THE #1 MAGAZINE FOR ATARI COMPUTER OWNERS

A.N.A.L.O.G. COMPUTING

T.M.

SEPTEMBER 1988
ISSUE 64

U.S.A. $3.50
CANADA $4.75

P
LAD

Entertainment Issue!

Snowplow

A COMMERCIAL QUALITY GAME

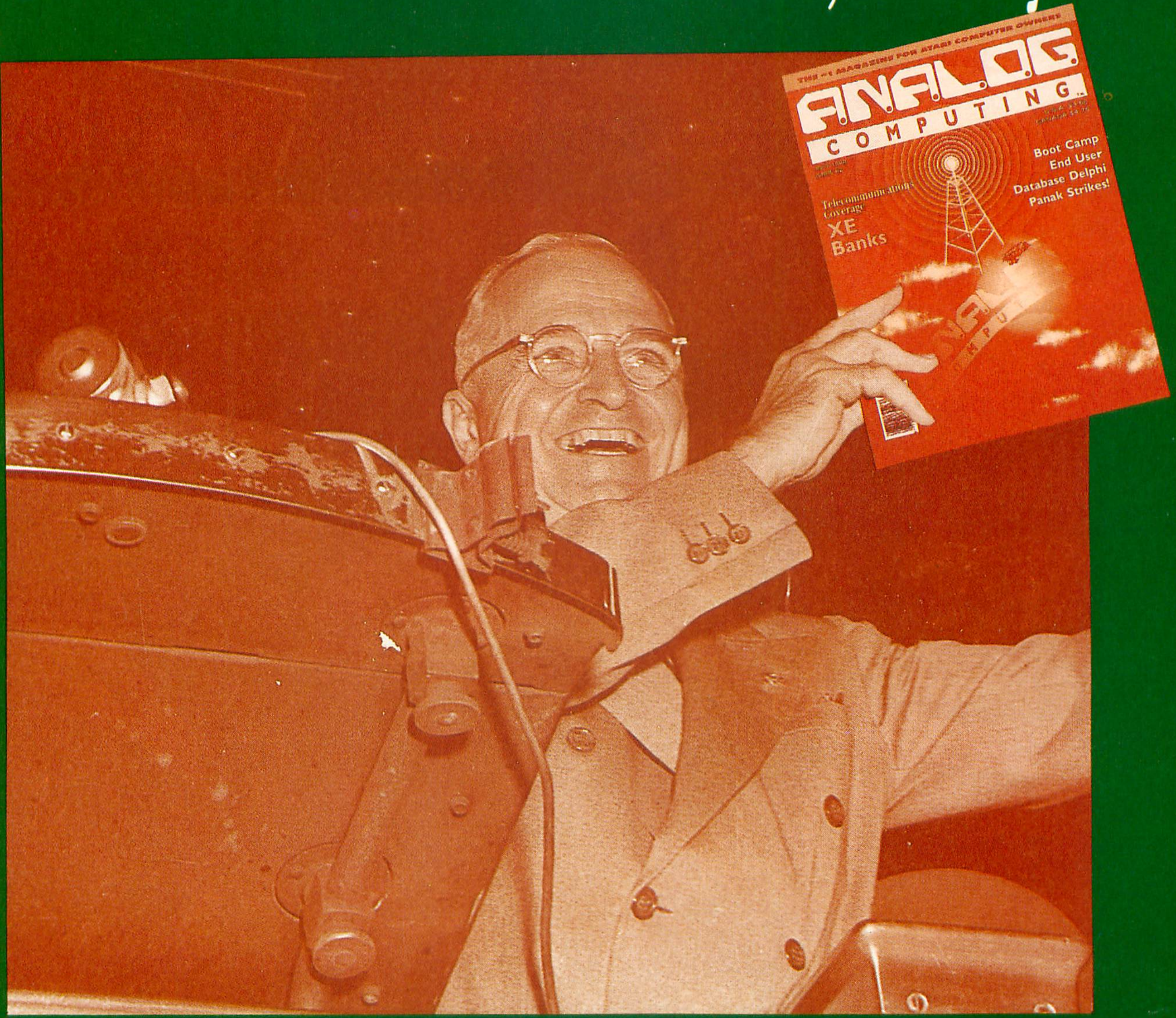
Kason's Tower

ALSO:

Master Memory Map

Game Design Workshop

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 $14 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 $14 OFF THE COVER PRICE

1 YEAR *WITH DISK* ONLY $79

NEW LOW PRICE!

Editorial

by Lee Pappas

It's true. Times are changing. Having just returned from my 14th Consumer Electronics Show (CES), I have nothing to report on the 8-bit news front. Of course, Atari was there pushing 8-bit products. . . that is just the XE Game System and software. But the days of dozens of new product announcements and releases are long gone—most likely for good.

CES was really lacking the multitude of software companies that have attended in the past, and many that did attend had their own rooms hidden away or went in with distributors. The big names in software now read Nintendo or Nintendo compatible. Even Apple, Mac and PC supporters were missing.

However, I must admit that what was missing for 8-bit was counterbalanced with quite a few new programs for the ST. And some neat stuff, too, which you can read about in next month's ST-Log.

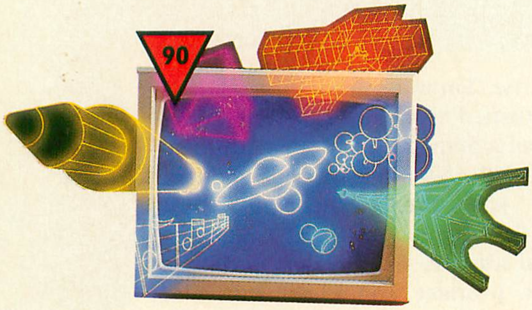
What's saddest is finding Atari far, far behind Nintendo in the video-game industry. Figure 70% to 80% belongs to Nintendo, with the rest split between Sega and Atari. And from what I saw at the show, it's Sega that gets my vote on gaining ground on Nintendo, not Atari.

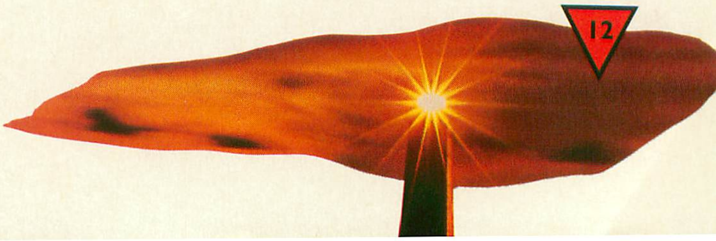
The problem with the XE Game System isn't the unit itself, but the software. Most of the games are starving for state-of-the-art graphics and just don't have the imagination that is clearly evident in the Nintendo and newer Sega products. Face it, the Nintendo and Sega don't have keyboards. In the Nintendo's case the unit is plain and boring in appearance, and the controls are simple. What those have, however, are spectacular, well-thought-out programs, many of which go far beyond the shoot-'em concept.

So that's my 8-bit report on the summer Consumer Electronics Show. Oh, for the days of pages and pages of exciting new 8-bit products to announce.

CONTE

FEATURES





T S

REVIEWS

COLUMNS

Front Cover Photography
Ladi Von Jansky
Front Cover Model
Laurie Ferguson

STAFF

Publisher
LEE H. PAPPAS

Executive Editor
CLAYTON WALNUM

Art Director
EDDY G. HERCH

Managing Editor
DEAN BRIERLY

East Coast Editor
ARTHUR LEYENBERGER

West Coast Editor
CHARLES F. JOHNSON

Contributing Editors
LEE S. BRILLIANT, M.D.,
MICHAEL BANKS, FRANK COHEN,
ANDY EDDY, MAURICE MOLYNEAUX,
STEVE PANAK, CRAIG PATCHETT,
MATTHEW J.W. RATCLIFF,
ROBIN SHERER, KARL E. WIEGERS

Entertainment Editors
ANDY EDDY, DAVID PLOTKIN

Illustrations
BRENT WATTS

Copy Chief
KATRINA VEIT

Copy Editors
ANNE DENBOK
SARA BELLUM PAT ROMERO

Typographers
KLARISSA CURTIS,
JUDY VILLANUEVA,
DAVID BUCHANAN

Contributors
JAMES J. GRECO
BARRY KOLBE
JOHN PILGE
JIM ROGERS
BRIAN SCHAPPEL

Production Director
DONNA HAHNER

National Advertising Director
JE PUBLISHERS REPRESENTATIVE
(213) 467-2266
(For regional numbers, see map)

Advertising Production Director
JANICE ROSENBLUM

Advertising Manager
PAULA THORNTON

Subscriptions Director
IRENE GRADSTEIN

Vice President, Sales
JAMES GUSTAFSON

Where to Write

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

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

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

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

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

Advertising Sales

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**

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

Permissions

No portion of this magazine may be reproduced in any form without written permission from the publisher. Many programs are copyrighted and not public domain.

Due, however, to many requests from Atari club libraries and bulletin-board systems, our new policy allows club libraries or individually run BBSs to make certain programs from **ANALOG Computing** available during the month printed on that issue's cover. For example, software from the July issue can be made available July 1.

This does not apply to programs which specifically state that they are not public domain and, thus, are not for public distribution.

In addition, any programs used must state that they are taken from **ANALOG Computing** Magazine. For more information, contact **ANALOG Computing** at (213) 858-7100, ext. 163.

Subscriptions

ANALOG Computing, P.O. Box 16927, North Hollywood, CA 91615; (818) 760-8983. Payable in U.S. funds only. U.S.: $28-one year, $54-two years, $76-three years. Foreign: Add $7 per year. For disk subscriptions, see the cards at the back of this issue.

Authors

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

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

ANALOG Computing (ISSN 0744-9917) is published monthly by L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210. © 1988 L.F.P., Inc. Return postage must accompany all manuscripts, drawings, photos, disks, etc., if they are to be returned, and no responsibility can be assumed for unsolicited materials. All rights reserved on entire contents; nothing may be reproduced in whole or in part without written permission from the publisher. U.S. subscription: $28 for one year (12 issues), $52 for two years (24 issues), $76 for three years (36 issues). Foreign subscription: Add $7 per year. Single copy $3.50 (add $1 for postage). Change of address: Six weeks advance notice, and both old and new addresses are needed. POSTMASTER: Send change of address to **ANALOG Computing** Magazine, P.O. Box 16927, North Hollywood, CA 91615. Second-class postage paid at Beverly Hills, CA, and additional mailing offices.

Reader Comment

The Whoops department

While I greatly enjoyed Charles F. Johnson's "Binary Load Pictures" in issue 60 (May '88), I had considerable trouble with Charles Bachand's "The MAC/65 Detokenizer." I've discovered several bugs for which I am enclosing fixes. After carefully typing and rechecking the MAC/65 Detokenizer, I found that:

1) If an error was made in the filename of the MAC/65 file to be read (like leaving off the device D:, for instance), and then the directory was checked for the correct filename, every filename entered after that continued to return errors, and the program had to be restarted, and the filename reentered correctly on the first try!

2) All two-byte hexadecimal literals (like $0342 or $E456, for instance) were mangled in the conversion to hex digits (like $0303 or $E404)!

3) After the file had been detokenized, the output file was never closed, making yet another error every time the program came to Line 1090, where it tried to open the still already open IOCB channel!

The following lines appear to fix these problems when they replace the lines of the same numbers in the original program.

```
1090 CLOSE #1:OPEN #1, 4, 0, A$
1100 GET #1, A:GET #1, B:TRAP 40000
1400 IF A = 5 THEN GET #1, B: GET #1,
A:L = L - 2:GOSUB 1680: A = D:GOSUB
1680:GOTO 1340
1440 A = PEEK (195):IF A = 136 THEN
CLOSE #2:GOTO 1070
1755 CLOSE #3:TRAP 40000:RETURN
```

—Bob Hardy
Chico, CA

I read and typed in all of your programs (even the ones in Base 16). I also typed in the little harmless four-liner on page 37 of the April '88 issue and found the good ol' "ERROR 3 AT LINE 20." I found out that the fifth data number in Line 10 is greater than 255 (298 to be exact). I experimented and found out that the following will do the job:

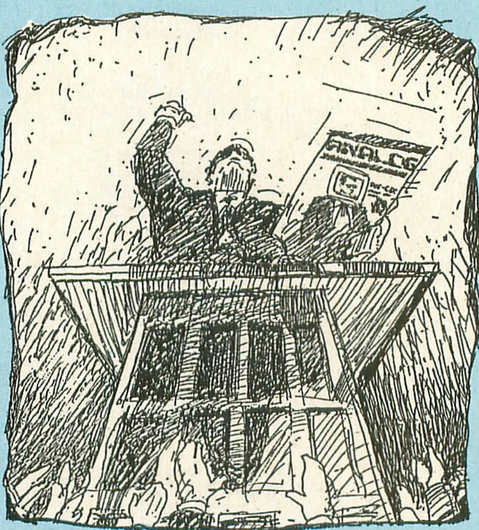
```
10 DATA 238, 198, 2, 238, 198, 2, 76
```

—Jason Locke
Levittown, NY

A bug or not?

I just happened to pick up the August '86 issue from the stack and happened to turn to Clayton Walnum's review of *AtariWriter Plus*. I had read it before I bought AW+ and remembered his comments about a few bugs. Well, there is one bug he did not mention; one that makes AW+ unusable.

If you happen to write a line that exceeds the number of columns allowed, word wrap is supposed to do its thing. Most of the time it does so. But if you happen to use a word that exceeds the limit by one character, word wrap falls down on the job; it moves that extra character down to the next line (at printout time) and does a line feed in doing so. Great! I had purchased AW+ sometime back in '87 but didn't use it until recently and ran into this problem. I put it back into the box and filed it with a few other programs that don't work.



Does anyone out there have a fix?

—Carl C. Springer
Merritt Island, FL

Yes, you did indeed catch a bug that I didn't spot, but to say that that small problem makes AtariWriter Plus unusable is a gross exaggeration. Just the fact that I didn't catch the bug shows how rarely it crops up. I wrote many, many articles with AW+ and never once ran into the problem you describe (although, I did test AW+ again based on your complaint, and found that the software does behave as you've described). When this problem crops up, it's a simple thing to fix; just a quick change of your margin setting will do the trick. I still say that AtariWriter Plus is an excellent product, and I easily recommend it to anyone looking for a good word processor for their 8-bit Atari.

M/L Editor modification

Who hasn't praised the use of M/L Editor by Clayton Walnum when entering all those data statements? However, being a fairly good typist, I had trouble getting used to one feature. I could not get used to seeing a comma and pressing the RETURN key. Nor did I want to. That dilemma prompted this letter and the following solution.

Changing the M/L Editor code to include the following will enable the user to press the comma or RETURN key interchangeably. Now I no longer get confused.

```
25 CMMA = 44
330 IF A < > RETRN AND A < > CMM A
AND A < > BACKSP AND (A < 48 OR
A > 57) THEN 320
335 IF (A = CMMA OR A = RETRN) AND
 = 0 AND X > 1 THEN 350
 40 IF (((A = RETRN OR A = CMMA)
 ND NOT EDIT) OR A = BACKSP) AND
 = 0 THEN 320
 50 IF A = RETRN OR A = CMMA THEN
 OKE 752,1:? " ":RETURN
```

—Larry Locke
Nazareth, PA

800 4 PIECE BOARD SET $28.50

SET INCLUDES 800 MAIN, CPU, 10K ROM AND POWER BRDS
ALL MODULES NEW AND COMPLETE WITH IC'S

800/400 MODULES

$9.50 ea.

- 800 Main Board
- 800/400 CPU with GTIA
- 800 10K "B" O.S. Module
- 16K RAM less RAM IC's
- 400 Main Board
- 800 Power Supply Board
- 400 Power Supply Board
- 800XL Modulator

IC's $4.50 ea.

• CPU	CO14806
• POKEY	CO12294
• PIA	CO14795
• GTIA ...	CO14805
• ANTIC ..	CO12296
• CPU	CO10745
• PIA	CO10750
• CPU	CO14377

SERVICE RATES

• 800	$39.50
• 850	$39.50
• 810	$69.50
• 1200XL ...	$49.50
• 1050	$75.00

INCLUDE $7.00 RETURN SHIPPING AND INSURANCE.

POWER PACKS

Replacement transformer for: 800/400, 810, 1050, 1200XL and 1020 units $14.50
1030 Power Pack ... $9.50

1050 MECH

Factory fresh TANDON mechs. make difficult repairs a snap. Units are complete with Head, Stepper, Spindle motor, belt etc. Just plug in, no difficult alignments or adjustments required. $47.50

MISC. HARDWARE

600XL 64K Upgrade	$29.95
Fastchip for 800/400	$15.50
Atari Joystick	$7.00
Atari Paddles	$6.50

BOARD SETS

New Parts complete with IC's

800 4 PIECE BOARD SET
Includes 800 Main, CPU, 10K ROM and Power Brd. $28.50

400 3 PIECE BOARD SET
Includes 400 Main, CPU and Power Board $19.50

810 BOARD SET
Sideboard with Sep., Power and Analog Boards . $67.50

REPAIR MANUALS

SAMS Service Manuals for the following units: 800, 800XL, 130XE, 400, 1050 .... $19.50
520 ST Service Man. . $37.50

PILOT PROGRAMMING CART + MANUALS $17.50

SOFTWARE

Pac-Man Cartridge ...	$4.00
Deluxe Invaders Cart. .	$4.00
Journey/Planets Cart.	$4.00
Gorf Cart. 800/400 ...	$4.00
Drop Zone Disk	$9.95
Big Math Attack Disk .	$5.00
Pathfinder Disk	$5.00
Encounter on Questar	$5.00
Musical Pilot Childware	$5.00
Basic Cartridge	$10.00
Q*Bert Cartridge	$10.00
Donkey Kong Cart....	$5.00
Eastern Front Cart. ...	$5.00
Atlantis Cartridge.....	$5.00
Crossfire Cart.	$5.00

COMPUTER BOOKS

Inside Atari Basic	$5.00
Atari Basic Ref. Man.	$5.00
XE Users Handbook	$17.50
XL Users Handbook	$17.50
Assembly Language	$19.95
Advanced Prog.	$19.50

CALL TOLL FREE 1-800-551-9995

INSIDE CALIFORNIA OR OUTSIDE U.S. PHONE 415-352-3787

AMERICAN TECHNA-VISION (T.V.)
Mail Order .... 15338 Inverness St., San Leandro, Ca. 94579
Service Center ..... 2098 Pike Ave., San Leandro, Ca. 94577

NO MINIMUM ORDER! We accept money orders, personal checks or C.O.D.s, VISA, Master/Card okay. Credit cards restricted to purchases over $20.00. No personal checks on C.O.D. - **Shipping:** $4.00 shipping and handling on orders under $150.00. Add $2.25 for C.O.D. orders. In Canada total $6.00 for shipping and handling. Foreign shipping extra. Calif. residents include 7% sales tax. Prices subject to change without notice. Atari is a trademark of Atari Corp.

CIRCLE #101 ON READER SERVICE CARD.

SOUTHERN SOFTWARE

205-956-0986

24 HOUR PHONE

CALL OR WRITE FOR FREE CATALOG
ALL 8 BIT AND ST SOFTWARE IN STOCK
CALL FOR PRICES AND LATEST TITLES
SOFTWARE IN STOCK FOR OTHER COMPUTERS

PRICES LISTED FOR MAIL ORDER ONLY - ADD 10% ON PHONE ORDERS

ST COMPUTERS	CALL	CHIPMUNK	29.95	80 COL CARD	79.95
1050 DISK DRIVE	239.95	ATARI WRITER PLUS	39.95	COMPULSIVE COPIER	29.95
1050 W/HAPPY INST	399.95	BASIC CARTRIDGE	19.95	COPY II ST	29.95
130 XE	139.95	BASIC XE	49.95	XM-301 MODEM	49.95
XF551 DISK DRIVE	189.95	BASIC XL	39.95	SX-212 MODEM	89.95
520 DISK DRIVES	CALL	SUPER ARCHIVER	59.95	SMM804 PRINTER	189.95
850 INTERFACE	109.95	BIT WRITER	69.95	1027 PRINTER	124.95
8K OMNIMOM	59.95	DOS 2.5 W/MANUAL	9.95	XDM121 LQ PRINTER	199.95
ACCESSORIES	CALL	XL/XE POWER SUPPLY	27.95	ATARI LAB ST KIT	49.95
APE FACE	49.95	1027 INK ROLLER	4.97	JOY STICK	8.95
AVATEX 1200	79.95	HAPPY ARCHIVER	34.95	LOGIKHRON CLOCK	39.95
AVATEX 2400	179.95	I/O CORD	13.95	MARK WILLIAMS C	124.95
CHIP/ARCHIVER 810	69.95	I/O CORD 10 FT	19.95	MEGAMAX C	139.95
PRO BURNER	179.95	MAC/65	64.95	MODULA-2	69.97
EZ RAM 520	169.95	MAC/65 TOOL KIT	27.95	P.R. CONNECTION	59.95
HAPPY 1050 ENHANC	119.95	ACTION	64.95	WARP SPEED DOS XL	24.95
HAPPY 810 ENHANCE	104.95	ACTION TOOL KIT	27.95	PC BOARD DESIGNER	169.95
HAPPY CONTROLER	39.95	BASIC XL TOOL KIT	27.95	PROLOG	69.95
HARD DISK DRIVES	CALL	BASIC XE TOOL KIT	27.95	PUBLISH PARTNER	119.95
ICD MIO 1 MEG	299.95	PERSONAL PASCAL	74.95	SOUND DIGITIZER	119.95
ICD MIO 256K	199.95	SPARTADOS TOOL KIT	24.95	* WE HAVE COLOR RIBBONS *	
XE ADAPTER FOR MIO	19.95	ST HOST ADAPTER	99.95	FOR ALL PRINTERS	CALL
POWER SUPPLIES	CALL	SPARTS DOS CART	69.95	------------------------	
PRINTERS	CALL	OMNIVIEW XL/XE	36.95	* ATARI REPAIR PRICES *	
PRINTER CONNECT.	39.95	NEWELL 256K	34.95	ITEMS NOT LISTED	CALL
R-TIME CARTRIDGE	49.95	OMNIMON 400/800	44.95	1050 DISK DRIVE	89.95
RAMBO XL UPGRADE	29.95	ST COPY	29.95	130 XE	69.95
256K CHIP SET	49.95	NUMERIC KEYPAD	39.95	65 XE	49.95
RAMROD XL	39.95	TOP DOS 1.5 PLUS	29.95	520 DISK DRIVE	89.95
LIGHT PEN	69.95	PRINT/MODEM CABLE	13.95	520 ST	139.95
U.S. DOUBLER	29.95	RAMCHARGER	139.95	850 INTERFACE	49.95
UPRINT INTERFACE	59.95	RAMCARD FOR 800	129.95	ATARI PRINTER	69.95
XL/XE BOS	49.95	SMART LINK MODEM	189.95	XL/XE/150 POW SUP	12.95

PRICES SUBJECT TO CHANGE WITHOUT NOTICE
ADD $5 FOR SHIPPING AND INSURANCE. MOST ORDERS SHIPPED SAME DAY. FOREIGN ORDERS WELCOME WITH SUFFICIENT POSTAGE INCLUDED. ALABAMA RESIDENTS ADD 7% SALES TAX. ADD 6% FOR VISA. ADD $5 FOR AIRMAIL. ADD $15 FOR OVERNIGHT SHIPMENT
ALLOW THREE WEEKS FOR PERSONAL CHECKS.

SOUTHERN SOFTWARE

1879 RUFFNER ROAD BIRMINGHAM, AL 35210

CIRCLE #102 ON READER SERVICE CARD.

ATARI 8-BIT POWER

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

NEW
HACK PACK
Special OFFER

The Alpha Systems HACK PACK contains all our finest products for making Back-up copies, Analyzing, Understanding and Protecting your Atari programs. It comes complete with Atari Protection Techniques (Book and Disk I), Advanced Protection Techniques (Book and Disk II), The Chipmunk, The Scanalyzer, The Impersonator and Disk Pack 1000. Worth over $150. Get them all for the special price of **Just $99.95**

Atari Software Protection Techniques Vol I & II

These Book and Disk packages detail the most advanced copy protection methods in use today. They guide you through the methods used to create the protection as well as the copying techniques to get around them. They include information on Phreaking • Hacking • On-line security • Black boxes • Self-destructing programs • Pirate bulletin board systems • Logic bombs • New piracy laws • Hardware data keys • Weak sectoring (Phantom, Fuzzy and unstable sectors) • Overfilled tracks • CRC errors • Bank Select cartridges and MUCH, MUCH MORE. The disks include automatic program protectors, Protection Scanners, directory hiding and more.

BOOK I and DISK I	**$24.95**
BOOK II (Advanced protection) and DISK II	**$24.95**
Special Offer, Order both sets for Only	**$39.95**

CHIPMUNK

Automatic Disk Back-Up System. Make perfectly running unprotected back-up copies of hundreds of the most popular Atari programs. Chipmunk's sophisticated programming Automatically finds and **REMOVES copy protection** from most Atari programs. Back-up even heavily protected programs with ease. Finally, a back-up system that needs no special hardware or skills.
(If you need a full list of what Chipmunk copies, call or write for our free catalog) **$34.95**

Scanalyzer Automatically scan & analyze commercial programs. Unlock programming secrets and learn from the masters **$29.95**

Impersonator Cartridge to Disk back up system. Create running back-up copies of any cartridge (up to 16K) **$29.95**

NEW
CHEAT

Get more from your games with CHEAT. Tired of spending days trying to beat a game? Tired of getting stuck just when you need another life? Cheat is an innovative new product that gives you the chance you need to beat your favorite games. Cheat works with hundreds of Atari games to give you unlimited lives or power. End the frustration and get hours more enjoyment from your games. (Call or write Alpha Systems for our free catalog with a full list of the programs that work with Cheat) **ONLY $24.95**

BASIC TURBOCHARGER

NOW for the first time a BASIC programmer can get the power, flexibility and incredible speed of machine language. BASIC TURBOCHARGER is a **book and disk package** that contains over 150 ready to use machine language routines. Complete instructions show how to add them to your own BASIC programs to get these features and more: • Smooth Scrolling • Player/Missile control • Load & Save Picture files • Sorting and Searching • Special Effects Graphics • Incredible Speed • Much, Much More • Over 150 programs. You've heard of the power of Assembler, now harness it for your own needs. **$24.95**

NEW
PARROT II

An All New Parrot sound digitizer for your Atari. Parrot II is a sophisticated new hardware device that plugs into your joystick port. Parrot II has two inputs, One for a microphone and one for a powered source such as a tape player, radio or Compact Disk.

The Powerful Parrot II software lets you record sounds into your computer and play them back on any Atari. Parrot II turns your computers keyboard into a musical instrument with nine different sounds covering three octaves each. The sounds can be anything, a dogs bark, a piano, a complete drum set, a symphony or your own voice.

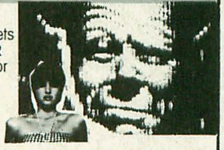
Parrot II lets you modify the sounds on a graphic display to create brand new sounds and special effects. Best of all, the sounds and voices can be put into your own programs that can be used on any standard Atari. Explore the world of digital sound and music. ONLY **$59.95**

Pre-Recorded Sound Disk More pre-recorded sounds for Parrot	**$4.95**
PARROT II Demo Disk (Does not require Parrot to run)	**$5.00**

NEW
POP-N-ROCKER

a fast paced, multi-player trivia game that mixes questions with real songs (digitized with Parrot). Be the first to identify the songs and answer the music trivia questions. *Pop-N-Rocker* comes with three data disks and lets you add new questions so it will never get old. You can use a Parrot Sound digitizer to add new songs too! Use any kind of music from Rock to Classical to Nursery Rhymes. A new concept in entertainment and a perfect add-on for Parrot. **$24.95**

COMPUTEREYES & MAGNIPRINT II +



GIANT WALL SIZED POSTERS.

Turn your computer into a digital portrait studio. This complete package lets you **capture, save & print** digital images from your **Video Camera, VCR or TV.** COMPUTEREYES hardware plugs directly into your joystick ports for easy use. Print your picture on a 6 foot poster. **$119.95**

ComputerEyes camera system

Comes complete with everything above, plus a black and white video camera and connecting cable. **$329.95**

Graphics 9 Software – Add a new dimension to your COMPUTEREYES pictures – captures images in 16 shades of grey. **$12.00**

Magniprint II +

Easily the most powerful print program available today. Print graphics from almost any format in hundreds of shapes, sizes, and shades. Supports **color printing** and lets you create **giant posters.** Magniprint II + lets you stretch and squeeze, invert, add text, adjust shading and much more. Works with EPSON, NEC, Citoh, Panasonic, Gemini, Star, XMM801, and compatible printers. (850 interface or equivalent required). **$24.95**

Graphics Transformer

Now you can combine the most powerful features of all your graphics programs. Create print shop icons from a Koala pad picture, from a photo digitized with ComputerEyes, or any picture file. Graphics Transformer lets you **Shrink, Enlarge and Merge** pictures for unequaled flexibility. **$22.95**

YOUR ATARI COMES ALIVE

SAVE MONEY. Finally an alternative to buying expensive computer add-ons. Your Atari Comes Alive shows you how to **built them yourself.** This 'How-To' **book and disk package** gives you complete step by step instructions and programs needed to built and control these exciting devices and MORE: • Light Pen • Light & Motor Controllers •Alarm Systems • Voice Recognition • Environmental Sensors •Data Decoders • More than 150 pages. **Your Atari Comes Alive** **$24.95**

ALPHA SYSTEMS

24 HOUR HOTLINE **216-374-7469**

VISA & MASTERCARD, ORDER BY PHONE, OR SEND MONEY ORDER TO:

ALPHA SYSTEMS 1012 SKYLAND DRIVE MACEDONIA, OH 44056 **FREE BONUS:** DELUXE SPACE GAMES (3 games on a disk) Free with any order of 3 or more items. Include $3.00 shp & hdlg (US Canada) Ohio res. add 5 1/2% sales tax. Foreign orders add $8.00 shp & hldg. Call or write for free catalog. Customer Service Line (216) 467-5665 M-F 9-3.

CIRCLE #103 ON READER SERVICE CARD.

Master Me

Part II

How to read the Memory Map

Beginning Users—Read the text that is printed in bold type only. These memory locations will be the easiest for you to use and usually don't involve assembly language.

Advanced Users—Read everything! Many areas of memory are not of any practical use, but you can learn a lot about how a computer works by reading the boring parts.

The information is formatted like this:

LABEL
DECIMAL # HEXADECIMAL #
DESCRIPTION

HEXadecimal numbers are often preceded by a "$".

Page Zero

Locations 0 to 255 are called "Page Zero" (in the language of computers, a "page" is 256 bytes). Since one byte can hold any number in the range of 0 to 255, the computer only needs one byte to hold the address of a page zero location. This saves time when you have to load or store a value in machine language so page zero is very important to machine-language programs that have to run as quickly as possible. That's why the operating system uses the first 128 bytes. The other 128, locations 128 through 255, can be used by BASIC and you for superfast machine code.

