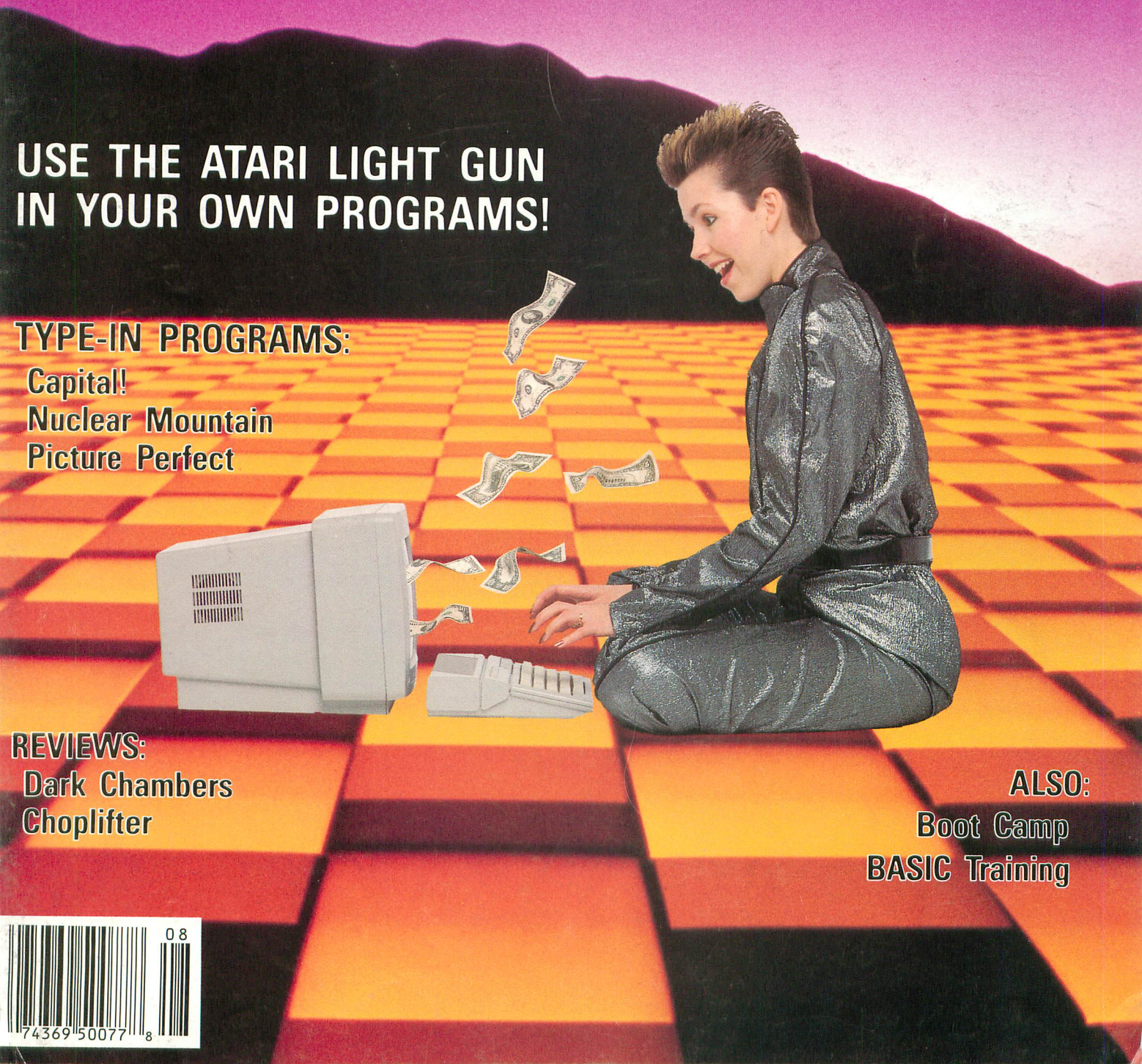
The #1 Magazine For Atari Computer Owners

ANALOG COMPUTING T.M.

FDC 50077

AUGUST 1989
ISSUE 75

DISK VERSION $12.95



USE THE ATARI LIGHT GUN IN YOUR OWN PROGRAMS!

TYPE-IN PROGRAMS:
- Capital!
- Nuclear Mountain
- Picture Perfect

REVIEWS:
- Dark Chambers
- Choplifter

ALSO:
- Boot Camp
- BASIC Training

08
74369 50077 8

Give 'Em A.N.A.L.O.G., Harry!

Two Historic Facts:

1 Dewey did not defeat Truman for the Presidency in 1945. Truman went on to be known for his truthful, forthright style and as one of the nation's most popular Chief Executive Officers.

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

1 YEAR FOR ONLY $28

SAVE $19 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $79

NEW LOW PRICE!

Editorial

ILLUSTRATION BY JOHN BERADO

BY CLAYTON WALNUM

One of the newest pieces of Atari 8-bit hardware is the light gun that comes with the XE Game System. It can also be purchased separately, although I've never seen it in a store. If you're interested in obtaining one, you can order directly from Atari.

The light gun can add a lot to computer games by allowing the player to interact in a more "natural" manner than the joystick allows. Unfortunately, up until now, there has been little or no documentation published on how to use the light gun from a programming point of view.

This month ANALOG makes up for that lack with the publication of Matthew Ratcliff's *Gun Assist*, an assembly language subroutine that you can use in your own programs. *Gun Assist* takes care of all the details involved in incorporating the light gun in your games. Further, Matt's informative article tells you exactly how the light gun works. He passes along everything he learned during his exploration of this new Atari peripheral.

Of course, we haven't published *Gun Assist* just to be nice guys. We hope that, in the near future, we'll see many game (and maybe even nongame) submissions that incorporate the light gun. We've supplied the documentation, now let's see what you can do with it!

Also in this issue, we have, from the prolific Bryan Schappel and Barry Kolbe, *Capital!*, a sensational game of high finance. This program was inspired by another popular board game, but adds its own twists and turns to the now famous real-estate buying-and-selling scenario.

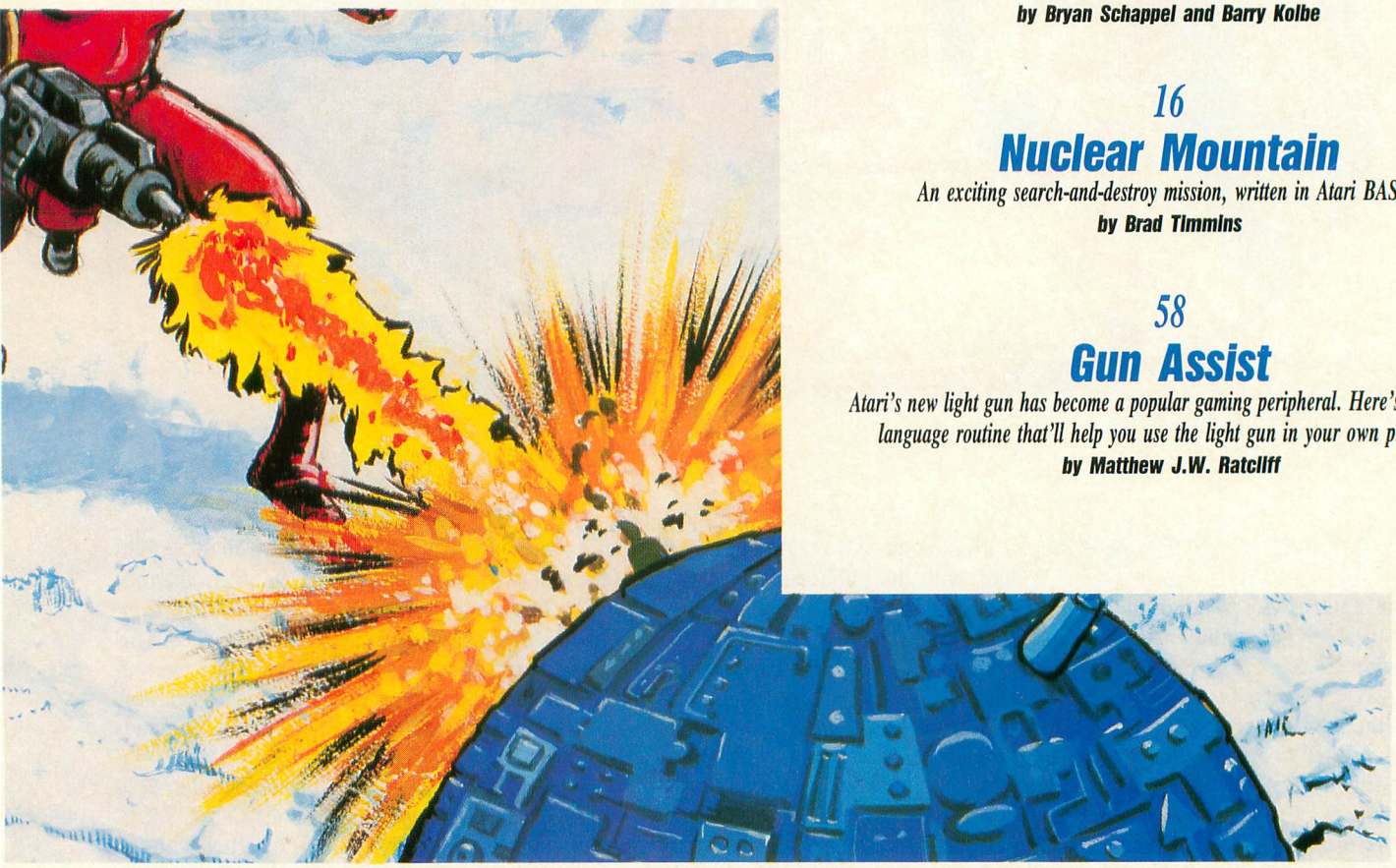
In addition, Tom Hudson's popular *Boot Camp* continues, and so does *BASIC Training*. When you put this all together with the reviews and the other regular features, we think you'll find this issue to be as exciting and informative as usual.

But enough of this chatter. Let's start reading.

Cont

on page 10

FEATURES



on page 16

ANALOG Computing (ISSN 0744-9917) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. U.S. subscription: $28 for one year (12 issues), $52 for two years (24 issues), $76 for three years (36 issues). Foreign subscription: Add $10 per year. Single copy $3.50 (add $1 for postage). Change of address: Six weeks advance notice, and both old and new addresses are needed. POSTMASTER: Send change of address to **ANALOG Computing** Magazine, P.O. Box 16927, North Hollywood, CA 91615. Second-class postage paid at Beverly Hills, CA, and additional mailing offices.

e n t s

R E V I E W S

C O L U M N S

D E P A R T M E N T S

ANALOG COMPUTING STAFF

Publisher
LEE H. PAPPAS
Executive Editor
CLAYTON WALNUM
Art Director
KRISTEL PECKHAM
Associate Editor
ANDY EDDY
Managing Editor
DEAN BRIERLY
East Coast Editor
ARTHUR LEYENBERGER
West Coast Editor
CHARLES F. JOHNSON
Contributing Editors
MICHAEL BANKS, FRANK COHEN, MATTHEW J. W. RATCLIFF
Cover Photography
GARRY BROD
PETE TURNER
Models
RUTH A. RINARD-FALEY
Illustrations
JOHN BERADO
BOB McMAHON
K.P.
Copy Chief
SARAH WEINBERG
Copy Editors
NORMA EDWARDS
RANDOLPH HEARD
TIM POWER
KIM TURNER
Editorial Assistant
PATRICIA KOURY
Chief Typographer
ALICE NICHOLS
Typographers
DAVID BUCHANAN
B. MIRO JR.
QUITA SAXON
LIGAYA RAFAEL
Contributors
JOE D. BRZUSZEK
TOM HUDSON
BARRY KOLBE
BRYAN SCHAPPEL
BRAD TIMMINS
Vice President, Production
DONNA HAHNER
Advertising Production Director
JANICE ROSENBLUM
Advertising Production Coordinator
MAGGIE CHUN
National Advertising Director
JAY EISENBERG
(213) 467-2266
(For regional numbers, see right)
Subscriptions Director
IRENE GRADSTEIN

Analog Computing Published By L.F.P., Inc.
President
JIM KOHLS
Vice President, Sales
JAMES GUSTAFSON
Vice President, Client Relations
VINCE DELMONTE
Corporate Director of Advertising
PAULA THORNTON
Corporate Editorial
TIM CONAWAY, PAMELA CARR

AUGUST 1989
ISSUE 75

Where to Write

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

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

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

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

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

Advertising Sales

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

Permissions

Subscriptions

ANALOG Computing, P.O. Box 16927, North Hollywood, CA 91615; (818) 760-8983. Payable in U.S. funds only. U.S.: $28-one year, $54-two years, $76-three years. Foreign: Add $10 per year. For disk subscriptions, see the cards at the back of this issue.

Authors

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

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

JE Publishers Representative

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

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

8 BIT NEWS

Express! *on cartridge*

Orion Micro Systems has announced a cartridge version of Keith Ledbetter's popular *Express!* telecommunications program. The program is stored in a 64K bank-switched cartridge and is written entirely in assembly language, making the program compact enough to allow the addition of many new features. The new version of *Express!* is completely rewritten and will work with any modem that has an "R:" handler available for it.

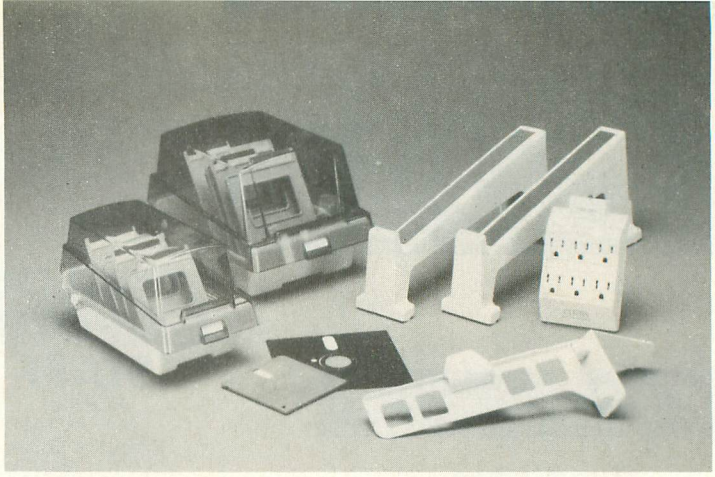
The new *Express!* uses drop-down menus like those on the ST. Also, the program supports the Atari XEP-80 80-column card, although, when running in 80 columns, the drop-down menus are replaced by standard text menus.

One of the new features added to this program is a full-screen editor that allows you to compose messages, as well as edit text in the capture buffer. The text editor supports such functions as cut-and-paste, line tagging, find and search-and-replace.

The cartridge has been designed so that other cartridges may be "piggy-backed" into it, allowing you to turn off *Express!* and switch to the another cartridge at will. The cartridge version of *Express!* is $69.95.

Orion Micro Systems
2211 Planters Row Drive
Midlothian, VA 23113
(804) 794-9437, 6 p.m. to 10 p.m.

CIRCLE #104 ON READER SERVICE CARD.



Starter pack

The Computer Starter Kit, marketed by Curtis Manufacturing Company, supplies new computer owners with five computer accessories that are designed to "protect computers, save desktop space and create greater user comfort." Included in the package is a copy holder, universal printer legs, computer cleaning kit, a surge protector, and a disk file for either 3½-inch or 5¼-inch diskettes.

All of the products that make up the kit are standard equipment that have been available separately, but which have been packaged together for "ultimate consumer convenience and additional savings," according to Tom Judd, president of Curtis.

The Computer Starter Kit carries a suggested retail price of $59.95.

Curtis Manufacturing Company, Inc.
30 Fitzgerald Drive
Jaffrey, NH 03452
(603) 532-4123

CIRCLE #105 ON READER SERVICE CARD.

8-bit Pascal

Just released from CLSN Software is CLSN Pascal. This implementation of the popular programming language boasts an in-line editor, a compilation speed of 1,000 lines per minute, run-time error locator and more. CLSN Pascal supports a full set of data types, including char, boolean, byte, shortint, word, integer and longint. Arrays, sets, file types and records are also supported. The following is a list of reserved words in CLSN Pascal:

absolute	end	mod	set
and	file	nil	shl
array	for	not	shr
begin	forward	of	string
case	function	or	then
const	goto	packed	to
div	if	procedure	type
do	in	program	until
downto	inline	record	var
else	label	repeat	while
			xor

CLSN Pascal sells for $39.95.

CLSN Software
10 Arlington Place
Kearny, NJ 07032
(201) 998-1554

CIRCLE #106 ON READER SERVICE CARD.

Revenge of the NERDS

The National Educational Report Drawing Services (NERDS) has just released a new set of graphics disks for use with Broderbund's *Print Shop*. The graphics on these disks are directed at the educational computer market and, according to NERDS, each pair represents about 150 hours of research and drawing.

The newly released Map Disks 3 and 4 include over 230 pictures of Russia, China, Africa and the Far East. Map Disks 1 and 2 are also available and include maps of the United States, South America, Central America, Europe and Canada. In addition, NERDS supplies biology and periodic table disks.

The NERDS graphics disks are $15 per pair.

NERDS Software
18 Wendy Drive
Farmingville, NY 11738

CIRCLE #107 ON READER SERVICE CARD.

READER COMMENT

Snowplow correction

There is a problem with the game *Snowplow* from the September '88 issue. It seems that the last game board does not load in. I worked on and tested the routines that are supposed to do this, and they seem to work by themselves but fail when put together. So I rewrote these routines, and the following BASIC program can be used to modify the original game. To be on the safe side, make a copy of *Snowplow* on another disk. Check the data over carefully when you type the listing. A tiny mistake can cause major problems. Place the disk containing your copy of *Snowplow* into your disk drive, and then run the program below. Follow the prompts, and your copy of *Snowplow* will be modified on the disk.

```
100 REM SAVE"D:SNOW12.FIX
110 DIM N$(20),W$(20):120 ? "K":? "NAME
 OF SNOWPLOW FILE";:INPUT N$
130 W$="D:":W$(3)=N$
140 TRAP 160:OPEN #1,12,0,W$
150 GOTO 180
160 ? "FILE NOT FOUND":END
170 REM POSITION AT NEWBEG
180 ? "FINDING PLACE"
190 FOR I=1 TO 1273:GET #1,A:NEXT I
200 REM PUT IN NEW STUFF
210 ? "INSERTING PATCH"
220 FOR I=1 TO 74:READ A:PUT #1,A:NEXT
I
230 REM SKIPPING
240 ? "ANOTHER PATCH"
250 FOR I=1 TO 93:GET #1,A:NEXT I
260 REM FIXING JMP
270 FOR I=1 TO 3:READ A:PUT #1,A:NEXT I
280 CLOSE #1:? "DONE":END
290 REM DIRECTORY PATCH
300 DATA 169,64,133,178
310 DATA 32,90,66,32,215,74,169,3,133,1
82
320 DATA 169,0,133,186,32,225,73,32,81,
78
330 DATA 32,36,75,32,48,71,32,130,70,32
340 DATA 215,74,32,50,65,165,178,201,64
,208
350 DATA 8,230,178,32,166,79,76,106,64,
165
360 DATA 178,141,235,70,32,156,70,16,6,
169
370 DATA 64,133,178,208,226,230,178,234
,234,234
380 REM 74 BYTES
385 REM JMP PATCH
390 DATA 76,53,64
```

There is a new restriction to the names of the game boards that you create. They must be named "SMAP.?" where the question mark must be a letter from A through Z. *Snowplow* will load these screens in alphabetical order and will go back to the built-in game board whenever it doesn't find the next letter in the alphabet. Any boards you have already created should be renamed in the order you wish to play them.

—Barry Kolbe
Mazomanie, WI

Libelous statements?

Shame on you for permitting Frank Cohen to deliver such a low blow to one of the finest Atari clubs in the country (top of page 51 in the April '89 issue).

Since when does a brief verbal disagreement between two spectators at a computer show rate news coverage in a national magazine? Frank Cohen should also get his quotes straight before he writes libelous statements that sully the pristine reputation of a fine club. The phrase as printed was never uttered, certainly not with DALACE as the subject. The rest of his comments about DALACE are equally absurd.

The Dallas Atarifest was well organized and took place in one of the finest exhibit halls in the country. Nearly 100 DALACE volunteers assisted in the setting up and tearing down of the show. These same volunteers manned the booths and the ticket sales, and frankly, I thought we all had a great time, with thousands of visitors attending.

We can understand that, after having produced 8 through 10 shows on the road prior to Dallas, Sandy Austin was no doubt very tired (it was almost inhumane to expect one person to handle that much responsibility), but to suggest that Sandy quit because of Dallas is ridiculous. Sandy quit almost two years after the Dallas Atarifest.

We wish you had not printed Mr. Cohen's unfounded remarks. The damage has been done. However, we do want you to know that we appreciate ANALOG's long history of support for the users' groups. Once mistake will never dim our long-standing admiration for your magazine. Long live ANALOG and the 8-bit Atari machines.

—Jeff Golden
Irving, TX

We're sorry if Frank's ST Notes *in the April issue was offensive to you, but we have to say that we're not really sure why you should be so upset. You seem to be reading things into the article that don't exist.*

First, Frank certainly never made any libelous statements. He did report that a users' group member shouted, "DALACE is a pirate club," but that wasn't his remark, and it certainly wasn't meant to imply that the members of DALACE were involved in software piracy. The quote was included to illustrate the growing tensions between the users' groups involved.

We just reviewed the article and, outside of saying that there was particularly heavy friction between DALACE and the North Texas Users' Group, we can find none of the "absurd" comments you are referring to.

Also, it wasn't reported that Sandy Austin quit her job right after the Dallas Atarifest; what Frank said was "Later, Austin confided to ANALOG *that she was looking for another job." There's certainly no indication of when Sandy actually made her separation from Atari.*

Once again, we apologize if the article in question was offensive to anyone. ANALOG *is quite aware of how difficult and frustrating it is to organize a large Atarifest. We did, after all, have a great deal to do with the Worcester Atarifest. We salute all those people whose efforts have made the past Atarifests such a success.*

Fixes for Ultimate Graphics Convertor *and* Secret Agent

We recently had production troubles with a couple of program listings, as undoubtedly many of you have noticed. If you follow the instructions given below when typing these listings, you should be able to reproduce them with little trouble.

First, Listing 1 of the *Ultimate Graphics File Convertor* in the May '89 issue was incorrectly laid out in the magazine, resulting in the order of the lines being jumbled. Worse, due to the jumbling, some of the lines were cut in half. To get this program running, type it from beginning to end (it doesn't matter that you will be typing some of the lines out of order), except for Lines 7000, 8236, 11200, 13501 and 13610. These are the jumbled lines. Once you've got the listing typed, add the following lines:

```
PA 7000 REM GR.8 PUT
AM 8236 FOR N=1 TO 5HIFT:A=U5R(ADR(ROL$),
   START+7680,7680):NEXT N
CG 11200 POP :GO5UB 13030:? "KERROR ";PEE
   K(195);". CORRECT AND PRE55 ANY KEY.":
   GET #2,K
QB 13501 5ECT=1:GO5UB 10000:IF BUF$(1,17)
   ="55I CLIP NEW5ROOM" THEN 13600
CM 13610 5IZE=A5C(BUF$(28)):COUNT=0:5=1:N
   =32:GO5UB 20:POKE 755,2:POKE 752,1:A=0
   :GOTO 13410
```

Listing 1 of *Secret Agent* in the June '89 issue was also jumbled when laid out. In addition, the 38th character of each full-length line was duplicated on the line below it. To type the listing (using M/L Editor), you must type each of the lines in order, which means you'll have to jump back and forth in the listing, finding where the lines actually continue. For example, at Line 4760 the program listing continues with Line 3920. You must look ahead until you find Line 4770 and continue typing there. Due to the jumbling, Lines 3490 and 4340 were divided in the magazine. Type those lines as shown below:

```
3490 DATA 232,200,192,5,208,242,230,197
,134,190,96,189,0,132,217,214,5353
4340 DATA 67,76,118,55,162,161,160,115,
32,143,49,173,10,210,201,180,2247
```

We apologize for any inconvenience this mix-up may have caused.

Picture Perfect

by Joe D. Brzuszek

Few commercial picture-editing programs offer cut and paste (or stamp) utilities that perform satisfactorily. One software package, for example, features a "rubber stamp" utility that is supposed to let the user stamp a graphics image "anywhere" on his picture. But the program does not allow the image to be moved horizontally by less than four pixels. *Picture Perfect* is a versatile, menu-driven picture-editing program that corrects this annoying problem.

Picture Perfect allows you to quickly duplicate an image at any pixel location (one pixel accuracy). You can also manipulate the image in the following ways: You can flip it, double or halve its size, fill all of its colors at once or transfer it to another picture. You can even bring a series of screen images to life with the included animator routine.

File Format

The program loads *MicroPainter* (MP) 62-sector picture files. If you're using *MicroIllustrator* (MI) software (*Atari Artist* or *Koala Pad*), you can save a picture in MP format under the filename PICTURE on Drive 1 by pressing the Insert key. The color data, however, is not saved. To load PICTURE with MI, press Clear.

Typing It In

Type in Listing 1, using BASIC Editor II to check your work, and save it to a DOS disk as PICTPERF.BAS. Next type Listing 2 and run it. It will create a file on your disk called PICT.LST. Once this file has been created, load the program you created from Listing 1, then type ENTER"D:PICT.LST" to merge the lines created by Listing 2. Now save the complete program as PICTPERF.BAS.

Plug a joystick into Port 1 and type RUN "D:PICTPERF.BAS". The title screen will appear after a short initialization delay. Press Start to view the main menu.

Loading a Picture

Select the "Disk" function from the main menu by placing the cursor over the Disk box and pressing the trigger or the Return key. A set of common disk commands will be displayed. At the "Select item or return:" prompt, select "F. Load File" and enter a filename. If the file is a valid MP file, the picture and its colors will be loaded without any problem.

If the file is PICTURE saved by MI, the colors will not be loaded and an Error-136 (End of File) will occur. That's because the color data occupies the last four bytes in a MP file, and for some strange reason, MI does not save them. You can load the color data out of a compacted MI file; this time type a Control-P at the end of the filename. Type "B" and then Return to begin editing your picture.

The Copy Function

Copy: Copy is accessed by pressing the space bar, and it's divided into "frame" and "ditto" modes. You start out in ditto mode. Press the space bar to enter frame mode (the console or TV speaker clicks twice).

Frame: Move the cursor to the upper-left corner of the image and tap the trigger. Press the space bar to try again, if necessary, then move the cursor to the lower-right corner. A flashing frame surrounds the image. The maximum area that you can frame is ½ the size of the picture (160 columns x 96 rows). Press "U" to cycle the frame's color (Colors 0-3). When the frame surrounds everything that you want to copy, press the trigger to activate ditto mode.

Ditto: Move the cursor and the image will follow. Press the trigger to make a copy of the image wherever you'd like. Press the space bar once to return to frame mode, or twice to return to the main menu.

Main Menu Functions

Choose a function and its submenu will appear. When you have made your selections and you're ready to execute the function, move the cursor over "OK" and press the trigger. If you decide not to execute the function, press the space bar instead to return to the main menu.

Mode: Switches the display between graphics modes 14 (also known as mode 7.5 or Antic $E) and 8.

Fill: This simultaneously fills Colors 0-3 of an image with replacement colors. Choose the replacement colors and then select "OK" to fill. Ditto is activated once the image is filled. Fill is useful for preparing a highly detailed picture for printer output.

Flip: "H" flips an image horizontally, and "V" flips it vertically. A Graphics 8 image, flipped horizontally, might look strange unless you also use Fill to switch Colors 1 and 2.

Halve and Double: You can halve or double the size of an image horizontally or vertically. The height of a doubled image cannot exceed 96 rows, so it might get cropped. A similar danger exists whenever you halve an image; if you double an image after halving it, the resulting image may not resemble the original.

Animate: You can animate a series of up to 50 on-screen images. Choose to create a "New" series or to "Append" new images to an existing series. Frame each one and then press the space bar. Select "Play," and the series will be played in the upper-left corner of the screen at the fastest speed. Press Option to slow the speed. (There are six speeds available.) To get maximum speed, change Line 220 to: 220 NEXT Y. Press the space bar to halt the animation.

Cel: "Cel" refers to the piece of transparent celluloid on which a cartoon character is painted. You can make any single color (0-3) or no color (N) in an image transparent to the background drawing.

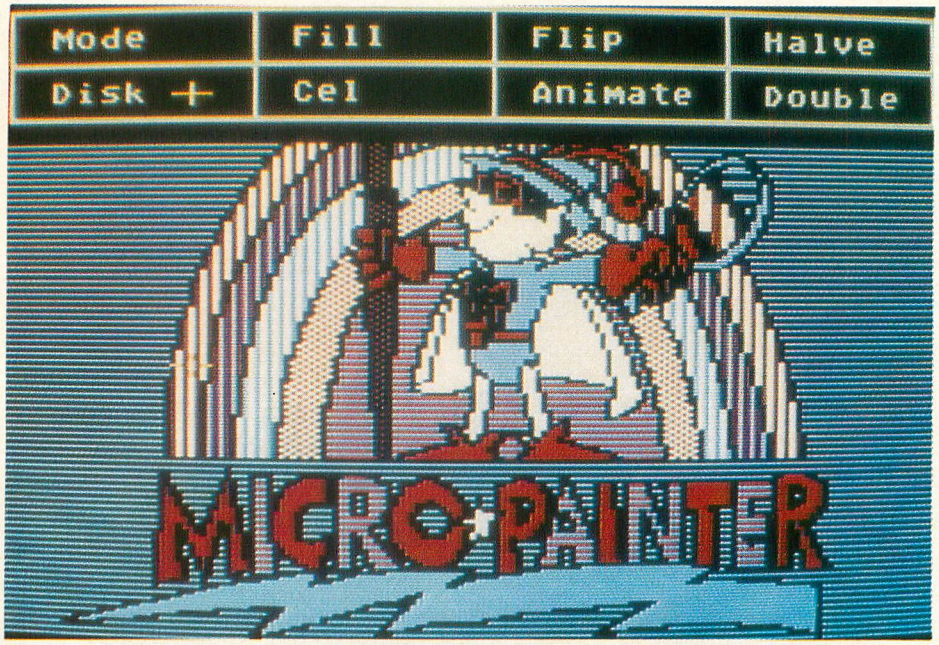
USR Information

The program makes use of a non-relocatable machine-language Copy routine (see Listing 3) stored in Memory Locations 23253-23539 ($5AD5-$5BF3). Copy makes it possible to move image data with one pixel accuracy in graphics mode 14. Copy will also work in any four-color graphics mode in which a single pixel requires two bits of memory (Graphics 3, 5, and 7). The USR call format is: A=USR(CY,S1,X1,Y1,X2,Y2,S2,X3,Y3,C2).

CY equals 23253, S1 is the starting address (corresponding to the upper-left corner of the screen) of memory from which the image is to be copied, X1 and Y1 are the upper-left coordinates of the image, X2 and Y2 are the lower-right coordinates of the image, S2 is the starting address of memory to which the image is to be copied, and X3 and Y3 are the upper-left coordinates of the new location of the image. The value of C2 determines which color to make transparent: 0 = Color 0, 64 = Color 1, 128 = Color 2, 192 = Color 3, 255 = no transparent color, 3 = fast copy, four-pixel resolution and no transparent color.

Cursor Control

Picture Perfect gives you control over the cursor's shape and speed. Pressing "I" changes its shape, and pressing "O" changes its speed. Two shapes are provided and this data is in Line 1610.



Last but Not Least

The Break key is disabled in Line 8000 and pressing System Reset will not erase your picture or colors but will wipe out the image in Copy memory. If you press Reset by mistake, just type "RUN" to continue working on your picture. *(Program on page 44)*

Joe Brzuszek is majoring in computer science at the University of Pittsburgh and has owned his Atari 800 since 1983. In addition to programming, Joe uses his Atari as a VT-100 terminal to communicate with a VAX main-frame computer system.

Capital!

by Bryan Schappel and Barry Kolbe

For all of you out there who have always wanted to own it all, here is your chance. Get your checkbook, grab your accountant and get ready for *Capital!*, a game of high finance where you get to live out your fondest capitalistic dreams. This game is written in 100% machine language; so play is fast and cruel.

Typing It In

Capital! is printed in two listings. Listing 1 is the BASIC data used to create the CAPITAL.OBJ file on your disk. Follow the directions in *M/L Editor*, found elsewhere in this issue, for typing instructions.

Listings 2 through 7 are the complete commented MAC/65 source code for *Capital!* They need not be typed in to play the game; they are provided for those readers interested in how the program works or who are learning 6502 assembly.

After you have created the CAPITAL.OBJ file just binary load it from DOS to play the game.

Rules of Play

To play *Capital!*, you must first choose your game options. There are only two: a fast or slow game and the number of players (2-4). The Option key is used to toggle between fast and slow. Use the Select key to choose the number of players. Pressing Start begins the game. Any time you wish to restart the game, press System Reset.

A fast game is the default setting. In a fast game, if a player cannot pay a fee or a debt, that person is "broke" and may no longer play. The player's token is removed from the game, and the player goes to debtor's prison. In a slow game a player must sell businesses in order to get enough cash to pay his debts. Of course, the player could still go broke in a slow game. The winner is the last player on the board.

One last item: All money in this game is in $1,000 denominations, denoted by a G following the amount.

Names Screen

Each player must enter his name (up to eight characters long). Only the letters A through Z may be used. The Return key is used to end the name, and Delete/Backspace is used to edit.

USA Map Screen

This is the game board: a map of the U.S. showing mountain ranges, dice and little squares and circles which represent businesses. Your task is to travel around the U.S. and buy up as many of these as possible, thus making as much money as you can and becoming a true capitalist. Near the bottom of the screen the current player's name is shown with his token number. You can see the tokens below the dice when the game is first started. The bottom line shows how much cash each player has.

Press Option to start the dice rolling and use Select to stop them. Your token will move automatically around the screen. The name of each business is shown as your token moves by or lands on it. If you land on one of these, the game will take you to the transaction screen (see below). The game waits eight seconds so you can see what happened. If you tire of waiting those eight seconds, just press a key to skip the wait.

THERE ARE SEVERAL SPECIFIC LOCATIONS:

Symbol	Location	Result
$	Stock Market	You receive $12G ($12,000).
L	Luck Square	Game goes to the transaction screen.
H	Hog Hilton	Nothing; it's a vacation.
T	Tax Square	You lose 12% of your cash.
C	Capital Gains Tax	Game goes to the transaction screen.

If you land on $, H or T, you are not allowed to carry on any transactions on that turn.

Transaction Screen

The Luck and Capital Gains squares will be discussed at the end of this section. Usual-

ly you arrive at this screen because you landed on a property. The top line has the property's name on it. Below the name is information about the business: purchase price, resale price, improvement level, double status, fee and the owner's name.

Resale price is the amount you will receive if you sell this property voluntarily. It is between half price and full price. If you are forced to sell a property (slow game) because of insufficient funds, the resale price is half of the purchase price.

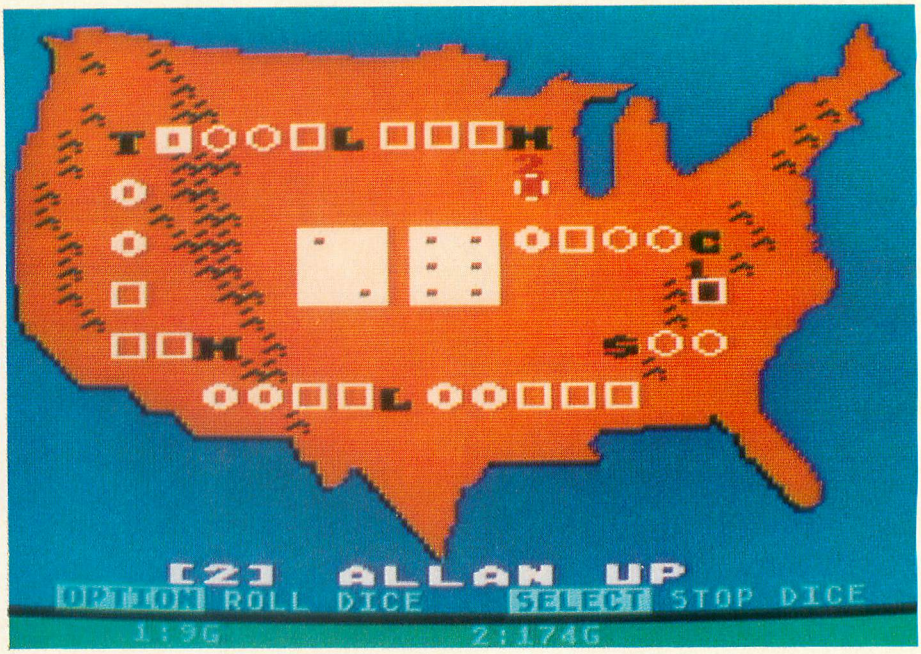
The improvement level of each property starts at 0 and may go up to 3. Improving a business costs $10G and increases the purchase price (and hence resale price) by $5G. More importantly, each increase doubles the fee other players must pay you when they land on this business.

Some businesses are "doubles." If you land on a double, you have the option of buying it and its other half on the same turn. Buying the second half costs one and one-half of the double's normal purchase price. The reason you want to get doubles is that anyone landing on one double gets charged a fee equal to the sum of both fees!

If the property you landed on is unowned you may purchase it, assuming you have enough cash. Next you may sell a property. Finally, you may improve a property. If you respond "y" to either of the last two options, the game goes to the pick property screen. You may cycle through the properties using the space bar. Press Return to choose a property. You may exit using the Escape key. The only time you cannot use the Escape key is in a slow game if you are forced to sell. You must sell enough properties to obtain the cash to pay your debts. You may sell and/or improve any property you own, not just the one you landed on.

If another player owns the property you landed on, you are required to buy goods and/or services by paying the player a fee. If you have enough cash, it is automatically deducted from your account. The message "Transaction completed" lets you know this has been done. In the fast game, if you don't have sufficient funds to pay, you are broke and may no longer play. Your properties are confiscated and will be available for sale to the other players (at half price!).

In the slow game you must sell properties to pay your debts. It is possible to get two "Transaction completed" messages. The first would be for selling a property; the second would be for paying a fee. This message stays on the screen for eight seconds. You may skip the wait by pressing any key.



The Luck Square

Luck, as we all know, can be good or bad. If your luck is good, you could win $10G, $15G, or a free improvement for a property. If you don't have a property to improve, you will get $10G instead. Bad luck results in losing $10G, in one business fee being cut in half, or in having the purchase price (and thus resale price) on a business cut in half.

Capital Gains Tax

This is like bad luck—only worse. If you land on this square you lose $45G.

Technical Notes

The program functions on a custom GRAPHICS 0 narrow playfield screen, an ANTIC 4 map screen and a unique intro screen. This intro screen was produced by using a very long DLI. The word "CAPITAL!" was made from ordinary control characters. Examine the source code to see how the letters were shaded.

This game can make friends or enemies for you—hope you can be a good capitalist!

Barry Kolbe is a high school math teacher who uses his Atari in the classroom to demonstrate graphing. Bryan Schappel is currently setting up a new home with his wife Carol.

LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,0,48,2,48,76,148,57,
64,49,85,75,0,0,0,7658
1010 DATA 0,0,0,0,0,253,245,215,215,21
5,215,245,253,127,95,215,5285
1020 DATA 215,215,215,95,127,253,247,2
23,223,223,223,247,253,127,223,247,374
1030 DATA 247,247,247,223,127,213,213,
215,215,215,215,213,213,87,87,215,6266
1040 DATA 215,215,215,87,87,255,250,23
4,235,235,235,234,250,255,175,175,368
1050 DATA 239,255,239,175,175,85,85,85
,85,85,85,85,85,85,85,85,4376
1060 DATA 105,105,85,85,85,255,234,238
,254,254,254,254,250,255,171,187,512
1070 DATA 191,191,191,191,175,255,171,
235,234,234,235,235,171,255,234,235,13
76
1080 DATA 171,171,235,235,234,255,254,
234,235,234,255,234,254,255,191,171,21
29
1090 DATA 255,171,235,171,191,213,223,
223,223,223,223,223,213,87,247,247,966
4
1100 DATA 247,247,247,247,87,255,234,2
35,235,235,235,234,234,255,255,255,342
8
```

```
1110 DATA 255,255,239,175,175,95,95,95
,95,95,95,95,95,245,245,245,2412
1120 DATA 245,245,245,245,245,239,191,
255,251,238,191,254,255,255,255,255,41
84
1130 DATA 255,255,255,255,255,0,255,25
5,255,255,255,255,0,0,255,6885
1140 DATA 255,255,255,255,255,0,0,0,25
5,255,255,255,255,0,0,0,8990
1150 DATA 0,255,255,255,255,0,0,0,0,0,
255,255,255,0,0,0,3900
1160 DATA 0,0,0,255,255,0,0,0,0,0,0,0,
255,192,192,240,6178
1170 DATA 240,252,252,255,255,240,240,
240,240,252,252,252,252,3,3,15,4084
1180 DATA 15,63,63,255,255,15,15,15,15
,63,63,63,63,255,255,252,8580
1190 DATA 252,252,240,240,192,252,252,
252,252,240,240,240,240,255,255,191,36
37
1200 DATA 191,175,47,43,11,191,191,191
,191,47,47,47,47,255,255,255,1476
1210 DATA 255,255,255,255,170,255,255,
255,255,255,255,170,170,255,255,255,33
40
1220 DATA 255,255,170,170,0,191,191,19
1,191,191,191,191,191,168,168,0,2563
1230 DATA 0,0,0,0,0,254,250,240,240,19
2,192,192,192,207,207,207,6731
1240 DATA 207,207,207,63,63,63,63,63,6
3,255,255,255,255,255,255,255,8144
1250 DATA 255,63,63,63,63,255,255,255,
255,255,255,63,15,240,240,240,7143
1260 DATA 240,240,192,192,0,252,252,25
2,252,252,252,252,252,3,3,3,2611
1270 DATA 3,15,15,15,15,63,63,63,63,25
5,255,255,255,192,192,192,3743
1280 DATA 192,240,240,252,255,3,3,3,3,
15,15,63,255,192,192,192,8071
1290 DATA 192,192,192,192,192,3,3,3,3,
3,3,3,3,0,0,0,4398
1300 DATA 3,3,15,15,63,240,252,255,255
,252,255,252,252,240,252,252,2065
1310 DATA 252,252,252,252,240,252,240,
192,192,0,0,0,0,252,240,240,2454
1320 DATA 240,192,192,192,192,192,192,
192,192,240,240,252,252,192,192,240,75
6
1330 DATA 240,240,252,252,252,252,252,
252,240,240,240,192,192,252,240,192,25
66
1340 DATA 240,252,252,255,255,192,0,0,
0,192,192,192,192,11,11,11,5614
1350 DATA 11,11,11,15,15,15,15,15,63,6
3,63,255,255,0,0,3,179
1360 DATA 3,3,15,15,63,0,48,188,191,47
,47,11,11,15,15,15,7285
1370 DATA 207,255,255,255,255,12,60,24
0,255,255,255,255,255,0,0,0,1584
1380 DATA 0,252,240,0,0,255,255,255,25
5,255,240,192,192,240,192,0,6484
1390 DATA 0,0,0,0,0,255,255,255,255,25
5,255,255,252,255,191,191,222
1400 DATA 191,175,47,43,10,63,63,63,63
,63,63,255,255,252,252,240,3040
1410 DATA 240,252,252,255,255,0,0,0,0,
3,15,63,255,0,3,15,9756
1420 DATA 63,255,255,255,255,0,0,0,0,3
,15,15,63,240,240,240,7047
1430 DATA 240,192,192,192,192,170,170,
0,0,0,0,0,0,10,10,0,6858
1440 DATA 0,0,0,0,0,2,2,0,0,0,0,0,0,0,
15,14,1915
1450 DATA 8,25,21,0,12,20,22,28,0,4,6,
15,7,5,0,5,2656
1460 DATA 8,14,18,0,7,19,3,0,16,15,21,
19,14,0,31,12,3312
1470 DATA 10,0,23,37,25,69,65,0,41,60,
73,84,0,12,15,38,6472
1480 DATA 17,13,0,18,27,36,37,0,22,51,
11,0,53,48,56,61,6211
1490 DATA 32,0,100,47,42,128,0,0,0,2,2
,4,0,0,2,2,3140
1500 DATA 128,0,0,0,2,2,128,0,2,2,0,4,
0,0,0,128,4680
1510 DATA 2,2,0,2,2,4,0,2,2,119,1,119,
1,119,1,119,7819
1520 DATA 1,0,11,17,26,0,9,18,27,0,0,0
,0,5,4,0,2332
1530 DATA 0,0,10,9,0,0,0,0,16,15,0,0,2
0,19,0,0,2416
1540 DATA 0,0,0,0,28,27,0,31,30,0,0,35
,34,15,13,10,3787
1550 DATA 5,3,1,112,112,112,70,67,53,6
,48,98,0,32,0,34,7503
1560 DATA 34,34,34,34,34,34,34,34,34,3
4,34,34,34,34,34,34,6184
1570 DATA 2,176,194,227,48,32,70,99,53
,65,166,52,112,112,112,112,5201
1580 DATA 112,112,70,119,53,112,112,70
,0,32,112,6,112,6,112,6,9823
1590 DATA 65,207,52,112,112,112,240,66
,139,53,2,2,2,2,48,6,8810
1600 DATA 112,6,112,6,48,86,143,54,48,
86,164,54,112,112,70,185,5051
1610 DATA 54,112,6,112,112,112,6,86,24
5,54,65,230,52,112,112,80,5740
1620 DATA 68,0,112,4,4,4,4,4,4,4,4,4,4
,4,4,4,2544
1630 DATA 4,4,4,4,4,4,132,70,99,53,194
,11,55,160,66,227,4462
1640 DATA 48,65,16,53,64,0,32,96,32,64
,0,96,0,0,0,0,5470
1650 DATA 0,0,0,0,35,33,48,41,52,33,44
,1,0,0,0,0,3981
1660 DATA 0,0,244,242,225,238,243,225,
227,244,233,239,238,243,0,0,5824
1670 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,1670
1680 DATA 0,0,0,0,0,0,101,110,116,101,
114,0,121,111,117,114,3281
1690 DATA 0,110,97,109,101,115,0,0,0,0
,0,72,128,128,74,0,9262
1700 DATA 72,128,128,74,0,72,128,128,7
4,0,72,128,74,0,72,128,2144
1710 DATA 74,0,72,128,128,74,0,74,0,0,
0,0,72,0,0,0,5124
1720 DATA 0,0,0,128,200,0,0,0,128,200,
202,128,0,128,200,202,8166
1730 DATA 128,0,0,128,0,0,0,128,0,0,12
8,200,202,128,0,128,3668
1740 DATA 0,0,0,0,128,0,0,0,0,0,0,128,
0,0,0,0,3916
1750 DATA 128,82,82,128,0,128,128,128,
200,0,0,128,0,0,0,128,872
1760 DATA 0,0,128,82,82,128,0,128,0,0,
0,0,128,0,0,0,6338
1770 DATA 0,0,0,128,74,0,0,0,128,0,0,1
28,0,128,0,0,7132
1780 DATA 0,0,0,128,0,0,0,128,0,0,128,
0,0,128,0,128,8564
1790 DATA 74,0,0,0,200,0,0,0,0,0,0,202
,128,128,200,0,1744
1800 DATA 200,0,0,202,0,200,0,0,0,0,20
2,128,200,0,0,128,2414
1810 DATA 0,0,200,0,0,202,0,202,128,12
8,200,0,79,0,0,0,897
1820 DATA 231,225,237,229,0,239,230,0,
232,233,231,232,0,230,233,238,7438
1830 DATA 225,238,227,229,0,0,0,112,11
4,111,103,114,97,109,109,101,5699
1840 DATA 100,0,98,121,90,0,0,0,0,0,0,
162,178,185,161,174,5215
1850 DATA 0,179,163,168,161,176,176,16
5,172,0,0,0,0,0,0,0,9330
1860 DATA 0,0,0,0,0,0,161,174,164,0,0,
```

