG$="3-farm belt":RETURN
704 RG$="4-mason dixon":RETURN
705 RG$="5-sunbelt":RETURN
706 RG$="6-cascades":RETURN
1010 B=N6:C=N0:SC=N0:SC1=15:GOTO N2000
1020 B=N4:C=N7:SC=N0:SC1=N2:GOTO N2000
1030 B=N8:C=12:SC=14:SC1=N2:GOTO N2000
1040 B=N8:C=21:SC=N2:SC1=N2:GOTO N2000
1050 B=9:C=30:SC=N2:SC1=15:GOTO N2000
1060 B=N10:C=40:SC=N8:SC1=N2:GOTO N200
0
2000 GRAPHICS N1:SETCOLOR N4,SC,SC1:SE
TCOLOR N2,SC,SC1:POSITION N3,N0:? #N6;
RG$
2001 RESTORE 410+R:GOSUB N300
2002 ? #N6;"WEEK NUMBER: ";WM
2005 RESTORE 3000+(R*N100)
2007 V=N4
2008 ? #N6;"********************"
2009 ? #N6;"state delg week rslt"
2010 FOR X=N0 TO B:READ S$,DELG,WEEK
2013 POSITION N1,V:? #N6;S$:POSITION N
7,V:? #N6;DELG:POSITION 13,V:? #N6;WEE
K
2014 IF ST(X+C)=N1 THEN POSITION 16,V:
? #N6;"WON"
2015 IF ST(X+C)=N2 THEN POSITION 16,V:
? #N6;"LOST"
2017 V=V+N1
2018 NEXT X
2023 ? #N6;" "
2025 ? #N6;"     coords:";CD(R):? #N6;
"     media:$";MD(R)
2032 ? #N6;"canvas/mail:$";MA(R)
2035 IF PA(R)=N1 THEN ? #N6;PERSAP$
2045 GOTO N600
2900 REM
2901 GRAPHICS N0:POKE 752,N1:SETCOLOR
N2,N0,N0:P=N0
2910 X=INT(20*RND(N1)+N5):Y=INT(12*RND
(N1)+N5)
2920 POSITION X,Y:? "┌─────────┐":POSIT
ION X,Y+N1:? "┤ BALLOT ├":POSITION X,Y
+N2:? "└─────────┘"
2930 POSITION X,Y+N3:? "┌─────────┐":PO
SITION X,Y+N4:? "┤SENATOR ├":POSITION
X,Y+N5:? "┤PLAYER  ├"
2940 POSITION X,Y+N6:? "└─────────┘":P=
P+N1:IF P=20 THEN RETURN
2944 IF X<N10 OR X>20 THEN POSITION X+
N8,Y+N4:? "X":GOTO 2910
2946 POSITION X+N8,Y+N5:? "X":GOTO 291
0
3100 DATA NH,13,1,RI,12,2,ME,12,5,VT,1
1,6,MA,42,8,NY,129,8,CT,24,9
3200 DATA IN,39,2,PA,87,5,NJ,51,8,OH,7
8,9,MI,63,9
3300 DATA ND,12,2,MO,36,2,WI,36,2,SD,1
2,5,IO,27,6,IL,78,7,KS,21,7,MN,30,8,NB
,15,9
3400 DATA NC,39,4,KY,27,4,DE,10,5,VA,3
6,6,WV,21,6,DC,11,6
3401 DATA TN,33,7,MD,30,7,SC,24,9
3500 DATA FL,42,2,AZ,15,3,GA,36,4,LA,3
0,4,MS,21,4,TX,75,5,OK,24,5,AL,30,6,AR
,18,8,NM,12,8
3600 DATA CA,120,3,WA,27,3,OR,18,3,HW,
12,4,WY,10,5,MT,12,6,UT,12,8
3601 DATA NV,11,8,CO,18,9,ID,12,9,AK,1
0,9
5000 TN=N0:TL=N0:RESTORE N3100:FOR X=N
0 TO 50:READ S$,DELG,WEEK
5002 IF ST(X)=N1 THEN TN=TN+DELG
5004 IF ST(X)=N2 THEN TL=TL+DELG
5006 NEXT X
5009 IF TN>812 THEN 8800
5010 IF TL>812 OR YD<N5 OR CF<N1 THEN
8900
5040 GOTO 8003
7000 GRAPHICS N1:SETCOLOR N4,14,N2:SET
COLOR N2,14,N2:? #N6;"  resource optio
ns  "
7002 GOSUB 152
7005 ? #N6;"   WEEK NUMBER: ";WM:? #N6
;" "
7025 GOSUB N520
7028 ? #N6;" "
7030 ? #N6;" allot coordinators  "
7031 ? #N6;" buy advertising":? #N6;"
"
7032 ? #N6;" canvas/mass mailing "
7033 ? #N6;" debate":? #N6;" "
7035 ? #N6;" election appearance "
7036 ? #N6;" fund raiser":? #N6;" "
7037 ? #N6;" goto weekly report"
7040 ? :? :? "              YOUR CHOICE?";
7042 TRAP N7000:POKE 764,255:OPEN #N1,
N4,N0,"K:"
7044 IF PEEK(764)=255 THEN 7044
7046 GET #N1,IRR:R=IRR-64:CLOSE #N1:IF
 R<N1 OR R>N7 THEN 7042
7047 IF R=N7 THEN GOSUB 150:GOTO 7050
7048 GRAPHICS 17:RESTORE 478:GOSUB N30
0
```