Machine-language programmers should note that Locations 2 through 7 are *not* cleared by either a coldstart (turning the computer off and then on again) or warmstart (pressing SYSTEM RESET) operation.

LINZBS
0,1 0000,0001

The great mystery location! Nobody seems to know exactly what this location does. According to Atari's operating system listing, it is "LINBUG RAM [and] will be replaced by [the] monitor RAM" (your guess is as good as mine), and the only time it uses it is to define it. It *does* seem to be used to store the VBLANK timevalue though, so it's probably not completely useless.

CASINI
2,3 0002,0003

This is used in "cassette initialization." As you probably already know, if you hold down the Start button while turning on the computer, the computer will beep. If you then press RETURN, the computer will expect a machine-language tape to be in the cassette recorder and will proceed to load it. This process is called "booting" a cassette. The first six bytes stored on the machine-language tape contain special information about the tape. The first byte, actually, is ignored. The second tells how many 128 byte "records" are on the tape (when you load in the tape, each beep while loading represents a record). The third and fourth give the starting address that the machine code is to be stored at, called the "load" address. The fifth and sixth give the "initialization" address (where to go to get the program set up and ready to run). The initialization address, as you may have guessed, gets stored here at CASINI. Once the whole program has loaded, the computer jumps to the load address plus six (to skip over these special bytes) where the program either tells it to load some more or RTS (ReTurn from Subroutine). When the computer comes across an RTS instruction, it looks in CASINI for the initialization address and JSRs (Jump to SubRoutine) to that address. Finally (and you thought cassette boots were easy), the computer JSRs to the address in DOSVEC (10,11), which gets the program running (DOSVEC should be set up by the program either in the initialization process or as part of a multiple load).

RAMLO
4,5 0004,0005

RAMLO has a bunch of uses, none of which will be useful to you. First, the OS uses it as an index (like the variable in a FOR/NEXT loop) while clearing out memory after you turn on the computer. It also uses it as an index while testing memory to make sure everything is A-okay. Finally, and you'll love this one, it's used to store the "disk boot address," which is usually 1798 in case you care, for the boot continuation routine (which is what happens when you want to load into more than one part of memory). By the way, it's real buddy-buddy with TRAMSZ and TSTDAT (the three work together in the RAM test routine).

TRAMSZ
6 0006

Another location with a whole bunch of uses. As mentioned, TRAMSZ helps out RAMLO in testing the RAM. Its value is then transferred to RAMTOP (location 106). But, before any of that happens, it is used in testing whether or not a left cartridge is plugged in. If there is a left cartridge (also known as cartridge A), then TRAMSZ is set to one. If not, it's set to zero.

TSTDAT
7 0007

This one only has two functions. First, as you already know, it helps out in the RAM test routine (see your OS listing if you're dying to find out what the RAM test routine is). Secondly, like TRAMSZ, it's used initially in testing whether or not the right or B cartridge is present.

Machine-language programmers: Locations 8 through 15 are cleared on coldstart only.

WARMST
8 0008

This is the warmstart flag, telling you whether you're in the middle of a warmstart or a coldstart. If WARMST equals 0, then you're in the middle of a coldstart. If it's anything else, then you're in the middle of a warmstart (pressing SYSTEM RESET will set WARMST to 255). The main purpose of WARMST is to make sure that if someone presses the SYSTEM RESET button before everything is initialized properly, the computer will know about it and start over instead of messing

mory Map

by Robin Sherer

TUTORIAL

everything up. Nice stuff to know, but generally useless. But wait, you say, can't I trick the computer into rebooting by changing the value of WARMST to 0, that way preventing people from using SYSTEM RESET to stop by BASIC program so they can LIST it? No. Although you can change the value to 0, as soon as you press SYSTEM RESET it will change back to 255. See Location COLDST (580) for a way that you *can* trick the computer. You might also look at locations POKMSK (16) and STMCUR (138,139) for other ways to protect your BASIC programs from other people's greedy eyes.

Incidentally, warmstart normally starts at location 58484.

BOOT?
9 0009

Booting, as you will recall, is the process of loading the program into the computer's memory. In our case the program is loaded from tape or disk. Sometimes a boot is not successful. Maybe you put a rock 'n' roll tape into your Atari recorder by mistake, or you forget to close the disk-drive door. In any case, BOOT? is used to tell the operating system whether or not the boot attempt was successful. If BOOT? is equal to one, then there was a successful disk boot. A two indicates a disk boot, and a three (a one plus the two) means both the disk and

continued on page 28

When you want to talk Atari

XL/XE HARDWARE

INTERFACES

ICD	
P:R Connection	59.99
Printer Connection	41.99
Supra	
1150	38.99
1151 (1200 XL)	41.99
Xetec	
Graphix Interface	38.99
Atari	
850 Interface	109.00

COMPUTERS

Atari 130XE **$139**

Atari	
65 XE	99.99

XL/XE ENHANCEMENTS

Axlon 32K Mem. Board (400/800)	19.99
Atari 80 Column Card	79.99

MODEMS

Atari	
SX212 300/1200 (ST)	89.99
835	19.99
XMM301	42.99
Anchor	
VM520 300/1200 ST Dir. Con.	119.00
Avatex	
1200 HC	99.99
2400	179.00
Supra	
2400 Baud XL, XE	169.00
2400 Baud ST	169.00
2400 Baud (no software)	149.00

MONITORS

Magnavox	
CM8505 14" Composite	199.00

ST HARDWARE

ATARI SM1224 RGB Color Monitor **$339**

520ST FM RGB/Color System	789.00
Includes: 520 ST FM with 3½" drive, mouse & 1224 color monitor.	
SM124 Monochrome Monitor	179.00

Call For Current Information On The Entire ST Line!

DRIVES

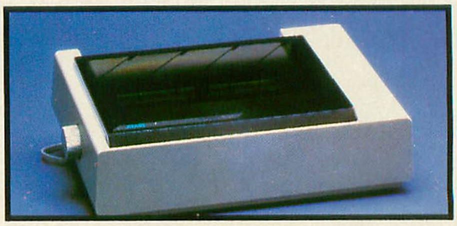
Atari	
XF551 Drive (XL/XE)	189.00
AA314 DS/DD (ST)	209.00
SHD204 20 Meg Hard Drive	Call

Supra 30 Meg ST Hard Drive **$699**

I.B.	
5¼" 40 Track (ST)	219.00
5¼" 80 Track (ST)	279.00
I.C.D.	
FA•ST 20 Meg	629.00
FA•ST 30 Meg	869.00
FA•ST Dual Hard Drives	Call
Indus	
GTS 100 3½" DS/DD (ST)	199.00
GT 1000 5¼" DS/DD (ST)	219.00
GT Drive (XL/XE)	189.00
Supra	
FD-10 10MB Removable Floppy w/SCSI	789.00
20 Meg Hard Drive (ST)	579.00
20 Meg Hard Drive (XL/XE)	Call

PRINTERS

Atari	
1027 LQ XL/XE	129.00



Atari XM-M801 XL/XE Dot Matrix **$199**

XM-M804 ST Dot Matrix	199.00
XDM 121 Letter Qlty. XL/XE	209.00
Brother	
M-1109 100 cps Dot Matrix	189.00
M-1409 180 cps Dot Matrix	299.00
HR-20 22 cps Daisywheel	339.00
Citizen	
120D 120 cps Dot Matrix	149.00
180D 180 cps Dot Matrix	179.00
Premier-35 35 cps Daisywheel	479.00
Epson	
LX-800 150 cps, 80 col	179.00
Hi-80 4 pen plotter	269.00
FX-86E 240 cps, 80 col	Call
FX-286E 240 cps, 132 col	Call
LQ-500 180 cps, 24-wire	Call
LQ-850 330 cps, 80 col	Call
FX-850, FX-1050	New
NEC	
P2200 pinwriter 24-wire	369.00
P660 pinwriter 24-wire	459.00
P760 pinwriter 132 col	659.00
Okidata	
Okimate 20 color printer	129.00
ML-182 + 120 cps, 80 column	229.00
ML-192 + 200 cps, 80 column	349.00
ML-193 + 200 cps, 132 col.	459.00
Panasonic	
KX-P1080i 144 cps, 80 col	179.00
KX-P1091i 194 cps, 80 col	199.00
Star Micronics	
NX-1000 140 cps, 80 column	179.00
NX-15 120 cps, 132 column	319.00
Toshiba	
P321-SL 216 cps, 24-wire	499.00

WE SHIP 90% OF ALL ORDERS WITHIN 24 HOURS

SELECT FROM OVER 3000 PRODUCTS

CMO

COMPUTER MAIL ORDER

..........you want to talk to us.

XL/XE SOFTWARE

Access	
Executive Disk	19.99
Accolade	
Hardball	19.99
Atari	
Atariwriter Plus	35.99
Filemanager	11.99
Music Painter	11.99

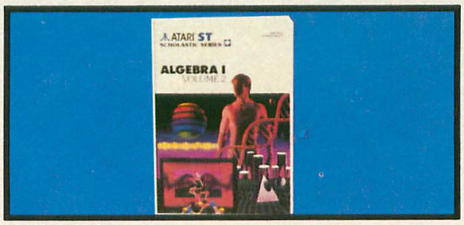
ACCESS
Leaderboard Golf **$13⁹⁹**

Atari Program Exchange	
Misc. Programs (cassettes)	at 1.99
Broderbund	
Graphics Library I, II, III	13.99
Printshop	25.99
Datasoft	
Alternate Reality (dungeon)	25.99
221 Baker St.	20.99
Electronic Arts	
Touchdown Football	12.99
Firebird	
Guild of Thieves	14.99
Silicon Dreams	14.99
Jewels of Darkness	14.99
Microprose	
Top Gunner	15.99
F-15 Eagle Strike	21.99
Silent Service	21.99
Origin Systems	
Ultima 4	36.99
Roklyn SPECIAL	
Anti-Sub/Journey to Planet	(ea.) 3.99
Strategic Simulations	
Battallion Commander	23.99
Gemstone Warrior	10.99
Sublogic	
Flight Simulator II	32.99
Scenery Arizona	14.99
X-Lent	
Typesetter	22.99
Printshop Interface	21.99

ACCESSORIES

MD1-M SS/DD 5¼"	8.49
MD2-DM DS/DD 5¼"	8.99
MF-1DDM SS/DD 3½"	11.99
MF2-DDM DS/DD 3½"	18.49
Sony	
MD1D SS/DD 5¼"	6.99
MD2D DS/DD 5¼"	7.99
MFD-1DD SS/DD 3½"	11.99
MFD-2DD DS/DD 3½"	17.99
Allsop Disk Holders	
Disk File 60-5¼"	9.99
Disk File 30-3½"	9.99
Curtis	
Emerald	39.99
Safe Strip	19.99
Universal Printer Stand	14.99
Tool Kit	22.99
ICD	
BBS Express (ST)	49.99
Sparta DOS Construction Set	28.99
US Doubler/Sparta DOS	47.99
Real Time Clock	48.99
Rambo XL	29.99
US Doubler	28.99

ST SOFTWARE

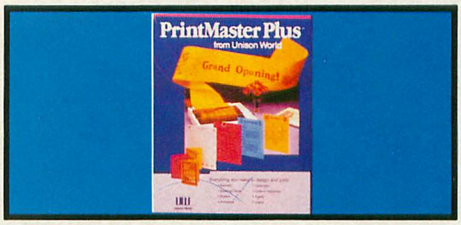


ATARI
Geometry
Vol. II **$16⁹⁹**

Abacus	
PC Board Designer	129.00
Access	
Leaderboard Golf	22.99
Activision	
Hacker II Doomsday	27.99
Antic	
CAD 3-D	31.99
Avant Garde	
PC Ditto	59.99
Batteries Included	
Degas Elite	38.99

ST SOFTWARE

Comnet	
ST Term	19.99
Electronic Arts	
Gridiron Football/Auto Duel. (ea.)	26.99
Isgur Portfolio	119.00
Firebird	
Silicon Dreams	19.99
The Sentry	19.99
Infocom	
Beyond Zork	34.99
Metacomco	
ISO Pascal	59.99
Michtron	
Major Motion	25.99
Microprose	
Gunship	28.99
F-15 Strike Eagle	24.99
Miles Software	
ST Wars	24.99
Mark Williams	
C	119.00
Paradox	
Wanderer (3D)	24.99
Progressive Computer	
Graphic Artist 1.5	129.00
Psygnosis	
Barbarian/Deep Space	(ea.) 25.99
Soft Logik Corp.	
Publishing Partner	54.99
Strategic Simulations	
Rings of Zilfin/Phantasie III (ea.)	25.99
Sublogic	
Flight Simulator II	33.99
Timeworks	
Desktop Publisher	89.99
Swiftcalc/Wordwriter	(ea.) 46.99
Partner ST	37.99



UNISON WORLD
Printmaster Plus **$25⁹⁹**

Word Perfect Corp	
Word Perfect 4.1	189.00

In the U.S.A. and in Canada

Call toll-free: 1-800-233-8950

Outside the U.S.A. call 717-327-9575, Fax 717-327-1217
Educational, Governmental and Corporate Organizations call toll-free 1-800-221-4283
CMO, 101 Reighard Ave., Dept. B7, Williamsport, PA 17701

OVER 350 000 SATISFIED CUSTOMERS • ALL MAJOR CREDIT CARDS ACCEPTED • CREDIT CARDS ARE NOT CHARGED UNTIL WE SHIP

POLICY: Add 3% (minimum $7.00) shipping and handling. Larger shipments may require additional charges. Personal and company checks require 3 weeks to clear. For faster delivery, use your credit card or send cashier's check or bank money order. Credit cards are not charged until we ship. Pennsylvania residents add 6% sales tax. All prices are U.S.A. prices and are subject to change, and all items are subject to availability. Defective software will be replaced with the same item only. Hardware will be replaced or repaired at our discretion within the terms and limits of the manufacturer's warranty. We cannot guarantee compatibility. All sales are final and returned shipments are subject to a restocking fee. We are not responsible for typographic or photographic errors.

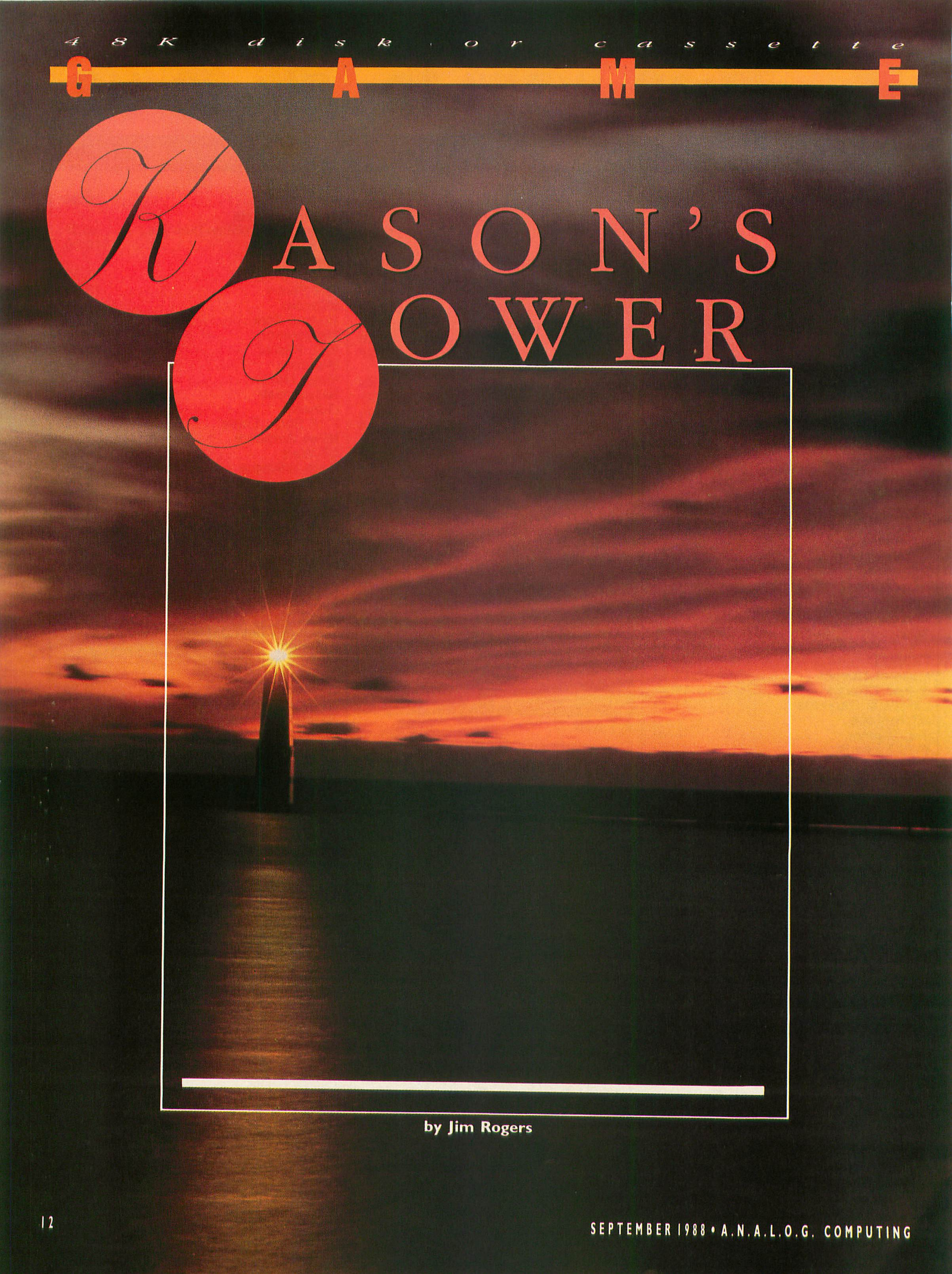
CIRCLE #104 ON READER SERVICE CARD.

B709

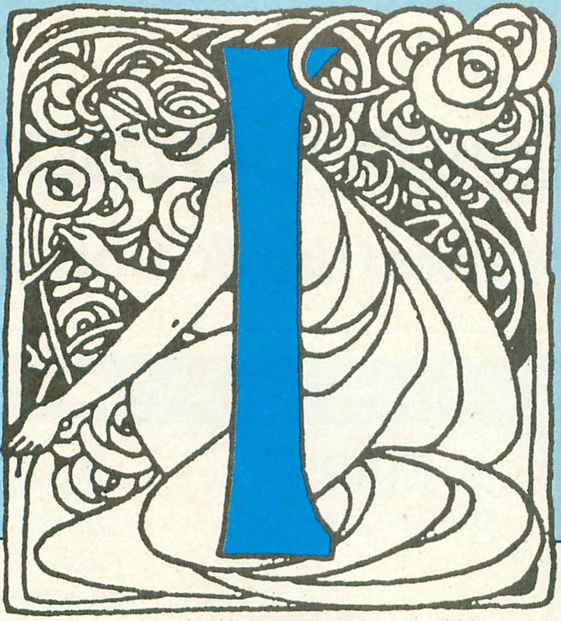
48K disk or cassette

GAME

KASON'S TOWER



by Jim Rogers



It's the year 3456, and even though the year is more than half over, you still continue to put 3455 on all of your checks. This is the fifth year in your search for the lost arts of civilization, the meaning of life, the secret of peace among all men and the ultimate free tabloid.

In disgust, you've taken refuge in the lost mountain town of Golenden, where your host (he's the ancestor of the man who was considered to be the most important citizen of Golenden at the time of the fall of civilization—the plumber) has been most gracious. Unfortunately, you rather tactlessly referred to his daughter as "that repulsive little squash-faced gnome," and now, as your punishment, you must scale the ancient Kason's Tower.

The game

Type in Listing 1 using *Basic Editor II* found elsewhere in this issue. *Save a copy of the program before you run it!* If you fail to do this, you'll have to retype the program because it erases itself from memory.

At the start of the game, your man is on one side of a floor of the tower, and a pole is on the other. Use your joystick to move your man across the floor to the pole, avoiding the arrows that will be flying toward you. Pull down on the joystick to duck beneath the high arrows, and press the trigger to jump over the low arrows. When you reach the pole, your man will be teleported to the next floor. Upon the completion of the top floor, your man will begin on the next level. You're given five men at the beginning of the game, and when these are used up, the game is over.

This program uses BASIC to directly load and run the large machine-language portion of the game. The program is erased after it is run, so be very careful to save a copy before running.

Jim Rogers has had his 800XL for four years, and is completely self-taught in both BASIC and machine language.

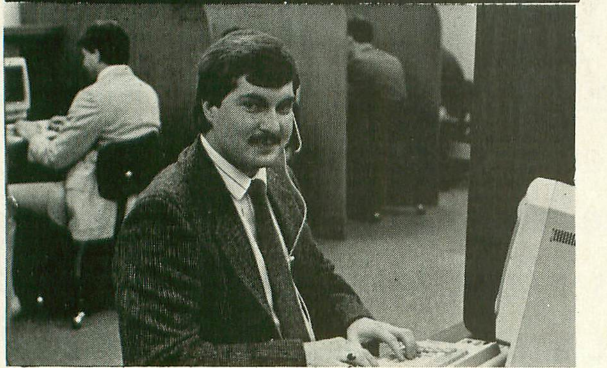
continued on page 40

Lyco Computer

Marketing & Consultants

Air orders processed within 24 hours.

Lyco Means Total Service.



Mark "Mac" Bowser, Sales Manager

I would personally like to thank all of our past customers for helping to make Lyco Computer one of the largest mail order companies and a leader in the industry. Also, I would like to extend my personal invitation to all computer enthusiasts who have not experienced the services that we provide. Please call our trained sales staff at our toll-free number to inquire about our diverse product line and weekly specials.

First and foremost our philosophy is to keep abreast of the changing market so that we can provide you with not only factory-fresh merchandise but also the newest models offered by the manufacturers at the absolute best possible prices. We offer the widest selection of computer hardware, software and accessories.

Feel free to call Lyco if you want to know more about a particular item. I can't stress enough that our toll-free number is not just for orders. Many companies have a toll-free number for ordering, but if you just want to ask a question about a product, you have to make a toll call. Not at Lyco. Our trained sales staff is knowledgeable about all the products we stock and is happy to answer any questions you may have. We will do our best to make sure that the product you select will fit your application. We also have Saturday hours — one more reason to call us for all your computer needs.

Once you've placed your order with Lyco, we don't forget about you. Our friendly, professional customer service representatives will find answers to your questions about the status of an order, warranties, product availability, or prices.

Lyco Computer stocks a multimillion dollar inventory of factory-fresh merchandise. Chances are we have exactly what you want right in our warehouse. And that means you'll get it fast. In fact, orders are normally shipped within 24 hours. **Free shipping on prepaid orders over $50, and there is no deposit required on C.O.D. orders.** Air freight or UPS Blue/Red Label shipping **is available, too. And all products carry the full manufacturers' warranties.**

I can't see why anyone would shop anywhere else. Selection from our huge in-stock inventory, best price, service that can't be beat—we've got it all here at Lyco Computer.

TO ORDER, CALL TOLL-FREE: 1-800-233-8760
New PA Wats: 1-800-233-8760
Outside Continental US Call: 1-717-494-1030

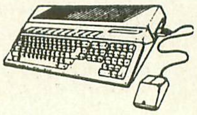
Hours: 9AM to 8PM, Mon. - Thurs.
9AM to 6PM, Friday — 10AM to 6PM, Saturday

For Customer Service, call 1-717-494-1670,
9AM to 5PM, Mon. - Fri.
Or write: Lyco Computer, Inc.
P.O. Box 5088, Jersey Shore, PA 17740

VISA® MasterCard

C.O.D. Risk-Free Policy: • full manufacturers' warranties • no sales tax outside PA • prices show 4% cash discount; add 4% for credit cards • APO, FPO, international: add $5 plus 3% for priority • 4-week clearance on personal checks • we check for credit card theft • sorry, compatibility not guaranteed • return authorization required • due to new product guarantee, return restrictions apply • price/availability subject to change • prepaid orders under $50 in Continental US, add $3.00

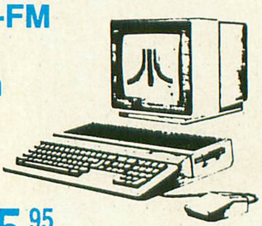
ATARI®
520 ST Computer



- Built-in Drive

ONLY **$469.95**

ATARI®
520 ST-FM Color System

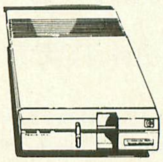


Internal drive included

$785.95

ATARI®
GTS-100

- Atari ST Drive
- 3.5" DSDD

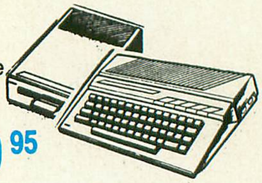


$195.95

ATARI® **130 XE System**

System Includes:

- 130 XE Computer
- 551 Drive



$299.95

HEADSTART
COLOR SYSTEM

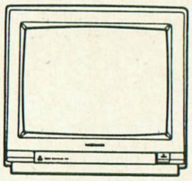
- plug in and use immediately
- IBM-XT compatible
- 2-360K Drives
- Free 1-year limited warranty

$989.95

- Hi Res color monitor included!

MAGNAVOX
CM-8502

- Composite Color
- Green Text Switch
- Speaker



$179.95 1-Year Warranty

ATARI®
HARDWARE

520 ST-FM Keyboard	$469.95
520 ST-FM Color	$785.95
130 XE Computer	$135.95
GTS 100 Drive	$195.95

Call for More Hardware Information

JOYSTICKS

Tac 3	$9.95
Tac 2	$10.95
Tac 5	$12.95
Boss	$11.99
3 Way	$19.99
Winner 909	$24.95
Wic IBM/AP	$29.95
I Controller	$13.95
Epyx 500XJ	$13.95
Kraft KCIII AP/PC	$16.95

1-800-233-8760

CIRCLE #105 ON READER SERVICE CARD.

star micronics

NX-1000

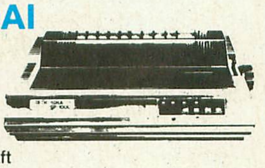
- 144 cps Draft
- 36 cps NLQ
- EZ Font Panel Control

$165.95*

*w/cable purchase

NX-1000 Rainbow Color Printer $225.95

SEIKOSHA SP-180AI

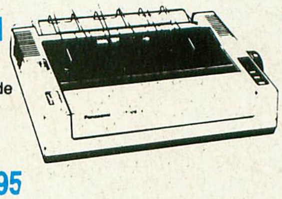


- 100 cps Draft
- 20 cps NLQ
- Std, Par, and IBM Graphics Compatible

$125.95

Panasonic Office Automation OA

1080i Model II



- 150 cps Draft Mode
- NLQ Mode
- Friction & Tractor Feed

$159.95

(2-Year Warranty)

PRINTERS

Seikosha

SP 180Ai	$125.95*
SP 180VC	$125.95*
SP 1000VC	$139.95
SP 1000AP	$159.95
SP 1200VC	$149.95
SP 1200Ai	$159.95
SP 1200AS RS232	$159.95
SL 80Ai	$289.95
MP5420FA	$999.95
SP Series Ribbon	$7.95
SK3000 Ai	$339.95
SK3005 Ai	$419.95
SPB 10	$CALL
SL 130Ai	$599.95

*Quantities Limited

Interfacing available for IBM C 64, Apple and Atari

Panasonic

1080i Model II	$159.95
1091i Model II	$189.95
1092i	$299.95
1592	$375.95
1595	$419.95
3131	$289.95
3151	$459.95
KXP 4450 Laser	$1649.95
1524 24 Pin	$529.95
Fax Partner	$579.95
Optical Scanner	$859.95

Epson

LX800	$184.95
FX86E	$279.95
FX286E	$424.95
EX800	$399.95
LQ500	$339.95
LQ2500	$789.95
GQ3500	$LOW
LQ850	$525.95
LQ1050	$699.95

STAR

NX-1000	$165.95*
NX-1000C	$169.95
NX-1000 Color	$225.95
NX-1000C Color	$229.95
NX-15	$289.95
NR-10	$319.95
NR-15	$419.95
NB-15 24 Pin	$669.95
NX-2400	$309.95
NB24-10 24 Pin	$399.95
NB24-15 24 Pin	$545.95
Laser 8	$1759.95
ND-15	$349.95
NL-10	$149.95

*w/cable purchase

Toshiba

321SL	$489.95
341 SL	$659.95
P351 Model II	$899.95
351 SX 400 cps	$979.95

Citizen

120 D	$149.95
180 D	$169.95
MSP-10	$259.95
MSP-40	$289.95
MSP-15E	$335.95
MSP-50	$399.95
MSP-45	$425.95
MSP-55	$479.95
Premiere 35	$459.95
Tribute 224	$619.95
Tribute 124	$489.95

Brother

M1109	$159.95
M1509	$335.95
M1709	$459.95
Twinwriter 6 Dot & Daisy	$899.95
M1724L	$619.95
HR20	$345.95
HR40	$559.95
HR60	$649.95

Okidata

Okimate 20	$129.95
Okimate 20 w/cart	$189.95
120	$189.95
180	$219.95
182	$209.95
182 +	$225.95
183	$239.95
192 +	$339.95
193 +	$449.95
292 w/interface	$449.95
293 w/interface	$585.95
294 w/interface	$819.95
393	$955.95
Laser 6	$CALL
390	$479.95
391	$649.95
320	$345.95
321	$445.95

Color Ribbons Available

ATARI

Access:	
Triple Pack	$11.95
Leader Board Pack	$9.99
Activision:	
Music Studio	$19.95
Solid Gold Vol.#1	$10.95
Batteries Included:	
Paperclip 80 Col	$31.95
Broderbund:	
Print Shop	$25.95
Print Shop Compan.	$22.95
Graphic Lib. I, II, III	$13.95
Bank St. Writer	$27.95
Electronic Arts:	
Pinball Con Set	$8.95
Lords of Conquest	$8.95
Starfleet I	$32.95
Chess Master 2000	$25.95
Music Con Set	$8.95
Super Boulderdash	$8.95
One on One	$8.95
Microleague:	
Microleag. Baseball	$22.95
General Manager	$16.95
Stat Disk	$13.95
'87 Team Disk	$13.95
Microprose:	
Conflict in Vietnam	$22.95
F-15 Strike Eagle	$19.95
Kennedy Approach	$13.95

ATARI ST

Access:	
Leader Board	$22.95
Tournament #1	$11.95
10th Frame	$22.95
Activision:	
Champion. Baseball	$22.95
GFL Football	$22.95
Music Studio	$27.95
GBA Basketball	$22.95
Beyond Zork	$28.95
Zork Trilogy	$27.95
Broderbund:	
Superbike Challenge	$11.95
Electronic Arts:	
Arctic Fox	$25.95
Starfleet I	$32.95
Chess Master 2000	$25.95
Gridiron	$32.95
Marble Madness	$23.95
Epyx:	
Sub Battle Simulator	$22.95
World Games	$22.95
Wrestling	$11.95
Winter Games	$11.95
Firebird:	
Pawn	$13.95
Starglider	$25.95
Golden Path	$11.95
Guild of Thieves	$25.95
Tracker	$11.95

ATARI ST

Microleague:	
Microleague Baseball	$33.95
General Manager	$16.95
Wrestling	$25.95
Microprose:	
Silent Service	$22.95
F-15 Strike Eagle	$24.95
Gunship	$28.95
Strategic Simulations:	
Phantasie	$22.95
Phantasie II	$22.95
Wargame Con. Set	$19.95
Phantasie III	$22.95
Sublogic:	
Flight Simulator II	$30.95
Scenery Disk	$14.95
Timeworks:	
Wordwriter ST	$44.95
Partner ST	$27.95
Data Manager ST	$44.95
Unison World:	
Art Gallery 1 or 2	$14.95
Print Master	$19.95
Fonts & Borders	$17.95
Art Gallery Fantasy	$13.95

We stock over 3,000 software titles!