```
0,0,0,0,0,5855
1870 DATA 0,0,0,0,0,0,0,162,161,178,17
8,185,0,171,175,172,8344
1880 DATA 162,165,0,0,0,0,0,239,240,24
4,233,239,238,0,0,0,7409
1890 DATA 102,97,115,116,0,103,97,109,
101,0,0,51,37,44,37,35,8897
1900 DATA 52,0,0,0,146,0,176,172,161,1
85,165,178,179,0,0,0,4867
1910 DATA 0,35,47,48,57,50,41,39,40,52
,0,17,25,24,24,0,5602
1920 DATA 0,0,0,0,0,0,98,98,107,0,101,
110,116,101,114,112,3208
1930 DATA 114,105,115,101,115,0,0,0,0,
66,175,176,180,169,175,174,8390
1940 DATA 86,50,47,44,44,0,36,41,35,37
,0,0,0,66,179,165,177
1950 DATA 172,165,163,180,86,51,52,47,
48,0,36,41,35,37,0,0,7430
1960 DATA 0,0,51,116,111,99,107,0,45,9
7,114,107,101,116,0,36,1901
1970 DATA 105,118,105,100,101,110,100,
115,0,112,97,121,0,121,111,117,4681
1980 DATA 0,4,17,18,39,0,0,0,0,0,41,50
,51,0,52,97,6352
1990 DATA 120,0,33,117,100,105,116,26,
0,108,111,115,101,0,17,18,364
2000 DATA 5,0,111,102,0,121,111,117,11
4,0,99,97,115,104,0,0,1415
2010 DATA 0,0,0,0,54,97,99,97,116,105,
111,110,0,116,105,109,3909
2020 DATA 101,0,97,116,0,116,104,101,0
,40,111,103,0,40,105,108,1828
2030 DATA 116,111,110,0,0,0,0,0,102,97
,115,116,115,108,111,119,3819
2040 DATA 66,82,79,75,69,0,125,32,32,8
0,114,111,112,58,155,32,3062
2050 DATA 80,114,105,99,101,58,32,32,3
2,32,32,32,32,32,32,82,8402
2060 DATA 101,115,97,108,101,58,155,32
,76,101,118,101,108,58,32,32,2720
2070 DATA 32,32,32,32,32,32,32,68,111,
117,98,108,101,58,155,32,3015
2080 DATA 32,32,70,101,101,58,32,32,32
,32,32,32,32,32,32,32,7323
2090 DATA 79,119,110,101,114,58,0,89,1
01,115,0,78,111,0,78,111,2155
2100 DATA 110,101,0,66,10,2,212,0,148,
6,0,66,0,196,150,10,1560
2110 DATA 15,0,32,104,97,115,32,119,11
1,110,33,155,80,114,101,115,5301
2120 DATA 115,32,97,32,107,101,121,32,
116,111,32,112,108,97,121,32,3901
2130 DATA 97,103,97,105,110,46,0,32,10
5,115,32,66,114,111,107,101,3722
2140 DATA 33,155,0,32,112,114,111,112,
101,114,116,121,32,91,89,47,4082
2150 DATA 78,93,63,0,155,155,73,109,11
2,114,111,118,101,32,97,0,3692
2160 DATA 155,155,83,101,108,108,32,97
,0,155,155,66,117,121,32,116,5064
2170 DATA 104,105,115,0,155,155,83,80,
67,61,110,101,120,116,44,32,3746
2180 DATA 82,69,84,61,99,104,111,111,1
15,101,0,44,32,69,83,67,1952
2190 DATA 61,101,120,105,116,0,155,155
,84,114,97,110,115,97,99,116,6615
2200 DATA 105,111,110,32,112,114,111,9
9,101,115,115,101,100,46,0,155,4758
2210 DATA 155,73,110,115,117,102,102,1
05,99,105,101,110,116,32,102,117,5782
2220 DATA 110,100,115,46,0,155,155,68,
111,117,98,108,101,32,79,112,4899
2230 DATA 116,105,111,110,46,0,155,155
,65,116,32,77,97,120,105,109,5165
2240 DATA 117,109,32,76,101,118,101,10
8,0,155,155,70,101,101,32,80,4341
2250 DATA 97,121,109,101,110,116,32,82
,101,113,117,105,114,101,100,46,5164
2260 DATA 0,155,155,65,109,111,117,110
,116,32,111,119,101,100,32,0,3411
2270 DATA 169,62,141,47,2,169,1,141,11
1,2,162,3,169,0,133,159,4906
2280 DATA 157,8,208,202,16,250,169,3,1
41,29,208,169,128,141,7,212,9682
2290 DATA 162,3,169,120,149,145,189,11
1,57,149,141,202,16,244,162,3,9351
2300 DATA 189,250,61,157,192,2,169,0,1
49,149,202,16,243,169,0,133,8853
2310 DATA 153,169,62,141,47,2,169,1,14
1,111,2,96,96,104,112,120,4846
2320 DATA 173,63,53,240,3,32,238,58,17
3,64,53,240,3,32,4,59,3226
2330 DATA 173,65,53,240,3,32,26,59,173
,66,53,240,3,32,48,59,2444
2340 DATA 96,216,162,255,154,32,7,75,3
2,132,60,32,41,72,32,35,1218
2350 DATA 57,32,211,58,32,200,70,32,22
7,60,32,70,62,169,16,141,4945
2360 DATA 48,2,169,53,141,49,2,169,48,
141,244,2,32,100,57,162,5309
2370 DATA 4,189,245,61,157,196,2,202,1
6,247,169,0,133,159,133,173,513
2380 DATA 169,142,141,0,2,169,59,141,1
,2,32,115,57,166,153,189,5966
2390 DATA 63,53,208,3,76,200,58,160,0,
189,120,52,133,130,169,40,7019
2400 DATA 133,131,177,130,240,3,200,20
8,249,132,174,169,20,56,229,174,2894
2410 DATA 74,170,160,19,169,0,153,99,5
3,136,16,250,200,169,59,157,9464
2420 DATA 95,53,169,61,157,97,53,165,1
53,24,105,17,157,96,53,177,6418
2430 DATA 130,240,10,32,70,71,157,99,5
3,232,200,208,242,169,53,157,2177
2440 DATA 100,53,169,48,157,101,53,173
,31,208,201,3,208,249,32,205,1047
2450 DATA 59,165,153,10,170,189,126,59
,141,99,58,189,127,59,141,100,8033
2460 DATA 58,189,134,59,141,129,58,189
,135,59,141,130,58,198,157,32,8240
2470 DATA 70,59,166,153,246,149,181,14
9,201,36,208,4,169,0,149,149,9672
2480 DATA 181,149,168,185,173,61,149,1
41,185,209,61,149,145,32,238,58,650
2490 DATA 32,146,60,166,153,181,149,20
8,12,162,0,32,168,60,169,12,6079
2500 DATA 164,153,32,155,70,32,86,60,3
2,155,62,165,157,208,190,166,269
2510 DATA 153,181,149,160,3,217,116,52
,240,12,136,16,248,32,97,60,6712
2520 DATA 32,211,58,32,250,72,32,227,6
0,166,153,181,149,240,3,32,8907
2530 DATA 238,60,32,70,62,230,153,165,
153,41,3,133,153,76,176,57,7366
2540 DATA 162,0,138,157,0,132,157,0,13
3,157,0,134,157,0,135,232,7788
2550 DATA 208,241,162,3,157,0,208,136,
16,250,96,162,0,164,145,189,206
2560 DATA 121,61,153,0,132,200,232,224
,13,208,244,165,141,141,0,208,2534
2570 DATA 96,162,0,164,146,189,134,61,
153,0,133,200,232,224,13,208,1851
2580 DATA 244,165,142,141,1,208,96,162
,0,164,147,189,147,61,153,0,7950
2590 DATA 134,200,232,224,13,208,244,1
65,143,141,2,208,96,162,0,164,412
2600 DATA 148,189,160,61,153,0,135,200
,232,224,13,208,244,165,144,141,4025
2610 DATA 3,208,96,164,145,169,0,162,1
2,153,0,132,209,202,16,249,9882
2620 DATA 96,164,146,169,0,162,12,153,
0,133,200,202,16,249,96,164,150
2630 DATA 147,169,0,162,12,153,0,134,2
00,202,16,249,96,164,148,169,1265
2640 DATA 0,162,12,153,0,135,200,202,1
```

```
6,249,96,70,59,84,59,98,6364
2650 DATA 59,112,59,238,58,4,59,26,59,
48,59,72,165,159,208,27,5444
2660 DATA 169,36,141,9,212,141,10,212,
169,130,141,24,208,169,218,141,2288
2670 DATA 23,208,169,230,141,25,208,23
0,159,104,64,201,1,208,17,169,158
2680 DATA 162,141,10,212,141,24,208,16
9,0,141,26,208,230,159,104,64,9651
2690 DATA 169,162,141,10,212,141,26,20
8,104,64,169,0,133,157,32,55,6120
2700 DATA 60,162,0,160,0,177,155,157,7
7,113,232,200,192,3,208,245,3480
2710 DATA 162,0,177,155,157,117,113,23
2,200,192,6,208,245,162,0,177,2724
2720 DATA 155,157,157,113,232,200,192,
9,208,245,32,55,60,160,0,162,8834
2730 DATA 5,177,155,157,77,113,232,200
,192,3,208,245,162,7,177,155,2794
2740 DATA 157,115,113,232,200,192,6,20
8,245,162,5,177,155,157,157,113,2632
2750 DATA 232,200,192,9,208,245,141,31
,208,32,86,60,173,31,208,201,616
2760 DATA 5,208,151,96,173,10,210,41,7
,201,6,176,247,170,24,101,8559
2770 DATA 157,133,157,230,157,138,10,1
70,189,55,61,133,155,189,56,61,8622
2780 DATA 133,156,96,169,0,133,20,165,
20,201,6,144,250,96,162,255,1535
2790 DATA 142,252,2,173,252,2,201,255,
208,244,169,0,133,20,133,19,9332
2800 DATA 173,252,2,201,255,208,6,165,
19,201,2,144,243,142,252,2,1062
2810 DATA 96,162,7,169,0,157,0,210,202
,16,250,141,8,210,96,169,157
2820 DATA 133,141,0,210,169,168,141,1,
210,32,86,60,169,0,141,0,5111
2830 DATA 210,141,1,210,96,189,224,60,
72,133,158,32,227,60,238,46,24
2840 DATA 53,208,3,238,47,53,198,158,3
2,25,72,165,158,208,239,32,9846
2850 DATA 97,60,104,133,158,206,46,53,
32,25,72,173,46,53,201,255,8524
2860 DATA 208,3,206,47,53,198,158,208,
236,32,227,60,96,40,80,120,8692
2870 DATA 169,11,141,46,53,169,55,141,
47,53,96,165,153,170,10,168,7656
2880 DATA 181,149,162,1,201,17,208,47,
185,183,48,133,155,185,184,48,540
2890 DATA 133,156,70,156,102,155,70,15
6,102,155,70,156,102,155,185,183,1656
2900 DATA 48,56,229,155,153,183,48,185
,184,48,229,156,153,184,48,32,370
2910 DATA 200,70,162,1,76,52,61,232,20
1,11,240,5,201,26,240,1,7927
2920 DATA 96,76,168,60,67,61,76,61,85,
61,94,61,103,61,112,61,3623
2930 DATA 49,49,49,49,50,49,49,49,49,5
0,49,49,49,49,49,49,9609
2940 DATA 49,50,50,49,49,49,50,49,49,4
9,50,50,49,50,49,49,9653
2950 DATA 49,50,49,50,50,49,50,49,50,4
9,50,49,50,50,49,50,9695
2960 DATA 50,49,50,50,49,50,96,224,96,
96,96,240,0,60,60,60,4927
2970 DATA 60,60,60,56,108,12,24,48,124
,0,60,60,60,60,60,60,694
2980 DATA 126,12,24,12,102,60,0,60,60,
60,60,60,60,27,27,31,9179
2990 DATA 31,3,3,0,60,60,60,60,60,60,1
52,152,144,136,128,120,6848
3000 DATA 112,104,96,88,80,80,72,64,64
,64,64,64,72,80,88,96,3456
3010 DATA 104,112,120,128,136,136,136,
144,152,160,168,168,168,160,120,136,30
42
3020 DATA 136,136,136,136,136,136,136,
136,136,120,120,120,104,88,72,56,7660
3030 DATA 56,56,56,56,56,56,56,56,56,7
2,88,88,88,88,88,104,3654
3040 DATA 120,120,78,160,244,224,130,2
,66,122,198,250,74,72,70,68,8864
3050 DATA 66,64,62,60,58,56,232,230,22
8,148,68,244,164,166,168,170,4664
3060 DATA 172,174,176,178,180,182,6,86
,88,90,92,94,174,254,252,113,2780
3070 DATA 114,114,114,114,114,114,114,
114,114,114,113,113,113,113,113,112,84
77
3080 DATA 112,112,112,112,112,112,112,
112,112,112,113,113,113,113,113,113,83
93
3090 DATA 113,113,113,162,35,189,254,6
1,133,155,189,34,62,133,156,189,1257
3100 DATA 147,48,48,13,208,15,160,0,16
9,57,145,155,200,169,58,145,9491
3110 DATA 155,202,16,225,96,201,1,208,
13,169,45,160,0,145,155,200,9751
3120 DATA 169,46,145,155,208,235,201,2
,208,13,160,0,169,43,145,155,9525
3130 DATA 200,169,44,145,155,208,218,2
01,3,208,214,160,0,169,41,145,1219
3140 DATA 155,200,169,42,145,155,208,2
01,32,203,62,169,11,141,46,53,7772
3150 DATA 169,49,141,47,53,166,153,181
,149,10,170,189,235,64,133,155,1813
3160 DATA 189,236,64,133,156,160,0,162
,10,177,155,240,10,32,70,71,6790
3170 DATA 157,11,49,200,232,208,242,96
,162,39,169,0,157,11,49,202,9035
3180 DATA 16,250,96,66,97,114,45,66,45
,81,32,82,97,110,99,104,4761
3190 DATA 0,82,105,110,103,97,100,105,
110,103,32,80,104,111,110,101,6250
3200 DATA 0,67,114,111,115,115,101,121
,101,32,67,97,98,108,101,32,5003
3210 DATA 84,86,0,79,108,100,101,32,89
,111,114,107,101,32,84,105,5035
3220 DATA 109,101,115,0,83,112,117,100
,115,32,80,111,116,97,116,111,6531
3230 DATA 32,70,97,114,109,0,66,108,97
,99,107,32,71,111,108,100,5141
3240 DATA 32,79,105,108,0,82,101,120,3
2,65,117,116,111,32,67,111,4625
3250 DATA 114,112,46,0,84,104,114,101,
100,98,97,114,101,32,67,108,5185
3260 DATA 111,116,104,105,110,103,0,75
,111,108,98,101,39,115,32,67,4141
3270 DATA 104,101,101,115,101,0,68,97,
105,115,121,39,115,32,68,97,4505
3280 DATA 105,114,121,0,66,108,97,99,1
07,32,74,97,99,107,39,115,4896
3290 DATA 32,67,97,115,105,110,111,0,6
8,101,97,116,104,32,86,97,4892
3300 DATA 108,108,101,121,32,83,112,97
,0,65,100,97,109,115,32,65,4090
3310 DATA 112,112,108,101,32,79,114,99
,104,97,114,100,0,83,111,117,5657
3320 DATA 114,32,71,114,97,112,101,115
,32,86,105,110,101,121,97,114,6860
3330 DATA 100,0,83,104,111,114,116,32,
67,105,114,99,117,105,116,32,5740
3340 DATA 80,32,38,32,76,0,72,101,97,1
18,121,32,87,97,116,101,5031
3350 DATA 114,32,67,111,109,112,97,110
,121,0,83,116,121,116,99,104,6689
3360 DATA 39,115,32,67,108,105,110,105
,99,0,83,99,104,97,112,112,5947
3370 DATA 101,108,39,115,32,83,99,97,1
08,112,101,108,115,0,67,114,5214
3380 DATA 111,115,115,116,114,97,120,3
2,82,97,105,108,119,97,121,0,5657
3390 DATA 68,114,105,112,32,68,114,121
,32,67,108,101,97,110,101,114,6281
3400 DATA 115,0,74,101,116,115,116,114
,101,97,109,32,65,105,114,108,6350
```

continued on page 21

Nuclear Mountain

by Brad Timmins

A nuclear-powered satellite just came down at the North Pole. Now it's sitting there, deep in ice, threatening to melt down all the ice and snow. Your job is to destroy it before disaster strikes.

You'll be outfitted with a special high-powered flame thrower to burn through the ice, but your fuel pack is good only for about fifteen burns. However, due to a lucky twist of fate, a supply ship went down in the same area as the satellite. As a result, you'll find many cans of fuel buried in the ice. Unfortunately, the ship also carried Thermite bombs—so be careful. Touch one of these little babies, and you'll be vaporized.

To activate your flame thrower, press the fire button and move the joystick in the direction you want to fire. If you should run out of fuel, and no fuel cans are in sight, you can dig through the ice by hand. To do this, simply press against the ice until it disappears. This method is slow and can be dangerous if you should run into a Thermite charge.

To help you locate the satellite, you'll be given a nuclear tracker, which will point in the general direction of the satellite. To tell when the satellite is about to melt down, watch the yellow bar marked with the word "RADIATION." As the game progresses, the yellow bar will creep across the screen. If it reaches the screen's edge, you will hear a warning tone, and the last location of the bar will beep red. This means that the satellite has reached critical mass. If it's not destroyed within a few moments, it will melt down, and the game will be over.

The satellite has a few weapons of its own. To guard the area, it has created an energy zombie, which will track you flawlessly through the ice. Because it's made of energy, the zombie is capable of burning through the ice when necessary, although this greatly reduces its speed. Touching the zombie means instant death. The only one way to hurt this creature is to lead it to a Thermite charge. If the zombie touches the charge, its energy will dissipate, and the creature will be sent back to its starting position.

If you get to close to the satellite, it will fire an energy ball. If you are hit, you die. At Level 1, the energy balls will only go in your general direction. But at Level 2 and above, they will actually track you until they dissipate. The only way to destroy the satellite is to touch it when its energy is depleted. You can tell how much energy the satellite has by its color. When its color is red, the satellite is fully charged. If you touch it in this state, it will teleport away to a random location. When the satellite's color is yellow, its energy is depleted; touching the satellite in this state will destroy it.

Nuclear Mountain has five levels of difficulty. As the game progress to higher levels, the zombie's speed will increase, the time you have until the meltdown will decrease, and more Thermite charges and less fuel cans will be buried in the ice.

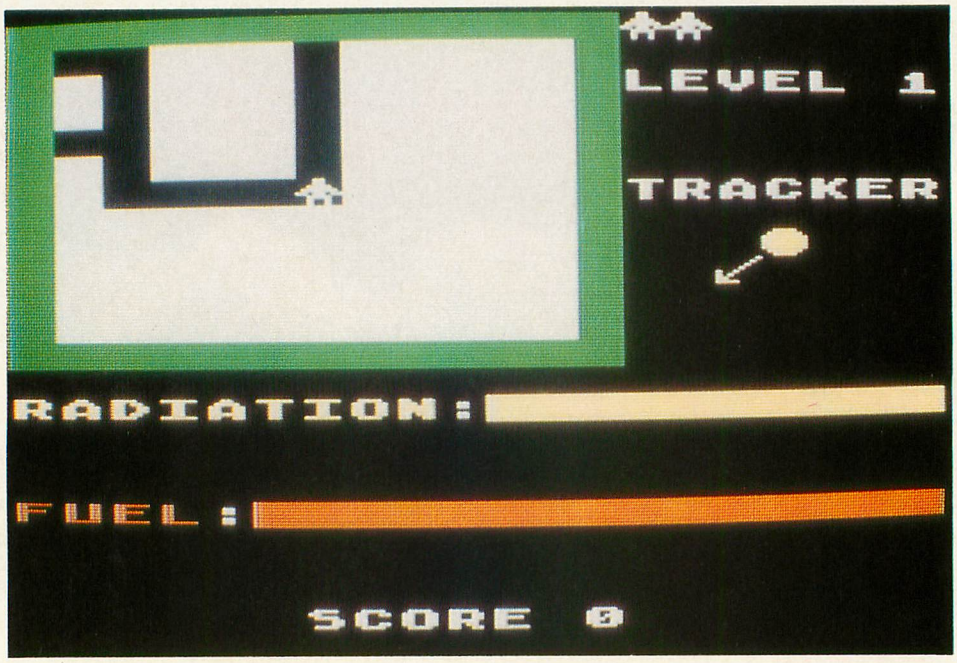
You begin the game with six men. When you reach the fourth level, you gain one extra man. Your score is displayed at the bottom of the screen. Fuel cans are worth 20 points, destroying the zombie by way of a Thermite charge is worth 100 points, and destroying the satellite is worth 500 points.

Good luck and happy burning.

Brad Timmins is a freelance programmer living in Draper, Utah. He has been programming Ataris for five years.


McMAHON

Nuclear Mountain



LISTING 1: BASIC

```
WO 1 REM ******************************
BL 2 REM *      NUCLEAR MOUNTAIN      *
ZS 3 REM *      BY BRAD TIMMINS       *
ZD 4 REM *                            *
BT 5 REM *       COPYRIGHT 1989       *
RX 6 REM *     BY ANALOG COMPUTING    *
WU 7 REM ******************************
QZ 10 GOSUB 2460
EI 20 POKE 106,PEEK(106)-5:GRAPHICS 1+16
VM 30 DIM MAP$(2475),BACK$(2475),SNOW$(55
   ),FONT$(34),DR(14),DZ(14),FUL$(15),G(3
   ,3)
ZY 40 DR(14)=132:DR(13)=134:DR(7)=129:DR(
   11)=131
BN 50 MY=6:MX=20:GU=24:OF=1:ME=6:TI=12:FL
   =19:O1=8:O2=TI:O3=10:LV=1:O4=20
BZ 60 DZ(14)=133:DZ(13)=133:DZ(7)=130:DZ(
   11)=130:M1=20:QC=7:O5=55
LN 70 SCR=PEEK(88)+256*PEEK(89):STR=(PEEK
   (106)+1)*256:ICE=ADR(MAP$):RD=9:M=55
FS 80 SHAD=ADR(BACK$):O7=2:IQ1=O7:FU=70:B
   M=10
KF 90 SETCOLOR 0,0,12:SETCOLOR 2,3,6:SETC
   OLOR 4,0,0:SETCOLOR 1,1,12:SETCOLOR 3,
   12,6
MA 100 FUL$=CHR$(167):FUL$(15)=FUL$:FUL$(
   2)=FUL$
SR 110 GOSUB 2400
OC 120 FOR T=1 TO 34:READ ML:FONT$(T,T)=C
   HR$(ML):NEXT T
BO 130 Q=USR(ADR(FONT$),STR,57344)
DZ 140 FOR A=1536 TO 1594:READ B:POKE A,B
   :NEXT A
MH 150 FOR T=0 TO 119:READ CR:POKE 8+STR+
   T,CR:NEXT T
AR 160 FOR T=0 TO 31:READ CR:POKE 216+STR
   +T,CR:NEXT T
KV 170 FOR T=0 TO 39:READ CR:POKE 472+STR
   +T,CR:NEXT T
ON 180 POKE 756,STR/256
VI 190 GOSUB 1540
IN 200 REM ML CHARACTER SET RELOCATER
HS 210 DATA 104,104,133,204,104,133,203,1
   04,133,206,104,133,205
XD 220 DATA 162,0,160,0,177,205,145,203,2
   00,208,249,230,204,230
QL 230 DATA 206,232,224,4,208,240,96
MZ 240 REM ML WINDOW ROUTINE
YP 250 DATA 104,104,133,204,104,133,203,1
   04,133,206,104,133,205,162,0,160,0,177
   ,203,145,205
HP 260 DATA 200,192,11,208,247,232,24,165
   ,203,105,55,133,203,165,204,105,0,133,
   204,24,165
DN 270 DATA 205,105,20,133,205,165,206,10
   5,0,133,206,224,11,208,214,96,0
DA 280 REM CHARACTER DATA
WZ 290 DATA 0,21,63,127,255,127,63,21
AJ 300 DATA 0,85,255,255,255,255,255,85
QW 310 DATA 0,84,252,254,255,254,252,84
UH 320 DATA 254,124,254,124,254,124,56,16
KT 330 DATA 254,124,254,124,254,124,254,1
   24
SP 340 DATA 16,56,254,124,254,124,254,124
JJ 350 DATA 255,255,255,255,255,255,255,2
   55
CI 360 DATA 24,24,60,126,219,60,36,36
VR 370 DATA 24,60,90,126,66,60,36,102
RD 380 DATA 34,145,154,124,60,90,137,69
AY 390 DATA 0,75,190,162,110,38,46,62
ZQ 400 DATA 60,126,255,255,255,255,126,60
LH 410 DATA 0,0,24,126,60,24,0,0
IK 420 DATA 85,186,87,234,85,186,87,170
QK 430 DATA 36,90,189,189,189,189,90,36
OB 440 DATA 15,3,5,9,16,32,64,128
PJ 450 DATA 240,192,160,144,8,4,2,1
JF 460 DATA 128,64,32,16,9,5,3,15
WV 470 DATA 1,2,4,8,144,160,192,240
EQ 480 DATA 16,56,84,16,16,16,16,16
IT 490 DATA 16,16,16,16,16,84,56,16
CF 500 DATA 0,0,32,64,255,64,32,0
LK 510 DATA 0,0,4,2,255,2,4,0
JF 520 DATA 255,255,255,255,255,255,255,2
   55
YV 530 DATA 28,91,27,93,12,94,30,92,29
QO 540 FOR T=1 TO 3
WY 550 FOR I=1 TO 3:READ CH:G(I,T)=CH:NEX
   T I:NEXT T:WX=+1
LU 560 FOR T=13 TO 18:POKE SCR+T+20*0,8:N
   EXT T
```

```
QC 570 POSITION 0,14:? #6;"radiation:"
RZ 580 POSITION 0,18:? #6;"FUEL:";FUL$
VN 590 POSITION 0,0:? #6;"\\\\\\\\\\\\\\"
WO 600 FOR T=1 TO 11:POSITION 0,T:? #6;"\
            \":NEXT T:POSITION 0,12:? #
   6;"\\\\\\\\\\\\\\"
KH 610 POSITION 16,8:? #6;CHR$(12)
ED 620 QX=6:QY=6:U=PEEK(ICE+MX+55*MY)
IX 630 POSITION 6,22:? #6;"SCORE ";SC
NU 640 POKE 559,AN
TA 650 POSITION 13,2:? #6;"LEVEL ";LV
GL 660 POKE ICE+QX+55*QY,8
YM 670 Q=USR(1536,ADR(MAP$)+QX-5+55*(QY-5
   ),SCR+1+20*1)
HS 680 POSITION 13,6:? #6;"TRACKER"
RI 690 GOTO 950
CG 700 POSITION 12,22:? #6;SC
KP 710 REM MAIN LOOP
PE 720 J=STICK(0):F=STRIG(0):TI=TI-1:GU=G
   U-1:M=M-1:IQ1=IQ1-1
XS 730 IF M=0 THEN GOSUB 1420
NF 740 IF TI=0 THEN GOSUB 1310
QL 750 IF GU=0 AND OF=1 THEN GOSUB 1920
UM 760 IF IQ=1 AND IQ1=0 THEN GOSUB 1840
OJ 770 IF F=0 AND FL>4 THEN 1100
OR 780 IF J=15 THEN 720
DZ 790 IF J=11 OR J=7 THEN RX=8:RY=J:GOTO
    820
LJ 800 IF J=14 OR J=13 THEN RY=13.5:RX=J:
   GOTO 820
OT 810 GOTO 720
JB 820 BX=QX+SGN(RX-J):BY=QY+SGN(RY-J)
BD 830 IF BX>49 OR BX<5 OR BY>39 OR BY<6
   THEN 720
WS 840 IF PEEK(ICE+BX+55*BY)=0 THEN 930
ZN 850 P=PEEK(ICE+BX+55*BY)
DG 860 IF P=63 THEN 1060
UC 870 IF P=77 THEN DES=1:GOSUB 1650
CM 880 IF P=138 THEN 1760
EY 890 IF P=74 THEN 2150
SY 900 IF P=139 THEN 1010
TQ 910 IF P=137 THEN 2020
OW 920 GOTO 720
NA 930 POKE ICE+QX+55*QY,0:QX=BX:QY=BY:DL
   =5
PV 940 SOUND 0,160,10,15:POKE ICE+QX+55*Q
   Y,8
GP 950 POSITION 16+WX,8+WY:? #6;CHR$(32)
RS 960 WX=SGN(IX-QX):WY=SGN(IY-QY)
EK 970 POSITION 16+WX,8+WY:? #6;CHR$(G(WX
   +2,WY+2))
UV 980 Q=USR(1536,ADR(MAP$)+QX-5+55*(QY-5
   ),SCR+1+20*1):SOUND 0,0,0,0
PK 990 GOTO 720
AF 1000 REM REFUEL ROUTINE
SJ 1010 POKE ICE+QX+55*QY,0:POKE SHAD+BX+
   55*BY,0:QX=BX:QY=BY:POKE ICE+QX+55*QY,
   8
WD 1020 Q=USR(1536,ADR(MAP$)+QX-5+55*(QY-
   5),SCR+1+20*1):SC=SC+20:POKE SHAD+QX+5
   5*QY,0
BD 1030 FOR T=255 TO 1 STEP -10:SOUND 3,T
   ,14,15:NEXT T:POSITION 5,18:? #6;FUL$:
   FL=19
WT 1040 SOUND 3,0,0,0:GOTO 700
HC 1050 REM DIG THROUGH ICE ROUTINE
SS 1060 DL=DL-1
KR 1070 IF DL=0 THEN POKE ICE+BX+55*BY,PE
   EK(SHAD+BX+55*BY):GOTO 980
RC 1080 GOTO 720
JX 1090 REM FLAME THROWER ROUTINE
VN 1100 IF J=11 OR J=7 THEN RX=8:RY=J:GOT
   O 1130
RQ 1110 IF J=14 OR J=13 THEN RY=13.5:RX=J
   :GOTO 1130
QM 1120 GOTO 720
VE 1130 BX=QX+SGN(RX-J):BY=QY+SGN(RY-J)
HS 1140 LX=BX:LY=BY
VX 1150 FOR T=1 TO 4:SOUND 1,128,4,11+T
ZO 1160 IF LX>49 AND J=7 OR LX<6 AND J=11
    THEN 1240
AD 1170 IF LY>39 AND J=13 OR LY<6 AND J=1
   4 THEN 1240
XG 1180 IF T=1 THEN POKE ICE+LX+55*LY,DR(
   J):GOTO 1200
CA 1190 POKE ICE+LX+55*LY,DZ(J)
ZD 1200 LX=LX+SGN(RX-J):LY=LY+SGN(RY-J)
JQ 1210 Z=USR(1536,ADR(MAP$)+QX-5+55*(QY-
   5),SCR+1+20*1)
JQ 1220 NEXT T
AD 1230 REM ERASE FLAME THROWER
HU 1240 LX=BX:LY=BY
VD 1250 FOR Q=1 TO T-1:SOUND 1,128,4,11+T
   -Q
HU 1260 POKE ICE+LX+55*LY,PEEK(SHAD+LX+55
   *LY)
KI 1270 Z=USR(1536,ADR(MAP$)+QX-5+55*(QY-
   5),SCR+1+20*1)
AB 1280 LX=LX+SGN(RX-J):LY=LY+SGN(RY-J)
AV 1290 NEXT Q:POSITION FL,18:? #6;CHR$(0
   ):FL=FL-1:SOUND 1,0,0,0:GOTO 720
GC 1300 REM MONSTER MOVE ROUTINE
NY 1310 GX=SGN(QX-MX):GY=SGN(QY-MY):P=PEE
   K(ICE+MX+GX+55*(MY+GY))
YK 1320 SOUND 1,90,10,15
TJ 1330 IF P=63 THEN TI=02:GOTO 1370
GV 1340 IF P=77 THEN DES=2:SC=SC+100:POSI
   TION 12,22:? #6;SC:SOUND 1,0,0,0:GOTO
   1650
DV 1350 IF P=8 THEN 2030
ZP 1360 TI=01
WE 1370 POKE ICE+MX+55*MY,U:MX=MX+GX:MY=M
   Y+GY:U=PEEK(ICE+MX+55*MY):IF U=63 OR U
   =143 THEN U=0
YK 1380 POKE ICE+MX+55*MY,137
IO 1390 Q=USR(1536,ADR(MAP$)+QX-5+55*(QY-
   5),SCR+1+20*1)
XU 1400 SOUND 1,0,0,0:RETURN
VC 1410 REM MELT DOWN ROUTINE
KS 1420 IF RD=19 AND M1>1 THEN 1460
EA 1430 IF RD=19 AND M1=1 THEN 1500
PC 1440 RD=RD+1:POSITION RD,14:? #6;CHR$(
   7):M=50:M1=M1-1
AW 1450 RETURN
AI 1460 IF QC=167 THEN QC=7:QV=0:GOTO 148
   0
KH 1470 IF QC=7 THEN QC=167:QV=15
DV 1480 POSITION 19,14:? #6;CHR$(QC):SOUN
   D 2,200,14,QV
VG 1490 M1=M1-0.2:M=4:RETURN
ZZ 1500 FOR T=7 TO 0 STEP -1
IS 1510 FOR I=1 TO 15:NEXT I:POKE 504+STR
   +T,0:SOUND 2,30*T,8,15:NEXT T:SOUND 2,
   0,0,0
YV 1520 POSITION 2,5:? #6;"melt down":FOR
    T=1 TO 200:NEXT T:POSITION 2,5:? #6;"
            ":GOTO 2290
CP 1530 REM RANDOM PLACEMENT ROUTINE
JB 1540 RX=INT(RND(1)*44)+5:RY=INT(RND(1)
   *24)+15
XJ 1550 POKE SHAD+RX+55*RY,138:IX=RX:IY=R
   Y
VI 1560 FOR T=1 TO FU
GP 1570 RX=INT(RND(1)*45)+5:RY=INT(RND(1)
   *33)+7
GL 1580 IF PEEK(SHAD+RX+55*RY)<>0 THEN 15
   70
QL 1590 POKE SHAD+RX+55*RY,139:NEXT T
MO 1600 FOR T=1 TO BM
FZ 1610 RX=INT(RND(1)*45)+5:RY=INT(RND(1)
   *33)+7
YH 1620 IF PEEK(SHAD+RX+55*RY)<>0 THEN 16
   10
NZ 1630 POKE SHAD+RX+55*RY,77:NEXT T:RETU
```

```
   RN
XL 1640 REM EXPLOSION ROUTINE
BV 1650 IF DES=1 THEN POKE ICE+QX+55*QY,0
   :QX=BX:QY=BY:CX=QX:CY=QY
AA 1660 IF DES=2 THEN POKE ICE+MX+55*MY,0
   :CX=MX+GX:CY=MY+GY:MX=20:MY=7:TI=02:U=
   PEEK(ICE+MX+55*MY)
NT 1670 IF DES=3 THEN POKE ICE+VX+55*VY,0
   :CX=VX+AX:CY=VY+AY:POKE ICE+IX+55*IY,1
   38:GU=03:IQ=0:OF=1:IQ1=07
QT 1680 POKE ICE+CX+55*CY,14:Q=USR(1536,A
   DR(MAP$)+QX-5+55*(QY-5),SCR+1+20*1)
JW 1690 FOR T=1 TO 30:SOUND 3,200,8,15:NE
   XT T
PL 1700 POKE ICE+CX+55*CY,0:POKE SHAD+CX+
   55*CY,0:SOUND 3,0,0,0
QD 1710 IF DES=3 OR DES=1 OR DES=4 THEN P
   OSITION 5,18:? #6;FUL$:FL=19:POKE SCR+
   12+ME,0:ME=ME-1
RS 1720 IF ME<1 AND DES<>2 THEN POP :GOTO
    2290
KC 1730 POKE ICE+QX+55*QY,8
ZZ 1740 Q=USR(1536,ADR(MAP$)+QX-5+55*(QY-
   5),SCR+1+20*1):RETURN
VN 1750 REM RANDOMLY TELEPORT REACTER
HJ 1760 RX=INT(RND(0)*44)+5:RY=INT(RND(0)
   *24)+15
RG 1770 POKE ICE+IX+55*IY,0:POKE SHAD+IX+
   55*IY,0:IX=RX:IY=RY
ML 1780 POKE SHAD+IX+55*IY,138
KY 1790 IF PEEK(ICE+IX+55*IY)=63 THEN 181
   0
TI 1800 POKE ICE+IX+55*IY,138
LN 1810 FOR T=1 TO 255 STEP +5:SOUND 3,T,
   14,T:NEXT T:SOUND 3,0,0,0
GN 1820 Q=USR(1536,ADR(MAP$)+QX-5+55*(QY-
   5),SCR+1+20*1):GOTO 670
YC 1830 REM FIRE SATELLITE GUN ROUTINE
HE 1840 POKE ICE+VX+55*VY,0:POKE SHAD+VX+
   55*VY,0:OF=0
RE 1850 IF CO=4 THEN POKE ICE+IX+55*IY,13
   8:GU=03:IQ=0:OF=1:GOTO 1900
AM 1860 IF LV>=2 THEN AX=SGN(QX-VX):AY=SG
   N(QY-VY)
DT 1870 IF PEEK(ICE+VX+AX+55*(VY+AY))=8 T
   HEN DES=3:GOTO 1650
QQ 1880 VX=VX+AX:VY=VY+AY:IQ1=07
WZ 1890 POKE ICE+VX+55*VY,143:CO=CO+1
ZR 1900 Q=USR(1536,ADR(MAP$)+QX-5+55*(QY-
   5),SCR+1+20*1):RETURN
SH 1910 REM AIM SATELLITE GUN
IQ 1920 VX=IX:VY=IY:AX=SGN(QX-IX):AY=SGN(
   QY-IY)
TH 1930 IF QX<VX+6 AND QX>VX-6 AND QY<VY+
   6 AND QY>VY-6 THEN 1950
KE 1940 GU=03:IQ=0:RETURN
ZX 1950 CO=0:GU=03:IQ=1:OF=0:IQ1=07
DS 1960 IF PEEK(ICE+VX+AX+55*(VY+AY))=8 T
   HEN DES=3:GOTO 1650
DG 1970 VX=VX+AX:VY=VY+AY
GL 1980 POKE ICE+VX+55*VY,143:POKE ICE+IX
   +55*IY,74
JA 1990 Q=USR(1536,ADR(MAP$)+QX-5+55*(QY-
   5),SCR+1+20*1)
JT 2000 FOR T=15 TO 0 STEP -1:SOUND 3,128
   +T,8,T:NEXT T:RETURN
BH 2010 REM PLAYER KILLED BY MONSTER
RX 2020 POKE ICE+QX+55*QY,0:QX=BX:QY=BY:G
   OTO 2040
ZX 2030 POKE ICE+MX+55*MY,U:MX=MX+GX:MY=M
   Y+GY
XT 2040 POKE ICE+MX+55*MY,137
HX 2050 Q=USR(1536,ADR(MAP$)+QX-5+55*(QY-
   5),SCR+1+20*1)
XA 2060 POKE ICE+MX+55*MY,0
FY 2070 FOR T=0 TO 15:SOUND 1,128-T,10,T:
   NEXT T:MY=6:MX=20:TI=02:SOUND 1,0,0,0
QF 2080 POKE SCR+ME+12,0:ME=ME-1
JO 2090 IF ME=0 THEN GOTO 2290
OC 2100 POSITION 5,18:? #6;FUL$:FL=19
JL 2110 POKE ICE+QX+55*QY,8
NR 2120 IF P=137 THEN GOTO 670
ZL 2130 Q=USR(1536,ADR(MAP$)+QX-5+55*(QY-
   5),SCR+1+20*1):RETURN
VO 2140 REM DESTROY SATELLITE ROUTINE
KC 2150 POKE ICE+QX+55*QY,0:QX=BX:QY=BY:P
   OKE ICE+QX+55*QY,8
IP 2160 SOUND 2,0,0,0
IF 2170 Q=USR(1536,ADR(MAP$)+QX-5+55*(QY-
   5),SCR+1+20*1)
QO 2180 POSITION 1,2:? #6;" satellite":PO
   SITION 5,4:? #6;"is":POSITION 2,6:? #6
   ;"destroyed"
CC 2190 FOR T=1 TO 150:SOUND 3,60+T,12,15
   :NEXT T:SOUND 3,0,0,0
FX 2200 IF LV<5 THEN O1=O1-1:O2=O2-1:O3=O
   3-1:O5=O5-10:O4=O4-2:FU=FU-5:BM=BM+20:
   LV=LV+1
VT 2210 IF LV=2 THEN O7=O7+2
NQ 2220 IF LV=3 OR LV=4 THEN O7=O7-1
KV 2230 FL=19:MX=20:MY=6:GU=03:TI=01:SC=5
   C+500:IQ=0:OF=1:RD=9:M1=04:M=05:IQ1=07
MV 2240 POKE 77,0
HI 2250 IF LV=4 THEN ME=ME+1:POKE SCR+ME+
   12+20*0,8
LK 2260 AN=PEEK(559):POKE 559,0:GOSUB 240
   0:GOSUB 1540
JR 2270 POSITION RD+1,14:? #6;"
    ":GOTO 580
YN 2280 REM GAME OVER
CI 2290 POSITION 2,2:? #6;"game over":POS
   ITION 5,4:? #6;"hit":POSITION 4,6:? #6
   ;"start"
IB 2300 SOUND 2,0,0,0
MG 2310 POSITION 5,8:? #6;"key"
TQ 2320 IF PEEK(53279)<>6 THEN 2320
RC 2330 FOR T=1 TO 8:SOUND 3,T*2,10,15:NE
   XT T:SOUND 3,0,0,0:O1=8:O2=12:O3=10:O4
   =20:O5=55
WI 2340 QC=7:M=55:M1=20:RD=9:O7=2:IQ1=2
ZG 2350 IQ=0:MY=6:MX=20:GU=10:OF=1:ME=6:T
   I=15:FL=19:SC=0:LV=1:FU=70:BM=10
LM 2360 AN=PEEK(559):POKE 559,0:GOSUB 240
   0:GOSUB 1540
MN 2370 FOR T=0 TO 7:POKE 504+STR+T,255:N
   EXT T:POSITION 11,22:? #6;"        "
OB 2380 POSITION RD,14:? #6;"
   ":GOTO 560
JZ 2390 REM SET MAP DATA IN STRING
VD 2400 SNOW$=CHR$(63):SNOW$(55)=SNOW$:SN
   OW$(2)=SNOW$
LR 2410 MAP$=CHR$(0):MAP$(2475)=MAP$:MAP$
   (2)=MAP$
ZL 2420 BACK$=CHR$(0):BACK$(2475)=BACK$:B
   ACK$(2)=BACK$
LN 2430 FOR T=7 TO 45:MAP$(55*T)=SNOW$:NE
   XT T
FZ 2440 MAP$(55*7)=CHR$(0):RETURN
NP 2450 REM TITLE SCREEN
IV 2460 GRAPHICS 0:DLIST=PEEK(560)+PEEK(5
   61)*256+4:SETCOLOR 0,0,0:SETCOLOR 2,0,
   0:SETCOLOR 1,0,0:POKE 82,0
ND 2470 POKE 752,1:POSITION 2,0:? " "
XN 2480 POKE DLIST+4,7:POSITION 2,3:? #6;
   "NUCLEAR MOUNTAIN":POSITION 38,3:? #6;
   "By"
CM 2490 POSITION 34,4:? #6;"Brad Timmins"
IG 2500 FOR T=0 TO 12
ZJ 2510 SETCOLOR 0,0,T:NEXT T
MS 2520 FOR U=1 TO 50:NEXT U
GO 2530 FOR T=0 TO 12:SETCOLOR 1,0,T:NEXT
    T
IL 2540 FOR T=1 TO 100:NEXT T:AN=PEEK(559
   ):POKE 559,0:? #6;CHR$(125):RETURN
```

continued from page 15

```
3410 DATA 105,110,101,115,0,72,65,76,3
2,67,111,109,112,117,116,101,5930
3420 DATA 114,115,0,83,101,108,109,97,
114,116,32,83,116,111,114,101,6710
3430 DATA 115,0,73,79,85,32,80,101,110
,110,101,121,115,0,87,105,5198
3440 DATA 108,116,39,115,32,86,101,103
,101,116,97,98,108,101,115,0,5419
3450 DATA 84,105,116,97,110,105,99,32,
83,104,105,112,112,105,110,103,7119
3460 DATA 0,83,116,101,97,108,32,83,11
6,101,101,108,119,111,114,107,7383
3470 DATA 115,0,83,105,108,105,99,111,
110,32,71,117,108,99,104,0,4850
3480 DATA 68,105,115,97,115,116,114,11
1,117,115,32,73,110,115,117,114,7498
3490 DATA 97,110,99,101,0,84,104,101,3
2,76,117,99,107,32,83,113,4963
3500 DATA 117,97,114,101,0,0,67,97,112
,105,116,97,108,32,71,97,4769
3510 DATA 105,110,115,32,84,97,120,0,2
16,64,113,64,131,64,56,63,5192
3520 DATA 253,63,237,63,200,64,17,64,4
1,63,72,64,165,64,216,64,6741
3530 DATA 119,63,139,63,86,64,156,63,1
76,63,216,64,35,64,23,63,4916
3540 DATA 214,62,53,64,200,64,228,62,1
01,64,71,63,216,64,90,63,6917
3550 DATA 105,63,148,64,244,62,6,63,21
7,64,179,64,217,63,197,63,9615
3560 DATA 165,163,170,189,75,48,133,17
2,169,0,133,173,189,124,52,208,1648
3570 DATA 1,96,168,185,3,48,197,166,20
8,14,185,75,48,24,101,172,8191
3580 DATA 133,172,165,173,105,0,133,17
3,96,32,41,67,16,1,96,169,4889
3590 DATA 103,160,56,32,125,71,32,186,
65,240,1,96,165,153,10,168,8345
3600 DATA 185,184,48,208,10,185,183,48
,201,11,176,3,76,225,66,32,7485
3610 DATA 45,66,166,168,48,2,208,1,96,
189,39,48,201,3,144,10,5407
3620 DATA 169,233,160,56,32,125,71,76,
97,60,189,75,48,10,157,75,5745
3630 DATA 48,254,39,48,189,111,48,24,1
05,5,157,111,48,169,10,164,6452
3640 DATA 153,32,178,70,76,208,66,169,
86,160,56,32,125,71,32,102,6218
3650 DATA 71,201,43,208,1,96,201,35,20
8,244,201,0,96,32,51,65,7376
3660 DATA 169,252,160,56,32,125,71,169
,0,133,171,240,4,169,1,133,8448
3670 DATA 171,165,153,10,168,185,184,4
8,197,173,240,2,176,9,185,183,2576
3680 DATA 48,197,172,144,27,240,25,164
,153,165,172,32,178,70,169,0,9408
3690 DATA 133,164,165,171,208,7,164,16
6,165,172,32,155,70,76,208,66,455
3700 DATA 169,1,133,164,32,22,74,165,1
39,240,10,32,225,66,32,238,9338
3710 DATA 66,165,168,16,188,104,104,76
,146,73,165,163,133,167,32,41,8592
3720 DATA 67,16,3,132,168,96,160,0,185
,3,48,132,163,197,153,208,1199
3730 DATA 77,32,61,74,230,170,165,164,
240,14,169,20,160,57,32,125,8744
3740 DATA 71,165,172,166,173,32,118,71
,169,135,160,56,32,125,71,165,8946
3750 DATA 164,208,7,169,158,160,56,32,
125,71,198,170,32,102,71,201,9603
3760 DATA 28,208,10,165,164,208,245,16
9,0,133,163,240,8,201,33,240,3285
3770 DATA 13,201,12,208,231,165,163,13
3,168,165,167,133,163,96,164,163,4529
3780 DATA 200,192,36,144,163,160,0,240
,159,169,124,160,56,32,125,71,9335
3790 DATA 32,186,65,240,1,96,165,153,1
64,163,10,170,189,184,48,208,2646
3800 DATA 10,189,183,48,217,111,48,144
,37,240,35,165,153,153,3,48,8210
3810 DATA 152,170,254,147,48,185,111,4
8,164,153,32,178,70,169,169,160,2028
3820 DATA 56,32,125,71,32,97,60,169,0,
133,171,133,173,96,169,194,1152
3830 DATA 160,56,32,125,71,32,97,60,16
9,0,96,32,41,67,16,3,1124
3840 DATA 132,168,96,165,164,208,13,16
9,115,160,56,32,125,71,32,186,8477
3850 DATA 65,240,1,96,32,45,66,165,168
,48,248,208,5,165,164,208,2373
3860 DATA 243,96,170,222,147,48,169,25
5,157,3,48,189,191,48,164,153,2241
3870 DATA 32,155,70,76,208,66,160,35,1
65,153,217,3,48,240,3,136,9205
3880 DATA 16,248,96,166,153,181,149,20
1,32,208,22,169,0,141,123,67,9375
3890 DATA 169,70,141,124,67,169,125,32
,17,70,169,217,160,64,76,112,8822
3900 DATA 67,174,10,210,224,6,176,219,
189,73,68,141,123,67,189,79,832
3910 DATA 68,141,124,67,169,125,32,17,
70,169,200,160,64,32,222,68,8993
3920 DATA 169,98,160,68,32,125,71,32,2
55,255,76,208,66,169,10,208,1579
3930 DATA 2,169,15,164,153,133,165,32,
155,70,32,158,67,169,117,160,9840
3940 DATA 68,32,125,71,165,165,162,0,7
6,118,71,169,85,160,68,32,7230
3950 DATA 125,71,96,32,183,68,48,213,1
70,189,39,48,201,3,176,205,996
3960 DATA 254,39,48,189,75,48,10,157,7
5,48,134,167,32,158,67,169,8151
3970 DATA 117,160,68,32,125,71,169,164
,160,68,32,125,71,165,167,10,8155
3980 DATA 170,189,236,64,168,189,235,6
4,76,125,71,32,249,67,169,126,1448
3990 DATA 160,68,32,125,71,169,10,133,
172,162,0,32,118,71,230,171,9651
4000 DATA 32,224,65,104,104,96,169,92,
160,68,76,125,71,32,183,68,7766
4010 DATA 48,217,170,189,75,48,201,2,1
44,209,74,157,75,48,169,136,286
4020 DATA 160,68,72,132,155,32,249,67,
32,21,70,104,164,155,32,125,7604
4030 DATA 71,165,168,10,170,189,236,64
,168,189,235,64,76,125,71,32,193
4040 DATA 183,68,48,167,170,189,111,48
,201,2,144,159,74,157,111,48,9230
4050 DATA 169,149,160,68,208,204,222,1
28,0,132,50,166,67,67,68,67,7874
4060 DATA 68,67,71,111,111,100,46,155,
0,66,97,100,46,155,0,155,5811
4070 DATA 155,89,111,117,114,32,108,11
7,99,107,32,119,97,115,32,45,5470
4080 DATA 32,0,89,111,117,32,119,105,1
10,32,0,89,111,117,32,108,4940
4090 DATA 111,115,101,32,0,49,47,50,32
,82,101,110,116,32,111,110,4805
4100 DATA 155,0,49,47,50,32,86,97,108,
117,101,32,111,110,32,155,5990
4110 DATA 0,97,110,32,105,109,112,114,
111,118,101,109,101,110,116,32,7340
4120 DATA 111,110,155,0,32,41,67,16,1,
96,160,0,162,0,185,3,3577
4130 DATA 48,197,153,208,5,152,157,0,4
1,232,200,192,36,208,239,232,5769
4140 DATA 138,32,6,74,170,189,255,40,1
33,168,96,133,155,132,156,169,3181
4150 DATA 2,32,17,70,160,0,132,174,177
,155,240,10,9,128,32,17,6227
4160 DATA 70,164,174,200,208,240,169,2
2,76,17,70,63,21,18,58,42,4166
4170 DATA 56,61,57,13,1,5,0,37,35,8,10
,47,40,62,45,11,8210
```