```
7050 ON R GOTO 7200,N7100,7600,7300,78
00,7500,8000
7100 GRAPHICS N1:SETCOLOR N4,N6,N0:SET
COLOR N2,N6,N0:? #N6;"current media st
atus"
7105 GOSUB N520:? #N6;" ":? #N6;"  min
imum: $100000    "
7106 ? #N6;"region       BUDGET"
7107 FOR X=N1 TO N6
7108 IF MD(X)>N0 THEN GOSUB 700+X:? #N
6;RG$:? #N6;"            $";MD(X)
7109 NEXT X
7110 ? :? "   ENTER REGION NUMBER (1-6
)";
7113 INPUT X
7114 IF X<N1 OR X>N6 THEN GOTO N7000
7115 ? "     MEDIA BUDGET FOR REGION";
:INPUT R
7117 IF R>9000000 THEN ? "OVER $9 MILL
ION":GOSUB N500:GOTO N7100
7118 IF R<100000 AND R>N0 THEN GOTO N7
100
7120 MD(X)=MD(X)+R:CF=CF-R:FD=FD+R
7125 GOTO N7100
7200 GRAPHICS N1:SETCOLOR N4,N8,N2:SET
COLOR N2,N8,N2:? #N6;" coordinator sta
tus "
7201 GOSUB N520:? #N6;" ":? #N6;"A COO
RD COSTS $7500 "
7205 ? #N6;"region       COORDS"
7210 FOR X=N1 TO N6:GOSUB 700+X:POSITI
ON N0,N5+X:? #N6;RG$
7212 IF CD(X)<N10 THEN POSITION 16,N5+
X:? #N6;"   "
7214 POSITION 17,N5+X:? #N6;CD(X):NEXT
 X
7215 POSITION 17,13:? #N6;"  "
7216 POSITION N0,13:? #N6;"TOT COORDS
AVAIL:";TC
7217 POSITION N10,17:? #N6;"        "
7218 POSITION N0,16:? #N6;"CURRENT COO
RD COST          ";:? #N6;"$";CC:? #
N6;" "
7225 ? :? "ENTER REGION TO BE CHANGED
(1-6)";:INPUT X
7227 ? "       HOW MANY COORDS IN REGI
ON";:INPUT R:? CHR$(125)
7230 CD(X)=R:TP=N0:FOR X=N1 TO N6:TP=T
P+CD(X):NEXT X
7231 TC=36-TP:IF TC<N0 THEN FOR X=N1 T
O N6:CD(X)=N0:NEXT X:TC=36:CC=N0:GOTO
7210
7232 CC=TP*7500
7235 GOTO 7210
7300 GRAPHICS N1:SETCOLOR N4,N4,N2:SET
COLOR N2,N4,N2:? #N6;"   debate status
    "
7303 ? #N6;" ":GOSUB N520
7305 IF DB>N0 THEN POSITION N1,N8:? #N
6;"  debate scheduled":GOSUB N500:GOTO
 N7000
7310 POSITION N3,N6:? #N6;"SEEKING DEB
ATE":POSITION N3,N7:? #N6;"COSTS $1000
00"
7315 POSITION N3,9:? #N6;"GETTING DEBA
TE":POSITION N3,N10:? #N6;"COSTS $3500
00"
7320 ? :? "    ENTER YES TO SEEK DEBAT
E";
7325 INPUT R$
7330 IF R$<>"Y" THEN GOTO N7000
7335 GOSUB 560:IF YD-N10>TW THEN DB=N1
:CF=CF-100000:GOTO N7000
7340 CF=CF-100000:FB=FB+100000:GOTO N7
000
7500 GRAPHICS N1:SETCOLOR N4,12,N2:SET
COLOR N2,12,N2:? #N6;"fund raising sta
tus ":? #N6;" "
7505 GOSUB N520:? #N6;" "
7510 IF Q=N1 THEN GOTO 7550
7515 POSITION N1,N8:? #N6;"A FUNDRAISE
R WILL":POSITION N3,9:? #N6;"COST: $50
000      "
7516 ? :? "    ENTER YES TO RAISE FUND
S";
7520 INPUT R$:IF R$<>"Y" THEN GOTO N70
00
7525 IF R$="Y" THEN Q=N1:CF=CF-50000:G
OTO N7000
7550 POSITION N2,N8:? #N6;"already sch
eduled"
7560 GOSUB N500:GOTO N7000
7600 GRAPHICS N1:SETCOLOR N4,N2,N2:SET
COLOR N2,N2,N2:? #N6;"canvassing/mass
mail"
7605 GOSUB N520:P=N0:? #N6;" "
7606 ? #N6;" CANVAS/MAIL COSTS   $2000
00 PER REGION  "
7610 FOR X=N1 TO N6:P=P+MS(X):NEXT X:I
F P=N6 THEN 7690
7615 ? #N6;"region       AMOUNT"
7619 ? #N6;" ":FOR X=N1 TO N6
7620 IF MA(X)>N0 THEN GOSUB 700+X:? #N
6;RG$:? #N6;"            $";MA(X)
7621 NEXT X
7636 ? "   ENTER REGION NUMBER (1-6)";
:INPUT X
7646 IF X<N1 OR X>N6 THEN GOTO N7000
7650 MA(X)=200000:MS(X)=N1
7665 CF=CF-MA(X):FM=FM+MA(X):GOTO 7600
7690 ? #N6;"all regions covered":GOSUB
 N500:GOTO N7000
7800 GRAPHICS N1:SETCOLOR N4,N6,N2:SET
COLOR N2,N6,N2:? #N6;"personal appeara
nce ":? #N6;" "
7805 GOSUB N520:P=N0:? #N6;" "
7810 FOR X=N1 TO N6:P=P+PA(X):NEXT X:I
F P>=N1 THEN 7855
7815 POSITION N0,N8:? #N6;" ";PERSAP$;
"  WILL COST $200000"
7817 ? "    ENTER REGION NUMBER (1-6)"
;:INPUT X
7826 IF X<N1 OR X>N6 THEN GOTO N7000
7830 PA(X)=N1:CF=CF-200000:FP=FP+20000
0:GOTO N7000
7855 POSITION N0,N8:? #N6;PERSAP$;"  A
LREADY SCHEDULED ":GOSUB N500:GOTO N70
00
8000 GRAPHICS N1:SETCOLOR N2,N0,N0:GOT
O 5000
8003 ? #N6;"  weekly report #";WM
8004 ? #N6;" ":GOSUB N520
8005 ? #N6;" ":RESTORE N3100:? #N6;"pr
imaries this week"
8006 FOR X=N0 TO 50:READ S$,DELG,WEEK
8007 IF WEEK=WM THEN ? #N6;S$;" ";
8008 NEXT X
8010 POSITION N5,N8:? #N6;"delg won: "
;TN:? #N6;"    delg lost: ";TL
8012 ? #N6;"needed to win: 813  "
8025 ? #N6;"      polls:"
8030 ? #N6;"  PLAYER    ";PS
8035 ? #N6;"  SENATOR   ";SM
8040 ? #N6;"  UNDECIDED ";UNDC:? #N6;"
 "
8050 IF DB=N1 THEN ? #N6;"debate this
week"
8060 GOTO N600
8100 GRAPHICS 18:? #N6;"   final spend
ing"
8103 ? #N6;"unspent   $";CF
8105 ? #N6;"raised    $";FF
8110 POSITION N0,N6:? #N6;"coords    $"
;FC
8120 ? #N6;"media     $";FD
8130 ? #N6;"canv mail$";FM
8140 ? #N6;"pers app $";FP
8145 ? #N6;"debate    $ ";FB
8150 ? #N6;"total delg won: ";TN
8162 C=-9000:GOTO 11550
8800 A$(20,57)="hurrah!! you have a fi
rst ballot win!":GOTO 8100
8900 A$(20,57)="the senator has enough
 votes to win!":GOTO 8100
9000 GRAPHICS N2:SETCOLOR N2,N4,N2:SET
COLOR N4,N4,N2:? #N6;"     !important!
"
9015 POSITION N6,N4:? #N6;"are you":?
#N6;" ready for primary"
9020 ? "     ENTER YES TO GO TO THE
          PRIMARIES THIS WEEK";:GO
SUB 151
9025 INPUT R$
9030 TRAP 40000
```

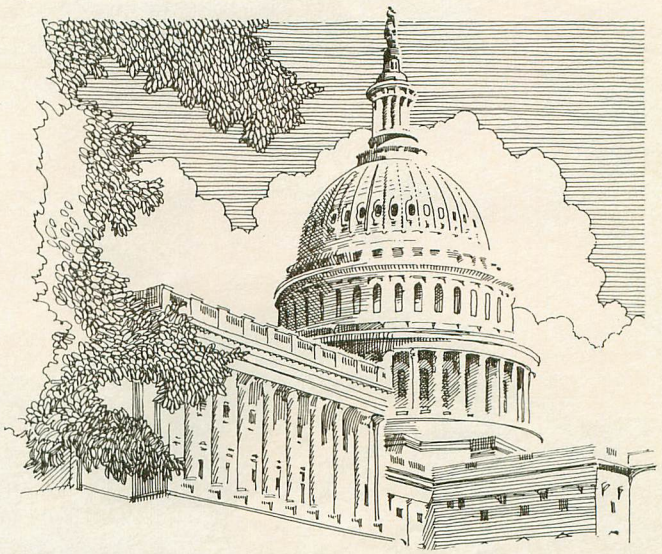
```
9035 IF R$<>"Y" THEN 8000
9040 IF Q=N0 THEN ADDFUND=50000
9045 IF Q=N1 THEN ADDFUND=INT(YD*RND(N
1)+N1)*10000
9046 GOSUB 11000:CF=CF+ADDFUND-50000:F
F=FF+ADDFUND-50000:ADDFUND=N0
9100 FOR X=N1 TO N6
9105 IF MD(X)<N1 THEN YD=YD-N2:GOTO 91
15
9107 IF MD(X)<200000 THEN 9115
9110 P=(MD(X)/500000):YD=YD+INT(P/N2)+
N1
9115 NEXT X
9140 IF DB=N2 THEN CF=CF-200000:FB=FB+
200000:YD=YD+N5:DB=N3
9300 REM
9305 P=N0:FOR X=N1 TO N6:P=P+PA(X):NEX
T X
9310 IF P=N1 THEN YD=YD+N2
9550 RESTORE N3100:B=N7:C=N1:GOSUB N55
0
9555 RESTORE N3100+N100:B=N4:C=N2:GOSU
B N550
9560 RESTORE 3300:B=9:C=N3:GOSUB N550
9565 RESTORE N3100+N300:B=N10:C=N4:GOS
UB N550
9570 RESTORE 3500:B=N10:C=N5:GOSUB N55
0
9575 RESTORE N3100+N500:B=11:C=N6:GOSU
B N550
9580 CF=CF-CC:FC=FC+CC
9600 FOR X=N1 TO N6
9605 IF RN(X)=N1 AND CD(X)>=N6 THEN YD
=YD+N1
9607 IF RN(X)=N1 AND CD(X)>9 THEN YD=Y
D+N2
9610 IF RN(X)=N1 AND CD(X)<N1 THEN YD=
YD-N1
9615 IF RN(X)=N1 AND M5(X)=N1 THEN YD=
YD+N1
9620 IF RN(X)=N0 AND M5(X)=N1 THEN YD=
YD+N1
9625 IF RN(X)=N1 AND M5(X)=N0 THEN YD=
YD-N2
9630 IF RN(X)=N0 AND M5(X)=N0 THEN YD=
YD-N1
9665 NEXT X
9775 RESTORE N3100:FOR X=N0 TO 50
9780 GOSUB 560
9785 READ S$,DELG,WEEK
9788 IF YD>75 THEN YD=75
9790 IF WEEK=WM AND YD>=TW THEN ST(X)=
N1:GOTO 9800
9795 IF WEEK=WM THEN ST(X)=N2
9800 NEXT X
9810 IF ST(N0)=N1 AND WM=N1 THEN YD=YD
+N5
9820 WM=WM+N1
9825 FOR X=N1 TO N6
9830 MD(X)=N0:MA(X)=N0:PA(X)=N0:M5(X)=
N0
9835 NEXT X
9850 SM=INT(N5*RND(N1)-N2)+40:PS=INT(Y
D/N2)+N10:UNDC=N100-PS-SM
9855 GOTO 8000
10000 GRAPHICS 18:SETCOLOR N4,N0,N10
10020 FOR Y=N0 TO N10 STEP N1:FOR X=N0
 TO 19 STEP N2:POSITION X,Y:? #N6;"[";
:NEXT X:FOR X=19 TO N0 STEP -N2
10025 POSITION X,Y:? #N6;"]":NEXT X:NE
XT Y
10030 POSITION N0,N5:? #N6;"      dark
horse      ":POSITION N8,N10:? #N6;"V1.
0":GOSUB N300
10060 GOTO 12000
10185 YD=INT(N6*RND(N1)+14)
10187 PS=YD
10200 FC=N0:FD=N0:FM=N0:FP=N0:FF=N0:FB
=N0
10700 SM=40
10702 UNDC=N100-YD-SM
10705 GOSUB 150:GOTO 8000
11000 GRAPHICS 18:SETCOLOR N2,N0,N2:GO
SUB 2900:RESTORE N300:GOSUB N300
11027 Q=N0
11029 A$(N1,19)="                    "
11030 GRAPHICS 18
11031 ? #N6;"    *news watch*":GOSUB 5
60:IF YD<TW THEN 11400
11032 CHA=CHA+N1:IF CHA>N5 THEN 11400
11035 RESTORE 11200+CHA
11040 C=N0:YD=YD+N2
11048 A$(20,33)="ENDORSEMENT:  "
11050 READ I$
11055 A$(34,57)=I$
11060 POSITION N1,N4:? #N6;A$(N1,19):C
$=A$(N2):C$(LEN(C$)+N1)=A$:A$=C$
11061 GOSUB 153
11062 C=C+N1
11070 IF C=57 THEN A$=" ":GOTO 11400
11075 GOTO 11060
11201 DATA SAVE THE MOON SOCIETY ,N0
11202 DATA CARWASH OWNERS ASSOC  ,N0
11203 DATA PRISONERS RIGHTS COMM ,N0
11204 DATA HIGHTECH WORKERS UNION,N0
11205 DATA VIGILANTE  BROTHERHOOD,N0
11400 RESTORE 11600
11500 FOR X=N0 TO EVENT:READ I$:NEXT X
11510 EVENT=EVENT+N1
11520 A$(20,57)=I$
11530 C=N0
11550 POSITION N1,N4:? #N6;A$(N1,19):C
$=A$(N2):C$(LEN(C$)+N1)=A$:A$=C$
11558 GOSUB 153
11560 C=C+N1
11565 IF C=57 THEN 11580
11568 IF C=-N1 THEN END
11570 GOTO 11550
11580 IF DB<>N1 THEN 11590
11585 DB=N2:A$(20,57)="primary debate
was held last week    ":GOTO 11530
11590 POSITION N1,N3:? #N6;"fundraiser
 receipts":RESTORE 450:GOSUB N300:POSI
TION N6,N6:? #N6;"$";ADDFUND-50000
11595 RESTORE 452:GOSUB N300:A$=" ":RE
TURN
11600 DATA SOVIETS DENY THEIR MOONBASE
 IS ARMED,CAMPAIGN AIDE IS CITED IN BR
IBE SCAM
11601 DATA FOOD RIOTS ROCK GREAT BRITA
IN
11602 DATA TITAN SPACE PROBE FINDS LIF
E FORMS,EIGHTY INJURED IN NEW YORK WAT
ER RIOT
11603 DATA SINO-FRENCH FOOD TREATY SIG
NED TODAY,JURY GIVES YOUTH DEATH IN N.
H. TRIAL
11604 DATA JAPAN ADMITS TO LUNAR MISSI
LE BASE
11605 DATA NUKE WEAPONS BANNED FROM MA
RS BASE
12000 GRAPHICS 18:SETCOLOR N4,N0,N0
12005 FOR X=N0 TO 50:ST(X)=N0:NEXT X
12010 POSITION N2,N0:? #N6;"COPYRIGHT
1983 BY"
12015 POSITION N3,N1:? #N6;"ken amidon
  AND":POSITION N3,N2:? #N6;"wayne und
erwood"
12025 POSITION N0,N10:? #N6;"PRESS STA
RT TO BEGIN"
12030 POSITION N2,N5:? #N6;"PRESS SELE
CT FOR":POSITION N5,N6:? #N6;"SKILL LE
VEL"
12035 POSITION N5,N7:? #N6;" beginner"
:OP=N1:CF=20000000
12037 FOR WAIT=N1 TO 50:NEXT WAIT
12040 SEL=PEEK(53279):IF SEL=N7 THEN 1
2040
12042 IF SEL=N6 THEN GRAPHICS 18:GOTO
10185
12045 IF SEL<N5 THEN GOTO 12037
12047 IF SEL=N5 AND OP=N0 THEN POSITIO
N N5,N7:? #N6;" beginner":OP=N1:CF=200
00000:GOTO 12037
12050 IF SEL=N5 AND OP=N1 THEN POSITIO
N N5,N7:? #N6;"intermediate":OP=N2:CF=
17000000:GOTO 12037
12055 IF SEL=N5 AND OP=N2 THEN POSITIO
N N5,N7:? #N6;"  expert    ":OP=N0:CF=
14000000:GOTO 12037
```

•

CHECKSUM DATA.

(see page 25)