Monitors

Thomson:

230 Amber TTL/12"	$69.95*
4120 CGA	$219.95
4160 CGA	$244.95*
4460 EGA	$349.95

Blue Chip:

BCM 12" Green TTL	$64.95
BCM 12" Amber TTL	$69.95

NEC

Multisync II	$589.95

Magnavox:

BM7652	$79.95
BM7622	$79.95
7BM-613	$79.95
7BM-623	$79.95
CM8502	$179.95
CM8505	$199.95
9CM-053	$CALL
CM8762	$245.95
8CM-515	$259.95
CM9043	$CALL
8CM-873	$499.95

Free shipping on prepaid orders over $50 in continental U.S.

Modems

Avatex:

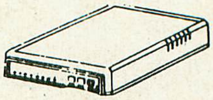
1200e	$69.95
1200i PC Card	$65.95
1200hc Modem	$69.95*
2400	$149.95
2400i PC Card	$139.95

*w/cable purchase

Hayes:

Smartmodem 300	$139.95
Smartmodem 1200	$279.95
Smartmodem 2400	$419.95

Avatex™ 2400



$149.95 *with cable purchase*

Your mother begged you to become an orthodontist and straighten out all those crooked teeth. But you just couldn't see spending eight hours a day with your hands in someone else's mouth. So here you are driving a truck for the highway department. Cutting weeds in the summer and filling potholes in the spring. Not to mention burning tons of leaves in the fall. Oh! Then there's winter and all that snow to plow.

Speaking of which, we just had a snowfall and another storm is on the way. And it looks like a big one. Your regular truck won't do the job. For this one you need a bulldozer. Here we go again. Put on your gloves, get your snow goggles, and grab a thermos of coffee (so you'll be able to stay awake for 24 hours straight). Now get out there and clean up those flakes! Wait a second—make sure you keep tuned to the weather station for the latest weather bulletins.

Starting your dozer

Type in LISTING 1 using the M/L Editor found elsewhere in this issue. To load the game, use the *L* option from the DOS menu.

If you wish to forgo watching the falling snow on the introduction screen (will you please watch at least once? After all, we did spend some time trying to create this special effect), press any key. After watching the scrolling weather message, press the START key to begin plowing. You may direct the dozer in any of the four directions using a joystick in port 1. No! You cannot go off the road—see the centerline markers? Follow them until the roads are completely clear of snow. If you accomplish this task there will probably be another snowfall to test your driving skills.

While you're pushing snowflakes around, storms will occasionally cross your path. If your dozer touches one, it will undoubtedly get diesel-line freeze and crash. You'll need a new one then. As time goes on the storms get more frequent and faster, so beware. Most of the streets are free of vehicles, but there are always some crazy people who venture out and risk their lives. Fortunately, a bell sound warns you that they are on the road. If you hit one, you score a 100-point bonus. These people will honk their horns at you, but I'd just ignore them and keep going. And don't forget to keep your gas tank full.

Put on your gloves, get your snow goggles, and grab a thermos of coffee—now get out there and clean up those flakes!

Remember the Knight-Rider and KIT? Me neither. But KIT had turbo boost. And so does your dozer. So once per level you can jump to the edge of the screen (assuming there's a road there; you wouldn't want to jump into a tree, would you?). If you need to clean off your windshield or take a sip of coffee during plowing, just hit any key to pause. Press a key again to continue. Well, what are you waiting for? Get going!

Technical Notes and Other Stuff

Snowplow uses the four color Antic mode 5. Antic 4 would give better resolution but would have taken much more memory. An entire screen takes 2560 bytes—roughly 2.5K or about ⅓ of the entire program. In *Snowplow* we have an entire screen, but it takes only 1/6th of the program because it is compacted. We used a compacter similar to the one in *BBK Artist* (See issue #56).

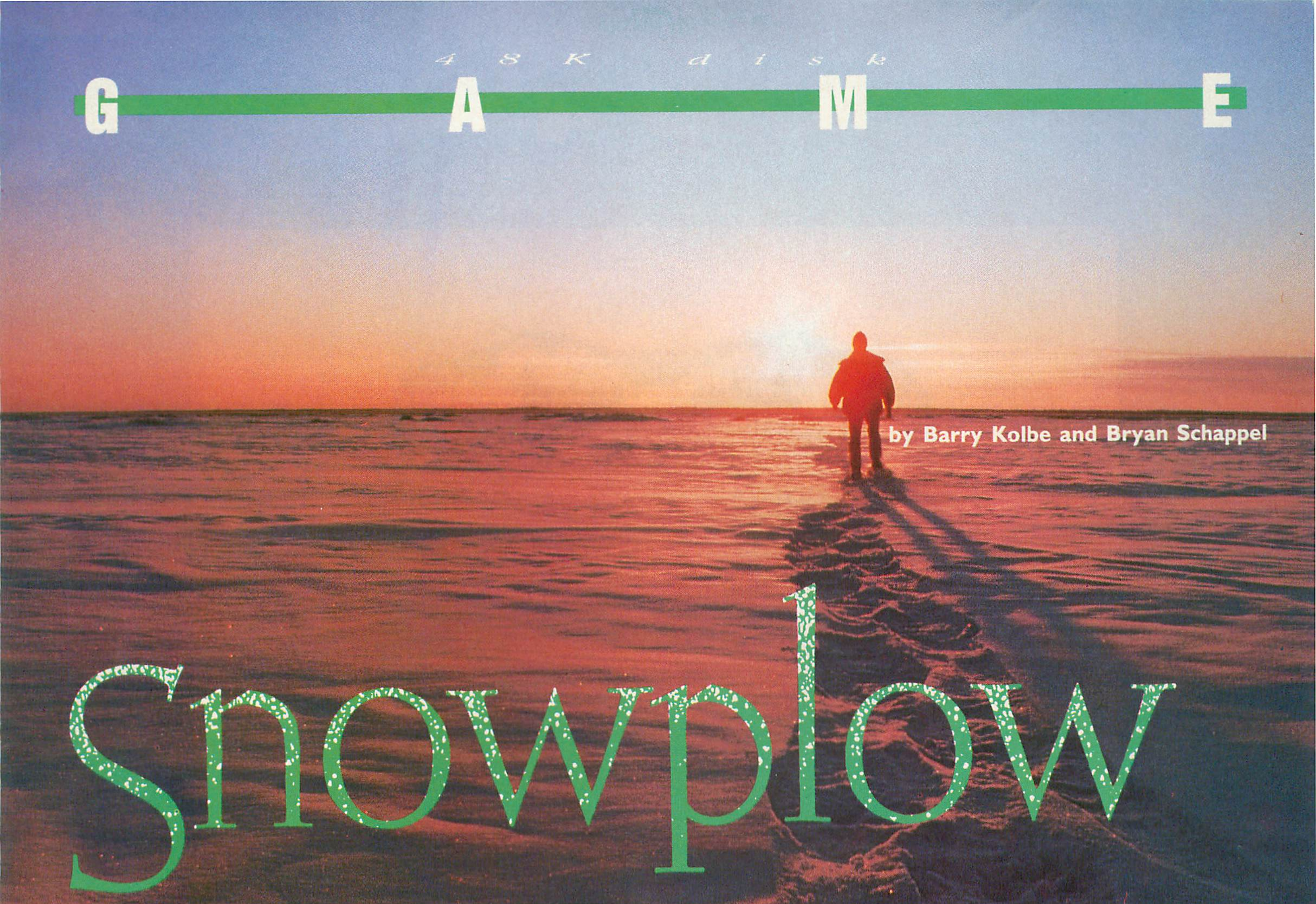
The falling letters on the intro screen are each composed of four defined graphic zero characters. Of course, there's also a complete character set in the DATA.

If you get tired of playing the same gameboard over and over, relief is in sight—a gameboard editor! That's why, when you're playing the game, the disk is being accessed. It's looking for files named SMAP??? where ??? can be anything that's legal. It will load these files in the order they appear on the disk. As you clear each board, the storm appears on screen more often and moves faster until the 6th board. So leave your drive on while playing *Snowplow*. Next month, when we present the *Snowplow Editor*, you'll be able to design your own screens. Excuse me a minute, I think I have to shovel the sidewalk. . . .

continued on page 56

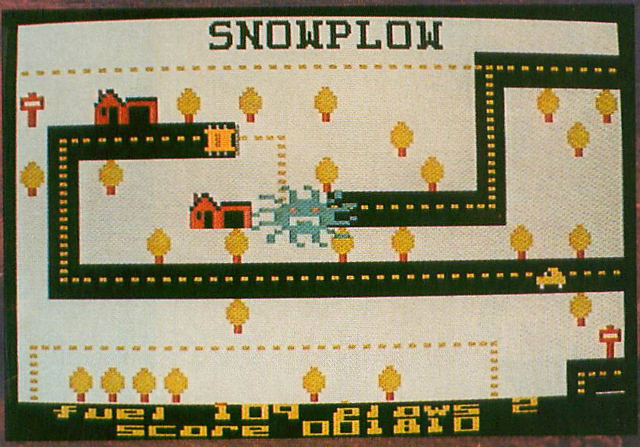
48K disk

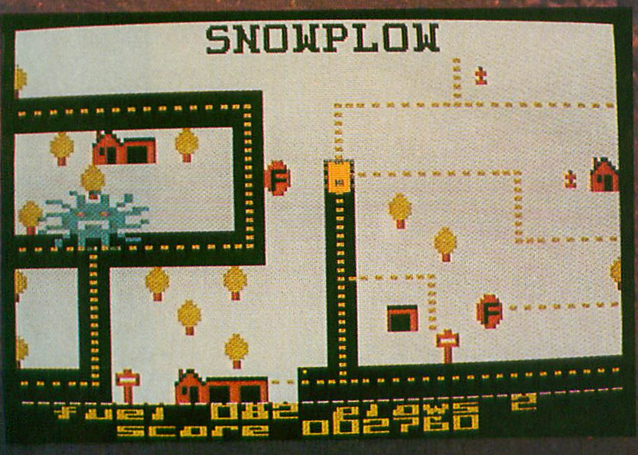
GAME



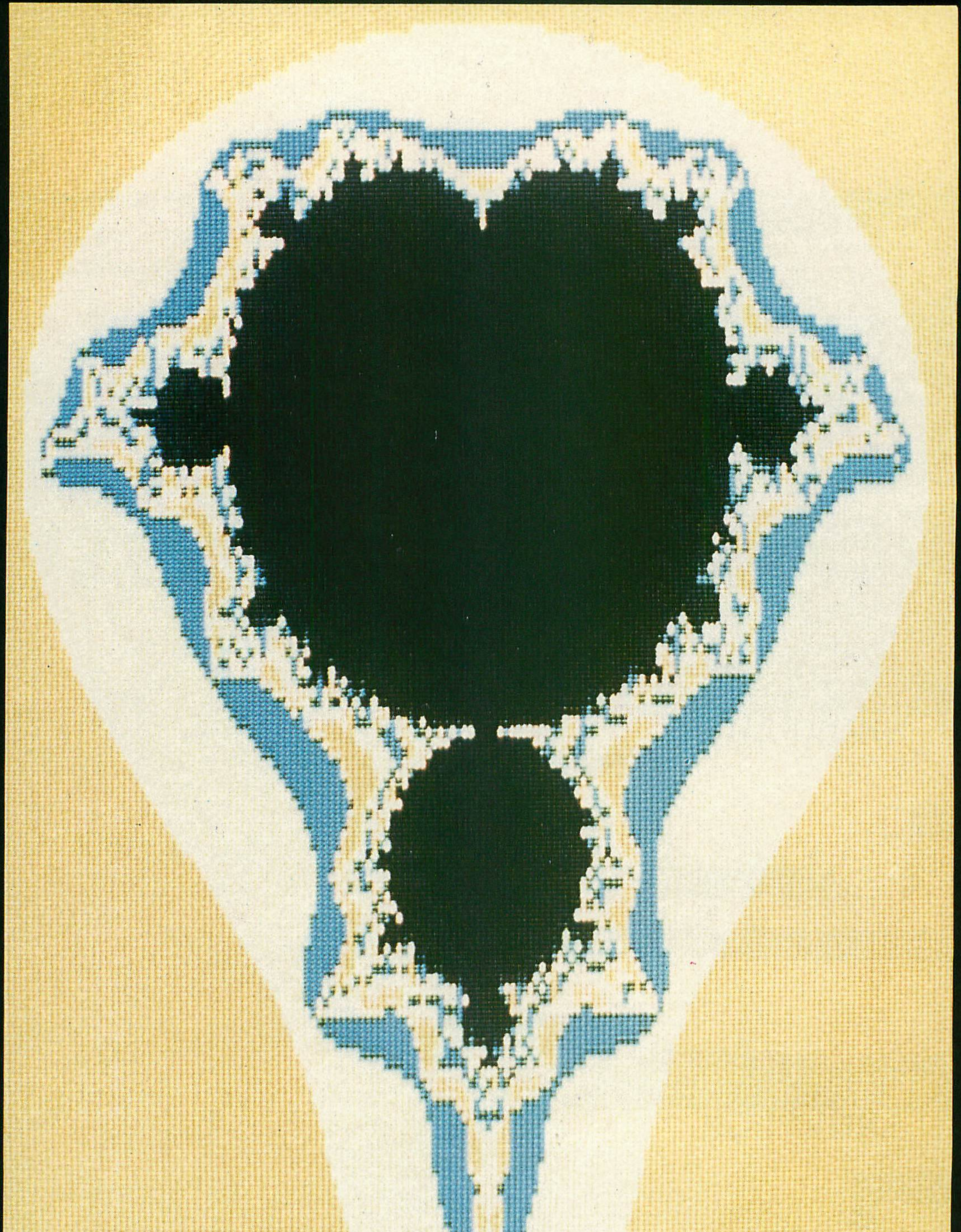
Snowplow

by Barry Kolbe and Bryan Schappel





GRAP



by James J. Greco

The Mandlebrot Set

FOR THE 8-BIT USER

In August of 1985 I received my issue of Scientific American. Inside it, in the "Computer Recreations" column by A.K. Dewdney, I found some fascinating computer-generated graphics, and the instructions for generating them. They were images of something called the Mandelbrot Set.

A brief look at the formula and method for generating these images told me that it was intended for a computer very unlike mine (an Atari 800 XL), a very expensive one.

In January of 1986 I received my issue of ANALOG Computing. To my surprise it contained a program for generating the Mandelbrot Set. There was hope my little Atari could produce these images after all! Hope was lost when I realized that the program was written for the 520ST. At that point I decided I was going to generate some of those images with a program I was going to have to write myself.

I went back to the *Scientific American* to find out exactly what the Mandelbrot Set was. If it were not for Mr. Dewdney's excellent article, I am sure I would not have ever found out. What follows is my attempt at explaining how they are generated.

The Mandelbrot Set is based on the repeated squaring of complex numbers. Complex numbers are numbers that involve the square root of -1. The square root of -1 is represented by the letter *i*, for imaginary. If *i* is the square root of -1, then *i* squared must be -1, a real number again. A complex number is a combination of the regular numbers we use every day (real numbers) and imaginary numbers. This combination is the addition of the two. Therefore $2+3i$ is a complex number, as is $.00531+213i$. As a sideline, even the real numbers are considered complex, as they can be expressed as a real number plus 0i.

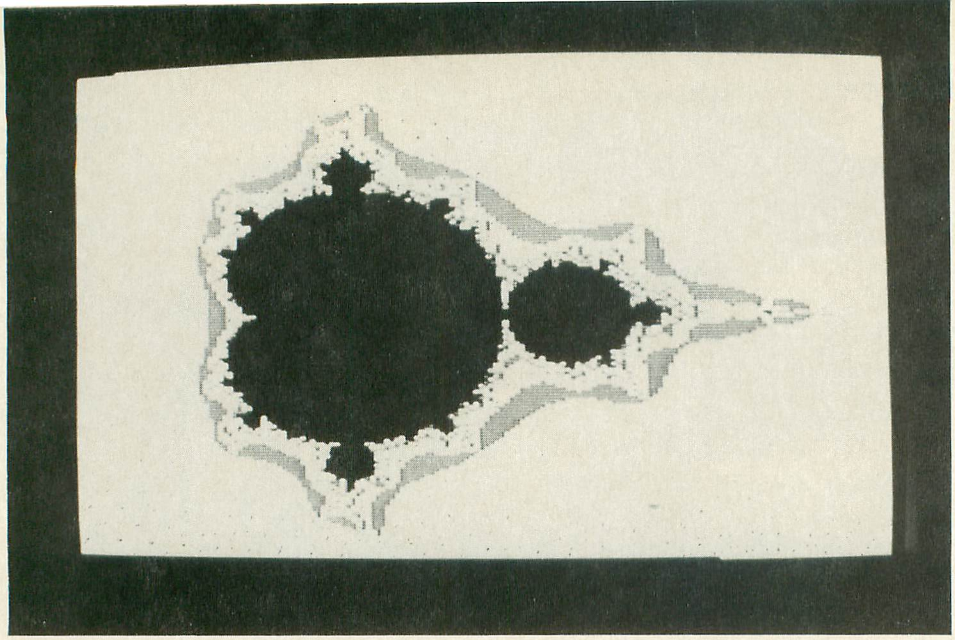
Addition of complex numbers is easy. All that needs to be done is to add the individual like parts (add the real parts together, then add the imaginary parts together), for example: $(2+3i)+(4+7i)=6+10i$.

As I mentioned above, the Mandelbrot Set is generated by the repeated squaring of complex numbers. An example will probably best serve to demonstrate the squaring of complex numbers. To square $2+3i$ we expand it to $(2+3i)*(2+3i)$. Multiplying this out we get: $4+6i+6i+9i^2$, collecting like terms we get $4+12i+9i^2$. As I mentioned before *i* squared is -1. This means that $9i^2=-9$, a real number again. Substituting -9 into the above complex number and collecting all the real parts we get: $-5+12i$.

Pairs of numbers can be plotted on a plane—this plane has two axes, a horizontal one and a vertical one. Complex numbers can also be plotted on a plane, called the complex plane. The complex plane has a real (horizontal) and an imaginary (vertical) axis.

The Mandelbrot Set is plotted on this complex plane. It seems that if certain complex numbers are continually squared, they will never exceed the size of two. The size of a complex number is the distance of that complex number from the origin. The origin is at (0,0i). The distance can be calculated by use of the Pythagorean theorem. If we take the sum of the squares of the sides, then take the square root of that, we will arrive at the size of that complex number. The numbers that never exceed the size of two are what make up the Mandelbrot Set. Most of the numbers in the complex plane quickly exceed the size of two, soon after the squaring process begins. These are the numbers that give the Mandelbrot Set its colors. The numbers that lie in the Mandelbrot Set are plotted black. The other numbers are plotted with a color representative of the number of times the squaring process is repeated before the size exceeds two.

Finding and plotting these numbers is



After selecting the desired graphics mode, you will be asked for the coordinates, size and iteration limit for the picture you wish to create. The coordinates and size can be determined from a map of the set or, as will be discussed shortly, from the program itself.

the task of this program. The monitor screen is going to represent the complex plane, with each pixel on the screen corresponding to an individual complex number. The value of the pixel is determined by two factors: 1) the center of the area that you wish to view (specified by a real and an imaginary number), and 2) the size of the area you desire to view. The latter can be likened to magnification. After these parameters are entered, the program calculates the complex values of each pixel. This is done in such a manner that the center of the screen represents the center specified by the entry of the real and the imaginary values entered. The size determines the range of values to either side (and top to bottom) of the center. Each pixel value is then processed in a manner, soon described, that calculates whether that value lies in the Mandelbrot Set (plotted black) or if that value should be plotted with a color.

The function that determines where this value lies is: $Z^2 \leftarrow Z + u$ where Z and u are both complex numbers that can be represented as follows: $Z = ar + ai$, $u = br + bi$, where *r* indicates real and *i* indicates imaginary. The values *br* and *bi* are the value of the pixel, the values *ar* and *ai* are initially set to zero. The arrow in $Z^2 \leftarrow Z + u$ indicates that the values Z and *u* are added, then squared. If the result of this squaring is less than two, the resultant real value is then put into *ar*. The resultant imaginary value is then put into *ai* and the constant values (for the pixel) of *br* and *bi* are then added to the new values of *ar* and *ai*, then squared again. This process continues until the size of Z^2 exceeds two or until we can safely assume that the value will never exceed two. This repeated performing of the same function is referred to as iteration. At the beginning of the program, you will be asked for the iteration limit. This is what determines the number of times the function will be implemented before it is assumed that a point lies in the Mandelbrot Set (assuming that the size does not first exceed two).

If the size does exceed two, the number of times it took to do so is used to determine the color that the pixel will be plotted with. The entire process is repeated for every pixel on the screen. If the area you want to look at lies entirely in the Mandelbrot Set (a very boring picture), and you choose 100 as the iteration limit in high-resolution mode, it would take 160*170*100 (2,720,000) calculations to complete the picture. This is why it takes more than a little time for this program to run.

About the program

When you first run the program, you will be asked if you want to save a screen, view a screen, complete a saved screen, see directory (disk version) or none of the above. If you're going to want to save the picture you're going to create, then select the save option. Selecting the view option allows you to view a screen that you have already saved. If, after you have started to create a screen that you are planning to save, you decide you want to turn off the computer, you can save a partial picture by pressing the HELP key. When this key is pressed, the computer dumps the current values and the portion of the screen that is completed to the I/O device (as soon as the line it is working on is completed). To complete the screen at a later time, choose the complete saved screen option from the menu. Simply pressing RETURN will allow you to create a picture, but it can not be saved. If you select the save option, you'll be prompted to insert your disk or cassette at this time. If you're using a cassette recorder, it will have to remain on until the screen is complete. Disk-drive users may remove their disk and turn off the drive; just turn on the drive and insert the disk when the screen is complete. After selecting the desired graphics mode, you will be asked for the coordinates, size and, iteration limit for the picture you wish to create. The coordinates and size can be determined from a map of the set or, as will be discussed shortly, from the program itself. A good iteration limit seems to me to be about 100; the higher the number you choose the longer the set will take to calculate, however, it will be more detailed and accurate.

After the set has been calculated, or retrieved from disk or cassette, a double cursor will appear. This cursor can be moved around by use of the arrow keys. Finer movement can be obtained by holding down the control key and the desired arrow key. After the cursor has been moved to a position of interest, holding down the RETURN key will return the screen to graphics 0, and will display the coordinates and sizes of the cursor's two squares. Following the prompts another set can be calculated in the same manner as the first. For an overall look at the Mandelbrot Set, enter the follwing: real −.75, imaginary 0 and a size of 3; the whole set will be plotted. This can be your map; just move the cursor to the area you desire to look at closer and press RETURN.

Entering the program:

For disk users simply type in the program as listed. Cassette users make the following changes.

Change Line 60 to read:

```
60 IF LOR=1 THEN FF=8:DQ$="C
:":GOTO 560
```

Change Line 150 to read:

```
150 PRINT " TO COMPLETE SAVE
D SCREEN PRESS C"
```

Change Line 220 to read:

```
220 DC$="C"
```

Change the last statement of Line 100 to read:

```
DQ$="C:"
```

Delete the following lines: 200, 240, 250, 1275, 1290, 1300, 1340, 1350, 1360, 1370, 1380, 1390, 1400.

Add the following lines:

```
230 IF FF=4 THEN PRINT:PRINT
 "CUE CASSETTE, PRESS PLAY,
THEN RETURN";:INPUT B$:GOTO
280
280 NAME$="C:":IF FF=8 THEN
PRINT "CUE TAPE, PRESS RECOR
D AND PLAY, THEN RETURN";:IN
PUT B$
285 IF LOR=1 THEN ? "WHEN CA
SSETTE STOPS LOADING SCREEN,
   REPOSITION TAPE, PRESS RE
CORD AND PLAY"
287 IF LOR=1 THEN ? "PRESS R
ETURN WHEN READY";:INPUT B$
```

A word of warning to cassette users—this program can be very frustrating. It takes a long time to save and load the screens, and you will often have errors trying to load a screen that will make hours of computing time worthless.

Some Final Words

The Mandelbrot Set has some properties that seem to me (a person with a limited math background) to be very unusual. It is infinitely complex; that is, no matter how high a magnification (small a size) one chooses, the Set remains just as complex as it is at lower magnification levels. This means that it has an infinite perimeter, yet it encloses a finite area. The set is also self similar; that is, at specific locations and levels of magnification, you will see shapes that are similar to, but not exactly like, shapes seen at other magnifications and locations.