```
4180 DATA 16,46,22,43,23,162,255,142,2
52,2,173,252,2,201,255,240,6254
4190 DATA 249,41,63,201,12,208,3,169,0
,96,201,52,208,3,169,126,9287
4200 DATA 96,160,25,217,254,68,240,5,1
36,16,248,48,216,152,24,105,621
4210 DATA 65,96,133,155,132,156,160,0,
132,174,169,127,145,155,32,24,9432
4220 DATA 69,162,0,142,31,208,164,174,
201,0,240,35,201,126,208,14,1714
4230 DATA 192,0,240,228,169,0,145,155,
136,145,155,76,75,69,192,8,9394
4240 DATA 176,214,153,51,49,32,70,71,9
,128,145,155,200,208,201,153,2793
4250 DATA 51,49,145,155,96,169,207,141
,48,2,169,52,141,49,2,169,7713
4260 DATA 58,141,47,2,160,4,185,32,56,
153,196,2,136,16,247,32,7547
4270 DATA 15,70,169,0,133,130,169,32,1
33,131,162,0,169,251,145,130,2071
4280 DATA 200,138,24,105,209,145,130,2
00,169,253,145,130,152,24,105,18,1054
4290 DATA 168,232,228,138,208,230,160,
0,132,175,165,130,24,105,4,164,477
4300 DATA 131,32,69,69,173,51,49,240,2
39,165,130,24,105,20,133,130,9651
4310 DATA 169,40,133,156,164,175,185,1
20,52,133,155,160,8,185,51,49,9373
4320 DATA 145,155,136,16,248,164,175,2
00,196,138,208,204,96,32,249,67,4679
4330 DATA 169,126,160,68,32,125,71,169
,45,76,234,67,169,125,201,155,2247
4340 DATA 208,7,169,0,133,128,230,129,
96,201,126,208,12,32,94,70,9034
4350 DATA 32,119,70,169,0,168,145,130,
96,201,125,208,20,160,0,152,246
4360 DATA 153,0,32,153,0,33,153,0,34,2
00,208,244,133,128,133,129,1592
4370 DATA 96,32,70,71,72,32,119,70,160
,0,104,145,130,166,128,232,939
4380 DATA 224,32,144,4,230,129,162,0,1
34,128,96,165,128,170,5,129,9879
4390 DATA 240,17,202,16,12,162,31,164,
129,208,4,162,0,240,2,198,9682
4400 DATA 129,134,128,96,169,224,133,1
30,169,31,133,131,164,129,165,130,3084
4410 DATA 24,105,32,133,130,144,2,230,
131,136,16,242,165,130,24,101,200
4420 DATA 128,133,130,144,2,230,131,96
,72,152,10,168,104,24,121,183,9580
4430 DATA 48,153,183,48,185,184,48,101
,173,153,184,48,76,200,70,133,1351
4440 DATA 160,152,10,168,185,183,48,56
,229,160,153,183,48,185,184,48,2695
4450 DATA 229,173,153,184,48,160,39,16
9,0,153,227,48,136,16,250,166,2046
4460 DATA 138,189,161,52,170,200,152,2
4,105,17,157,227,48,169,26,157,431
4470 DATA 228,48,134,161,132,162,152,1
0,168,185,184,48,170,185,183,48,2891
4480 DATA 32,25,71,166,161,160,255,200
,177,243,240,9,32,70,71,157,2333
4490 DATA 229,48,232,208,242,164,138,1
65,161,24,121,158,52,170,164,162,3847
4500 DATA 200,196,138,208,193,96,133,2
12,134,213,5,213,208,9,169,179,4682
4510 DATA 133,243,169,55,133,244,96,32
,170,217,32,230,216,160,255,200,7798
4520 DATA 177,243,16,251,41,127,145,24
3,200,169,71,145,243,169,0,200,4897
4530 DATA 145,243,96,32,88,71,29,55,53
,166,136,96,32,88,71,29,5048
4540 DATA 59,53,166,136,96,72,42,42,42
,42,41,3,134,136,170,104,6434
4550 DATA 41,159,96,162,255,142,252,2,
173,252,2,201,255,240,249,142,8945
4560 DATA 252,2,96,32,25,71,165,243,16
4,244,141,132,71,140,133,71,1947
4570 DATA 173,255,255,240,20,32,17,70,
165,170,208,3,32,25,72,238,9112
4580 DATA 132,71,208,236,238,133,71,20
8,231,96,72,138,72,152,72,169,2906
4590 DATA 192,160,7,162,2,141,24,208,7
3,15,141,23,208,73,15,141,7300
4600 DATA 10,212,202,16,240,24,105,2,1
36,16,232,160,7,169,204,162,1764
4610 DATA 2,141,24,208,73,15,141,23,20
8,73,15,141,10,212,202,16,8267
4620 DATA 240,56,233,2,136,16,232,169,
4,141,23,208,104,168,104,170,1610
4630 DATA 104,64,72,165,159,208,28,169
,58,141,0,212,169,10,141,23,8625
4640 DATA 208,141,22,208,169,162,141,2
4,208,141,10,212,141,26,208,230,3957
4650 DATA 159,104,64,169,0,141,10,212,
141,26,208,104,64,169,0,133,8888
4660 DATA 77,133,159,76,98,228,72,169,
0,133,20,165,20,240,252,104,2092
4670 DATA 96,134,128,132,129,96,169,7,
162,72,160,16,32,92,228,32,8172
4680 DATA 211,58,169,230,141,48,2,169,
52,141,49,2,169,62,141,47,7166
4690 DATA 2,32,25,72,169,36,141,244,2,
169,157,141,0,2,169,71,7945
4700 DATA 141,1,2,160,4,185,27,56,153,
196,2,136,16,247,169,3,8496
4710 DATA 141,4,212,169,0,133,139,133,
140,169,2,133,138,169,37,32,8801
4720 DATA 215,72,162,0,165,139,10,10,1
68,185,171,55,157,195,54,200,2058
4730 DATA 232,224,4,208,244,165,138,9,
144,141,215,54,32,237,72,74,1219
4740 DATA 144,3,76,191,72,74,144,19,16
9,32,32,215,72,230,138,165,1485
4750 DATA 138,201,5,144,205,169,2,133,
138,208,199,74,144,196,169,16,2804
4760 DATA 32,215,72,165,139,73,1,133,1
39,76,117,72,160,3,169,0,7121
4770 DATA 153,63,53,136,16,250,200,169
,1,153,63,53,200,196,138,208,3694
4780 DATA 248,76,136,69,141,2,210,169,
164,141,3,210,160,192,202,208,5968
4790 DATA 253,134,77,136,208,248,140,3
,210,96,173,31,208,168,69,140,3074
4800 DATA 37,140,132,140,201,4,96,169,
229,141,0,2,169,71,141,1,7943
4810 DATA 2,169,36,141,244,2,169,166,1
41,48,2,169,52,141,49,2,6781
4820 DATA 169,61,141,47,2,32,15,70,160
,4,185,22,56,153,196,2,6210
4830 DATA 136,16,247,169,0,133,170,133
,164,32,22,74,166,153,181,149,1792
4840 DATA 133,163,168,185,147,48,201,4
,208,6,32,54,67,76,140,73,7140
4850 DATA 32,61,74,164,163,185,3,48,13
3,166,48,10,197,153,240,57,932
4860 DATA 32,208,65,76,140,73,32,156,6
6,208,46,164,163,185,124,52,966
4870 DATA 240,39,133,163,168,185,3,48,
16,31,185,111,48,72,74,24,5541
4880 DATA 121,111,48,153,111,48,32,61,
74,169,216,160,56,32,125,71,8373
4890 DATA 32,156,66,164,163,104,153,11
1,48,32,238,66,76,92,65,32,7411
4900 DATA 21,70,32,21,70,165,153,72,17
0,222,63,53,168,32,235,74,648
4910 DATA 169,74,160,56,32,125,71,104,
10,168,169,0,153,183,48,153,9518
4920 DATA 184,48,162,35,189,3,48,197,1
53,208,14,169,255,157,3,48,666
4930 DATA 189,191,48,157,111,48,222,14
7,48,202,16,232,32,200,70,32,36
4940 DATA 97,60,162,0,160,3,185,63,53,
240,1,232,136,16,247,224,3213
```

```
4950 DATA 2,144,3,76,185,58,32,21,70,1
60,3,185,63,53,208,3,6430
4960 DATA 136,16,248,32,235,74,169,37,
160,56,32,125,71,32,102,71,6987
4970 DATA 76,148,57,133,154,173,10,210
,41,127,240,249,197,154,240,2,5219
4980 DATA 176,243,96,160,35,185,111,48
,240,9,74,133,165,32,6,74,7543
4990 DATA 24,101,165,153,191,48,136,16
,236,165,164,240,12,160,35,185,2985
5000 DATA 111,48,74,153,191,48,136,16,
246,96,230,170,169,185,160,55,4175
5010 DATA 32,125,71,162,7,160,0,32,36,
72,165,163,10,170,189,236,1340
5020 DATA 64,168,189,235,64,32,222,68,
162,8,160,1,32,36,72,162,7439
5030 DATA 0,164,163,185,111,48,32,118,
71,169,24,133,128,162,0,164,9343
5040 DATA 163,185,191,48,32,118,71,164
,163,185,39,48,9,16,141,72,6945
5050 DATA 32,162,24,160,2,32,36,72,164
,163,185,124,52,208,10,169,219
5060 DATA 14,160,56,32,125,71,76,163,7
4,169,10,160,56,32,125,71,7150
5070 DATA 162,8,160,3,32,36,72,164,163
,185,3,48,133,174,185,75,9998
5080 DATA 48,133,165,185,124,52,240,16
,170,189,3,48,197,174,208,8,1643
5090 DATA 189,75,48,24,101,165,133,165
,165,165,162,0,32,118,71,169,169
5100 DATA 24,133,128,32,119,70,164,163
,185,3,48,168,32,235,74,198,1592
5110 DATA 170,162,0,160,4,76,36,72,185
,120,52,133,155,169,40,133,9690
5120 DATA 156,192,255,208,8,169,17,133
,155,169,56,133,156,165,155,164,4078
5130 DATA 156,76,125,71,160,35,169,255
,153,3,48,169,0,153,39,48,7788
5140 DATA 136,16,243,160,115,185,0,52,
153,75,48,136,16,247,160,0,9131
5150 DATA 185,0,224,153,0,36,185,0,225
,153,0,37,185,0,226,153,372
5160 DATA 0,38,185,0,227,153,0,39,200,
208,229,160,7,169,255,153,5205
5170 DATA 248,37,136,16,250,169,148,13
3,12,169,57,133,13,169,192,141,2020
5180 DATA 14,212,96,0,112,71,115,40,11
1,112,64,66,67,69,71,40,5174
5190 DATA 40,40,40,40,40,40,40,40,40,4
0,40,40,40,40,40,40,630
5200 DATA 40,40,40,40,40,40,40,40,40,4
0,40,40,110,73,40,40,2012
5210 DATA 108,64,63,64,64,63,64,64,64,
66,66,66,67,67,68,68,4220
5220 DATA 68,66,64,67,69,40,40,40,40,4
0,40,40,40,40,40,40,1065
5230 DATA 40,40,40,40,119,64,94,40,109
,64,64,64,64,64,63,64,4377
5240 DATA 64,64,64,64,64,64,64,64,64,6
4,64,64,76,40,98,113,5058
5250 DATA 68,68,114,40,40,40,40,40,40,
123,67,65,64,115,101,40,4700
5260 DATA 64,64,63,64,64,64,63,63,64,6
4,64,64,64,64,64,64,3946
5270 DATA 64,64,64,64,64,64,64,64,85,1
25,109,100,40,40,40,40,4308
5280 DATA 121,64,64,63,120,40,40,75,64
,63,64,51,52,57,58,43,3247
5290 DATA 44,43,44,57,58,59,60,57,58,5
7,58,57,58,53,54,64,3044
5300 DATA 107,97,64,64,96,40,40,121,64
,64,63,76,116,40,40,87,4898
5310 DATA 64,63,64,64,64,64,63,63,63,6
4,64,64,64,64,64,64,3988
5320 DATA 64,64,64,64,64,64,64,64,96,9
7,64,64,103,123,122,64,6845
5330 DATA 64,63,76,40,40,40,40,64,63,6
4,64,43,44,64,63,63,2988
5340 DATA 64,64,64,64,64,64,64,64,64,6
4,64,64,64,43,44,64,3450
5350 DATA 94,95,64,64,64,64,64,64,64,6
4,100,40,40,40,40,64,3246
5360 DATA 63,64,64,64,64,63,64,63,63,6
4,64,64,64,64,64,64,4040
5370 DATA 64,64,64,64,64,64,64,64,64,6
4,64,64,64,64,64,63,4058
5380 DATA 64,64,100,40,40,40,40,64,64,
63,64,43,44,64,63,63,3111
5390 DATA 63,64,64,62,49,49,49,61,62,5
0,49,50,61,43,44,57,2555
5400 DATA 58,43,44,43,44,47,48,64,63,6
4,99,40,40,40,40,64,2678
5410 DATA 64,63,64,64,64,64,64,63,63,6
4,64,62,49,50,49,61,3407
5420 DATA 62,50,49,50,61,64,64,64,64,6
4,64,64,64,64,64,63,3962
5430 DATA 64,64,64,104,40,40,40,83,64,
63,64,57,58,64,64,64,3842
5440 DATA 63,63,64,62,49,49,49,61,62,5
0,49,50,61,64,64,64,3309
5450 DATA 64,64,64,64,63,57,58,64,64,6
4,64,105,40,40,40,79,3789
5460 DATA 64,64,63,64,64,64,64,64,64,6
3,64,64,64,64,64,64,4151
5470 DATA 64,64,64,64,64,64,64,64,64,6
4,64,64,63,64,64,64,4161
5480 DATA 64,64,101,40,40,40,40,126,78
,64,64,57,58,57,58,53,3863
5490 DATA 54,64,63,64,64,64,64,64,64,6
4,64,64,64,64,64,64,4181
5500 DATA 64,55,56,43,44,43,44,64,64,1
01,40,40,40,40,40,40,2138
5510 DATA 127,78,64,64,64,64,64,64,64,
63,63,64,64,64,64,64,4284
5520 DATA 64,64,64,64,64,64,64,64,64,6
4,64,63,64,64,64,64,4212
5530 DATA 102,40,40,40,40,40,40,40,40,
127,125,125,78,64,64,43,5095
5540 DATA 44,43,44,57,58,57,58,59,60,4
3,44,43,44,57,58,57,2662
5550 DATA 58,57,58,64,64,64,64,64,96,4
0,40,40,40,40,40,40,2320
5560 DATA 40,40,40,40,126,125,78,64,64
,64,64,63,64,64,64,64,4786
5570 DATA 64,64,64,64,64,64,64,64,64,8
1,81,82,82,82,78,64,5543
5580 DATA 72,40,40,40,40,40,40,40,40,4
0,40,40,40,40,127,125,3717
5590 DATA 125,125,125,79,64,64,64,64,6
4,64,64,64,64,64,64,64,4720
5600 DATA 120,40,40,40,40,40,40,79,64,
72,40,40,40,40,40,40,1968
5610 DATA 40,40,40,40,40,40,40,40,40,4
0,40,126,79,76,78,64,4047
5620 DATA 64,64,64,76,125,125,125,125,
125,40,40,40,40,40,40,126,5699
5630 DATA 78,64,104,40,40,40,40,40,40,
40,40,40,40,40,40,40,1348
5640 DATA 40,40,40,40,126,40,126,78,64
,64,77,40,40,40,40,40,3279
5650 DATA 40,40,40,40,40,40,40,40,126,
118,117,40,40,40,40,40,3491
5660 DATA 40,40,40,40,40,40,40,40,40,4
0,40,40,40,40,40,127,2492
5670 DATA 78,64,124,40,40,40,40,40,40,
40,40,40,40,40,40,40,1448
5680 DATA 40,127,84,40,40,40,40,40,40,
40,40,40,40,40,40,40,1426
5690 DATA 40,40,40,40,40,40,40,40,127,
78,40,40,40,40,40,40,2293
5700 DATA 40,40,40,40,40,40,40,40,40,4
0,40,40,40,40,40,226,4116
5710 DATA 2,227,2,0,48,0,0,0,0,0,0,0,0
,0,0,0,6412
```

LISTING 2: ASSEMBLY

```
0100 ;SAVE#D:CAPITAL1.M65
0110      .OPT NO LIST
0120 ;-------------------------;
0130 ;                         ;
0140 ;        CAPITAL!         ;
0150 ;                         ;
0160 ;by: Bryan Schappel       ;
0170 ;and Barry Kolbe          ;
0180 ;                         ;
0190 ;(C) 1988 BBK Enterprises;
0200 ;                         ;
0210 ;-------------------------;
0220 ;
0230      *=  0
0240 PASS .= PASS+1
0250        .IF PASS=1
0260        .INCLUDE #D:SYSEQU.M65
0270        .ENDIF
0280 ;
0290 ;Zero Page Equates
0300 ;
0310      *=  $80
0320 XPOS .DS 1        ;'cursor' x,y
0330 YPOS .DS 1        ;positions
0340 SCR .DS 2         ;ind. scr loc
0350 SCR2 .DS 2
0360 SAVEX .DS 1       ;save X
0370 XSV1 .DS 1        ;register
0380 XSV2 .DS 1
0390 PLAYER .DS 1
0400 MAXP .DS 1        ;max # plyrs
0410 SPEED .DS 1       ;game speed
0420 CONSAV .DS 1      ;CONSOL save
0430 PX0 .DS 4         ;PMG x,y
0440 PY0 .DS 4         ;positions
0450 PTHP0 .DS 4       ;plyr map pos
0460 PNUM .DS 1        ;curr. plyr #
0470 RNUM .DS 1
0480 L   .DS 2         ;indirect
0490 SQRS .DS 1        ;squares to mov
0500 SCNT .DS 1
0510 BDL .DS 1
0520 CTMP .DS 1        ;cash tmp save
0530 XT  .DS 1         ;save x & y
0540 YT  .DS 1
0550 CPROP .DS 1       ;cur. property
0560 DISFLAG .DS 1     ;disaster flag
0570 VALUE .DS 1
0580 OWNER .DS 1       ;owner of prop
0590 TPROP .DS 1       ;temp prop. #
0600 PPROP .DS 1       ;picked property
0610 ESCFLAG .DS 1     ;print 'ESC' ?
0620 PAUSE? .DS 1      ;print speed
0630 RFLAG .DS 1       ;pay to whom?
0640 AMOUNT .DS 2      ;amount owed
0650 SVY .DS 1         ;save y
0660 SVY2 .DS 1        ;register
0670 ;
0680 ;Other Memory Usage
0690 ;
0700 GRMEM = $2000     ;txt scrn RAM
0710 MSET =  $2400     ;a char. set
0720 NAMEBUF = $2800
0730 GRTAB = $2900
0740 BOTLIN = 19       ;txt scrn length
0750 PMB =   $8000     ;P/M base
0760 P0  =   $8400     ;player 0
0770 P1  =   $8500     ;player 1
0780 P2  =   $8600     ;player 2
0790 P3  =   $8700     ;player 3
0800 MONEY_CLR = $A2 ;$ line color
0810 ;
0820      .MACRO PRINT
0830      LDA # <%1
0840      LDY # >%1
0850      JSR EPRINT
0860      .ENDM
0870 ;
0880      .MACRO POSITION
0890      LDX # <%1
0900      LDY # <%2
0910      JSR POSIT
0920      .ENDM
0930 ;
0940 ;Start of Assembly
0950 ;
0960      *=  $3000
0970 CSET JMP FAKE
0980 ;
0990 ;Tables for property ownership
1000 ;rent, cost and double Xref.
1010 ;
1020 WHO .DS 36        ;owners
1030 IMP .DS 36        ;improvements
1040 RENT .DS 36       ;rent
1050 COST .DS 36       ;value
1060 REFTAB .DS 36     ;shadow table
1070 CASH .DS 8        ;players money
1080 RANTAB .DS 36     ;selling price
1090 MONEY .DS 40      ;money line
1100 PRPLIN .DS 40     ;prop. line
1110 IBUF .DS 16       ;input buffer
1120 ;
1130 ;Fetch the character set for
1140 ;the Map of the US
1150 ;
1160      *=  $3140
1170      .INCLUDE #D:CAPITAL3.M65
1180 ;
1190 ;Initial Table Values
1200 ;
1210 ;the initial rent values
1220 ;must be less than 32 since
1230 ;inprovements are done by
1240 ;multiplying by 8 (3 ASL's)
1250 ;
1260 RENTORG .BYTE 0,15,14,8,25,21
1270      .BYTE 0,12,20,22,28,0
1280      .BYTE 4,6,15,7,5,0
1290      .BYTE 5,8,14,18,0,7
1300      .BYTE 19,3,0,16,15,21
1310      .BYTE 19,14,0,31,12,10
1320 ;
1330 ;the initial cost of properties
1340 ;
1350 COSTORG .BYTE 0,23,37,25,69,65
1360      .BYTE 0,41,60,73,84,0
1370      .BYTE 12,15,38,17,13,0
1380      .BYTE 18,27,36,37,0,22
1390      .BYTE 51,11,0,53,48,56
1400      .BYTE 61,32,0,100,47,42
1410 ;
1420 ;initial status of each property
1430 ;128 =special, like TAX
1440 ;0=single,free - 1=single,bought
1450 ;2=double,free - 3=double,bought
1460 ;4=LUCK, including CAP GAINS
1470 ;
1480 REFORG .BYTE 128,0,0,0,2,2
1490      .BYTE 4,0,0,2,2,128
1500      .BYTE 0,0,0,2,2,128
1510      .BYTE 0,2,2,0,4,0
1520      .BYTE 0,0,128,2,2,0
1530      .BYTE 2,2,4,0,2,2
1540      .WORD 375,375,375,375
1550 ;
1560 ;these are the special squares
1570 ;which have scrolling messages
1580 ;
1590 SPLC .BYTE 0,11,17,26
1600 X9  .BYTE 0,9,18,27
1610 ;
1620 ;These cross reference tables
1630 ;identify the double properties
1640 ;by halves and allow the program
1650 ;to find the other half of a
1660 ;property.
1670 ;
1680 XREF .BYTE 0,0,0,0,5,4
1690      .BYTE 0,0,0,10,9,0
1700      .BYTE 0,0,0,16,15,0
1710      .BYTE 0,20,19,0,0,0
1720      .BYTE 0,0,0,28,27,0
1730      .BYTE 31,30,0,0,35,34
1740 ;
1750 ;these are used to center the
1760 ;info on the money line
1770 ;
1780 OFFSET .BYTE 15,13,10
1790 XST .BYTE 5,3,1
1800 ;
1810 ;Display List (Transactions)
1820 ;
1830 DLTXT .BYTE $70,$70,$70,$46
1840      .WORD TITLE
1850      .BYTE $06,$30,$62
1860      .WORD GRMEM
1870      .BYTE 0,$22,$22,$22,$22,$22
1880      .BYTE $22,$22,$22,$22,$22
1890      .BYTE $22,$22,$22,$22,$22
1900      .BYTE $22,$22,$02,$B0,$C2
1910      .WORD MONEY
1920      .BYTE $20,$46
1930      .WORD PLRN
1940      .BYTE $41
1950      .WORD DLTXT
1960 ;
```

```
1970 ;Display List (Names)
1980 ;
1990 NDLIST .BYTE $70,$70,$70,$70,$70
2000      .BYTE $70,$46
2010      .WORD NAMEL
2020      .BYTE $70,$70,$46
2030      .WORD GR0MEM
2040      .BYTE $70,$06,$70,$06,$70
2050      .BYTE $06,$41
2060      .WORD NDLIST
2070 ;
2080 ;Display List (Intro)
2090 ;
2100 IDL .BYTE $70,$70,$70,$F0,$42
2110      .WORD INTL
2120      .BYTE $02,$02,$02,$02
2130      .BYTE $30,$06,$70,$06,$70
2140      .BYTE $06,$30,$56
2150      .WORD AND
2160      .BYTE $30,$56
2170      .WORD KNAME
2180      .BYTE $70,$70,$46
2190      .WORD INTM
2200      .BYTE $70,$06,$70,$70,$70
2210      .BYTE $06,$56
2220      .WORD BBK
2230      .BYTE $41
2240      .WORD IDL
2250 ;
2260 ;Display List (US map)
2270 ;
2280 DL  .BYTE $70,$70,$50,$44
2290      .WORD SCRN
2300      .BYTE $04,$04,$04,$04,$04
2310      .BYTE $04,$04,$04,$04,$04
2320      .BYTE $04,$04,$04,$04,$04
2330      .BYTE $04,$04,$04,$04,$84
2340      .BYTE $46
2350      .WORD PLRN
2360      .BYTE $C2
2370 TWIN .WORD RLMES
2380      .BYTE $A0,$42
2390      .WORD MONEY
2400      .BYTE $41
2410      .WORD DL
2420 ;
2430 ;tables to convert ATASCII
2440 ;to IC or vice versa
2450 ;
2460 A2I .BYTE $40,$00,$20,$60
2470 I2A .BYTE $20,$40,$00,$60
2480 ;
2490 ;table for which players
2500 ;are on the screen 0=off,1=on
2510 ;
2520 ONBRD .BYTE 0,0,0,0
2530 ;
2540 ;screen titles
2550 ;
2560 TITLE .SBYTE "    CAPITAL!    "
2570      .SBYTE "  transactions  "
2580 PLRN .SBYTE "                "
2590      .SBYTE "                "
2600 NAMEL .SBYTE "  enter yo"
2610      .SBYTE "ur names  "
2620 ;
2630 ;Data for the word CAPITAL!
2640 ;on the intro screen. This
2650 ;was done with contro chars.
2660 ;
2670 INTL .BYTE $00,$00,$00,$48,$80
2680      .BYTE $80,$4A,$00,$48,$80
2690      .BYTE $80,$4A,$00,$48,$80
2700      .BYTE $80,$4A,$00,$48,$80
2710      .BYTE $4A,$00,$48,$80,$4A
2720      .BYTE $00,$48,$80,$80,$4A
2730      .BYTE $00,$4A,$00,$00,$00
2740      .BYTE $00,$48,$00,$00,$00
2750      .BYTE $00,$00,$00,$80,$C8
2760      .BYTE $00,$00,$00,$80,$C8
2770      .BYTE $CA,$80,$00,$80,$C8
2780      .BYTE $CA,$80,$00,$00,$80
2790      .BYTE $00,$00,$00,$80,$00
2800      .BYTE $00,$80,$C8,$CA,$80
2810      .BYTE $00,$80,$00,$00,$00
2820      .BYTE $00,$80,$00,$00,$00
2830      .BYTE $00,$00,$00,$80,$00
2840      .BYTE $00,$00,$00,$80,$52
2850      .BYTE $52,$80,$00,$80,$80
2860      .BYTE $80,$C8,$00,$00,$80
2870      .BYTE $00,$00,$00,$80,$00
2880      .BYTE $00,$80,$52,$52,$80
2890      .BYTE $00,$80,$00,$00,$00
2900      .BYTE $00,$80,$00,$00,$00
2910      .BYTE $00,$00,$00,$80,$4A
2920      .BYTE $00,$00,$00,$80,$00
2930      .BYTE $00,$80,$00,$80,$00
2940      .BYTE $00,$00,$00,$00,$80
2950      .BYTE $00,$00,$00,$80,$00
2960      .BYTE $00,$80,$00,$00,$80
2970      .BYTE $00,$80,$4A,$00,$00
2980      .BYTE $00,$C8,$00,$00,$00
2990      .BYTE $00,$00,$00,$CA,$80
3000      .BYTE $80,$C8,$00,$C8,$00
3010      .BYTE $00,$CA,$00,$C8,$00
3020      .BYTE $00,$00,$00,$CA,$80
3030      .BYTE $C8,$00,$00,$80,$00
3040      .BYTE $00,$C8,$00,$00,$CA
3050      .BYTE $00,$CA,$80,$80,$C8
3060      .BYTE $00,$4F,$00,$00,$00
3070 ;
3080 ;Other Screen Memory
3090 ;
3100      .SBYTE "game of high fin"
3110      .SBYTE "ance"
3120 LO  .SBYTE "   programmed by"
3130      .BYTE "Z",0,0,0
3140      .SBYTE "   BRYAN SCHAPPEL"
3150      .SBYTE "   "
3160 AND .SBYTE "            AND   "
3170      .SBYTE "     "
3180 KNAME .SBYTE "       BARRY KOLBE"
3190      .SBYTE "    "
3200 INTM .SBYTE " option   fast "
3210      .SBYTE "game "
3220 PLN .SBYTE " SELECT   2 "
3230      .SBYTE "PLAYERS "
3240      .SBYTE "   COPYRIGHT "
3250      .SBYTE "1988   "
3260 BBK .SBYTE "    bbk enterprises"
3270      .SBYTE "   "
3280 ;
3290 ;Scrolling Text Messages
3300 ;
3310 RLMES .SBYTE "  OPTION R"
3320      .SBYTE "OLL DICE  "
3330      .SBYTE "  SELECT S"
3340      .SBYTE "TOP DICE  "
3350 ;
3360 STKMES .SBYTE "  Stock Ma"
3370      .SBYTE "rket Divid"
3380      .SBYTE "ends pay y"
3390      .SBYTE "ou $12G   "
3400 ;
3410 TAXMES .SBYTE "  IRS Tax "
3420      .SBYTE "Audit: los"
3430      .SBYTE "e 12% of y"
3440      .SBYTE "our cash  "
3450 ;
3460 VACMES .SBYTE "    Vacati"
3470      .SBYTE "on time at"
3480      .SBYTE " the Hog H"
3490      .SBYTE "ilton     "
3500 ;
3510 FAST .SBYTE "fast"
3520 SLOW .SBYTE "slow"
3530 BROKE .BYTE "BROKE",0
3540 ;
3550 ;Property Selection Box
3560 ;
3570 BOX .BYTE 125
3580      .BYTE "  Prop:",EOL
3590      .BYTE " Price:          "
3600      .BYTE "Resale:",EOL
3610      .BYTE " Level:          "
3620      .BYTE "Double:",EOL
3630      .BYTE "   Fee:          "
3640      .BYTE " Owner:",0
3650 ;
3660 YES .BYTE "Yes",0
3670 NO  .BYTE "No",0
3680 NA  .BYTE "None",0
3690 ;
3700 ;colors for various screens
3710 ;
3720 TCLR .BYTE $42,$0A,$02,$D4,$00
3730 ICLR .BYTE $94,$06,$00,$42,$00
3740 NCLR .BYTE $C4,$96,$0A,$0F,$00
3750 ;
3760 ;more messages
3770 ;
3780 WINNER .BYTE " has won!"
3790      .BYTE EOL,"Press a ke"
3800      .BYTE "y to play again.",0
3810 DEAD_MES .BYTE " is Broke!",EOL
3820      .BYTE 0
3830 YNTXT .BYTE " property [Y/N]?",0
3840 IMP_TXT .BYTE EOL,EOL,"Improve "
3850      .BYTE "a",0
3860 SELL_TXT .BYTE EOL,EOL,"Sell a"
3870      .BYTE 0
3880 BUY_MESS .BYTE EOL,EOL,"Buy thi"
3890      .BYTE "s",0
3900 PICK_MESS .BYTE EOL,EOL,"SPC=ne"
3910      .BYTE "xt, RET=choose",0
3920 PICK_REST .BYTE ", ESC=exit",0
3930 TRAN_OK .BYTE EOL,EOL,"Tran"
3940      .BYTE "saction processed.",0
3950 NCASH .BYTE EOL,EOL,"Insufficie"
3960      .BYTE "nt funds.",0
3970 DOPT .BYTE EOL,EOL,"Double Opti"
3980      .BYTE "on.",0
3990 MAX .BYTE EOL,EOL,"At Maximum "
4000      .BYTE "Level",0
4010 FEE .BYTE EOL,EOL,"Fee Payment"
4020      .BYTE " Required.",0
4030 AMTOWD .BYTE EOL,EOL,"Amount o"
4040      .BYTE "wed ",0
4050 ;
4060 ;Get the Map Handler Half of
4070 ;the game.
4080 ;
4090      .INCLUDE #D:CAPITAL4.M65
4100      .INCLUDE #D:CAPITAL5.M65
4110 ;
4120 ;
4130 ;Put a byte to the screen
4140 ;
4150 CLRSCR LDA #$7D ;clear scn
4160 EPUT CMP #EOL   ;a CR?
4170      BNE TRYDEL
4180 PUTCR LDA #0    ;linefeed
4190      STA XPOS
4200      INC YPOS
4210      RTS
4220 ;
4230 TRYDEL CMP #$7E ;delete?
4240      BNE TRYCLR
4250      JSR MOVELEFT ;DEL-BCKSP
4260      JSR GETSCR  ;get scrn loc
4270      LDA #0      ;zap char
4280      TAY
4290      STA (SCR),Y
4300      RTS
4310 TRYCLR CMP #$7D ;clear?
4320      BNE NORMAL
4330      LDY #0
4340      TYA         ;clear the
4350 CLRSC STA GR0MEM,Y ;screen
4360      STA GR0MEM+$0100,Y
4370      STA GR0MEM+$0200,Y
4380      INY
4390      BNE CLRSC
```

```
4400      STA XPOS
4410      STA YPOS
4420      RTS
4430 ;
4440 NORMAL JSR ASC2IC ;to INT. code
4450      PHA          ;save it
4460      JSR GETSCR   ;get scn loc
4470      LDY #0
4480      PLA          ;get byte
4490      STA (SCR),Y  ;on screen
4500 ;
4510 GORIGHT LDX XPOS ;move 'cursor'
4520      INX          ;to right
4530      CPX #$20     ;at end of
4540      BCC EPLEAV   ;line?
4550      INC YPOS     ;yes
4560      LDX #0       ;left margin
4570 EPLEAV STX XPOS
4580      RTS
4590 ;
4600 MOVELEFT LDA XPOS ;can we
4610      TAX          ;go left?
4620      ORA YPOS
4630      BEQ GOLEAV   ;no
4640      DEX          ;yes
4650      BPL GOX
4660      LDX #31      ;if $FF go
4670      LDY YPOS     ;up 1 line
4680      BNE MVUP     ;if not 0
4690      LDX #0
4700      BEQ GOX
4710 MVUP DEC YPOS     ;up 1 line
4720 GOX STX XPOS
4730 GOLEAV RTS        ;exit
4740 ;
4750 ;Get Scr Mem Loc of Cursor
4760 ;
4770 GETSCR LDA # <[GR0MEM-32]
4780      STA SCR
4790      LDA # >[GR0MEM-32]
4800      STA SCR+1
4810      LDY YPOS     ;get the
4820 GETSC1 LDA SCR    ;correct
4830      CLC          ;row
4840      ADC #$20     ;narrow plfld
4850      STA SCR
4860      BCC GODWN
4870      INC SCR+1
4880 GODWN DEY         ;at row yet?
4890      BPL GETSC1   ;no
4900      LDA SCR      ;now get the
4910      CLC          ;column
4920      ADC XPOS     ;add in x
4930      STA SCR      ;position
4940      BCC GETSCDN
4950      INC SCR+1
4960 GETSCDN RTS
4970 ;
4980 ;Add Money to Player cash
4990 ;
5000 ADD PHA           ;save $
5010      TYA          ;plyr #
5020      ASL A        ;x2
5030      TAY
5040      PLA          ;get $
5050      CLC
5060      ADC CASH,Y   ;add to
5070      STA CASH,Y   ;plyr's cash
5080      LDA CASH+1,Y
5090      ADC AMOUNT+1 ;if >255
5100      STA CASH+1,Y
5110      JMP FORMDOL  ;show result
5120 ;
5130 ;Subtract Money from player
5140 ;
5150 SUB STA CTMP      ;save $
5160      TYA          ;get plyr #
5170      ASL A        ;x2
5180      TAY
5190      LDA CASH,Y   ;get cash
5200      SEC          ;subt. $
5210      SBC CTMP
5220      STA CASH,Y
5230      LDA CASH+1,Y
5240      SBC AMOUNT+1 ;if $ > 255
5250      STA CASH+1,Y ;fall through
5260 ;
5270 ;Format MONEY line
5280 ;
5290 FORMDOL LDY #39 ;zero out
5300      LDA #0       ;the line
5310 FD1 STA MONEY,Y
5320      DEY
5330      BPL FD1
5340      LDX MAXP     ;max plyrs
5350      LDA XST-2,X  ;for centering
5360      TAX          ;info
5370      INY          ;y was $FF
5380 FD2 TYA           ;plyr #
5390      CLC
5400      ADC #$11     ;for screen IC
5410      STA MONEY,X  ;show it
5420      LDA #$1A     ; '$'
5430      STA MONEY+1,X
5440      STX XT       ;save x,y
5450      STY YT
5460      TYA          ;plyr #
5470      ASL A        ;x2
5480      TAY
5490      LDA CASH+1,Y ;get his cash
5500      TAX          ;x=high byte
5510      LDA CASH,Y   ;a=low
5520      JSR FORMNUM  ;to ATASCII
5530      LDX XT       ;position
5540      LDY #$FF     ;now
5550 FD3 INY           ;change cash
5560      LDA (INBUFF),Y ;to
5570      BEQ FD4      ;if '0' done
5580      JSR ASC2IC   ;internal code
5590      STA MONEY+2,X ;show it
5600      INX
5610      BNE FD3      ;next digit
5620 FD4 LDY MAXP      ;get offset
5630      LDA XT       ;to next
5640      CLC          ;plyr's info
5650      ADC OFFSET-2,Y
5660      TAX
5670      LDY YT
5680      INY          ;next plyr
5690      CPY MAXP     ;last one?
5700      BNE FD2      ;no
5710      RTS          ;whew!
5720 ;
5730 ;Format a number
5740 ;
5750 FORMNUM STA FR0 ;low byte
5760      STX FR0+1    ;hi
5770      ORA FR0+1    ;hello!
5780      BNE GFP
5790      LDA # <BROKE ;if 0 you
5800      STA INBUFF   ;are 'broke'
5810      LDA # >BROKE
5820      STA INBUFF+1
5830      RTS
5840 GFP JSR IFP       ;INT to FP
5850      JSR FASC     ;FP to ATASCII
5860      LDY #$FF
5870 FM1 INY           ;find last
5880      LDA (INBUFF),Y ;inversed
5890      BPL FM1      ;digit
5900      AND #$7F     ;& make it
5910      STA (INBUFF),Y ;regular
5920      INY          ;add a 'G'
5930      LDA #'G
5940      STA (INBUFF),Y ;and an EOL
5950      LDA #0       ;i.e. a 0
5960      INY
5970      STA (INBUFF),Y
5980      RTS          ;done
5990 ;
6000 ;Convert ASCII to ICODE
6010 ;
6020 ASC2IC JSR BITER
6030      ORA A2I,X
6040      LDX XSV2
6050      RTS
6060 ;
6070 ;Convert ICODE to ASC
6080 ;
6090 IC2ASC JSR BITER
6100      ORA I2A,X
6110      LDX XSV2
6120      RTS
6130 ;
6140 ;Make char an index
6150 ;
6160 BITER PHA
6170      ROL A
6180      ROL A
6190      ROL A
6200      ROL A
6210      AND #3
6220      STX XSV2
6230      TAX
6240      PLA
6250      AND #$9F
6260      RTS
6270 ;
6280      .INCLUDE #D:CAPITAL2.M65
6290 ;
6300      *=  $7000
6310 ;the screen data for the
6320 ;map of the U.S.A.
6330 ;
6340      .INCLUDE #D:CAPITAL6.M65
6350      *=  $02E0
6360      .WORD CSET
6370      .END
```

LISTING 3: ASSEMBLY

```
0100 ;SAVE#D:CAPITAL2.M65
0110 ;
0120 ;--------------------;
0130 ;                    ;
0140 ; CAPITAL! part 2    ;
0150 ;                    ;
0160 ;by: Bryan Schappel  ;
0170 ;                    ;
0180 ;--------------------;
0190 ;
0200 ;Get a key press
0210 ;
0220 GETCH LDX #$FF
0230      STX CH
0240 GETC LDA CH
0250      CMP #$FF
0260      BEQ GETC
0270      STX CH
0280      RTS
0290 ;
0300 ;Print a Number
0310 ;
0320 PRNUM JSR FORMNUM ;to IC first
0330      LDA INBUFF  ;fall through
0340      LDY INBUFF+1
0350 ;
0360 ;Eprint Routine
0370 ;
0380 EPRINT STA EPL+1 ;A= low byte
0390      STY EPL+2   ;Y = hi byte
0400 EPL LDA $FFFF    ;print until
```

```
0410      BEQ EPO      ;hitting a 0
0420      JSR EPUT     ;put it
0430      LDA PAUSE?   ;fast or slow
0440      BNE EPN      ;printing?
0450      JSR JIFF     ;wait a jiff
0460 EPN  INC EPL+1    ;next byte
0470      BNE EPL
0480      INC EPL+2
0490      BNE EPL
0500 EPO  RTS          ;exit
0510 ;
0520 ;Intro DLI
0530 ;
0540 IDLI PHA          ;save all the
0550      TXA          ;registers
0560      PHA
0570      TYA
0580      PHA
0590      LDA #$C0     ;green hue
0600      LDY #7       ;dark lum
0610 IDLP LDX #2
0620 IDL1 STA COLPF2 ;plyfld 2
0630      EOR #$0F     ;switch lum
0640      STA COLPF1   ;plyfld 1
0650      EOR #$0F     ;swtich back
0660      STA WSYNC    ;wait sync
0670      DEX          ;3 lines
0680      BPL IDL1
0690      CLC
0700      ADC #2       ;increase lum
0710      DEY          ;8 times
0720      BPL IDLP     ;more shades
0730      LDY #7       ;now start
0740      LDA #$CC     ;with bright
0750 IDL2 LDX #2       ;and shade
0760 IDL3 STA COLPF2
0770      EOR #$0F
0780      STA COLPF1
0790      EOR #$0F
0800      STA WSYNC
0810      DEX
0820      BPL IDL3
0830      SEC          ;bring lum
0840      SBC #2       ;down
0850      DEY
0860      BPL IDL2
0870      LDA #$04     ;gray
0880      STA COLPF1
0890      PLA          ;restore
0900      TAY          ;registers
0910      PLA
0920      TAX
0930      PLA
0940      RTI          ;exit
0950 ;
0960 ;Text Screen DLI
0970 ;
0980 TDLI PHA          ;save A
0990      LDA BDL      ;DLI cntr
1000      BNE TDL2
1010      LDA #58      ;to narrow
1020      STA DMACTL   ;plyfld
1030      LDA #10      ;white
1040      STA COLPF1
1050      STA COLPF0
1060      LDA #MONEY_CLR ;$ line
1070      STA COLPF2
1080      STA WSYNC
1090      STA COLBK    ;& bckgrnd
1100      INC BDL
1110      PLA          ;restore A
1120      RTI          ;exit
1130 TDL2 LDA #0       ;top DLI
1140      STA WSYNC
1150      STA COLBK    ;black bckgnd
1160      PLA
1170      RTI
1180 ;
1190 ;Simple VBI to keep DLI'S OK
1200 ;
1210 VBI  LDA #0
1220      STA ATRACT
1230      STA BDL      ;DLI counter
1240      JMP XITVBV
1250 ;
1260 ;Wait Routine
1270 ;
1280 JIFF PHA          ;wait 1 jiffy
1290      LDA #0
1300      STA RTCLOK
1310 W1   LDA RTCLOK
1320      BEQ W1
1330      PLA
1340      RTS
1350 ;
1360 ;Position Cursor
1370 ;
1380 POSIT STX XPOS
1390      STY YPOS
1400      RTS
1410 ;
1420 ;Intro Routine
1430 ;
1440 INTRO LDA #7      ;set up Vblank
1450      LDX # >VBI
1460      LDY # <VBI
1470      JSR SETVBV
1480      JSR CLRPMG   ;erase PMG
1490      LDA # <IDL   ;install DLIST
1500      STA SDLSTL
1510      LDA # >IDL
1520      STA SDLSTL+1
1530      LDA #62      ;normal plyfd
1540      STA SDMCTL
1550      JSR JIFF     ;wait a jiff
1560      LDA # >MSET  ;our char
1570      STA CHBAS    ;set
1580      LDA # <IDLI  ;install DLI
1590      STA VDSLST
1600      LDA # >IDLI
1610      STA VDSLST+1
1620      LDY #4       ;get intro
1630 INTY LDA ICLR,Y   ;colors
1640      STA COLOR0,Y
1650      DEY
1660      BPL INTY
1670      LDA #3       ;center some
1680      STA HSCROL   ;text
1690      LDA #0       ;0 = fast
1700      STA SPEED    ;game speed
1710      STA CONSAV
1720      LDA #2       ;default #
1730      STA MAXP     ;players
1740      LDA #$25     ;make a tone
1750      JSR BUZZER
1760 INTRL LDX #0      ;get the
1770      LDA SPEED    ;word 'fast'
1780      ASL A        ;or 'slow'
1790      ASL A
1800      TAY
1810 RL1  LDA FAST,Y   ;put it on
1820      STA INTM+10,X ;screen
1830      INY
1840      INX
1850      CPX #4       ;done?
1860      BNE RL1
1870      LDA MAXP     ;# plyrs
1880      ORA #$90     ;in COLOR!
1890      STA PLN+10   ;show it
1900      JSR CONC     ;get CONSOL
1910      LSR A        ;/2
1920      BCC ST2
1930      JMP INTLV    ;START!
1940 ;
1950 ST2  LSR A        ;SELECT?
1960      BCC ST3
1970      LDA #$20     ;tone
1980      JSR BUZZER
1990      INC MAXP     ;more plyrs
2000      LDA MAXP     ;too many?
2010      CMP #5
2020      BCC INTRL    ;no
2030      LDA #2       ;reset to 2
2040      STA MAXP
2050      BNE INTRL    ;looooopppp
2060 ;
2070 ST3  LSR A        ;is it OPTION?
2080      BCC INTRL    ;naw
2090      LDA #$10     ;another tone
2100      JSR BUZZER
2110      LDA SPEED    ;change speed
2120      EOR #1
2130      STA SPEED
2140      JMP INTRL    ;loop de loop
2150 INTLV LDY #3      ;yea, we start
2160      LDA #0       ;zap all
2170 TCL  STA ONBRD,Y
2180      DEY
2190      BPL TCL
2200      INY          ;put 1's for
2210      LDA #1       ;those plyrs
2220 TCL2 STA ONBRD,Y ;getting
2230      INY          ;tokens
2240      CPY MAXP     ;enuf?
2250      BNE TCL2     ;yup
2260      JMP GETNAMES ;who are they?
2270 ;
2280 ;Buzzer sound
2290 ;
2300 BUZZER STA AUDF2 ;doesn't
2310      LDA #$A4     ;sound like
2320      STA AUDC2    ;a buzzer to me
2330      LDY #$C0     ;but ok
2340 BZ   DEX          ;make a sound
2350      BNE BZ
2360      STX ATRACT
2370      DEY
2380      BNE BZ
2390      STY AUDC2
2400      RTS
2410 ;
2420 ;Console Checker
2430 ;
2440 CONC LDA CONSOL ;get button
2450      TAY          ;i wonder
2460      EOR CONSAV   ;what this
2470      AND CONSAV   ;does except
2480      STY CONSAV   ;waste time
2490      CMP #4
2500      RTS
2510 ;
2520 ;Transaction Processor
2530 ;
2540 TRANS LDA # <TDLI ;put in DLI
2550      STA VDSLST
2560      LDA # >TDLI
2570      STA VDSLST+1
2580      LDA # >MSET  ;our char set
2590      STA CHBAS
2600      LDA # <DLTXT ;& our display
2610      STA SDLSTL   ;list
2620      LDA # >DLTXT
2630      STA SDLSTL+1
2640      LDA #61      ;->narrow<-
2650      STA SDMCTL   ;plyfld
2660      JSR CLRSCR   ;clear screen
2670      LDY #4       ;get our
2680 TC   LDA TCLR,Y   ;coloring set
2690      STA COLOR0,Y
2700      DEY
2710      BPL TC
2720      LDA #0       ;0=fast print
2730      STA PAUSE?
2740      STA DISFLAG ;no disaster
2750      JSR MAKE_RAN ;get selling
2760      LDX PNUM     ;prices
2770      LDA PTHP0,X ;get property
2780      STA CPROP    ;landed on
2790      TAY
2800      LDA REFTAB,Y ;what is it?
2810      CMP #4       ;LUCK??
2820      BNE GWHO     ;no who has it?
2830      JSR DO_LUCK ;U Lucky dog!
```