```
100 DATA 267,97,603,753,594,484,957,66
6,252,991,40,639,669,350,673,8035
153 DATA 658,415,13,971,401,313,378,64
7,819,698,53,91,924,832,634,7847
476 DATA 258,20,21,459,634,231,889,962
,788,612,586,407,735,609,367,7578
626 DATA 553,724,156,283,15,460,663,61
7,840,341,813,917,803,855,682,8722
1060 DATA 37,654,727,585,161,302,982,6
94,649,60,462,340,466,548,442,7109
2025 DATA 964,479,239,805,304,482,851,
76,277,257,948,18,530,913,141,7284
3400 DATA 622,934,985,634,694,141,37,3
9,549,954,335,738,967,749,838,9216
7025 DATA 989,462,821,993,350,856,401,
72,413,22,708,881,889,986,109,8952
7050 DATA 401,479,93,766,637,717,562,3
03,720,645,29,961,848,494,22,7677
7200 DATA 489,204,585,467,963,466,538,
230,225,360,20,498,180,66,924,6215
7235 DATA 749,873,150,465,61,98,287,95
2,30,828,841,476,172,492,488,6962
7516 DATA 308,181,678,889,291,543,612,
461,953,820,477,706,563,489,667,8638
7650 DATA 94,600,172,741,618,264,906,1
74,671,975,10,503,925,145,810,7608
8006 DATA 971,326,559,521,974,155,136,
444,835,340,809,201,590,117,71,7049
8120 DATA 872,723,604,276,628,726,9,88
9,148,993,636,947,890,242,184,8767
9045 DATA 304,584,627,244,487,979,559,
10,300,333,423,96,655,19,721,6341
9570 DATA 974,630,14,642,542,277,452,5
22,511,528,515,579,334,778,204,7502
9788 DATA 82,331,885,569,265,959,660,9
73,582,133,772,904,299,795,439,8648
10060 DATA 193,480,607,299,430,883,461
,508,574,472,309,663,737,892,643,8151
11048 DATA 807,853,169,463,60,646,275,
217,884,791,43,976,998,662,152,7996
11510 DATA 880,139,532,466,84,645,966,
167,221,278,565,933,689,353,783,7701
11602 DATA 69,252,244,179,704,797,105,
791,553,15,453,528,280,898,263,6131
12047 DATA 119,856,971,1946
```



Put a $20,000,000 Thrill in your Computer!!!

The F-15 STRIKE EAGLE, one of the world's most sophisticated fighters, costs the Air Force more than $20,000,000.00 each. Now you too can strap into your ejection seat and prove how good you really are in exciting modern jet fighter combat. Fly combat missions, engage enemy aircraft, and destroy enemy ground targets from historic missions over Southeast Asia to today's defense of the strategic oil routes through the Straits of Hormuz.

F-15 STRIKE EAGLE (simulator) has all the features that makes the real EAGLE the great fighter aircraft it is — afterburners, multiple radars, air to air missiles, high explosive bombs, cannon, drop tanks, Electronic Counter Measures (ECM) flares, electronic ground tracking maps, Heads Up Display (HUD), outstanding 3-dimensional cockpit visibility, and realistic F-15 maneuverability. Your mission is to take off from your base, fly to and destroy your primary target through all the dangers of enemy territory including Surface to Air Missiles and enemy aircraft. Of course, you have to get back home again, too!!

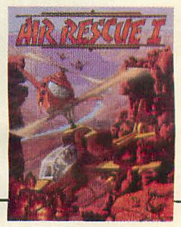
F-15 STRIKE EAGLE is a very real simulation, accomplished with the guidance of real fighter pilots, and includes seven different combat missions, four skill levels, and an infinite number of exciting scenarios. F-15 will thrill and challenge you and give you the chance to prove you have the "Right Stuff" of an EAGLE fighter pilot!!

F-15 STRIKE EAGLE is available for Commodore 64, ATARI (48K), and Apple II (64K) computers. Suggested retail price is only $34.95. Find STRIKE EAGLE at your local retailer, or call or write for MC/VISA, or COD orders. Add $2.50 for Postage and Handling (Int'l add $4.00USD). MD residents add 5% sales tax.

Experience the reality of these other great simulations from MicroProse

As close to the Real Thrill of Flying as You Can Handle!!!

Heart Pounding Accelerated Real-Time Defense of Europe Against Soviet Invasion!!!



Sensational Assault Chopper Rescue Raids for Daring Pilots!!!

MicroProse Software

The Action is Simulated — the Excitement is REAL!!

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

CIRCLE #131 ON READER SERVICE CARD

TOP-DOS
ECLIPSE SOFTWARE
1058 Marigold Court
Sunnyvale, CA 94086
(408) 246-8325
32K Disk $49.95

by Charles Bachand

I am usually not a fan of menu-driven DOSs. They always seemed too slow and restrictive. It was also difficult to customize them to your own tastes and preferences. And, since they almost always came as two or more files on a disk, whenever you needed to perform something as trivial as getting a disk directory, you were forced to sit and wait while a DUP.SYS file was loaded into memory. You were also forced to associate a letter with a DOS function that sometimes didn't even contain the letter! For example, does the letter A have anything to do with getting a disk directory? Or, why should typing a *B* run the cartridge? I won't even go into locking and unlocking files; it's just too strange. Because of all this, one of my two favorite Disk Operating Systems happens to be **DOS-XL** by OSS, a command-driven DOS. But this review is not about that.

It *is* about **TOP-DOS** by Eclipse Software, a menu-driven DOS. By now you might be thinking, "Oh boy, he's going to tear into this DOS and give it a really bad review!" Well, I hate to disappoint you, but **TOP-DOS** is *great*! It happens to be that other DOS that I like so very, very much.

How well DOS it work?

TOP-DOS uses the same menu structure as Atari's **DOS II**, and if you don't use any of the extra commands or features that are incorporated in it, **TOP-DOS** will *act* just like **DOS II**. This is far from a drawback. In fact, it is just one of the program's major attributes: compatibility. If it will work with **DOS II**, it will work with **TOP-DOS**.

Another attribute of **TOP-DOS** is its flexibility. A good example to demonstrate this is the procedure used to list a disk directory. I've picked this as an example because it usually generates the most screen output. Flexibility in **TOP-DOS** is the rule and not the exception.

Directory options in **DOS II** are limited to a source and destination combination. The fanciest you could get here is to perform something like listing onto the printer all of the filenames from D2: that started with the letters *ANALOG*. With **TOP-DOS**, it's a whole new ball game! You can specify things like the number of columns (up to six) the filenames will produce across a page and/or list the files that have been deleted (they remain in the directory until replaced by a new filename). Also, if you are using the special **TOP-DOS** format (an option when formatting), **TOP-DOS** handles the file number byte differently—which also produces a slight incompatibility if the files are ever accessed with **DOS II**, although you might throw your **DOS II** disk out after having used **TOP-DOS** for a while. This alternate handling allows you to alphabetize the directory, so that filenames that start with *A* are at the top, and the *Z*s show up near the bottom of the listing. Compressing a directory is also now allowed. This moves the filenames to the beginning of the directory (overwriting deleted filenames), thus allowing for faster filename lookups.

The directory listing produced by **TOP-DOS** is highly informative and not limited to the likes of filenames, protection status, file lengths and number of free sectors. It will inform you if a file is deleted or was left open (not properly closed), as well as how many files and sectors are in use or available. Single, double or quad density (double-sided, double density) disks are also flagged, and the disks are identified as being formatted by **DOS II** or **TOP-DOS**.

Also, if you don't need anything fancy here, simply typing the number of the drive the disk is occupying will produce a listing onto the screen.

Help is on the way.

One of the nicest features of **TOP-DOS** is that you really don't have to read the instruction book (which is over seventy pages long and quite excellent) to use it! Eclipse has incorporated into **TOP-DOS** some of the most extensive "help" files I have ever run across. If you are wondering how **TOP-DOS** works, simply type a question mark at the prompt, and a general overview of the program will appear on the screen. If you have a question about one of the commands, typing the command's letter and another question mark will produce a detailed breakdown, including all the allowed options. It couldn't be simpler.

Error messages are another area where **TOP-DOS** is truly helpful. Imagine that you are developing a BASIC program, and the first time you run it, an error 141 flashes onto the screen. You can't be expected to remember every single error code that comes along (after all, a computer's supposed to *help* you, not force you to dash for the reference manual), but with **TOP-DOS** there is hope. One simply has to call up the DOS menu and type a capital *T*, which stands for trouble (right here in River City), followed by the error number—in this case, *141*. Your computer will print out *Cursor Out of Range*, the meaning of an error 141. How very nice, indeed!

Other new and wonderful commands include Undelete and Read/Store. Undelete is a much-needed command that allows you to literally raise a file "from the dead." Many a time I've erased a file—only to realize a moment later that I needed that file desperately! Undelete will return the file to its previous condition—truly a lifesaver. If you are ever in a bind and need to examine and/or change bytes in memory, **TOP-DOS** will do that, too—with the Read/Store com-

mand. It's almost like having an **OMNIMON** board in your computer (*almost*).

Yet another menu!

To allow you to customize **TOP-DOS**, there is yet another menu, accessible through the Set/Status command. This menu (an even more extensive one than the main menu) will allow you to: (1) change prompt character; (2) change left margin; (3) change system drive number; (4) change number of open file buffers; (5) add drive to drive list; (6) remove drive from drive list; (7) set drive to single density; (8) set drive to double density; (9) set drive to quad density; (10) modify drive control bytes; (11) display status; (12) initialize disk buffers; (13) toggle RS-232/MEM.SAV option; (14) toggle cartridge bypass option; (15) toggle resident DOS option; and, finally, (16) toggle Write/Verify option.

Most of these are pretty self-explanatory, so I will only touch upon the more unusual ones here.

(10) Modify drive control bytes — PERCOM-compatible drives (Indus, Trak, Rana, etc.) have twelve bytes that can be transferred to or from the drive in order to control things like density, head-access time, and maximum sector count, to mention a few. This option will allow you not only to examine these bytes, but to change them as necessary. While you'll probably never use this feature, it's still nice to know it's there.

(14) Toggle cartridge bypass option — Allows you to go directly to **TOP-DOS** upon power-up, even though a cartridge (like BASIC) is installed. Normally, we would come up in BASIC and have to type the word *DOS* to get to the **TOP-DOS** menu.

(15) Toggle resident DOS option — If you don't like the wait to load the DUP.SYS file and have plenty of free memory to play with, you may configure **TOP-DOS** to be resident. This means that the entire DOS is loaded into the computer's memory, instead of merely a part of it, and the DOS's menus are instantly accessible. The only drawback is that you lose an extra 10K of RAM space. The choice is yours.

Bells and whistles.

I have come to realize that R.K. Bennett (the author of **TOP-DOS**) loves his work and, above all else, writes software to please himself. This is noticeable in **TOP-DOS**'s extras—things which no software spec writer would *ever* have thought up. . . Little things, like:

(1) RAMDISK Support — **TOP-DOS** can be configured to use either an AXLON 128K or

MOSAIC 64K bank select memory board as an extra disk drive. This RAMDISK, which is set as drive 8, allows for instantaneous disk accessing to speed up your program execution. You're really accessing RAM, but, for all intents and purposes, the software will treat it as just another disk drive.

(2) Interception of BRK instruction — About the only use for machine language BRK instructions is in the development of machine language software, and they are removed after the software has been fully debugged. If a standard Atari computer tries to execute a BRK instruction, it will more than likely do something vaguely nasty—like lock up the keyboard. **TOP-DOS** will not allow this to happen and will bring you back to DOS, as well as flag the address where the BRK occurred.

(3) New COPY options — MERGE copies to the destination disk only those files from the source disk that are not already present on the destination disk. The reverse of this would be the UPDATE option, which only copies files that the source and destination disks both contain.

(4) Formatting disks — You can specify that the format command only initialize the VTOC and filename sectors on a disk. This is a fast way (three seconds!) to clear the filenames from a previously formatted disk. One may also specify the number of sectors to enable on a nonstandard disk drive, up to 944 (720 is the default).

(5) Command files — DOS files containing **TOP-DOS** commands which can be executed from DOS or at system power-up. Command files can even call other command files.

(6) Entry points — **TOP-DOS** has had some of its memory locations set aside as flags, pointers and machine language entry vectors. This allows your machine language programs easy access to some of the **TOP-DOS** routines, such as reading filenames.

The big finish.

TOP-DOS retails for $49.95, and I feel it's worth every penny. It also comes with a thirty-day money-back guarantee! You can't ask for more than that.

It seems fairly safe to say that this reviewer really *really* likes **TOP-DOS**. The only negative aspect that I could find was that, in this article, I've mentioned the name **TOP-DOS** over thirty times! Let's see, if Eclipse were paying me on that basis *(sure, Charlie—Ed.)*, it would come to... □

ALL THE BEST PRICES

ATARI HOME COMPUTERS

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

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

MasterCard

800-648-3311

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

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 #133 ON READER SERVICE CARD

DONKEY KONG JUNIOR
ATARI, INC.
Sunnyvale, CA 94086
16K Cartridge $49.95

by David Shen

While strolling through a local Toys R Us store, I chanced upon a **Donkey Kong Junior** cartridge hidden amidst some unwanted Cabbage Patch Dolls. Hurriedly whipping out my wallet, I bought the game and rushed home. I slammed the cartridge into my trusty old 800, flipped the power switch and was greeted by a nice title screen and a rambling tune.

Game setup.

For those of you who are not familiar with **Donkey Kong Junior**, here's a brief storyline. In **Donkey Kong**, Mario, the heroic carpenter, had to rescue his girl from the clutches of the gorilla, **Donkey Kong**. Naturally, Mario wanted revenge. So, in **Donkey Kong Junior**, he's captured the confused ape, and it's up to Kong's little son, **Donkey Kong Junior**, to save his chained and caged father. But no one said this task would be easy!

Junior must climb multitudes of vines and chains in the presence of various comical menaces, all sent by Mario to ruin Junior's day. He has to survive the trip through four aptly-named screens of danger and peril.

In the Vine Scene, Junior has to fight his way up vines crawling with deadly Snapjaws to reach the key that hangs next to his father's cage. His only defense against these attackers is the fruit growing on the vines, which he can drop onto them.

In the Chain Scene, Junior must insert six keys into the six locks which hold Kong's cage. But there is a new, added creature to contend with—birds traverse the screen, hoping to give Junior a bad time. Once Junior unlocks all the locks, Kong literally gives Mario the boot.

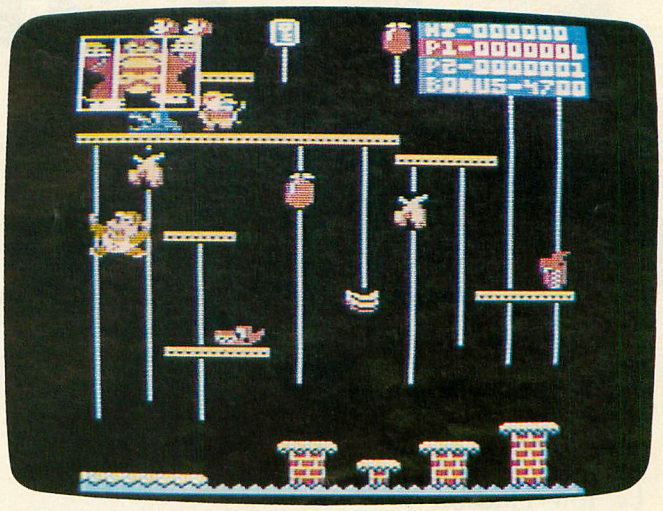
In the third screen, the Jump Board Scene, Junior has to leap onto a jump board, which carries him to a moving platform. His moves must be precise, since an accident would waste one of his precious lives. After that, Junior still has to make his way through hoards of flying Nitpickers. They drop eggs which can send Junior plummeting.

Providing that he has gotten this far, Junior comes to Mario's Hideout, where glowing Sparks and Globes protect Mario and his prize. Yes, Junior still can use fruits, however, the more he lingers, the more dangers Mario releases.

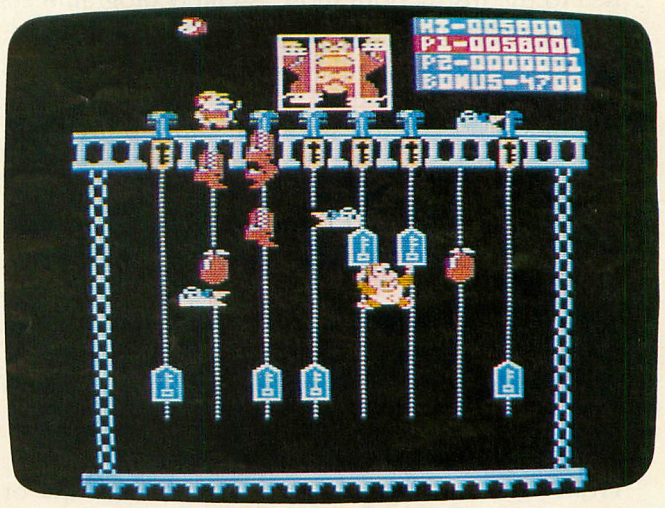
Getting to business.

Enough talk on the game setup; now into the game itself. This verion of **Donkey Kong Junior** looks almost like its arcade counterpart. Each screen is wonderfully akin to one in the quarter-snatching original. There is, I'm happy to say, little, if any, flickering.

Playwise, there is some acclimatization to be done. You may be a whiz at **Donkey Kong Junior** in the arcades, but find yourself not getting past the first board at home. I was frustrated beyond belief on my first few tries. The semi-instruction book hardly aided me on my quest. Some will absolutely hate this game for being so tough; others will, as I did, enjoy the challenge.



Donkey Kong Junior screen 1.

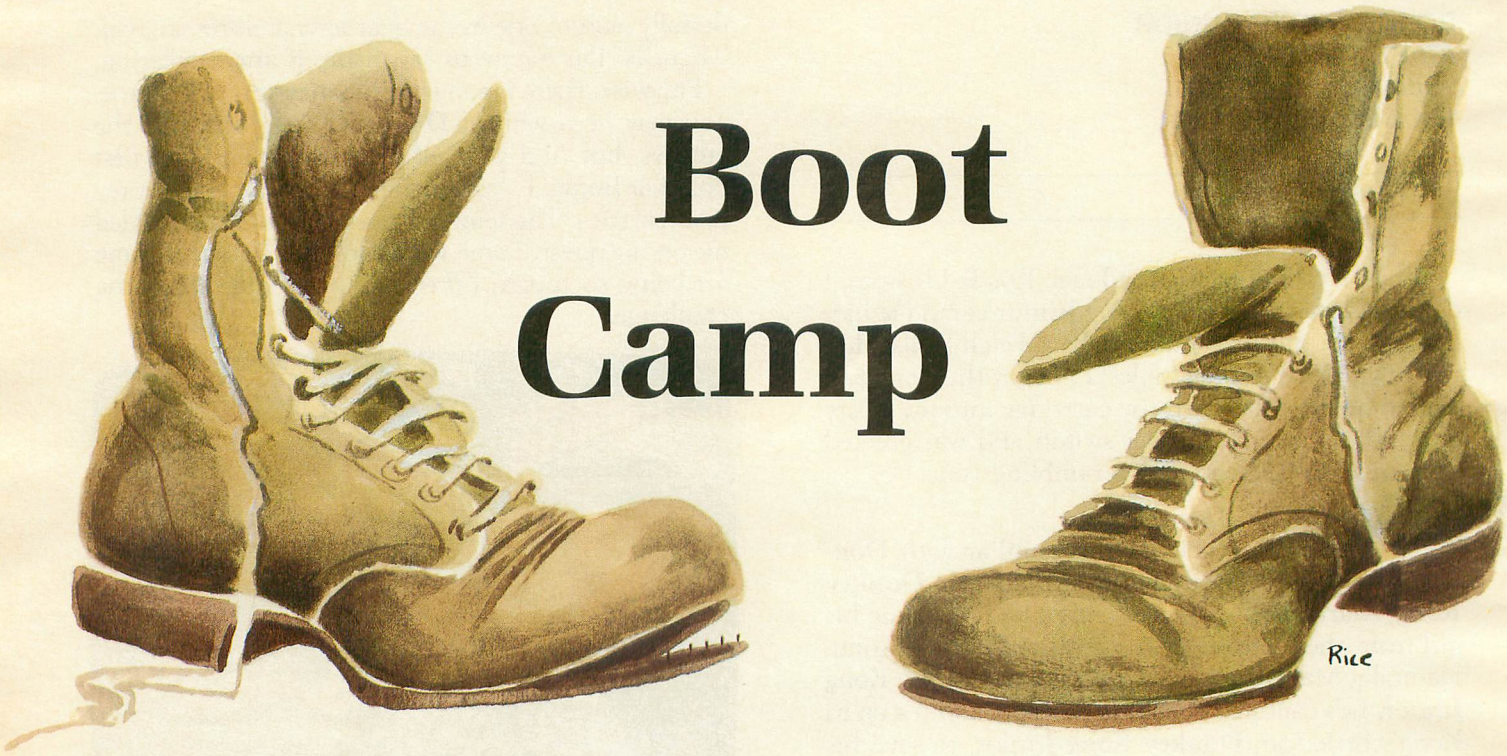


Donkey Kong Junior screen 2.

Where did they go?

Beneath all this good fun, there are some missing elements and problems. There is a slight crudeness in the controls, and both the introduction of the arcade version and the unwanted GAME OVER are missing. These problems are compensated for by the great intermissions and musical tunes. Hardcore arcade game fans—or anyone else—shouldn't be disappointed. Putting **Donkey Kong Junior** on a scale of one to ten, I'd give it a hearty eight and a half. □

Boot Camp



by Tom Hudson

As of now, all **Boot Camp** readers have been exposed to the instructions most assembly programmers consider important. Sure, there are a few we skipped, but they're primarily used for advanced applications, such as interrupt handling. We'll discuss them later.

This month's column presents some information that will prepare you for next issue's subject: BASIC USR calls.

I'm assuming that most readers of **Boot Camp** want to speed up their BASIC programs with ultra-fast machine code. If you don't, you could probably skip this issue's column, but it's best that you read it and understand it. If you gain more knowledge of assembly language now, you'll have fewer "unsolvable" problems later.

Tailor-made or off-the-rack.

From time to time, while reading **ANALOG** or other magazines dealing with assembly language, you may have heard the term *relocatable* used to describe an assembly program. What does this term mean? I find a good analogy in the clothing industry, suits in particular.

When a wealthy executive goes out to buy a suit, he probably won't go to the local self-service "Bargain Barn" to find one. No, he'll usually see a tailor in order to have one custom-made. More than likely, the suit he has made will fit him perfectly, and nobody else.

When I, on the other hand, go out to buy a suit, I like to get it over with as soon as possible, since I have more important things to do than buy something I'll use at *most* a dozen times a year. I'll go straight to the "Bargain Barn" and pick out an off-the-rack, stock suit. The suit would probably fit thousands of other people fairly well. If I'm lucky, it'll look just about as good as a tailor-made suit. Usually, though, you'll have to compromise in some area, such as "perfect" fit.

Assembly programs are sort of like suits. Some programs are written to run only in a *specific* area of memory and are known as *non-relocatable*.

Here's an example: my program **Retrofire** (issue 14) was written to reside in the area of memory starting at $0800. If you try to load it at $6000, it just won't work. At best, the screen may change color, and the system will crash. Even if you place it as little as *one* byte off, at $0801, it will crash. That's because this program is tailor-made to work only at $0800, and no amount of work (short of re-assembly) *will* make it operate elsewhere. **Retrofire** is *non-relocatable*.

On the other hand, let's say you write a short assembly routine that is going to be used by a BASIC USR call. You've placed the object code bytes in the BASIC string ML$ and are going to call it with the statement:

```
A=USR(ADR(ML$))
```

Since you don't know where in memory BASIC will put the string ML$, this routine *must* be relocatable. It must be able to operate wherever BASIC puts it.

Just like you would with an off-the-rack suit, you'll have to be willing to compromise to a certain degree, by writing your code so that it can be placed anywhere in memory.

Let's take a look at how relocatable routines are written.

The ABCs of relocatability.

When you're writing normal, non-relocatable code, you don't have to worry about anything. You simply write to your heart's content and let the computer do the rest.

Not so with relocatable code. There is one rule that must be followed without exception: *never* refer to a location or label within the relocatable code with an absolute format instruction.

What does this mean? Take a look at Figure 1.

```
0000            10              *=   $0600
0600 A901       20              LDA  #$01
0602 200606     30              JSR  SUBR01
0605 60         40              RTS
0606 8D0A06     50 SUBR01       STA  TEMP
0609 60         60              RTS
060A            70 TEMP         *=   *+1
```

Figure 1.

Let's assume we want to use the code in Figure 1 as a relocatable subroutine. We've got two problems.

First, the JSR instruction is an *absolute* addressing instruction and it is referring to the label SUBR01, which is within our routine. What does the relocation rule say? We *cannot* use an absolute addressing instruction which refers to a label within the code to be relocated. This JSR is a definite no-no.

Second, the STA TEMP instruction is also absolute and it refers to TEMP, a label within the routine. Sorry, but you can't do this, either!

Let's see what happens if this routine is relocated to $6000, instead of $0600, where it was assembled. Figure 2 shows the program image stored in memory at $6000, with the source code shown to the right.

```
 ADDR
------
(6000) A901     =  LDA  #$01
(6002) 200606   =  JSR  $0606
(6005) 60       =  RTS
(6006) 8D0A06   =  STA  $060A
(6009) 60       =  RTS
(600A)          =  *=   *+1
```

Figure 2.

First, the LDA #$01 is executed. Since this is an *immediate* format instruction, all is well so far.

Next, the JSR SUBR01 instruction executes. If you look at Figure 1, you'll see that SUBR01 is *supposed* to be at location $0606, but the program has been relocated to $6000! The code at SUBR01 is now at $6006, yet the 6502 has no alternative but to follow its instructions. It JSRs to $0606!

What happens next is anybody's guess. Location $0606 may contain BRK instructions, garbage or even Aunt Mary's recipe program. There's simply no way of telling, and the system will probably crash.

How do we avoid such a catastrophe? It takes a little work, but it can be done. Rethink your program so that it does not use absolute addressing instructions. Sometimes this is easier said than done, but if you want it relocatable, you've got to work a little harder.

The most common problem in relocating comes when you need to JMP to another part of the routine. Remember, the most common JMP instruction is (you guessed it) absolute! Here's an uncomplicated solution . . .

All of the 6502 branch instructions use *relative* addressing. This isn't absolute, so we can use all the branch instructions in our relocatable routines. The only problem is that all the branch instructions are conditional. In order to branch each time the branch instruction is executed, we'll have to make sure its branch condition is *true*. All the following combinations will replace the JMP instruction:

```
        CLC
        BCC LABEL
;
        SEC
        BCS LABEL
;
        LDA #0
        BEQ LABEL
;
        LDA #1
        BNE LABEL
;
        LDA #$FF
        BMI LABEL
```

All of these branch instructions replace the JMP instruction, but their branch range is limited to about 128 bytes. That is, if your relocatable routine is 200 bytes long, and you need to branch from the end to the beginning, one branch won't go far enough. You'll have to set up a "bucket brigade" branch. This is accomplished by branching to a second branch, which, in turn, branches to the final destination label. We'll look at this process in detail in another installment.

Where to put data?

Another common problem in relocatable routines is being uncertain about where to place data values. They can't be placed in the routine itself, because to load and store the data requires the use of absolute addressing.

If your relocatable routine is for Atari BASIC, you can use the zero page locations $CB through $D1. Page 6 ($0600-06FF) is also available for data storage. When your relocatable routine utilizes data in these areas, all is well because they never move.

Subroutines in relocatable code.

Using subroutines in relocatable code is a particularly messy problem and one for which I've never seen a good solution. For now, try to write any relocatable routines with the subroutine code in-line. This is usually acceptable for short subroutines.

Making code relocatable.

As we have seen, the code in Figure 1 is far from being relocatable. However, we can make it relocatable with a few small changes.

First, let's get rid of the subroutine. It's a short one, so there is no real problem with putting it in-line. Figure 3 shows the code modified to eliminate the subroutine.