LISTING 1: BASIC

```
QK 1 REM *******************************
WY 2 REM *      8-BIT MANDELBROT       *
XC 3 REM *             BY              *
OY 4 REM *       JAMES J. GRECO        *
GM 5 REM *                             *
OV 6 REM *       COPYRIGHT 1988        *
XG 7 REM *             BY              *
GV 8 REM *      ANALOG COMPUTING       *
QS 9 REM *******************************
LQ 10 GOTO 100
VV 20 C=C+1:X=I*I-J*J+V:U=2*I*J+Z:IF C<IL
    AND X*X+U*U<4 THEN J=U:I=X:GOTO 20
HE 30 T(G)=P(C):C=0:I=0:J=0:IF G<>Y THEN
   G=G+1:V=Q(G):GOTO 20
FN 40 FOR G=0 TO Y:COLOR T(G):PLOT G,K:NE
   XT G:G=0:K=K+1:V=Q(G):Z=R(K):IF PEEK(7
   32)=17 THEN POKE 732,0:GOTO 560
UQ 50 IF K<>F THEN 20
BL 60 IF LOR=1 THEN FF=8:DQ$="D:":DQ$(LEN
   (DQ$)+1)=NAME$:GOTO 560
YE 70 GOTO 560:REM LINE 20 CALCULATES COM
   PLEX SQUARE OPERATION, COUNTS NUMBER O
   F ITERATIONS AND, DETERMINES SIZE
DQ 80 REM LINE 30 ASSIGNES COLOR TO PIXEL
   , STORES COLOR, ZEROS VARIABLES AND, I
   NCREMENTS COLUMN NUMBER
UW 90 REM LINE 40 PLOTS COLORS WHEN MAX N
   UMBER OF COLUMNS IS REACHED AND, INCRE
   MENTS ROW NUMBER
CX 100 DIM DQ$(12),Z$(10),GR$(2),NAME$(15
   ),SM$(10),DC$(10),B$(10):FOR A=1536 TO
    1542:READ B:POKE A,B:NEXT A:DQ$="D:"
AH 110 DIM R$(20),X$(20)
PL 120 DIM NEXT$(10):PRINT "K":PRINT :PRI
   NT "       MANDELBROT SET GENERATOR":PR
   INT :PRINT :TRAP 1270:REM BY JAMES GRE
YW 130 DIM Q(200),R(200),T(200),P(200),RC
   $(20),IC$(20),S$(10),II$(10):RC$="REAL
    CENTER ":IC$="IMAGINARY CENTER "
DE 140 DIM TH$(4):TH$="THE ":S$="SIZE ":I
   I$="ENTER "
KZ 150 PRINT "TO COMPLETE A SAVED SCREEN
   PRESS C":PRINT :PRINT "TO SEE DIRECTOR
   Y PRESS D"
VH 160 PRINT :PRINT "TO SAVE SCREEN PRESS
    S":PRINT :PRINT "TO VIEW SAVED SCREEN
    PRESS V":PRINT
SJ 170 ? "FOR NONE OF THE ABOVE PRESS RET
   URN":? :? "ENTER SCREEN MODE";:INPUT S
   M$:IF SM$="S" THEN FF=8:GOTO 220
VB 180 IF SM$="V" THEN FF=4:GOTO 220
ZU 190 IF SM$="C" THEN LOR=1:SM$="V":FF=4
   :GOTO 220
OD 200 IF SM$="D" THEN GOTO 1350
FQ 210 FF=0:GOTO 300
GY 220 PRINT :PRINT "INSERT DISK, PRESS R
   ETURN WHEN READY";:INPUT B$
SW 240 DC$="D:":? "ENTER NAME";:INPUT NAM
   E$:IF SM$="S" THEN 1350
BO 250 IF SM$="S" OR SM$="V" THEN DQ$(LEN
   (DQ$)+1)=NAME$
TV 290 IF SM$="V" THEN GOTO 610
OB 300 PRINT :PRINT "FOR HIGH RESOLUTION
   GRAPHICS PRESS H"
VO 310 PRINT :PRINT "FOR MEDIUM RESOLUTIO
   N GRAPHICS PRESS M"
CU 320 PRINT "FOR LOW RESOLUTION GRAPHICS
     PRESS L"
GL 330 PRINT :PRINT "FOR VERY LOW RESOLUT
   ION PRESS V"
ST 340 PRINT :PRINT :PRINT "MAKE GRAPHICS
     SELECTION";:INPUT GR$:IF Z$="O" OR B$
   ="I" THEN GOTO 380
KL 350 PRINT :PRINT ;II$;;RC$;;:INPUT E
RW 360 PRINT ;II$;;IC$;;:INPUT H
UL 370 PRINT ;II$;;S$;;:INPUT S:IF S<=0 T
   HEN PRINT "SIZE MUST BE POSITIVE":GOTO
    370
RP 380 PRINT "ENTER ITERATION LIMIT";:INP
   UT IL
XP 381 PRINT :IF IL<10 THEN PRINT "ITERAT
   ION LIMIT MUST BE TEN OR GREATER":GOTO
    380
SV 390 REM LINES 410-450 ASSIGN SPECIC VA
   LUES TO BE USED IN CALCULATIONS DEPEND
   ING UPON GRARHICS MODE
KT 400 PRINT "PLEASE WAIT, INITIALIZING V
   ARIABLES"
JY 410 IF GR$="L" OR GR=21 THEN L=80:D=1.
   54:Y=79:W=3:F=48:GR=21:LO=150:HI=4:GOT
   O 450
CF 420 IF GR$="V" OR GR=19 THEN L=40:D=1.
   54:Y=39:W=3:F=24:GR=19:LO=168:HI=1:GOT
   O 450
EJ 430 IF GR$="H" OR GR=31 THEN L=160:D=0
   .77:Y=159:W=3:F=170:GR=31:LO=202:HI=31
   :GOTO 450
NG 440 F=170:L=80:D=0.385:Y=79:W=15:GR=11
   :LO=202:HI=31
FC 450 A=S/L:M=E-S/2:N=H+S/2.6
LN 460 REM LINE 470 CALCULATES COLUMN AND
    ROW VALUES FOR COORDS AND SIZE SELECT
   ED IT THEN STORES THEM IN ARRAYS
EU 470 FOR G=0 TO Y:Q(G)=M+G*A:NEXT G:FOR
    K=0 TO F:R(K)=N-K*A*D:NEXT K:C=0
OF 480 REM LINES 490-540 ASSIGN COLOR VAL
   UES FOR SPECIFIC COUNTS
NE 490 C=C+1:B=1:VO=C:IF C=IL THEN 530
CF 500 VO=VO-1:IF VO<=1 THEN P(C)=B:GOTO
   490
YP 510 IF B=W THEN B=0
```

```
QD 520 B=B+1:GOTO 500
EG 530 IF LOR=1 THEN P(C)=0:G=0:K=K1:V=Q(
   G):Z=R(K):C=0:I=0:J=0:GOTO 20
EK 540 P(C)=0:G=0:K=0:C=0:V=Q(G):Z=R(K):G
   RAPHICS GR:GOTO 20
PZ 550 REM LINES 560-630 ARE I/O ROUTINES
   RUN
WN 560 IF FF=0 THEN 680
XB 570 CLOSE #1:POKE 764,12:OPEN #1,8,0,D
   Q$:PUT #1,GR:PRINT #1,E:PRINT #1,H:PRI
   NT #1,S:PUT #1,IL
FB 580 PUT #1,HI:PUT #1,LO:PUT #1,G:PUT #
   1,K
QV 590 POKE 764,12:POKE 852,PEEK(88):POKE
    853,PEEK(89):POKE 856,LO:POKE 857,HI:
   POKE 850,FF+3:A=USR(1536,16):CLOSE #1
QT 600 GOTO 680
JQ 610 CLOSE #1:POKE 764,12:OPEN #1,4,0,D
   Q$:GET #1,GR:INPUT #1,E:INPUT #1,H:INP
   UT #1,S:GET #1,IL:GET #1,HI
QZ 620 GET #1,LO:GET #1,G1:GET #1,K1:GRAP
   HICS GR:SETCOLOR 4,0,2
QK 630 POKE 764,12:POKE 852,PEEK(88):POKE
    853,PEEK(89):POKE 856,LO:POKE 857,HI:
   POKE 850,FF+3:A=USR(1536,16):CLOSE #1
RF 640 IF LOR=1 THEN GOTO 410
RD 650 GOTO 680
JT 670 DATA 104,104,104,170,76,86,228
WG 680 SOUND 0,0,0,0
AL 690 POKE 53774,247:REM STARTS PM
HS 700 DATA 255, 129,129,129,129,129,129,
   129,189,165,165,165,165,165,165,165,16
   5,165,165,165
ZC 710 DATA 165,165,165,189,129,129,129,1
   29,129,129,129,129,129,255
JO 720 DATA 104,160,1,177,203,136,145,203
   ,196,205,200,200,144,245,198,203,96,10
   4,164,205,177
PU 730 DATA 203,200,145,203,136,136,16,24
   7,230,203,96
ZU 740 A=PEEK(106)-48:POKE 106,A:POKE 542
   79,A:ST=256*A:POKE 559,62:POKE 53277,3
   :POKE 53256,3
HU 750 FOR I=ST+1024 TO ST+1280:POKE I,0:
   NEXT I:PST=ST+1025:POKE 204,INT(PST/25
   6):POKE 203,PST-(PEEK(204)*256)-1
IC 760 POKE 205,34:RESTORE 700:FOR I=PST
   TO PST+33:READ A:POKE I,A:NEXT I:FOR I
   =1550 TO 1581:READ J:POKE I,J:NEXT I
PL 770 FOR Z=1 TO 30:GOSUB 890:NEXT Z:I=3
   8:POKE 764,7:VR=5
VI 780 CU=PEEK(764):REM LINES 780-930 ARE
    FOR CURSOR MOVEMENT 930, IS END  OF P
   M
TD 790 IF CU=7 THEN I=I+10:POKE 53248,I:I
   F I>=240 THEN I=10
TS 800 IF CU=135 THEN I=I+1:POKE 53248,I:
   IF I>=240 THEN I=10
FL 810 IF CU=6 THEN I=I-10:POKE 53248,I:I
   F I<=10 THEN I=200
GP 820 IF CU=134 THEN I=I-1:POKE 53248,I:
   IF I<=10 THEN I=200
MP 830 IF CU=14 THEN FOR GG=1 TO 20:GOSUB
    910:NEXT GG
KH 840 IF CU=142 THEN GOSUB 910
YH 850 IF CU=15 THEN FOR GG=1 TO 20:GOSUB
    890:NEXT GG
SG 860 IF CU=143 THEN GOSUB 890
CC 870 IF CU=12 THEN POKE 559,0:GOTO 930
YO 880 POKE 764,32:GOTO 780
HQ 890 IF VR>=180 THEN RETURN
NW 900 VR=VR+1:A=USR(1567):POKE 704,200:R
   ETURN
BL 910 IF VR<=-20 THEN RETURN
IB 920 VR=VR-1:A=USR(1550):POKE 704,200:R
   ETURN
YV 930 POKE 53248,0:POKE 106,PEEK(106)+48
   :REM FOLLOWING LINES DET. COORDS AND S
   TARTING OF ANOTHER SET
KE 940 IF GR=11 THEN G=INT(I/2)-16:K=VR+1
   2:JJ=1:D=0.385:L=80
UX 950 IF GR=19 THEN G=INT(((I/2)-16)/2):
   K=INT((VR/4+2)/2):JJ=0.42:D=1.54:L=40
GD 960 IF GR=21 THEN G=INT(I/2)-16:K=VR/4
   +2:JJ=1:D=1.54:L=80
EA 970 IF GR=31 THEN G=2*(INT(I/2)-22)+12
   :K=VR+12:JJ=2:D=0.77:L=160
FN 980 A=S/L:M=E-S/2:N=H+S/2.6
HT 990 E=M+G*A:H=N-K*A*D
```

```
MO 1000 PRINT "↰":PRINT "THE REAL CENTER
   OF THE CURSER IS ";E
WE 1010 PRINT :PRINT "THE IMAGINARY CENTE
   R IS ";H
IY 1020 PRINT :PRINT "THE SIZE OF THE INN
   ER CURSOR IS ";:SI=5*A*JJ:PRINT SI
YO 1030 PRINT :PRINT "THE SIZE OF THE OUT
   ER CURSOR IS ";:SO=12*A*JJ:PRINT SO
RM 1040 PRINT :PRINT "DO YOU WANT TO CREA
   TE A NEW SET Y OR N";:POKE 764,28:INPU
   T Z$
RK 1050 IF Z$="Y" THEN GOTO 1150
PR 1060 GOTO 1230
AL 1070 PRINT "WHICH PARAMETER(S) DO YOU
   WISH TO      CHANGE R, I, S":INPUT Z$
JV 1080 IF Z$="R" THEN PRINT ;II$;;RC$;;:
   INPUT E:GOTO 1120
ZF 1090 IF Z$="I" THEN PRINT ;II$;;IC$;;:
   INPUT H:GOTO 1120
GG 1100 IF Z$="S" THEN PRINT ;II$;;S$;;:I
   NPUT S:IF S<=0 THEN PRINT "SIZE MUST B
   E GREATER THAN ZERO":GOTO 1100
OF 1110 GOTO 1120
AN 1120 PRINT ;TH$;;RC$;;E;:PRINT :PRINT
   :PRINT ;TH$;;IC$;;H;:PRINT :PRINT :PRI
   NT ;TH$;S$;;S;
IS 1121 PRINT :PRINT :PRINT "DO YOU WISH
   TO CHANGE ANY MORE Y OR N";:INPUT B$
KB 1130 IF B$="Y" THEN GOTO 1070
ON 1140 GOTO 1210
FX 1150 PRINT :PRINT :PRINT "SELECT SIZE
   VALUE:":PRINT
OE 1160 PRINT "INNER CURSORS SIZE VALUE":
   PRINT :PRINT "OUTER CURSORS SIZE VALUE
   "
FC 1161 PRINT :PRINT "YOUR OWN SIZE":PRIN
   T :PRINT :PRINT "SELECT I, O OR S";
UB 1170 INPUT B$:IF B$="S" THEN Z$="S":GO
   TO 1100
KC 1180 IF B$="O" THEN S=SO:GOTO 1120
BR 1190 IF B$="I" THEN S=SI:GOTO 1120
NE 1200 GOTO 1100
EK 1210 Z$="O":PRINT "TO SAVE SCREEN PRES
   S S OTHERWISE         RETURN":INPUT SM$:I
   F SM$="S" THEN FF=8:GOTO 220
TT 1220 FF=0:B$="O":GRAPHICS 11:GOTO 300
RS 1230 PRINT :PRINT "DO YOU WANT TO QUIT
   , GO BACK TO          SELECTED VALUES, OR
    START A NEW          SET Q, G, OR S";
OP 1231 INPUT Z$
UQ 1240 IF Z$="Q" THEN END
EE 1250 IF Z$="S" THEN CLR :GOTO 10
NK 1260 GOTO 1000
KC 1265 REM LINES 1270-1330 ARE ERROR HAN
   DLING/DETECTING ROUTINES
ND 1270 LINERR=PEEK(187)*256+PEEK(186):ER
   R=PEEK(195):TRAP 40000:TRAP 1270
XM 1275 IF ERR=165 THEN ? "BAD FILE NAME,
    TRY AGAIN":POKE 764,28:GOTO 240
DI 1280 IF ERR=139 OR ERR=140 OR ERR=142
   OR ERR=143 OR ERR=163 OR ERR=136 THEN
   PRINT "I/O ERROR":END
SW 1285 IF ERR=138 OR ERR=144 THEN GOTO L
   INERR
EX 1290 IF ERR=145 OR ERR=160 THEN PRINT
   "DISK ERROR":END
QH 1300 IF ERR=170 THEN PRINT "FILE NOT O
   N DISK, TRY AGAIN":POKE 764,28:DQ$="D:
   ":GOTO 240
MD 1310 IF ERR=8 THEN FOR AA=1 TO 50:SOUN
   D 0,100,10,15:NEXT AA
EQ 1320 SOUND 0,0,0,0:PRINT :PRINT "*****
   *REENTER******":PRINT :GOTO LINERR
XY 1330 PRINT "ERROR NUMBER ";ERR;" AT LI
   NE ";LINERR;:END
WU 1340 REM LINES 1350-1400 ARE USED WITH
    I/O ROUTINES ABOVE
PK 1350 CLOSE #4:OPEN #4,6,0,"D:*.*":IF S
   M$<>"S" THEN GOTO 1390
HH 1360 INPUT #4,R$:TRAP 1400:X$=R$(3,LEN
   (NAME$)+2):IF X$<>NAME$ THEN IF R$(10,
   16)<>"SECTORS" THEN GOTO 1360
RB 1370 IF X$=NAME$ THEN PRINT "NAME ALRE
   ADY USED ON DISK, TRY AGAIN":TRAP 4000
   0:TRAP 1270:GOTO 240
TG 1380 TRAP 40000:TRAP 1270:GOTO 250
OF 1390 INPUT #4,R$:TRAP 1400:PRINT R$:IF
    R$(10,16)<>"SECTORS" THEN 1390
OE 1400 CLOSE #4
```

Joytype

by John Pilge

Joytype is a replacement for the typewriter and is not a word processor. It is designed for people who cannot use a typewriter because of limited movement caused by stroke, cerebral palsy or injury. While you may not have a need to use it yourself, you may want to type it in for a friend or charity.

You can also use some of the routines for later programs of your own (directory, printing, scrolling, etc.).

The first step is to get a joystick the handicapped can use. Many prefer joysticks by Wico.

Instructions

The first menu lets you choose the drive on which you will store files. Should you change your mind after making this selection, you return to this menu when you choose to erase a letter from the option menu.

Typing is simple. Move the joystick (or trackball) to the letter of choice. Select the letter for printing by pressing the firebutton. Each space between the letters on the header can be used as a space (just like pressing the spacebar on a typewriter.)

Special functions are on the bottom right of the header. Selecting either *D* or *L* is delete. *C* or *R* is a carriage return. *E* is an escape character. *O*, *P* or *T* changes the screen to the option menu. The triangle is a backspace. Backspace would usually be used for forming special characters (using "c" and "/" to make a cents marker) or underlining (if you don't know the escape commands for your printer).

The option menu has functions to save a file, delete a file, load a file, print, type (returns you to the typing screen), erase the current letter and format a disk.

To avoid mistakes, "FORMAT" doesn't work. If the person who uses the program has enough control and confidence to risk using FORMAT, merely delete Line 810 of the program, and FORMAT will work.

The first command of the Option menu is "O.NOTHING." This is to prevent an accident in case the user still has the firebutton pressed at the time the screen is displayed.

While directory will show all the files on disk, you cannot access any files with an extension (or any filename with a period). This is to prevent accidental deletion or loading of the DOS.SYS program or the JOYTYPE.BAS program (and the AUTORUN.SYS, if you have the program autoloaded).

To select files for loading, savings, etc., a menu appears with the alphabet (uppercase only), a space on either side of the alphabet, an *M* to return to the Option menu and a *DL* to delete errors.

Files can have up to eight letters in the filename. If a filename has less than eight letters, use spaces to make eight characters. A filename cannot start with a space. After eight characters have been selected, the file is acted on in accordance with your choice.

Once the user has worked with the program, it may seem to be too slow. Line 1820 has the variable DLAY that controls the speed of some movements. You can change that number to as low as one or as high (slow) as needed.

The left margin for printing is set on Line 1820 as the variable MRG$. You can set this in accordance to the needs of your printer. The printer line length is set by PLL on Line 1820.

Joytype files can be checked with most spelling checker programs and can be read by most other word processors. There is a difference between return markers, but not text.

Who gets the program

Just because you spent all night typing the program doesn't mean everyone will want it. You needed patience to type in the program, and you are going to need more patience finding someone to use it.

You will find a lot of charities unable to use the program or willing to make it available to people who need it.

Some charities are only able to deal with donations of money. Some only handle specific needs (such as raising money). Some can only accept programs on Apple or IBM no matter how inexpensive an Atari costs. You will learn a lot about the handicapped services of your community by trying to pass along this program. I have a letter from one local "charity" that doesn't want the program unless it is protected, and they can sell it.

But at least you can make it known that the programs exists and is available to work on a sturdy and inexpensive Atari.

You might be called on to modify the program. Larger characters on the screen is a popular request. There is an easier fix than to rewrite the program. The characters get larger as the TV screen gets larger.

If more than one letter appears as the fire button is pressed, try another joystick. Some joysticks have a rapid-fire ability that speeds up the response of the button. You could also move the button commands (STRIG) after the DLAY loops or make the DLAY value a higher number (slower). For any additions larger than 4K you may have to shorten the string length of LINE$, or you run out of memory.

I would like to thank Jimmy Montoya, Jr. (know locally as the "Wizard of OS") for his suggestions, and the Cabrillo College Stroke Center for their encouragement.

John Pilge is your typical, fun-loving Atari computer owner who gets a thrill finding new uses for the Atari. He is known in Santa Cruz County as J.P. or Bladerunner.

joytype

LISTING 1: BASIC

```
QJ 0 REM *****************************
UL 1 REM *        JOYTYPE            *
CL 2 REM *      BY JOHN PILGE        *
GK 3 REM *                           *
ZZ 4 REM *   PUBLISHED IN ANALOG     *
SU 5 REM *   COMPUTING, SEPT. 88     *
QP 6 REM *****************************
GT 10 POKE 566,PEEK(566)+12
AP 20 REM LINE 1820 HAS DELAY, MARGIN AND
    PRINTER LINE LENGTH VARIABLES.
GL 40 REM POKES:77 IS ATTRACT MODE,84-85
   IS CURSOR POSITION,88-89 IS SCREEN MEM
   ORY
RS 50 TRAP 1930:GOTO 1800
XG 60 X=129:SCREEN=PEEK(88)+PEEK(89)*256
SA 70 FOR I=2 TO 38 STEP 2:POKE SCREEN+I,
   X:POKE SCREEN+(I+1),128:X=X+1:NEXT I
NR 80 FOR I=42 TO 65 STEP 2:POKE SCREEN+I
   ,X:POKE SCREEN+(I+1),128:X=X+1:NEXT I
UG 90 X=X+1:FOR I=66 TO 79 STEP 2:POKE SC
   REEN+I,X:POKE SCREEN+(I+1),128:X=X+1:N
   EXT I
LL 100 FOR I=82 TO 119 STEP 2:POKE SCREEN
   +I,X:POKE SCREEN+(I+1),128:X=X+1:NEXT
   I
IU 110 X=187:FOR I=122 TO 128 STEP 2:POKE
    SCREEN+I,X:POKE SCREEN+I+1,128:X=X+1:
   NEXT I
FJ 120 POKE SCREEN+130,191:POKE SCREEN+13
   1,128
PI 130 X=225:FOR I=132 TO 158 STEP 2:POKE
    SCREEN+I,X:POKE SCREEN+(I+1),128:X=X+
   1:NEXT I
MO 140 FOR I=162 TO 184 STEP 2:POKE SCREE
   N+I,X:POKE SCREEN+(I+1),128:X=X+1:NEXT
    I
DQ 150 POKE SCREEN+186,37
ZZ 160 POKE SCREEN+187,128:POKE SCREEN+18
   8,47:POKE SCREEN+189,48:POKE SCREEN+19
   0,52:POKE SCREEN+191,128
LH 170 POKE SCREEN+192,36:POKE SCREEN+193
   ,44:POKE SCREEN+194,128
FY 180 POKE SCREEN+195,128:POKE SCREEN+19
   6,126:POKE SCREEN+197,128
ME 190 POKE SCREEN+198,35:POKE SCREEN+199
   ,50
HD 200 POKE SCREEN+2,1:SPT=2:F=129
UA 210 FOR I=1 TO 4:POSITION 2,7:PRINT CH
   R$(156);:NEXT I:RETURN
EE 220 X=STICK(0):Y=STRIG(0):IF Y=0 THEN
   GOSUB 490
ZE 230 FOR SLOW=1 TO DLAY:NEXT SLOW
XZ 240 IF X=14 AND (SPT-40)>1 THEN GOSUB
   300
SQ 250 IF X=13 AND (SPT+40)<200 THEN GOSU
   B 340
HQ 260 IF X=7 THEN GOSUB 430
NY 270 IF X=11 THEN GOSUB 380
ND 280 GOTO 220
LP 290 R=R-3:POSITION 2,7:? CHR$(156);:PO
   SITION 2,R:RETURN
DM 300 L=PEEK(SCREEN+(SPT-40)):POKE SCREE
   N+SPT,F
AH 310 IF L>127 THEN POKE SCREEN+(SPT-40)
   ,L-128
WZ 320 IF L<128 THEN POKE SCREEN+(SPT-40)
   ,L+128
DA 330 SPT=SPT-40:F=L:RETURN
CA 340 L=PEEK(SCREEN+(SPT+40)):POKE SCREE
   N+SPT,F
XX 350 IF L>127 THEN POKE SCREEN+(SPT+40)
   ,L-128
UP 360 IF L<128 THEN POKE SCREEN+(SPT+40)
   ,L+128
CK 370 SPT=SPT+40:F=L:RETURN
QG 380 CNG=1:IF SPT=2 OR SPT=42 OR SPT=82
    OR SPT=122 OR SPT=162 THEN CNG=-37
JK 390 POKE SCREEN+SPT,F:L=PEEK(SCREEN+(S
   PT-CNG))
ZJ 400 IF L>127 THEN POKE SCREEN+(SPT-CNG
   ),L-128
VZ 410 IF L<128 THEN POKE SCREEN+(SPT-CNG
   ),L+128
GQ 420 F=L:SPT=SPT-CNG:RETURN
NW 439 CNG=1:IF SPT=39 OR SPT=79 OR SPT=1
   19 OR SPT=159 OR SPT=199 THEN CNG=-37
FX 440 POKE SCREEN+SPT,F:L=PEEK(SCREEN+(S
   PT+CNG))
XB 450 IF L>127 THEN POKE SCREEN+(SPT+CNG
   ),L-128
TR 460 IF L<128 THEN POKE SCREEN+(SPT+CNG
   ),L+128
FU 470 F=L:SPT=SPT+CNG:RETURN
JX 480 Z=PEEK(85):FOR SPC=Z TO 39:PRINT "
    ";:NEXT SPC:RETURN
QU 490 A=PEEK(SCREEN+SPT):POKE 77,0:IF A=
   164 OR A=172 THEN GOTO 620
CQ 500 C=PEEK(83):R=PEEK(84):IF A=175 OR
   A=176 OR A=180 THEN POP :? CHR$(125):G
   OTO 710
KM 510 IF A=254 THEN A=194
GR 520 IF A<64 THEN A=A+32
ZV 530 IF A=163 OR A=178 THEN A=5
QI 540 IF A=165 THEN A=27
BY 550 SP=SP+1
VF 560 IF PEEK(84)=23 THEN GOSUB 290
OV 570 PRINT CHR$(A);:IF SP<1 THEN SP=1
ER 580 IF A=5 THEN GOSUB 480
PS 590 IF A=27 THEN PRINT CHR$(27);
PA 600 LINE$(SP,SP)=CHR$(A):R=PEEK(84):C=
   PEEK(85)
ZE 610 RETURN
IK 620 IF LINE$(SP,SP)=CHR$(5) THEN GOSUB
    660
ZH 630 A=32:LINE$(SP,SP)=CHR$(163):SP=SP-
   1:IF SP<1 THEN SP=1
YH 640 C=C-1:IF C<2 THEN C=39:R=R-1:IF R<
   6 THEN R=6:C=2
HZ 650 POSITION C,R:PRINT " ";CHR$(30);:G
   OTO 610
ZT 660 R=R-1:IF R<6 THEN R=R+1:RETURN
XS 670 POSITION 2,R:FOR I=39 TO 2 STEP -1
YY 680 PRINT CHR$(30);:IF PEEK(93)=69 THE
   N POP :C=I+1:RETURN
GO 690 NEXT I
KC 700 RETURN :REM ERROR TRAP
DZ 710 REM MENU FOR FUNCTIONS
PX 720 POSITION 2,14
FB 730 PRINT ,,"0. NOTHING":? ,,"1. SAVE
   FILE":? ,,"2. PRINT IT":? ,,"3. LOAD F
   ILE":? ,,"4. DIRECTORY":? ,,"5. TYPE"
FD 740 PRINT ,,"6. DELETE FILE":? ,,"7. F
   ORMAT DISK":? ,,"8. ERASE LETTER":POSI
   TION 22,13:PRINT CHR$(29);
YH 750 X=STICK(0):Y=STRIG(0):ROW=PEEK(84)
TY 760 IF Y=0 THEN GOTO 800
HW 770 IF X=13 AND ROW<22 THEN PRINT CHR$
   (29);
JF 780 IF X=14 AND ROW>14 THEN PRINT CHR$
   (28);
HW 790 FOR SLOW=1 TO DLAY:NEXT SLOW:GOTO
   750
IR 800 GET #6,A:POKE 77,0:POKE 85,22:A=A-
   175
SW 810 IF A=8 THEN GOTO 750
IE 820 ON A GOTO 750,1050,1120,1300,1370,
   1450,1550,1590,1810
QE 830 GOTO 750
PR 840 SX=1:DR$="          ":DI$(4,13)=DR$:P
   RINT CHR$(125):PRINT "WHAT IS THE NAME
    OF FILE IN DRIVE ";DI$(2,2)
VE 850 PRINT :PRINT :PRINT :PRINT "M";CHR
   $(160);:FOR I=193 TO 218:PRINT CHR$(I)
   ;:NEXT I
QZ 860 PRINT CHR$(160);CHR$(160);"DL":? "
   E":? "N":? "U"
HZ 870 POSITION 2,10:PRINT DI$
AY 880 COL=3
```

```
OT 890 POSITION COL-2,5:PRINT CHR$(31);CH
   R$(31);:FOR SLOW=1 TO DLAY:NEXT SLOW
XO 900 X=STICK(0):Y=STRIG(0):COL=PEEK(85)
ZU 910 IF Y=0 THEN GOTO 960
PD 920 IF X=7 AND COL<32 THEN PRINT CHR$(
   31);
PV 930 IF X=11 AND COL>2 THEN PRINT CHR$(
   30);
ZN 940 FOR SLOW=1 TO DLAY:NEXT SLOW
PA 950 GOTO 900
NZ 960 GET #6,A:IF A=196 THEN GOTO 1020
UT 970 IF A=205 THEN GOTO 710
WM 980 IF A=32 AND SX=1 THEN GOTO 890
FS 990 IF SX>8 THEN GOTO 1040
TQ 1000 DR$(SX,SX)=CHR$(A):POSITION 5,10:
   PRINT DR$:? SX:SX=SX+1
TY 1010 GOTO 890
IR 1020 SX=SX-1:IF SX=0 THEN SX=1
GR 1030 DR$(SX,SX)=" ":POSITION 5,10:? DR
   $;" ":? SX:GOTO 890
AY 1040 DI$(4,11)=DR$:RETURN
TB 1050 GOSUB 840:OPEN #2,8,0,DI$:FOR I=1
    TO LEN(LINE$):X$=LINE$(I,I)
EJ 1060 IF X$=CHR$(163) THEN POP :GOTO 10
   80
AS 1070 PRINT #2;X$:NEXT I
NV 1080 PRINT #2;CHR$(163)
UG 1090 CLOSE #2:PRINT CHR$(125):? ,DI$;"
   IS SAVED"
ZZ 1100 GOSUB 1610
PX 1110 GOTO 710
XQ 1120 PRINT CHR$(125):PRINT "SINGLE SPA
   CED OR DOUBLE SPACED?":GOSUB 1870
WG 1130 IF A=49 THEN LS=2
RU 1140 IF A=50 THEN LS=1
ZM 1150 OPEN #2,8,0,"P:":PRINT CHR$(125):
   ? ,,"WAIT":PRINT #2
MW 1160 FOR I=1 TO LEN(LINE$)
FC 1170 IF LINE$(I,I)=CHR$(163) THEN POP
   :GOTO 1320
ER 1180 FIN=I
YL 1190 NEXT I:PRINT ,,"PRINTING"
AI 1200 LL=PLL:SP=1:B=1:Y=0:PRINT #2;MRG$
   ;
YN 1210 Y=Y+1:IF Y>LL OR Y=LL THEN GOSUB
   1640
ZK 1220 IF SP=FIN+1 THEN GOSUB 1770:CLOSE
    #2:GOTO 710
TU 1230 IF P>53 THEN GOSUB 1700
XJ 1240 IF LINE$(SP,SP)=CHR$(32) AND Y=1
   THEN SP=SP+1:GOTO 1210
QA 1250 IF LINE$(SP,SP)=CHR$(5) THEN GOSU
   B 1720:GOTO 1210
SM 1260 IF LINE$(SP,SP)=CHR$(27) THEN PLL
   =LL+2:SP=SP+1:GOTO 1210
MP 1270 IF LINE$(SP,SP)=CHR$(126) THEN LL
   =LL+2:GOTO 1210
CQ 1280 IF LINE$(SP,SP)=CHR$(32) THEN PRI
   NT #2;LINE$(B,SP);:SP=SP+1:B=SP:GOTO 1
   210
GK 1290 SP=SP+1:GOTO 1210
NY 1300 GOSUB 840:LINE$=" ":SP=1:PRINT CH
   R$(125)
JA 1310 OPEN #2,4,0,DI$
NG 1320 INPUT #2,X$:IF X$=CHR$(163) THEN
   SP=SP-1:GOTO 1340
YV 1330 LINE$(SP,SP)=X$:SP=SP+1:GOTO 1320
AW 1340 CLOSE #2:PRINT CHR$(125):PRINT :P
   RINT DI$;" LOADED":CLOSE #2
QN 1350 GOTO 710
TW 1360 CLOSE #2:PRINT "ERROR--NO SUCH FI
   LE":FOR I=1 TO 200:NEXT I
MQ 1370 DB$(2,2)=DI$(2,2):PRINT CHR$(125)
   :OPEN #1,6,0,DB$
AK 1380 INPUT #1;F$:IF ASC(F$(3,3))<65 TH
   EN 1430
PE 1390 PRINT F$(3,13);MRG$;
UW 1400 INPUT #1;F$:IF ASC(F$(3,3))<65 TH
   EN GOTO 1430
WP 1410 PRINT F$(3,13)
SM 1420 GOTO 1380
BT 1430 CLOSE #1:PRINT :GOSUB 1610
QM 1440 GOTO 710
BT 1450 ? CHR$(125):POSITION 2,7:PRINT
MP 1460 GOSUB 60:FIN=LEN(LINE$):IF FIN<1
   THEN GOTO 220
NV 1470 FOR I=1 TO FIN
KR 1480 IF LINE$(I,I)=CHR$(163) THEN R=PE
   EK(84):C=PEEK(85):POP :GOTO 220
```

```
QE 1490 IF LINE$(I,I)=CHR$(27) THEN PRINT
    CHR$(197);:GOTO 1530
GR 1500 IF LINE$(I,I)=CHR$(126) THEN PRIN
   T CHR$(194);:GOTO 1530
VN 1510 IF LINE$(I,I)=CHR$(5) THEN PRINT
   CHR$(5);:GOSUB 480:GOTO 1530
MS 1520 PRINT LINE$(I,I);
AN 1530 R=PEEK(84):C=PEEK(85):IF R=23 THE
   N GOSUB 290
LM 1540 NEXT I:SP=I-1:GOTO 220
AW 1550 GOSUB 840
AY 1560 XIO 33,#1,0,0,DI$:PRINT CHR$(125)
   :? ,DI$;" IS GONE"
BC 1570 GOSUB 1610
RA 1580 GOTO 710
BC 1590 PRINT CHR$(125);"NOW FORMATING DR
   IVE ";DI$(2,2):XIO 254,#1,0,0,DI$
GF 1600 PRINT "DISK IS NOW FORMATTED":GOS
   UB 1610:GOTO 710
IA 1610 PRINT ,"PRESS FOR MENU"
KB 1620 Y=STRIG(0):IF Y<>0 THEN GOTO 1620
AU 1630 RETURN
HP 1640 IF LINE$(SP+1,SP+1)=CHR$(32) THEN
    PRINT #2;LINE$(B,SP):SP=SP+2:B=SP:IF
   Y=LL THEN GOTO 1670
UF 1650 IF (SP-B)>40 THEN PRINT #2;LINE$(
   B,SP):IF LS=2 THEN PRINT #2:GOTO 1680
UV 1660 PRINT #2
YZ 1670 IF LS=2 THEN PRINT #2
CF 1680 Y=1:LL=PLL:PRINT #2;MRG$;
BM 1690 RETURN
IF 1700 FOR I=1 TO 12:PRINT #2:NEXT I
MJ 1710 PRINT #2;MRG$;:P=0:RETURN
MA 1720 IF B=SP THEN PRINT #2:P=P+1
GB 1730 IF B<SP THEN PRINT #2;LINE$(B,SP-
   1):P=P+1
BA 1740 IF LS=2 THEN PRINT #2:P=P+1
LE 1750 SP=SP+1:B=SP:Y=0:LL=PLL:PRINT #2;
   MRG$;
BF 1760 RETURN
EX 1770 IF B=SP THEN GOTO 1790
ZN 1780 PRINT #2;LINE$(B,SP-1);CHR$(155)
YT 1790 CLOSE #2:SP=1:RETURN
XW 1800 OPEN #6,4,0,"S:":SETCOLOR 2,0,0
PP 1810 CLR :DIM LINE$(19955),X$(1),F$(15
   ),DR$(8),DB$(6),DI$(13):SP=0:R=6:C=2:D
   I$="D :":DB$="D :*.*"
MR 1820 DLAY=10:DIM MRG$(7):MRG$="
   ":PLL=64
XB 1830 ? CHR$(125):PRINT "STORE MESSAGES
    TO DRIVE ONE OR TWO?":GOSUB 1870
LP 1840 IF A=49 THEN DI$(2,2)="2"
GW 1850 IF A=50 THEN DI$(2,2)="1"
UJ 1860 PRINT CHR$(125):POSITION 2,7:PRIN
   T :GOSUB 60:GOTO 220
MH 1870 PRINT ,,"1":PRINT ,,"2":POSITION
   22,2:PRINT CHR$(29);:FOR I=1 TO 100:NE
   XT I
ZG 1880 X=STICK(0):Y=STRIG(0):R=PEEK(84):
   C=PEEK(85)
PL 1890 IF R>2 AND X=13 THEN PRINT CHR$(2
   8);
QK 1900 IF R<3 AND X=14 THEN PRINT CHR$(2
   9);
OG 1910 IF Y<>0 THEN GOTO 1880
VH 1920 GET #6,A:RETURN
QG 1930 POP :OOPS=PEEK(195):IF OOPS=138 T
   HEN PRINT "CHECK PRINTER OR DRIVE":GOT
   O 710
VY 1940 IF OOPS=139 THEN PRINT "FAULTY DR
   IVE?":CLOSE #2:GOTO 710
WL 1950 IF OOPS=5 AND SP<2 THEN PRINT "NO
   THING WRITTEN.":GOTO 710
DO 1960 IF OOPS=5 THEN PRINT "TOO MANY CH
   ARACTERS. SUGGEST SAVE":GOTO 710
DI 1970 IF OOPS=144 THEN PRINT "DISK PROT
   ECTED":CLOSE #2:GOTO 710
MP 1980 IF OOPS=167 THEN PRINT "FILE LOCK
   ED":CLOSE #2:GOTO 710
XY 1990 IF OOPS=169 OR OOPS=162 THEN PRIN
   T "DISK FULL -- TRY AGAIN WITH ANOTHER
    DISK":CLOSE #2:GOTO 710
HB 2000 IF OOPS=170 THEN GOTO 1360
YI 2010 IF OOPS<143 THEN PRINT "WHAT HAVE
    YOU DONE TO THIS PROGRAM?":? "ERROR -
   - ";OOPS:GOTO 710
IL 2020 IF OOPS=160 THEN PRINT "WRONG DRI
   VE?":CLOSE #2:FOR SLOW=1 TO 200:NEXT S
   LOW:GOTO 1830
```

8-Bit News

Mydos 5.0 Goes Shareware

Atari's on-again, off-again disk operating system (DOS) developments have caused many XE/XL owners to write their own DOS utility programs to solve the lack of a good operating system. Many DOS utilities have been written since Atari DOS 2.OS back in the early '80s. *MYDOS,* written by Steve Marcelette, was one of the first full-function DOS programs for the XE. MYDOS was originally packaged for sale through the normal dealer channels, but now MYDOS 5.0 has been put into the public domain as a shareware product.