```
2840      JMP OVER     ;skip this
2850 GWHO JSR SHOPROP ;show prop info
2860      LDY CPROP    ;curr prop #
2870      LDA WHO,Y    ;get owner
2880      STA OWNER
2890      BMI TRY_BUY ;$FF=no one
2900      CMP PNUM     ;is it US(ME?)
2910      BEQ OVER     ;yes
2920      JSR PAY_RENT ;Ugh pay rent
2930      JMP OVER     ;skip buying
2940 TRY_BUY JSR BUY_IT ;wanna buy
2950      BNE OVER     ;a bridge?
2960      LDY CPROP    ;check for a
2970      LDA XREF,Y   ;double
2980      BEQ OVER     ;nope
2990      STA CPROP    ;save it
3000      TAY
3010      LDA WHO,Y    ;who owns it?
3020      BPL OVER     ;+ = owned
3030      LDA COST,Y   ;how much? is
3040      PHA          ;that double
3050      LSR A
3060      CLC          ;add 1/2 again
3070      ADC COST,Y   ;as much for
3080      STA COST,Y   ;doubles
3090      JSR SHOPROP ;show prop info
3100       PRINT  DOPT ;'Doubl option'
3110      JSR BUY_IT   ;'wanna buy?'
3120      LDY CPROP    ;location
3130      PLA          ;restore origin-
3140      STA COST,Y   ;al cost of prop
3150 OVER.JSR WANT_SALE ;sell a prop
3160      JMP IMPROVE ;improve one?
3170 ;
3180 ;Handle Player Death
3190 ;
3200 DEAD JSR PUTCR   ;linefeeds
3210      JSR PUTCR
3220      LDA PNUM     ;get plyr #
3230      PHA          ;save it
3240      TAX
3250      DEC ONBRD,X ;make 0 to
3260      TAY          ;remove
3270      JSR SHO_NAME ;who was it?
3280       PRINT  DEAD_MES ;'gone'
3290      PLA          ;plyr # again
3300      ASL A        ;x2
3310      TAY          ;zap his/her
3320      LDA #0       ;cash
3330      STA CASH,Y
3340      STA CASH+1,Y
3350      LDX #35      ;sell props
3360 SELL_OFF LDA WHO,X ;back to
3370      CMP PNUM     ;the bank
3380      BNE SELL_DN
3390      LDA #$FF     ;$FF = free
3400      STA WHO,X    ;property
3410      LDA RANTAB,X ;new cost
3420      STA COST,X   ;is half price
3430      DEC REFTAB,X ;what a
3440 SELL_DN DEX       ;bargain!
3450      BPL SELL_OFF
3460      JSR FORMDOL ;show $ line
3470      JSR WAITKEY ;pause
3480      LDX #0       ;is anyone
3490      LDY #3       ;left-if so
3500 FNW LDA ONBRD,Y ;s/he is
3510      BEQ FNW2     ;the winner
3520      INX
3530 FNW2 DEY
3540      BPL FNW
3550      CPX #2       ;LT 2 means
3560      BCC SHO_WIN ;a winner
3570      JMP NOW      ;still going
3580 ;
3590 ;Show the winner
3600 ;
3610 SHO_WIN JSR PUTCR ;linefeed
3620      LDY #3
3630 SW1 LDA ONBRD,Y ;get plyr's #
3640      BNE GWN
3650      DEY
3660      BPL SW1
3670 GWN JSR SHO_NAME ;winner's name
3680       PRINT  WINNER ;'wins'
3690      JSR GETCH    ;get a key
3700      JMP FAKE     ;restart
3710 ;
3720 ;Get a RND # between 1 and Acc
3730 ;
3740 GET_RND STA RNUM ;save A
3750 RND LDA RANDOM
3760      AND #$7F     ;0-127
3770      BEQ RND      ;no 0
3780      CMP RNUM     ;LT A?
3790      BEQ RNDO     ;yup
3800      BCS RND      ;overs
3810 RNDO RTS          ;done
3820 ;
3830 ;Make the random property
3840 ;value table. Values are bet-
3850 ;ween 1/2 and full price.
3860 ;
3870 MAKE_RAN LDY #35 ;36 props
3880 MK1 LDA COST,Y   ;if 0 no
3890      BEQ NORAN
3900      LSR A        ;div by 2
3910      STA VALUE    ;save it
3920      JSR GET_RND ;get rand #
3930      CLC          ;LT value
3940      ADC VALUE    ;add to 1/2
3950 NORAN STA RANTAB,Y ;price
3960      DEY          ;next prop
3970      BPL MK1
3980      LDA DISFLAG ;if forced
3990      BEQ MKO      ;sale,just
4000      LDY #35      ;make prices
4010 MK2 LDA COST,Y   ;1/2 price
4020      LSR A
4030      STA RANTAB,Y
4040      DEY
4050      BPL MK2
4060 MKO RTS
4070 ;
4080 ;Show a Property
4090 ;
4100 SHOPROP INC PAUSE? ;fast print
4110       PRINT  BOX ;print
4120       POSITION  7,0 ;outline
4130      LDA CPROP    ;curr. prop
4140      ASL A        ;get name
4150      TAX          ;x2 for
4160      LDA PRPTAB+1,X ;offset from
4170      TAY          ;table
4180      LDA PRPTAB,X ;A=hi Y=lo
4190      JSR INVPROP ;inverse
4200       POSITION  8,1 ;name
4210      LDX #0       ;X=lo cost
4220      LDY CPROP
4230      LDA COST,Y   ;A = hi
4240      JSR PRNUM    ;show cost
4250      LDA #24      ;move cursor
4260      STA XPOS
4270      LDX #0
4280      LDY CPROP    ;get prop#
4290      LDA RANTAB,Y ;sale price
4300 SPQ JSR PRNUM     ;show it
4310      LDY CPROP    ;get improve-
4320      LDA IMP,Y    ;ment level
4330      ORA #$10     ;for screen
4340      STA GR0MEM+72 ;show it
4350       POSITION  24,2
4360      LDY CPROP    ;see if doubl
4370      LDA XREF,Y
4380      BNE SPY      ;yes
4390       PRINT  NO   ;no
4400      JMP SPO      ;skip
4410 SPY  PRINT  YES
4420 SPO  POSITION  8,3
4430      LDY CPROP    ;who owns it
4440      LDA WHO,Y
4450      STA SVY      ;save it
4460      LDA RENT,Y   ;get 'rent'
4470      STA VALUE    ;= fee
4480      LDA XREF,Y   ;double?
4490      BEQ SHV      ;no
4500      TAX          ;who owns doubl?
4510      LDA WHO,X    ;owner
4520      CMP SVY      ;same as ME?
4530      BNE SHV      ;no
4540      LDA RENT,X   ;yes so
4550      CLC          ;rent is more
4560      ADC VALUE
4570      STA VALUE
4580 SHV LDA VALUE     ;show rent
4590      LDX #0       ;A=hi,X=lo
4600      JSR PRNUM
4610      LDA #24
4620      STA XPOS
4630      JSR GETSCR   ;screen loc
4640      LDY CPROP    ;prop #
4650      LDA WHO,Y    ;owner?
4660      TAY          ;get name
4670      JSR SHO_NAME
4680      DEC PAUSE?   ;slow print
4690      LDX #0       ;position
4700      LDY #4       ;cursor
4710      JMP POSIT    ;& exit
4720 ;
4730 ;Show a player's name
4740 ;Y=player # to show
4750 ;
4760 SHO_NAME LDA X9,Y ;get right
4770      STA L        ;name
4780      LDA # >NAMEBUF ;via offset
4790      STA L+1
4800      CPY #$FF     ;is it no one?
4810      BNE SNM0     ;no someone
4820      LDA # <NA    ;'none'
4830      STA L
4840      LDA # >NA
4850      STA L+1
4860 SNM0 LDA L        ;print name
4870      LDY L+1
4880      JMP EPRINT
4890 ;
4900 ;Initialize game tables
4910 ;
4920 INITAB LDY #35    ;36 locations
4930 IT1 LDA #$FF      ;$FF = no
4940      STA WHO,Y    ;owner
4950      LDA #0       ;0 = no
4960      STA IMP,Y    ;inprovement
4970      DEY
4980      BPL IT1
4990      LDY #115     ;copy all
5000 IT2 LDA RENTORG,Y ;preset
5010      STA RENT,Y   ;tables to
5020      DEY          ;working tables
5030      BPL IT2
5040      LDY #0       ;download
5050 MVSET LDA $E000,Y ;ROM
5060      STA MSET,Y   ;character
5070      LDA $E100,Y  ;set
5080      STA MSET+256,Y
5090      LDA $E200,Y
5100      STA MSET+512,Y
5110      LDA $E300,Y
5120      STA MSET+768,Y
5130      INY
5140      BNE MVSET
5150      LDY #7       ;all this to
5160      LDA #$FF     ;define a
5170 ASET STA MSET+504,Y ;block
5180      DEY          ;cursor for
5190      BPL ASET     ;name screen!!!
5200      LDA # <FAKE ;SYSTEM RESET
5210      STA DOSINI   ;restarts
5220      LDA # >FAKE ;the game
5230      STA DOSINI+1
5240      LDA #$C0     ;enable DLI's
5250      STA NMIEN
5260      RTS
```

LISTING 4: ASSEMBLY

```
0100 ;SAVE#D:CAPITAL3.M65
0110 ;
0120 ;-------------------;
0130 ;     CAPITAL!      ;
0140 ; redefined char.   ;
0150 ; set for the MAP   ;
0160 ;                   ;
0170 ; by Barry Kolbe    ;
0180 ;                   ;
0190 ;-------------------;
0200 ;
0210 ;includes most of a character
0220 ;set for ANTIC mode 4.
0230 ;
0240      .BYTE $00,$00,$00,$00
0250      .BYTE $00,$00,$00,$00
0260      .BYTE $FD,$F5,$D7,$D7
0270      .BYTE $D7,$D7,$F5,$FD
0280      .BYTE $7F,$5F,$D7,$D7
0290      .BYTE $D7,$D7,$5F,$7F
0300      .BYTE $FD,$F7,$DF,$DF
0310      .BYTE $DF,$DF,$F7,$FD
0320      .BYTE $7F,$DF,$F7,$F7
0330      .BYTE $F7,$F7,$DF,$7F
0340      .BYTE $D5,$D5,$D7,$D7
0350      .BYTE $D7,$D7,$D5,$D5
0360      .BYTE $57,$57,$D7,$D7
0370      .BYTE $D7,$D7,$57,$57
0380      .BYTE $FF,$FA,$EA,$EB
0390      .BYTE $EB,$EB,$EA,$FA
0400      .BYTE $FF,$AF,$AF,$EF
0410      .BYTE $FF,$EF,$AF,$AF
0420      .BYTE $55,$55,$55,$55
0430      .BYTE $55,$55,$55,$55
0440      .BYTE $55,$55,$55,$69
0450      .BYTE $69,$55,$55,$55
0460      .BYTE $FF,$EA,$EE,$FE
0470      .BYTE $FE,$FE,$FE,$FA
0480      .BYTE $FF,$AB,$BB,$BF
0490      .BYTE $BF,$BF,$BF,$AF
0500      .BYTE $FF,$AB,$EB,$EA
0510      .BYTE $EA,$EB,$EB,$AB
0520      .BYTE $FF,$EA,$EB,$AB
0530      .BYTE $AB,$EB,$EB,$EA
0540      .BYTE $FF,$FE,$EA,$EB
0550      .BYTE $EA,$FF,$EA,$FE
0560      .BYTE $FF,$BF,$AB,$FF
0570      .BYTE $AB,$EB,$AB,$BF
0580      .BYTE $D5,$DF,$DF,$DF
0590      .BYTE $DF,$DF,$DF,$D5
0600      .BYTE $57,$F7,$F7,$F7
0610      .BYTE $F7,$F7,$F7,$57
0620      .BYTE $FF,$EA,$EB,$EB
0630      .BYTE $EB,$EB,$EA,$EA
0640      .BYTE $FF,$FF,$FF,$FF
0650      .BYTE $FF,$EF,$AF,$AF
0660      .BYTE $5F,$5F,$5F,$5F
0670      .BYTE $5F,$5F,$5F,$5F
0680      .BYTE $F5,$F5,$F5,$F5
0690      .BYTE $F5,$F5,$F5,$F5
0700      .BYTE $EF,$BF,$FF,$FB
0710      .BYTE $EE,$BF,$FE,$FF
0720      .BYTE $FF,$FF,$FF,$FF
0730      .BYTE $FF,$FF,$FF,$FF
0740      .BYTE $00,$FF,$FF,$FF
0750      .BYTE $FF,$FF,$FF,$FF
0760      .BYTE $00,$00,$FF,$FF
0770      .BYTE $FF,$FF,$FF,$FF
0780      .BYTE $00,$00,$00,$FF
0790      .BYTE $FF,$FF,$FF,$FF
0800      .BYTE $00,$00,$00,$00
0810      .BYTE $FF,$FF,$FF,$FF
0820      .BYTE $00,$00,$00,$00
0830      .BYTE $00,$FF,$FF,$FF
0840      .BYTE $00,$00,$00,$00
0850      .BYTE $00,$00,$FF,$FF
0860      .BYTE $00,$00,$00,$00
0870      .BYTE $00,$00,$00,$FF
0880      .BYTE $C0,$C0,$F0,$F0
0890      .BYTE $FC,$FC,$FF,$FF
0900      .BYTE $F0,$F0,$F0,$F0
0910      .BYTE $FC,$FC,$FC,$FC
0920      .BYTE $03,$03,$0F,$0F
0930      .BYTE $3F,$3F,$FF,$FF
0940      .BYTE $0F,$0F,$0F,$0F
0950      .BYTE $3F,$3F,$3F,$3F
0960      .BYTE $FF,$FF,$FC,$FC
0970      .BYTE $FC,$F0,$F0,$C0
0980      .BYTE $FC,$FC,$FC,$FC
0990      .BYTE $F0,$F0,$F0,$F0
1000      .BYTE $FF,$FF,$BF,$BF
1010      .BYTE $AF,$2F,$2B,$0B
1020      .BYTE $BF,$BF,$BF,$BF
1030      .BYTE $2F,$2F,$2F,$2F
1040      .BYTE $FF,$FF,$FF,$FF
1050      .BYTE $FF,$FF,$FF,$AA
1060      .BYTE $FF,$FF,$FF,$FF
1070      .BYTE $FF,$FF,$AA,$AA
1080      .BYTE $FF,$FF,$FF,$FF
1090      .BYTE $FF,$AA,$AA,$00
1100      .BYTE $BF,$BF,$BF,$BF
1110      .BYTE $BF,$BF,$BF,$BF
1120      .BYTE $A8,$A8,$00,$00
1130      .BYTE $00,$00,$00,$00
1140      .BYTE $FE,$FA,$F0,$F0
1150      .BYTE $C0,$C0,$C0,$C0
1160      .BYTE $CF,$CF,$CF,$CF
1170      .BYTE $CF,$CF,$3F,$3F
1180      .BYTE $3F,$3F,$3F,$3F
1190      .BYTE $FF,$FF,$FF,$FF
1200      .BYTE $FF,$FF,$FF,$FF
1210      .BYTE $3F,$3F,$3F,$3F
1220      .BYTE $FF,$FF,$FF,$FF
1230      .BYTE $FF,$FF,$3F,$0F
1240      .BYTE $F0,$F0,$F0,$F0
1250      .BYTE $F0,$C0,$C0,$00
1260      .BYTE $FC,$FC,$FC,$FC
1270      .BYTE $FC,$FC,$FC,$FC
1280      .BYTE $03,$03,$03,$03
1290      .BYTE $0F,$0F,$0F,$0F
1300      .BYTE $3F,$3F,$3F,$3F
1310      .BYTE $FF,$FF,$FF,$FF
1320      .BYTE $C0,$C0,$C0,$C0
1330      .BYTE $F0,$F0,$FC,$FF
1340      .BYTE $03,$03,$03,$03
1350      .BYTE $0F,$0F,$3F,$FF
1360      .BYTE $C0,$C0,$C0,$C0
1370      .BYTE $C0,$C0,$C0,$C0
1380      .BYTE $03,$03,$03,$03
1390      .BYTE $03,$03,$03,$03
1400      .BYTE $00,$00,$00,$03
1410      .BYTE $03,$0F,$0F,$3F
1420      .BYTE $F0,$FC,$FF,$FF
1430      .BYTE $FC,$FF,$FC,$FC
1440      .BYTE $F0,$FC,$FC,$FC
1450      .BYTE $FC,$FC,$FC,$F0
1460      .BYTE $FC,$F0,$C0,$C0
1470      .BYTE $00,$00,$00,$00
1480      .BYTE $FC,$F0,$F0,$F0
1490      .BYTE $C0,$C0,$C0,$C0
1500      .BYTE $C0,$C0,$C0,$C0
1510      .BYTE $F0,$F0,$FC,$FC
1520      .BYTE $C0,$C0,$F0,$F0
1530      .BYTE $F0,$FC,$FC,$FC
1540      .BYTE $FC,$FC,$FC,$F0
1550      .BYTE $F0,$F0,$C0,$C0
1560      .BYTE $FC,$F0,$C0,$F0
1570      .BYTE $FC,$FC,$FF,$FF
1580      .BYTE $C0,$00,$00,$00
1590      .BYTE $C0,$C0,$C0,$C0
1600      .BYTE $0B,$0B,$0B,$0B
1610      .BYTE $0B,$0B,$0F,$0F
1620      .BYTE $0F,$0F,$0F,$3F
1630      .BYTE $3F,$3F,$FF,$FF
1640      .BYTE $00,$00,$03,$03
1650      .BYTE $03,$0F,$0F,$3F
1660      .BYTE $00,$30,$BC,$BF
1670      .BYTE $2F,$2F,$0B,$0B
1680      .BYTE $0F,$0F,$0F,$CF
1690      .BYTE $FF,$FF,$FF,$FF
1700      .BYTE $0C,$3C,$F0,$FF
1710      .BYTE $FF,$FF,$FF,$FF
1720      .BYTE $00,$00,$00,$00
1730      .BYTE $FC,$F0,$00,$00
1740      .BYTE $FF,$FF,$FF,$FF
1750      .BYTE $FF,$F0,$C0,$C0
1760      .BYTE $F0,$C0,$00,$00
1770      .BYTE $00,$00,$00,$00
1780      .BYTE $FF,$FF,$FF,$FF
1790      .BYTE $FF,$FF,$FF,$FC
1800      .BYTE $FF,$BF,$BF,$BF
1810      .BYTE $AF,$2F,$2B,$0A
1820      .BYTE $3F,$3F,$3F,$3F
1830      .BYTE $3F,$3F,$FF,$FF
1840      .BYTE $FC,$FC,$F0,$F0
1850      .BYTE $FC,$FC,$FF,$FF
1860      .BYTE $00,$00,$00,$00
1870      .BYTE $03,$0F,$3F,$FF
1880      .BYTE $00,$03,$0F,$3F
1890      .BYTE $FF,$FF,$FF,$FF
1900      .BYTE $00,$00,$00,$00
1910      .BYTE $03,$0F,$0F,$3F
1920      .BYTE $F0,$F0,$F0,$F0
1930      .BYTE $C0,$C0,$C0,$C0
1940      .BYTE $AA,$AA,$00,$00
1950      .BYTE $00,$00,$00,$00
1960      .BYTE $0A,$0A,$00,$00
1970      .BYTE $00,$00,$00,$00
1980      .BYTE $02,$02,$00,$00
1990      .BYTE $00,$00,$00,$00
```

LISTING 5: ASSEMBLY

```
0100 ;SAVE#D:CAPITAL4.M65
0110 ;
0120 ;-----------------;
0130 ;     CAPTIAL!    ;
0140 ;                 ;
0150 ; PMG & U.S.A. MAP ;
0160 ;                 ;
0170 ; by: Barry Kolbe ;
0180 ;                 ;
0190 ;-----------------;
0200 ;
0210 ;Set up PMG
0220 ;
0230 SETPMG LDA #$3E ;enable PMG
0240      STA SDMCTL
0250      LDA #1       ;set priority
0260      STA GPRIOR
0270      LDX #3       ;set size to 0
0280      LDA #0
0290      STA BDL
0300 PLS STA SIZEP0,X
0310      DEX
0320      BPL PLS
0330      LDA #3
0340      STA GRACTL
0350      LDA # >PMB   ;set PMG base
0360      STA PMBASE
```

```
0370 ;
0380 ;Initially put Players (Tokens)
0390 ;beneath the DICE
0400 ;
0410      LDX #3
0420 PP5 LDA #$78     ;vertical pos
0430      STA PY0,X
0440      LDA IXP,X   ;initial horz
0450      STA PX0,X   ;position
0460      DEX
0470      BPL PP5
0480 ;
0490 ;Set Token Colors
0500 ;
0510      LDX #3
0520 PC1 LDA PCLR,X
0530      STA PCOLR0,X
0540      LDA #0      ;all begin at
0550      STA PTHP0,X ;'Stock Market'
0560      DEX
0570      BPL PC1
0580      LDA #0      ;player 1
0590      STA PNUM    ;starts
0600 ;
0610 RSTPMG LDA #$3E ;reset PMG
0620      STA SDMCTL  ;for MAP
0630      LDA #1      ;screen
0640      STA GPRIOR
0650      RTS
0660 ;
0670 ;Initial X Positions of Tokens
0680 ;
0690 IXP .BYTE $60,$68,$70,$78
0700 ;
0710 ;Put P/M on screen
0720 ;
0730 SHOWPM LDA ONBRD ;is '1' alive?
0740      BEQ J1      ;nope
0750      JSR DEFP0   ;show '1'
0760 J1  LDA ONBRD+1 ;'2' on?
0770      BEQ J2      ;etc.
0780      JSR DEFP1
0790 J2  LDA ONBRD+2
0800      BEQ J3
0810      JSR DEFP2
0820 J3  LDA ONBRD+3
0830      BEQ J4
0840      JSR DEFP3
0850 J4  RTS
0860 ;
0870 ;Entry Point While Playing
0880 ;
0890 FAKE CLD         ;clear decimal
0900      LDX #$FF    ;reset stack
0910      TXS
0920      JSR INITAB  ;redo tables
0930      JSR SNDOFF  ;init snd
0940      JSR INTRO   ;show intro
0950      JSR SETPMG  ;set up PMG
0960      JSR CLRPMG  ;clear PMG mem
0970      JSR FORMDOL ;show $ line
0980      JSR RESROL  ;DICE mesg
0990      JSR SHWPUR  ;show 'free'
1000 ;
1010 ;Entry point of MAP moves
1020 ;
1030 ENTRY LDA # <DL ;display list
1040      STA SDLSTL
1050      LDA # >DL
1060      STA SDLSTL+1
1070      LDA # >CSET ;char. set
1080      STA CHBAS
1090      JSR RSTPMG  ;reset PMG
1100      LDX #4      ;get MAP
1110 MC1 LDA MAPCL,X ;colors
1120      STA COLOR0,X
1130      DEX
1140      BPL MC1
1150      LDA #0      ;DLI cntr
1160      STA BDL
1170      STA AMOUNT+1 ;safety
1180      LDA # <DLI  ;set up DLI
1190      STA VDSLST
1200      LDA # >DLI
1210      STA VDSLST+1
1220      JSR SHOWPM  ;put tokens on
1230 ;
1240      LDX PNUM    ;get plyr #
1250      LDA ONBRD,X ;alive?
1260      BNE MV1     ;yes
1270      JMP NXP     ;no do next
1280 MV1 LDY #0       ;put plyr name
1290      LDA X9,X    ;on screen
1300      STA SCR
1310      LDA # >NAMEBUF
1320      STA SCR+1
1330 GNL LDA (SCR),Y ;get length
1340      BEQ GNE     ;of name
1350      INY
1360      BNE GNL
1370 GNE STY SVY      ;save Y
1380      LDA #20     ;20 bytes/line
1390      SEC         ;subt from 20
1400      SBC SVY
1410      LSR A       ;1/2
1420      TAX         ;use as index
1430      LDY #19     ;0 out line
1440      LDA #0      ;20 bytes
1450 GNC STA PLRN,Y
1460      DEY
1470      BPL GNC
1480      INY
1490      LDA #59     ;put '[#]'
1500      STA PLRN-4,X ;on screen
1510      LDA #61
1520      STA PLRN-2,X
1530      LDA PNUM    ;get plyr #
1540      CLC
1550      ADC #$11    ;for screen
1560      STA PLRN-3,X ;show #
1570 GNP LDA (SCR),Y ;next the name
1580      BEQ GNU     ;done on 0
1590      JSR ASC2IC  ;INT code
1600      STA PLRN,X  ;show char
1610      INX
1620      INY
1630      BNE GNP
1640 GNU LDA #53      ;' UP '
1650      STA PLRN+1,X
1660      LDA #48
1670      STA PLRN+2,X
1680 SCC LDA CONSOL  ;wait for
1690      CMP #3      ;OPTION
1700      BNE SCC
1710      JSR ROLDIC  ;roll 'em
1720 ;
1730 ;Move the Tokens
1740 ;
1750      LDA PNUM    ;plyr #
1760      ASL A       ;x2 for table
1770      TAX
1780      LDA ETAB,X  ;erase table
1790      STA EJSR+1
1800      LDA ETAB+1,X
1810      STA EJSR+2
1820      LDA DTAB,X  ;define table
1830      STA DJSR+1
1840      LDA DTAB+1,X
1850      STA DJSR+2
1860 ;
1870 MAIN DEC SQRS    ;# to move
1880 EJSR JSR ERSP0   ;erase token
1890      LDX PNUM    ;plyr #
1900      INC PTHP0,X ;next square
1910      LDA PTHP0,X ;all the way
1920      CMP #36     ;around yet?
1930      BNE MOK     ;no
1940      LDA #0      ;yes reset to 0
1950      STA PTHP0,X
1960 MOK LDA PTHP0,X ;find new
1970      TAY         ;position of
1980      LDA PTHX,Y  ;token(Player)
1990      STA PX0,X
2000      LDA PTHY,Y
2010      STA PY0,X
2020 DJSR JSR DEFP0   ;show token
2030      JSR BEEP    ;make a sound
2040      LDX PNUM    ;plyr #
2050      LDA PTHP0,X ;see if on $
2060      BNE MKK     ;no
2070      LDX #0      ;scroll message
2080      JSR MESSAG
2090      LDA #12     ;add 12G to
2100      LDY PNUM    ;player
2110      JSR ADD
2120 MKK JSR WAIT     ;wait a little
2130      JSR SHWPRP  ;show name
2140      LDA SQRS    ;done moving?
2150      BNE MAIN    ;no
2160      LDX PNUM    ;get location
2170      LDA PTHP0,X
2180 ;
2190 ;dont wait on specials
2200 ;
2210      LDY #3
2220 SCM CMP SPLC,Y   ;is it LUCK
2230      BEQ NOW     ;HILTON, TAX
2240      DEY         ;CAP GAINS
2250      BPL SCM
2260 ;
2270 ;This is a square that requires
2280 ;a transaction, so we remove
2290 ;the P/M and call the TRANS
2300 ;routine.  Then we restore the
2310 ;MAP screen display and loop.
2320 ;
2330      JSR WAITKEY ;wait or key
2340      JSR CLRPMG  ; remove PMG
2350      JSR TRANS   ;go to TRANS
2360 ;
2370 ;NOW is entry spot from T_OK
2380 ;
2390 NOW JSR RESROL   ;reset ROLL DICE
2400      LDX PNUM    ;plyr #
2410      LDA PTHP0,X ;location
2420      BEQ COV     ;if it is $
2430      JSR CKMES   ;other specials?
2440 COV JSR SHWPUR   ;show name
2450 NXP INC PNUM     ;next plyr
2460      LDA PNUM    ;only 0-3
2470      AND #3
2480      STA PNUM    ;next person's
2490      JMP ENTRY   ;move
2500 ;
2510 ;Clear PMG Memory
2520 ;
2530 CLRPMG LDX #0
2540      TXA
2550 CP1 STA P0,X
2560      STA P1,X
2570      STA P2,X
2580      STA P3,X
2590      INX
2600      BNE CP1
2610      LDX #3      ;put plyrs off
2620 CP2 STA HPOSP0,X ;screen
2630      DEY
2640      BPL CP2
2650      RTS
2660 ;
2670 ;Four routines to define
2680 ;each of the four players
2690 ;vertically & horizontally
2700 ;
2710 DEFP0 LDX #0     ;define
2720      LDY PY0     ;player 0
2730 DF0 LDA P0DEF,X ;vertically
2740      STA P0,Y
2750      INY
2760      INX
2770      CPX #13
2780      BNE DF0     ;and set its
2790      LDA PX0     ;horizontal
```

```
2800      STA HPOSP0  ;position
2810      RTS
2820 ;
2830 DEFP1 LDX #0
2840      LDY PY0+1
2850 DF1 LDA P1DEF,X
2860      STA P1,Y
2870      INY
2880      INX
2890      CPX #13
2900      BNE DF1
2910      LDA PX0+1
2920      STA HPOSP0+1
2930      RTS
2940 ;
2950 DEFP2 LDX #0
2960      LDY PY0+2
2970 DF2 LDA P2DEF,X
2980      STA P2,Y
2990      INY
3000      INX
3010      CPX #13
3020      BNE DF2
3030      LDA PX0+2
3040      STA HPOSP0+2
3050      RTS
3060 ;
3070 DEFP3 LDX #0
3080      LDY PY0+3
3090 DF3 LDA P3DEF,X
3100      STA P3,Y
3110      INY
3120      INX
3130      CPX #13
3140      BNE DF3
3150      LDA PX0+3
3160      STA HPOSP0+3
3170      RTS
3180 ;
3190 ;Four routines to erase each
3200 ;of the four players
3210 ;
3220 ERSP0 LDY PY0    ;get vert.
3230      LDA #0      ;position
3240      LDX #12     ;erase the
3250 EP0 STA P0,Y     ;13 bytes
3260      INY
3270      DEX
3280      BPL EP0
3290      RTS
3300 ;
3310 ERSP1 LDY PY0+1
3320      LDA #0
3330      LDX #12
3340 EP1 STA P1,Y
3350      INY
3360      DEX
3370      BPL EP1
3380      RTS
3390 ;
3400 ERSP2 LDY PY0+2
3410      LDA #0
3420      LDX #12
3430 EP2 STA P2,Y
3440      INY
3450      DEX
3460      BPL EP2
3470      RTS
3480 ;
3490 ERSP3 LDY PY0+3
3500      LDA #0
3510      LDX #12
3520 EP3 STA P3,Y
3530      INY
3540      DEX
3550      BPL EP3
3560      RTS
3570 ;
3580 ;Table of Erase Players Routines
3590 ;
3600 ETAB .WORD ERSP0
3610      .WORD ERSP1
3620      .WORD ERSP2
3630      .WORD ERSP3
3640 ;
3650 ;
3660 ;Table of Define Player Routines
3670 ;
3680 DTAB .WORD DEFP0
3690      .WORD DEFP1
3700      .WORD DEFP2
3710      .WORD DEFP3
3720 ;
3730 ;Change Character Sets in DLI
3740 ;
3750 DLI PHA
3760      LDA BDL     ;which DLI?
3770      BNE BL2     ;not 1st
3780      LDA # >MSET ;switch to
3790      STA CHBASE  ;our ROM set
3800      STA WSYNC
3810      LDA #$82    ;change colors
3820      STA COLPF2  ;blue
3830      LDA #$DA    ;bright green
3840      STA COLPF1
3850      LDA #$E6    ;Orng Green
3860      STA COLPF3
3870      INC BDL     ;next DLI
3880      PLA         ;routine
3890      RTI
3900 BL2 CMP #1       ;next DLI?
3910      BNE BL3     ;no
3920      LDA #MONEY_CLR ;color
3930      STA WSYNC   ;for $ line
3940      STA COLPF2
3950      LDA #0      ;backgrnd
3960      STA COLBK   ;scan line
3970      INC BDL     ;next DLI
3980      PLA
3990      RTI
4000 BL3 LDA #MONEY_CLR ;bottom
4010      STA WSYNC   ;all one
4020      STA COLBK   ;color
4030      PLA
4040      RTI
4050 ;
4060 ;Roll DICE with OPTION
4070 ;stop them with SELECT
4080 ;
4090 ROLDIC LDA #0    ;sqrs=# to move
4100      STA SQRS
4110      JSR GETRND  ;get random #
4120      LDX #0      ;now show it
4130      LDY #0      ;on DIE # 1
4140 DR1 LDA (L),Y    ;move 3 bytes
4150      STA DICPOS-$27,X ;to
4160      INX         ;screen:x is
4170      INY         ;offset
4180      CPY #3
4190      BNE DR1
4200      LDX #0      ;move 3 more
4210 DR2 LDA (L),Y    ;1 line down
4220      STA DICPOS+1,X
4230      INX
4240      INY
4250      CPY #6
4260      BNE DR2
4270      LDX #0      ;& 3 more
4280 DR3 LDA (L),Y
4290      STA DICPOS+$29,X ;another
4300      INX         ;line down
4310      INY
4320      CPY #9
4330      BNE DR3
4340      JSR GETRND  ;now do 2nd
4350      LDY #0      ;DIE
4360      LDX #5      ;move x over
4370 DT1 LDA (L),Y
4380      STA DICPOS-$27,X
4390      INX
4400      INY
4410      CPY #3      ;3 bytes from
4420      BNE DT1     ;dice definition
4430      LDX #7      ;table
4440 DT2 LDA (L),Y
4450      STA DICPOS-1,X
4460      INX
4470      INY
4480      CPY #6      ;3 more
4490      BNE DT2
4500      LDX #5      ;next line
4510 DT3 LDA (L),Y
4520      STA DICPOS+$29,X
4530      INX
4540      INY
4550      CPY #9
4560      BNE DT3
4570      STA CONSOL  ;clear CONSOL
4580      JSR WAIT
4590      LDA CONSOL  ;wait for
4600      CMP #5      ;SELECT
4610      BNE ROLDIC  ;keep rollin'
4620      RTS
4630 ;
4640 ;Get Random DICE Throws
4650 ;
4660 GETRND LDA RANDOM ;only 0-5
4670      AND #7
4680      CMP #6
4690      BCS GETRND
4700      TAX         ;current throw
4710      CLC
4720      ADC SQRS    ;0-5
4730      STA SQRS
4740      INC SQRS    ;1-6
4750      TXA
4760      ASL A       ;x2 for table
4770      TAX
4780      LDA RTAB,X  ;get right
4790      STA L       ;bytes for
4800      LDA RTAB+1,X ;DICE
4810      STA L+1
4820      RTS
4830 ;
4840 ;Wait a few jiffies
4850 ;
4860 WAIT LDA #0
4870      STA RTCLOK
4880 WA  LDA RTCLOK
4890      CMP #6
4900      BCC WA
4910      RTS
4920 ;
4930 ;Wait for 8 seconds or until
4940 ;a key is pressed.
4950 ;
4960 WAITKEY LDX #$FF ;clear CH
4970      STX CH
4980      LDA CH      ;read keyboard
4990      CMP #$FF    ;unitl released
5000      BNE WAITKEY
5010      LDA #0      ;set clock
5020      STA RTCLOK
5030      STA RTCLOK-1
5040 WK  LDA CH       ;key pressed?
5050      CMP #$FF
5060      BNE WTDON   ;yes-exit
5070      LDA RTCLOK-1 ;no see if
5080      CMP #2      ;time is up
5090      BCC WK
5100 WTDON STX CH     ;clear CH
5110      RTS
5120 ;
5130 ;Initialixe Sound Registers
5140 ;
5150 SNDOFF LDX #7
5160      LDA #0
5170 SN1 STA AUDF1,X
5180      DEX
5190      BPL SN1
5200      STA $D208
5210      RTS
5220 ;
```

```
5230 ;Make a BEEP sound
5240 ;
5250 BEEP LDA #$85
5260      STA AUDF1
5270      LDA #$A8
5280      STA AUDF1+1
5290      JSR WAIT
5300      LDA #0
5310      STA AUDF1
5320      STA AUDF1+1
5330      RTS
5340 ;
5350 ;Scroll Messages for Special
5360 ;Properties. X = msg #
5370 ;
5380 MESSAG LDA SCTAB,X ;how far to
5390      PHA           ;scroll
5400      STA SCNT      ;save it
5410      JSR RESROL    ;reset line
5420 SC4 INC TWIN       ;coarse scroll
5430      BNE SC1
5440      INC TWIN+1
5450 SC1 DEC SCNT       ;decrease cntr
5460      JSR JIFF      ;wait a jiffy
5470      LDA SCNT      ;done yet?
5480      BNE SC4       ;no
5490      JSR WAITKEY   ;wait 8 secs
5500      PLA           ;amount to
5510      STA SCNT      ;scroll back
5520 SC3 DEC TWIN       ;do the back
5530      JSR JIFF      ;ward scolling
5540      LDA TWIN
5550      CMP #$FF
5560      BNE SC2
5570      DEC TWIN+1
5580 SC2 DEC SCNT       ;countdown
5590      BNE SC3       ;not done
5600      JSR RESROL    ;reset line
5610      RTS           ;exit
5620 ;
5630 ;Number of bytes to scroll
5640 ;
5650 SCTAB .BYTE 40,80,120
5660 ;
5670 ;reset the ROLL DICE message
5680 ;
5690 RESROL LDA # <RLMES
5700      STA TWIN
5710      LDA # >RLMES
5720      STA TWIN+1
5730      RTS
5740 ;
5750 ;Check for Scolling Message
5760 ;
5770 CKMES LDA PNUM     ;plyr #
5780      TAX
5790      ASL A
5800      TAY
5810      LDA PTHP0,X   ;which prop
5820      LDX #1        ;for scroll
5830      CMP #17       ;taxes?
5840      BNE NTAX      ;no
5850      LDA CASH,Y    ;if taxes
5860      STA L         ;div cash by
5870      LDA CASH+1,Y  ;8 and
5880      STA L+1       ;subtract
5890      LSR L+1       ;from cash
5900      ROR L         ;if able to
5910      LSR L+1       ;otherwise
5920      ROR L         ;just exit
5930      LSR L+1
5940      ROR L         ;this is a
5950      LDA CASH,Y    ;tax break
5960      SEC           ;for the poor
5970      SBC L
5980      STA CASH,Y
5990      LDA CASH+1,Y
6000      SBC L+1
6010      STA CASH+1,Y
6020      JSR FORMDOL   ;show new $
6030      LDX #1        ;now scroll
6040      JMP MJM
6050 NTAX INX           ;vacation=2
6060      CMP #11       ;HOGG #1
6070      BEQ MJM
6080      CMP #26       ;HOGG #2
6090      BEQ MJM
6100      RTS           ;done
6110 MJM JMP MESSAG     ;go scroll
6120 ;
6130 ;Dice Definition Tables
6140 ;
6150 RTAB .WORD ONE
6160      .WORD TWO
6170      .WORD TRE
6180      .WORD FOR
6190      .WORD FIV
6200      .WORD SIX
6210 ;
6220 ONE .SBYTE "QQQQRQQQQ"
6230 TWO .SBYTE "RQQQQQQQR"
6240 TRE .SBYTE "RQQQRQQQR"
6250 FOR .SBYTE "RQRQQQRQR"
6260 FIV .SBYTE "RQRQRQRQR"
6270 SIX .SBYTE "RQRRQRRQR"
6280 ;
6290 ;These are the Token Definitions
6300 ;
6310 P0DEF .BYTE $60,$E0,$60,$60
6320      .BYTE $60,$F0,$00,$3C,$3C
6330      .BYTE $3C,$3C,$3C,$3C
6340 ;
6350 P1DEF .BYTE $38,$6C,$0C,$18
6360      .BYTE $30,$7C,$00,$3C,$3C
6370      .BYTE $3C,$3C,$3C,$3C
6380 ;
6390 P2DEF .BYTE $7E,$0C,$18,$0C
6400      .BYTE $66,$3C,$00,$3C,$3C
6410      .BYTE $3C,$3C,$3C,$3C
6420 ;
6430 P3DEF .BYTE $1B,$1B,$1F,$1F
6440      .BYTE $03,$03,$00,$3C,$3C
6450      .BYTE $3C,$3C,$3C,$3C
6460 ;
6470 ;These are the Horizontal
6480 ;positions of the Players
6490 ;
6500 PTHX .BYTE $98,$98,$90,$88
6510      .BYTE $80,$78,$70,$68
6520      .BYTE $60,$58,$50,$50
6530      .BYTE $48,$40,$40,$40
6540      .BYTE $40,$40,$48,$50
6550      .BYTE $58,$60,$68,$70
6560      .BYTE $78,$80,$88,$88
6570      .BYTE $88,$90,$98,$A0
6580      .BYTE $A8,$A8,$A8,$A0
6590 ;
6600 ;These are the Vertical
6610 ;positions of the Players
6620 ;
6630 PTHY .BYTE $78,$88,$88,$88
6640      .BYTE $88,$88,$88,$88
6650      .BYTE $88,$88,$88,$78
6660      .BYTE $78,$78,$68,$58
6670      .BYTE $48,$38,$38,$38
6680      .BYTE $38,$38,$38,$38
6690      .BYTE $38,$38,$38,$48
6700      .BYTE $58,$58,$58,$58
6710      .BYTE $58,$68,$78,$78
6720 ;
6730 ;The MAP colors and the
6740 ;TOKEN colors
6750 ;
6760 MAPCL .BYTE $4E,$A0,$F4,$E0,$82
6770 PCLR .BYTE $02,$42,$7A,$C6
6780 ;
6790 ;These are the low and hi bytes
6800 ;of the screen position of each
6810 ;property square on the MAP
6820 ;
6830 SCL .BYTE $FA,$4A,$48,$46,$44
6840      .BYTE $42,$40,$3E,$3C,$3A
6850      .BYTE $38,$E8,$E6,$E4,$94
6860      .BYTE $44,$F4,$A4,$A6,$A8
6870      .BYTE $AA,$AC,$AE,$B0,$B2
6880      .BYTE $B4,$B6,$06,$56,$58
6890      .BYTE $5A,$5C,$5E,$AE,$FE
6900      .BYTE $FC
6910 ;
6920 SCH .BYTE $71,$72,$72,$72,$72
6930      .BYTE $72,$72,$72,$72,$72
6940      .BYTE $72,$71,$71,$71,$71
6950      .BYTE $71,$70,$70,$70,$70
6960      .BYTE $70,$70,$70,$70,$70
6970      .BYTE $70,$70,$71,$71,$71
6980      .BYTE $71,$71,$71,$71,$71
6990      .BYTE $71
7000 ;
7010 ;If a property is bought
7020 ;show it with a thicker
7030 ;box or circle. Each symbol
7040 ;is 2 ANTIC 4 characters wide,
7050 ;
7060 SHWPUR LDX #35     ;go thru all
7070 SHH LDA SCL,X      ;36-first
7080      STA L         ;get its
7090      LDA SCH,X     ;screen location
7100      STA L+1
7110      LDA REFTAB,X  ;find out its
7120      BMI FJ        ;status: special
7130      BNE FI
7140      LDY #0        ;0=single free
7150      LDA #$39      ;left side box
7160      STA (L),Y
7170      INY
7180      LDA #$3A      ;right side
7190      STA (L),Y     ;box
7200 FJ   DEX           ;next property
7210      BPL SHH
7220      RTS           ;done
7230 FI   CMP #1        ;1=singl,bought
7240      BNE FK
7250      LDA #$2D      ;thick box,left
7260      LDY #0
7270      STA (L),Y
7280      INY
7290      LDA #$2E      ;thick,right
7300      STA (L),Y
7310      BNE FJ
7320 FK   CMP #2        ;double,free?
7330      BNE FL        ;no
7340      LDY #0
7350      LDA #$2B      ;circle
7360      STA (L),Y     ;left side
7370      INY
7380      LDA #$2C      ;right side
7390      STA (L),Y
7400      BNE FJ
7410 FL   CMP #3        ;double,bought
7420      BNE FJ
7430      LDY #0
7440      LDA #$29      ;thick circle
7450      STA (L),Y     ;left side
7460      INY
7470      LDA #$2A      ;right side
7480      STA (L),Y
7490      BNE FJ
7500 ;
7510 ;Show the Property Name as
7520 ;you go by it.
7530 ;
7540 SHWPRP JSR CLRPRP ;clear line
7550      LDA # <PRPLIN ;point to
7560      STA TWIN      ;line in
7570      LDA # >PRPLIN ;display
7580      STA TWIN+1    ;list
7590      LDX PNUM      ;plyr #
7600      LDA PTHP0,X   ;get prop #
7610      ASL A
7620      TAX           ;x2 for table
7630      LDA PRPTAB,X  ;get name
7640      STA L
7650      LDA PRPTAB+1,X
```

```
7660      STA L+1
7670      LDY #0
7680      LDX #10      ;posit. on
7690 SP1 LDA (L),Y     ;line,get byte
7700      BEQ SP2      ;0=done
7710      JSR ASC2IC   ;to INT code
7720      STA PRPLIN,X ;show it
7730      INY
7740      INX
7750      BNE SP1
7760 SP2 RTS
7770 ;
7780 ;Clear Property Show Line
7790 ;
7800 CLRPRP LDX #39
7810      LDA #0
7820 CPL STA PRPLIN,X
7830      DEX
7840      BPL CPL
7850      RTS
7860 ;
7870 ;The Property Names
7880 ;
7890 P01 .BYTE "Bar-B-Q Ranch",0
7900 P02 .BYTE "Ringading Phone",0
7910 P03 .BYTE "Crosseye Cable TV",0
7920 P04 .BYTE "Olde Yorke Times",0
7930 P05 .BYTE "Spuds Potato Farm",0
7940 P06 .BYTE "Black Gold Oil",0
7950 P07 .BYTE "Rex Auto Corp.",0
7960 P08 .BYTE "Thredbare Clothing",0
7970 P09 .BYTE "Kolbe's Cheese",0
7980 P10 .BYTE "Daisy's Dairy",0
7990 P11 .BYTE "Black Jack's Casino"
8000      .BYTE 0
8010 P12 .BYTE "Death Valley Spa",0
8020 P13 .BYTE "Adams Apple Orchard"
8030      .BYTE 0
8040 P14 .BYTE "Sour Grapes Vineyard"
8050      .BYTE 0
8060 P15 .BYTE "Short Circuit P & L"
8070      .BYTE 0
8080 P16 .BYTE "Heavy Water Company"
8090      .BYTE 0
8100 P17 .BYTE "Stytch's Clinic",0
8110 P18 .BYTE "Schappel's Scalpels"
8120      .BYTE 0
8130 P19 .BYTE "Crosstrax Railway",0
8140 P20 .BYTE "Drip Dry Cleaners",0
8150 P21 .BYTE "Jetstream Airlines",0
8160 P22 .BYTE "HAL Computers",0
8170 P23 .BYTE "Selmart Stores",0
8180 P24 .BYTE "IOU Penneys",0
8190 P25 .BYTE "Wilt's Vegetables",0
8200 P26 .BYTE "Titanic Shipping",0
8210 P27 .BYTE "Steal Steelworks",0
8220 P28 .BYTE "Silicon Gulch",0
8230 P29 .BYTE "Disastrous Insurance"
8240      .BYTE 0
8250 P30 .BYTE "The Luck Square",0
8260 P31 .BYTE 0
8270 P32 .BYTE "Capital Gains Tax",0
8280 ;
8290 ;The Look-Up Table for each
8300 ;square on the MAP board.
8310 ;* marks the doubles
8320 ;
8330 PRPTAB .WORD P31 ;stock market
8340      .WORD P25    ;wilt veg     *
8350      .WORD P26    ;titanic ship*
8360      .WORD P07    ;rex auto
8370      .WORD P18    ;schappel *
8380      .WORD P17    ;stytches *
8390      .WORD P30    ;L (luck)
8400      .WORD P19    ;crosstrax rail
8410      .WORD P06    ;black gold oil
8420      .WORD P22    ;HAL computer *
8430      .WORD P28    ;silicon gulch*
8440      .WORD P31    ;H (holiday)
8450      .WORD P11    ;casino
8460      .WORD P12    ;death valley
8470      .WORD P23    ;selmart
8480      .WORD P13    ;adams apple *
8490      .WORD P14    ;sour grapes *
8500      .WORD P31    ;T (taxman)
8510      .WORD P20    ;drip dry
8520      .WORD P05    ;spud potatoes *
8530      .WORD P01    ;bbq ranch     *
8540      .WORD P21    ;jetstream air
8550      .WORD P30    ;L (luck)
8560      .WORD P02    ;phone co
8570      .WORD P24    ;iou pennys
8580      .WORD P08    ;threadbare
8590      .WORD P31    ;H (holiday)
8600      .WORD P09    ;kolbe *
8610      .WORD P10    ;daisy *
8620      .WORD P27    ;steal steel
8630      .WORD P03    ;crosseye TV *
8640      .WORD P04    ;olde yorke  *
8650      .WORD P32    ;C (capital tax)
8660      .WORD P29    ;insurance
8670      .WORD P16    ;heavy H2O *
8680      .WORD P15    ;P&L       *
```

LISTING 6: ASSEMBLY