```
        LDA #$01
        STA TEMP
        RTS
TEMP    *=*+1
```

Figure 3.

Okay, that takes care of the subroutine problem, but there's still the matter of the TEMP storage location.

No problem, we'll simply place it in a free location on page zero, as shown in Figure 4.

```
TEMP =   $CB
     LDA #$01
     STA TEMP
     RTS
```

Figure 4.

As you can see, we have merely told the assembler that TEMP is at location $00CB. This shows the use of the EQUATE directive. Your assembler may use the directive EQU instead of the "equal" sign. Check your assembler manual.

That was simple enough, right? Let's do another one.

```
START LDA BYTE1
      CLC
      ADC #1
      STA BYTE2
      JMP PART3
PART2 CMP #4
      BNE PART3
      JMP START
PART3 LSR A
      JMP PART2
BYTE1 *=*+1
BYTE2 *=*+1
```

Figure 5.

Figure 5 shows a slightly larger program that is not relocatable. It has two data items and three JMP instructions that must be altered in order to make the program relocatable.

```
PART2 CMP #4
      BNE PART3
      JMP START
PART3 LSR A
      JMP PART2
BYTE1 =   $600
BYTE2 =   $601

START LDA BYTE1
      CLC
      ADC #1
      STA BYTE2
      JMP PART3
```

Figure 6.

Let's change the data items first. They're easiest, since the only action needed is to place them in fixed memory somewhere. We'll put them on page 6, the area of memory set aside for our use. Figure 6 shows the program after we make the data item change.

Now let's tackle the JMP instructions. The first JMP jumps to PART3. If you examine the code at PART3, you'll see that it expects the accumulator to contain the result of the add in the START section. Therefore, we cannot alter the accumulator. In this case, let's replace JMP PART3 with the code:

```
CLC
BCC PART3
```

This code clears the carry flag, forcing the BCC PART3 to branch. It's simple and it works just like the JMP did.

The next JMP, the one in the PART2 section, will JMP to START. We need to replace the JMP with a branch, and this case is particularly easy.

If you look at the instruction preceding the JMP, you'll see that it's a BNE (Branch Not Equal) instruction. This means that the JMP START instruction will only execute if the accumulator is *equal* to four.

Avalanche Correction

In **ANALOG Computing** issue 21, the assembly language game **Avalanche** was given a memory requirement of 16K for cassette.

As listed, it is too large to fit in these systems. However, by removing Lines 10, 20, 150, 180 and 190, the program will execute properly in 16K.

When run, the modified program will create a cassette copy of the game, with no cassette or disk prompt.

We can take advantage of this fact when we replace the JMP. In this situation, the JMP START can be replaced with:

```
BEQ START
```

The last JMP, in the PART3 section, is right after an LSR instruction. We don't want to disturb the accumulator, so we can replace the JMP with the code:

```
CLC
BCC PART2
```

The final, relocatable code for the program is shown in Figure 7.