Shareware is a new method of distributing software through public bulletin board systems and your local dealer or user group. You are free to make a copy of MYDOS 5.0 and use it for home or business. If you decide to keep it, the author asks you to send him $10, as a royalty for writing a useful program.

MYDOS 5.0 is a pretty hefty program. In addition to a disk sector editor, command line interpreter (CLI), and multiple autorun file support, MYDOS supports many different types of disk drives for your XL/XE computer. If you have a Happy drive, Atari XF551 or even an ICD hard disk drive, MYDOS allows you to create custom disk drivers to operate your drive. MYDOS is density smart.

Shareware is a new method of distributing software through public bulletin board systems and your local dealer or user group.

Bill Wilkinson, Where Are You?

Last June, ANALOG reported that Optimized Systems Software (OSS), the popular manufacturer of products for the XE/XL computer, had been bought out by ICD Computers. OSS was one of the first companies to develop software for the Atari 800; they even worked on some of the original operating system routines before the 400/800 went into production. One of the founding principals of OSS is Bill Wilkinson, a prolific writer who professes the inherent beauty of the Atari home computer.

Bill has been found at some of the Atarifests, the larger trade shows like Comdex and CES, and some of the local user group meetings. He always has something interesting to say and gets to the point clearly and quickly. But, since the OSS/ICD buyout, no one has seen or heard from Bill. If he has completely removed himself from the Atari community, we ve lost a dedicated friend.

Atari Founds Atari Computers

Much has been written about Atari's positioning of the XL/XE home computer as a high-end video game machine. Most XL/XE owners become disturbed to find Atari openly telling of how bad sales are when they try to sell their 8-bit computers as home computer systems. But, the truth is that a majority of XL/XE owners surveyed by ANALOG indicate that they have real-world applications for their Atari computers in small businesses and at home.

Speaking at a panel discussion on niche marketing at the Comdex computer trade show in Atlanta this past April, Neil Harris told the small but interested audience that "Atari's roots were firmly placed in games, even before the Tramiels took over the company." Neil said that the slump in the 8-bit market has partly been due to slow product releases (disk drives and software) and to the lack of a gameplan to revamp the Atari 8-bit home computer market.

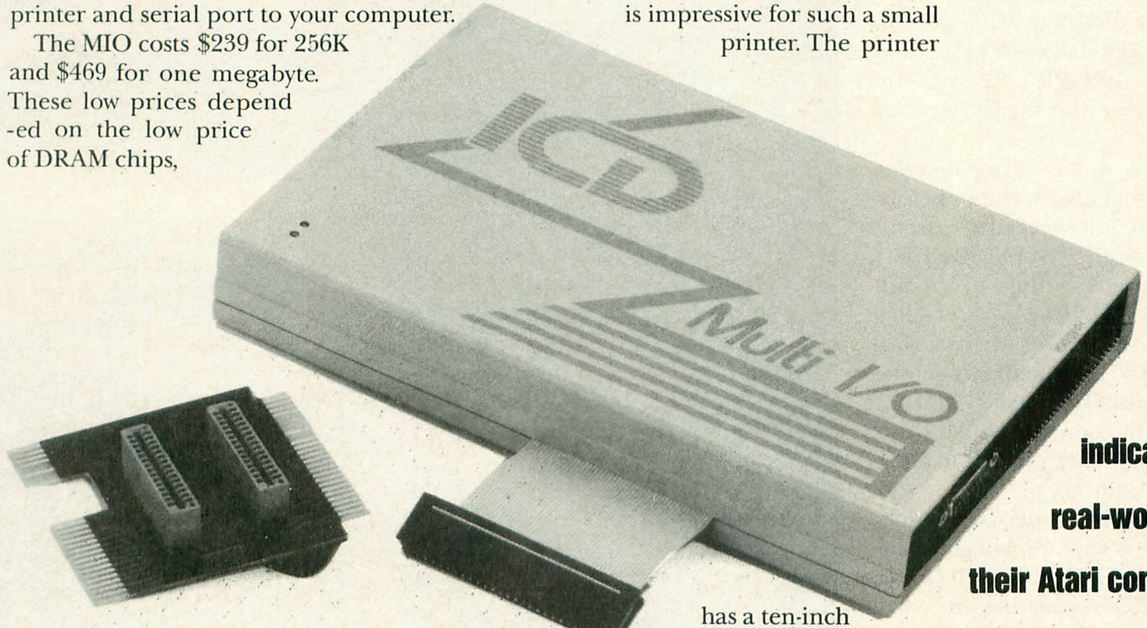
Neil was previously employeed by Atari as director of corporate communications, which made him the mouthpiece through which the company would communicate both rumors and facts about news and information pertaining to Atari computers. With a new title, Director of Product Marketing, Neil is now working for a new entity within Atari called *Atari Computers.* In an age where you can buy an Atari calculator in your local grocery store, or buy a 7600 cartridge-based game machine at your local toy store, Atari Computers is a newly founded division to market Atari's home and business computers in the US market.

Previously, the Tramiels had set up one person to be the marketing director for Atari Corp. Jerry Brown was a well established marketing director at IBM before joining Atari in 1987. Jerry arrived with great fanfare, but left the company six weeks later. Four other marketing directors floated through Atari in 1987. The new Atari Computers is headed by Chuck Babbit, President; Tony Gould, V.P. Sales; and Neil Harris, Marketing. Hopefully the new combination will turn things around for the XL/XE line.

No More MIO Boards (for now...)

After President Reagan's "let's get tough" policy on opening foreign markets caused Dynamic Random Access Memory (DRAM) chips to skyrocket, many companies depending on a supply of these high-capacity memory chips began to feel the squeeze. ICD makes the MIO, a popular add-on board for the XE/XL home computer. The MIO adds 256K or one Megabyte of extra memory, a hard-disk port, printer and serial port to your computer.

The MIO costs $239 for 256K and $469 for one megabyte. These low prices depended on the low price of DRAM chips, which now cost significantly more. So, ICD has stopped making MIO boards. The few that are left in ICD's warehouse are available directly from ICD (this excludes dealers) while supplies last.



ICD
1220 Rock Street
Rockford, IL 61101-1437
(815) 968-2228

Home Printer

Seikosha has begun shipping its new SP-1600 AI printer. The SP-1600 AI prints 160 characters per second (CPS) in draft mode and 33 CPS in near-letter-quality mode. The printer is exceptionally quiet for an impact-style dot matrix printer. It prints with a noise level below 52 dBA, which is quiet enough for a small business that doesn't have a lot of space.

The SP-1600 AI is compatible with Epson FX and IBM graphics printers, so all the usual programs for the XE/XL will work with it. The printer has both serial and parallel connections, so you can use it with an Atari 850 interface or ICD MIO board.

The printer uses a 9-pin print head which gives you enough resolution to print graphics, reports and light business correspondence; the letter quality mode is impressive for such a small printer. The printer has a ten-inch carriage and comes with tractor and friction feed. A sheet feeder is also available. The SP-1600 AI has a suggested list price of $349.

Seikosha America Inc.
1111 Macarthur Blvd.
Mahwah, NJ 07430
(201) 529-4655

Turkey Atari Users Group

A group of Atari 8-bit users in Turkey has been sending letters to user groups in the United States and Canada. The Turkish group has been sending money in the hopes that the domestic user groups will return public-domain software and utilities. Apparently, the Turks love the 130XE computer. But, software and hardware are very expensive overseas. An 800XL can cost as much as $800, and monitors can be priced over $1,000.

ANALOG *encourages foreign Atari user groups to contact us about your group's activities and interest.*

A majority of XL/XE owners, surveyed by *ANALOG,* indicate that they have real-world applications for their Atari computers in small businesses and at home.

Master Me

continued from page 9

tape booted. A zero means that everything bit the big one.

If a cassette boot attempt doesn't work, then the OS goes on as though there were no attempt. If the disk boot attempt fails, and this has happened to most of us, then a lovely "BOOT ERROR" message appears on the screen and the OS gives it another try.

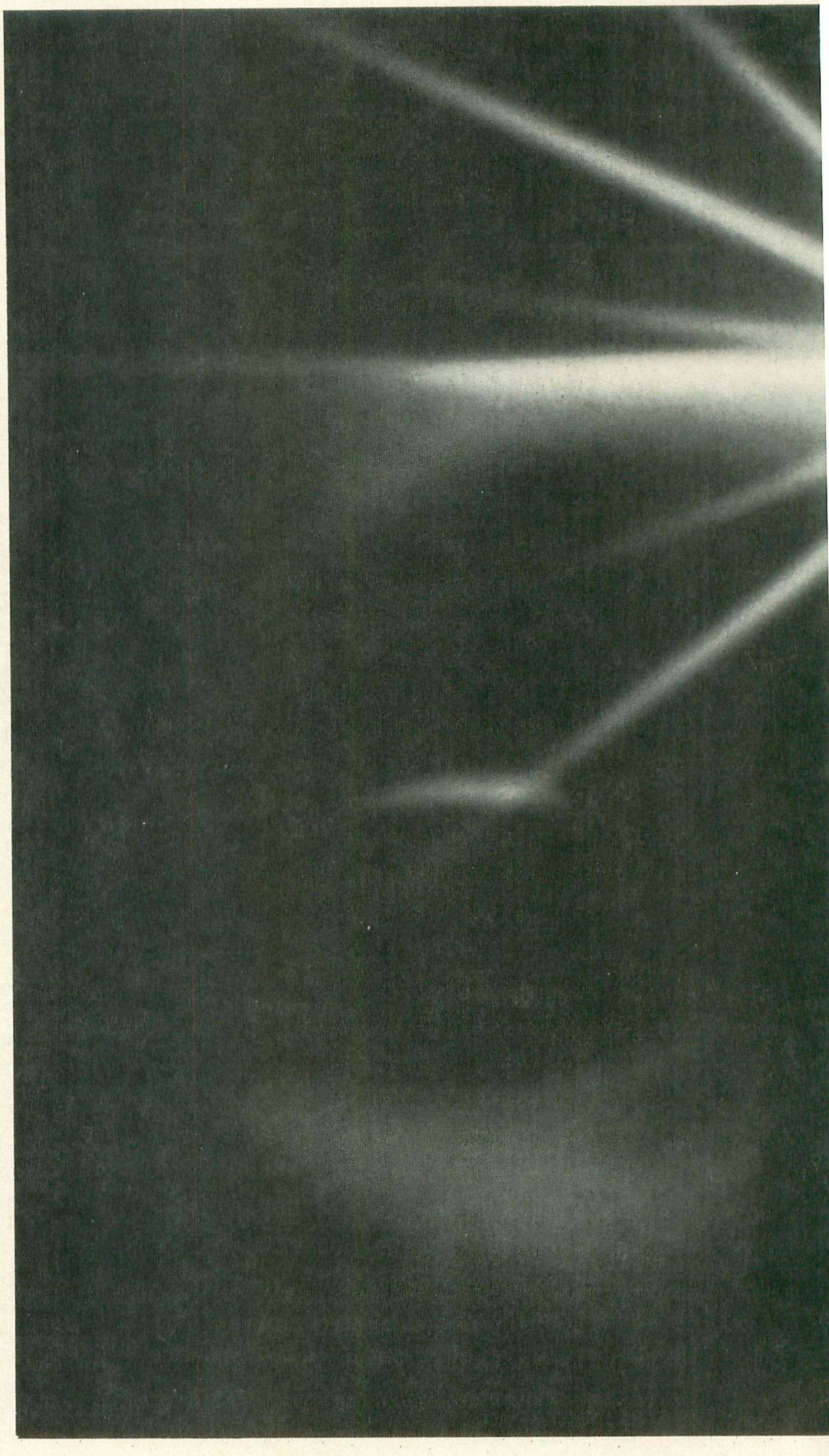
Okay, now for some miscellaneous stuff. A cassette boot always comes before a disk one. If there is a successful cassette boot, then every time SYSTEM RESET is pressed the computer will go to the address stored in CASINI.

The address is a location where a routine you want to use is located in memory. This address is usually called a "vector," because it points to something. You can JSR in machine language or USR in BASIC to get to the routine.

Back to CASINI. If the disk boots successfully, then the computer will go to the address stored in DOSVEC (10,11). If BOOT? is set to 255 by you, then the computer will "lockup" if SYSTEM RESET is pressed. This is a great way to keep people from looking at your programs. Incidentally, "lockup" means that the computer will not do anything until you turn it off.

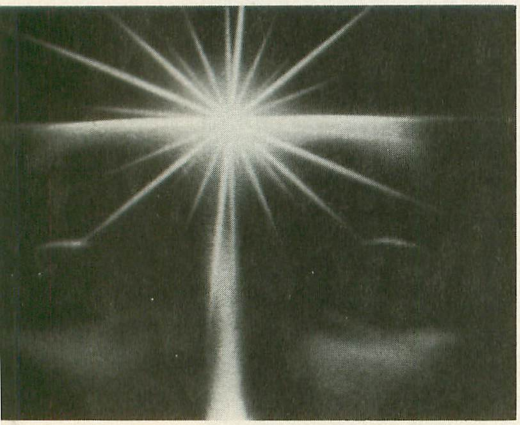
DOSVEC
10,11 000A, 000B

This is another vector, used to tell the OS what to do when SYSTEM RESET is pressed. It holds the cassette-boot starting address, the disk-boot starting address, or the address of the "blackboard mode" routine (type "BYE" from BASIC and press RETURN; that's the blackboard mode and the routine for it starts at location 58481). It's called DOSVEC, because if you're using DOS from BASIC, DOSVEC holds the address that BASIC jumps to when you call DOS (DOSVECtor—get it?). If you want to use this location from BASIC to point it to your own routine, then you'll have to



mory Map

Master Me



make a small change to DOS, since in this case SYSTEM RESET restores DOSVEC to its original value. The change is easy to make, though. All you have to do is POKE 5446 with the Least Significant Byte (LSB) of the address of your routine, and 5447 with the Most Significant Byte (MSB) of the address. The MSB is the first two digits of the hex address, the LSB is the last two. You can compute MSB and LSB from a decimal address with the following formulas: MSB = INT (address/256), LSB = address – (256*MSB). Then call DOS and resave it using the WRITE DOS FILES option. This will give you a custom version of DOS that will allow your routine to run every time SYSTEM RESET is pressed or DOS is called.

Miscellaneous stuff again; DOSVEC is set to 6047, the address of a routine to load in the DUP.SYS file, if DOS is used and it is not told otherwise (i.e., no user boot programs). And, for you machine-language dabblers, if you create an AUTORUN.SYS file that doesn't end with an RTS, make sure you set BOOT? to 1 and COLDST (580) to 0 (so as not to confuse the computer).

DOSINI
12,13 000C, 000D

This one's easy. Essentially, it is the disk equivalent of CASINI. As a matter of fact, the cassette initialization address is stored here before the OS realizes it's doing a cassette boot and moves it to CASINI. If there is no cassette or disk boot, DOSINI will read 0, 0.

DOSINI can be very useful because it holds the address that the OS jumps to when SYSTEM RESET is pressed. If you have a machine-language routine that you want to go to whenever SYSTEM RESET is pressed, store its address here.

APPMHI
14,15 000E, 000F

This location helps prevent your programs from accidentally being written over by the OS. If you're using BASIC, it points to the end of your BASIC program. The OS uses it to determine whether or not there's room for the graphics mode you want to use. As you probably know, the graphics mode stuff (screen memory and display list) is stuck way up at the top of memory. When you tell the OS to set up a graphics mode (with either a GRAPHICS or OPEN "S:" command), it tries to put the display list and screen memory right below the top of memory. Unfortunately, sometimes there isn't enough room, and they would extend down into your program, which you obviously don't want to happen (unless it's a horrible program). APPMHI to the rescue! Before it sets up the requested graphics mode, the OS checks APPMHI to see if there's enough room. If there isn't, it tells you so and sets up a GRAPHICS 0 screen instead, updating MEMTOP (741,742) in the process. MEMTOP, in case you didn't guess, holds the address of the last possible memory location you can use for your program, i.e., the memory location right before the display list. On the other hand, if there *is* enough room, the desired mode will be set up and MEMTOP updated accordingly.

Sometimes you may want to use the memory between the end of your program and MEMTOP to store character sets or player/missile information. That's fine, but make sure you change APPMHI

mory Map

so that the OS knows that you're using that memory (in other words, set APPMHI to point to the memory address *after* the last one you use).

Other locations that might be of interest here are CHBASE (54281), PMBASE (54279) and RAMTOP (106).

Machine-language programmers: Locations 16 through 127 are cleared on either coldstart or warmstart.

POKMSK
16 0010

POKMSK is used to turn various types of "interrupts" on or off. An interrupt is exactly what it sounds like; the computer gets interrupted from whatever it's doing and is told to do something else (it then usually returns to what it was doing before it was so rudely interrupted).

For machine-language programmers, POKMSK deals with POKEY interrupts and is used and altered by the IRQ service. It's also a shadow register for IRQEN (53774).

The following chart (Figure 3) shows exactly what part of POKMSK deals with which interrupts. Change a specific bit to a one to turn on that interrupt, zero to turn it off.

Before we decide whether or not any of this is useful, a few notes for the diehards. The default value for POKMSK is 192, BREAK key and "other key" interrupts enabled. When you enable a timer interrupt, the associated AUDF register will be used as a timer and will generate an interrupt request (IRQ) when it has counted down to zero. See VTIMR1/2/4 (528 to 535) and the POKEY chip (53760 to 54015) for more details.

BIT NO.	DECIMAL VALUE	TYPE OF INTERRUPT
7	128	BREAK key
6	64	"Other key"
5	32	Serial input data ready
4	16	Serial output data required
3	8	Serial out transmission finished
2	4	POKEY timer four ("B" and later OS ROMs only)
1	2	POKEY timer two
0	1	POKEY timer one

FIGURE 3. POKMSK Chart

For you beginners, as well as the pros, there is a handy-dandy use for POKMSK. If you haven't guessed already it allows you to disable the BREAK key so that nobody can BREAK into your program and steal your code. All you have to do is turn bit seven off. How do you do that? Try the following subroutine:

```
1000 BK=PEEK(16):IF BK>128 T
HEN POKE 16,BK-128:POKE53774
,BK-128
1010 RETURN
```

Notice that we also change Location 53774. As mentioned before, POKMSK is a shadow register for 53774, and therefore both must be changed. We also check first to make sure that bit seven is on. We do this because, unfortunately, this routine has to be called more than once. You see, the BREAK key is re-enabled by the first PRINT statement that prints to the screen, by an OPEN "S:" or OPEN "E:" statement, by the first PRINT statement after such an OPEN, by the first PRINT statement after a GRAPHICS command, or by a SYSTEM RESET. Phew! To make sure you keep the BREAK key disabled, you'll want to GOSUB to the preceding routine after each such command.

More for the machine-language programmer. If you have the newer OS 'B' ROM, there is a vector for the BREAK key interrupt that allows you to write your own routine for the BREAK key. It is called BRKKEY, and can be found at locations 566 and 567.

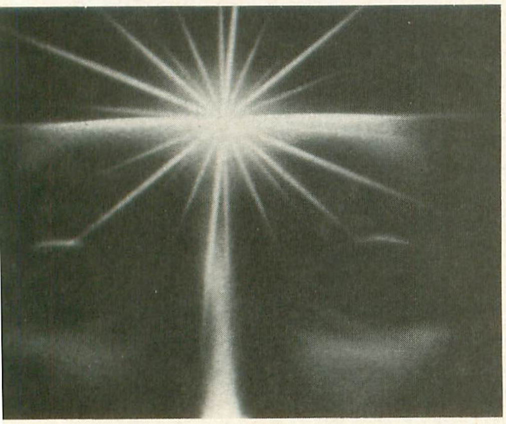
BRKKEY
17 0011

Okay, you've used POKMSK to zonk out the BREAK key. What happens if for some reason you need to know if somebody's pressing it? BRKKEY tells you just that. If it's equal to zero then the BREAK key is pressed (if it's not then it isn't!). If you're looking at BRKKEY from BASIC, remember that you'll have to keep checking it over and over again; BRKKEY tells if BREAK is pressed, not if it were

Machine-language programmers, this location along with POKMSK lets you write your own BREAK key routines if you don't have the 'B' ROM, or if you want to make sure your software will work on the old ROMs. If you do have the 'B' ROM (location 58383 will equal zero if you do), you can use the vector mentioned under POKMSK.

A few boring bits if information. If the BREAK key is pressed during an input/output (I/O) opertion, BRKKEY will read 128, *not* 0. The keyboard, display, screen, and cassette handlers all check BRKKEY to see if they should BREAK (why else?), as do I/O routines and scroll and draw routines. Also look at locations STATUS (48) and DSTAT (76) for related stuff.

Master Me



RTCLOK
18-20 0012-0014

This one's actually fun and interesting, and you may even have used it already. It's a clock—the "internal real-time clock" (which just means that it's inside the machine and actually keeps good time). It doesn't count in seconds though, but rather "jiffies." A jiffy is 1/60 of a second, which happens, not by coincidence, to be the time that it takes the television to fill the screen. After the screen is filled, a special interrupt occurs, called the Vertical BLANK (VBLANK) interrupt. The OS gets a lot of things done during VBLANK, one of which is updating RTCLOK. Every jiffy (during VBLANK), Location 20 gets increased by 1 until it equals 255. At that point, since 255 is the largest number a memory location can hold, it gets reset to 0 during the next VBLANK, and Location 19 gets increased by 1. You can probably guess what happens next. When Location 19 reaches 255, it gets set to 0 during the next VBLANK and Location 18 gets increased by 1. Finally, when Location 18 reaches 255, everything gets reset to 0 and the whole thing starts all over again. So, to put things in a more understandable perspective, Location 20 increases by 1 every 1/60 of a second, location 19 every 4.27 seconds (256/60), and location 18 every 18.2 minutes (4.27 seconds*256).

The following routine will tell you the number of jiffies, seconds and minutes that the clock has been running, i.e., since you turned on the computer or last POKEd 18 to 20 with zeros.

```
10 J=PEEK(20)+PEEK(19)*256+P
EEK(18)*256*256
20 S=J/60
30 M=S/60
40 PRINT "RTCLOCK reads ";J;
" jiffies, or ";S;" seconds,
 or ";M;" minutes."
```

All three locations are set to zero when you turn on the computer or press SYSTEM RESET. You can set them to whatever values you want just by POKEing them. Possible uses for RTCLOK include timing things that need precise timing. You can even use it to keep track of the time (what an absurd use for a clock).

BUFADR
21,22 0015,0016

This is a temporary register used to store the disk buffer address. It exists so that the OS can use indirect addressing to access the disk buffer. If this doesn't make sense, the BUFADR is not the place for you.

ICCOMT
23 0017

Another hard-core location. ICCOMT holds the CIO (Central Input Output) command and is used as an index into the command table to find the offset for the correct vector to the desired handler routine. Like I said, for hard-cores only.

DSKFMS
24,25 0018,0019

mory Map

This is used as a vector to the FMS (File Management System). It is called JMPTBL by DOS (which doesn't know any better).

DSKUTL
26,27 001A,001B

Another location used by DOS. DOS calls it BUFADR, but we'll continue to call it DSKUTL so as not to get confused with the OS BUFADR (21,22). DSKUTL points to a buffer that the disk utilities package (DUP) uses when copying or duplicating a file. If the user says it's okay to use the program area while copying or duplicating, then DSKUTL gets the value in MEMLO (743,744). If the user says no way to the program area, then DSKUTL gets the address of DBUF, a special 250-byte buffer at Location 7668.

PTIMOT
28 001C

If you're not a big fan of machine language I/O, then skip this one. PTIMOT is the printer timeout value. It's set by your printer handler software, and initialized by the OS to 30, which represents 32 seconds. If you're good at math you'll realize that 60 would represent 64 seconds. It's updated after each printer status request, getting the specific timeout status from DVSTAT+2(748).

A timeout is essentially what it sounds like. The printer (it could also be a disk drive or similar device) says, "Hey, timeout," and takes five. This has the noticeable effect of the printer just sitting there for a brief period of time doing nothing. Then it decides to come back and get to work again. What are you going to do, fire it? Anyway, those of you with the original OS may be very familiar with this situation, since that version of the OS contained a bug causing unnecessary timeouts. You would be doing something like printing when all of a sudden the computer would stop everything for up to five minutes. Version B did away with it.

PBPNT
29 001D

PBPNT is an index (pointer) into the print buffer. It tells the OS how full or empty the buffer is, and can therefore have any value from zero up to the size of the print buffer, PBUFSZ (30).

PBUFSZ
30 001E

PBUFSZ is the size of the print buffer, but not necessarily the size of the print line. The normal buffer size is 40 bytes (which is obviously not the normal line size for most printers). It is initialized to zero by the OS (and not set until P: is opened), and set to four in the case of a printer status request.

Characters get stored in the print buffer on their way to the printer. The OS checks PBPNT (29) to see whether it's equal to the buffer size (which would mean that the buffer is full) and, if it is, the buffer gets sent to the printer. If the buffer gets an EOL (End Of Line) character, then the OS fills the rest of the buffer with spaces and sends it to the printer.

PTEMP
31 001F

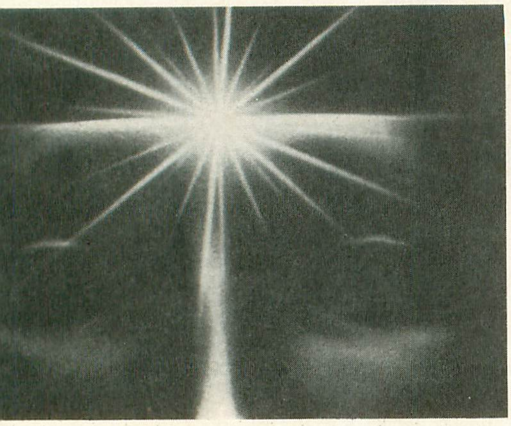
This is used by the printer handler to temporarily hold the character being sent to the printer while it goes off and does some chores.

Zero Page Input/Output Control Block (ZIOCB)

The 16 locations from 32 to 47 are used by CIO to make I/O as efficient as possible (remember the speed advantage of page zero). They are set up in the same way as the regular IOCBs (832 to 959) and essentially act as a mirror for the IOCB that wants to be used. In other words, when a CIO operation gets going, the information in the IOCB that's involved is moved to here, where it is used by the CIO routines. When the CIO is all done, then the updated information is moved back to the IOCB. Remember, as complicated as this sounds, it's only done for the sake of speed.

ICHIDZ
32 0020

This serves as an index into the handler address table for the file that's currently open on this particular IOCB. If there is no such open file (i.e., the IOCB is free), then ICHIDZ gets set to 255.



ICDNOZ
33 0021

The device or drive number. DOS uses it to tell the maximum number of devices, and therefore calls it MAXDEV (I'll bet you can see a connection there). It gets initialized to one.

ICCOMZ
34 0022

This is the command byte, which is set by the user, in the course of setting up the regular IOCB, to tell CIO what kind of operation is to be performed (GET, PUT, FORMAT, etc.). It also determines the format of the rest of the IOCB (which will be different for different commands).

ICSTAZ
35 0023

ICSTAZ is the status of the last IOCB action taken. The device in question tells CIO what happened, CIO tells the OS, and the OS sets ICSTAZ (a little chain of command here). Hopefully everything went okay, but if it didn't, ICSTAZ is the guy who'll know.

ICBALZ,ICBAHZ
36,37 0024,0025

Another buffer address, this one for data transfer. The OS also uses the ICBAZ twins to get the device name from the user (in this case ICBALZ/HZ holds the address of the location where the device name has been stored).