```
0100 ;SAVE#D:CAPITAL5.M65
0110 ;
0120 ;--------------------;
0130 ;                    ;
0140 ;CAPITAL! Subroutines;
0150 ;                    ;
0160 ;by: Bryan Schappel  ;
0170 ;                    ;
0180 ;--------------------;
0190 ;
0200 ;Get the Rent of a property
0210 ;
0220 GET_RENT LDA CPROP ;curr prop
0230      TAX          ;offset in table
0240      LDA RENT,X   ;get rent
0250      STA AMOUNT   ;save
0260      LDA #0       ;0 hi byte
0270      STA AMOUNT+1
0280      LDA XREF,X   ;is this a dbl?
0290      BNE GOTD     ;YES!
0300      RTS          ;no.
0310 ;
0320 GOTD TAY          ;half #
0330      LDA WHO,Y    ;get owner
0340      CMP OWNER    ;same as landed
0350      BNE ROUT     ;on? no!
0360      LDA RENT,Y   ;YES! Make rent
0370      CLC          ;=half+half
0380      ADC AMOUNT
0390      STA AMOUNT
0400      LDA AMOUNT+1
0410      ADC #0
0420      STA AMOUNT+1
0430 ROUT RTS          ;exit
0440 ;
0450 ;Handle an Improvement
0460 ;
0470 IMPROVE JSR HAVE_ANY ;own any?
0480      BPL IM1      ;YES
0490      RTS          ;none if -
0500 IM1  PRINT  IMP_TXT ;'Improve?'
0510      JSR GETYN    ;Yes/No
0520      BEQ GIM      ;0 = yes
0530      RTS          ;out
0540 GIM LDA PNUM      ;plyr #
0550      ASL A        ;x2
0560      TAY
0570      LDA CASH+1,Y ;see if plyr
0580      BNE GIMP     ;has $10G
0590      LDA CASH,Y   ;plus $1G
0600      CMP #11
0610      BCS GIMP     ;sure
0620      JMP NO_CASH ;not enuf
0630 GIMP JSR PICK_PROP ;pick a prop
0640      LDX PPROP    ;picked prop #
0650      BMI GOUT     ;- = none pick
0660      BNE IMP_IT
0670 GOUT RTS          ;0 = no
0680 ;
0690 IMP_IT LDA IMP,X ;get cur level
0700      CMP #3       ;at max. level?
0710      BCC DO_IMP   ;no
0720       PRINT  MAX ;'At max'
0730      JMP WAITKEY ;exit
0740 DO_IMP LDA RENT,X ;get rent
0750      ASL A        ;x2
0760      STA RENT,X   ;save it
0770      INC IMP,X    ;inc level
0780      LDA COST,X   ;add 5 to
0790      CLC          ;prop cost
0800      ADC #5
0810      STA COST,X
0820      LDA #10      ;costs $10g
0830      LDY PNUM     ;to inprove
0840      JSR SUB
0850      JMP T_OK     ;'Trans. comp'
0860 ;
0870 ;Get a Yes or No
0880 ;0= yes <> = no
0890 ;
0900 GETYN  PRINT  YNTXT ;'Yes No'
0910 GYN JSR GETCH
0920      CMP #43      ;a Y
0930      BNE TRYN
0940      RTS
0950 TRYN CMP #$23     ;an N
0960      BNE GYN
0970      CMP #0       ;force BNE
0980      RTS
0990 ;
1000 ;Make a Player PAY
1010 ;
1020 PAY_RENT JSR GET_RENT ;get rent
1030       PRINT  FEE ;print it
1040      LDA #0       ;pay to
1050      STA RFLAG    ;player
1060      BEQ JJSS
1070 JUST_PAY LDA #1 ;just lose $
1080      STA RFLAG
1090 JJSS LDA PNUM     ;plyr #
1100      ASL A        ;x2
1110      TAY
1120      LDA CASH+1,Y ;get CASH
1130      CMP AMOUNT+1 ;is it enuf?
1140      BEQ PALO     ;check low
1150      BCS PAY_IT   ;yes
1160 PALO LDA CASH,Y ;enuf low?
1170      CMP AMOUNT
1180      BCC FORCE    ;no,force sell
1190      BEQ FORCE    ;ditto
1200 PAY_IT LDY PNUM ;pay
1210      LDA AMOUNT   ;this amount
1220      JSR SUB
1230      LDA #0       ;turn disaster
1240      STA DISFLAG ;flag off
1250      LDA RFLAG    ;just lose if
1260      BNE TOK      ;<> 0
1270      LDY OWNER    ;pay owner
1280      LDA AMOUNT   ;the amount
1290      JSR ADD      ;add to owner
1300 TOK JMP T_OK      ;exit
1310 FORCE LDA #1      ;force sale
1320      STA DISFLAG ;of props
1330      JSR MAKE_RAN ;selling prices
1340      LDA SPEED    ;fast game=0
```

```
1350      BEQ DED
1360      JSR NO_CASH ;'No cash'
1370      JSR WANT_SALE ;sell props
1380      LDA PPROP    ;get prop #
1390      BPL JJSS     ;ok,try pay
1400 DED PLA           ;no props so
1410      PLA          ;player dies
1420      JMP DEAD     ;tell everyone
1430 ;
1440 ;Pick a Property
1450 ;
1460 PICK_PROP LDA CPROP ;save curr.
1470      STA TPROP    ;property #
1480      JSR HAVE_ANY ;have any?
1490      BPL G_PICK   ;yes
1500      STY PPROP    ;no Y=$FF
1510      RTS          ;exit
1520 G_PICK LDY #0     ;cycle thru
1530 PCK1 LDA WHO,Y    ;props. who
1540      STY CPROP    ;owns this one
1550      CMP PNUM     ;ME?
1560      BNE PCKU     ;no
1570      JSR SHOPROP ;show prop info
1580      INC PAUSE?   ;1 = fast prnt
1590      LDA DISFLAG ;disaster?
1600      BEQ PCKQ     ;no
1610       PRINT  AMTOWD ;'Amt owed'
1620      LDA AMOUNT   ;print amnt
1630      LDX AMOUNT+1
1640      JSR PRNUM
1650 PCKQ  PRINT  PICK_MESS ;'pick?'
1660      LDA DISFLAG ;if disaster
1670      BNE PCKX     ;must pick
1680       PRINT  PICK_REST ;'ESC'
1690 PCKX DEC PAUSE? ;slow down
1700 PCKK JSR GETCH    ;get key
1710      CMP #28      ;ESC?
1720      BNE NOESC    ;no
1730      LDA DISFLAG ;yes,but if
1740      BNE PCKK     ;disaster -no
1750      LDA #0       ;if ESC set to
1760      STA CPROP    ;0 => none
1770      BEQ PICK_LV ;leave
1780 NOESC CMP #33     ;SPACE?
1790      BEQ PCKU     ;next one
1800      CMP #12      ;RETURN?
1810      BNE PCKK     ;picked one
1820 PICK_LV LDA CPROP ;get picked
1830      STA PPROP    ;property
1840      LDA TPROP    ;restore
1850      STA CPROP    ;cur. prop
1860      RTS
1870 ;
1880 PCKU LDY CPROP    ;get prop
1890      INY          ;next one
1900      CPY #36      ;at end?
1910      BCC PCK1     ;no-cycle
1920      LDY #0       ;reset to 0
1930      BEQ PCK1
1940 ;
1950 ;Handle Buying a Property
1960 ;
1970 BUY_IT  PRINT  BUY_MESS ;'Buy?'
1980      JSR GETYN    ;choice
1990      BEQ BUY1     ;yes
2000      RTS          ;don't want to
2010 ;
2020 BUY1 LDA PNUM     ;which plyr
2030      LDY CPROP    ;cur. prop
2040      ASL A        ;x2
2050      TAX
2060      LDA CASH+1,X ;enuf CASH?
2070      BNE BUY_OK   ;yes
2080      LDA CASH,X   ;check low bytes
2090      CMP COST,Y
2100      BCC NO_CASH ;not enuf
2110      BEQ NO_CASH ;  "
2120 BUY_OK LDA PNUM ;show who
2130      STA WHO,Y    ;now owns it
2140      TYA
2150      TAX          ;show it has
```

continued on page 64

```
2160      INC REFTAB,X ;been purchased
2170      LDA COST,Y   ;get price
2180      LDY PNUM     ;subtract
2190      JSR SUB      ;from CASH
2200 T_OK  PRINT  TRAN_OK ;'Complted'
2210      JSR WAITKEY ;wait 8 secs
2220      LDA #0       ;set pay to
2230      STA RFLAG    ;flag & hi
2240      STA AMOUNT+1 ;byte of pay
2250      RTS          ;amount
2260 ;
2270 ;Sorry no CASH
2280 ;
2290 NO_CASH LDA # <NCASH
2300      LDY # >NCASH
2310      JSR EPRINT
2320      JSR WAITKEY ;wait 8 secs
2330      LDA #0
2340      RTS
2350 ;
2360 ;Sell a Property
2370 ;
2380 WANT_SALE JSR HAVE_ANY ;any?
2390      BPL P_SALE   ;yes if +
2400      STY PPROP    ;$FF if -
2410      RTS
2420 P_SALE LDA DISFLAG ;disaster?
2430      BNE GO_SELL ;yes-sell
2440       PRINT  SELL_TXT ;choice
2450      JSR GETYN    ;if not forced
2460      BEQ GO_SELL ;sell
2470 SLV RTS
2480 ;
2490 GO_SELL JSR PICK_PROP ;pick one
2500      LDA PPROP    ;picked
2510      BMI SLV      ;none available
2520      BNE SELL_IT ;if $FF. 0
2530      LDA DISFLAG ;=none picked
2540      BNE GO_SELL ;forced sell
2550      RTS
2560 ;
2570 SELL_IT TAX       ;which prop
2580      DEC REFTAB,X ;sell it in
2590      LDA #$FF     ;table, $FF =
2600      STA WHO,X    ;no one owns
2610      LDA RANTAB,X ;another
2620 ;    sta cost,X   ;way to make
2630      LDY PNUM     ;the game
2640      JSR ADD      ;add $ to CASH
2650      JMP T_OK     ;done
2660 ;
2670 ;Own any Properties
2680 ;Y has propery # or 0
2690 ;+ if prop owned, - if not
2700 ;
2710 HAVE_ANY LDY #35 ;check list
2720      LDA PNUM     ;get our #
2730 HAV1 CMP WHO,Y    ;in table?
2740      BEQ HAV2     ;yes
2750      DEY
2760      BPL HAV1
2770 HAV2 RTS
2780 ;
2790 ;Handle LUCK locations
2800 ;
2810 DO_LUCK LDX PNUM ;plyr #
2820      LDA PTHP0,X ;cur prop #
2830      CMP #32      ;cap gains?
2840      BNE DO_RLUK ;no
2850      LDA # <CAPGNS ;CG routine
2860      STA LJMP+1
2870      LDA # >CAPGNS
2880      STA LJMP+2
2890      LDA #$7D     ;clear scn
2900      JSR EPUT
2910      LDA # <P32   ;CG prop name
2920      LDY # >P32
2930      JMP DO_CG
2940 DO_RLUK LDX RANDOM ;get random
2950      CPX #6       ;LUCK
2960      BCS DO_LUCK
```

```
2970      LDA LUCKL,X ;get LUCK
2980      STA LJMP+1   ;routine
2990      LDA LUCKH,X
3000      STA LJMP+2
3010      LDA #$7D     ;clear scn
3020      JSR EPUT
3030      LDA # <P30   ;'LUCK 5q'
3040      LDY # >P30
3050 DO_CG JSR INVPROP ;inverse name
3060       PRINT  YL   ;'You lose'
3070 LJMP JSR $FFFF    ;go to routine
3080      JMP T_OK     ;done
3090 ;
3100 ;** Luck Routines **
3110 ;
3120 ;Win 10G free (WOW for ME?)
3130 ;
3140 G1  LDA #10
3150      BNE G2.1
3160 ;
3170 ;Win 15G free
3180 ;
3190 G2  LDA #15       ;give $
3200 G2.1 LDY PNUM     ;to whom?
3210      STA VALUE
3220      JSR ADD      ;add to CASH
3230      JSR PG       ;Luck is 'good'
3240       PRINT  YW   ;'You win'
3250      LDA VALUE    ;'$nnG'
3260      LDX #0
3270      JMP PRNUM    ;print & exit
3280 PG   PRINT  GOOD ;print 'good'
3290      RTS
3300 ;
3310 ;Get a FREE Improvement
3320 ;
3330 G3  JSR GRPROP    ;get a prop
3340      BMI G1       ;none so
3350      TAX          ;give $
3360      LDA IMP,X    ;get level
3370      CMP #3       ;at max?
3380      BCS G1       ;yup give $
3390      INC IMP,X    ;increase it
3400      LDA RENT,X   ;get fee
3410      ASL A        ;x2
3420      STA RENT,X   ;save it
3430      STX TPROP    ;save prop #
3440      JSR PG       ;'good'
3450       PRINT  YW   ;'You win'
3460       PRINT  AN_IMP ;'improv..'
3470      LDA TPROP    ;get prop #
3480      ASL A        ;x2
3490      TAX          ;get name
3500      LDA PRPTAB+1,X
3510      TAY
3520      LDA PRPTAB,X
3530      JMP EPRINT   ;print & exit
3540 ;
3550 ;Loose 10G
3560 ;
3570 B1  JSR PB        ;'Bad luck'
3580       PRINT  YD   ;'You lose'
3590      LDA #10      ;'$10G'
3600 B1A STA AMOUNT    ;low amt
3610      LDX #0       ;high amt
3620      JSR PRNUM    ;show it
3630      INC RFLAG    ;pay to bank
3640      JSR JUST_PAY ;take it away
3650      PLA          ;pull return
3660      PLA
3670      RTS          ;exit
3680 ;
3690 PB  LDA # <BAD    ;print 'bad
3700      LDY # >BAD   ;luck'
3710      JMP EPRINT
3720 ;
3730 ;Cut the Rent(Fee) in half
3740 ;
3750 B2  JSR GRPROP    ;grab prop #
3760      BMI B1       ;none
3770      TAX
```

NEW ARRIVALS

NEW CARTRIDGES!

STAR RAIDERS
E.T.
DIG DUG

ASTEROIDS
DEFENDER
MISSLE COMMAND

GALAXIAN
FACEMAKER

1020 COLOR PRINTER PLOTTER

Complete with:
- 2 Pen Sets
- 1 Roll Paper
- Power Supply & Cable

$14.98 Brand New

EXTRA PEN SETS ..... $ .89 (Black) $3.98 (Color)

1200XL 64K COMPUTER

w/PAC-MAN **$49.** Reconditioned

1027 PRINTER

- 80 Column
- Direct Connect
- Letter Quality

$79. Brand New

1050 DISK DRIVE

$169. Reconditioned

ATTN. DEALERS

PAC-MAN Case of 120 **$69.**

800 48K COMPUTER

w/PAC-MAN **$69.95** Reconditioned

1025 PRINTER

- 80 Column
- Dot Matrix
- Friction/Tractor
- Direct Connect

$79.98 Reconditioned

800 BOARDS

- Mother
- Power
- CPU
- ROM

YOUR CHOICE **$8.98** Brand New

8Bit I/O Cable . $4.98

850 Interface Serial, parallel **$79.98** Reconditioned

810 DRIVES **$129.** Reconditioned

BASIC TUTOR 4 BOOKS **$4.98**

5.25" DISKS 20 CENTS EA.*

QTY.	PRICE
10	$1.00
100	$29.95
*1000	$200

MAJORITY ARE UNNOTCHED CONTAINING OLD SOFTWARE

NEW DOS XE .. $9.95

DOS 2.5 w/manual .. $4.98

1010 Tape drive ..... $29.
10' Joystick ext. ..... $ .99
Trackball $9.95

CARTRIDGES FOR 800, XL, XE

BASIC CARTRIDGE	$4.95	MATH ENCOUNTER	$7.98	DEFENDER	$14.95	BALLBLAZER	$19.98
BASIC TUTOR (4 BOOKS)	$4.95	DANCE FANTASY	$8.98	ROBOTRON	$19.98	BLUE MAX	$19.98
TURMOIL	$4.95	LOGIC LEVELS	$8.98	TENNIS	$19.98	STAR RAIDERS II	$19.98
PAC-MAN (no box)	$4.95	MEMORY MANOR	$8.98	FINAL LEGACY	$19.98	DAVID'S MIDNIGHT MAGIC	$19.98
DONKEY KONG (no box)	$4.95	LINKING LOGIC	$8.98	MARIO BROS.	$19.98	ARCHON	$19.98
GORF (400,800)	$4.95	CHICKEN	$8.98	DONKEY KONG JR.	$19.98	KARATEKA	$19.98
DEMON ATTACK (400,800)	$4.95	CLAIM JUMPER	$8.98	JUNGLE HUNT	$19.98	CHOPLIFTER	$19.98
DELUXE INVADERS	$4.95	DELTA DRAWING	$8.98	MOON PATROL	$19.98	GATO	$24.98
JOURNEY TO THE PLANETS	$4.95	HEY DIDDLE DIDDLE	$8.98	BATTLEZONE	$19.98	ACE OF ACES	$24.98
STAR RAIDERS	$4.95	SLIME (400/800)	$8.98	FOOD FIGHT	$19.98	LODE RUNNER	$24.98
MISSLE COMMAND	$4.95	ALPHABET ZOO	$8.98	HARDBALL	$19.98	BARNYARD BLASTER (gun req.)	$24.98
ASTEROIDS	$4.95	ALF	$8.98	FIGHT NIGHT	$19.98	DARK CHAMBERS	$29.98
GALAXIAN	$4.95	ADVENTURE CREATOR	$8.98	ONE ON ONE BASKETBALL	$19.98	AIRBALL	$29.98
DEFENDER	$4.95	DIG DUG	$14.95	DESERT FALCON	$19.98	SUMMER GAMES	$29.98
E.T.	$4.95	SKY WRITER	$14.95	NECROMANCER	$19.98	CROSSBOW (gun req.)	$29.98
FACEMAKER	$4.95	FOOTBALL	$14.95	RESCUE ON FRACTALUS	$19.98	CRIME BUSTERS (gun req.)	$29.98

DISK SOFTWARE FOR 800, XL, XE

DAVID'S MIDNIGHT MAGIC	$4.98	CRYSTAL RAIDER	$4.98	PREPPIE II	$4.98	PIRATE ADVENTURE	$4.98
ZORRO	$4.98	DESPATCH RIDER	$4.98	THE COUNT	$4.98	SECRET MISSION	$4.98
BANDITS (48K 400,800)	$4.98	MISSION ASTEROID	$4.98	DISK 50 (50 GAMES)	$4.98	VOODOO CASTLE	$4.98
PROTECTOR II	$4.98	SPIDERMAN	$4.98	FREAKY FACTORY	$4.98	TECHNICOLOR DREAM	$4.98
CLAIM JUMPER	$4.98	HULK	$4.98	LASER HAWK	$4.98	STRANGE ODYSSEY	$4.98
SYNTREND	$4.98	COMMBAT	$4.98	ROCKET REPAIRMAN	$4.98	VISICALC	$24.98
CROSSCHECK	$4.98	PREPPIE I	$4.98	ADVENTURELAND	$4.98	BOOKKEEPER W/ num. keypad	$29.98
MOLECULE MAN	$4.98					GET RICH	$29.98

Astra

Double Density
Dual Disk Drives
(2 drives in 1 case)

$199. *New!*

Special buys on 8-BIT Hardware!

SAN JOSE COMPUTER

THE ATARI STORE

Sunrise Plaza 640 Blossom Hill Rd. San Jose, CA 95123
(408) 224-8575 • BBS (408) 224-9052

Light Gun

For use with pistol games on 8-Bit systems

$34.98

Light Gun Package

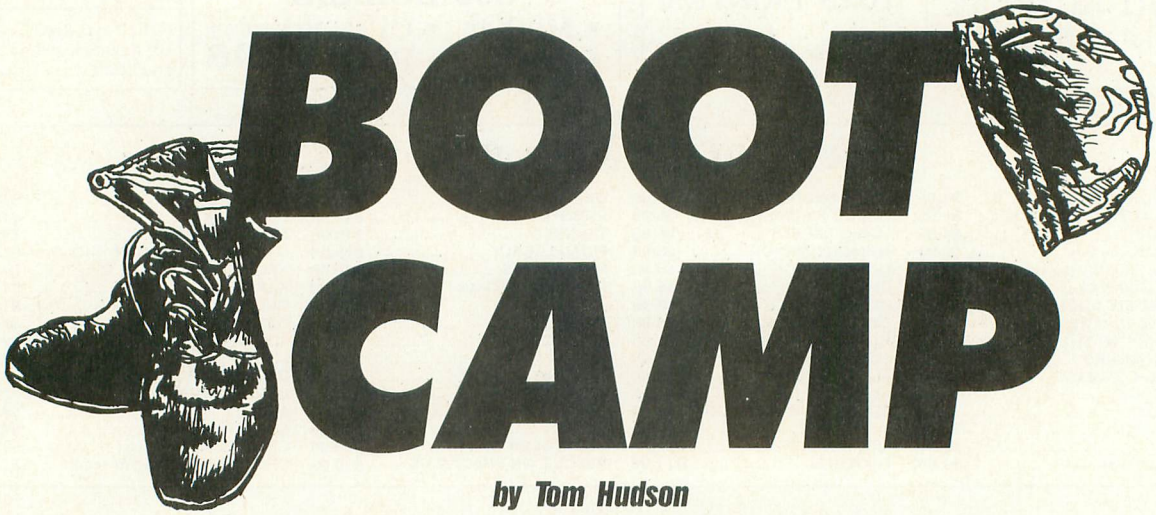
Includes:
- Light Gun
- Crime Busters
- Crossbow
- Barnyard Blaster

$109.

SHIPPING: ADD $5.00 TO ALL ORDERS. AIR AND INTERNATIONAL SHIPPING EXTRA. THAT'S IT.
WARRANTY: 90 DAY WARRANTY ON ALL ITEMS. **TAX:** CALIFORNIA RESIDENTS ADD 7% SALES TAX.
PREPAYMENT: USE VISA, MASTERCARD, MONEY ORDER, CASHIER'S CHECK OR PERSONAL CHECK. PERSONAL CHECK MUST CLEAR PRIOR TO SHIPMENT. **C.O.D.:** CASH, CASHIER'S CHECK OR M.O. ONLY.
Prices subject to change without notice. Brand and/or product names are trademarks or registered trademarks of their respective holders.
Ad produced on an ATARI ST using Publishing Partner and printed on an ATARI SLM804 PostScript compatible laser printer.

CIRCLE #102 ON READER SERVICE CARD.

BOOT CAMP



by Tom Hudson

I hope all **Boot Camp** readers have been practicing their addition, subtraction and X-Y register manipulations, because we're moving on to bigger and better things. We'll be dabbling with comparisons, branching and indexing this month, giving you even more tools to work with in assembly language.

First Things First

Last month, I gave you a simple data-manipulation problem:

PROBLEM: Write a program which starts with A=$03, X=$07 and Y=$14. Then write the code necessary to change these registers so that when the program ends, the registers are A=$07, X=$14 and Y=$03.

As most readers know, there are hundreds of ways to solve any programming problem, and this one is no exception. The objective is not just to solve the problem, but to do it in the most efficient way possible. I'll show you two ways to solve the above problem, and discuss the pros and cons of each.

```
10        STA AHOLD
20        STX XHOLD
30        STY YHOLD
40        LDA XHOLD
50        LDX YHOLD
60        LDY AHOLD
70        BRK
80 AHOLD  *=*+1
90 XHOLD  *=*+1
0100 YHOLD *=*+1
0110        .END
```

The above shows an easy-to-understand, straightforward solution to our problem. It stores each register in hold areas, then loads the registers from the appropriate hold area. Lines 10-60 perform the register exchange function, and Lines 80-100 set up the one-byte storage areas.

This solution is easy to understand by simply looking at it, and is a solution that most beginners would probably use. However, from a memory usage standpoint, this routine requires 22 bytes. We can do the same exchange in only ten bytes with this routine:

```
10        STY HOLD
20        TAY
30        TXA
40        LDX HOLD
50        BRK
60 HOLD   *=*+1
70        .END
```

As you can see, this code uses two of the transfer instructions, TAY and TXA, to eliminate two of the temporary storage areas used in the first version. Since the transfer instructions use only one byte versus the six bytes for a LDA and STA instruction, this version of the exchange code uses less than half the memory.

Although we gain memory savings, we lose some readability. Let's say you use the first routine in a program and don't look at the program for a year. If you need to make a change, it's easy to see what the routine does. The second version may not be so easy to decipher. Since you never know when you'll have to make a change to a program, it's a good idea to comment your code heavily, in order to let yourself know what you were doing.

What If...?

The great thing about computers is that they can perform calculations quickly. Without the ability to make decisions though, a computer would be almost useless.

For this reason, the 6502 microprocessor in your Atari is equipped with 14 comparison instructions. These instructions are designed to test the values contained in the Accumulator, X and Y registers. Each of these instructions compares the desired register with the memory byte specified in the operand and sets the 6502 status flags accordingly.

The Accumulator comparison instructions are:

CMP	#n	(IMMEDIATE)
CMP	nn	(ABSOLUTE)
CMP	n	(ZERO PAGE)
CMP	(n,X)	(PRE-INDEXED INDIRECT)
CMP	(n),Y	(POST-INDEXED INDIRECT)
CMP	n,X	(ZERO PAGE INDEXED X)
CMP	nn,X	(INDEXED X)
CMP	nn,Y	(INDEXED Y)

The X register comparison instructions are:

CPX	#n	(IMMEDIATE)
CPX	nn	(ABSOLUTE)
CPX	n	(ZERO PAGE)

The Y register comparison instructions are:

CPY	#n	(IMMEDIATE)
CPY	nn	(ABSOLUTE)
CPY	n	(ZERO PAGE)

All comparison instructions affect only three status flags. These are the Sign, Zero and Carry flags.

What happens in a comparison? Internally, the computer will subtract the operand byte from the register contents, set the status flags just like a subtract, but will *not* alter the register. Simple, right? Let's look at a few examples.

Assume the accumulator contains $45,

and we execute the instruction:

CMP #$31

Inside the computer, the faithful 6502 would subtract $31 from $45 and obtain the following result:

```
$45  =  0 1 0 0 0 1 0 1
$31  =  0 0 1 1 0 0 0 1
        ---------------
        0 0 0 1 0 1 0 0= $14
```

Since the result is not 0, the Zero flag is set to 0. The Sign flag is set to Bit 7 or the result, which is 0. The Carry flag is set to 1, since no borrow was required. The Carry flag is always the inverse of the borrow status.

By looking at the result of this comparison, we can say that the accumulator is not equal to $31, since the result of the compare was not 0. We can also say that the accumulator is greater than $31, since the Carry flag is set.

Assume the X register contains $7F and we want to compare it with $7F. We would use the following instruction:

```
$7F  =  0 1 1 1 1 1 1 1
$7F  =  0 1 1 1 1 1 1 1
        ---------------
        0 0 0 0 0 0 0 0= $00
```

The result is 0, so the Zero flag is set to 1. The 7 bit of the result is 0, so the Sign flag is set to 1.

After this comparison is complete, we can conclude that the register is equal to $7F because the Zero flag is set.

Assume the Y register contains $12 and we want to compare it to $4E. We would use the following instruction:

CPY #4E

The subtract operation inside the 6502 would look like:

```
$12  =  0 0 0 1 0 0 1 0
$4E  =  0 1 0 0 1 1 1 0
        ---------------
        1 1 0 0 0 1 0 0= $C4
```

Before you get confused with the above binary operation, remember how subtraction works in Base 10. If the number being subtracted (minuend) is larger than the subtrahend, a borrow is necessary from the next higher digit. This case of the compare requires a borrow.

In this case, the Zero flag will be set to 0, indicating a non-zero result. The Sign flag will be set to the contents of Bit 7 of the result, which is a 1. The Carry flag will be set to 0, the inverse of the borrow status.

From these flags, we can conclude that the Y register is *less* than $4E because the Carry flag is cleared (0).

That's all there is to using the compare instructions. They work the same way, regardless of the address mode.

Comparisons are just about worthless without the ability to do something based on the result of a comparison, so next we'll look at the 6502 branch-on-condition instructions.

Branches Conveniently Located

So far, the only means of transferring program execution we've looked at has been the JMP (Jump to location) instruction. Now we'll look at the eight branch-on-condition instructions used by the 6502. The eight formats are:

BCS	n	(BRANCH IF CARRY= 1)
BCC	n	(BRANCH IF CARRY = 0)
BEQ	n	(BRANCH IF ZERO = 1)
BNE	n	(BRANCH IF ZERO = 0)
BMI	n	(BRANCH IF SIGN = 1)
BPL	n	(BRANCH IF SIGN = 0)
BVC	n	(BRANCH IF OFLOW = 0)
BVS	n	(BRANCH IF OFLOW = 1)

Observant readers may note that operand of the branch instructions consists of only one byte. As you may recall, the JMP instruction was able to jump to any memory location because its operand consisted of two bytes. Branches are another story altogether.

With only one byte in their operands, branch instructions are only able to branch backward 128 bytes or forward 127 bytes. This is known as "relative" addressing. Fortunately, most assemblers will calculate the distance of a branch for you. However, if a branch distance is more than the branch limit, you'll have to restructure your branch by using a JMP or multiple branch instructions.

Let's look at a few typical branch applications. Here's the comparison/branch structure for the condition:

IF X = 7 THEN GOTO START

```
        CPX #7
        BEQ START
          .
          .
          .
START
```

As you can see, the CPX instruction is followed by a branch instruction. In this case, if the X register is equal to 7, the program will go to the location labeled START.

For the condition:

IF A <> 52 THEN GOTO POINTA

we would use:

```
        CMP #52
        BNE POINTA
          .
          .
          .
POINTA
```

Multiple conditions may require some extra effort, such as the condition:

IF Y <=242 THEN GOTO MAIN

The code for this condition is:

```
        CPY #242
        BEQ MAIN
        BMI MAIN
          .
          .
          .
MAIN
```

These multiple conditions are really quite easy; you just have to use the instructions provided.

The nice thing about branch instructions is that you don't have to use them after a compare instruction. You can place them anywhere in a program. For example, in addition or subtraction instructions, which set the status flags just like a compare, a zero result in an operation will set the proper branch flags. Look at the following code:

```
LDA BYTE1
SEC
SBC BYTE2
CMP #0
BEQ ZERO
```

The CMP #0 instruction is not necessary, since the SBC operation set the flags for us! The optimized code would look like:

```
LDA BYTE1
SEC
SBC BYTE2
BEQ ZERO
```

Remember, branches can be done anywhere the status flags are altered, giving you incredible flexibility in program design.

"I Wish I Were Indexing..."

Now we can start combining some of our new programming tools to do meaningful work. With the added function of branching, we can start using the X and Y registers as counters and indexes.

Indexing was discussed in the second installment of *Boot Camp* in ANALOG, so I won't repeat all the basics. The first example I'll show is the use of the X and Y registers as counters.

Let's say we want to execute a section of code ten times. Since the program uses the Accumulator and X register in the loop,

we'll use the Y register as a counter to control the loop.

In order to use the X and Y registers as indexes, we have been given the four instructions:

INX	(INCREMENT X BY 1)
INY	(INCREMENT Y BY 1)
DEX	(DECREMENT X BY 1)
DEY	(DECREMENT Y BY 1)

These four instructions simply add or subtract 1 from the X or Y registers, allowing you to use the registers as indexes easily. These registers affect the Zero and Sign flags.

Here's the code necessary to perform a loop ten times:

```
        LDY #10
LOOP    .
        .
        .
        DEY
        BNE LOOP
```

This is a simple counter example. Note that, in this case, we have set up the Y register as a countdown counter, from 10 to 0. After the DEY instruction is executed, we BNE LOOP. If the Y register decremented to 0, the program will not take the branch, and the loop is finished. No CPY #0 instruction was needed, since the DEY instruction set the zero flag for us.

We could have used the Y register as a count-up counter, from 0 to 10, like this:

```
        LDY #0
LOOP    .
        .
        .
        INY
        CPY #10
        BNE LOOP
```

Note that in the count-up example an extra compare is needed (CPY #10) to see if the Y register has reached 10 yet. If it has not, the program will take the BNE LOOP branch to continue looping.

Using the X and Y registers for *indexing* is similar to using them for counters. The main difference is that the register is used inside the loop to point to varying places in memory. This code shows an example of indexing that will copy the six bytes of TABLE1 into TABLE2:

```
10      LDX #5
20 COPY LDA TABLE1,X
30      STA TABLE2,X
40      DEX
50      BPL COPY
60      BRK
70 TABLE1 .BYTE 10,20,30,40,50,60
80 TABLE2 *=*+6
90        .END
```

The program begins with the X register set at 5. Remember, when referencing individual elements in a table, the indexes for the elements range from 0 to 1 less than the number elements. In this case, the element numbers range from 0-5. As the loop (labeled COPY) executes, each byte of TABLE1 will be moved to TABLE2. This looping will continue until the X register is decremented past 0, where it will equal 255 due to wraparound. At this point, the Sign flag will be 1, indicating a negative number. When this happens, the BPL COPY instruction will be ignored and the looping will end. Try assembling this routine into memory and tracing its execution.

What if we want to copy TABLE1 into TABLE2 in reverse order? This is a nifty little problem that will help you understand X-Y indexing more thoroughly. Try writing the code, using as many memory locations as necessary. Next issue, I'll show a way to do this with only three changes to the above example.

No More Time

I had wanted to cover multi-byte math this issue, but due to space limitations, I'll have to delay this until next time. Until then, play around with comparisions and branching, and try to find a solution to the above problem.

FOR OUR DISK SUBSCRIBERS

The following programs from this issue are on disk:

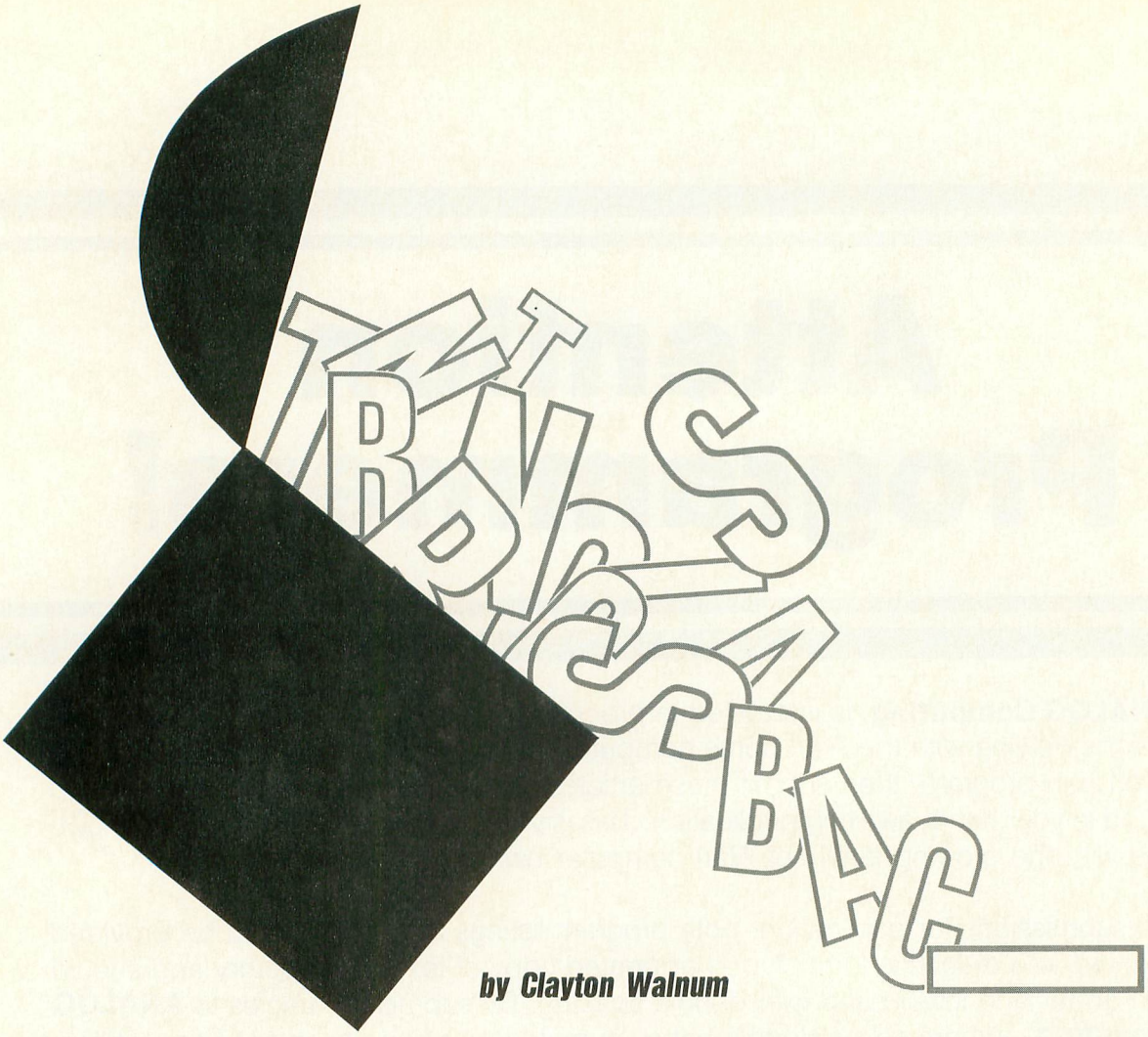
```
THE ANALOG #75 DISKETTE CONTAINS 18
MAGAZINE FILES.  THEY ARE LISTED BELOW:

SIDE 1:

FILENAME.EXT  LANG.   LOAD  ARTICLE NAME
------------  -----   ----  ------------

MOUNTAIN.BAS  BASIC   LOAD  NUCLEAR MOUNTAIN
GUNTEST .BAS  BASIC   LOAD  GUN ASSIST
GUNVBIM .COM  ML      (#4)  GUN ASSIST
CAPITAL .COM  ML      (#3)  CAPITAL!
PICTPERF.BAS  BASIC   LOAD  PICTURE PERFECT
MLEDITOR.BAS  BASIC   LOAD  M/L EDITOR
EDITORII.LST  BASIC   LIST  BASIC EDITOR II

SIDE 2:

FILENAME.EXT  LANG.   LOAD  ARTICLE NAME
------------  -----   ----  ------------

SYSEQU  .M65  MAC/65  LOAD  GUN ASSIST SOURCE
GRAPHICS.M65  MAC/65  LOAD  GUN ASSIST SOURCE
GUNVBIM .M65  MAC/65  LOAD  GUN ASSIST SOURCE
IOMAC   .LIB  MAC/65  LOAD  GUN ASSIST SOURCE
CAPTIAL1.M65  MAC/65  LOAD  CAPTIAL! SOURCE
CAPTIAL2.M65  MAC/65  LOAD  CAPTIAL! SOURCE
CAPTIAL3.M65  MAC/65  LOAD  CAPTIAL! SOURCE
CAPTIAL4.M65  MAC/65  LOAD  CAPTIAL! SOURCE
CAPTIAL5.M65  MAC/65  LOAD  CAPTIAL! SOURCE
CAPTIAL6.M65  MAC/65  LOAD  CAPTIAL! SOURCE
PICTPERF.M65  MAC/65  LOAD  PIC. PERFECT SOURCE

TO LOAD YOUR ANALOG DISK
------------------------

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

WARNING: BEFORE YOU RUN A PROGRAM, READ THE
         APPROPRIATE ARTICLE IN THE MAGAZINE.
         FAILURE TO DO SO MAY YIELD CONFUSING
         RESULTS.

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

 EXT  DESCRIPTION
 ---  -----------

.M65  REQUIRES THE MAC/65 ASSEMBLER
.AMA  REQUIRES THE ATARI MACRO ASSEMBLER
.ASM  REQUIRES THE ATARI ASSEMBLER/EDITOR
.ACT  REQUIRES THE ACTION! CARTRIDGE
.LGO  REQUIRES THE ATARI LOGO CARTRIDGE
.SYN  REQUIRES THE SYNAPSE SYN ASSEMBLER

LOADING NOTES
-------------

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

#1: SEE ACTION! MANUAL.
#2: SEE ATARI MACRO ASSEMBLER MANUAL.
#3: MAY ALSO BE LOADED FROM DOS USING THE "L"
    OPTION OF THE DOS MENU.
#4: THIS FILE SHOULD BE TRANSFERRED TO ANOTHER
    DISK AND RENAMED "AUTORUN.SYS".
#5: READ THE APPROPRIATE ARTICLE FOR INSTRUCTIONS
    ON USING THIS FILE.
```

Attention Programmers!

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

All submissions for publication, both program listings and text, should be provided in printed and magnetic form. Typed or printed copy of text is mandatory and should be in upper and lower case with double spacing. By submitting articles to **ANALOG Computing**, authors acknowledge that such materials, upon acceptance for publication, become the exclusive property of **ANALOG Computing**. If not accepted for publication, the articles and/or programs will remain the property of the author. If submissions are to be returned, please supply a self-addressed, stamped envelope. All submissions of any kind must be accompanied by the author's full address and telephone number.

Send your programs and articles to:
ANALOG Computing
P.O. Box 1413-M.O.
Manchester, CT 06040-1413



by Clayton Walnum

Previously, I referred to BASIC as an "interactive" language; that is, a language that easily allows for data to flow from the user to the computer and vice versa. BASIC's excellent string-handling functions are probably as much responsible for this ease of interaction as any other element of the language.

A string is simply a series of characters stored consecutively in memory. This data type allows us to store text in variables as easily as we can store numbers. This text can be a single character, a single word, a single sentence or even a complete document.

String variables, unlike numerical variables, must be "dimensioned" before we can use them. When we dimension them, we're telling the computer how much memory we need for the string. We don't have this problem with numerical variables because a numerical value, no matter how large it is, always fits in the same amount of memory.

To dimension a string variable, we use the BASIC keyword DIM:

```
10 DIM NAME$(20)
```

With this single line, we have told the computer that we are going to be using a string variable called STRING$, and that we will be storing no more than 20 characters in that string. We can store less than 20 if we wish, but BASIC won't allow us to store more than 20.

Another thing you should notice is the "$" in the variable name. String variable names always end with a dollar sign. That's how BASIC knows we're talking about a string and not a numerical array. (We'll get into arrays later.) Here are some other examples of string variable names:

ADDRESS$
TITLE$
PAGE$
WORD$

Once dimensioned, we can use a string variable in our program in a number of ways. There are many BASIC functions that allow us to manipulate strings in useful ways, the most obvious function being a string assignment:

```
20 NAME$="ANALOG Computing
"
```

In the line above, we've stored the two words, "ANALOG Computing," into the string variable NAME$. Notice that we've used only 16 characters, rather than the full 20. (The space character counts.) To verify that those two words are indeed stored in the variable, we can add this line to Lines 10 and 20:

```
30 PRINT NAME$
```

Type Lines 10 through 30 into your computer and run the program. This is what you'll see on the screen:

```
ANALOG Computing
```

The print statement in Line 30 has proven to us that NAME$ contains the two words we assigned to it in Line 20. Now add these lines to the program:

```
40 NAME$="ANALOG Computing
 magazine"
50 PRINT NAME$
```

Do you see a potential problem with Line 40? The string that we're assigning to NAME$ is 25 characters long, five characters longer than we dimensioned the string for. What do you suppose will happen? Run the program, and you'll see this on your screen:

BASIC TRAINING: STRINGS

```
ANALOG Computing
ANALOG Computing mag
```

BASIC didn't care that we tried to assign an oversized string to NAME$; on the other hand, it didn't let us get away with it either. The string was truncated to fit, so that NAME$ would contain only the first 20 characters.

Another way we can assign a string to a string variable is to use the INPUT command. We've used INPUT previously to get values from the keyboard for numerical variables. Using INPUT with strings is very similar:

```
10 DIM IN$(20)
20 PRINT "Type a string:"
30 INPUT IN$
40 PRINT "Your string is:
";IN$
```

Type the above program into your computer and run it. You should see something like this on your screen:

```
Type a string:
?TEST
Your string is: TEST
```

In the above program run, the user typed the word TEST after the ? prompt. Thanks to the INPUT statement in Line 30, the string was stored in IN$. In Line 40 the string was printed to the screen.

String variables can also be assigned values from other string variables:

```
10 DIM S1$(10),S2$(10)
20 PRINT "Type a string: "
30 INPUT S1$
40 S2$=S1$
50 PRINT "S1$ = ";S1$
60 PRINT "S2$ = ";S2$
```

When you run the above program, you'll get something like this on your screen:

```
Type a string:
?TEST
S1$ = TEST
S2$ = TEST
```

In Line 40 we assign the value of S1$ to S2$. Lines 50 and 60 print the text stored in both variables, proving that the assignment took place just as we expected. Also notice, in Line 10, that we can dimension more than one string variable with a single DIM statement by separating each string variable with a comma.

Advanced string manipulation

Some forms of BASIC allow programmers to combine strings with simple assignments such as this:

```
C$=A$+B$
```

Unfortunately, Atari BASIC won't allow us to do that, or at least won't allow us to do it that simply. Let's say that we've got two strings, FIRSTNAME$ and LASTNAME$, which we want to combine into a single string called NAME$. We can do it, but first we have to understand a little more about how strings are placed in memory.

When we assign a string to a string variable, each of the characters that make up the string are stored in contiguous bytes of memory. For example, let's say that FIRSTNAME$ contains BENNY. In the computer's memory, it looks something like this:

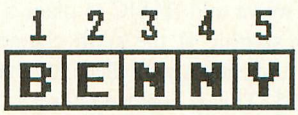


FIGURE 1

Now let's assign to each letter of the string a number based on its position in the string. In other words, B gets the number 1, E the number 2, and so on all the way to Y, which gets the number 5. Now we have a way to refer to each letter in the string. In BASIC we use those position numbers by adding "subscripts" to the string variable's name.

For example, if we wanted to refer to only the first three letters of FIRSTNAME$, we would use FIRSTNAME$(1,3). This tells BASIC that we want a portion of FIRSTNAME$ that starts with the first character and ends with the third. If we wanted the last three letters, we would use FIRSTNAME$(3,5). If we wanted the two N's, we would use FIRSTNAME$(3,4). Do you see how it works?

Type in and run the following program:

```
10 DIM FIRSTNAME$(10),LAST
NAME$(10),NAME$(20)
20 FIRSTNAME$="BENNY"
30 LASTNAME$="HILL"
40 NAME$=FIRSTNAME$
50 NAME$(6,6)=" "
60 NAME$(7,10)=LASTNAME$
70 PRINT "FIRSTNAME$ = ";F
IRSTNAME$
80 PRINT "LASTNAME$ = ";LA
STNAME$
90 PRINT "NAME$ = ";NAME$
```

A run of this program should give you:

```
FIRSTNAME$ = BENNY
LASTNAME$ = HILL
NAME$ = BENNY HILL
```

Let's see what's going on here. Line 10 dimensions the three strings we'll be using in the program. Lines 20 and 30 assign values to FIRSTNAME$ and LASTNAME$. In Line 40 we indirectly assign the string BENNY to NAME$.

Now comes the tricky part. First we need a space between the first and last names. In Line 50 we add this space to NAME$ giving us "BENNY". Translating the BASIC into English, we've told BASIC to place a space character starting at the sixth character of NAME$ and ending with the sixth character of NAME$. This is how we refer to a single character in a string.

In Line 60 we tell BASIC to place whatever is in LASTNAME$ (in this case, HILL) into the seventh through tenth characters of NAME$, giving us BENNY HILL. Then in Lines 70 through 90, we print out the strings stored in each of the variables, proving that we really did combine the two strings into one.

Of course, we're not always going to know exactly how large a string is, so we're not always going to be able to refer to the end of a string with a number. Suppose, for example, that we changed Lines 20 and 30 to:

```
20 PRINT "First name";:INP
UT FIRSTNAME$
30 PRINT "Last name";:INPU
T LASTNAME$
```

If I were to run this program and use my own name as input, the results would look like this:

```
First name?CLAYTON
Last name?WALNUM
FIRSTNAME$ = CLAYTON
LASTNAME$ = WALNUM
NAME$ = CLAYT WALN
```

Clearly, we need a way to refer to the length of a string without knowing in advance what that length is. And, happily, we have just such a function in Atari BASIC—the LEN function.

The LEN function returns the number of characters currently stored in a string variable. For instance, if NAME$ contained the string FELIX, the BASIC command LEN(NAME$) would return a value of 5. We can use the LEN function to modify our string-manipulation program so that it can accept any first and last names and still produce correct results (as long as the names don't exceed the dimensioned lengths of our strings). Change Lines 50 and 60 of the sample program to:

```
50 L=LEN(NAME$):NAME$(L+1)
=" "
60 NAME$(L+2)=LASTNAME$
```

Now when I run the program and input my name, I get:

```
First name?CLAYTON
Last name?WALNUM
FIRSTNAME$ = CLAYTON
LASTNAME$ = WALNUM
NAME$ = CLAYTON WALNUM
```

That's more like it!

Can you follow the program? After we assign the string in FIRSTNAME$ to NAME$, we get the length of the string and store it in the numerical variable L. The string CLAYTON is seven characters long, so we want to place the space in the eighth character, or in L+1.

Notice that, in Line 50, we're using only one subscript. This is perfectly legal, but we have to make sure we understand what we're doing. With the command NAME$(L+1)=" ", we're telling BASIC to place a space in position L+1 and to delete anything else from that point on. For example, if NAME$ was equal to "FREDDY," the command NAME$(4)=" " would leave us with "FRE." If we had used two subscripts, NAME$(4,4)=" ", we'd have gotten "FRE DY". Big difference.

Getting back to the sample program, in Line 60 we add the last name to NAME$ at the location L+2, which, when using my own name, adds the string WALNUM after the space we just placed in position 7.

One final note: we don't need to use the variable L in the above program. We can use the LEN function itself as our subscript, like this:

```
50 NAME$(LEN(NAME$)+1)=" "
60 NAME$(LEN(NAME$)+1)=LAS
TNAME$
```

This may look confusing at first. Just remember that LEN(NAME$) in Line 50 and LEN(NAME$) in Line 60 are not the same value. Why? Because in Line 50 we added a space to the string, making it one character longer.

Conclusion

Strings in Atari BASIC are powerful data types, and this discussion has only scratched the surface. In future installments, we'll see some very unusual ways to use strings, but for now make sure you understand the basics.

B&C ComputerVisions

3257 KIFER ROAD
SANTA CLARA, CA 95051
(408) 749-1003
(408) 749-9389 FAX

STORE HOURS
TUE - FRI 10am - 6pm
SAT - 10am - 5pm
CLOSED SUN - MON

800/XL/XE SOFTWARE
ALL TITLES ON DISK

ENTERTAINMENT

12 ADAMS ADVENTURES	14.95
ALIANTS	26.95
ALT. REALITY CITY	26.95
ALT. REALITY DUNGEON.	26.95
BEYOND CASTLE WOLF	14.95
BISMARK	26.95
BOP & WRESTLE	26.95
BORDINO:1812	26.95
BOULDERDASH CONSTR.SET	17.95
BRUCE LEE	17.95
CASTLE WOLFENSTEIN	14.95
DALLAS QUEST	7.95
D-BUG	7.95
F-15 STRIKE EAGLE	31.50
FIGHT NIGHT	17.95
GAUNTLET (64K)	31.50
DEEPER DUNGEONS	22.50
GUNSLINGER	26.95
HARD HAT MAC	7.95
JAWBREAKER	9.95
KARATEKA	13.50
KNICKERBOCKERS	13.50
KORONIS RIFT	13.50
LAST V-8	8.95
LEADERBOARD	13.50
MAIL ORDER MONSTERS	13.50
MICROLEAGUE BASEBALL	35.95
MONTEZUMA'S REVENGE	14.95
MOUSEQUEST	17.95
MOON SHUTTLE	7.95
NINJA	8.95
OIL'S WELL	9.95
O'RILEY'S MINE	9.95
PIRATES OF BARB. COAST	22.50
PREPPIE I & II	9.95
RESCUE ON FRACTALAS	13.50
SILENT SERVICE	31.50
SPEEDKING	8.95
SPIDERMAN	5.35
SPITFIRE 40	31.50
STARFLEET I	44.95
SPY VS. SPY III	17.95
STOCKMARKET	22.50
STRIP POKER	26.95
SUMMER GAMES	17.95
TAX DODGE	9.95
THE HULK	5.35
TOMAHAWK (64K)	26.95
TOP GUNNER	17.95
TOUCHDOWN FOOTBALL	13.50
TRAILBLAZER	26.95
UNIVERSE	44.95
ZAXXON (400/800)	13.50

PROGRAMMING

ACTION!	71.95
ACTION! TOOLKIT	26.95
BASIC XL	53.95
BASIC XL TOOLKIT	26.95
BASIC XE	71.95
DOS 2.5	7.95
DOS XE	10.00
DISK I/O	26.95
KYAN PASCAL	62.95
LIGHTSPEED C	35.95
LOGO	19.95
MAC/65	71.95
MAC/65 TOOLKIT	26.95
MACRO ASSEMBLER	22.50
PILOT	19.95
SPARTA DOS X	71.95

PRODUCTIVITY

ANIMATION STATION	89.95
ATARIWRITER	29.95
ATARIWRITER (CART ONLY)	19.95
ATARIWRITER+	39.95
ATARI BOOKKEEPER	24.95
ATARI MUSIC II	14.95
AWARDWARE (1050)	13.50
BANK STREET WRITER	14.95
BLAZING PADDLES	31.50
CELEBRITY COOKBOOK	26.95
COMPUTE YOUR ROOTS	35.95
DATAMANAGER	17.95
FAMILY FINANCE	6.95
GUITAR WIZARD	26.95
HOME ACCOUNTANT	19.95
HOME FILING MANAGER	6.95
HOMEPAK	24.95
INVENTORY MASTER	80.95
LETTER WIZARD	29.95
MUSIC CONSTRUCTION SET	13.50
NEWSROOM (1050 - 64K)	44.95
NEWS STATION	26.95
NEWS STA. COMPANION	26.95
PAGE DESIGNER	26.95
PRINT POWER (1050)	13.50
PRINTKIT (1050)	13.50
PRINTSHOP	34.95
P.S. COMPANION (64K)	24.95
P.S.GRAPHICS LIBRARY 1	17.95
P.S.GRAPHICS LIBRARY 2	17.95
P.S.GRAPHICS LIBRARY 3	17.95
PROOF READER	17.95
PUBLISHING PRO	35.95
RUBBER STAMP	26.95
SYNTREND (130XE)	35.95
SUPER MAILER	35.95
THE LOTTO PROGRAM	17.95
TIMEWISE	6.95
TURBOWORD/80 COLUMN REQUIRES XEP80	44.95
VIDEO TITLESHOP (64K)	26.95
GRAPHICS COMPANION	17.95
VIRTUOSO	29.95
VISICALC	24.95

EDUCATION

ATARI LIGHT MODULE (REQ. STARTER SET)	9.95
BUZZWORD	35.95
GRANDMA'S HOUSE (-10)	9.95
HEY DIDDLE (AGE 3-10)	9.95
MASTER TYPE	14.95
STATES AND CAPITALS	9.95
TOUCH TYPING	9.95
CBS (AGE 3-6):	
ASTROGROVER	8.95
BIG BIRD SPEC DELIVE	8.95
ERNIE'S MAGIC SHAPE.	8.95
DESIGNWARE:	
MATHMAZE (6-11)	35.95
MISSION ALGEBRA (13+)	35.95
SPELLICOPTER (6-11)	35.95
TINK TONK (AGE 4-6):	
ABC'S	8.95
COUNT AND ADD	8.95
SMART THINKER	8.95
SPELLING	8.95
SUBTRACTION	8.95
THINKING SKILLS	8.95
ALL 6 TINK TONKS	39.95
UNICORN:	
10 LITTLE ROBOTS (PRE-SCHOOL)	26.95
FUN BUNCH (6-ADULT)	26.95
RACECAR RITHMETIC (AGE 6+)	26.95
WEEKLY READER (PRE-SCHOOL):	
STICKY BEAR SHAPES	26.95
STICKY BEAR NUMBERS	26.95
STICKY BEAR ABC'S	26.96
STICKY BEAR OPPOSITE	26.95
SB BASKET BOUNCE	26.95
STICKY BEAR BOP	26.95
RUN FOR IT	26.95
PIC BUILDER	26.95

800/XL/XE SOFTWARE
ALL TITLES ON CARTRIDGE

ENTERTAINMENT

3D TIC-TAC-TOE	9.95
AIRBALL (XL/XE)	24.95
ALIEN AMBUSH	9.95
ACE OF ACES (XL/XE)	24.95
ARCHON	19.95
ASTEROIDS	15.95
ASTRO CHASE	14.95
ATARI TENNIS	9.95
ATLANTIS	14.95
BALL BLAZER	19.95
BARNYARD BLASTER (REQ. LIGHT GUN)	24.95
BATTLEZONE	19.95
B.C. QUEST FOR TIRES	19.95
BLUE MAX	19.95
BOULDERS & BOMBS	14.95
CAVERNS OF MARS	14.95
CENTIPEDE	14.95
CHICKEN	9.95
CHOPLIFTER	14.95
CLAIM JUMPER (400/800)	9.95
CLOUDBURST	9.95
CRIME BUSTER (REQ. LIGHT GUN)	24.95
CROSSBOW	24.95
CROSSFIRE	9.95
CRYSTAL CASTLES(XL/XE)	19.95
DARK CHAMBERS (XL/XE)	24.95
DAVIDS MIDNIGHT MAGIC	19.95
DEFENDER	14.95
DELUXE INVADERS	7.95
DESERT FALCON	19.95
DIG DUG	19.95
DONKEY KONG	5.00
DONKEY KONG JR.	19.95
EASTERN FRONT (1941)	19.95
E.T. PHONE HOME	9.95
FIGHT NIGHT	19.95
FINAL LEGACY	19.95
FOOD FIGHT (XL/XE)	19.95
FOOTBALL	14.95
FROGGER	14.95
GALAXIAN	9.95
GATO	24.95
GORF (400/800)	5.00
GYRUSS	14.95
HARDBALL	19.95
INTO THE EAGLES NEST	19.95
JOURNEY TO PLANETS	9.95
JOUST	19.95
JUNGLE HUNT	19.95
KABOOM!	14.95
KARATEKA	19.95
KRAZY ANTICS	14.95
LODE RUNNER	24.95
MARIO BROS.	19.95
MEGAMANIA	9.95
MILLIPEDE	14.95
MISSILE COMMAND	5.00
MOON PATROL	19.95
MR. COOL	9.95
MS. PAC MAN	19.95
NECROMANCER	19.95
ONE ON ONE (XL/XE)	19.95
PAC MAN	5.00
PENGO	19.95
POLE POSITION	19.95
POPEYE	14.95
Q-BERT	14.95
QIX	14.95
RESCUE ON FRACTALAS	19.95
RETURN OF THE JEDI	14.95
ROBOTRON:2084	19.95
SKY WRITER	14.95
SLIME (400/800)	9.95
SPACE INVADERS	14.95
STAR RAIDERS	5.00
STAR RAIDERS II	19.95
SUBMARINEE COMMANDER	14.95
SUMMER GAMES (XL/XE)	24.95
SUPER BREAKOUT	9.95
SUPER COBRA	14.95
THUNDERFOX	19.95
TRACK & FIELD	24.95
TURMOIL	9.95
WIZARD OF WOR	5.00

PRODUCTIVITY

ATARIWRITER	19.95
MICROFILER	22.50

EDUCATION

ATARILAB STARTER SET	29.95
FUN WITH ART	14.95
MATH ENCOUNTERS	9.95
FISHER PRICE (PRE SCHOOL):	
DANCE FANTASY	8.95
LINKING LOGIC	8.95
LOGIC LEVELS	8.95
MEMORY MANOR	8.95
SPINNAKER (AGE 3-10):	
ALF IN COLOR CAVES	9.95
ALPHABET ZOO	9.95
DELTA DRAWING	9.95
FACEMAKER	9.95
KIDS ON KEYS	9.95
KINDERCOMP	9.95
(AGE 7 - ADULT):	
ADV.CREATOR (400/800)	9.95
FRACTION FEVER	9.95

SUPER SPECIALS
RECONDITIONED ATARI MERCHANDISE
30 DAY WARRANTY

800 (48K) COMPUTER $79.95	**SPACE AGE JOYSTICK** $5.00.	**1030 MODEM WITH EXPRESS!** $24.95
400 (16K) COMPUTER $29.95	**ATARI TRACKBALL** $9.95	**1010 PROGRAM RECORDER** $29.95
1020 COLOR PRINTER/PLOTTER **$19.95** 40 COLUMNS WIDE (new in box)	**ATARI BOOKKEEPER** $14.95 - NO BOX **ATARI NUMERIC KEYPAD** $7.95	**DISKETTES AS LOW AS 20 CENTS** 10 FOR $4.00 100 FOR $29.95 1000 FOR $200 MOST ARE UNNOTCHED WITH OLD SOFTWARE

SHIPPING INFORMATION - Prices do not include shipping and handling. Add $5.00 for small items ($8.00 Min. for Canada). Add $8.00 for disk drive. Add $2.75 for C.O.D. Calif. res. include 7% sales tax. Mastercard and Visa accepted if your telephone is listed in your local phone directory. Orders may be pre-paid with money order, cashier check, or personal check. Personal checks are held for three weeks before order is processed. C.O.D orders are shipped via UPS and must be paid with cash, cashier check or money order. International and APO orders must be pre-paid with cashier check or money order. $20.00 minimum on all orders. All sales are final - no refunds - prices are subject to change.

Phone orders accepted **TUESDAY THROUGH FRIDAY** from 10:00 am to 6:00 pm PST.
We carry a full line of **ATARI** products - large public domain library - write or call for **free** catalogue

PRICES SUBJECT TO CHANGE WITHOUT NOTICE - ALL SALES ARE FINAL

CIRCLE #103 ON READER SERVICE CARD.

Picture Perfect

LISTING 1: BASIC

```
WO 1 REM ******************************
XE 2 REM *      PICTURE PERFECT      *
OG 3 REM *    BY JOE D. BRZUSZEK     *
ZD 4 REM *                           *
IN 5 REM *      COPYRIGHT 1989       *
PM 6 REM *    BY ANALOG COMPUTING    *
WU 7 REM ******************************
KQ 9 GOTO 8000
AF 20 T=USR(23043):T=PEEK(206):X1=PEEK(20
   3):Y1=PEEK(204):IF  NOT T THEN 20
WF 30 IF T>V1 THEN GOSUB VC:GOTO VA
ZU 40 RETURN
FI 50 T=USR(23043):T=PEEK(206):X1=PEEK(20
   3):Y1=PEEK(204):RETURN
EZ 99 REM KEYBOARD
HN 100 ON T=V33 AND AN GOTO 2200:IF T=V33
    THEN 1700
HD 110 IF T=13 THEN GOSUB 1600
IX 120 IF T=V8 THEN SP=SP+V1:IF SP>V4 THE
   N SP=V1
UX 130 IF T=11 THEN C1=C1+V1:IF C1=V4 THE
   N C1=Z
TN 140 POKE 22876,INT(SP*1.3):RETURN
QE 199 REM ANIMATE/PLAY
UK 200 GOSUB VN:X=Z:ON A$="" GOTO 2200:A=
   USR(CY,S1,Z,Z,159,95,S2,Z,V96,V3)
CX 210 FOR Y=V1 TO LEN(A$) STEP V4:A=USR(
   CY,S1,ASC(A$(Y)),ASC(A$(Y+V1)),ASC(A$(
   Y+V2)),ASC(A$(Y+V3)),S1,Z,Z,V3)
RW 220 FOR Y1=Z TO X:NEXT Y1:NEXT Y:IF PE
   EK(53279)=V3 THEN X=X+VA:POKE 53279,Z:
   IF X>VC THEN X=Z
XL 230 GOSUB VB:IF T THEN A=USR(CY,S2,Z,V
   96,159,190,S1,Z,Z,V3):GOTO 2200
MK 240 GOTO 210
ZX 259 REM STAMP
GU 260 GOSUB VN:IF  NOT S THEN GOSUB VB:S
   =V1:X4=X1:Y4=Y1
XE 270 POKE 23030,X4+45:POKE 23031,Y4+36:
   GOSUB VB
RC 280 A=USR(CY,S1,X1,Y1,X1+X3,Y1+Y3,S2,X
   1,V96,V3):A=USR(CY,S2,Z,Z,X3,Y3,S1,X1,
   Y1,C2)
AJ 290 X4=X1:Y4=Y1:GOSUB VB:ON T=V1 GOTO
   280
KR 300 IF T=V33 THEN A=USR(CY,S2,X4,V96,X
   4+X3,Y3+V96,S1,X4,Y4,V3):POKE 53279,Z:
   GOTO 900
UM 310 IF T THEN GOSUB VC
BJ 320 ON X1=X4 AND Y1=Y4 GOTO 290:A=USR(
   CY,S2,X4,V96,X4+X3,Y3+V96,S1,X4,Y4,V3)
   :GOTO 280
DQ 389 REM HALVE&DOUBLE
ZB 390 GOSUB VM:? "K▶▶▶▶Halve":IF X=V8 TH
   EN ? "K▶▶▶▶Double"
WN 400 ? "▶▶▶▶H↓←V↓←OK";:Y=V1
XO 410 POSITION 30,Y:? " ";:GOSUB VA:Y1=I
   NT(Y1/V8):IF Y1<V1 OR Y1>V3 THEN RETUR
   N
VG 420 IF Y1<V3 THEN Y=Y1:GOTO 410
CO 430 X=V8/X:? "KPlease wait↑↑":ON Y=V2
   GOTO 470:IF X3>79 AND X=V1 THEN X3=79
FK 450 FOR Y=Z TO X3 STEP X:A=USR(CY,S2,Y
   ,Z,Y,Y3,S2,Y*V2/(X*X),V96,V4)
AA 460 A=USR(CY,S2,Y,Z,Y,Y3,S2,Y*V2/(X*X)
   +ABS(V2-X),V96,V4):NEXT Y:POKE 203,(X3
   +V1)*V2/X/X:X3=PEEK(203)-V1:GOTO 560
SA 470 IF Y3>47 AND X=V1 THEN Y3=47
GZ 480 FOR Y=Z TO Y3 STEP X:A=USR(CY,S2,Z
   ,Y,X3,Y,S2,Z,V96+Y*V2/(X*X),V4)
FN 490 A=USR(CY,S2,Z,Y,X3,Y,S2,Z,V96+Y*V2
   /(X*X)+ABS(V2-X),V4):NEXT Y:POKE 203,(
   Y3+V1)*V2/X/X:Y3=PEEK(203)-V1:GOTO 560
IY 499 REM FLIP
HX 500 GOSUB VM:? "K▶▶▶Flip↓←H↓←V↓←OK";:Y
   =V1
FY 510 POSITION 25,Y:? " ";
IN 520 GOSUB VA:Y1=INT(Y1/V8):IF Y1<V1 OR
    Y1>V3 THEN RETURN
WM 530 IF Y1<V3 THEN Y=Y1:GOTO 510
RL 540 ? "KPlease wait↑↑":IF Y=V1 THEN FO
   R X=Z TO X3:A=USR(CY,S2,X,Z,X,Y3,S2,X3
   -X,V96,V4):NEXT X:GOTO 560
KR 550 FOR X=Z TO Y3:A=USR(CY,S2,Z,X,X3,X
   ,S2,Z,V96+Y3-X,V3):NEXT X
JH 560 A=USR(CY,S2,Z,V96,X3,Y3+V96,S2,Z,Z
   ,V3):GOTO 260
OR 899 REM FRAME
AB 900 GOSUB VA:X2=X1:Y2=Y1:A=USR(CY,S1,X
   2,Y2,159,Y2+95,S2,X2,V96,V3)
OH 910 TRAP 920:IF AN THEN POKE 23030,X2+
   ASC(A$(V3))-ASC(A$(V1))+45:POKE 23031,
   Y2+ASC(A$(V4))-ASC(A$(V2))+36
GB 920 GOSUB VB:IF Y1<Y2 THEN Y1=Y2
VE 930 IF X1<X2 THEN X1=X2
SO 940 IF Y1-Y2>95 THEN Y1=Y2+95
WM 950 X=USR(23540,X2,Y2,X1,Y1,C1):A=USR(
   CY,S2,X2,V96,X1,V96+Y1-Y2,S1,X2,Y2,V3)
   :IF  NOT T THEN 920
PC 960 IF T>V1 AND T<>V33 THEN GOSUB VC:G
   OTO 920
BQ 970 ON T=V33 GOTO 900:IF  NOT AN THEN
   A=USR(CY,S1,X2,Y2,X1,Y1,S2,Z,Z,V4):X3=
   X1-X2:Y3=Y1-Y2:X4=X2:Y4=Y2:GOTO 260
BA 980 Y=LEN(A$):IF Y=V2*VC THEN 2200
HZ 990 A$(Y+V1)=CHR$(X2):A$(Y+V2)=CHR$(Y2
   ):A$(Y+V3)=CHR$(X1):A$(Y+V4)=CHR$(Y1):
   POKE 23030,X2+45:POKE 23031,Y2+36
QC 1000 GOTO 900
DZ 1599 REM CURSOR
SS 1600 P1=ABS(P1-V1)
EF 1610 S1$="♥♣♣♥F♥♣♣♥":IF P1 THEN S1$="♥
   ♥♥♥♣♥♥♥♥"
CY 1620 FOR T=V1 TO 9:POKE 23032+T,ASC(S1
   $(T,T)):NEXT T:RETURN
ZA 1699 REM MENU
HQ 1700 POP :Y=20828:GOSUB 2010
XK 1710 GOSUB VB:ON  NOT T GOTO 1710:ON T
   =V1 OR T=V33 OR T=12 GOTO 1720:GOSUB 1
   10:GOTO 1710
YD 1720 IF X1>159 OR Y1<V4 OR Y1>35 OR T=
   V33 THEN GOSUB 260:GOTO 1700
RN 1730 X=INT((Y1-V4)*0.0634)*V4+INT(X1*0
   .0246)+V1:ON X GOSUB 2100,3000,500,390
   ,2500,3300,2200,390:GOTO 1700
DV 2000 Y=40524
LH 2010 POKE 559,Z:S1$="♥♣♣♣[♣♣♣♥":GOSUB
   1620:POKE 22876,V1:M=M1:X1=15:Y1=191:Y
   2=S1+1640:GOSUB 2090
TK 2020 POKE DL+468,65:POKE DL+469,PEEK(5
   60):POKE DL+470,PEEK(561):M=66:X1=V3:Y
   1=V8:Y2=Y:GOSUB 2090
YD 2030 POKE 709,10:POKE 710,V2:FOR Y=18
   TO VA:POKE DL+Y,Z:NEXT Y:POKE DL+15,19
   4:POKE 559,62:RETURN
EV 2050 POKE 22876,INT(SP*1.3):M=M1:X1=V3
   :Y1=195:Y2=S1:POKE DL+579,65:POKE DL+5
   80,PEEK(560):POKE DL+581,PEEK(561)
SL 2060 POKE 709,PEEK(CR+V2):POKE 710,PEE
   K(CR+V3):GOSUB 2090:GOSUB 1610:RETURN
CT 2090 T=USR(23138,Y2,M,X1,Y1,Z):RETURN
PP 2099 REM MODE
NC 2100 IF M1=79 THEN M1=78:RETURN
SF 2110 M1=79:RETURN
QA 2199 REM ANIMATE
WJ 2200 AN=Z:GOSUB VM:? "K▶▶Animate↓ New↓
   ⇧▶▶▶Append↓⇧▶▶▶Play↓"
TW 2210 GOSUB VA:X=INT(Y1/V8):ON X<V1 OR
   X>V3 GOTO 1700:ON X=V3 GOTO 200:IF X=V
   1 THEN A$=""
```

```
OL 2220 AN=V1:GOSUB VN:GOTO 900
UB 2499 REM DISK
AV 2500 GOSUB VM
GJ 2510 ? "↰A. Directory▸▸E. Rename File"
UK 2520 ? "B. Main Menu▸▸F. Load File"
RX 2530 ? "C. Format Disk▸G. Save File"
MY 2540 ? "D. Delete File▸H. Unlock File"
SM 2550 TRAP 2550:POKE 84,V4:? "↑Select i
   tem or return:";:INPUT #16,S1$:IF S1$=
   "" THEN 2510
QR 2560 X=ASC(S1$)-64:ON X<V1 OR X>V8 GOT
   O 2550:IF X=V2 THEN RETURN
IM 2590 ON X<V4 OR X=5 GOTO 2620:? "↑↑FIL
   ENAME or return:";:INPUT #16,S1$:IF S1
   $="" THEN 2550
SB 2600 IF S1$(V2,V2)=":" OR S1$(V3,V3)="
   :" THEN 2620
QX 2610 X1=LEN(S1$)+V1:S1$(X1)=S1$(V1,X1-
   V1):S1$(V1,V2)="D:":S1$(V3)=S1$(X1,X1+
   X1-V2)
AV 2620 TRAP 2780:ON X GOSUB 2630,2570,26
   60,2680,2690,2710,2710,2770:CLOSE #V1:
   GOTO 2550
HF 2630 OPEN #V1,6,Z,"D:*.*":TRAP 2650:?
   "↰";
DS 2640 POSITION Z,Z:? "↑":POSITION V2,V3
   :INPUT #V1,S1$:? S1$,;:INPUT #V1,S1$:?
    S1$:GOTO 2640
BB 2650 RETURN
BC 2660 ? "↰Y to format drive #1 or retur
   n:";:INPUT #16,S1$:IF S1$<>"Y" THEN RE
   TURN
TT 2670 XIO 254,#V1,Z,Z,"D:":RETURN
JJ 2680 XIO V33,#V1,Z,Z,S1$:RETURN
QV 2690 ? "↰D:OLDNAME,NEWNAME or return:"
   :INPUT #16,S1$:IF S1$="" THEN RETURN
BS 2700 XIO 32,#V1,Z,Z,S1$:RETURN
XV 2710 X=(X-5)*V4:OPEN #V1,X,Z,S1$:ON X=
   V8 GOTO 2730:ON S1$(LEN(S1$))<>"♣" GOT
   O 2730:FOR Y=Z TO 12:GET #V1,X1:NEXT Y
WH 2720 FOR Y=V1 TO V3:GET #V1,X1:POKE CR
   +Y,X1:NEXT Y:GET #V1,X1:GET #V1,X1:POK
   E CR,X1:GOTO 2760
NT 2730 GOSUB VN:POKE 850,X+V3:POKE 852,8
   0:POKE 853,97:POKE 856,Z:POKE 857,30:T
   =USR(ADR("hhh☒LVd"),16)
QA 2740 T=Z:IF X=V8 THEN FOR X=Z TO V3:PU
   T #V1,PEEK(CR+X):NEXT X:GOSUB VM:RETUR
   N
BK 2750 FOR X=Z TO V3:GET #V1,Y:POKE CR+X
   ,Y:NEXT X
YC 2760 X=USR(ADR(S2$)):GOSUB VM:RETURN
EF 2770 XIO 36,#V1,Z,Z,S1$:RETURN
FT 2780 X=PEEK(195):GOSUB VM:? "↰Error-↰"
   ;X:IF X=160 THEN RUN
BP 2790 RETURN
SI 2999 REM FILL
NS 3000 GOSUB VM:? "↰Fill 0 with ";PEEK(1
   643):FOR X=V1 TO V3:? "▸";X;" with ";P
   EEK(1643+X):NEXT X:? "▸OK"
NX 3010 GOSUB VA:Y1=INT(Y1/V8):IF Y1>V4 T
   HEN RETURN
KS 3020 IF Y1=V4 THEN ? "↰Please wait↑↑":
   X=USR(1552):GOTO 260
TF 3030 X=PEEK(1643+Y1):X=X+V1:IF X=V4 TH
   EN X=Z
OU 3040 POKE 1643+Y1,X:POSITION 14,Y1:? X
   ;"←";:GOTO 3010
CS 3299 REM CEL
ZW 3300 GOSUB VM:? "↰▸    Cel 0↓←1↓←2↓←3↓
   ←N↓←OK";:POSITION 14,(C2/64):? " ";:GO
   SUB VA:ON Y1>39 GOTO 1700
MQ 3310 C2=INT(Y1/V8)*64:IF C2=256 THEN C
   2=255
QL 3320 GOTO 260
XG 8000 POKE 559,Z:POKE 566,143:POKE 567,
   231:POKE 54279,80
UJ 8010 READ V1,V2,V3,V4,V8,V33,V96,VA,VB
   ,VC,VM,VN,CY,S1,S2,C1,C2,CR,SP,DL:DIM
   A$(V2*VC),S1$(63),S2$(26)
OZ 8020 READ S1$:FOR X=Z TO V1:READ A$:Y=
   USR(ADR(S1$),ADR(A$),1536+X*VC,LEN(A$)
   ):NEXT X
JF 8030 FOR X=Z TO 9:READ A$:Y=USR(ADR(S1
   $),ADR(A$),22841+X*VC,LEN(A$)):NEXT X
VS 8040 FOR X=Z TO V1:READ A$:Y=USR(ADR(S
   1$),ADR(A$),20828+X*VC,LEN(A$)):NEXT X
   :A$="":POKE 560,80:POKE 561,93
OJ 8060 FOR X=Z TO V2:POKE DL+X,112:NEXT
   X:M1=79:GOSUB VM:? "↰Picture Perfect↓↑
   by↓↑Joe D. Brzuszek↓↓↑Press start"
NL 8070 ON PEEK(53279)=7 GOTO 8070:M1=78:
   GOTO 1700
JI 9000 DATA 1,2,3,4,8,33,96,20,50,100,20
   00,2050,23253,24912,32624,1,255,1788,1
   ,23888
WL 9009 REM MOVE STRING
ZA 9019 REM DLI,FILL
KN 9039 REM VBI,DISPLAY,COPY,FRAME
DB 9199 REM MENU
```

LISTING 2: BASIC

```
QY 1 REM PICTURE PERFECT - LISTING2
NS 2 REM by Joe D. Brzuszek
TF 3 REM Creates lines 8050 & 9010-9210
TV 4 REM and saves them in D:PICT.LST
NK 5 REM
IT 6 REM ENTER "D:PICT.LST" to load
NM 7 REM
CA 10 DIM H$(101):LINE=600:LAST=900
SN 20 GRAPHICS Z:POKE 752,1:POKE 710,2
UD 30 OPEN #1,8,Z,"D:PICT.LST":? "OPENING
    D:PICT.LST FOR OUTPUT":TRAP 380
DG 40 READ H$:IF LEN(H$)<>100 AND LINE<>L
   AST THEN ? "DATA LENGTH ERROR IN LINE
   ";LINE:GOTO 400
KC 50 LN=PEEK(183)+PEEK(184)*256:IF LN<>L
   INE THEN ? "LINE ";LINE;" IS MISSING!"
   :GOTO 400
DL 60 ? "CHECKING LINE ";LINE:FOR X=1 TO
   99 STEP 2:Y1=ASC(H$(X,X))-48:Y2=ASC(H$
   (X+1,X+1))-48
IW 70 IF Y1>9 THEN Y1=Y1-7:IF Y1>15 THEN
   390
LV 80 IF Y2>9 THEN Y2=Y2-7:IF Y2>15 THEN
   390
LG 90 PUT #1,Y2+Y1*16:NEXT X:LINE=LINE+10
   :GOTO 40
TZ 380 IF PEEK(195)=5 AND LINE=LAST THEN
   ? "NO ERRORS FOUND, BUT SAVE LISTING2
   AS A BACKUP JUST IN CASE!":GOTO 400
OE 390 ? "ERROR AT LINE ";LINE
XT 400 ? "↰CLOSING FILE":POKE 752,Z:END
LY 600 DATA 383035305332243D2268ADFC068DC
   802ADFD068DC402ADFE068D0206ADFF068D0A0
   660223A583D555352284144522853322 4
RZ 610 DATA 29293A583D555352284144522822 6
   8A907A039A259205CE4A9068D0102A9008D000
   2A9C08D0ED4602229299B393031304441
XR 620 DATA 5441686885CC6885CB6885CE6885C
   D681865CE85D0681865CD85CF9002E6D0A000B
   1CB91CDE6CBD002E6CCE6CDD002E6CEA5
EH 630 DATA CDC5CFD0EAA5CEC5D0D0E4609B393
   0323044415441 48A9CA8D0AD48D17D0A9948D1
   8D0684068A97085CBA97F85CCA2048E6F
GL 640 DATA 06A000B1CBCE6F06F0054A4A4C220
   62903A8B96B068D7006A9038E6F06CE6F06F00
   B0A0A0E70060E70064C3A0649FFA00031
PQ 650 DATA CB0D700691CBCAD0C3E6CBD002E6C
```

continued on page 56

BASIC Editor II

by Clayton Walnum

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

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

Typing in the Editor

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

Disk version:

(1) Type in Listing 1, then verify your work with Unicheck (see Issue 39).

(2) Save the program to disk with the command *SAVE "D:EDITORLI.BAS"*.

(3) Clear the computer's memory with the command *NEW*.

(4) Type in Listing 2, then verify your work with Unicheck.

(5) Run the program (after saving a backup copy) and follow all the on-screen prompts. A data file will be written to your disk.

(6) Load Listing 1 with the command *LOAD "EDITORLI.BAS"*.

(7) Merge the file created by Listing 2 with the command *ENTER "D:ML.DAT"*.

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

Cassette version:

(1) Type in Listing 1 and verify your work with Unicheck.

(2) Save the program to cassette with the command *CSAVE*. (Do not rewind the cassette.)

(3) Clear the computer's memory with the command *NEW*.

(4) Type in Listing 2 and verify your work with Unicheck.

(5) Run the program and follow the on-screen prompts. A data file will be written to your cassette.

(6) Rewind the cassette.

(7) Load Listing 1 with the command *CLOAD*.

(8) Merge the file created by Listing 2 with the command *ENTER "C:"*.

(9) On a new cassette, save the resultant program with the command *LIST "C:"*.

Using the Editor

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

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

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

Your screen will now be divided into two "windows." The upper window will display each line after it's processed, as well as the checksum generated for that line. The lower window is where program lines are typed and edited.

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

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

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

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

Leaving the Editor

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

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

Large listings

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

The post-processor

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

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

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

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

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

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

Murphy's Law

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

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

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

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

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

Listing 1.
BASIC listing.

```
32600 IF FL THEN 32616
32602 DIM L$(115),5V$(115),C2$(2),B$(1
15),M$(119),5$(98),E$(69),A$(1):FL=1:5
TMTAB=PEEK(136)+PEEK(137)*256
32604 GRAPHICS 0:POKE 710,0:P=0:ABR=0:
? "ALLOW ABBREVIATION5";:INPUT A$:IF A
$="Y" OR A$="y" THEN ABR=1
32606 B$(1)=" ":B$(115)=" ":B$(2)=B$
32616 OPEN #17,4,0,"E:":L$=" ":GOSUB 3
2662:5TART=0
32618 POKE 766,1:POKE 83,39:POSITION 1
,3:IF LEN(L$)<39 THEN ? L$:GOTO 32624
32620 IF LEN(L$)<77 THEN ? L$(1,38):?
L$(39,LEN(L$)):GOTO 32624
32622 ? L$(1,38):? L$(39,76):? L$(77,L
EN(L$))
32624 POKE 752,0:POKE 766,0:POKE 559,3
4:POKE 82,1:POKE 83,38:POSITION 0,10:?
 " ";:INPUT #17;L$:POKE 766,1
32626 IF (L$="P" OR L$="p") AND 5TART=
0 THEN P=1:L$=""
32628 IF L$="E" OR L$="e" THEN E=1:POS
ITION 1,10:? 5V$ GOTO 32624
32630 IF L$="Q" OR L$="q" THEN 32690
32632 IF L$="" AND P=1 THEN 32686
32634 IF L$="" THEN 32624
32636 IF L$="B" OR L$="b" THEN GRAPHIC
5 0:? "TYPE 'GOTO 32600' TO CONTINUE":
END
32638 IF L$(1,1)="E" OR L$(1,1)="e" TH
EN E=1:TRAP 32624:EL=VAL(L$(2)):POSITI
ON 1,9:LIST EL:GOTO 32624
32640 5V$=L$:TRAP 32624:X=VAL(L$)
32642 5TART=1:IF P AND  NOT E THEN 326
52
32644 GOSUB 32674:IF  NOT ABR OR P THE
N 32652
32646 POKE 766,0:? CHR$(125):POSITION
0,3:L=VAL(L$):LIST L:? :? :? "CONT":L$
=B$
32648 POSITION 0,0:POKE 842,13:5TOP
32650 POKE 842,12:A=U5R(ADR(5$),ADR(L$
),4):L$=L$(1,A)
32652 CHK5UM=U5R(ADR(M$),ADR(L$),LEN(L
$)):CHK5UM=CHK5UM+PEEK(1542)*65536
32654 CHK=CHK5UM-(INT(CHK5UM/676)*676)
:HI=INT(CHK/26):LO=CHK-(HI*26):C2$(1)=
CHR$(HI+65):C2$(2)=CHR$(LO+65)
32656 IF  NOT P OR E THEN E=0:GOSUB 32
662:IF  NOT P THEN 32660
32658 POKE 83,39:POKE 752,1:FOR X=3 TO
 5:POSITION 1,X:? B$(1,38):POSITION 1,
X+7:? B$(1,38):NEXT X:POKE 83,38
32660 POKE 766,1:POKE 83,38:POSITION 6
,7:? C2$:POKE 752,0:GOTO 32618
32662 GOSUB 32702:POKE 766,0:POKE 752,
1:? "K":POKE 82,1:DL=PEEK(560)+256*PEE
K(561)+4
32664 POKE DL-1,70:POKE DL+2,6:POKE DL
+3,112:POKE DL+4,112:POKE DL+5,112:POK
E DL+13,112:POKE DL+14,112
32666 POKE DL+22,112:POKE DL+23,112:PO
KE DL+24,65:POKE DL+25,PEEK(560):POKE
DL+26,PEEK(561):POKE 83,39
32668 POSITION 20,0:? "  basic editor
II  ":POSITION 0,7:? "                    TYP
ING WINDOW                  "
32670 POSITION 0,1:? "                    MESS
AGE WINDOW                  ":POSITION 1,7
:? "CODE:";
32672 POKE 559,34:RETURN
32674 GRAPHICS 0:POKE 559,0:POKE 766,1
:POKE 82,0:POKE 83,39:POSITION 0,3:? L
$:? :? :? :? "CONT":POSITION 0,0
32676 POKE 842,13:5TOP
32678 POKE 842,12:TRAP 32682:A=U5R(ADR
(E$),VAL(L$)):IF A=4 THEN POP :GOTO 32
682
32680 RETURN
32682 GOSUB 32662:5OUND 0,75,10,8:FOR
X=1 TO 20:NEXT X:5OUND 0,0,0,0:POSITIO
N 1,3:? "SYNTAX ERROR!":POKE 766,1
32684 POKE 83,38:POSITION 1,10:? 5V$:G
OTO 32624
32686 LINE=PEEK(5TMTAB)+PEEK(5TMTAB+1)
*256:IF LINE>32599 THEN 32690
32688 OF5=PEEK(5TMTAB+2):5TMTAB=5TMTAB
+OF5:POSITION 1,9:LIST LINE:GOTO 32624
32690 POKE 766,0:POSITION 1,10:? "READ
Y TO QUIT";:INPUT A$:IF A$<>"Y" THEN P
OSITION 1,10:? B$(1,38):GOTO 32624
32692 GRAPHICS 0:? :? :? :FOR X=32600
TO 32636 STEP 2:? X:NEXT X:? "CONT":PO
SITION 0,0:POKE 842,13:5TOP
32694 POKE 842,12:GRAPHICS 0:? :? :? :
FOR X=32638 TO 32674 STEP 2:? X:NEXT X
:? :? "CONT":POSITION 0,0
32696 POKE 842,13:5TOP
32698 POKE 842,12:GRAPHICS 0:? :? :? :
FOR X=32676 TO 32702 STEP 2:? X:NEXT X
:? :? "POKE 842,12":POSITION 0,0
32700 POKE 842,13:5TOP
32702 POKE 16,112:POKE 53774,112:RETUR
N
```

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

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

Listing 2.
BASIC listing.

```
10 DIM L$(120),ML$(119),A$(1)
20 GRAPHICS 0:POKE 710,0:? "DISK OR CA
SSETTE";:INPUT A$:IF A$<>"C" AND A$<>"
D" THEN 20
30 IF A$="C" THEN 50
40 ? "PLACE FORMATTED DISK IN DRIVE":?
 "THEN PRESS RETURN":INPUT L$:OPEN #1,
8,0,"D:ML.DAY":GOTO 60
50 ? :? "READY CASSETTE, PRESS RETURN"
;:INPUT L$:OPEN #1,8,0,"C:"
60 L$="32608 M$(1)=":L$(13)=CHR$(34)
70 N=119:GOSUB 130:L$(14)=ML$(1,58):L$
(LEN(L$)+1)=CHR$(34):? #1;L$
80 L$(1)="32610 M$(59)=":L$(14)=CHR$(3
4):L$(15)=ML$(59):L$(LEN(L$)+1)=CHR$(3
4):? #1;L$
90 L$(1)="32612 5$=":L$(10)=CHR$(34)
100 ML$="":N=98:GOSUB 130:L$(11)=ML$:L
$(LEN(L$)+1)=CHR$(34):? #1;L$
110 L$(1)="32614 E$=":L$(10)=CHR$(34)
120 ML$="":N=69:GOSUB 130:L$(11)=ML$:L
$(LEN(L$)+1)=CHR$(34):? #1;L$:END
130 FOR X=1 TO N:READ A:ML$(X)=CHR$(A)
:NEXT X:RETURN
140 DATA 104,104,133,204,104,133,203,1
04,104,133,205,169,0,141,3,6,141,2,6,1
41,4,6,141,5,6
150 DATA 141,6,6,238,3,6,32,68,218,172
,2,6,177,203,133,212,32,170,217,32,182
,221,32,68,218
160 DATA 173,3,6,133,212,32,170,217,32
,219,218,32,210,217,165,212,141,0,6,16
5,213,141,1,6,24
170 DATA 173,0,6,109,4,6,141,4,6,173,1
,6,109,5,6,141,5,6,144,3,238,6,6,238,2
180 DATA 6,172,2,6,196,205,208,176,173
,4,5,133,212,173,5,6,133,213,96
190 DATA 104,104,133,204,104,133,203,1
04,104,141,255,6,169,0,133,213,216,165
,88,133,205,165,89,133,206
200 DATA 174,255,6,24,165,205,105,40,1
33,205,144,2,230,206,202,208,242,160,0
,177,205,201,64,144,18
210 DATA 201,96,144,19,201,128,144,18,
201,192,144,6,201,224,144,7,176,8,24,1
05,32,144,3,56,233
220 DATA 64,145,203,200,192,114,240,2,
208,215,177,203,201,32,208,3,136,208,2
47,200,132,212,96
230 DATA 104,104,141,254,6,104,141,253
,6,169,0,133,213,216,165,136,133,205,1
65,137,133,206,160,0,177
240 DATA 205,205,253,6,208,8,200,177,2
05,205,254,6,240,15,160,2,177,205,24,1
01,205,133,205,144,228
250 DATA 230,206,176,224,160,4,177,205
,201,55,240,4,160,0,240,0,132,212,96
```

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

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

END USER

by Arthur Leyenberger

Recently, I was preparing to attend yet another computer show. In a small effort to get organized, I normally update my list of industry contacts prior to leaving for the show. This time was no different—add a few new names here, delete a few names there—instant new contact list.

In the seven years I have been going to these trade shows, much has changed. Change couldn't be more evident than within the world of Atari. Since I have kept all of my old contact lists—going back to 1984—I have what amounts to an archaeological record of companies and products that have both become a household word and later are all but forgotten.

In late 1984, the Atari 8-bit computer was in its prime. Despite the ongoing confusion caused by the "old Atari" and the "new old Atari," there were dozens of companies supporting the machine with products ranging from the innovative to the dogs. My historical file also shows a number of trends which have come, gone, and well, come again.

The early and mid-1980s saw a number of companies jumping on the software bandwagon. Many of these didn't have the staying power either in management prowess or quality products to endure. Traditional board game companies such as Parker Brothers, Ideal (CBS), and Milton-Bradley saw the boom in computer games and decided to get in on the action.

Some of the games from these companies stand out as nostalgic reminders of a more innocent age of software. For example, CBS's *K-razy Shoot Out* was an adaptation of the arcade game *Berzerk*. Killing robot sentries always seemed to be an enjoyable and challenging pastime and one which I spent many hours playing. Although simplistic by today's standards, it was good for its time.

Yet CBS Software had its share of mediocre titles. *K-Star Patrol* never reached the popularity of *K-razy Shoot Out*. In this shoot-'em-up game, you had to destroy alien ships as you moved through enemy space sectors. One of the main problems with *K-Star Patrol* was the sensitivity of the joystick. A required maneuver involved rotating the joystick through all four positions to release a bomb. I never did master this technique, which made the game virtually not playable for me.

Boulders and Bombs was another CBS disappointment. It was difficult to play, uninteresting, and not challenging. On the other hand, CBS Software had better success with their Sesame Street-inspired children's educational games. Titles like *Astro Grover* were good back then and are still considered some of the best available educational games.

Not so good, by any standards, was Parker Brothers' *Popeye*. It looked much like a remake of Epyx's *Jumpman* and *Jumpman Junior* (both excellent games). *Miner 2049er* (Big Five Software) was a much better implementation of this type of climbing and jumping game. And, of course, *Donkey Kong* (Atari) was a tremendous hit both in the arcade as well as on the Atari 8-bit machine.

Parker Brothers should not be thought of as a maker of poor games. One of their big hits, if not the biggest, was *Q*Bert*. The Atari cartridge was a true representation of the arcade game. Although the theme was simple, *Q*Bert* was (is) the type of game that can hold your interest. It also qualifies for the non-violent game hall of fame since the game can be played without "killing the aliens."

Looking down my list of contacts I see the name Educalc. This was one of the many small companies producing quality software without attracting very much attention. I still have two of their titles: *Addition and Subtraction* and *Multiplication and Division*.

Although both of these software titles lacked advanced graphics, they were good at teaching basic math through the use of drill and practice. Each offered several different "games" which one or more children could play. Each game taught a different aspect of math, such as straightforward counting or counting by increments.

I particularly liked the way the games handled errors. Some drill and practice programs would not permit the user to advance until the correct answer was given. This heavy-handed approach would lead to frustration and little learning. Educalc used a gentler approach that gave the youngster several chances before presenting the correct answer.

Unfortunately, I don't think you can find Educalc games anymore, even on the discounters' shelves. That's a shame.

Another long-forgotten name on my list is Maximus. Maximus was a management consulting company located in Virginia, I be-

lieve. They had two products: *Storyline* and *Safetyline* which both used a combination of disk and cassette tape.

The tape contained a story narration that was synchronized to graphics images on the disk. The animation and mouth movements of the characters matched the narration on the tape. It was all well done. *Safetyline* also included several games that reinforced the concepts (safety tips for crossing the street and what to do if lost) taught by the story.

I occasionally still see *Storyline* and *Safetyline* for sale in some software stores and toy outlets. If you have both a disk drive and a cassette recorder attached to your Atari computer, and you have children aged four to seven years, you should check out these programs. They are well-designed and fun for children.

Another company name I see on my list is Eastern House Software. I don't know what happened to them but they had an excellent product called the *Monkey Wrench*. It was a cartridge that fit into the right slot of an Atari 800 and gave Atari cartridge BASIC the power it should have had from the start.

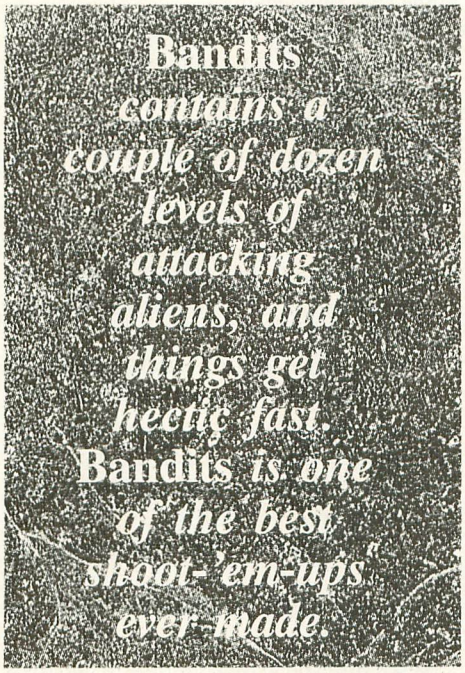
Monkey Wrench provided automatic line numbering and renumbering, allowed you to delete ranges of lines and had a hexadecimal/decimal number conversion utility. You could also use it to display the contents of memory without leaving BASIC. I still have mine, but the availability of Microsoft BASIC, BASIC XL and others has forced *Monkey Wrench* to collect its share of dust.

It's been a long time since Odesta has marketed 8-bit programs, having left for the green pastures of the Mac world. Still, its *Chess*, *Checkers*, and *Odin* (otherwise known as Othello and Reversi) games are exceptional examples of quality software. What made these products definitive video board-game translations was the user interface (still one of the best I've seen on any computer) and the excellent documentation, which not only included game play but also histories of the game and strategy hints.

These Odesta games also had a plethora of options that let you play at many levels, take back any number of moves, switch players during a game, request help, and play back the entire game much like a movie. These games were as much a means for learning as they were for playing, and they still can be found if you are willing to look hard for them. If you don't have one, you ought to pick one up. You'll then have something with which to compare every other 8-bit product you own.

I forgot about a company called Tronix. They had a couple of games in the mid-1980s, but the one I remember is *Juice*. One of the best of the jumping/hopping genre, it was in some ways a better game than the more popular *Q*Bert*. *Juice* used a three-dimensional or perspective view for you to hop around on.

The goal was to jump on every square of the board, leaving a trace which completed a "circuit." Once the circuit was complete, you advanced to the next level. At higher levels it took two or more jumps to turn a square into an electrical trace. Like many of the classic games, *Juice* was simple in concept but had depth.



I had to do a little digging through the "junk," ah, old software box, to find out why Utopia Software was on my list. I found it: *Pinhead*. That's the name of a game that captured my fancy years ago in which you move a clown on a unicycle across the bottom of the screen in order to catch falling balloons on his head.

I know it sounds lame, but the circus-style music and excellent graphics added to the excitement. *Pinhead* was modeled after *Kickman* which made the rounds at the arcade at the time. I first saw this game at a users' group meeting on a projection TV. Once I saw it I was hooked.

Not every software title can be a hit. Few companies learned this lesson better than Sirius. Sirius had two of my favorite 8-bit games as well as more than their share of turkeys. *Sneakers*, one of their better efforts, consumed a good portion of my life.

Sneakers was a light-hearted *Space Invaders*-type game in which cute and humorous creatures descended from the top and sides of the screen as targets for you to shoot at. There were eight or nine types of creatures, but I recall only the Sneakers, Daggers and Cyclops. Like other games of this ilk, each screen brought faster and deadlier critters, and the object was to score as high a score as possible. I no longer have my *Sneakers* disk, but if I did, I would play a round right now. It always brought a smile to my face.

The other excellent Sirius game was *Bandits*. I call it a *Galaxian*-to-the-max type of game. Like *Galaxian*, you move a gun turret across the bottom of the screen, aiming and shooting at the enemy ships that descend. Some of the enemies shoot back, others dive and regroup and they come in groups of twos, threes and fives.

Bandits contains a couple of dozen levels of attacking aliens, and things get hectic fast. *Bandits* is one of the best shoot-'em-ups ever made. I wore out one Atari joystick with this game and suffered multiple calluses on my fingers.

Sirius had some awful titles as well. *Alpha Shield* and *Repton* were mediocre at best. *Cyclod* and *Fast Eddie* were poor excuses for even a VCS game. *Squish 'Em* and *Wavy Navy* were uninspired and dull. Too bad Sirius is no longer around because when they did it right, they did it right.

Many more names appear on my old contact lists, but I have run out of space. Some of the products were not worth purchasing when they were new, but there are hundreds more that were good. The good news is that many of the products from companies no longer in business can still be purchased. Like Classic Coke, some of these programs have been purchased, repackaged and reissued. In many cases, the programs now cost less than they did when they were new.

So the next time someone says there isn't any good software for the 8-bit Atari, tell them they are not looking hard enough for it. All in all, there are plenty of programs of all types to choose from for your computer.

Arthur Leyenberger is a computer analyst and free-lance writer who works out of his home in New Jersey. He can be reached on CompuServe at 71266,46 or on DELPHI as ARTL.

U T I L I T Y

M/L EDITOR

For use in machine-language entry.

by Clayton Walnum

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

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

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

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

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

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

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

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

M/L Editor will display, at the top of the screen, the number of the line you're currently working on. As you go through the line, you'll be prompted for each entry. Simply type the number and press Return. If you press Return without a number, the default is the last value entered.

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

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

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

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

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

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

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

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

LISTING 1: BASIC LISTING