```
START LDA BYTE1
      CLC
      ADC #1
      STA BYTE2
      CLC
      BCC PART3
PART2 CMP #4
      BNE PART3
      BEQ START
PART3 LSR A
      CLC
      BCC PART2
BYTE1 =   $600
BYTE2 =   $601
```

Figure 7.

The important thing to remember when making a program relocatable is to avoid disturbing any registers the program is using. Don't make any assumptions about what the program is doing—*check it out.*

Review the instructions.

It's a good idea, at this point, for you to go back and review all the operation codes we've discussed so far, noting all those which use the *absolute* addressing mode. It's important that you get to know all of the assembly instructions as well as you know the BASIC commands. This will avoid wasting a lot of time looking instructions up in a book when you start programming.

Next issue, we'll talk more about relocatable code, when we start examining BASIC USR calls. Until then, *review*! □

Send all letters to:

Boot Camp
c/o ANALOG Computing
P.O. Box 23
Worcester, MA 01603

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 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

Lyco Computer Marketing & Consultants

TO ORDER CALL US

TOLL FREE 800-233-8760

In PA 1 717-327-1824

SAVE ON THESE IN-STOCK PRINTERS

MANNESMANN TALLY

SPIRIT 80	$299.00
MTL-160L	$559.00
MTL-180L	$775.00

NEC

NEC8023	$369.00
NEC8025	$699.00

PRINTER INTERFACING

AVAILABLE FOR COMMODORE VIC, APPLE, ATARI, IBM-PC, TRS-80, TI, AND OTHERS

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

CITOH

GORILLA GX-100	$169.00
GORILLA SERIAL	$199.00
PROWRITER 8510	$335.00
PROWRITER II	$575.00
8600	$899.00
STARWRITER	$999.00
PRINTMASTER	$1299.00
SHEET FEEDER	$425.00
620	$929.00
630	$1699.00
8510 SP	$499.00
8510 SCP	$559.00
A10 LETTER QUAL	$499.00

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

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

WICO COMMODORE & ATARI

JOYSTICK	$16.75
RED BALL	$18.75
BOSS JOYSTICK	$12.75
TRACK BALL	$32.75
12 FT EXT CORD	$6.95
TI ADAPTER	$9.95
APPLE ANALOG	$34.75
IBM-PC ANALOG	$34.75

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

DISKETTES

ELEPHANT

5¼" SSSD	$16.75
5¼" SSDD	$19.95
5¼" DSDD	$25.95

MAXELL

5¼" MD-1	$24.75
5¼" MD-2	$34.75

CERTRON CASSETTES

CC-10 (12)	$15.99
CC-20 (12)	$17.99

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

DUST COVERS

1200	$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
KOALA TOUCH TABLET	$69.75

MONITORS

SAKATA COLOR	$249.00
TAXAN GREEN	$119.00
TAXAN AMBER	$129.00
TAXAN RGB COMPOSITE	$289.00
ZENITH AMBER	$105.00
ZENITH GREEN	$88.00
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

ATARI

800XL	CALL
1450XL	CALL
1050 DRIVE	$299.00
1027 PRINTER	CALL
1025 PRINTER	$299.00
1020 PRINTER	$219.00
1010 RECORDER	$74.00

ATARI

PACMAN	$29.75
DONKEY KONG	$29.75
DIG DUG	$29.75
DEFENDER	$29.75
CENTIPEDE	$29.75
DEADLINE	$34.75
ENCHANTER	$34.75
INFIDEL	$34.75
PLANETFALL	$34.75
STAR CROSS	$34.75
SUSPENDED	$34.75
WITNESS	$34.75
ZORK I	$34.75
ZORK II	$34.75
ZORK III	$34.75

BUSINESS

VISICALC	$159.75
LETTER PERFECT	$89.75
LETTER PERFECT R	$89.75
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

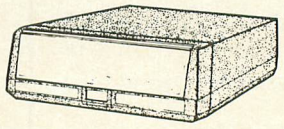
SPINNAKER

STORY MACHINE R	$26.75
FACE MAKER R	$24.75
KINDEROMP R	$20.75
FRACTION FEVER R	$24.75
DELTA DRAWING R	$26.75

PARKER BROTHERS

TUTANKHAM R	$33.75
SUPER COBRA R	$33.75
ASTRO CHASE R	$33.75
FROGGER R	$33.75
Q BERT R	$33.75
POPEYE R	$33.75
RISK R	$42.75
CHESS R	$42.75

DISK DRIVES



RANA 1000 .... $299

INDUS GT $315

CONCORD

ATARI 176K MASTER	$289.00
ATARI 348K MASTER	$369.00
ATARI ADD-ON DRIVES	$CALL

PERCOM

AT8851	$249.00
AT8851PD	$299.00
ADD-ON DRIVES	CALL
AT 88 DOUBLER	$119.00

TRACK DRIVES

AT-D2	$329.00
AT-D2 TURBO PAK	$22.95
AT-DH	CALL
PRINTER CABLE	$22.95

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

MasterCard TO ORDER VISA

CALL TOLL FREE

800-233-8760

Customer Service 1-717-327-1825

or send order to Lyco Computer P O Box 5088 Jersey Shore PA 17740

POLICY

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

CIRCLE #134 ON READER SERVICE CARD

ATR-8000
SOUTHWEST MICROCOMPUTER PRODUCTS
2500 E. Randol Mill Rd.
Arlington, TX 76011
16K $450.00 64K $599.95

by Philip Altman

Now that you've had your Atari for a while and you've seen what it can do, wouldn't you like to expand its capabilities? Consider an Atari peripheral that will support up to four disk drives, not only Atari-compatible drives like the 810, but also standard 5¼" and 8" drives—in any combination—single, double or quad density, and single- or double-sided. Add the capacity to communicate with a variety of serial and parallel devices without a costly interface, including printers and modems, plus a printer buffer. Now, what if this device were a computer itself. . .with a 4K mHz, Z-80 microprocessor and 64K of RAM, capable of supporting both Ataris (XLs, too) and standard 80-column terminals, *and* it came with CP/M 2.2? Southwest Microcomputer Products offers all this and more in the **ATR-8000**, and the 64K version costs less than the original 810 drive!

Honest. . .

Like other Atari peripherals, the **ATR** attaches to the serial I/O port. The unit is attractively packaged in a slim cabinet measuring 11½" × 12½" × 2½" and weighs just 8 lbs. There is an illuminated front-panel power switch, and at the rear are sockets for daisy-chaining Atari-only peripherals, like 810s and 40-column printers, as well as card-edge connectors for standard disk drives, printer and serial devices. At least one standard disk drive must be connected to the drive bus. "Standard" disk drives use a 34-pin (5¼") or 50-pin (8") connector with a uniform pinout, independent of manufacturer. Standard drives are more widely used and don't need the complex decoding circuitry of Atari drives, so they are usually available for considerably less. High quality, 40-track, double-sided, double-density 5¼" drives, like Tandon and Teac, for example, sell for about $300, including enclosure. If you already own Percom Atari drives, you're in luck, since they can easily be reconfigured as standard drives. To support double-density and 8" disk drives, SWP offers MYDOS, a sophisticated disk operating system with built-in RS-232 handler. Yet other popular disk operating systems, like OS/A+, are also usable, as well as Atari DOS 2.0, with certain limitations.

The **ATR-8000** is available in several different versions. The simplest includes the Z-80 microprocessor, a 4K ROM, 16K of RAM and a 4K printer buffer. This unit offers most of the features mentioned above, although it cannot be used as a self-contained computer and lacks CP/M capability. The version which may be of most interest to Atari owners is similar and comes with 64K and standard Digital Research CP/M 2.2. The 16K unit, incidentally, can be upgraded. In this configuration, the **ATR** is a stand-alone computer which can be used with any Atari, as well as with an 80-column terminal in the CP/M mode. When the **ATR** is not being used for CP/M, the capacity of the built-in printer buffer jumps to 48K. For those who need more, SWP offers the Co-Power-88 board, which adds a 16-bit, 8088 microprocessor with either 128K or 256K of RAM. This enables the **ATR** to run CP/M-86 and MS-DOS software for the IBM-PC, so long as the programs don't rely on specific hardware features.

CP/M—An operating system.

An operating system is a program that allows the user to interact with and control a computer, while enabling the components of the computer to communicate with each other. Since its development in 1976, CP/M (Control Program/Microprocessor) has become the "standard" operating system for 8-bit computers based on the Z-80 microprocessor family. There are numerous languages and thousands of programs, many public domain, which run under CP/M. Custom operating systems, like Atari's, are hardware-dependent, but CP/M is not designed around any one machine, so CP/M programs run on any CP/M computer.

The ATR-8000

Let's take a closer look at a CP/M system. At a minimum, it must include a central processor, free RAM, one disk drive and a console device for communicating with the processor. Other devices, such as a modem or a printer, are optional. Input/output occurs

through I/O ports. CP/M distinguishes between "logical" and "physical" I/O devices. A logical device is a symbol which can be any one of several physical devices (these may be reassigned). There are four logical I/O devices: console, list device and two for paper tape, the punch and the reader. The console device may be a terminal, teletype or another computer, like the Atari. The list device is nearly always a printer. CP/M differs among computers in the software needed to support a particular set of physical devices.

The CP/M operating system consists of four major parts, including RAM-resident software and programs loaded from disk when needed.

1. FDOS is the core of CP/M. It consists of two parts, BDOS and BIOS, and is always present in memory. BDOS, the Basic Disk Operating System, is identical in all CP/M systems. It processes I/O requests, handles sector allocation, maintains disk files, and so on. BDOS passes its information to BIOS, the Basic Input Output System, which is made up of the subroutines that control a specific computer's I/O devices. BIOS is, therefore, hardware-dependent and must be customized for each CP/M system. The **ATR-8000** CP/M BIOS, for example, contains instructions for communicating with the Atari computer, its console device.

2. The Console Command Processor (CCP) deals with input from the console keyboard. It includes software supporting a standard set of intrinsic commands, like DIRectory, RENname and ERAse, which mainly manipulate disk files. Commands which cannot be directly handled by CCP are passed to BDOS for further processing.

3. The Transient Program Area (TPA) is the CP/M's main memory. Its size depends on the host computer's available RAM (64K maximum). It is here that user programs are loaded and executed. The CP/M system also includes a number of transient commands provided as utility programs on the CP/M master disk. These are loaded into the TPA as needed. Among these are PIP, STAT, SAM, ED, and others, all standard with CP/M.

4. The CP/M system (BIOS, BDOS, CCP) is stored on the first two tracks of the system disk. The CP/M bootstrap program, also on these reserved tracks, directs loading of CP/M into RAM and transfers control to the CCP. With the **ATR**, this process occurs with the "B" command of ATRMON, the **ATR** monitor program.

Using the ATR-8000.

The **ATR-8000** comes with a detailed manual describing the various possible system configurations and includes instructions for interconnecting devices and preparing cables. Atari peripherals are daisy-chained with the usual serial I/O cords. But you'll have to buy or make cables for the standard disk drives, printer and modem, which require multi-conductor ribbon and card-edge connectors.