ICPTLZ,ICPTHZ
38, 39 0026, 0027

Each device has its own routine to "put" a byte into the device. The OS sets this location to hold the address (minus one) of the routine for the device being used. When the file is CLOSEd (and on powerup), it is set to the address of CIO's error routine for an illegal put (because you can't put something into a device unless it's open).

ICBLLZ,ICBLHZ
40,41 0028,0029

More buffer stuff. This time we have a counter that is initially set to the maximum number of bytes to PUT or GET in an I/O operation. It gets decremented every time a byte is put or gotten.

Machine language programmers can set this location to the size of the memory block they want to transfer. By checking after each PUT/GET to see if it's equal to zero, you'll be able to tell when the transfer is done.

ICAX1Z
42 002A

This is the first byte in the OPEN command after the IOCB number. It tells whether the user wants to READ, WRITE, or both.

ICAX2Z
43 002B

Okay, the last location was the *first* byte after the IOCB number, so guess which one this is? Hey, you're on the ball! ICAX2Z has no specific function, it really depends on the device you're using. CIO pretty much uses it as a working variable, although some serial port functions also use it.

Locations 44 to 47 are also called ICSPRZ or ENTVEC and are spare bytes for local CIO usage.

ICAX3Z,ICAX4Z
44,45 002C,002D

BASIC's NOTE and POINT commands use these locations to transfer disk sector numbers.

ICAX5Z
46 002E

ICAX3Z/4Z give the sector, ICAX5Z gives the byte within the sector. It is also used to store the IOCB number times 16 (since each IOCB is 16 bytes long, this gives an index to the beginning of the IOCB). In this case, it is called ICIDNO.

ICAX6Z
47 002F

Sometimes this doesn't do anything. But sometimes (only sometimes) it is called CIOCHR and used to temporarily store the byte that's getting ready to be PUT somewhere (aren't computers wonderful?).

Examples of using IOCBs from BASIC

(ICAX1Z and ICAX2Z are referred to as AUX1 and AUX2 respectively).

BASIC Command **Operating System IOCB Parameters**

OPEN #1,12,0,"E:"	**IOCB = 1** **Command = 3 (OPEN)** **AUX1 = 12 (READ and WRITE)** **AUX2 = 0** **Buffer Address = ADR ("E:")**
GET #1,X	**IOCB = 1** **Command = 7 (Get character)** **Buffer length = 0** **The gotten character is stored in the accumulator.**
PUT #1,X	**IOCB = 1** **Command = 11 (Put character)** **Buffer length = 0** **The character is output through the accumulator.**
INPUT #1,A$	**IOCB = 1** **Command = 5 (Getrecord)** **Buffer length = Len (A$) – 1 (no more than 255)** **Buffer address = Input line buffer**
PRINT #1,A$	**IOCB = 1** **BASIC uses a special put byte vector in the CB to talk directly to the handler.**
XIO 18,#6,12,0, "S:"	**IOCB = 6** **Command = 18 ("fill")** **AUX1 = 12** **AUX2 = 0**

STATUS
48 0030

A couple of uses for this guy. First, and probably most important (after all, it got its name for this one), it is used to hold the status of the SIO (Serial Input/Output) routine currently taking place. Figure 4 lists known values:

1	($01)	Operation complete (no problems)
138	($8A)	Device timeout (no response)
139	($8B)	Device NAK (no acknowledgement)
140	($8C)	Serial bus input framing error (your guess)
142	($8E)	Serial bus data frame overrun error (worse and worse)
143	($8F)	Serial bus data frame checksum error
144	($90)	Device done error (it packed up shop)

FIGURE 4. Status Chart

STATUS also uses TSTAT (793) as a temporary storage location. The other use, you may recall, is as a storage register during SIO routines for the BREAK abort, timeout and error values.

CHKSUM
49 0031

SIO's data frame checksum. A (much) simplified explanation of checksum is called for here. A checksum is essentially a sum of values used to check that the values were received correctly. When data gets somewhere, the computer adds all the values sent into one byte, and then sends that byte as the checksum value. When data is being received, the values are again added and the result compared to the checksum. If the two aren't equal, that means that at least one of the bytes received was incorrect, and the computer usually responds with an error message. In case you're wondering how you can add a whole bunch of bytes together and store the result in just one byte, you can't. If the checksum exceeds 255, then the carry is just added onto it. For example, in the world of checksums, 254 + 31 = 2,128 + 128 = 1, and so on.

A "checksum sent" flag is located at CHKSNT (59). CHKSUM relies on BUFRFL (56) to tell when the checksum is to be sent or received.

BUFRLO,BUFRHI
50,51 0032,0033

Hey, it's another data buffer! This one is used to hold the stuff that gets sent out or received during I/O. Actually, BUFRLO/HI is a dynamic pointer into the buffer (which just means that it points to the next byte to be sent/ received rather than always pointing to the beginning of the buffer).

SIO and DCB (Device Control Block) both use this pointer.

BFENLO,BFENHI
52,53 0034,0035

A pointer to the byte right after the end of the data buffer described in the previous location. This helps SIO and the DCB determine when the buffer is full.

CRETRY
54 0036

Sometimes you may get an error message trying to do stuff like reading from or formatting the disk. Before you tell the user to go toss the disk in the trash, however, you'll probably want to double-check to make sure that there really is something wrong with the disk, and it wasn't just a temporary boo-boo. CRETRY specifies how many times to try again before giving up. It is initialized to 13.

DRETRY
55 0037

The same basic idea as CRETRY, but where CRETRY double-checks that a specific *command* doesn't work, DRETRY double-checks to make sure that the whole *device* doesn't work. It is initialized to one.

BUFRFL
56 0038

If BUFRFL equals 255, then the date buffer is full. If it doesn't, it isn't.

RECVDN
57 0039

If RECVDN equals 255, then all the data that was supposed to be received has been. If it doesn't, it hasn't.

XMTDON
58 003A

If XMTDON equals 255, then all the data that was meant to be sent was. If it doesn't, it wasn't.

CHKSNT
59 003B

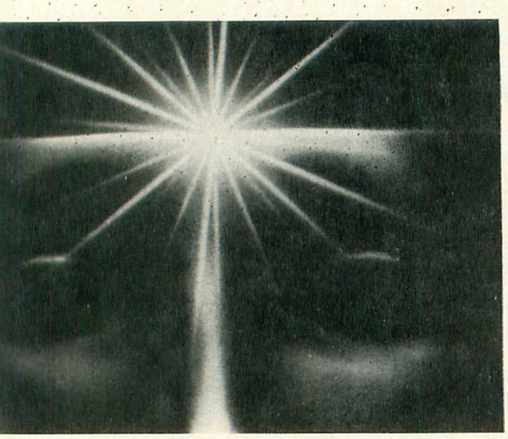
If CHKSNT equals 255 (you should know this already), then the checksum was sent.

NOCKSM
60 003C

More checksum stuff. A zero here means that a checksum follows the current transmission. No zero means no checksum.

BPTR
60 003D

By now you should be getting the idea that buffers are pretty popular items around a computer. Here's another buffer to further enforce that idea. This time we



have one for cassette data. Like BUFRLO/HI, BPTR is actually a pointer into the buffer (which is located at CASBUF [1021 to 1151]), indicating how full or empty the buffer is. It can be anything from zero to the value in BLIM (650). If it's equal to BLIM, then the buffer is either empty or full (depending on whether it was being read into or written out of, respectively). It is initialized to 128.

FTYPE
62 003E

You load in a program from cassette and while it's loading, the computer goes "beeeep (pause) beeeep (pause) etc.," right? Well, the pause has a name. It's called an "inter record gap." Can you say "inter record gap"? Sure, I knew you could. Anyway (so much for the comic relief), FTYPE specifies the kind of gap to put on the tape. It equals 0 for normal gaps (like in a CLOAD tape), 128 for continuous (long) gaps (like in an ENTER "C:" tape).

FTYPE gets its value from ICAX2Z (43), which gets it from DAUX2 (779), which gets it from the user.

FEOF
63 003F

Okay, we're still loading from cassette. How do we know when there's no more to read? The last record (each beep when loading represents a record) on a cassette file has a command byte of 254 and is called the EOF (End Of File) record. FEOF is set to 255 when the EOF record is reached, and 0 before that.

See CASBUF (1021) for an explanation of the way cassette records are structured.

FREQ
64 0040

Quite simply, the number of beeps that the Atari makes when you OPEN the cassette handler: one beep for read, two for write (type "CLOAD" and press RETURN for a demonstration).

SOUNDR
65 0041

SOUNDR is used to turn the beeping off (or back on) while the cassette or disk program is loading. A zero here will stop the beeping, anything else will get it going again. Also see location PACTL (54018). The beeping is caused by the loading of data from the right channel. Atari added this to the computer so that its educational tapes can talk to you while loading programs. Ah, hah! This must mean that the left channel still can be heard even if you change the value in location 65.

CRITIC
66 0042

CRITIC is used to tell the OS that the current I/O operation is time-critical (disk or cassette operations, for example). This is important, because in the case of time-critical I/O it is important that the computer spend as little time in vertical blank as possible. When CRITIC is a nonzero value, the OS knows not to execute the second stage of the VBLANK process (CRITIC is checked at the end of Stage 1). Since there are some things happening during Stage 2 that you may not want to interrupt (check the OS listing if this is really of concern to you), CRITIC should be used only when necessary. To experiment, poke a 2 into 66 and then press any letter. The repeat capability will not work and CONTROL-2 will sound funny. You can't press any key twice in a row.

The following seven bytes are called FMSZPG and serve as zero-page registers for the disk-file manager system (FMS).

ZBUFP
67,68 0043,0044

When the FMS does disk I/O, it needs to know the user filename so it can OPEN the file. It expects to find it in a buffer pointed to by ZBUFP.

ZDRVA
69,70 0045,0046

Zero-page drive pointer. FMS also uses ZRDVA in its setup, free sector and get sector routines. I know this sounds somewhat cryptic, but it's that kind of location.

ZSBA
71,72 0047,0048

A pointer to the sector buffer.

ERRNO
73 0049

If things go wrong during disk I/O, this is where you can find the error number. FMS initializes it to 159.

CKEY
74 004A

If the START button is held down when the Atari is first turned on, CKEY is set to one (zero otherwise). This indicates that a cassette file is to be booted.

CASSBT
75 004B

If a cassette file is booted and the boot is successful, CASSBT gets set to one. Zero means boot no goot. Also see BOOT? (9).

DSTAT
76 004C

A location of all trades, DSTAT is used mainly by the display handler to indicate display status and as a keyboard register. It is also used to indicate a cursor out of range error, the BREAK abort status, and too little memory for the desired screen mode.

ATRACT
77 004D

Try leaving the Atari on for about nine minutes without pressing any keys (or save yourself some time by POKEing ATRACT with 128). You've probably run across this effect before. It's called the "attract mode" and, as you can see, causes the colors on the screen to change every four seconds or so, at subdued brightnesses. Why, you may ask? If you leave your computer alone for several hours with a picture on the screen that doesn't change (like when you break for lunch and forget to turn the TV off), it can "burn" the picture tube of your television set and leave a permanent, although faint, image on the screen. You obviously don't want this to happen, so Atari thoughtfully created this solution.

Whenever you press a key IRQ (Interrupt ReQuest) sets ATRACT to 0. Otherwise, every four seconds VBLANK increments it by 1. When it reaches 127 it gets set to 254, and the Atari enters the attract mode. That's the way it stays until a key is pressed.

The attract mode only changes the four color registers COLPF0 to COLPF3 (53270 to 53273) and the background COLBAK (53274). That means that you'll have to write your own atract routine for DLI-induced colors.

If you're using joysticks but not the keyboard, you'll have to set ATRACT to zero every few minutes within your program.

DRKMSK
78 004E

This is one of the two locations used to change the colors in the attract mode (COLRSH is the other). DRKMSK makes sure that the colors aren't too bright. It's normally set to 246 during the attract mode.

For the curious machine-language programmers, DRKMSK is ANDed with the original color to mask out part of the brightness nibble. This is done during stage two VBLANK.

COLRSH
79 004F

The other location for changing colors, COLRSH actually *does* change the colors. It contains the current value of RTCLOK+1 (19).

Machine-language programmers, COLRSH gets EORed with the color registers (and background) before DRKMSK does its stuff.

Locations 80 to 122 are used by the screen editor and the display handler.

TMPCHR
80 0050

Guess what "TEMP" stands for? That's right, this is a TEMPorary (get it?) register used to move data to and from the screen. TEMP gets used by the display handler, which also calls it TMPCHR.

HOLD1
81 0051

Another temporary register for the display handler, this time used to hold the number of entries in the display list.

LMARGN
82 0052

Another tough name to figure out. If you're using graphics mode zero (or have a text window in the mode you're using), LMARGN determines the left margin for text. It's initialized to 2, but you can set it to whatever you want (up to 38). Try POKEing various values into this location.

RMARGIN
83 0053

The right margin (I'll bet that somehow you'd figure that out already). It's initialized to 38, and you can also set it to whatever you want (try and set it higher than the left margin though, and less than 40, okay?).

A few words about margins. SYSTEM RESET will restore them to their initial values. Text that is already on the screen will not be affected when you change the margins. Finally, logical lines (the longest a BASIC line can be) couldn't care less when you put the margins. Three lines on the screen and that's it for your logical line, baby, whether that means 120 characters or three.

ROWCRS
84 0054

This tells you the row on the screen that the cursor is currently on. It works in all the GRAPHICS modes and therefore has a range of 0 to 191 depending on the mode being used. Don't forget that a row is a horizontal line, not a vertical one (you'd be surprised at some of the people that forget). Rows are numbered from top to bottom, 0 being the top.

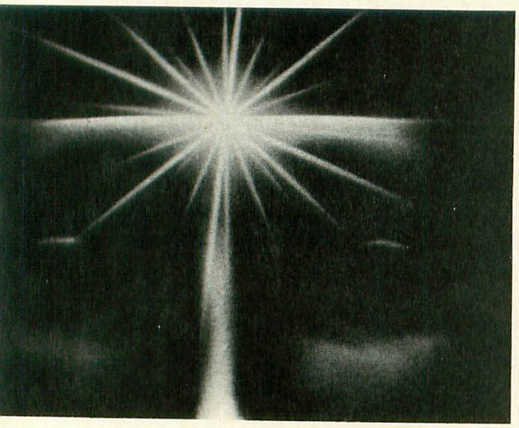
COLCRS
85,86 0055,0056

The column that the cursor is on, ranging from 0 to 319. Location 86 can only get set to 1 in graphics mode 8 (where the column number can exceed 255). Columns are numbered from left to right, 0 being the leftmost column. Incidentally, ROWCRS and COLCRS define the next cursor position to be read or written to, not the last one.

DINDEX
87 0057

This location tells the OS what graphics mode is currently being used (so it knows how to respond to a PLOT or some other screen I/O command). When you OPEN the screen (which the GRAPHICS command takes care of for you), the value of the AUX1 byte is stored in DINDEX. This means that DINDEX can have a meaningful value of anything from 0 to 11, keeping in mind the GTIA modes are numbered 9 through 11.

Most of the time you'll just leave DINDEX alone, because BASIC takes care of it for you. The times that it does come in handy, however, is when you want to use mixed mode display lists. It also comes in handy when you want to use the so-called "GRAPHICS 7.5," which gives you twice the resolution of graphics mode 7 with the same number of colors (machine-language programmers *also* know this mode as ANTIC mode "E"). The problem with using this mode is that it is, obviously, halfway between graph-



ics modes 7 and 8. That means that the display list is structured the same as a graphics mode 8 display list, but you have to PLOT to it like it was graphics mode 7. So what, you say? Let's look at an example? The following routine sets up what is called a GRAPHICS 7.5 screen by changing a GRAPHICS 8 display list:

```
100 GRAPHICS 24
110 DLIST=PEEK(560)+PEEK(561
)*256
120 POKE DLIST+3,78
130 FOR LINE=DLIST+6 TO DLIS
T+204
140 TYPE=PEEK(LINE)
150 IF TYPE=15 OR TYPE=79 TH
EN TYPE=TYPE-1
160 POKE LINE,TYPE
170 NEXT LINE
180 COLOR 3
190 PLOT 0,0:DRAWTO 79,85
200 POKE 89,PEEK(89)+15
210 PLOT 80,0:DRAWTO 159,95
999 GOTO 999
```

A brief explanation of what's going on here. We first set up for a graphics mode 8 screen with no text window. Then we find out where the display list is (see SDLSTL [560,561]) and then change each of the graphics mode 8 commands in it to graphics mode 7.5s. Then, since we have no text window, we must go into a continuous loop or else the screen will switch back to graphics mode 0 (take out line 1000 and see for yourself). RUN the program and you will see the screen go from blue to black as the display list changes. You now have a screen that is 160 dots wide and 192 dots high. Try adding the following lines to the preceding routine:

```
180 COLOR 3
190 PLOT 0,0:DRAWTO 159,191
```

Now RUN the whole thing. Uhh, oh! What happened? It's supposed to draw a blue line from the top left corner of the screen all the way down to the bottom right corner. Well, unfortunately the OS still thinks that it's in graphics mode 8, and in graphics mode 8 things get plotted differently than we want here. Let's trick the OS into thinking it's in graphics mode 7. That way it'll plot properly (technically speaking, we want two bits to represent a pixel rather than one). Add the following line:

```
175 POKE 87,7
```

RUN it again and whoops! ERROR 141? That means that the cursor went out of its allowed range. We forgot that graphics mode 7 only allows 96 rows. Change Line 190 to the following:

```
190 PLOT 0,0:DRAWTO 79,95
```

Now we're okay, but how do we draw in the lower half of the screen? Unfortunately, the tables that tell the OS how many rows and columns each mode has are in ROM, so we can't fool the OS into thinking that there are more rows. The only way around this problem is to treat a GRAPHICS 7.5 screen as being two separate screens, a top and a bottom (machine-language programmers can also write their own plot and draw routines). You can use SAVMSC (88,89) to pick the screen you want to use. Try the following program additions and then look at SAVMSC to see what's going on:

```
200 POKE 89,PEEK(89)+15
210 PLOT 80,0:DRAWTO 159,95
```

(This is a tedious process but it's the price you have to pay if you want the benefits of GRAPHICS 7.5)

SAVMSC
88,89 0058,0059

This is the location of the place in memory where the data is kept that goes onto the screen. Each number in memory represents one character on your TV or several pixels if in a graphics mode. The value at memory location SAVMSC goes at the upper left-hand corner of the screen. The next memory location then goes left side, one row down.

When you do I/O to the screen, the OS uses this address to figure out where to PLOT and PRINT. So, for example, the following line will put the letter "A"; in the upper left-hand corner of your graphics zero (or one or two) screen.

```
SCRMEM=PEEK(88)+PEEK(89)*256
:POKE SCRMEM,33
```

But wait, you say. CHR$(33) doesn't give us an "A"; what's going on here? I'll tell you. The Atari stores the characters in memory in a different order than the ATASCII order (which is what CHR$ uses). See CHRORG (57344) to find out how to convert from one to the other. Anyway, the values in screen memory represent the internal character order rather than the ATASCII one.

If you're not using a text mode, the values you poke to the screen will, obviously, affect the pixels on the screen (the dots on the screen). A pixel is represented by one, two or four bits. See location DMASK (672) to find out what bits in a byte affect which pixels in each mode (that was easy for me to say). Then try POKEing around. You may want to check CHRORG again; it has an example of using such POKEs to get characters on the screen in graphics mode 8.

Okay, so now you know how to change the first character on the screen. What if you want to change the sixth character on the tenth row; how do we know how to find it? Figure 5 shows how many bytes per row are required for each graphics mode.

MODE	0	1	2	3	4	5	6	7	8	9-11
BYTES/ROW	40	20	20	10	10	20	20	40	40	40

FIGURE 5. Number of bytes per row

Now, if you want to change character X in row Y, just multiply Y by the number of bytes per row for the mode you're using and add X (don't forget that the first row and column are numbered zero, not one). Add this value to the address in SAVMSC, and POKE away. For example, let's put the letter "B" in the middle of a graphics zero screen (row 11, column 19):

```
100 GRAPHICS 0
110 SCREEN=PEEK(88)+PEEK(89)
*256
120 POS=11*40+9
130 POKE SCREEN+POS,34
```

We want to make sure that we don't try and change a byte that isn't part of the screen, so let's add another line to our chart, this one giving the number of rows in each mode. We'll also multiply the number of rows times the bytes per row to get the total number of bytes taken up by the screen memory (Figure 6).

MODE	0	1	2	3	4	5	6	7	8	9-11
ROWS	24	24	12	24	48	48	96	96	192	192
BYTES	960	480	240	240	480	960	1920	3840	7680	7680

FIGURE 6. Screen memory requirements

Now these values, when added to the address in SAVMSC, will give you the value of the first byte after the end of screen memory. What they don't tell you is how much memory the *whole* graphics mode takes up. Why not? Because they don't take into account the display list (see SDLSTL [560,561]) and a few bytes that get trapped in the middle of everything. So how do we get this total memory amount? Well, it turns out that RAMTOP (106) points to the top of free memory, which coincides with the first byte after the end of screen memory. MEMTOP (741,742) points to the top of BASIC memory, which coincides with the first byte before the display list. So, if we subtract MEMTOP + 1 from RAMTOP * 256 (RAMTOP is in terms of pages), we'll get the total memory required. I'll save you the trouble and just give you the values. Our final chart is Figure 7.

Mode	0	1	2	3	4	5	6	7	8	9-11
Bytes/Row	40	20	20	10	10	20	20	40	40	40
No. of Rows	24	24	12	24	48	48	96	96	192	192
Total Screen Bytes	960	480	240	240	480	960	1920	3840	7680	7680
Total Mode Bytes (normal screen)	992	672	420	432	696	1176	2184	4200	8138	8138
(Split Screen)	----	674	424	434	694	1174	2174	4190	8112	

FIGURE 7. Screen requirements chart

You may have told yourself by now that you can change the values in SAVMSC and thereby change where the screen is. And if you can change where the screen is, you can keep more than one screen in memory at the same time. Well, you're half right. You definitely *can* have more than one screen in memory at the same time, but unfortunately SAVMSC only tells the OS where to PRINT and PLOT (and the like) to. It doesn't tell the computer what to display on the television screen. Fortunately, there is another pair of locations that tell what to display, and the word "display" should tip you off to where they are; they're in the display list (this is kind of like adult *Sesame Street*, isn't it?). Specifically, they're the fifth and sixth bytes in a normal (unaltered by you) display list. Try the following:

```
100 DLIST=PEEK(560)+PEEK(561
)*256
110 LOW=PEEK(DLIST+4)
120 LOW=LOW+1
130 IF LOW=256 THEN LOW=0:PO
KE DLIST+5,PEEK(DLIST+5)+1
140 POKE DLIST+4,LOW
150 FOR DELAY=1 TO 10:NEXT D
ELAY
160 GOTO 120
```

This will move the starting address of the screen one byte forward at a time, having the effect of swallowing up whatever was on the screen when you ran it. Press SYSTEM RESET to stop it and get everything back to normal.

A few things to note here. First, if you let this run for a while (get rid of Line 150 to make it hapen faster), the screen will suddenly fill up with a whole bunch of garbage. This "garbage" is actually your BASIC cartridge! The starting screen address has been moved so far forward that it has now entered the BASIC zone. You may have astutely noted that the garbage didn't scroll onto screen smoothly, but rather just sort of suddenly appeared. This is because the screen memory has committed a no-no. It has crossed a 4K boundary. What is a 4K boundary? It's the boundary between one group of 4096 bytes and the next one. How do you tell where one is? Well, first of all, the address of a 4K boundary is a multiple of 4096. Better yet, ıf you're working in hexadecimal, the leftmost digit in the four-digit hex number is the "4K digit" (this is not an official term). When it gets changed, a 4K boundary has been crossed. Okay? In any case, the whole purpose of this explanation was simply to tell you that the screen memory is not allowed to cross over a 4K boundary. The GRAPHICS command usually takes care of this for you, but if you're setting up more than one screen, you'll have to be careful.

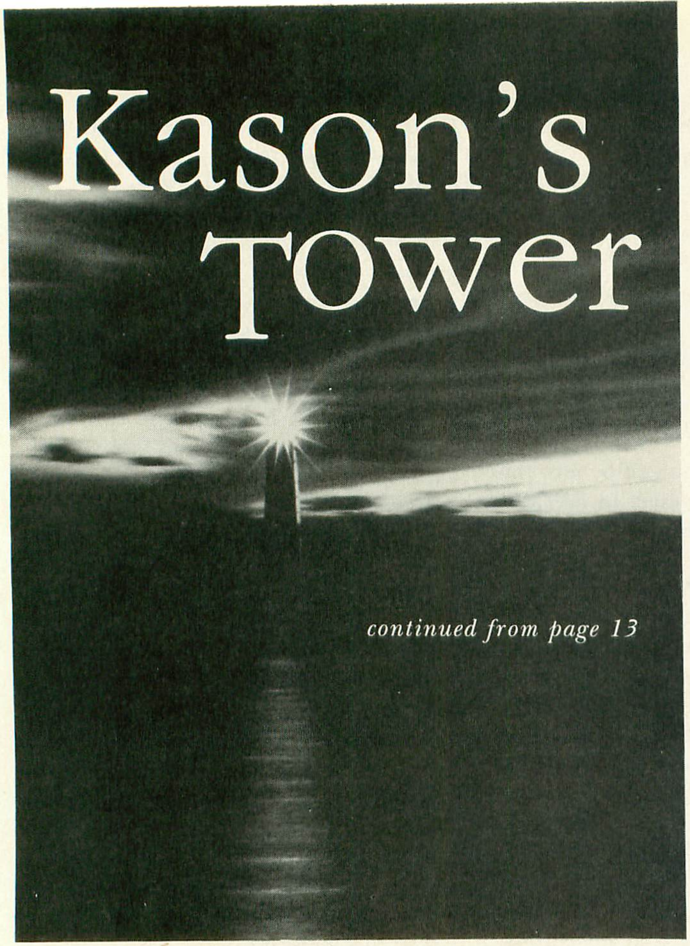
Going way back to our program example, you should also note that despite what's happening on the TV set, the OS still thinks that the screen is where it was originally, since we haven't changed SAVMSC. If you expect the OS to keep up with you, change SAVMSC as well as the display list.

Finally (and you thought it would never end), before we move onward and upward, a few bits of memory trivia. The address of the text window memory can be found at TXTMSC (660,661). And, in case you thought you weren't going to get a good multiple screen example, you're right. Just kidding.

```
99 REM Get everything set up
100 GRAPHICS 1:PRINT #6;"THI
S IS SCREEN ONE"
110 DLIST1L=PEEK(560):DLIST1
H=PEEK(561)
120 DLIST1=DLIST1L+DLIST1H*2
56
130 SCRMEM1L=PEEK(DLIST1+4):
SCRMEM1H=PEEK(DLIST1+5)
140 POKE 106,DLIST1H-4
150 GRAPHICS 2:PRINT #6;"THI
S IS SCREEN TWO"
150 DLIST2L=PEEK(560):DLIST2
H=PEEK(561)
170 DLIST2=DLIST2L+DLIST2H*2
56
180 SCRMEM2L=PEEK(DLIST2+4):
SCRMEM2H=PEEK(DLIST2+5)
189 REM Do the flipping
190 POKE 560,DLIST1L:POKE 56
1,DLIST1H
200 POKE 88,SCRMEM1L:POKE 89
,SCRMEM1H
210 GOSUB 1000
220 POKE 560,DLIST2L:POKE 56
1,DLIST2H
230 POKE 88,SCRMEM2L:POKE 89
,SCRMEM2H
240 GOSUB 1000
250 GOTO 190
999 REM  Pause between scree
ns
1000 FOR PAUSE=1 TO 200:NEXT
 PAUSE
1010 RETURN
```

Sorry, but no explanation for this one. You should ne able to figure it by yourself. I will, however, give you the following lines which you may want to add to make the screen look a little less messy.

```
205 POKE 559,34
235 POKE 559,34
1005 POKE 559,0
```