```
AZ 10 DIM BF(16),N$(4),A$(1),B$(1),F$(15)
   ,F1$(15)
LF 11 DIM MOD$(4)
BN 20 LINE=1000:RETRN=155:BACK5P=126:CHK5
   UM=0:EDIT=0
GO 30 GO5UB 450:POSITION 10,6:? "Start or
   Continue? ";:GO5UB 500:? CHR$(A)
ZG 40 POSITION 10,8:? "FILENAME";:INPUT F
   $:POKE 752,1:? " "
FE 50 IF LEN(F$)<3 THEN POSITION 20,10:?
   " ":GOTO 40
NF 60 IF F$(1,2)<>"D:" THEN F1$="D:":F1$(
   3)=F$:GOTO 80
KL 70 F1$=F$
TN 80 IF CHR$(A)="5" THEN 120
FD 90 TRAP 430:OPEN #2,4,0,F1$:TRAP 110
HQ 100 FOR X=1 TO 16:GET #2,A:NEXT X:LINE
   =LINE+10:GOTO 100
WM 110 CLOSE #2:OPEN #2,9,0,F1$:GOTO 170
VT 120 TRAP 160:OPEN #2,4,0,F1$:GO5UB 440
   :POSITION 10,10:? "FILE ALREADY EXIST5
   !!";POKE 752,0
ZU 130 POSITION 10,12:? "ERA5E IT? ";:GO5
   UB 500:POKE 752,1:? CHR$(A)
VH 140 IF CHR$(A)="N" OR CHR$(A)="n" THEN
   CLOSE #2:GOTO 30
QG 150 IF CHR$(A)<>"Y" AND CHR$(A)<>"y" T
   HEN 130
BH 160 CLOSE #2:OPEN #2,8,0,F1$
IE 170 GO5UB 450:POSITION 10,1:? "NOW ON
   LINE: ";LINE:CHK5UM=0
GH 180 L1=3:FOR X=1 TO 16:POSITION 13*(X<
   10)+12*(X>9),X+2:POKE 752,0:? "BYTE #"
   ;X;": ";:GO5UB 310
KH 190 IF EDIT AND L=0 THEN BYTE=BF(X):GO
   TO 210
FY 200 BYTE=VAL(N$)
OZ 201 MOD$=N$
BU 210 POSITION 22,X+2:? BYTE;"  "
YZ 220 BF(X)=BYTE:CHK5UM=CHK5UM+BYTE*X:IF
   CHK5UM>9999 THEN CHK5UM=CHK5UM-10000
MS 230 NEXT X:CHK5UM=CHK5UM+LINE:IF CHK5U
   M>9999 THEN CHK5UM=CHK5UM-10000
IG 240 POSITION 12,X+2:POKE 752,0:? "CHEC
   K5UM: ";:L1=4:GO5UB 310
EW 250 IF EDIT AND L=0 THEN 270
QH 260 C=VAL(N$)
SY 270 POSITION 22,X+2:? C;"   "
IL 280 IF C=CHK5UM THEN 300
DI 290 GO5UB 440:EDIT=1:CHK5UM=0:GOTO 180
LW 300 FOR X=1 TO 16:PUT #2,BF(X):NEXT X:
   LINE=LINE+10:EDIT=0:GOTO 170
FV 310 L=0
KZ 320 GO5UB 500:IF (A=A5C("Q") OR A=A5C(
   "q")) AND X=1 AND  NOT EDIT THEN 420
PO 330 IF A<>RETRN AND A<>BACK5P AND (A<4
   8 OR A>57) THEN 320
DX 331 IF A=RETRN AND N$="" THEN N$=MOD$
TD 335 IF A=RETRN AND L=0 AND X>1 THEN 35
   0
JR 340 IF ((A=RETRN AND  NOT EDIT) OR A=B
   ACK5P) AND L=0 THEN 320
DW 350 IF A=RETRN THEN POKE 752,1:? " ":R
   ETURN
GG 360 IF A<>BACK5P THEN 400
SA 370 IF L>1 THEN N$=N$(1,L-1):GOTO 390
A5 380 N$=""
RE 390 ? CHR$(BACK5P);:L=L-1:GOTO 320
BB 400 L=L+1:IF L>L1 THEN A=RETRN:GOTO 35
   0
WK 410 N$(L)=CHR$(A):? CHR$(A);:GOTO 320
KN 420 GRAPHIC5 0:END
YT 430 GO5UB 440:POSITION 10,10:? "NO 5UC
   H FILE!":FOR X=1 TO 1000:NEXT X:CLO5E
   #2:GOTO 30
FD 440 POKE 710,48:50UND 0,100,12,8:FOR X
   =1 TO 50:NEXT X:50UND 0,0,0,0:RETURN
MY 450 GRAPHIC5 23:POKE 16,112:POKE 53774
   ,112:POKE 559,0:POKE 710,4
XR 460 DL=PEEK(560)+256*PEEK(561)+4:POKE
   DL-1,70:POKE DL+2,6
HW 470 FOR X=3 TO 39 5TEP 2:POKE DL+X,2:N
   EXT X:FOR X=4 TO 40 5TEP 2:POKE DL+X,0
   :NEXT X
ZW 480 POKE DL+41,65:POKE DL+42,PEEK(560)
   :POKE DL+43,PEEK(561):POKE 87,0
AC 490 POSITION 2,0:? "analog ml editor":
   POKE 559,34:RETURN
HZ 500 OPEN #1,4,0,"K:":GET #1,A:CLO5E #1
   :RETURN
```

Choplifter

Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94086
(408) 745-2000
8-bit cartridge, $24.95

Reviewed by Matthew J.W. Ratcliff

The original *Choplifter*, vintage 1982, was a disk-based game by Broderbund software. The cartridge version from Atari has been given a graphics face-lift, without sacrificing the playability of the original.

The game scenario is simple and challenging. "The Bungelings have kidnapped 64 of our delegates from the World Peace Conference." Your task is to rescue them. You pilot a helicopter from your secret base, which is disguised as a post office, into the Bungeling territory. The backdrop is a colorful scene of blue sky, clouds, mountains and the flat ground below. You have three "sorties," or lives, in which to complete your mission.

Your chopper is equipped with a machine gun for shooting open the barracks that contain the hostages, and for air-to-air combat (extremely difficult) with enemy jet fighters and drone air mines. You must overcome an onslaught of tanks, as well, which may be taken out with your bombs. Quickly pressing the fire button shoots, while holding it down briefly turns the helicopter to face right, forward and left. When facing forward, the bombs are activated by the fire button, and the machine gun is active when facing either side.

The most difficult component of the game to master is the timing used on the fire button. Fire too slowly, and you end up turning the helicopter instead of shooting. If you hold the button too long, the chopper will turn too far with generally fatal results if an enemy is in hot pursuit. A good tactile feedback joystick, such as the Epyx 500XJ is best suited for *Choplifter*.

No points are scored for shooting the enemies. Your mission is to save lives, not destroy them, though you must eliminate hostiles as necessary in your effort to rescue your allies. This is no easy task. As you fly along, tanks will track you from below. Drop bombs on them to give you enough time to land. Once on the ground, the hostages will board your aircraft.

The Bungelings are always on your trail. Stops to pick up hostages will be brief, and you must land close to them to save time. This requires precision flying, since it is easy to squash a friend. As the tanks relentlessly hunt you down, they may shoot some of the hostages. It is up to you to lift off and draw the tanks' fire to save your men.

Only 16 men at a time will fit in the chopper. They must be returned to the post office, where they will wave in gratitude as they get off the helicopter. Subsequent trips will be much tougher.

A perfect score in this game is 64 lives saved. Saving all 64 men is a difficult but attainable goal. I have not mastered it yet, but my local Atari dealer, Jeff Randall, has the original, and he tells me that the disk-based game presented a special screen graphic if you achieved a perfect score. I don't know if the same game-winning graphic is included in the cartridge version.

The first *Choplifter* was done in Atari's high-resolution, two-color graphics mode 8. All the colors in the game were achieved through "artifacting," where combinations of even and odd pixels gave different color effects. It is clear that a lot of work went to the development of the new graphics for this game.

Matthew Ratcliff, a frequent contributor to ANALOG Computing, *lives in St. Louis, Missouri with his wife and two children.*

DATABASE DELPHI

by Michael A. Banks

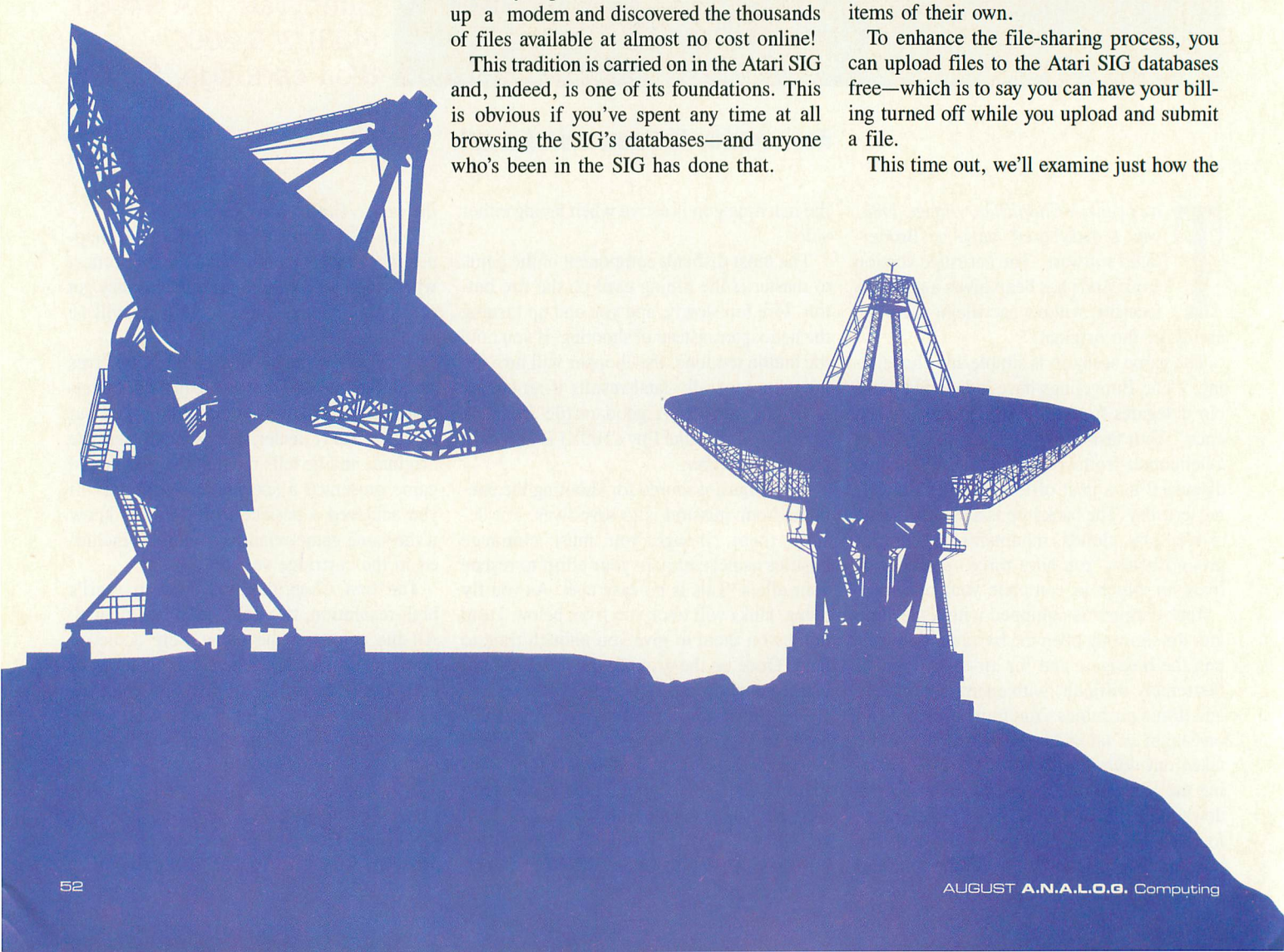
Sharing files and information is a time-honored tradition among modem users, and for many the major reason for being online. It was certainly an eye-opener for me the first time I fired up a modem and discovered the thousands of files available at almost no cost online!

This tradition is carried on in the Atari SIG and, indeed, is one of its foundations. This is obvious if you've spent any time at all browsing the SIG's databases—and anyone who's been in the SIG has done that.

But, have you ever wondered just where all those files come from? A good percentage of the files in the Atari SIG database come from people like you, who enjoy sharing files from their local BBSs and/or creations or news items of their own.

To enhance the file-sharing process, you can upload files to the Atari SIG databases free—which is to say you can have your billing turned off while you upload and submit a file.

This time out, we'll examine just how the



database file-submission process works. I will also bring you up to date on some new wrinkles in submitting files to Atari SIG databases that you may have missed. (I'll tackle the latter first.)

New Database Submission Procedure

Whether you're a frequent contributor to the Atari SIG or you've never submitted a file, you'll be pleased to know that recent changes in the database-submission procedure make contributing files much easier.

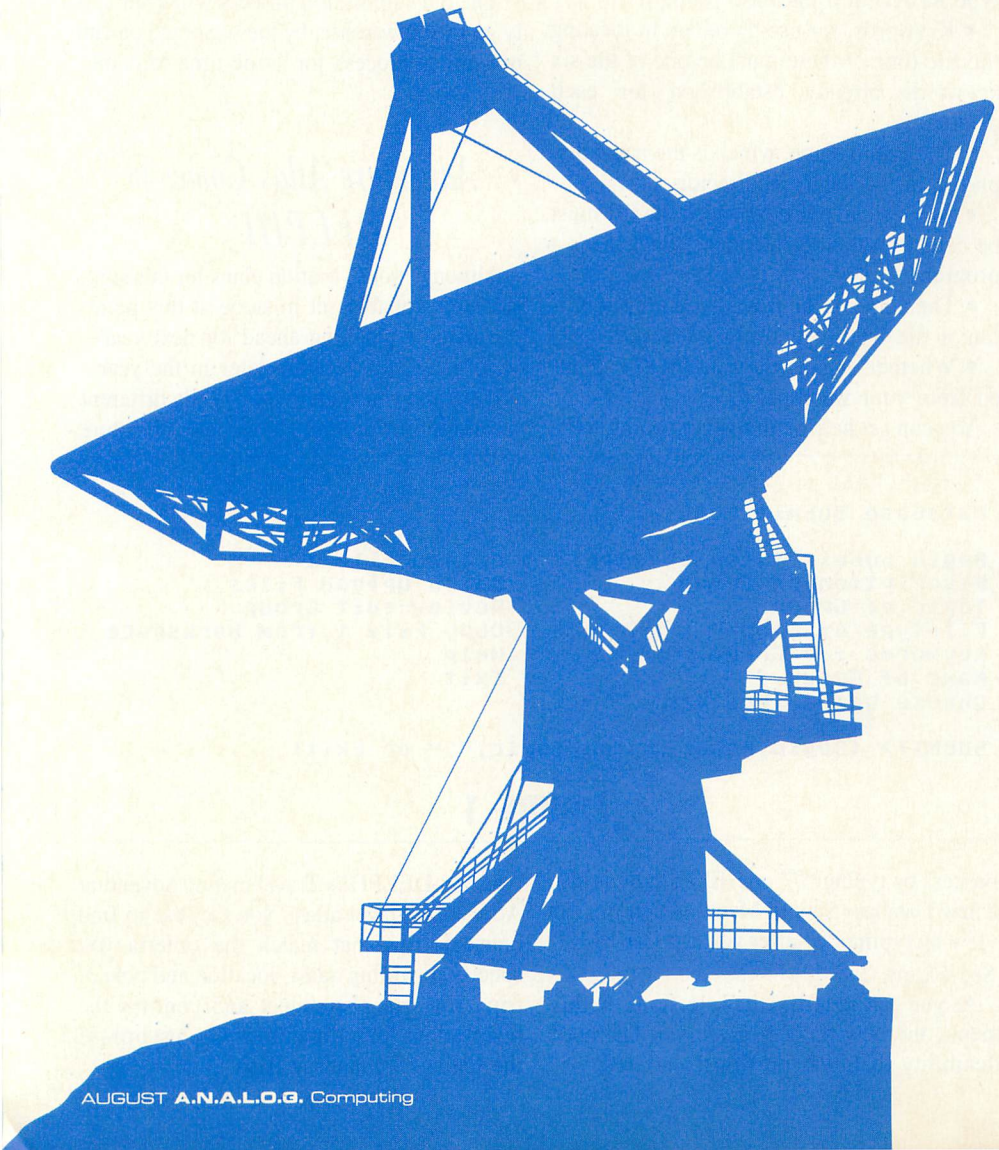
In the past, submissions were handled in such a manner that you had to first upload the file in question to your personal workspace, after which you could submit the file from either workspace or an Atari SIG database prompt.

As with the old procedure, typing SUBMIT at either the workspace or a database prompt initiates the process. Now, however, when you submit a file to an Atari SIG database, you are offered the option of uploading the file direct as a part of the submission process. (You can still submit a file that has been previously uploaded to your workspace, if you wish.)

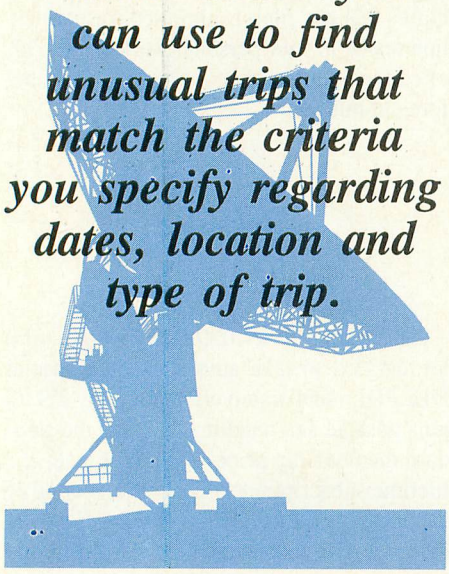
When you type SUBMIT at a database prompt, you will see the new Choose Submit Procedure Menu:

```
NEW Submit Procedure
OLD Submit Procedure
Help
Exit

CHOOSE-SUBMIT> (NEW, OLD, Help, Exit)
```

If you select OLD, you will go through the standard submission procedure, which requires that any file(s) submitted for publication reside in your personal workspace. This of course requires that you upload them before submitting.



Adventure Atlas is a "travel atlas" you can use to find unusual trips that match the criteria you specify regarding dates, location and type of trip.



If you select NEW, you will be prompted at the appropriate point in the submission process to upload the file. The new submission process still allows you to submit files from your workspace, by the way.

Selecting the new procedure gives you some additional options, including the ability to do batch-file uploads, change your upload protocol, and review and edit the information you've input per the steps outlined below.

Step-by-step

Outside of the origin of the submitted file and the options just discussed, the submission process is basically the same whether you select NEW or OLD after you type SUBMIT. You'll be taken through the database submission process by a series of prompts, at which you'll be asked to enter the following information:

- The number of files you are submitting (and, if you are submitting more than file, whether they are related and should be placed together as a single database entry)
- The type of file you are submitting (article, program, data file, etc.)
- The Atari SIG database (topic) in which the file should be placed; a description of the file (this is the description you see when you type READ at a database prompt)
- Keywords, for use by others in locating the file (one of these must be one of the six keywords already established for each database)
- The group name, which is the name displayed in the database directory
- Any special name by which the file must be called on the downloader's disk, if it's a program
- The name of the file (if you are submitting a file from your workspace)
- Whether or not you want the file deleted from your workspace

You can get help at almost any point in the process by typing "?" or HELP; there is also a new Database Submit Menu, which you can view by typing "?" after you type SUBMIT: See Figure 1

As you can see from the selections on this menu, the new process gives you far more flexibility than was previously offered.

```
Database Submit Menu:

Begin Submit (Step by step)        Upload File 1
Description of Group               Batch Upload Files
Topic of Group                     Review-edit Group
Filetype of Group                  Copy File 1 from Workspace
Keywords for Group                 Help
Name of Group                      Exit
Choose Upload Protocol

SUBMIT> (Begin,Description,Topic,"?" or Exit)
```

FIGURE 1

Notes

If you select the new submission process, your billing is automatically turned off during the submit. If you use the old process, you will want to select "Request free upload" from the Atari SIG main menu beforehand, to set an appointment with the SIG manager for your free upload and submission.

In every instance, the file you submit will not immediately appear in the database you specify. It will be stored temporarily in a special database where the SIG manager can examine it, and, if necessary, add or change the description or keywords.

The old submission process will eventually go by the wayside, by the way, but you can use either process for some time to come.

Adventure Atlas Comes to DELPHI

Although your vacation plans for this summer are probably set in stone at this point, you may be planning ahead for next year—or looking at a vacation later in the year.

If so, and you want to try a really different vacation, browse through the new Adventure Atlas on DELPHI's Travel menu. Adventure Atlas is a "travel atlas" you can use to find unusual trips that match the criteria you specify regarding dates, location and type of trip. You can also select and confirm the reservations for a trip online. Or, just browse the listings for fantasy trips.

The game sets you up with $1,000 in virtual funds in the bank. You can buy as many chips as you wish for use in betting.

Trip categories available include bicycle tours, cruises (down the Nile or up the Yangtse River), golfing, hiking, rafting and more. As the online description of Adventure Atlas implies, you can find an experience here—not just a vacation.

Type GO TRAVEL ADV at the Atari SIG Main Menu to see more of the Adventure Atlas.

Poker on DELPHI

Poker Showdown! is the latest addition to DELPHI's growing collection of online games. A real-time version of draw poker that accommodates single or multiple players, *Poker Showdown!* is an exciting pastime, especially when there are other players to compete against.

The game sets you up with $1,000 in virtual funds in the bank. You can buy as many chips as you wish for use in betting. Then, pick a table—novice or advanced. Your winnings (or losses) are remembered from one game to the next, so bet wisely or you won't be able to play the big stakes anymore. To reach *Poker Showdown!* from the Atari SIG, type GO ENT POKER.

If your poker skills are rusty, or you're unfamiliar with the name, type HELP to get a brief review of the relative values of the poker hands. You'll also find detailed information about the game of poker—and many other topics—in Grolier's Encyclopedia. Type GO LIB ENC, then ENCY, then type POKER as the search term.

Try DELPHI/Regional for Savings

ANALOG readers in the Kansas City area may wish to look into joining DELPHI/Kansas City. This regional version of DELPHI provides economical access at a flat rate of just $9.95 per month. Access to the national DELPHI service is surcharged at normal DELPHI rates, but the local service has a lot to offer—including access to E-mail to and from all DELPHI members. If you're a heavy E-mail user, this alone will mean a big savings.

DELPHI/Boston likewise offers lower rates than the national DELPHI service, as well as the E-mail interconnect. Both regional services provide a number of extra services specific to their cities.

For more info, you can check out DELPHI/Kansas City or DELPHI/Boston direct the next time you're on DELPHI by typing GO DELPHI at the SIG's main menu.

Low-cost PC Pursuit Access Instituted

DELPHI has implemented access via PC Pursuit. This new feature will be welcomed by PC Pursuit users, as connecting with DELPHI via PC Pursuit means billing at the direct-dial rate of $6.60 per hour, non-prime time. DELPHI Advantage Plan members are not affected by PC Pursuit access; they are charged the same low rate of $4.80 per hour whether or not they use PC Pursuit.

PC Pursuit users should use this ID at the Telenet prompt:

```
@C DELPHI,PCP#,PCPPASSWORD
```

The PC Pursuit sign-on is not available from Canada.

Frequent DELPHI users who don't yet use PC Pursuit might consider signing up for it, both to benefit from the lower DELPHI charge and the nationwide BBS access it provides. Call Telenet customer service at 1-800-336-0437 or 703-689-6400 for more information.

Send FAX Without a FAX Machine!

You can now send text FAX messages to anyone who has a FAX machine. DELPHI's FAX service offers all the convenience of E-mail, including being able to upload a message from disk and not having to wait while the receiving FAX machine is connected.

To access DELPHI's FAX service, type DELPHI at the SIG main menu; this will take you to the DELPHI Mail menu. Select FAX or type HELP FAX for complete information on preparing FAX messages, rates, etc.

The DELPHI Mail menu also offers gateways into EasyLink, a specialized E-mail service in the U.S. and England, and Telex, the messaging system that connects you with millions of Telex terminals worldwide.

That's it for now. See you in Conference! Tuesday evening, 10:00 P.M., Eastern time: Be there, or be an obtuse rectangle!

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

continued from page 45

```
   CA5CBC970D0B59B39303330444154 41A5CCC98
   ED0AF60000102039B3930343044415441
OI 660 DATA A909854DA902CDC002F0068DC0024
   C4F59A91C8DC002A9FFD0034C62E4CE025AD05
   EA9018D025ACEF659CEF759AD78024A4A
NN 670 DATA 49021869014AAAF006EEF659CAD0F
   8AD78022903490218690 14AAAF007EEF759CA4
   C8659ADF659C9CD9005A9CC8DF659C92D
AC 680 DATA 9B39303530444154 41B005A92D8DF
   659ADF759C9E49005A9E38DF759C924B005A92
   48DF759ACF859A209A90099005488CAD0
MA 690 DATA F9ACF7598CF859A209BDF85999005
   488CAD0F6ADF6598D00D0A200F010CAADF6598
   D205AADF7598D275A8EE0594C62E44C40
EO 700 DATA 4000101000C60010109B393036304
   4415441000168A9018DE059ADE059D0FB20555
   AADFC02C9FFF00785CDA9FF8DFC02A900
XZ 710 DATA 38E92D85CBA90038E92485CCA5CD8
   5CEF01CA0008C1FD0A900C901F01120555AA5C
   DD0F9A9048514A514C90590FAA9068D1D
PV 720 DATA D060AD8402490185CDA9028D1DD06
   06868859B39303730444154 41CD85596885CC8
   55868688DA75A68688DA35A68688DD15A
AI 730 DATA AD3002186DA35A85CEAD310285CF6
   868F014A249ADA75AC94ED004E88EA75AC94FD
   0038EA75AA200A000A90091CEE6CED002
AU 740 DATA E6CFA5CC91CEE6CED002E6CFA5CD9
   1CEA9281865CC85CC9002E6CD9B39303830444
   15441E6CED002E6CFE8E000D0D2606868
OR 750 DATA 85CC6885CB686885CF8DA45B68688
   D945B68688D805B68688D965BEE965B6885CE6
   885CD686885D08DA85B68688D9E5B6868
KK 760 DATA 8D565BAE945BF00EA5CB18692885C
   B9002E6CCCAD0F2AE9E5BF00EA5CD18692885C
   D90029B39303930444154 41E6CECAD0F2
UO 770 DATA AD565BC903D0034CCB5BA5CF4A4AA
   8A5CF2903AAE8B1CBCAF0050A0A4C495B29C08
   5D1C9FFF024A5D04A4AA8C028B02EA5D0
HH 780 DATA 2903AAE8BDEF5BCAF00746D146D14
   C6B5B49FF31CD05D191CDA5CFC900F00DE6D0E
   6CFAD565BC903D0AEF055EE9B39313030
DN 790 DATA 44415441945BA900C900D00160EE9
   E5BA900C9C0B0F6A90085CFA90085D0A928186
   5CB85CB9002E6CCA9281865CD85CD9002
ZK 800 DATA E6CEAD565BC903F01D4C3C5BA5CF4
   A4A85CF8DA45BAD805B4A4A8D805BA5D04A4A8
   5D08DA85BA4CFB1CBA4D091CD4C7D5BC0
PY 810 DATA 300C03689B39313130444154 41686
   885CD8D465C4A4A8D875C68688D9E5C68688DD
   55C4A4A8D915C68688DA05C6868186A6A
UP 820 DATA 6A8DAF5CA95085CBA96185CCAE9E5
   CF00EA5CB18692885CB9002E6CCCAD0F220AE5
   CAD9E5CCDA05CD00160A9002903AAE8BD
EN 830 DATA EF5B49FF85CFADAF5CCAF0054A9B3
   93132304441544 14A4C555C85D0ADD55C2903A
   AE8BDEF5B49FF8D935CADAF5CCAF0054A
CO 840 DATA 4A4C715C85D1A9281865CB85CB900
   2E6CCA000A5CF31CB05D091CBA000A90031CB0
   5D191CBEE9E5CA900C900F002D0D6AD46
ZN 850 DATA 5C85CD20AE5C60A90085CEA5CD4A4
   AA8A5CD2903AAE89B39313330444154 41BDEF5
   BCAF00746CE46CE4CC05C49FF31CB05CE
OD 860 DATA 91CBA5CDC900F005E6CD4CAE5C609
   B39323030444154415152525252525252525
   75252525252525252525752525252525252
HJ 870 DATA 52525257525252525252525252457C0
   02D6F6465000000007C0026696C6C000000007
   C00266C6970000000007C0028616C7665
PN 880 DATA 00007C415252525252525252535
   25252525252525252529B39323130444154 41535
   25252525252525252535252525252525252
ND 890 DATA 52447C00246973 6B000000007C002
   3656C00000000007C00216E696D617465007C0
   0246F75626C65007C5A52525252525252
OC 900 DATA 52525852525252525252525258525
   25252525252525258525252525252525252439B
```

LISTING 3 : ASSEMBLY

```
1000 ; PICTURE PERFECT Copy Routine
1010 ; Assembly Listing
1020 ; Written by Joe D. Brzuszek
1030 ;
1040 ; Call from BASIC with
1050 ; A=USR(23253,S1,X1,Y1,X2,Y2,S2,X
3,Y3,CEL)
1060 ;
1070 ; program equates
1080 X1      = $CF
1090 X3      = $D0
1100 GRABIT = $D1
1110 ;
1120          *=$5AD5 ;=23253 decimal
1130 ;
1140          PLA ;pull accumulator fr
om stack
1150          PLA
1160          STA $CC ;hi byte, copy f
rom
1170          PLA
1180          STA $CB ;lo byte, copy f
rom
1190          PLA
1200          PLA
1210          STA X1
1220          STA AX1+1 ;backup storag
e
1230          PLA
1240          PLA
1250          STA Y1+1
1260          PLA
1270          PLA
1280          STA X2+1
1290          PLA
1300          PLA
1310          STA Y2+1
1320          INC Y2+1
1330          PLA
1340          STA $CE ;hi byte, copy t
o
1350          PLA
1360          STA $CD ;lo byte, copy t
o
1370          PLA
1380          PLA
1390          STA X3
1400          STA AX3+1 ;backup storag
e
1410          PLA
1420          PLA
1430          STA Y3+1
1440          PLA
1450          PLA
1460          STA CEL+1 ;indicates tra
nsparent color
1470 ;
1480 ;find row to copy from
1490          LDX Y1+1
1500          BEQ NOPLUS ;if Y1=0, no
addition is needed
1510 DW1      LDA $CB
1520          CLC
1530          ADC #40 ;each mode 14 li
ne is 40 bytes wide
1540          STA $CB
1550          BCC NX1
```

continued on page 63

Dark Chambers

Atari Corp.
1196 Borregas Avenue
Sunnyvale, CA 94086
(408) 745-2000
8-bit cartridge, $39.95

Reviewed by Matthew J.W. Ratcliff

A cold chill shoots up your spine as you face the *Dark Chambers*. The cobblestone floor, moldy brick walls, flickering torches, and a pair of glowing eyes beckons to you, promising a magical mystery maze of excitement, danger and adventure.

This is a one- or two-player graphics adventure with twenty-six different levels, A through Z. Each screen is a maze, viewed from above, that presents a puzzle to be solved—with a few complications!

Each chamber is a highly detailed, smoothly scrolling room, littered with treasures and weapons to collect. But the ghouls of this expedition will try to thwart you every step of the way. The most difficult ghoul to destroy is the Grim Reaper. Each time he is blasted, he will transform into the next weakest type of ghoul—such as a wizard, then wraith and so on—until he reaches the weakest form and is obliterated.

Your soldier of fortune responds quickly to joystick commands, and by simply walking over objects will acquire them. Collecting a gun will increase shooting speed, and a dagger will make your shots more powerful.

A "lifeline", a horizontal bar at the bottom of the screen, is displayed for each player. It decreases each time you collide with a ghoul, poison or a booby trap. If this line disappears, your life and the game has ended. Two people can play simultaneously, which makes *Dark Chambers* a lot more fun. In the two-player, cooperative mode, a dead player may be resuscitated by shooting the beating heart that is hidden somewhere in each level. The two players must share the collection of food and weapons, to maximize progress through the game.

Where do ghouls come from? Their parents are called spawners; five different kinds create the various foes you will face. Each time a spawner is blasted, it too mutates into the next weakest form, until it is finally destroyed. The "underground" spawner, producing an infinite supply of ghouls, cannot be destroyed. Fortunately, up to 15 bombs may be collected and carried. Whenever the screen is choked with ghouls and spawners, a quick double press of the fire button eliminates everything in sight.

Keys must be collected to unlock doors which lead to more difficult levels in the game. You must find the hole with a ladder and climb down to explore the next maze.

Dark Chambers has three difficulty levels. On the beginner level, I was able to master the game in just a few days, and make it through all 26 screens. I was disappointed to see Screen A again, right after conquering Level Z. There is no ultimate goal in this game—no damsel in distress, no evil king to destroy, and no special award for completing the game. It simply continues with the first screen, so that you may explore the same mazes all over again.

At standard and advanced levels, *Dark Chambers* is challenging, with more and nastier ghouls to deal with. The mazes at each level seem to be the same from one game to the next, with the exits, spawners, keys and locked doors in the same locations. Only the placement of food and treasures seems to be random from game to game.

One of the nicest features about *Dark Chambers* is that you can always continue where you left off. Simply press Fire to pick up at the level you died on. You must start over collecting your shield, weapons and treasures, however.

Dark Chambers sports superb graphics, good sound effects, and playability that outshines many other games. It is much more responsive to user inputs than *Gauntlet*, a similar (but more sophisticated) game from Mindscape. It is much easier to master than *Into The Eagle's Nest* (another magnificent graphics adventure from Atari, which will take much longer to beat even though it has only four different mazes to conquer). Except for the minor disappointment of having nothing special happen when completing all 26 levels, *Dark Chambers* is a top-notch game that will provide plenty of challenge and excitement.

Matthew Ratcliff, a frequent contributor to ANALOG Computing, *lives in St. Louis, Missouri with his wife and two children.*

GUN ASSIST

by Matthew Ratcliff

The light gun that comes with the Atari XEGS is an exciting alternative to joysticks for video games. While new light-gun games are slow to come to market, you may be interested in writing your own applications for it. If you don't have a light gun, they should be available separately at many toy stores that carry the Atari XEGS. It is called the XES2001 (XES) and is packaged with Atari's original *Bug Hunt* game for $34.95. If you can't find one in your area, call Atari at 408-745-2000 and order it direct through their customer service.

The XES is a difficult creature to understand, but it can be thought of as a "long distance light pen." Whenever you wish to determine the current light-gun position, simply PEEK its horizontal and vertical positions into BASIC variables. These locations are referred to as LPENH (Memory Location 564) and LPENV (565) in Compute!'s *Mapping the Atari*. These are "shadow" locations for the hardware registers at 54284 (X position) and 54285 (Y position). I've found no difference in using one over the other, and generally use the hardware registers.

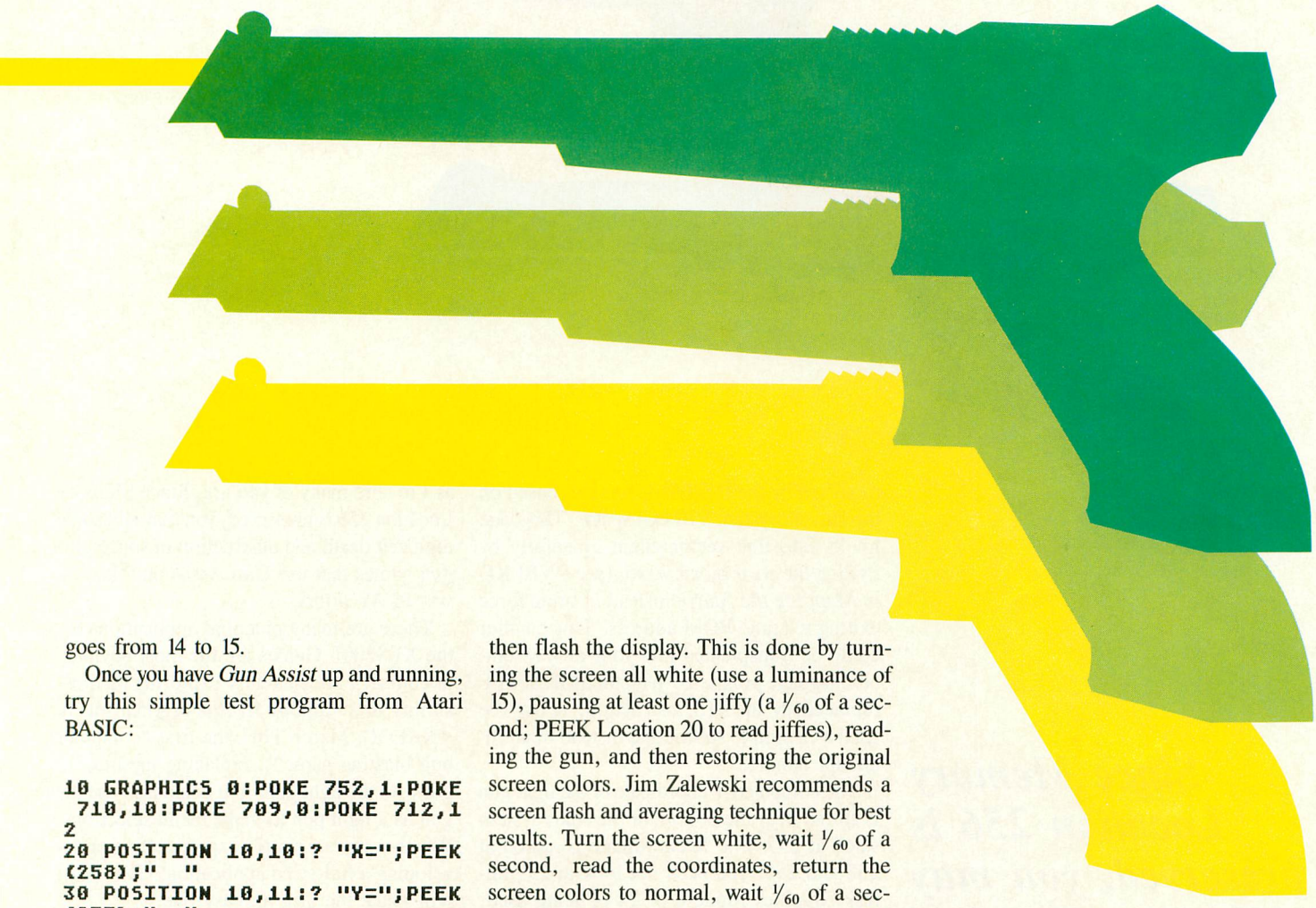
The problem with interpreting the light gun lies in the seemingly bizarre numeric values it will return. For the Y position (LPENV), readings will vary from a low of about 17 to a high of 112 inclusive, for a range of 96, in all graphics modes. The X position (LPENH) is extremely weird. It starts at a low of about 90, increases to 227, drops to 0 at about text Column 34, and then increases again to a high of about 30. These readings can be adjusted as follows:

```
1000 LPENH=54284:LPENV=54285
1010 X=PEEK(LPENH)
1020 Y=PEEK(LPENV)
1030 IF (X<40) THEN X=X+227:I
F (X>255) THEN X=255
1040 X=X-90:IF (X<0) THEN X=0
1050 Y=Y-17:IF (Y<0) THEN Y=0
```

This code will convert the X readings to a value from 0 to 159, and the Y readings will vary from 0 to 95. These ranges of 160 and 96 turn out to be exact multiples, by powers of 2 (ideal for assembly language programmers), of the horizontal and vertical resolutions of all 16 different graphics modes! For example, you would multiply the X and Y readings above by 2, to convert gun position to screen coordinates in Graphics Mode 24, the 320 x 192 mode (implying that you can "shoot" only every other pixel). In Mode 0, the X and Y readings should be divided by 4 to map directly onto the screen.

All of this dirty work can now be done for you by *Gun Assist*. Listing 1 is the BASIC data used to create your copy of *Gun Assist*. You should type this data using the M/L Editor in this issue. The resultant file, a machine-language program, may be executed from DOS. *Gun Assist* will install a small VBI routine in Page 4 of memory (the 128-byte cassette buffer). This routine is called 60 times a second by the operating system. It continuously monitors the current graphics mode at Memory Location 87 (DINDEX in *Mapping the Atari*). From this mode *Gun Assist* determines the proper multiplier or divisor to convert the adjusted gun readings to screen coordinates.

Gun Assist must be installed on your boot disk as an AUTORUN.SYS file. When run, *Gun Assist* checks to see if it has already been installed. If not, it will copy the VBI handler into the cassette buffer and enable it. If *Gun Assist* is already in memory, it will not load again (this would crash the system). You are warned of the conflict, and prompted to press a key to continue. Once *Gun Assist* has installed the new handler, it displays a title screen, with all the proper credits. If the light gun is detected in the first or second joystick port, you are informed. Detecting the location of the gun is simply a matter of reading all the joystick ports, and looking for the one that returns a 14 instead of the usual 15. When the trigger is pressed, the STICK reading of that port



goes from 14 to 15.

Once you have *Gun Assist* up and running, try this simple test program from Atari BASIC:

```
10 GRAPHICS 0:POKE 752,1:POKE
 710,10:POKE 709,0:POKE 712,1
2
20 POSITION 10,10:? "X=";PEEK
(258);"  "
30 POSITION 10,11:? "Y=";PEEK
(257);"  "
40 GOTO 20
```

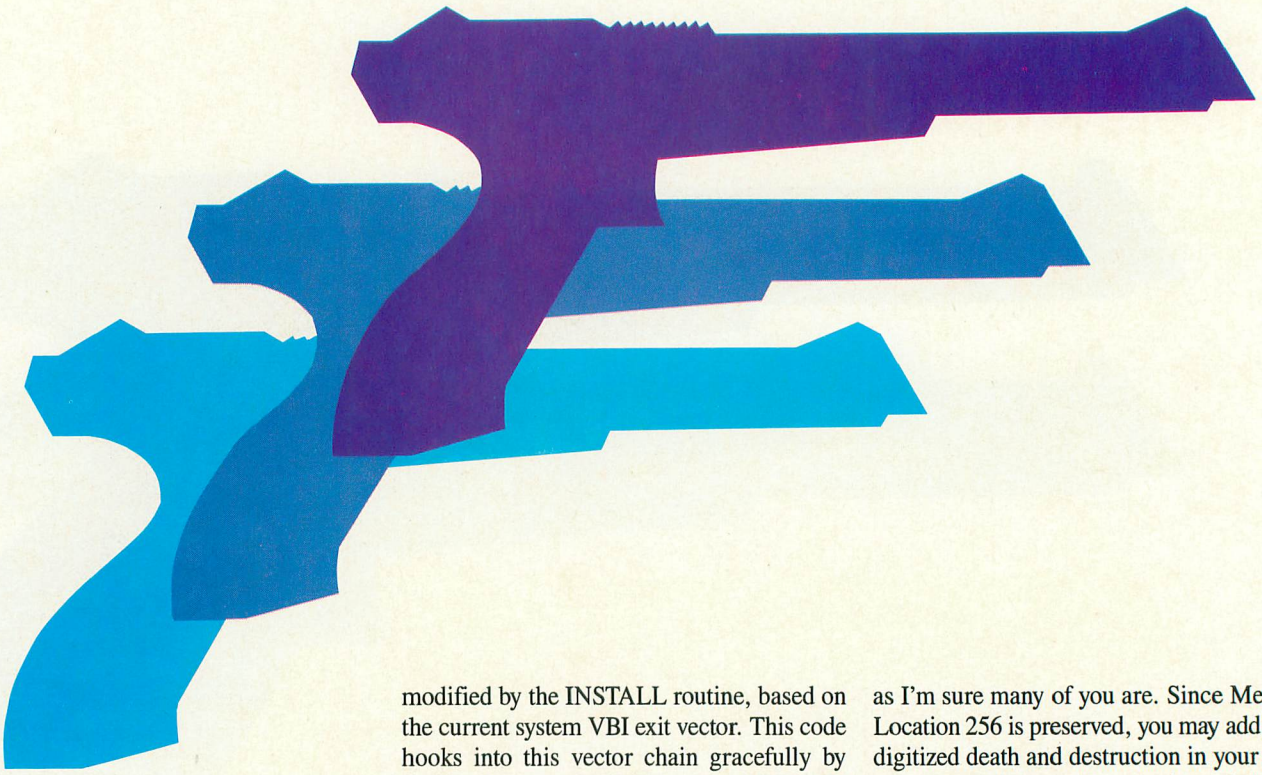
Notice that we are printing a small amount of black text on a white screen. Touch the screen with the light gun and move it to all four corners. Values should range from 0 to 39 in the X, and 0 to 23 in the Y directions. Point the light gun at the letter X on the screen; it should display 10,10 for the current position. Your screens must be bright to get accurate readings. The screen borders should also be of a light shade to improve gun accuracy near the edges of the display. Jim Zalewski, author of *Barnyard Blaster*, warns that interrupts, such as keyboard input or disk I/O, can adversely affect gun accuracy. Neither of these will be a problem most of the time.

If your screens must be dark, or you are simply not satisfied with the gun's accuracy, then flash the display. This is done by turning the screen all white (use a luminance of 15), pausing at least one jiffy (a $1/60$ of a second; PEEK Location 20 to read jiffies), reading the gun, and then restoring the original screen colors. Jim Zalewski recommends a screen flash and averaging technique for best results. Turn the screen white, wait $1/60$ of a second, read the coordinates, return the screen colors to normal, wait $1/60$ of a second, and take a second set of readings. Add the pair of readings and divide by 2. *Barnyard Blaster*, which is noticeably more accurate than *Bug Hunt*, employs this technique.

Gun Assist pays no attention to the light gun's trigger, screen colors, or much of anything else. It just takes the readings and performs the conversions for you. Not much else would fit in 128 bytes of the "safe" memory of the cassette buffer. (Actually, I could have squeezed in one more byte.) It is up to you to ensure light-gun accuracy with one of the techniques noted above.

Type in and run Listing 2, a short demo for *Gun Assist*. Point the gun at the screen and squeeze the trigger. When pressed, the current coordinates are displayed in the text window and a line is drawn to that position. Try holding the gun very steady while pressing the trigger and note the horizontal "jitter." The XES has much more "noise" in the horizontal direction than vertical. This is due to some hardware limitations in the way the POKEY chip scans the gun. In your programs which use the XES for input, it is wise to make the objects that you select, or "shoot," wide enough to compensate for this. Experimentation will be required to determine the optimum technique; *Gun Assist* makes it easy.

Listing 3 is the Mac/65 source code for *Gun Assist*. The file contains complete documentation on the technique employed. The code beginning at the label INSTALL checks to see if the VBI is already present, installs the handler and displays the title screen. The code from labels DUMMY to LT480 is the entire handler, 127 bytes of object code. The JMP $FFFF is code that is



modified by the INSTALL routine, based on the current system VBI exit vector. This code hooks into this vector chain gracefully by finding the correct exit vector (see VVBLKD in *Mapping the Atari*) instead of brute force exiting to some ROM address. This handler could be transplanted into your own assembly language programs. With the JMP $FFFF replaced by an RTS instruction, it may be used as a simple subroutine instead of an interrupt handler.

The computed Y coordinate of the light gun is stored near the bottom of the Atari system stack at 257. The X coordinate is stored at 258 and 259. You need not look at 259 unless Graphics Mode 8 or 24 is being used, where X can be greater than 256. Why didn't I use the very bottom of the stack, Memory Location 256? The Covox *Voice Master Junior* uses this location as a page pointer to its speech data. I am a big fan of the Covox, as I'm sure many of you are. Since Memory Location 256 is preserved, you may add some digitized death and destruction in your light-gun games that use *Gun Assist* and the Covox PLAY utility.

Since Memory Location 256 is preserved, you may add some digitized death and destruction in your light-gun games that use* Gun Assist *and the Covox PLAY utility.

There are many potential applications for the XES2001. Games are the most obvious, of course. I am currently exploring the possibility of a *Rambug III* (the original was in ANALOG, March 1987, the first "computer bug blasting game"), replacing joystick input with the light gun. You could also use the gun as a light pen to make menu selections. Whether you are simply experimenting or developing a full-sized application, *Gun Assist* will make the task an easy one.

Matthew Ratcliff, a frequent contributor to ANALOG, *lives in St. Louis, Missouri, with his wife and two children. Matt has had over 100 computer articles published.*

LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,0,52,175,52,2,3,3,1,
1,1,0,0,255,1,7099
1010 DATA 1,1,2,2,0,0,2,2,3,2,1,1,0,0,
255,255,9032
1020 DATA 255,255,2,3,255,255,166,87,1
73,12,212,201,40,176,6,105,7641
1030 DATA 227,144,2,169,255,201,90,176
,2,169,90,56,233,90,201,160,9980
1040 DATA 144,2,169,159,141,2,1,189,0,
4,240,22,48,9,168,78,2029
1050 DATA 2,1,136,208,250,240,11,160,0
,140,3,1,14,2,1,46,8747
```

```
1060 DATA 3,1,173,13,212,56,233,17,176
,2,169,0,141,1,1,189,3148
1070 DATA 16,4,48,11,240,12,168,78,1,1
,136,208,250,240,3,14,5244
1080 DATA 1,1,76,255,255,83,58,155,0,7
2,162,96,169,12,157,66,5180
1090 DATA 3,32,86,228,162,96,169,3,157
,66,3,169,127,157,68,3,3971
1100 DATA 169,52,157,69,3,104,157,75,3
,41,240,73,16,9,12,157,1437
1110 DATA 74,3,32,86,228,96,176,52,171
,53,173,36,2,201,32,208,6046
1120 DATA 59,173,37,2,201,4,208,52,76,
209,52,253,65,108,114,101,6610
1130 DATA 97,100,121,32,105,110,32,117
,115,101,155,162,0,169,9,157,4970
1140 DATA 66,3,169,193,157,68,3,169,52
,157,69,3,169,16,157,72,3818
1150 DATA 3,169,0,157,73,3,32,86,228,7
6,151,57,173,36,2,141,3610
1160 DATA 125,52,173,37,2,141,126,52,1
62,128,160,0,185,0,52,153,4341
1170 DATA 0,4,200,202,208,246,169,32,1
60,4,166,20,228,20,240,252,963
1180 DATA 141,36,2,140,37,2,169,0,32,1
31,52,76,54,53,17,18,8408
1190 DATA 18,18,18,18,18,18,18,18,18,1
8,18,18,18,18,5,155,5635
1200 DATA 162,0,169,9,157,66,3,169,36,
157,68,3,169,53,157,69,3535
1210 DATA 3,169,18,157,72,3,169,0,157,
73,3,32,86,228,76,105,3484
1220 DATA 53,124,32,32,160,199,245,238
,160,193,243,243,233,243,244,160,8968
1230 DATA 32,124,155,162,0,169,9,157,6
6,3,169,87,157,68,3,169,4225
1240 DATA 53,157,69,3,169,18,157,72,3,
169,0,157,73,3,32,86,902
1250 DATA 228,76,156,53,1,18,18,18,18,
18,18,18,18,18,18,18,4493
1260 DATA 18,18,18,18,4,155,162,0,169,
9,157,66,3,169,138,157,4641
1270 DATA 68,3,169,53,157,69,172,53,16
7,54,3,169,18,157,72,3,2554
1280 DATA 169,0,157,73,3,32,86,228,76,
207,53,124,32,65,78,89,3590
1290 DATA 32,65,116,97,114,105,32,77,1
11,100,101,124,155,162,0,169,5813
1300 DATA 9,157,66,3,169,189,157,68,3,
169,53,157,69,3,169,18,3401
1310 DATA 157,72,3,169,0,157,73,3,32,8
6,228,76,2,54,124,88,2391
1320 DATA 61,80,69,69,75,40,50,53,56,4
1,32,32,32,32,124,155,267
1330 DATA 162,0,169,9,157,66,3,169,240
,157,68,3,169,53,157,69,5501
1340 DATA 3,169,18,157,72,3,169,0,157,
73,3,32,86,228,76,53,2782
1350 DATA 54,124,32,43,50,53,54,42,80,
69,69,75,40,50,53,57,9198
1360 DATA 41,124,155,162,0,169,9,157,6
6,3,169,35,157,68,3,169,3740
1370 DATA 54,157,69,3,169,18,157,72,3,
169,0,157,73,3,32,86,1033
1380 DATA 228,76,104,54,124,32,89,61,8
0,69,69,75,40,50,53,55,175
1390 DATA 41,32,32,32,124,155,162,0,16
9,9,157,66,3,169,86,157,4740
1400 DATA 68,3,169,54,157,69,3,169,18,
157,72,3,169,0,157,73,3049
1410 DATA 3,32,86,228,76,155,54,1,18,1
8,18,18,18,18,18,18,6143
1420 DATA 18,18,18,18,18,18,18,4,155,1
62,0,169,9,157,66,3,352
1430 DATA 169,137,157,68,3,169,168,54,
163,55,54,157,69,3,169,18,3510
1440 DATA 157,72,3,169,0,157,73,3,32,8
6,228,76,206,54,124,32,4277
1450 DATA 85,115,101,32,194,210,201,19
9,200,212,32,32,32,32,124,155,7285
1460 DATA 162,0,169,9,157,66,3,169,188
,157,68,3,169,54,157,69,5177
1470 DATA 3,169,18,157,72,3,169,0,157,
73,3,32,86,228,76,1,2080
1480 DATA 55,124,32,115,99,114,101,101
,110,32,99,111,108,111,114,115,5272
1490 DATA 32,124,155,162,0,169,9,157,6
6,3,169,239,157,68,3,169,6309
1500 DATA 54,157,69,3,169,18,157,72,3,
169,0,157,73,3,32,86,1163
1510 DATA 228,76,52,55,124,32,102,111,
114,32,114,101,108,105,97,98,4389
1520 DATA 108,101,32,32,124,155,162,0,
169,9,157,66,3,169,34,157,4295
1530 DATA 68,3,169,55,157,69,3,169,18,
157,72,3,169,0,157,73,3183
1540 DATA 3,32,86,228,76,103,55,124,32
,114,101,97,100,105,110,103,4923
1550 DATA 115,46,32,32,32,32,32,124,15
5,162,0,169,9,157,66,3,1945
1560 DATA 169,85,157,68,3,169,55,157,6
9,3,169,18,157,72,3,169,3836
1570 DATA 0,157,73,3,32,86,228,76,154,
55,124,32,84,101,99,104,4334
1580 DATA 32,65,115,115,105,115,116,32
,98,121,124,155,162,0,169,9,4931
1590 DATA 157,66,3,169,136,157,164,55,
159,56,68,3,169,55,157,69,4975
1600 DATA 3,169,18,157,72,3,169,0,157,
73,3,32,86,228,76,205,5474
1610 DATA 55,124,32,74,105,109,32,90,9
7,108,101,119,115,107,105,32,4000
1620 DATA 32,124,155,162,0,169,9,157,6
6,3,169,187,157,68,3,169,5815
1630 DATA 55,157,69,3,169,18,157,72,3,
169,0,157,73,3,32,86,1294
1640 DATA 228,76,0,56,1,18,18,18,18,18
,18,18,18,18,18,18,4427
1650 DATA 18,18,18,18,4,155,162,0,169,
9,157,66,3,169,238,157,6531
1660 DATA 68,3,169,55,157,69,3,169,18,
157,72,3,169,0,157,73,3313
1670 DATA 3,32,86,228,76,51,56,124,40,
99,41,32,49,57,56,57,349
1680 DATA 32,65,110,97,108,111,103,124
,155,162,0,169,9,157,66,3,3875
1690 DATA 169,33,157,68,3,169,56,157,6
9,3,169,18,157,72,3,169,3869
1700 DATA 0,157,73,3,32,86,228,76,102,
```

```
56,124,66,121,32,77,97,3487
1710 DATA 116,42,82,97,116,32,32,32,32
,32,124,155,162,0,169,9,2413
1720 DATA 157,66,3,169,84,157,68,3,169
,56,157,69,3,169,18,157,4379
1730 DATA 72,3,169,0,157,73,3,32,86,22
8,76,153,56,124,82,97,4787
1740 DATA 116,119,97,114,101,32,32,32,
32,32,32,32,32,124,155,162,2431
1750 DATA 0,169,9,157,66,3,160,56,155,
57,169,135,157,68,3,169,5845
1760 DATA 56,157,69,3,169,18,157,72,3,
169,0,157,73,3,32,86,1425
1770 DATA 228,76,204,56,124,83,111,102
,116,119,111,114,107,115,44,32,4693
1780 DATA 49,57,56,57,124,155,162,0,16
9,9,157,66,3,169,186,157,6860
1790 DATA 68,3,169,56,157,69,3,169,18,
157,72,3,169,0,157,73,3447
1800 DATA 3,32,86,228,173,120,2,201,14
,208,54,76,6,57,124,71,3828
1810 DATA 117,110,32,105,110,32,112,11
1,114,116,32,48,32,32,124,155,3395
1820 DATA 162,0,169,9,157,66,3,169,244
,157,68,3,169,56,157,69,6069
1830 DATA 3,169,18,157,72,3,169,0,157,
73,3,32,86,228,76,151,4840
1840 DATA 57,173,121,2,201,14,208,54,7
6,67,57,124,71,117,110,32,3783
1850 DATA 105,110,32,112,111,114,116,3
2,49,32,32,124,155,162,0,169,4614
1860 DATA 9,157,66,3,169,49,157,68,3,1
69,57,157,69,3,169,18,3165
1870 DATA 157,72,3,169,0,157,73,3,32,8
6,228,76,151,57,76,121,4738
1880 DATA 57,124,71,117,110,32,238,239
,244,32,102,111,117,110,100,32,7229
1890 DATA 32,124,155,162,0,169,9,157,6
6,3,169,103,157,68,3,169,5077
1900 DATA 57,157,69,3,169,18,157,72,3,
169,0,157,73,3,32,86,1566
1910 DATA 228,76,172,57,124,82,156,57,
56,58,69,83,69,84,32,116,2942
1920 DATA 111,32,114,101,109,111,118,1
01,124,155,162,0,169,9,157,66,5868
1930 DATA 3,169,154,157,68,3,169,57,15
7,69,3,169,18,157,72,3,3082
1940 DATA 169,0,157,73,3,32,86,228,76,
223,57,26,18,18,18,18,402
1950 DATA 18,18,18,18,18,18,18,18,18,1
8,18,3,155,162,0,169,161
1960 DATA 9,157,66,3,169,205,157,68,3,
169,57,157,69,3,169,18,4201
1970 DATA 157,72,3,169,0,157,73,3,32,8
6,228,76,16,58,32,208,3829
1980 DATA 242,229,243,243,160,225,160,
235,229,249,174,174,174,155,162,0,4946
1990 DATA 169,9,157,66,3,169,0,157,68,
3,169,58,157,69,3,169,4150
2000 DATA 16,157,72,3,169,0,157,73,3,3
2,86,228,169,255,205,252,1989
2010 DATA 2,240,251,141,252,2,96,226,2
,227,2,176,52,0,0,0,2659
```

LISTING 2: BASIC

```
YL 10 REM SAVE"D:GUNTEST.BAS"
OQ 20 REM For use with Gun Assist VBI
AD 30 REM routine by Mat*Rat, for ANALOG
KR 40 REM Computing (c) 1989.  You
UY 50 REM must execute GUNVBIM.COM from
AM 60 REM DOS before running me.
DY 70 GRAPHICS 8:POKE 712,10:POKE 710,10:
   POKE 709,0:COLOR 1
SS 80 X=PEEK(258)+256*PEEK(259):Y=PEEK(25
   7)
RK 90 IF (X<320) AND (Y<160) THEN PLOT X,
   Y:GOTO 110
RU 100 GOTO 80
JH 110 POKE 752,1
UI 120 IF STICK(1)=14 THEN 120
PW 130 IF PEEK(764)<>255 THEN POKE 764,25
   5:GRAPHICS 8:POKE 712,10:POKE 710,10:P
   OKE 709,0:COLOR 1:POKE 752,1
OV 140 FOR I=1 TO 10:NEXT I
GD 150 X=PEEK(258)+256*PEEK(259):Y=PEEK(2
   57)
LA 160 ? "X=";X;" Y=";Y;"  "
QS 170 IF (X<320) AND (Y<160) THEN DRAWTO
    X,Y
MS 180 GOTO 120
```

LISTING 3: ASSEMBLY

```
0 *SAVE#D:GUNVBIM.M65
10 *ASM,,#D:GUNVBIM.COM
20        .OPT NO LIST
30 * Version 1.00, (c) 1989, Analog
40 * Computing
50 * By Matthew J. W. Ratcliff
60 * Ratware Softworks
70 * Install a VBI
80 * handler for the
90 * Atari light gun.
0100 *  Each VBI
0110 * LPENV and LPENH
```

continued on page 66

continued from page 56

```
1560            INC $CC
1570 NX1        DEX
1580            BNE DW1
1590 ;
1600 ;find row to copy to
1610 NOPLUS     LDX Y3+1 ;if Y3=0, no ad
dition is needed
1620            BEQ NOPLUZ
1630 DW2        LDA $CD
1640            CLC
1650            ADC #40 ;each mode 14 li
ne is 40 bytes wide
1660            STA $CD
1670            BCC NX2
1680            INC $CE
1690 NX2        DEX
1700            BNE DW2
1710 NOPLUZ     LDA CEL+1
1720            CMP #3
1730            BNE NEWPLOT
1740            JMP FSTCOPY ;fast copy
1750 ;
1760 ;begin copying the image pixel
1770 ;by pixel...
1780 ;find column to copy from
1790 NEWPLOT    LDA X1
1800            LSR A ;divide by 4
1810            LSR A
1820            TAY ;Y = horizontal offs
et
1830            LDA X1
1840            AND #03
1850            TAX
1860            INX
1870 ;
1880 ;locate pixel to copy
1890            LDA ($CB),Y
1900 SHIFT      DEX
1910            BEQ ANDIT
1920            ASL A
1930            ASL A
1940            JMP SHIFT
1950 ANDIT      AND #192
1960            STA GRABIT ;store bit pa
ttern of pixel
1970 ;
1980 ;make pixel color transparent?
1990 CEL        CMP #255
2000            BEQ NOPLOT
2010 ;
2020 ;find horizontal column to copy t
o
2030            LDA X3
2040            LSR A ;divide by 4
2050            LSR A
2060            TAY ;Y = horizontal offs
et
2070            CPY #40 ;screen is only
40 bytes wide!
2080            BCS ENDLINE ;off screen?
2090            LDA X3 ;no.
2100            AND #03
2110            TAX
2120            INX
2130 ;
2140 ;adjust copy data
2150            LDA HBITS-1,X
2160 RIGHT      DEX
2170            BEQ STOREIT
2180            LSR GRABIT
2190            LSR GRABIT
2200            JMP RIGHT
2210 STOREIT    EOR #255
2220            AND ($CD),Y
2230            ORA GRABIT
2240            STA ($CD),Y ;pixel is st
ored
2250 NOPLOT     LDA X1
2260 X2         CMP #0
2270            BEQ ENDLINE
2280            INC X3
2290            INC X1
2300            LDA CEL+1
2310            CMP #3
2320            BNE NEWPLOT
2330            BEQ ZIPCOPY
2340 ENDLINE    INC Y1+1 ;end of line re
ached
2350 Y1         LDA #0
2360 Y2         CMP #0
2370            BNE BELOW
2380 EXIT       RTS ;return to BASIC
2390 BELOW      INC Y3+1
2400 Y3         LDA #0
2410            CMP #192 ;off bottom of
screen?
2420            BCS EXIT
2430 AX1        LDA #0 ;No, carriage ret
urn.
2440            STA X1
2450 AX3        LDA #0
2460            STA X3
2470            LDA #40 ;40 bytes wide
2480            CLC ;next line to copy f
rom
2490            ADC $CB
2500            STA $CB
2510            BCC FROM
2520            INC $CC
2530 FROM       LDA #40 ;mode 14 line is
 40 bytes wide
2540            CLC ;next line to copy t
o
2550            ADC $CD
2560            STA $CD
2570            BCC JUMP
2580            INC $CE
2590 JUMP       LDA CEL+1
2600            CMP #3
2610            BEQ ZIPCOPY
2620            JMP NEWPLOT
2630 FSTCOPY    LDA X1 ;fast copy routin
e
2640            LSR A ;divide all X valu
es by 4
2650            LSR A
2660            STA X1
2670            STA AX1+1
2680            LDA X2+1
2690            LSR A
2700            LSR A
2710            STA X2+1
2720            LDA X3
2730            LSR A
2740            LSR A
2750            STA X3
2760            STA AX3+1
2770 ZIPCOPY    LDY X1
2780            LDA ($CB),Y ; load 4 pix
els
2790            LDY X3
2800            STA ($CD),Y ; store 4 pi
xels
2810            JMP NOPLOT ;increment an
d check X and Y values
2820 HBITS .BYTE 192,48,12,3
```

continued from page 34

```
3780      LDA RENT,X  ;get rent
3790      CMP #2      ;don't fall
3800      BCC B1      ;below 1
3810      LSR A       ;div 2
3820      STA RENT,X  ;save it
3830      LDA # <HR   ;'1/2 value
3840      LDY # >HR   ;'on improv..'
3850 ;
3860 SB   PHA         ;save hi/lo
3870      STY L       ;of msg
3880      JSR PB      ;'bad luck'
3890      JSR PUTCR   ;linefeed
3900      PLA         ;get msg
3910      LDY L
3920      JSR EPRINT  ;print it
3930      LDA PPROP   ;picked prop
3940      ASL A
3950      TAX         ;get name
3960      LDA PRPTAB+1,X
3970      TAY
3980      LDA PRPTAB,X
3990      JMP EPRINT  ;print name
4000 ;
4010 ;Halve the Value of a Property
4020 ;
4030 B3   JSR GRPROP  ;grab a prop #
4040      BMI B1      ;none
4050      TAX
4060      LDA COST,X  ;get price
4070      CMP #2      ;not LT 1
4080      BCC B1
4090      LSR A       ;div by 2
4100      STA COST,X  ;save it
4110      LDA # <HV   ;'1/2 value'
4120      LDY # >HV
4130      BNE SB      ;print it
4140 ;
4150 ;Luck address tables
4160 ;alternate for randomness
4170 ;
4180 LUCKL .BYTE  <B1, <G1, <B2
4190      .BYTE  <G2, <B3, <G3
4200 LUCKH .BYTE  >B1, >G1, >B2
4210      .BYTE  >G2, >B3, >G3
4220 ;
4230 ;Luck Text
4240 ;
4250 GOOD .BYTE "Good.",EOL,0
4260 BAD .BYTE "Bad.",EOL,0
4270 YL  .BYTE EOL,EOL,"Your luck "
4280      .BYTE "was - ",0
4290 YW  .BYTE "You win ",0
4300 YD  .BYTE "You lose ",0
4310 HR  .BYTE "1/2 Rent on",EOL,0
4320 HV  .BYTE "1/2 Value on ",EOL,0
4330 AN_IMP .BYTE "an improvement "
4340      .BYTE "on",EOL,0
4350 ;
4360 ;Get a Random Property
4370 ;WHO =owners, GRTAB cur player
4380 ;
4390 GRPROP JSR HAVE_ANY ;any to get?
4400      BPL GR2     ;yes indeed
4410      RTS         ;nawh.
4420 ;
4430 GR2 LDY #0       ;indx to WHO
4440      LDX #0      ;indx to GRTAB
4450 GR3 LDA WHO,Y    ;who owns it
4460      CMP PNUM    ;ME?
4470      BNE GR4     ;no
4480      TYA         ;yes,save #
4490      STA GRTAB,X
4500      INX         ;next one
4510 GR4 INY
4520      CPY #36     ;at end?
4530      BNE GR3     ;no
4540      INX         ;now pick a
4550      TXA         ;random one
4560      JSR GET_RND ;from those
4570      TAX         ;we've found
4580      LDA GRTAB-1,X
4590      STA PPROP   ;picked prop
4600      RTS
4610 ;
4620 ;Print a Property in Inverse
4630 ;
4640 INVPROP STA L    ;prop name
4650      STY L+1     ;address
4660      LDA #2
4670      JSR EPUT
4680      LDY #0
4690 INV1 STY SVY
4700      LDA (L),Y   ;get byte
4710      BEQ INV2    ;done on 0
4720      ORA #$80    ;inverse
4730      JSR EPUT    ;show it
4740      LDY SVY
4750      INY         ;next byte
4760      BNE INV1
4770 INV2 LDA #22     ;end
4780      JMP EPUT    ;exit
4790 ;
4800 ;Get Key Table (A - Z)
4810 ;
4820 KEY_TAB .BYTE 63,21,18,58,42
4830      .BYTE 56,61,57,13,1
4840      .BYTE 5,0,37,35,8
4850      .BYTE 10,47,40,62,45
4860      .BYTE 11,16,46,22,43,23
4870 ;
4880 ;Get a Key for Input
4890 ;
4900 GET_KEY LDX #$FF ;clear CH
4910      STX CH
4920 GETK1 LDA CH     ;raw code
4930      CMP #$FF
4940      BEQ GETK1   ;none pressed
4950      AND #63     ;no INV or CTRL
4960      CMP #12     ;RETURN?
4970      BNE TRY7F   ;no
4980      LDA #0      ;0=RET
4990      RTS
5000 TRY7F CMP #52    ;delete?
5010      BNE SCANT   ;no
5020      LDA #$7E    ;DELETE
5030      RTS
5040 SCANT LDY #25    ;scan table
5050 GETK2 CMP KEY_TAB,Y ;is it?
5060      BEQ GETK3   ;YES!
5070      DEY
5080      BPL GETK2   ;try again
5090      BMI GET_KEY ;no match
5100 GETK3 TYA        ;index
5110      CLC         ;add 'A'
5120      ADC #'A
5130      RTS         ;out
5140 ;
5150 ;Input Owner Names
5160 ;
5170 INPUT STA L      ;where in NAME
5180      STY L+1     ;BUF to store
5190      LDY #0      ;names
5200 INP1 STY SVY     ;counter
5210      LDA #127    ;our BLOCK
5220      STA (L),Y   ;cursor
5230      JSR GET_KEY ;get key
5240      LDX #0      ;poke speaker
5250      STX CONSOL  ;for click
5260      LDY SVY     ;get Y
5270      CMP #0      ;is A =0(RET?)
5280      BEQ INDONE  ;yes
5290      CMP #$7E    ;DEL-BK SPC?
5300      BNE INEXT   ;no
5310      CPY #0      ;is Y=0
5320      BEQ INP1    ;yes,no delete
5330      LDA #0      ;zap char
5340      STA (L),Y
5350      DEY         ;back up 1
5360      STA (L),Y
5370      JMP INP1    ;get input
5380 ;
5390 INEXT CPY #8     ;end of input?
5400      BCS INP1    ;yes
5410      STA IBUF,Y  ;input buffer
5420      JSR ASC2IC  ;to INT code
5430      ORA #$80    ;inverse for
5440      STA (L),Y   ;color
5450      INY         ;next char
5460      BNE INP1
5470 ;
5480 INDONE STA IBUF,Y ;0 for an
5490      STA (L),Y   ;EOL &
5500      RTS         ;exit
5510 ;
5520 ;Get Player Names
5530 ;
5540 GETNAMES LDA # <NDLIST ;install
5550      STA SDLSTL  ;display list
5560      LDA # >NDLIST
5570      STA SDLSTL+1
5580      LDA #58     ;normal plyfld
5590      STA SDMCTL
5600      LDY #4      ;screen
5610 GNM1 LDA NCLR,Y ;colors
5620      STA COLOR0,Y
5630      DEY
5640      BPL GNM1
5650      JSR CLRSCR  ;clear screen
5660 ;
5670      LDA # <GR0MEM ;get start
5680      STA SCR     ;of memory
5690      LDA # >GR0MEM
5700      STA SCR+1
5710      LDX #0      ;plyr #
5720 GNM0 LDA #251    ;'[
5730      STA (SCR),Y
5740      INY         ;scrn position
5750      TXA
5760      CLC
5770      ADC #209    ;plyr # in
5780      STA (SCR),Y ;color
5790      INY         ;next position
5800      LDA #253    ;']'
5810      STA (SCR),Y
5820      TYA         ;add 18 onto
5830      CLC         ;screen position
5840      ADC #18
5850      TAY         ;now in Y
5860      INX         ;next plyr
5870      CPX MAXP    ;at max yet?
5880      BNE GNM0    ;no do more
5890      LDY #0      ;now get names
5900 GNM2 STY SVY2    ;save Y
5910      LDA SCR     ;move over 4
5920      CLC         ;bytes
5930      ADC #4      ;A= low byte
5940      LDY SCR+1   ;Y = hi
5950      JSR INPUT   ;get a name
5960      LDA IBUF    ;get 1st char
5970      BEQ GNM2    ;if none try
5980      LDA SCR     ;again
5990      CLC         ;add 20 bytes
6000      ADC #20     ;to scrn pntr
6010      STA SCR
6020      LDA # >NAMEBUF ;high byte
6030      STA L+1     ;low =0
6040      LDY SVY2    ;keep Y
6050      LDA X9,Y    ;get place in
6060      STA L       ;buffer
6070      LDY #8      ;move name to
6080 GNM3 LDA IBUF,Y ;NAMEBUF
6090      STA (L),Y
6100      DEY
6110      BPL GNM3    ;9 chars total
6120      LDY SVY2    ;retrieve Y
6130      INY         ;next plyr
6140      CPY MAXP    ;at Max?
6150      BNE GNM2
6160      RTS         ;YES
6170 ;
6180 ;Handle Captial Gains Tax
6190 ;
6200 CAPGNS JSR PB    ;'bad LUCK'
```

```
6210      PRINT  YD   ;'You lose $45G'
6220     LDA #45
6230     JMP B1A      ;jmp into BAD
```

LISTING 7: ASSEMBLY

```
0100 ;SAVE#D:CAPITAL6.M65
0110 ;
0120 ;-----------------;
0130 ;     CAPITAL!     ;
0140 ;    (c)  1988     ;
0150 ;   MAP of U.S.A.  ;
0160 ;  By: Barry Kolbe ;
0170 ;                  ;
0180 ;-----------------;
0190 ;
0200 ;Map Screen Data
0210 ;
0220 ;H = Hog Hilton C=Capital Gains
0230 ;T = Tax db =double property
0240 ;L = LUCK Dicepos = DICE
0250 ;
0260 ;
0270 SCRN .BYTE $28,$6F,$70,$40
0280      .BYTE $42,$43,$45,$47
0290      .BYTE $28,$28,$28,$28
0300      .BYTE $28,$28,$28,$28
0310      .BYTE $28,$28,$28,$28
0320      .BYTE $28,$28,$28,$28
0330      .BYTE $28,$28,$28,$28
0340      .BYTE $28,$28,$28,$28
0350      .BYTE $28,$28,$28,$28
0360      .BYTE $28,$6E,$49,$28
0370      .BYTE $28,$6C,$40,$3F
0380      .BYTE $40,$40,$3F,$40
0390      .BYTE $40,$40,$42,$42
0400      .BYTE $42,$43,$43,$44
0410      .BYTE $44,$44,$42,$40
0420      .BYTE $43,$45,$28,$28
0430      .BYTE $28,$28,$28,$28
0440      .BYTE $28,$28,$28,$28
0450      .BYTE $28,$28,$28,$28
0460      .BYTE $28,$77,$40,$5E
0470      .BYTE $28,$6D,$40,$40
0480      .BYTE $40,$40,$40,$3F
0490      .BYTE $40,$40,$40,$40
0500      .BYTE $40,$40,$40,$40
0510      .BYTE $40,$40,$40,$40
0520      .BYTE $40,$4C,$28,$62
0530      .BYTE $71,$44,$44,$72
0540      .BYTE $28,$28,$28,$28
0550      .BYTE $28,$28,$7B,$43
0560      .BYTE $41,$40,$73,$65
0570      .BYTE $28,$40,$40,$3F
0580      .BYTE $40,$40,$40,$3F
0590      .BYTE $3F,$40,$40,$40
0600      .BYTE $40,$40,$40,$40
0610      .BYTE $40,$40,$40,$40
0620      .BYTE $40,$40,$40,$40
0630      .BYTE $40,$55,$7D,$6D
0640      .BYTE $64,$28,$28,$28
0650      .BYTE $28,$79,$40,$40
0660      .BYTE $3F,$78,$28,$28
0670      .BYTE $4B,$40,$3F,$40
0680      .BYTE $33,$34,$39,$3A
0690      .BYTE $2B,$2C,$2B,$2C ;db
0700      .BYTE $39,$3A,$3B,$3C ;L
0710      .BYTE $39,$3A,$39,$3A
0720      .BYTE $39,$3A,$35,$36 ;H
0730      .BYTE $40,$6B,$61,$40
0740      .BYTE $40,$60,$28,$28
0750      .BYTE $79,$40,$40,$3F
0760      .BYTE $4C,$74,$28,$28
0770      .BYTE $57,$40,$3F,$40
0780      .BYTE $40,$40,$40,$3F
0790      .BYTE $3F,$3F,$40,$40
0800      .BYTE $40,$40,$40,$40
0810      .BYTE $40,$40,$40,$40
0820      .BYTE $40,$40,$40,$40
0830      .BYTE $40,$60,$61,$40
0840      .BYTE $40,$67,$7B,$7A
0850      .BYTE $40,$40,$3F,$4C
0860      .BYTE $28,$28,$28,$28
0870      .BYTE $40,$3F,$40,$40
0880      .BYTE $2B,$2C,$40,$3F ;db
0890      .BYTE $3F,$40,$40,$40
0900      .BYTE $40,$40,$40,$40
0910      .BYTE $40,$40,$40,$40
0920      .BYTE $40,$40,$2B,$2C ;db
0930      .BYTE $40,$5E,$5F,$40
0940      .BYTE $40,$40,$40,$40
0950      .BYTE $40,$40,$40,$64
0960      .BYTE $28,$28,$28,$28
0970      .BYTE $40,$3F,$40,$40
0980      .BYTE $40,$40,$3F,$40
0990      .BYTE $3F,$3F,$40,$40
1000      .BYTE $40,$40,$40,$40
1010      .BYTE $40,$40,$40,$40
1020      .BYTE $40,$40,$40,$40
1030      .BYTE $40,$40,$40,$40
1040      .BYTE $40,$40,$40,$40
1050      .BYTE $3F,$40,$40,$64
1060      .BYTE $28,$28,$28,$28
1070      .BYTE $40,$40,$3F,$40
1080      .BYTE $2B,$2C,$40,$3F ;db
1090      .BYTE $3F,$3F,$40,$40
1100      .BYTE $3E,$31,$31,$31 ;H
1110      .BYTE $3D,$3E,$32,$31
1120      .BYTE $32,$3D,$2B,$2C ;db
1130      .BYTE $39,$3A,$2B,$2C ;db
1140      .BYTE $2B,$2C,$2F,$30 ; C
1150      .BYTE $40,$3F,$40,$63
1160      .BYTE $28,$28,$28,$28
1170      .BYTE $40,$40,$3F,$40
1180      .BYTE $40,$40,$40,$40
1190      .BYTE $3F,$3F,$40,$40
1200 DICPOS .BYTE $3E,$31,$32,$31
1210      .BYTE $3D,$3E,$32,$31
1220      .BYTE $32,$3D,$40,$40
1230      .BYTE $40,$40,$40,$40
1240      .BYTE $40,$40,$40,$40
1250      .BYTE $3F,$40,$40,$40
1260      .BYTE $68,$28,$28,$28
1270      .BYTE $53,$40,$3F,$40
1280      .BYTE $39,$3A,$40,$40
1290      .BYTE $40,$3F,$3F,$40
1300      .BYTE $3E,$31,$31,$31 ;H
1310      .BYTE $3D,$3E,$32,$31
1320      .BYTE $32,$3D,$40,$40
1330      .BYTE $40,$40,$40,$40
1340      .BYTE $40,$3F,$39,$3A
1350      .BYTE $40,$40,$40,$40
1360      .BYTE $69,$28,$28,$28
1370      .BYTE $4F,$40,$40,$3F
1380      .BYTE $40,$40,$40,$40
1390      .BYTE $40,$40,$3F,$40
1400      .BYTE $40,$40,$40,$40
1410      .BYTE $40,$40,$40,$40
1420      .BYTE $40,$40,$40,$40
1430      .BYTE $40,$40,$40,$40
1440      .BYTE $40,$3F,$40,$40
1450      .BYTE $40,$40,$40,$65
1460      .BYTE $28,$28,$28,$28
1470      .BYTE $7E,$4E,$40,$40
1480      .BYTE $39,$3A,$39,$3A
1490      .BYTE $35,$36,$40,$3F
1500      .BYTE $40,$40,$40,$40
1510      .BYTE $40,$40,$40,$40
1520      .BYTE $40,$40,$40,$40
1530      .BYTE $40,$40,$37,$38
1540      .BYTE $2B,$2C,$2B,$2C ;db
1550      .BYTE $40,$40,$65,$28
1560      .BYTE $28,$28,$28,$28
1570      .BYTE $28,$7F,$4E,$40
1580      .BYTE $40,$40,$40,$40
1590      .BYTE $40,$40,$3F,$3F
1600      .BYTE $40,$40,$40,$40
1610      .BYTE $40,$40,$40,$40
1620      .BYTE $40,$40,$40,$40
1630      .BYTE $40,$40,$40,$40
1640      .BYTE $3F,$40,$40,$40
1650      .BYTE $40,$66,$28,$28
1660      .BYTE $28,$28,$28,$28
1670      .BYTE $28,$28,$7F,$7D
1680      .BYTE $7D,$4E,$40,$40
1690      .BYTE $2B,$2C,$2B,$2C ;db
1700      .BYTE $39,$3A,$39,$3A
1710      .BYTE $3B,$3C,$2B,$2C ;L
1720      .BYTE $2B,$2C,$39,$3A ;db
1730      .BYTE $39,$3A,$39,$3A
1740      .BYTE $40,$40,$40,$40
1750      .BYTE $40,$60,$28,$28
1760      .BYTE $28,$28,$28,$28
1770      .BYTE $28,$28,$28,$28
1780      .BYTE $28,$7E,$7D,$4E
1790      .BYTE $40,$40,$40,$40
1800      .BYTE $3F,$40,$40,$40
1810      .BYTE $40,$40,$40,$40
1820      .BYTE $40,$40,$40,$40
1830      .BYTE $40,$40,$51,$51
1840      .BYTE $52,$52,$52,$4E
1850      .BYTE $40,$48,$28,$28
1860      .BYTE $28,$28,$28,$28
1870      .BYTE $28,$28,$28,$28
1880      .BYTE $28,$28,$28,$7F
1890      .BYTE $7D,$7D,$7D,$7D
1900      .BYTE $4F,$40,$40,$40
1910      .BYTE $40,$40,$40,$40
1920      .BYTE $40,$40,$40,$40
1930      .BYTE $40,$78,$28,$28
1940      .BYTE $28,$28,$28,$28
1950      .BYTE $4F,$40,$48,$28
1960      .BYTE $28,$28,$28,$28
1970      .BYTE $28,$28,$28,$28
1980      .BYTE $28,$28,$28,$28
1990      .BYTE $28,$28,$28,$28
2000      .BYTE $7E,$4F,$4C,$4E
2010      .BYTE $40,$40,$40,$40
2020      .BYTE $4C,$7D,$7D,$7D
2030      .BYTE $7D,$7D,$28,$28
2040      .BYTE $28,$28,$28,$28
2050      .BYTE $7E,$4E,$40,$68
2060      .BYTE $28,$28,$28,$28
2070      .BYTE $28,$28,$28,$28
2080      .BYTE $28,$28,$28,$28
2090      .BYTE $28,$28,$28,$28
2100      .BYTE $28,$7E,$28,$7E
2110      .BYTE $4E,$40,$40,$4D
2120      .BYTE $28,$28,$28,$28
2130      .BYTE $28,$28,$28,$28
2140      .BYTE $28,$28,$28,$28
2150      .BYTE $28,$7E,$76,$75
2160      .BYTE $28,$28,$28,$28
2170      .BYTE $28,$28,$28,$28
2180      .BYTE $28,$28,$28,$28
2190      .BYTE $28,$28,$28,$28
2200      .BYTE $28,$28,$28,$28
2210      .BYTE $7F,$4E,$40,$7C
2220      .BYTE $28,$28,$28,$28
2230      .BYTE $28,$28,$28,$28
2240      .BYTE $28,$28,$28,$28
2250      .BYTE $28,$28,$7F,$54
2260      .BYTE $28,$28,$28,$28
2270      .BYTE $28,$28,$28,$28
2280      .BYTE $28,$28,$28,$28
2290      .BYTE $28,$28,$28,$28
2300      .BYTE $28,$28,$28,$28
2310      .BYTE $28,$7F,$4E,$28
2320      .BYTE $28,$28,$28,$28
2330      .BYTE $28,$28,$28,$28
2340      .BYTE $28,$28,$28,$28
2350      .BYTE $28,$28,$28,$28
2360      .BYTE $28,$28,$28,$28
```

continued from page 62

```
0120 * are read and
0130 * converted to
0140 * SCREEN COORDINATE
0150 * equivalents, based
0160 * on the current
0170 * graphics mode -
0180 * found in DINDEX.
0190 *
0200 * Algorithm:
0210 * X range for
0220 * gun is 0-159
0230 * Y range for
0240 * gun is 0-95
0250 * These values
0260 * are close
0270 * approximations,
0280 * and the
0290 * unusual readings
0300 * from the light
0310 * gun do require
0320 * adjustments.
0330 * However the 160
0340 * range for X
0350 * and 96 range
0360 * for Y work out
0370 * to be PERFECT
0380 * multiples for
0390 * ALL graphics
0400 * modes as follows:
0410 *
0420 * GRAPHICS  X-wd Y-ht
0430 *   MODE    - Convert
0440 *             factor
0450 * -  ------- --------
0460 * 0   40-1/4  24-1/4
0470 * 1   20-1/8  24-1/4
0480 * 2   20-1/8  12-1/8
0490 * 3   80-1/2  24-1/4
0500 * 4   80-1/2  48-1/2
0510 * 5   80-1/2  48-1/2
0520 * 6  160-1/1  96-1/1
0530 * 7  160-1/1  96-1/1
0540 * 8  320-2/1 192-2/1
0550 * 9   80-1/2 192-2/1
0560 * 10  80-1/2 192-2/1
0570 * 11  80-1/2 192-2/1
0580 * 12  40-1/4  24-1/4
0590 * 13  40-1/4  12-1/8
0600 * 14 160-1/1 192-2/1
0610 * 15 160-1/1 192-2/1
0620 *
0630 *
0640 * Note that EVERY
0650 * conversion is
0660 * a multiple of 2!
0670 *  What a stroke
0680 * of luck (or was
0690 * it pure genius
0700 * on the part of
0710 * Atari 800 designer
0720 * Jay Miner?).
0730 * The current GR.
0740 * mode can be used
0750 * as an index
0760 * into a shift
0770 * table. If the
0780 * entry is zero,
0790 * then no shift.
0800 * If positive, then
0810 * shift right
0820 * to divide.
0830 * If negative then
0840 * shift left to
0850 * multiply.
0860 *
0870 LPENH = $D40C
0880 LPENV = $D40D
0890 VVBLKD = 548
0900 DOSLOAD = $3400
0910      .OPT OBJ
0920      .ORG DOSLOAD
0930      .INCLUDE #D1:SYSEQU.M65
0940      .INCLUDE #D1:IOMAC.LIB
0950 *
0960 * The VBI code
0970 * for gun reading
0980 *
0990 DOSOFFSET = $3000
1000 CASBUF = $0400
1010 BOTSTACK = 257
1020 YGUN =  BOTSTACK
1030 XGUN =  BOTSTACK+1
1040 XSHIFT = CASBUF
1050 YSHIFT = XSHIFT+16
1060 *
1070 * Actual YGUN,
1080 * XGUN, XSHIFT and
1090 * YSHIFT data that
1100 * will be copied
1110 * to page 4 - if
1120 * the VBI isn't
1130 * already there.
1140 *
1150 DUMMY
1160      .BYTE 2,3,3,1
1170      .BYTE 1,1,0,0
1180      .BYTE -1,1,1,1
1190      .BYTE 2,2,0,0
1200      .BYTE 2,2,3,2
1210      .BYTE 1,1,0,0
1220      .BYTE -1,-1,-1,-1
1230      .BYTE 2,3,-1,-1
1240 *
1250 * Start of VBI
1260 * handler for
1270 * light gun
1280 *
1290 START
1300      LDX DINDEX
1310      LDA LPENH
1320      CMP #40
1330      BCS HCK2
1340      ADC #227
1350      BCC HCK2
1360      LDA #255
1370 HCK2 CMP #90
1380      BCS HCK3
1390      LDA #90
1400 HCK3 SEC
1410      SBC #90
1420      CMP #160
1430      BCC HCK4
1440      LDA #159
1450 HCK4 STA XGUN
1460      LDA XSHIFT,X
1470 *    No shift reqd
1480      BEQ SETYGUN
1490      BMI SHLEFT
1500      TAY
1510 SHRIGHT LSR XGUN
1520      DEY
1530      BNE SHRIGHT
1540      BEQ SETYGUN
1550 SHLEFT LDY #0
1560 * In case of mode 8
1570 * where x can be > 256
1580      STY XGUN+1
1590      ASL XGUN
1600      ROL XGUN+1
1610 SETYGUN LDA LPENV
1620      SEC
1630      SBC #17
1640      BCS SAVY
1650      LDA #0
1660 SAVY STA YGUN
1670      LDA YSHIFT,X
1680      BMI SHYLEFT
1690      BEQ EXITNOW
1700      TAY
1710 SHYRIGHT LSR YGUN
1720      DEY
1730      BNE SHYRIGHT
1740      BEQ EXITNOW
1750 SHYLEFT ASL YGUN
1760      .OPT LIST
1770 * If address at
1780 * label LT480 must
1790 * be at $47F or below!
1800 *
1810 * Modified code
1820 EXITNOW JMP $FFFF
1830 LT480
1840      .OPT NO LIST
1850      .INCLUDE #D1:GRAPHICS.M65
1860 *
1870 * VBI install code
1880 *
1890 VBISTART = START-DOSOFFSET
1900 INSTALL LDA VVBLKD
1910      CMP # <VBISTART
1920      BNE INSTALL1
1930      LDA VVBLKD+1
1940      CMP # >VBISTART
1950      BNE INSTALL1
1960 ALHERE
1970       PRINT  0,"Already in use"
1980      JMP GUNEXIT
1990 INSTALL1 LDA VVBLKD
2000      STA EXITNOW+1
2010      LDA VVBLKD+1
2020      STA EXITNOW+2
2030      LDX #LT480-DUMMY+1
2040      LDY #0
2050 MOVBI LDA DUMMY,Y
2060      STA CASBUF,Y
2070      INY
2080      DEX
2090      BNE MOVBI
2100      LDA # <VBISTART
2110      LDY # >VBISTART
2120      LDX JIFFY
2130 INST1 CPX JIFFY
2140      BEQ INST1
2150      STA VVBLKD
2160      STY VVBLKD+1
2170       GRAPHICS  0
2180       PRINT  0,"┌──────────────┐"
2190       PRINT  0,"│  Gun Assist  │"
2200       PRINT  0,"├──────────────┤"
2210       PRINT  0,"│ ANY Atari Mode│"
2220       PRINT  0,"│X=PEEK(258)   │"
2230       PRINT  0,"│ +256*PEEK(259)│"
2240       PRINT  0,"│ Y=PEEK(257)  │"
2250       PRINT  0,"├──────────────┤"
2260       PRINT  0,"│ Use BRIGHT   │"
2270       PRINT  0,"│ screen colors │"
2280       PRINT  0,"│ for reliable  │"
2290       PRINT  0,"│ readings.     │"
2300       PRINT  0,"│ Tech Assist by│"
2310       PRINT  0,"│ Jim Zalewski  │"
2320       PRINT  0,"├──────────────┤"
2330       PRINT  0,"│(c) 1989 Analog│"
2340       PRINT  0,"│By Mat*Rat     │"
2350       PRINT  0,"│Ratware        │"
2360       PRINT  0,"│Softworks, 1989│"
2370      LDA STICK0
2380      CMP #14
2390      BNE CKST1
2400       PRINT  0,"│Gun in port 0  │"
2410      JMP GUNEXIT
2420 CKST1 LDA STICK0+1
2430      CMP #14
2440      BNE CKST2
2450       PRINT  0,"│Gun in port 1  │"
2460      JMP GUNEXIT
2470 CKST2
2480       PRINT  0,"│Gun not found  │"
```

GUN ASSIST

```
2490 GUNEXIT
2500      PRINT  0,"|RESET to remove|"
2510      PRINT  0,"└──────────────┘"
2520      PRINT  0," Press a key..."
2530     LDA #255
2540 VHOLDER CMP 764
2550     BEQ VHOLDER
2560     STA 764
2570     RTS
2580 *
2590 * Code startup header
2600 *
2610     *=  $02E0
2620     .WORD INSTALL
2630 *
2640 * End of program
2650 *
2660     .END
```

LISTING 4: ASSEMBLY

```
0 *SAVE#D:GRAPHICS.M65
10 *
15 * Mat*Rat's FAVORITE Macro!
16 *
20 * GRAPHICS g
30 *
40 * ENTRY: A-REG CONTAINS
45 *        GRAPHICS MODE 'G'
50 *  EXIT: Y-REG HAS
55 *        COMPLETION STATUS
60 *
70 SNAME .BYTE "S:",155,0
80 GRAFIX
90       PHA          ;SAVE 'G'
0100     LDX #6*$10   ;FILE 6
0110     LDA #CCLOSE
0120     STA ICCOM,X
0121 * First we must close IOCB #6
0130     JSR CIO
0140 * Ignore any errors from close
0150 *
0160     LDX #6*$10
0161 * We will open this IOCB
0170     LDA #COPN
0180     STA ICCOM,X
0190     LDA # <SNAME
0191 * We use the filename "S:"
0200     STA ICBADR,X
0210     LDA # >SNAME
0211 * By pointing to it
0220     STA ICBADR+1,X
0230 *
0240 * All is setup for open, now
0250 * we tell CIO (& S:) what
0260 * kind of open.
0261 *
0262 * Retrieve our saved 'G'
0263 * graphics mode
0270     PLA
0271 * Is given to 'S:'
0280     STA ICAUX2,X
0281 * (Note that S: ignores upper
0282 * bits of AUX2
0290 *
0291 * Now we get the upper bits
0300     AND #$F0
0301 * And flip bit 4
0310     EOR #$10
0311 * (S: expects it to be inverted
0330 * from the normal BASIC usage)
0331 *
0332 * Allow read and write access
0333 * for CIO and S:
0340     ORA #$0C
0350     STA ICAUX1,X
0360     JSR CIO      ;OPEN S:
0365 * Graphics mode 'G' now open
0370     RTS
0380     .MACRO GRAPHICS
0390       .IF %0<>1
0400       .ERROR "GRAPHICS LEN"
0410       .ENDIF
0420       .IF %1>255
0430       LDA %1
0440       .ELSE
0450       LDA #%1
0460       .ENDIF
0470     JSR GRAFIX
0480     .ENDM
```

LISTING 5: ASSEMBLY

```
0 *SAVE#D:SYSEQU.M65
10 * IOCB and other important
20 * system equates, from Compute's
30 * mapping the Atari and the Mac/65
40 * reference manual from ICD/OSS
50 * ---------------------------------
60 * I/O CONTROL BLOCK EQUATES
70 *
80 IOCB
90 *
0100 * Device handler, set by OS
0110 ICHID = $0340
0120 * Device number, set by OS
0130 ICDNO = $0341
0140 * I/O Command
0150 ICCOM = $0342
0160 * I/O Status, same as error code
0170 ICSTA = $0343
0180 * Buffer address, 2 bytes
0190 ICBADR = $0344
0200 * Disk handler put routine,
0210 * address - 1, goes here
0220 ICPUT = $0346
0230 * Buffer length, 2 bytes
0240 ICBLEN = $0348
0250 * Auxillary control bytes
0260 ICAUX1 = $034A  ;AUX 1
0270 ICAUX2 = $034B  ;AUX 2
0280 ICAUX3 = $034C  ;AUX 3
0290 ICAUX4 = $034D  ;AUX 4
0300 ICAUX5 = $034E  ;AUX 5
0310 ICAUX6 = $034F  ;AUX 6
0320 * Length of one IOCB
0330 IOCBLEN = 16
0340 *
0350 * IOCB COMMAND VALUE EQUATES
0360 *
0370 * Open channel
0380 COPN =  3
0390 * Get binary record
0400 CGBINR = 7
0410 * Get text record
0420 CGTXTR = 5
0430 * Put binary record
0440 CPBINR = 11
0450 * Put text record
0460 CPTXTR = 9
0470 * Open IOCB
0480 CCLOSE = 12
0490 * Check status
0500 CSTAT = 13
0510 *
0520 * DEVICE DEPENDENT COMMAND
0530 * EQUATES FOR FILE MANAGER
0540 *
0550 CREN =  32      ;RENAME
0560 CERA =  33      ;ERASE
0570 CPRO =  35      ;PROTECT
0580 CUNP =  36      ;UNPROTECT
0590 CPOINT = 37     ;POINT
0600 CNOTE = 38      ;NOTE
0610 *
0620 * AUX1 VALUES REQD FOR OPEN
0630 *
0640 OPIN =  4       ;OPEN INPUT
0650 OPOUT = 8       ;OPEN OUTPUT
0660 OPUPD = 12      ;OPEN UPDATE
0670 OPAPND = 9      ;OPEN APPEND
0680 OPDIR = 6       ;OPEN DIRECTORY
0690 *
0700 *
0710 *
0720 * MISC ADDRESS EQUATES
0730 *
0740 CPALOC = $0A
0750 * Warm start, 0=cold
0760 WARMST = $08
0770 * Store here,wait for horiz sync
0780 WSYNC = 54282
0790 * Available memory, low
0800 MEMLO = $02E7
0810 * Available memory, high
0820 MEMTOP = $02E5
0830 * Upper limit of application RAM
0840 APPMHI = $0E
0850 * Atari load/init addr
0860 INITADR = $02E2
0870 * Atari Load/Go addr
0880 GOADR = $02E0
0890 * Cartridge RUN location
0900 CARTLOC = $BFFA
0910 * CIO Entry address
0920 CIO =   $E456
0930 * End of line character
0940 EOL =   $9B
0950 *
0960 * Very useful Atari reserved
0970 * memory and hardware register
0980 * locations.
0990 *
1000 RANDOM = 53770
1010 SAVMSC = $58
1020 CONSOL = 53279
1030 COLOR0 = 708
1040 CRSINH = 752
1050 BOOT? = 9
1060 DINDEX = 87
1070 HATABS = 794
1080 CH  =   764
1090 STRIG0 = 644
1100 STICK0 = 632
1110 POKMSK = $10
1120 RAMTOP = $6A
1130 SDMCTL = $022F
1140 STACK = $0100
1150 *
1160 IRQEN = $D20E
1170 NMIEN = $D40E
1180 SKCTL = $D20F
1190 DMACTL = $D400
1200 KBCODE = $D209
1210 POTGO = $D20B
1220 AUDCTL = $D208
1230 AUDF3 = $D204
1240 AUDC3 = $D205
1250 PACTL = $D302
1260 PORTA = $D300
1270 PORTB = $D301
1280 *
1290 * Timer counter:
1300 JIFFY = 20       ; 60Hz clock
1310 * Cassette buffer at $400
1320 * use as general filename buff
1330 FILENAME = $0400
1340 * Display list pointer
1350 DLIST = $0230
```

INSIDE THIS ISSUE:

MORE

BOOT CAMP

PLUS

END USER

DATABASE DELPHI