The 64K **ATR-8000** operates in either Atari or CP/M mode. As an Atari, the system is booted normally by inserting a disk in drive 1 and turning the Atari on. A standard drive is recommended as drive 1, since it can boot DOS for the Atari as well as for CP/M (an 810-type drive cannot be used for CP/M). In this mode, the **ATR** should read Atari disks, protected and unprotected, with no problem. It also replaces the 850 interface and adds the printer buffer. With MYDOS, the print and RS-232 handler I/O commands work as usual. Using the system in double density is easy with MYDOS.

In the CP/M mode, the **ATR-8000** is no longer an intelligent peripheral, but a computer. The Atari acts as a terminal. SWP supplies a 40-column Auto-Term disk which is booted before CP/M is loaded. CP/M requires an 80-column display, which the Atari does not support. The Auto-Term software sets up a scrolling window that shows forty columns at a time on the TV or monitor screen. The window is moved horizontally in either direction with certain key combinations. If you find this annoying, you can opt for an 80-column board for the 800—or you may purchase a version of the Auto-Term software that very nicely emulates an 80-column display. A monochrome or high-quality color monitor is a necessity for the 80-column format.

Booting Auto-Term loads ATRMON. The CP/M system disk is then swapped into drive 1 and, using the "B" command, CP/M is loaded. SWP supplies CP/M Version 2.2 configured for 60K. The system master includes the standard Digital Research utilities (PIP, STAT, etc.) and several programs developed by SWP. The accompanying manual describes CP/M in brief, but you'll undoubtedly have to buy one of the many good books on CP/M to learn the details. SWP disk files include software for formatting and creating system disks, for customizing CP/M to non-standard peripherals, and for using a modem. The **ATR** is capable of reading the CP/M disk formats of certain computers with no modifications. For others, there is DISKDEF.COM, a utility that sets up a selected drive to emulate the disk characteristics of another CP/M machine. Among these are Osborne, Kaypro, TRS-80, and Xerox, as well as a variety of generic 5¼" and 8" disk formats. With this program, CP/M-based software written for these computers, like **Wordstar** and **Microsoft BASIC**, can be read into the **ATR-8000**. CP/M programs can also be transferred via modem. Public domain software of all kinds is freely available from a wide network of CP/M bulletin boards.

The **ATR-8000** is a truly innovative product for the Atari. With CP/M, it closes the gap between the Atari community and the computer world at large, and it offers an outstanding value for Atari users interested in economically expanding their systems. □

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

DR. C. WACKO'S MIRACLE GUIDE TO DESIGNING AND PROGRAMMING YOUR OWN ATARI COMPUTER ARCADE GAMES
Written by David L. Heller, John F. Johnson and Robert Kurcina
Published by Addison-Wesley
$24.95

by Stephen Paul James

Dr. C. Wacko's Miracle Guide explains basic Atari character graphics, flip-flop animation, sound and player/missile graphics. It lists sample programs and includes a floppy disk, so you don't have to type and debug the programs. In addition, the book throws in numerous cartoon illustrations that simplify animation concepts and, generally, teaches in a humorous style.

The introductory chapter describes overall game design themes for arcade games, and chapter two starts in on the programs. The first programs illustrate arcade game screens—just backgrounds, nothing else. Thus, the beginning program titles are a little misleading; booting up the programs **Super Breakout** or **Tron**, you think you're going to get a full game, but all you get is a static display with some nice colors. However, the displays *are* colorful, and the strange doctor teaches good PLOT, DRAWTO, LOCATE, etc. techniques.

This second chapter also contains a **Bong** program that is reminiscent of the old **Pong** arcade game. The book explains the program in detail, but it isn't utilized in the book's final game—leading readers to figure out by themselves how to incorporate the **Bong** concepts into a full-blown game.

Character graphics (customizing your own set of character symbols to use as arcade figures, monsters, space ships, etc.) is discussed in chapter three. The character generator utility is called the **Monster Maker** and is easy to use and nicely explained.

Flip-flop animation follows, as Dr. Wacko gives an example of how to animate a character. The only problem here is that, as his pupils are learning elementary programming techniques, the professor adds machine language with the USR function. The machine language is not explained, which seems logical for an introductory book, but the machine language routine changes in each animation sample program, leaving a sense of wonder—how transportable are the machine language routines in other applications? Still, experimentation is easy and will validate any assumptions.

A plus for the doctor's class is that one of the animation programs can be used in conjunction with the **Monster Maker** (character generator), letting you develop your own monster face and feet, watching it blink its eyes and dance.

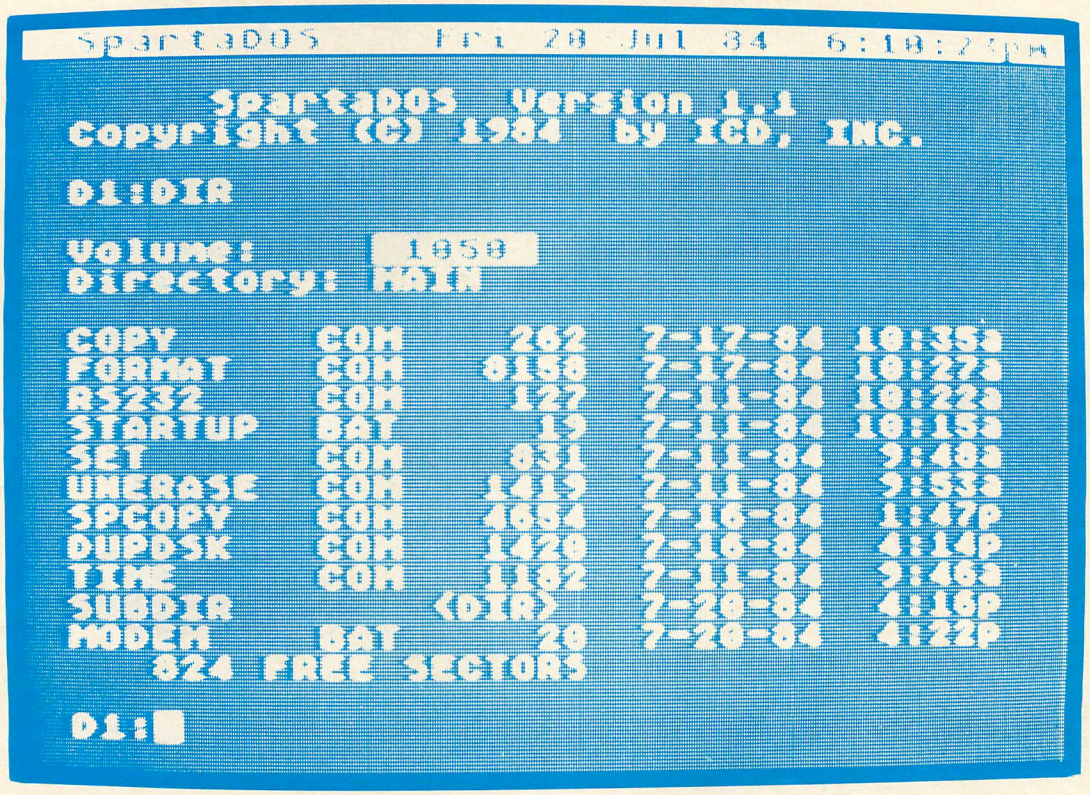
The fifth and sixth chapters deal with joystick movement (simple concepts), and then chapter seven molds animated figures with joystick controls—a good chapter.

Chapter eight advances to figures that chase you around the screen. A nice tutorial on collisions and wrap-around is included. Nine covers sounds—notes and chords. General music theory is lightly covered, but (since this isn't a book on music) you aren't being trained to conduct a symphony. I thought that it would have been *nice* to see a utility for making a bar of music on the screen, hearing it and loading it into data statements. But the wacko professor lets you use paper and a lot of manual experimenting to get the right tune. The book gives a good enough foundation for music and other aspects so that you could create your own music utility.

The chapter on player/missile graphics is a good introductory tutorial on the use of one player. The use

(continued on page 92)

. . . 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.

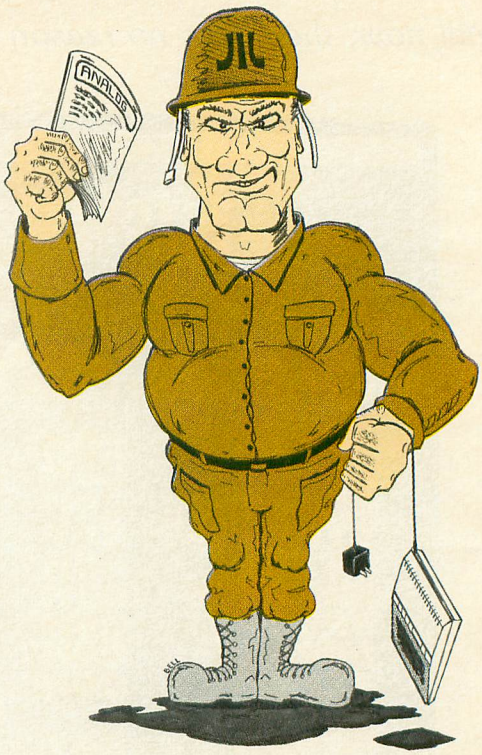
other fine products from ICD include: The Chip with Archiver I™ . . $99.95, Happy Archiver™ . . $39.95, and coming soon Archiver II™.

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.

BASIC Training



by Tom Hudson

As was explained last issue, Joel Gluck is unable to complete the **Our Game** column begun in **ANALOG Computing** issue 12. I was "volunteered" to take over the task of developing a BASIC game and that's just what I'm going to do.

But let's get one thing straight. . .This isn't "our game." It's *my* game and my rules. We're going to take the next four issues and write this game—piece by piece, discussing as many details as possible. We'll go from simple input routines all the way to artificial intelligence.

Are we going to vote on what the game will be? No way. This is **BASIC Training**, and I'm the D.I., the Drill Instructor. What I say, goes. Of course, if you have any suggestions for our *next* game, after this first one is complete, let me know. I know we'll be a great team.

The game.

True to the military theme of this column, our first game will be an adaptation of that old pencil-and-paper favorite, **Battleship**. Before you start moaning about "old" ideas, stop and think. The object of this column is to teach techniques for writing a BASIC game. This game won't win any beauty contests, but it'll sure teach you a thing or two.

The game will fit in 16K. This was quite a trick, and we'll see how it was done.

It's you against the computer, not you against another player (yecch). The computer's response time is very acceptable.

If you want to make improvements in sound, graphics and so on, you can. The game's artificial intelligence (AI) routines are very solid; it even beats me! I left out fancy graphics, so it could fit in 16K (most of the memory is used by the AI routines). All the game will need is some polish and it'll be great.

Okay, let's get started.

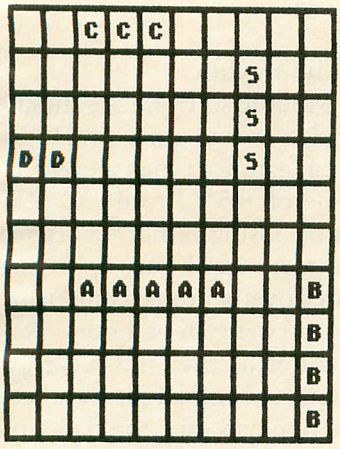
What's Battleship?

Battleship is a game for two players (*or* one player and a computer), in which each players tries to sink the other player's fleet of five ships. The ships are as follows: destroyer (2 units long), cruiser (3 units long), submarine (3 units long), battleship (4 units long) and aircraft carrier (5 units long). Each player sets up his fleet on a ten-by-ten-unit "ocean" grid. The ships will be placed either horizontally or vertically. They can touch each other, but no overlaps are allowed. Figure 1 shows a typical fleet placement on the ocean grid.

After the ships are set up for both players, they take turns "shooting" at each other's ships. Shooting is done by calling out the coordinates of the desired target location. Figure 2 shows how the coordinate system of the ocean grid is set up.

The coordinates used are the letters *A* through *J* along the left edge of the grid, and the numbers *0* through *9* along the top. The individual squares are referenced by a letter-number combination. For example, the square in Figure 2 marked with an *X* would be referenced as *G4*.

On each turn, the players shoot at one square. The



D = DESTROYER
C = CRUISER
S = SUBMARINE
B = BATTLESHIP
A = A. CARRIER

Figure 1.

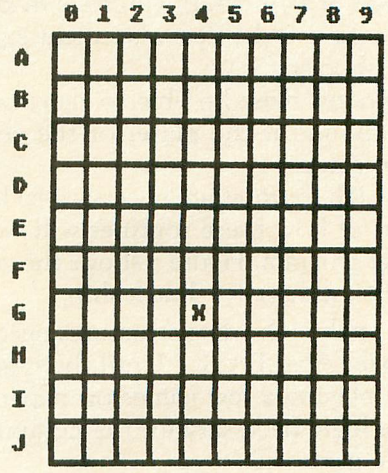


Figure 2.

opponent fired upon tells whether the shot was a "hit" or a "miss." If a ship was sunk, the opponent must inform the shooting player that the ship was sunk, as well as indicating what type of ship it was.

Shooting continues until all five of a player's ships are sunk.

Writing the game.

This issue, we're just going to examine the game itself and decide what our program will have to do. We won't write any code for the game for a couple of installments. Why? The answer is simple.

When you're thinking of writing a game, you should have a very clear idea of *exactly* how the program will work. If you just sit down at the computer and start coding, you'll be sorry later. One of the best things that ever happened to me was when I was thinking of writing **Livewire** (issue 12) while at the West Coast Computer Faire. I didn't have my computer and was forced to think the idea through for over three days! By the time I got back to ANALOG, I had the data storage format—and most of the code—written in my head.

Okay, now let's see what routines **Battleship** will need. First, the game's ten-by-ten-unit ocean grids must be represented in memory somehow and then initialized.

Second, the computer must randomly determine who shoots first, the player or the computer. I usually refer to this process as the "coin toss."

Next, there must be a human interface which will allow ship positioning and shooting. Both these actions will require a grid coordinate conversion from the G4-type coordinates to the internal grid representation.

Then, the program must have enough intelligence to play a challenging game. This is the key to the

(continued on next page)

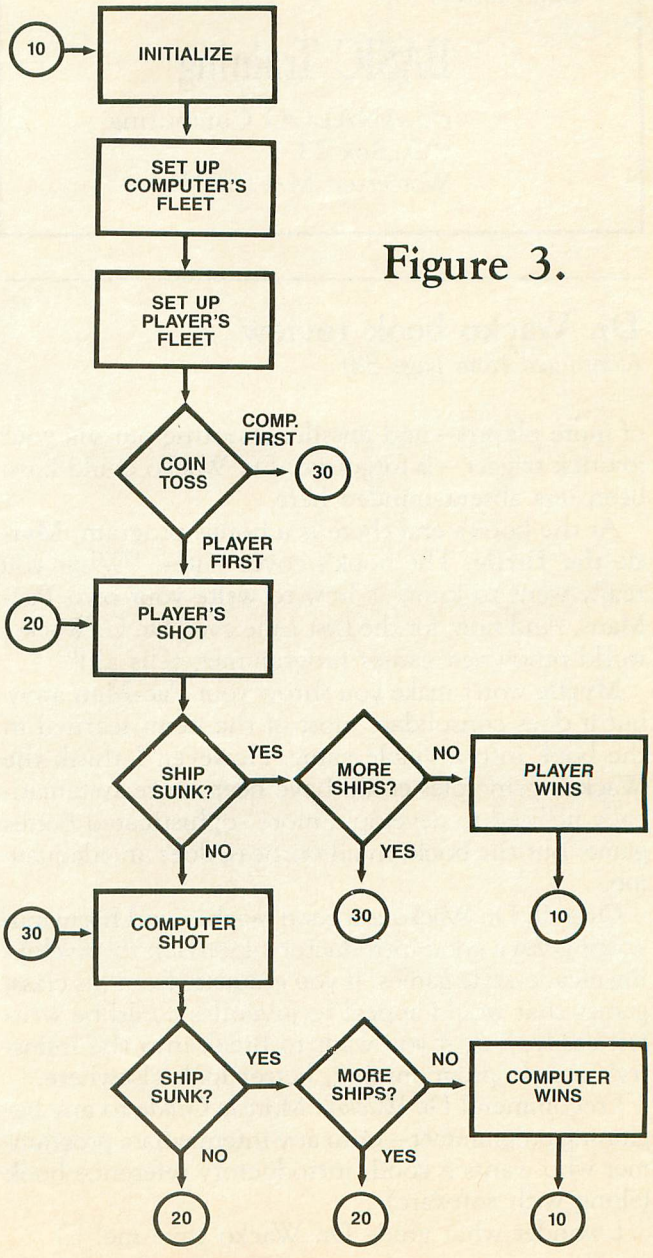


Figure 3.