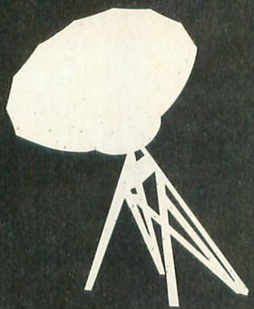
Kason's Tower

continued from page 13

LISTING 1: BASIC

```
UA 10 REM ******************************
NL 11 REM *        KASON'S TOWER       *
WX 12 REM *        BY JIM ROGERS       *
NG 13 REM *                            *
IA 14 REM *        COPYRIGHT 1988      *
GC 15 REM *     BY ANALOG COMPUTING    *
UM 16 REM ******************************
YJ 100 GRAPHICS 18:POSITION 3,5:? #6;"KAS
   ON'S TOWER"
TJ 200 RESTORE 5001:FOR A=1536 TO 1745:RE
   AD B:POKE A,B:SOUND 0,121+42*(A<1600)+
   81*(A<1670),10,4:NEXT A
VN 300 SOUND 0,0,0,0
GH 1001 DATA 76,73,32,56,124,124,252,240,
   240,230,48,0,252,252,92,92,108,108,104
   ,108,56,124,112,230,254
XS 1002 DATA 255,248,124,56,48,56,60,124,
   108,108,104,0,56,124,124,252,240,240,2
   30,48,0,252,252,60,60
JY 1003 DATA 126,118,224,224,56,124,112,2
   30,254,255,248,124,56,98,50,60,60,124,
   112,96,0,56,124,124,252
BS 1004 DATA 240,240,230,48,0,112,112,116
   ,62,62,254,246,134,56,124,112,230,254,
   255,248,124,56,60,12,12
EY 1005 DATA 62,62,126,116,0,56,124,124,2
   52,240,240,230,48,252,240,92,110,204,0
   ,0,0,0,56,124,112
KV 1006 DATA 230,254,255,248,124,48,56,60
   ,110,0,0,0,0,0,28,62,62,63,15,15,103,1
   2,0,63,63
QK 1007 DATA 58,58,54,54,22,54,28,62,14,1
   03,127,255,31,62,28,12,28,60,62,54,54,
   22,0,28,62
IL 1008 DATA 62,63,15,15,103,12,0,63,63,6
   0,60,126,110,7,7,28,62,14,103,127,255,
   31,62,28,70
XY 1009 DATA 76,60,60,62,14,6,0,28,62,62,
   63,15,15,103,12,0,14,14,46,124,124,127
   ,111,97,28
YN 1010 DATA 62,14,103,127,255,31,62,28,6
   0,48,48,124,124,126,46,0,28,62,62,63,1
   5,15,103,12,63
QJ 1011 DATA 59,58
WQ 1013 DATA 118,51,0,0,0,0,28,62,14,103,
   127,255,31,62,12,28,60,118,0,0,0,0,0,2
   4,60
BC 1014 DATA 126,24,24,24,60,24,60,24,60,
   126,35,65,255,65,35,196,130,255,130,19
   6,40,40,37,35,35
NZ 1015 DATA 37,40,42,45,47,50,53,57,60,6
   4,68,72,76,81,173,165,174,16,0,0,0,0,0
   ,0,172
TP 1016 DATA 165,182,165,172,16,121,96,72
   ,60,72,96,121,96,96,48,50,37,51,51,0,5
   1,52,33,50,52
OU 1017 DATA 0,52,47,0,34,37,39,41,46,3,5
   ,6,6,7,7,7,6,6,5,3,0,0,1,1,1
IE 1018 DATA 1,1,1,1,1,1,1,0,0,34,0,68,16
   2,144,144,128,121,108,96,91,60,81,121,
   121,121
QY 1019 DATA 81,60,60,60,81,60,60,60,81,6
   0,47,47,60,47,47,47,47,72,72,81,81,96,
   96,121,96
TY 1020 DATA 121,144,121,144,121,60,47,60
   ,134,62,254,46,0,0,154,44,0,56,0,79,72
   ,138,72,152,72
RG 1021 DATA 169,1,141,10,212,173,172,6,4
   1,1,201,1,208,30,172,164,6,185,165,6,1
   41,2,208,185,225
LE 1022 DATA 6,141,20,208,238,164,6,173,1
   64,6,201,7,144,5,169,0,141,164,6,172,1
   74,6,185,175,6
HO 1023 DATA 141,3
XV 1025 DATA 208,238,174,6,173,174,6,201,
   14,144,5,169,0,141,174,6,238,172,6,104
   ,168,104,170,104,64
DJ 1026 DATA 72,138,72,152,72,169,0,141,1
   64,6,141,172,6,141,174,6,165,224,24,10
   5,17,133,226,165,225
NZ 1027 DATA 105,0,133,227,160,0,162,17,1
   69,0,145,203,145,205,200,202,208,246,1
   73,173,6,133,203,133,205
JR 1028 DATA 160,0,162,17,177,224,145,203
   ,177,226,145,205,200,202,208,244,173,1
   89,6,201,1,208,53,238,190
GO 1029 DATA 6,174,190,6,224,4,144,5,169,
   0,141,1,210,224,5,144,34,169,0,141,190
   ,6,174,191,6
ZL 1030 DATA 189,137,30,141,0,210,169,163
   ,141,1,210,238,191,6,174,191,6,224,38,
   144,5,169,0,141,191
NK 1031 DATA 6,173,195,6,201,1,208,38,169
   ,42,141,3,210,174,196,6,189,37,30,141,
   2,210,238,196,6
YR 1032 DATA 173,196,6,201,19,208,14,169,
   0,141,195,6,141,196,6,141,3,210,141,2,
   210,173,211,6,201
RE 1033 DATA 1,208,38,169,166,141,5,210,1
   74,212,6,189,129,30,141,4,210,238,212,
   6,173,212,6,201,8
KY 1034 DATA 208,14,169,0,141,211,6,141,2
   12,6,141,5,210,141,4,210,104,168,104,1
   70,104,76,98,228,173
PA 1035 DATA 197,6
MY 1037 DATA 10,141,194,6,169,12,56,237,1
   94,6,168,169,38,153,175,6,153,176,6,17
   3,197,6,41,1,201
VW 1038 DATA 1,208,8,169,214,153,175,6,15
   3,176,6,160,0,162,6,169,0,153,216,6,20
   0,202,208,247,96
VA 1039 DATA 169,175,162,24,56,237,197,6,
   202,208,249,96,169,198,141,207,6,141,0
   ,208,141,1,208,169,175
GE 1040 DATA 141,173,6,169,0,133,20,165,2
   0,201,3,144,250,96,56,165,106,56,233,1
   6,141,7,212,141,192
IP 1041 DATA 6,169,62,141,47,2,169,3,141,
   29,208,173,192,6,24,105,3,133,208,169,
   0,133,207,162,5
QK 1042 DATA 160,0,169,0,145,207,200,208,
   249,230,208,202,208,242,173,192,6,24,1
   05,6,133,208,169,36,133
KB 1043 DATA 207,162,7,142,193,6,160,0,16
   2,12,185,15,30,145,207,200,202,208,247
   ,165,207,24,105,24,133
QL 1044 DATA 207,174,193,6,202,208,227,23
   0,208,169,29,133,207,162,7,142,193,6,1
   62,2,142,194,6,160,0
HO 1045 DATA 162,5,185,32,30,145,207,200,
   202,208,247,165,207,24,105,13,133,207,
   174,194,6,202,208,227,165
UX 1046 DATA 207,56,233,2,133,207,174,193
```

```
   ,6,202,240,47,142,193,6,162,2,142,194,
   6,160,0,162,5,185
JT 1047 DATA 27,30
JZ 1049 DATA 145,207,200,202,208,247,165,
   207,24,105,13,133,207,174,194,6,202,20
   8,227,165,207,56,233,2,133
HZ 1050 DATA 207,174,193,6,202,208,162,17
   3,192,6,24,105,4,133,204,133,206,230,2
   06,162,7,160,0,169,50
FD 1051 DATA 153,165,6,200,202,240,9,169,
   198,153,165,6,200,202,208,238,169,0,14
   1,8,208,141,9,208,141
UZ 1052 DATA 10,208,141,11,208,141,8,210,
   169,3,141,15,210,160,0,162,7,185,175,3
   0,153,192,2,200,202
OU 1053 DATA 208,246,169,33,141,111,2,169
   ,0,141,172,6,141,195,6,141,196,6,141,1
   97,6,141,203,6,141
PR 1054 DATA 214,6,162,7,160,0,169,38,153
   ,175,6,200,153,175,6,200,202,240,13,16
   9,214,153,175,6,200
AI 1055 DATA 153,175,6,200,202,208,230,16
   2,7,160,0,169,254,153,225,6,200,202,20
   8,247,169,29,133,225,169
AU 1056 DATA 135,133,224,32,46,32,173,48,
   2,133,207,173,49,2,133,208,160,1,169,2
   40,145,207,160,3,177
NN 1057 DATA 207,24,105,128,145,207,160,8
   ,162,12,169,138,145,207,200,200,200,20
   2,208,246,169,187,141,0,2
TG 1058 DATA 169,30,141,1,2,169,192,141,1
   4,212,160,13,162,31,169,7,32,92,228,17
   3,48,2,24,105,134
WH 1059 DATA 133,207
UR 1061 DATA 173,49,2,105,0,133,208,162,6
   ,142,193,6,160,0,162,20,169,255,145,20
   7,200,202,208,248,165
OS 1062 DATA 207,24,105,120,133,207,165,2
   08,105,0,133,208,174,193,6,202,208,222
   ,173,48,2,24,105,41,133
OZ 1063 DATA 207,173,49,2,105,4,133,208,1
   60,0,162,16,185,56,30,145,207,200,202,
   208,247,160,0,162,3
WY 1064 DATA 173,48,2,24,121,182,30,153,2
   18,0,173,49,2,105,4,153,219,0,200,200,
   202,208,233,169,5
LX 1065 DATA 141,201,6,169,0,141,199,6,14
   1,189,6,141,0,210,141,1,210,169,20,141
   ,200,6,160,0,162
ET 1066 DATA 20,185,81,30,145,222,200,202
   ,208,247,173,31,208,201,6,208,249,160,
   0,162,20,169,0,145,222
VK 1067 DATA 200,202,208,248,169,1,141,18
   9,6,160,0,173,201,6,24,105,16,145,218,
   32,46,32,169,0,141
QB 1068 DATA 197,6,160,0,173,199,6,24,105
   ,16,145,220,32,20,32,169,2,141,30,208,
   173,14,208,41,3
TV 1069 DATA 201,0,208,3,76,20,35,169,0,1
   41,189,6,169,6,56,237,197,6,168,162,24
   7,169,164,141,1
JQ 1070 DATA 210,189,81,29,141,0,210,173,
   10,210,153,225,6,169,0,133,20,165,20,2
   01,20,144,250,232,208
BE 1071 DATA 226,169
VC 1073 DATA 0,141,0,210,141,1,210,169,25
   4,153,225,6,169,1,141,189,6,32,237,31,
   238,197,6,173,197
MJ 1074 DATA 6,201,7,144,9,238,199,6,206,
   200,6,76,145,34,32,34,32,141,173,6,32,
   62,32,173,15
FL 1075 DATA 208,41,3,201,0,240,56,169,1,
   141,195,6,162,193,173,197,6,41,1,201,1
   ,208,2,162,50
MT 1076 DATA 142,207,6,142,0,208,142,1,20
   8,32,237,31,206,201,6,160,0,173,201,6,
   24,105,16,145,218
LL 1077 DATA 32,62,32,173,201,6,201,0,208
   ,3,76,74,34,169,2,141,30,208,173,120,2
   ,141,208,6,173
EU 1078 DATA 205,6,201,1,208,3,76,232,35,
   173,202,6,201,1,240,21,173,132,2,201,0
   ,208,14,173,120
TU 1079 DATA 2,141,204,6,169,1,141,202,6,
   141,211,6,173,202,6,201,1,208,91,32,34
   ,32,174,203,6
WI 1080 DATA 56,253,101,30,141,173,6,172,
   204,6,192,11,240,7,172,204,6,192,10,20
   8,10,173,207,6,56
LU 1081 DATA 253,113,30,141,207,6,172,204
   ,6,192,7,240,7,172,204,6,192,6,208,10,
   173,207,6,24,125
WP 1082 DATA 113,30,141,207,6,173,207,6,1
   41,0,208,141,1,208,238,203,6,173,203,6
   ,201,12,208,8,169
HH 1083 DATA 0,141
XU 1085 DATA 203,6,141,202,6,76,215,36,17
   3,205,6,201,1,240,39,173,208,6,201,13,
   208,88,169,1,141
AU 1086 DATA 205,6,160,0,177,224,201,56,2
   40,11,169,237,133,224,169,29,133,225,7
   6,22,36,169,101,133,224
GC 1087 DATA 169,29,133,225,173,205,6,201
   ,1,208,49,32,34,32,24,105,4,141,173,6,
   238,210,6,173,210
BE 1088 DATA 6,201,12,208,27,32,34,32,141
   ,173,6,169,0,141,205,6,141,210,6,165,2
   24,56,233,102,133
MC 1089 DATA 224,165,225,233,0,133,225,76
   ,215,36,173,208,6,201,10,240,14,173,20
   8,6,201,8,240,7,173
DX 1090 DATA 208,6,201,11,208,21,169,135,
   133,224,169,29,133,225,173,207,6,201,4
   7,240,3,206,207,6,76
EL 1091 DATA 159,36,173,208,6,201,6,240,1
   4,173,208,6,201,5,240,7,173,208,6,201,
   7,208,74,169,255
NS 1092 DATA 133,224,169,28,133,225,173,2
   07,6,201,200,240,3,238,207,6,173,207,6
   ,141,0,208,141,1,208
RL 1093 DATA 172,214,6,165,224,24,121,125
   ,30,133,224,165,225,105,0,133,225,238,
   215,6,173,215,6,201,3
SE 1094 DATA 208,20,169,0,141,215,6,238,2
   14,6,173,214,6,201,4,208,5,169,0,141,2
   14,6,160,0,162
JD 1095 DATA 2,142
LV 1097 DATA 193,6,173,197,6,10,141,194,6
   ,152,24,105,12,56,237,194,6,170,185,21
   6,6,201,1,208,89
HQ 1098 DATA 185,218,6,201,1,240,18,185,2
   20,6,56,233,1,153,220,6,201,0,208,69,1
   69,1,153,218,6
NC 1099 DATA 173,197,6,41,1,201,1,240,29,
   254,175,6,254,175,6,189,175,6,201,214,
   144,13,169,38,157
YN 1100 DATA 175,6,169,0,153,216,6,153,21
   8,6,76,78,37,222,175,6,222,175,6,189,1
   75,6,201,38,176
MG 1101 DATA 13,169,214,157,175,6,169,0,1
   53,216,6,153,218,6,200,174,193,6,202,2
   08,134,173,216,6,201
LW 1102 DATA 1,240,38,173,217,6,201,1,240
   ,31,173,10,210,41,15,24,105,16,141,220
   ,6,141,221,6,173
ZM 1103 DATA 10,210,41,1,168,169,1,153,22
   0,6,141,216,6,141,217,6,174,200,6,160,
   255,169,0,133,77
CT 1104 DATA 136,208,249,202,208,244,76,1
   71,34
DT 4000 GRAPHICS 5
GR 4010 XX=USR(1536)
WW 5001 DATA 104,169,28,133,206,169,252,1
   33,205,165,129,201,7,208,8,169,7,133,2
   06,169,0,133,205,165,136
TE 5002 DATA 133,203,165,137,133,204,162,
   240,224,3,208,108,160,4,200,177,203,20
   1,48,144,6,177,203,201,155
WO 5003 DATA 208,243,152,72,136,177,203,5
   6,233,48,133,207,169,10,133,209,162,2,
   134,208,136,177,203,201,48
MW 5004 DATA 240,20,177,203,201,48,144,23
   ,56,233,48,170,165,207,24,101,209,202,
   208,250,133,207,169,100,133
KH 5005 DATA 209,166,208,202,208,218,160,
   0,165,207,145,205,230,205,165,205,201,
   0,208,2,230,206,104,168,177
GH 5006 DATA 203,201,155,208,165,160,2,16
   5,203,24,113,203,133,203,165,204,105,0
   ,133,204,160,4,177,203,201
OJ 5007 DATA 1,240,140,224,240,240,229,16
   5,129,201,7,208,34,169,0,133,207,169,7
   ,133,208,169,252,133,205
UQ 5008 DATA 169,28,133,206,162,8,160,0,1
   77,207,145,205,200,208,249,230,206,230
   ,208,202,208,240,169,252,141
SV 5009 DATA 231,2,169,28,141,232,2,76,25
   2,28
```



DATA BASE DELPHI

If you haven't been online in a while, you'll be surprised to find a slightly altered set of database topics.

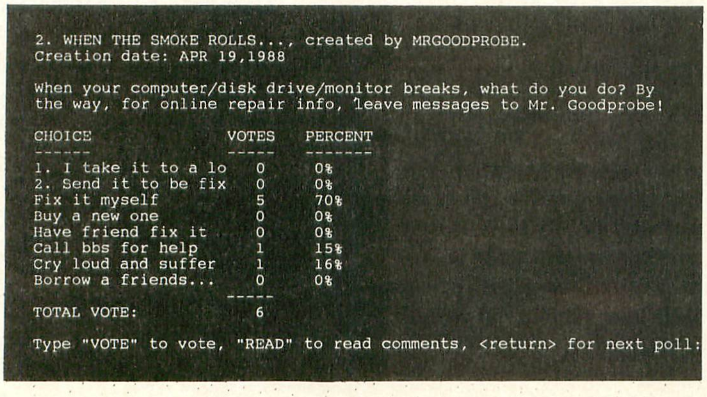
by Michael A. Banks

Keeping in touch is important for those of us online, which is one reason why E-mail, Forum and real-time Conference (described in recent installments of this column) are popular features in ANALOG's Atari Users' Group. These are not the only means of information interchange in the SIG, however. There are two special information pipelines provided by the SIG: announcements and polls.

Finding out what's new: Announcements

As you've probably noticed in the past, "one shot" announcements are occasionally displayed when you enter the Atari Users' Group, after the "ANALOG Computing" logo and before the Atari menu. These are special messages called "Briefs." (No, not the kind you wear!) They're displayed to you only once, unless updated by the group manager. Similar messages may pop up when you enter the Conference, Database, Forum or Poll areas.

Ever wonder where those messages come from—or go? Did you miss some-



```
2. WHEN THE SMOKE ROLLS..., created by MRGOODPROBE.
Creation date: APR 19,1988

When your computer/disk drive/monitor breaks, what do you do? By
the way, for online repair info, leave messages to Mr. Goodprobe!

CHOICE                  VOTES   PERCENT
------                  -----   -------
1. I take it to a lo    0        0%
2. Send it to be fix    0        0%
Fix it myself           5        70%
Buy a new one           0        0%
Have friend fix it      0        0%
Call bbs for help       1        15%
Cry loud and suffer     1        16%
Borrow a friends...     0        0%
                        -----
TOTAL VOTE:             6

Type "VOTE" to vote, "READ" to read comments, <return> for next poll:
```

Figure 1

If you're interested in voicing your opinion in public, or collecting opinions of others, check out the Polls area in the Atari Users' Group.

thing important the last time one scrolled by? Want to find out what else is new in the SIG? It's easy to reread these messages, and see others. Type ANNOUNCEMENTS at the Atari menu. This selection leads you to the Announcements area and this category selection menu:

```
Announcements Menu:

What's New          Recent Software Changes
Conference News     Main Banner Display
Database News       Membership Agreement Display
Forum News          New Member Welcome Display
Poll News           Exit

ANNOUNCE>Which Announcement Category?
```

The selections on the left side of the menu lead to those "Briefs." Select one of those topics, and you'll see an appropriate prompt (such as WHAT'S-NEW®), and DELPHI will tell you how many announcements are available in this particular category. Type SCAN to see a directory of waiting messages, which will be similar to this sample:

```
Contents
--------
  1   3-MAY    BRIEF MAY PROGRAMS
  2  21-APR    BRIEF ST TOPIC DELETIONS
  3  14-APR    BRIEF Z-MAG AND ST REPORT
  4  21-MAR    HAPPY BIRTHDAY
  5   7-MAR    BRIEF PROGRAMMER'S PAL
  6   4-MAR    BRIEF ATTENTION AUTHORS
  7  12-FEB    BRIEF ON-LINE SHOPPING

WHAT'S-NEW>(Scan, Read, "?" or Exit):
```

To read a message, simply type its number. Press RETURN to see the message immediately following. (Type ? to see a menu of other choices.)

(The one-time "Brief" messages contai: the word BRIEF in their titles. The What's-New category contains the Briefs that are displayed when you enter the SIG.)

Of the selections on the right side of the category selection menu, all but one display specific announcements, then return you to the Announcements category selection menu. Thus, "Main Banner Display" displays the SIG's banner (the "ANALOG Computing" logo and accompanying text); "Membership Agreement Display" displays the terms to which you agreed when you joined; and "New Member Welcome Display" displays the welcoming message you read when you first joined the Atari Users' Group.

"Recent Software Changes" leads to a self-directing database which you can search for information about new features in the SIG's operating software.

Take a vote: Poll

If you're interested in voicing your opinion in public, or collecting the opinions of others, check out the Polls area in the Atari Users' Group. This is where you can survey opinions on the computer topics and other matters. You can express your opinion by voting in a poll and adding comments, scanning poll results or creating your own polls. To enter the Poll area, type POLL at the Atari menu. You'll see this menu:

continued on page 44

Make the Delphi Connection!

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

To join Delphi:

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

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

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

```
POLL Menu:

BROWSE through poll results
CREATE a new poll
EDIT your poll comment
EXIT
HELP
LIST poll names
RESULTS with comments
VOTE on a poll

POLL>(BROWSE,CREATE,EDIT,LIST,RESULTS,VOTE)
```

The commands are pretty much self-explanatory, but here's a quick-reference guide to using them. BROWSE is used to view the current voting results of any or all polls. Type BROWSE alone to see the results of all polls in sequence. Each poll's results will be displayed, followed by a prompt asking if you want to vote on the current poll, read the current poll's comments or skip to the next poll. (This feature is a convenient alternate to using VOTE, if you wish to vote on every poll.)

If you type BROWSE followed by the first few letters of the name of a poll you wish to peruse, you'll see the results of that poll only. (To see the names of the available polls, type ? after you type BROWSE.) The option to read, vote or skip will be presented after the poll's voting results are displayed.

CREATE lets you create a poll to sample the opinions of your fellow Atari users. After selecting CREATE, type a descriptive name for your poll (up to 60 characters in length) and select a poll format from among the three available (YES-NO, Degree of agreement or disagreement, or Multiple choice). Next, enter a few lines of text to describe and present your issue to the voters. Enter CTRL-Z, and your poll will be posted.

Use EDIT to amend or add to your comments after you have voted on a poll. You'll be prompted to enter the text that will replace your current comments. Enter CTRL-Z when you're finished, or CTRL-C to abort and leave your comment unchanged.

LIST lists the names of all the polls available.

Use RESULTS if you wish to see the complete results of a specific poll, including all comments. Type RESULTS followed by the name of a poll to see the results of that poll. (If you type RESULTS alone, you will be prompted for the name of a poll.)

To vote on a specific poll, type VOTE. Voting is easy—just follow the online prompts. You will be prompted for the poll name, after which DELPHI will display the text presenting the issue and prompt you for your vote and comments (up to four 80-character lines). Polls show votes by number and percentage, and users may add voting choices to some polls. A typical poll (with results) looks like Figure 1:

Conferences are a great place to share info, get answers to your questions about Atari computers and the Atari Users' Group, and to meet other Atari users.

Database reorganization

If you haven't been online in a while, you'll be surprised to find a slightly altered set of database topics (these changes are reflected by the Forum topics as well). The ST topics are gone—there's now a totally separate group for ST users, hosted by ST-Log Magazine (type ST at the GROUPS menu to visit it). This has left room for our 8-bit databases to "stretch out," both topic- and content-wise. Here's the new lineup:

```
Databases Available Menu:

General Interests          Education
Games & Entertainment      Reviews & News
Telecommunications         Koala Pictures
Utilities                  Current Issue
Sight & Sound              Home use

TOPIC>Which topic?
```

(By the way, if you're not yet online and all this looks interesting, check elsewhere in these pages for a DELPHI online sign-up and membership offer provided especially for ANALOG readers!)

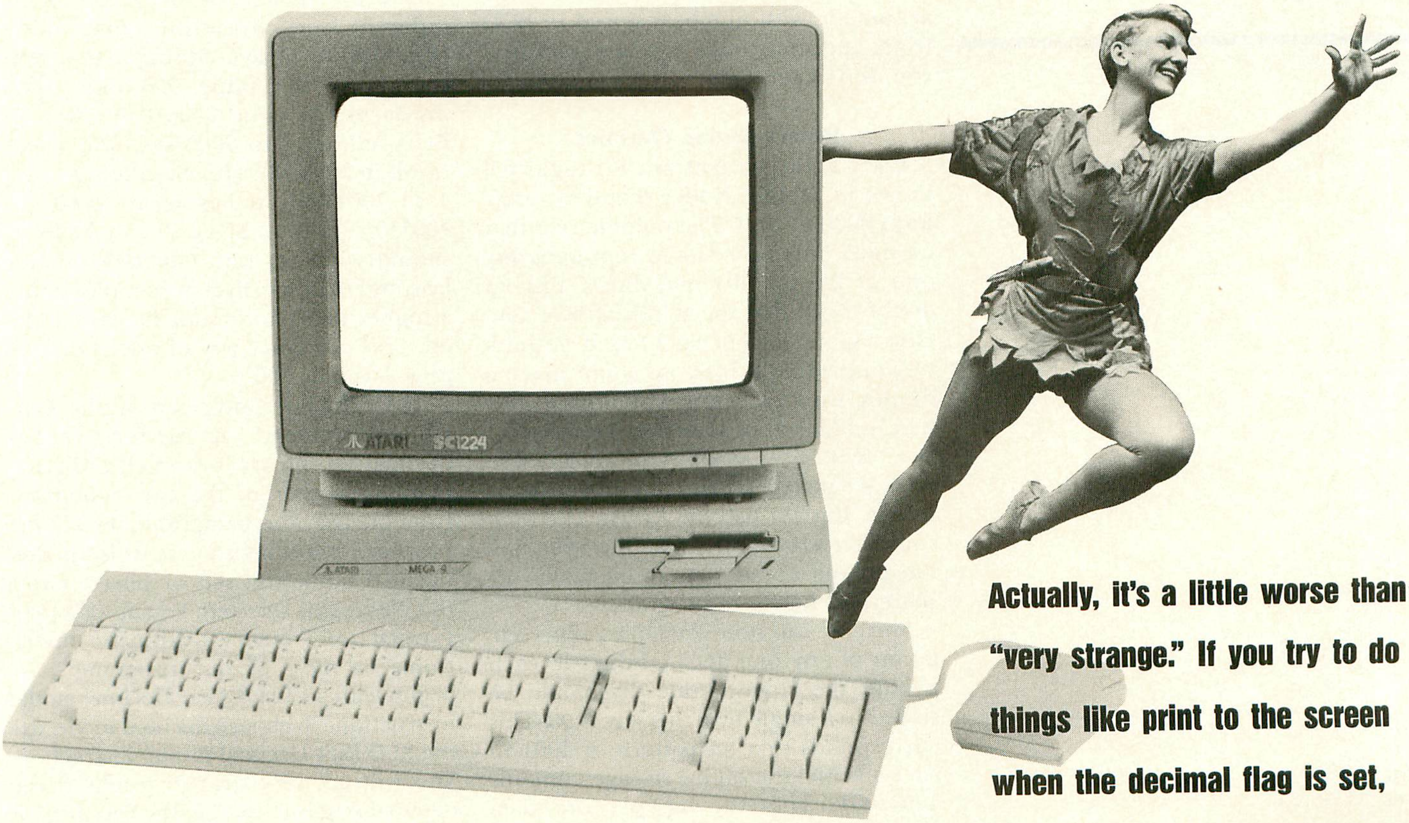
Conference reminder

Don't forget the real-time conference held in the Atari ST Users' Group every Tuesday at 10 p.m., EST. You don't have to wait for Tuesday to roll around to chat with other Atarians. Type WHO when you enter the SIG to see who's in conference (and chances are very good there will be a conference going on). If there are no conferences going on when you enter the SIG, you'll still see a list of members currently in the SIG. You usually start your own conference by going to the conference area and typing /PAGE followed by one or more of the member-names, separated by commas, listed when you typed WHO. (Example: /PAGE KZIN, ANALOG4.)

Conferences are a great place to share information, get answers to your questions about Atari computers and the Atari Users' Group, and to meet other Atari users. That's it for now. See you next month with more tips and a few surprises. See you online!

In addition to having published science fiction novels and books on rocketry, Michael A. Banks is the author of DELPHI: The Official Guide *and* The Modem Reference—*both from Brady Books/Simon & Schuster. To order* DELPHI: The Official Guide, *type GO GUIDE at any SIG prompt. You can contact Banks on DELPHI by sending E-mail to membername KZIN.*

BOOT CAMP



Actually, it's a little worse than "very strange." If you try to do things like print to the screen when the decimal flag is set, you can wind up in computer never-never land.

The rather cryptic title of this article can be translated as "Binary-Coded Decimal and You." Last month we discussed some routines for interconverting strings of numeric ASCII characters and their binary representations as integers.

I know you're eager to dive into the sample program for today, but I'm going to hold you back a little longer.

BCD N U

This time we tackle another commonly used method for storing numbers in computers: binary-coded decimal, or BCD for short. After I explain the BCD representation, we'll see how to change an ASCII string into a BCD storage format. I also have some examples of how to do arithmetic with numbers stored in BCD from, and some traps you can fall into if you don't keep your wits about you.

Binary-Coded Decimal

Look at the bit patterns for digits 0-9 shown in Table 1. Notice that they range form 0000 to 1001. The point here is that we need only four bits to represent any one of the ten decimal digits. You no doubt recall that the standard byte contains a grand total of eight bits. If we think of subdividing a byte, we could make a duplex with each unit containing four bits. A 4-bit unit is sometimes referred to as a "nybble" (a small byte—get it?). I've seen it spelled more conventionally as nibble, but I'll use the "y" so the non-computer whizzes who read this will think I'm talking about something really obscure and hence important.

Since we can store the binary representation of any one decimal digit in each nybble, the largest value that could be stored in a single byte this way is 99. This corresponds to a bit pattern of 1001 in each nybble; the entire storage contains 10011001. This two-digit-per-byte data storage method is the infamous binary-coded decimal.

There are two ways to interpret a bit pattern of 10011001. In pure hexadecimal, it is $99, which corresponds to decimal 153. But if we think of it as two decimal digits, that bit pattern means decimal 99. We need some way to tell the computer which meaning we have in mind at any given time.

Doing arithmetic on BCD numbers is different from processing binary numbers also. In binary, adding 1 to a byte containing the value 00001001 ($09) produces the value 00001010 ($0A). In BCD, adding 1 to 00001001 (09) would result in 00010000 (10). Similarly, adding 1 to 10011001 in hex terms produces 10011010 ($99 to $9A). But in BCD we should wind up with 00000000 in this byte and the carry flag set to indicate that a higher order byte must be incremented. This is a fancy way of saying that 99 plus 1 equals 100.

The 6502 microprocessor in the Atari 8-bit computers can perform either decimal or binary arithmetic, thereby handling either of the two conditions from the previous paragraph. Bit 3 in the processor status register controls whether decimal mode (bit set) or binary mode (bit cleared) is selected. So far, we've performed only binary arithmetic operations, so most of our programs have begun with a CLD (CLear Decimal mode) instruction. To choose decimal mode, use the SED (SEt Decimal mode) instruction. You will get very strange results if the decimal flag isn't set the way you think it is; so it's always a good idea to explicitly select the desired mode.

Actually, it's a little worse than "very strange." If you try to do things like print to the screen when the decimal flag is set, you can wind up in computer never-never land, with a coldstart being the only way back. Always clear decimal mode with the CLD instruction when you've finished your decimal arithmetic operations.

Interconverting ASCII and BCD

Today's example is similar in format to last month's discussion of how to interconnect ASCII and binary storage formats for integers. Listing 1 contains two macros in MAC/65 format that should be appended to your MACRO.LIB file using the line numbers shown. Similarly, Listing 2 contains a pair of subroutines called by these macros; append Listing 2 to your SUBS.LIB file.

My MACRO.LIB file is now an even 100 single-density sectors long. If you're using a RAM disk for assemblies, this is only a minor nuisance. However, reading a file that large from a physical disk each time you do an assembly takes a long time, and it doesn't do your disk drive any good. You may want to think about splitting the MACRO.LIB file into several smaller library files, perhaps grouped logically by function. You can do this any way you like, and just .INCLUDE the ones you need for your current project. Be sure to keep the equates needed by the macros accessible (and unduplicated). In fact, you might just collect all the equates into a separate EQUATES.LIB file. I'll leave the details of the MACRO.LIB dissection to each of you.

The two new macros, and their corresponding subroutines, are named ASC2BCD and BCD2ASC. These complement the ASC2INT and INT2ASC routines from the previous Boot Camp. ASC2BCD takes a string of up to six numeric characters and converts it into a three-byte BCD number. Not surprisingly, BCD2ASC takes a three-byte BCD number and transforms it right back into a printable ASCII string. The macros themselves do some error checking and use parameters to handle ASCII strings and BCD numbers stored at any address, while their subroutine partners do most of the real work.

ASCII to BCD

We'll start at the beginning. Please turn your attention to the ASC2BCD macro in Listing 1. ASC2BCD expects two parameters, the address of the ASCII string to convert, and the address where the resulting three-byte BCD number is to be stashed. An error message appears if the number of parameters is not two (Lines 8130-8140).

This macro begins just like the ASC2INT macro from last time. Lines 8160-8220 copy the characters from the input string at the address specified in parameter %1 to a work address labeled ASCII. The ASCII address was defined in Listing 1 from last month as $0690. The input string must terminate with an end-of-line character ($9B). In our sample program today, the numeric string to convert is read from the keyboard using our INPUT macro, which automatically guarantees that an EOL character will be present.