game, because nobody wants to play a "pushover." There are two phases of artificial intelligence in **Battleship**. The first of these is the fleet placement logic; the second is the shooting logic.

Finally, the computer must be able to determine when ships are sunk and if the player or the computer has won the game.

Now we know which routines are necessary, so let's look at a flowchart of how these routines will work together in the final program. Figure 3 shows the general, non-detailed flowchart for **Battleship**.

Next issue, we'll analyze the data structures needed to write **Battleship** for Atari BASIC. Until then, maybe you should try playing a few games on paper to get a "feel" for the logic necessary for the computer to play this simple game. □

Send letters to:

BASIC Training
c/o **ANALOG Computing**
P.O. Box 23
Worcester, MA 01603

Dr. Wacko book review

(continued from page 88)

of more players—and missiles shooting out via your joystick trigger—is forgotten. Dr. Wacko could have been less absent-minded here.

At the book's end there is a bonus program, **Myrtle the Turtle**. The book's cover states, "What you really want to know is how to write your own **Pac-Mans**. And now, for the first time ever, Dr. C. Wacko, world-renowned games programmer, tells all!"

Myrtle won't make you throw your **Pac-Man** away, but it does consolidate most of the items learned in the book into a viable game. However, I think the Wacko's principles could have been more systematically utilized to develop a more sophisticated bonus game. But the book, in all fairness, does an adequate job.

Overall, Dr. Wacko is a good teacher, and his manuscript gives a good introductory approach to developing arcade-style games. If you graduate from his class, games that would appeal to juveniles could be written easily; but, if you want to break into the industry's arcade programming, better look elsewhere.

I recommend *Dr. Wacko's Miracle Guide* to any beginning programmer—or to any intermediate programmer who wants a good, introductory reference book (along with software).

I wonder what grade Dr. Wacko gave me! □

Griffin's Lair

(continued from page 10)

The module.

The **Temperature Module** software is contained in the 16K ROM cartridge and is quite simple to use. It includes a number of features. One can record temperatures in the range of −5°C to 45°C (23°F to 113°F) with the sensor. Using either Centigrade or Fahrenheit, an experiment can be set up to measure up to 121 temperatures over a pre-selected timespan of from ten seconds to twenty-four hours. A graph displays the temperatures as they are recorded.

Once the experiment is completed, signified by a helpful musical tone, one can save the data on disk, view the data in tabular form or even print out the graph on an Epson printer. Dual temperatures can be measured with an additional sensor.

The sensor can be extended a longer distance from the interface with phonojack cables, readily available at electronic stores. The sensor can be calibrated for greater accuracy using a BASIC program included in the manual.

The manual.

The instruction manual is an integral part of this science package. The software contains its own documentation, and it's not necessary to use the manual for this purpose. However, the manual does explain how to go about setting up an experiment in general and describes several projects dealing with temperature in particular. These activities, though quite simple, are presented in a concise manner which is easy to understand.

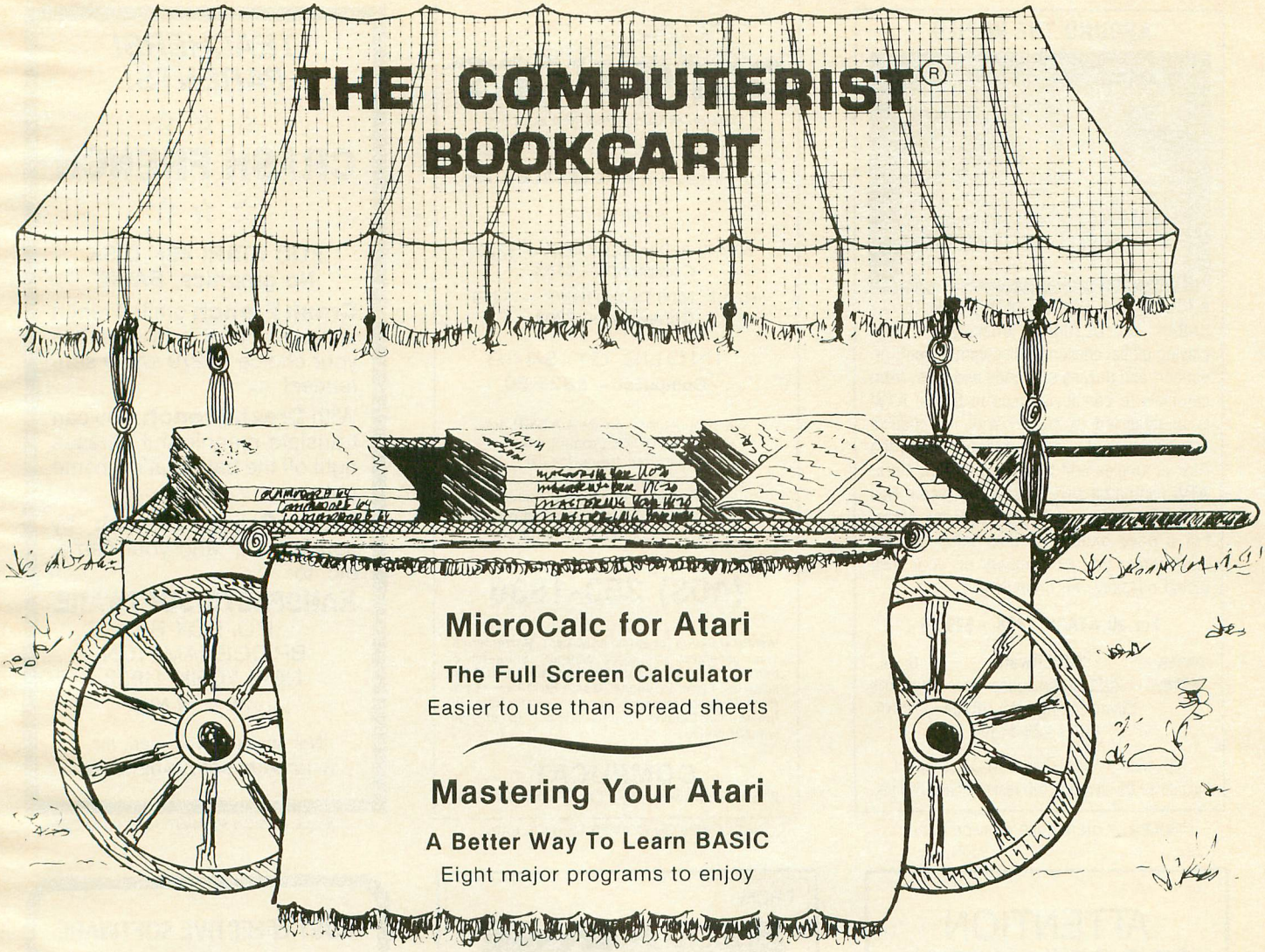
There is an excellent chapter which discusses historical perspectives in the measurement of temperature. Following each section is an extensive suggested reading list. Several pages of sample tables and graphs are included, so that permanent records of the data may be kept. The manual, similar to the laboratory workbooks used in school, is well organized and very successful in its attempts to guide the young investigator.

The proverbial words of wisdom . . .

Atari has certainly gotten off to a good start with the first of their science series, this one developed at Dickinson College. For ages nine to adult, this educational package will be valuable, whether used at home or in the classroom. At the very beginning of the manual, a Chinese proverb is quoted which summarizes the benefits of this innovative approach to home computer education:

I hear, I forget.
I see, I remember.
I do, I understand.

Atari's entry into this field is exciting, and I'm eagerly looking forward to the other modules planned for this series. □



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**

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.

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.

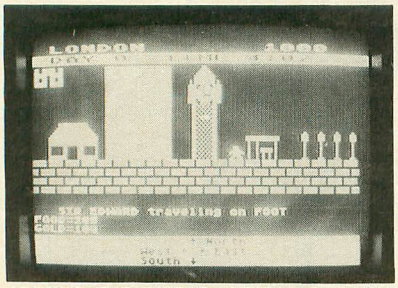
- Unlimited calculation length & complexity
- Screens can be linked and saved on disk/cassette
- Build a library of customized screens
- Provide formatted printer output

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

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)

AROUND THE WORLD



AROUND THE WORLD (ATW), akin to ALI BABA® and ULTIMA®, offers you a role-playing global challenge filled with adventure, danger, and dozens of friends and foes, from cardinals to cannibals, asps to tigers. **ATW** is unparalleled in scope, with 40 endless scrolling regions - from London through the African jungles and the Orient to New York. **ATW** has game saves and a high score Hall of Fame. No two games are ever the same. For a price much less than other graphic adventures, you can embark on your own global odyssey. So order NOW!

For all ATARI® 48K - $29.95

ORDER NOW!

SUPERware
2028 Kingshouse Rd.
Silver Spring, MD 20904
(301) 236-4459

Dealer Inquiries Invited

MC, VISA, AMX, Check, M.O. accepted
Include $2.00 shipping, MD residents add 5% tax.

CIRCLE #137 ON READER SERVICE CARD

COMPUCAT

Catalog of Computers and Supplies

Our Prices Are **WHOLESALE + 10%**

SAMPLES!!!

ATARI 850 INTERFACE — $220
Compucat — $163

RANA 1000 DISK DRIVE — $449
Compucat — $308

INDUS GT—$449
Compucat — $325.60

We support the complete **ATARI** and **COMMODORE** product lines.
Ask for our free price list.

You may order in the regular manner or download our *TeleCatalog* and order from your computer or terminal.

(408) 353-1836

Instant shipping (or as fast as we can). *Mastercard & Visa accepted* (no extra charge). Shipping & handling add 5%. California customers add 6.5% sales tax. Order by *phone* (Mon. - Fri. 10 am - 5 pm PST). Order by *modem* (Daily 6 pm - 9 am) from our online TeleCatalog. Prices subject to change without notice.

COMPUCAT
24500 Glenwood Hwy., Los Gatos, CA 95030

CIRCLE #138 ON READER SERVICE CARD

TEACHERS! PARENTS!

☆ ☆ ☆

CRASH FRENCH

☆ ☆ ☆

FOR ATARI 48K DISK
No graphics, BASIC

Crash French omits French accents, but it can translate your choice of 279 entire sentences!

With **Crash French**, you can translate meaningful phrases right off the bat! That's a home run.

Order **Crash French** by sending $25 and your name, etc. to:

HARDFACTS SOFTWARE
P.O. BOX BF
BRIDGEHAMPTON
NEW YORK 11932
(+7¼% IN NY)

Not affiliated with Atari, Inc.
HARDFACTS SOFTWARE © 1984

CIRCLE #139 ON READER SERVICE CARD

ATTENTION PROGRAMMERS

Our established literary agency is seeking to represent talented software authors. If you've written "The Great American Program" but are unsure how to market it or to whom or for how much, call us.

Whether it's recreational, educational or business, we'll make sure that your creation gets the audience it deserves—Top Executives at the leading software houses.

Our clients receive better treatment and earn more royalties because we negotiate the best possible deals. Give us a chance to go to bat for you. For further information on the benefits of representation please contact:

THE ROBERT JACOB AGENCY
(805) 492-3597
P.O. Box 4073
Thousand Oaks, CA 91359

CIRCLE #140 ON READER SERVICE CARD

FROM

SENECOM

THREE PDQ DISKS AND A NEW FUTURE
$9.95

There is untapped potential in your Atari. And, perhaps, in you. . .

We'd like to share an opportunity too unique and worthwhile to describe in an ad. We're offering this remarkable bargain so you can see it for yourself without risk.

Order three PDQ (Premium Disk Quality) Disks (DSDD, 21-year Warranty) for only $9.95. The front of each disk is for your normal use. Boot in the program on the back (min. system: 48-K), and you may be surprised at what your Atari tells you.

Order PDQ! Send just $9.95 to:
SENECOM
Dept. 101
13 White St.
Seneca Falls, N.Y. 13148

We'll pay for shipping.
NYS Residents: add 7% Sales Tax.

Offer limited to one order per address at this price.

Atari® is a registered trademark of Atari Corporation.
SENECOM is a registered trademark of Seneca Computer Company, Inc.

CIRCLE #106 ON READER SERVICE CARD

COST EFFECTIVE SOFTWARE

BY

"The Best Little Software House In Texas"

HOMEBASE by SOFT SECTRE is a versatile database program for the home, small business or lab. In a 'USER FRIENDLY' MENU driven format HOMEBASE utilizes TWENTY COMMANDS: CREATE, ADD, LIST, CLIST, CHANGE, CONCATENATE, SEARCH, SUM, SORT, DELETE, LABELS, LOAD, PRINT, SAVE, DIRECTORY, HELP, DRIVE, AUDIO, LOWER CASE, END. An optional SECURITY CODE prevents unathorized data file retrieval and manipulation. Optional AUDIO FEEDBACK signals the end of a command response. The ATARI version uses a MACHINE LANGUAGE SORT!

HOMEBASE is ideal for: MAILING LISTS (we use it ourselves); household, insurance and equipment INVENTORIES; MAINTENANCE schedules; a PRESCRIPTION list; refund COUPON and GROCERY lists. Ready-to-use data files for these tasks are included. Why buy several programs? Purchase the ONE program that will handle all of your database needs. HOMEBASE by SOFT SECTRE will SAVE you TIME and MONEY!

SELECTRE by SOFT SECTRE is included in the disk based package at no additional charge (a $29.95 value!). Disk DIRECTORIES and PROGRAMS are loaded with numeric key input allowing even the most inexperienced person to easily operate the computer.

ATARI OR PET DISK 32K $49.95
PET TAPE 16K $49.95

Add $2.00 for shipping.

SOFT SECTRE
P.O. BOX 1821, PLANO, TX 75074
Send for FREE catalog! VISA/MasterCard WELCOMED

CIRCLE #142 ON READER SERVICE CARD

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.

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**

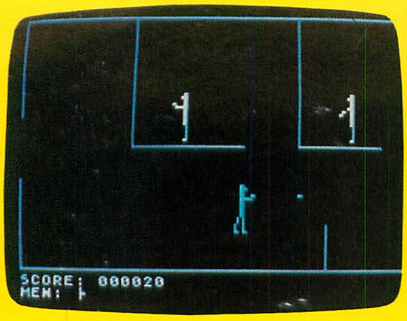
Now Available At Your Software Dealer

OSS™

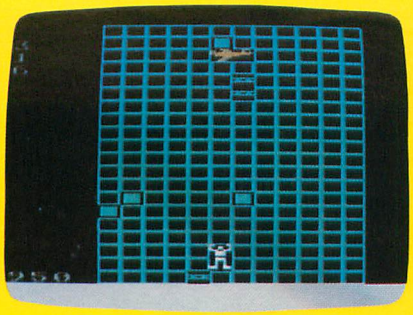
Optimized Systems Software, Inc.

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

CIRCLE #143 ON READER SERVICE CARD

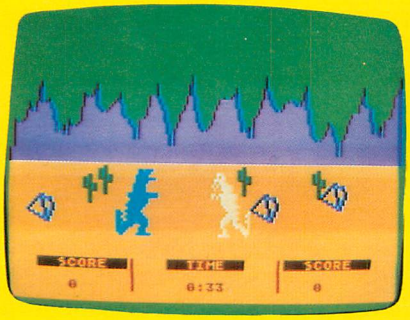


Maniac!

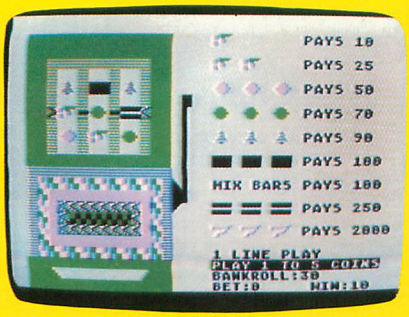


Stuntman

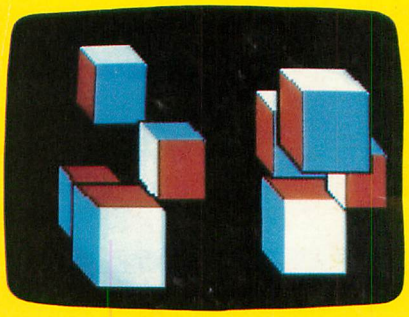
Fill 'Er Up



Dino Battle

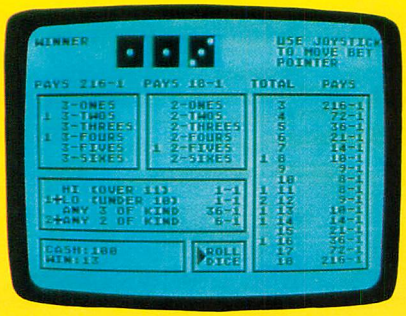


Color Slot Machine

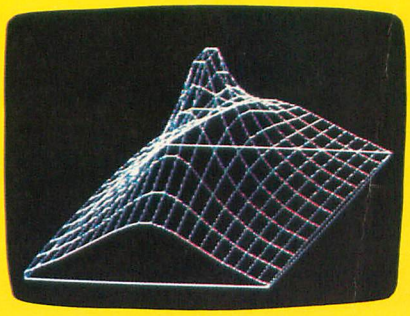


Cubes

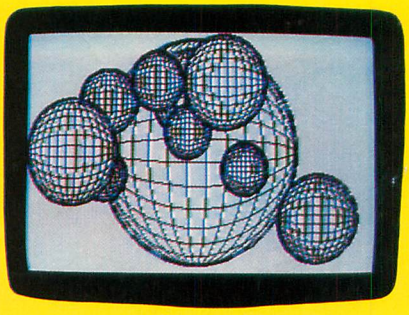
Where can you get all of these programs (and dozens more!) for only $14.95?



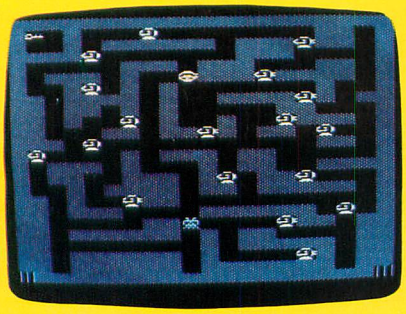
Triple Threat Dice



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.**