Line 8230 calls the VALIDASC subroutine from last month, which makes sure that all the characters in the string are in fact digits in the range 0-9. If not, the carry flag is set in the subroutine to indicate an error. Line 8240 handles this condition by simply short-circuiting around the rest of the macro code. The main program that invoked this macro handles the error condition, as we'll see a little later. I don't have any provision for handling negative numbers.

Subroutine VALIDASC retains only the lower four bits from the ASCII character. That is, if you entered the digit "7" at the keyboard, the ASCII value is $37, and VALIDASC changes this back into a plain "7" after confirming that it is a legal entry.

After the conversion, the BCD number resides at a work location called NUM, defined as $0696 last month. Lines 8280-8350 copy the BCD result to the desired output address specified in parameter %2.

After all this monkey business, we wind up with a string of characters at address ASCII which looks exactly like what we typed at the keyboard. Let's pretend we typed the number "7239." Our goal is to convert the input string, now typed in five bytes like this (showing both nybbles in each byte):

```
07 02 03 09 9B
```

into BCD format stored in three bytes like this:

```
00 72 39
```

Notice that this numeric storage format is different from the low-byte/high-byte format used for binary integers.

Line 8250 of Listing 1 calls the ASC2BCD subroutine in Listing 2 to handle the details of the conversion. Now please direct your attention to Listing 2.

First we need to know how many input digits to convert to BCD. When we get to today's example in Listing 3, you'll see that

The non-computer whizzes who read this will think I'm talking about something really obscure and important.

The 6502 microprocessor knows how to do arithmetic on numbers stored in both binary and decimal modes. There are a few differences.

the value we need was stored in a work address labeled CHARCTR. The value in CHARCTR includes the EOL character, so it is one larger than the actual number of digits in the input number. Lines 4720-4740 of Listing 2 set up the X-register as an offset for the characters for the BCD bytes. Here's the conversion plan.

We'll begin by zeroing the three bytes where the BCD result will be stored. Lines 4760-4790 handle this task. The conversion step will begin with the least significant (rightmost) digit in the entered ASCII string. This number becomes the low-order nybble in the least significant (rightmost) BCD byte (Lines 4810-4820).

If the ASCII string to be converted contains an odd number of digits, the high-order nybble of one of the the BCD bytes will remain zero. This should be apparent to you. Line 4830 in Listing 2 points to the next ASCII character, which is destined to go into the high-order nybble of the current BCD byte. Line 4840 checks to see if we've reached the end of the ASCII string yet. If not, fetch the contents of the next ASCII byte (Line 4850). Remember that we've already changed this from the original ASCII value to the value of the digit itself (e.g., $37 was changed to 7).

Lines 4860-4890 shift this number four bits to the left, thereby relocating it to the high-order nybble of the accumulator. Line 4900 combines the result with the low-order nybble from the previous ASCII digit, and the completed BCD byte (now containing two digits) is stored back where it belongs (Line 4910). Lines 4920-4960 check to see if we're done with the ASCII string yet and loop back to continue if not.

This discussion is a little confusing. You might find it illuminating to use your debugger to trace through a stepwise processing of a sample input number after entering Listing 3, and see how the ASC2BCD subroutine does its thing.

I know you're eager to dive into the sample program for today, but I'm going to hold you back a little longer. The sample program goes through a bunch of BCD arithmetic examples, which we'll get to in a moment. But while the details of the ASCII-to-BCD conversion are fresh in your mind, I want to tackle the reverse process. Bear with me.

BCD to ASCII

I'm sure you can figure out what we must do to change a number stored in BCD format into a printable ASCII string. There are two basic steps. First, split the high and low nybbles of each byte in the BCD number into separate bytes in the output string. And second, convert the digits into their corresponding ASCII values. As an additional cosmetic nicety, we'll also convert any leading zeros to leading blanks.

The BCD2ASC macro begins at line 8540 of Listing 1, and the complementary BCD2ASC subroutine starts at line 5150 of Listing 2. The macro again requires two parameters, the address of the BCD number to be converted and the address where the resulting ASCII string should be stored. Three bytes at address NUM and six bytes at address ASCII are again used as work locations. Lines 8580-8630 of the macro copy the contents of the BCD number into work location NUM. The subroutine BCD2ASC is then called. Lines 8650-8710 then copy the resulting string from address ASCII into the address specified in parameter %2. Lines 8720-8740 tack an EOL character on the end so the string can be printed.

This time the conversion proceeds from left to right (high order to low order). In the BCD2ASC subroutine, I've set aside one byte (called ZEROBLANK, defined in Line 5570 of Listing 2) to indicate whether a zero digit is to be represented as a zero ASCII character ($30) or as a blank ($20). ZEROBLANK is initially set to $20 in Lines 5160-5170 so as to print leading zeros as blanks. However, as soon as a non-zero digit is encountered, ZEROBLANK is set to $30 so that zeros in the middle of the number appear properly.

Line 5210 gets the first (leftmost) BCD digit, which is saved temporarily on the program stack (Line 5220). (The X-register is used as an offset into the BCD number, and the Y-register as an offset into the ASCII string.) The high nybble is moved into the low nybble with a series of four right shifts; this is the opposite of the four ASLs we used in the ASC2BCD process. If the result is a zero, Lines 5290-5300 store the current value of ZEROBLANK into the next position in the output string. If the digit is not a zero, Lines 5330-5360 convert the digit to ASCII by adding $30 to it, store the result in the output string, and set the value of ZEROBLANK to an ASCII zero.

Lines 5380-5400 point to the next output character, retrieve the BCD byte, and strip off the four most significant bits. This leaves just the low nybble, which is the second of the two digits in the BCD byte. Then the same activities are performed as for the first digit, depending on whether the digit is a zero or not (Lines 5410-5440). After processing all three BCD bytes, we wind up with a printable ASCII string. *Voila.*

BCD Arithmetic

Now for the interesting part. The 6502 microprocessor knows how to do arithmetic on numbers stored in both binary and decimal modes. There are a few differences you should keep in mind, and Listing 3 will help you out.

The program in Listing 3 asks you to enter a number up to six digits long, verifies that you entered only digits, converts the string of ASCII characters to a three-byte BCD number, and performs some representative arithmetic operations in both BCD and decimal mode. The results from each operation are printed on the screen in a little table. Let's walk though Listing 3 now.

Line 160 pulls in the macros from our library file. Be sure to change this statement if you are using a real disk drive instead of the D8: RAM disk, or if you segmented the MACRO.LIB file as I suggested earlier. Some work variables are defined in Lines 280-310. BCD is the home of the BCD number. CHARCTR contains the number of ASCII characters you entered (including the EOL character). INBUF is an input buffer for the number you enter, and OUTBUF is an output buffer for the printable ASCII result.

As usual, the executable code begins at address $5000. Lines 520-590 clear decimal mode (for now), clear the screen, prompt you to enter a number, store the number at INBUF, and store the number of characters you entered at CHARCTR. Lines 650-760 set up the column and row headings for the output table; the text strings to be printed are stored in Lines 2350-2480. Line 830 converts the input string in INBUF to BCD representation at address BCD. We'll have to repeat this after each sample calculation to make sure the BCD number starts out the same way every time. If there's an error in the BCD conversion, the carry flag will be set and the program terminates due to Lines 840-850.

The program has four sample calculations: increment the lowest BCD byte; add decimal 25 to the BCD number; add hex 25 to the BCD number; and add the contents of the middle BCD byte to the whole BCD number. Each calculation is done in both binary and decimal modes. I suggest you try this program with several sample entries, to see what happens. Press return after the output appears to try another number. Four interesting numbers to try are 0, 1234, 999 and 7239. On the off-chance that you don't really want to spend the time typing in the listings, I've included tables to the output you would see for each of these test cases (Tables 2-5).

Incrementing

The simplest arithmetic operation you can do in 6502 assembly language is to increment the contents of a byte. The opcode for this is, of course, INC. Lines 980-1030 increment the least significant byte of the BCD number (BCD$2) in binary mode, and Lines 1070-1130 do the same in decimal mode. Notice the SED instruction in Line 1070 to set the decimal mode flag. Line 1090 clears the flag immediately after the arithmetic is done to avoid problems with subsequent operations. After each operation, the resulting value at address BCD is converted to ASCII at address OUTBUF and printed on the screen.

The first line of Table 2 shows that INC works just fine when the target number is 0, giving the expected result of 1 in each case. Table 3 shows that INC works fine for the number 1234 also. But wait! An input value of 999 gives the bizarre result of 99:. Something similar happens in Table 5 with 7239. How can this be?

Well, for Table 4, BCD + 2 contains "99," which is incremented to "9A." Converting to ASCII gives two bytes, containing $39 (prints as a 9) and $3A (prints as a colon, :). Hmmmm. We really wanted the BCD number "99" to increment to "00," setting the carry flag to indicate that the next higher order byte should also be incremented. It appears that the INC instruction has the same effect in decimal mode as it does in binary. Moral: Don't use INC to add 1 to a BCD number. Instead, go through the cumbersome motions of actually adding 1.

The simplest arithmetic operation you can do in 6502 assembly language is to increment the contents of a byte. The opcode for this is, of course, INC.

Adding 25

Okay, so let's add something to a BCD number. Lines 1210-1330 add an immediate value of 25 decimal to the BCD number you entered in binary mode. Lines 1370-1490 do the same in decimal mode. These routines use a subroutine called INCREMBCD (Lines 2630-2750 of Listing 3) to handle the case where the carry flag is set after the addition, so that the next higher order byte must be incremented (by adding 1 to it, of course). This in itself might reset the carry flag, so that the highest order BCD byte also has to be incremented. These operations should make sense to you by now.

Let's do it. Now look at the second output line in Tables 2-5 to see how our sample numbers respond. A problem is immediately apparent in Table 2. Adding 25 to 0 gave 19, not 25. Why? Well, the hex equivalent of 25 is $19. Last month we added decimal 25 (using an ADC #25 instruction) to a number stored as binary, went through the binary-to-ASCII conversion, and got the right answer. But we've scrambled our conventions here. We added a decimal number (stored internally as hex, of course) to a BCD number, using binary mode, and converted the presumed BCD result to ASCII for printing. It's not surprising that the wrong result shows up.

The same thing happens with all the other input numbers. The weird characters in Tables 3 and 4 appear again because the addition results have gone out of the legal 0-9 BCD range, into values which print as other ASCII characters. Check out your table of hex codes for ASCII characters if you don't believe me.

The correct method for adding an immediate value to a stored BCD number is to use the desired decimal digits for the immediate number, but tell the computer that it's a hex number.

Adding $25

The correct method for adding an immediate value to a stored BCD number is to use the desired decimal digits for the immediate number, but tell the computer that it's a hex number. That is, to add decimal 25 to a BCD number, use an ADC #$25 instruction. The third output line in each table shows that this approach does indeed produce the result of adding 25. The initial entry of 1234 fortuitously gives the correct answer in either binary or decimal modes (Table 3). However, an entry of 999 works right only in decimal mode (Table 4). In binary mode, the computer sets the carry flag when the byte's contents exceed $FF, not $99 as it does in decimal.

Adding Two Stored Numbers

The final line of each table shows the result of adding the middle byte of the BCD number to the entire number, just to show how things work when you add together two stored values. Table 2 correctly shows no output for this line, since 0 + 0 = 0, which we print as all blanks. For the other three cases, the correct answer is always obtained when the decimal flag is set, and only in some cases (e.g., Table 4) when in binary mode.

So What?

Now you know more about binary-coded decimal than you ever dreamed possible. But why should you care? Burrow back through your archives to the yellowed, brittle pages of ANALOG #43. The *Boot Camp* in that issue discussed floating point numbers and mathematics in the Atari. Floating point numbers use the BCD representation as a compact way to stuff several digits of precision into a minimum number (six) of bytes. A special notation is used to keep track of the decimal point, exponent and negative sign in floating point numbers. BCD turns out to be a pretty efficient storage format for base-10-type numbers, and most computers use some form of BCD for floating point storage.

You may recall that the main alternative character-coding method in common computer use is called EBCDIC (pronounced ebb-see-dick), used mainly by IBM mainframe computers. That acronym stands for "Extended Binary-Coded Decimal Interchange Code." See? You can run, but you just can't hide from binary-coded decimal.

There's another advantage. In today's example program, we converted BCD numbers to ASCII strings and printed them on the screen. However, you could also take each BCD digit, convert it to the Atari internal character code by ANDing it with $10 (as opposed to $30, which converts it to ASCII), and poke the result directly into the screen RAM for the current display. This is simpler and faster than printing on the screen, and the visual result is the same. A good example of this technique can be found in James Hague's *Streamliner* from ANALOG #56. See the right column of page 37 in that issue.

Promise

I promise: no more hard-core computing for awhile. We'll get back to some graphics (wanna know how to draw circles?), sound effects and real-time clocks (how about a metronome program?), and maybe even the kernel of an adventure program; a simple vocabulary parser. Stay tuned.

Table 1. ASCII Codes for Decimal Characters

Character	ASCII Values	Binary Value
0	$30	0000
1	$31	0001
2	$32	0010
3	$33	0011
4	$34	0100
5	$35	0101
6	$36	0110
7	$37	0111
8	$38	1000
9	$39	1001

Table 2. Sample Output From Input Number of 0

	Binary Mode	*Decimal Mode*
INC	1	1
Add 25	19	19
Add $25	25	25
Add 2nd Byte		

Table 4. Sample Output From Input Number of 999

	Binary Mode	*Decimal Mode*
INC	99:	99:
Add 25	9;2	1018
Add $25	9;>	1024
Add 2nd Byte	9:2	1008

Table 3. Sample Output From Input Number of 1234

	Binary Mode	*Decimal Mode*
INC	1235	1235
Add 25	124=	1253
Add $25	1259	1259
Add 2nd Byte	1246	1246

Table 5. Sample Output From Input Number of 7239

	Binary Mode	*Decimal Mode*
INC	723:	723:
Add 25	7252	7258
Add $25	725>	7264
Add 2nd Byte	72:;	7311

LISTING 1: ASSEMBLY

```
8010 ;
8020 ;*******************************
8030 ;
8040 ;ASC2BCD macro
8050 ;
8060 ;Usage:  ASC2BCD chars,number
8070 ;'chars' is address of ASCII
8080 ; string to convert,ending w/ EOL
8090 ;'number' is address of BCD
8100 ; representation of the string
8110 ;
8120     .MACRO ASC2BCD
8130       .IF %0<>2
8140       .ERROR "Error in ASC2BCD"
8150       .ELSE
8160       LDX #255
8170 @ASCLOOP2
8180       INX
8190       LDA %1,X
8200       STA ASCII,X
8210       CMP #EOL
8220       BNE @ASCLOOP2
8230       JSR VALIDASC
8240       BCS @DONE2
8250       JSR ASC2BCD
8260       BCS @BCDERROR
8270       LDX #0
8280 @ASCLOOP3
8290       LDA NUM,X
8300       STA %2,X
8310       INX
8320       CPX #3
8330       BNE @ASCLOOP3
8340       CLC
8350       BCC @DONE2
8360 @BCDERROR
8370        PRINT  CONVERTMSG2
8380       SEC
8390 @DONE2
8400       .ENDIF
8410     .ENDM
8420 ;
8430 ;*******************************
8440 ;
8450 ;BCD2ASC macro
8460 ;
8470 ;Usage:  BCD2ASC number,chars
8480 ;'number' is address of BCD
8490 ; number to convert
8510 ;'chars' is address of resulting
8520 ; ASCII string, ending with EOL
8530 ;
8540     .MACRO BCD2ASC
8550       .IF %0<>2
8560       .ERROR "Error in BCD2ASC"
8570       .ELSE
8580       LDA %1
8590       STA NUM
8600       LDA %1+1
8610       STA NUM+1
8620       LDA %1+2
8630       STA NUM+2
8640       JSR BCD2ASC
8650       LDX #255
8660 @BCDLOOP
8670       INX
8680       LDA ASCII,X
8690       STA %2,X
8700       CPX #5
8710       BNE @BCDLOOP
8720       INX
8730       LDA #EOL
8740       STA %2,X
8750       .ENDIF
8760     .ENDM
```

LISTING 2: ASSEMBLY

```
4600 ;
4610 ;*******************************
4620 ;
4630 ;subroutine ASC2BCD
4640 ;called by ASC2BCD macro
4650 ;
4660 ;converts string of ASCII digits
4670 ;at address ASCII to a 3-byte
4680 ;binary-coded decimal represen-
4690 ;tation at address NUM
4700 ;
4710 ASC2BCD
4720     LDX CHARCTR ;how many chars
4730     DEX         ;to convert?
4740     DEX
```

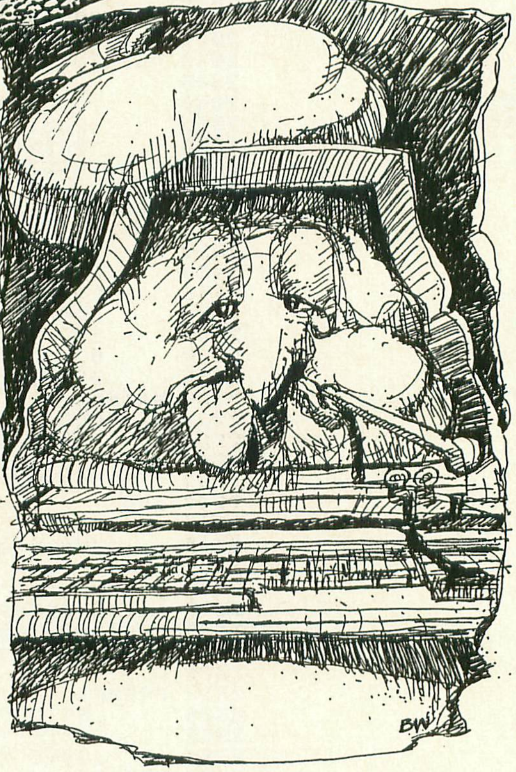
```
4750      LDY #2
4760      LDA #0       ;zero 3 bytes
4770      STA NUM      ;where BCD value
4780      STA NUM+1    ;will go
4790      STA NUM+2
4800 NXTDIG
4810      LDA ASCII,X ;get next char
4820      STA NUM,Y    ;low BCD digit
4830      DEX          ;point to next
4840      BMI BCDDONE ;done yet?
4850      LDA ASCII,X ;get new char
4860      ASL A        ;shift into
4870      ASL A        ;high nybble
4880      ASL A
4890      ASL A
4900      ORA NUM,Y    ;becomes high
4910      STA NUM,Y    ;BCD digit
4920      DEX          ;point to prev.
4930      BMI BCDDONE ;done yet?
4940      DEY          ;point to next
4950      CLC          ;BCD digit
4960      BCC NXTDIG   ;go get it
4970 BCDDONE
4980      CLC          ;all done, so
4990      RTS          ;leave
5000 CONVERTMSG2
5010      .BYTE "ASCII to BCD con"
5020      .BYTE "version error",EOL
5030 ;
5040 ;******************************
5050 ;
5060 ;subroutine BCD2ASC
5070 ;called by BCD2ASC macro
5080 ;
5090 ;converts 3-byte BCD number at
5100 ;address NUM to a 6-byte ASCII
5110 ;string at address ASCII
5120 ;leading zeros are changed to
5130 ;leading blanks
5140 ;
5150 BCD2ASC
5160      LDA #$20     ;init leading
5170      STA ZEROBLANK ;char to blank
5180      LDX #0       ;pointer to digit
5190      LDY #0       ;pointer to char
5200 NXTDIG2
5210      LDA NUM,X    ;get 1st digit
5220      PHA          ;stash on stack
5230      CLC
5240      LSR A        ;move high nybble
5250      LSR A        ;into low nybble
5260      LSR A
5270      LSR A
5280      BNE NONZERO1 ;equal to 0?
5290      LDA ZEROBLANK ;yes, set to
5300      STA ASCII,Y ;leading char
5310      BPL DOLOW    ;do low half
5320 NONZERO1
5330      ORA #$30     ;change to ASCII
5340      STA ASCII,Y ;add to string
5350      LDA #$30     ;set leading
5360      STA ZEROBLANK ;char to '0'
5370 DOLOW
5380      INY          ;aim at next char
5390      PLA          ;get BCD digit
5400      AND #$0F     ;keep low nybble
5410      BNE NONZERO2 ;equal to 0?
5420      LDA ZEROBLANK ;yes, set to
5430      STA ASCII,Y ;leading char
5440      BPL BCDDONE2 ;all done
5450 NONZERO2
5460      ORA #$30     ;conver to ASCII
5470      STA ASCII,Y ;add to string
5480      LDA #$30     ;set leading char
5490      STA ZEROBLANK ;to zero
5500 BCDDONE2
5510      INY          ;point to next
5520      INX          ;digit and char
5530      CPX #3       ;done 3 digits?
5540      BNE NXTDIG2 ;no, continue
5550      CLC          ;yes, all done
5560      RTS          ;exit
5570 ZEROBLANK .DS 1
```

LISTING 3: ASSEMBLY

```
0100 ;Example 1. Interconverting ASCII
0110 ;strings and BCD numbers
0120 ;
0130 ;by Karl E. Wiegers
0140 ;
0150      .OPT NO LIST,OBJ
0160      .INCLUDE #D8:MACRO.LIB
0170 ;
0180 ;------------------------------
0190 ;
0200 ;store some work variables at
0210 ;$4FE0 so you can examine them
0220 ;if you like
0230 ;
0240 ;------------------------------
0250 ;
0260      *=  $4FE0
0270 ;
0280 BCD .DS 3
0290 CHARCTR .DS 1
0300 INBUF .DS 7
0310 OUTBUF .DS 7
0320 ;
0330 ;------------------------------
0340 ;
0350 ;       PROGRAM STARTS HERE
0360 ;
0370 ;You'll be prompted to enter a
0380 ;number with 1-6 digits.  This
0390 ;is stored at address INBUF.
0400 ;The BCD number produced is
0410 ;stored in 3 bytes starting at
0420 ;address BCD.  Then several
0430 ;arithmetic operations are done
0440 ;in both binary and decimal mode,
0450 ;and a table of results is
0460 ;printed out.
0470 ;
0480 ;------------------------------
0490 ;
0500      *=  $5000
0510 ;
0515 START
0520      CLD           ;binary mode!
0530      JSR CLS       ;clear screen
0540       PRINT  PROMPT ;get input
0550       POSITION  2,2 ;number
0560       INPUT  0,INBUF
0570      LDX #$00      ;get number of
0580      LDA ICBLL,X ;chars entered
0590      STA CHARCTR
0600 ;
0610 ;------------------------------
0620 ; lay out the table of results
0630 ;------------------------------
0640 ;
0650       POSITION  12,5
0660       PRINT  TITLE
0670       POSITION  12,6
0680       PRINT  HYPHENS
0690       POSITION  2,8
0700       PRINT  INCRE
0710       POSITION  2,10
0720       PRINT  DEC25
0730       POSITION  2,12
0740       PRINT  HEX25
0750       POSITION  2,14
0760       PRINT  ADDBYTE
0770 ;
0780 ;------------------------------
0790 ;convert string to BCD, abort if
0800 ;have a conversion problem
0810 ;------------------------------
0820 ;
0830       ASC2BCD  INBUF,BCD
0840      BCC NOPROBLEM
0850      JMP END
0860 ;
0870 ;------------------------------
0880 ;First line:  increment the BCD
0890 ;number in binary and decimal
0900 ;modes; be sure to set back to
0910 ;binary before doing anything
0920 ;else!
0930 ;reconvert from input string to
0940 ;BCD after each operation
0950 ;------------------------------
0960 ;
0970 NOPROBLEM
0980      CLD
0990      INC BCD+2
```

```
1000       BCD2ASC  BCD,OUTBUF
1010       POSITION  14,8
1020       PRINT  OUTBUF
1030       ASC2BCD  INBUF,BCD
1040 ;
1050 ;increment in decimal mode
1060 ;
1070      SED
1080      INC BCD+2
1090      CLD
1100       BCD2ASC  BCD,OUTBUF
1110       POSITION  29,8
1120       PRINT  OUTBUF
1130       ASC2BCD  INBUF,BCD
1140 ;
1150 ;------------------------------
1160 ;Second line:  add 25 to the BCD
1170 ;number in binary and decimal
1180 ;modes
1190 ;------------------------------
1200 ;
1210      CLD
1220      CLC
1230      LDA BCD+2
1240      ADC #25
1250      STA BCD+2
1260      BCC NOINC1
1270      JSR INCREMBCD
1280 NOINC1
1290      CLD
1300       BCD2ASC  BCD,OUTBUF
1310       POSITION  14,10
1320       PRINT  OUTBUF
1330       ASC2BCD  INBUF,BCD
1340 ;
1350 ;add 25 in decimal mode
1360 ;
1370      SED
1380      CLC
1390      LDA BCD+2
1400      ADC #25
1410      STA BCD+2
1420      BCC NOINC2
1430      JSR INCREMBCD
1440 NOINC2
1450      CLD
1460       BCD2ASC  BCD,OUTBUF
1470       POSITION  29,10
1480       PRINT  OUTBUF
1490       ASC2BCD  INBUF,BCD
1500 ;
1510 ;-------------------------------
1520 ;Third line:  add hexadecimal 25
1530 ;to the BCD number in binary and
1540 ;binary modes
1550 ;-------------------------------
1560 ;
1570      CLD
1580      CLC
1590      LDA BCD+2
1600      ADC #$25
1610      STA BCD+2
1620      BCC NOINC3
1630      JSR INCREMBCD
1640 NOINC3
1650      CLD
1660       BCD2ASC  BCD,OUTBUF
1670       POSITION  14,12
1680       PRINT  OUTBUF
1690       ASC2BCD  INBUF,BCD
1700 ;
1710 ;add $25 in decimal mode
1720 ;
1730      SED
1740      CLC
1750      LDA BCD+2
1760      ADC #$25
1770      STA BCD+2
1780      BCC NOINC4
1790      JSR INCREMBCD
1800 NOINC4
1810      CLD
1820       BCD2ASC  BCD,OUTBUF
1830       POSITION  29,12
1840       PRINT  OUTBUF
1850       ASC2BCD  INBUF,BCD
1860 ;
1870 ;-------------------------------
1880 ;Fourth line:  add second byte
1890 ;of BCD number to the entire
1900 ;number, in binary and decimal
1910 ;modes.  If number was 1-2 digits
1920 ;long, will just add zero
1930 ;-------------------------------
1940 ;
1950 ;ADD 2ND BYTE TO 3RD - BINARY
1960      CLD
1970      CLC
1980      LDA BCD+1
1990      ADC BCD+2
2000      STA BCD+2
2010      BCC NOINC5
2020      JSR INCREMBCD
2030 NOINC5
2040      CLD
2050       BCD2ASC  BCD,OUTBUF
2060       POSITION  14,14
2070       PRINT  OUTBUF
2080       ASC2BCD  INBUF,BCD
2090 ;
2100 ;add 2nd byte to total - decimal
2110 ;
2120      SED
2130      CLC
2140      LDA BCD+1
2150      ADC BCD+2
2160      STA BCD+2
2170      BCC NOINC6
2180      JSR INCREMBCD
2190 NOINC6
2200      CLD
2210       BCD2ASC  BCD,OUTBUF
2220       POSITION  29,14
2230       PRINT  OUTBUF
2240 END
2244       INPUT  0,INBUF
2248      JMP START
2250 ;
2260 ;------------------------------
2270 ;text lines for prompt and for
2280 ;output table
2290 ;------------------------------
2300 ;
2310 PROMPT
2320      .BYTE "Enter a number "
2330      .BYTE "up to 6 digits "
2340      .BYTE "long:",EOL
2350 TITLE
2360      .BYTE "Binary Mode    "
2370      .BYTE "Decimal Mode",EOL
2380 HYPHENS
2390      .BYTE "-----------    "
2400      .BYTE "------------",EOL
2410 INCRE
2420      .BYTE "INC",EOL
2430 DEC25
2440      .BYTE "Add 25",EOL
2450 HEX25
2460      .BYTE "Add $25",EOL
2470 ADDBYTE
2480      .BYTE "Add 2nd byte",EOL
2490 ;
2500 ;-------------------------------
2510 ;don't forget the subroutines!
2520 ;-------------------------------
2530 ;
2540      .INCLUDE #D8:SUBS.LIB
2550 ;
2560 ;*****************************
2570 ;subroutine do handle carry if
2580 ;adding to the third BCD byte
2590 ;went above 99; can't increment,
2600 ;so must add 1 to higher order
2610 ;bytes as needed
2620 ;
2630 INCREMBCD
2640      SED           ;still in decimal
2650      CLC
2660      LDA #1        ;add 1 to second
2670      ADC BCD+1     ;BCD byte
2680      STA BCD+1     ;and store
2690      BCC NOMOREINC ;cause carry?
2700      CLC
2710      LDA #1        ;yes, so add 1 to
2720      ADC BCD       ;first BCD byte
2730      STA BCD       ;and store
2740 NOMOREINC
2750      RTS           ;all done, exit
```

ST-Notes



In the age of technological marvels, even the common flu can get your ST down.

New Atari Corp. documentation

If you were one of the lucky developers to purchase the Atari ST Developers Kit for $300, occasionally you will receive mailings of new information from Atari. Recently, Atari mailed detailed information on the Mega ST system's internal expansion bus. The expansion bus was included inside the Mega for Atari and third-party developers to create additional support products that can easily be interfaced with the Mega's hardware.

At the fall 1987 COMDEX show, Atari's research and development team indicated that several expansion bus products are in the works for the Mega computer. A Local-Area-Network card (LAN) is a natural product for the Mega to give businesses the ability to share information from a central hard disk or use a common printer.

What makes the Mega expansion bus documentation so interesting is that it was printed on a laser printer and is easy to understand with complete explanations. This marks a decided change in Atari's previous documentation efforts. Atari documentation is normally poorly written and sloppy in presentation. With evidence of better documentation, Atari might be signaling the industry that it is cleaning up its internal problems and working towards a bright future.

Aaaahhhh-chooooo!

Has your computer been feeling run down? Maybe even a little feverish? It might be due to a computer virus. When the idea of a computer virus was first presented, it seemed like a joke. How could your computer get-sick? But we have found in the age of technological marvels, even the common flu can get your ST down.

A virus is a program that unscrupulous programmers place on boot ST floppy disks. When you first power-up your ST, a program recorded in the "boot-sector" of the diskette in drive A is loaded into your ST's memory. The program initializes some memory and variables and then launches the main operating system stored in ROM.

Viruses change the "boot-sector" by recording their own style of boot program. Instead of the usual coldstart process, the virus will install itself into your system's memory, and then proceed with the initialization as normal. Your ST turns on and the friendly GEM Desktop appears.

Not all viruses are alike. Their side effects determine the destructiveness of a virus. For example, one virus will remain dormant for a couple of hours. When you try to save a file to your floppy disk, the virus will ocassionally cause the save operation to save incorrect data. When you thought you were saving a letter to your mother, the virus was saving a bunch of random characters.

Viruses can spread, because they can also change the boot-sector of the other disks you insert into your disk drive after the virus has been loaded into your computer system. So, once a virus has been detected, it is important to immunize all of your other diskettes.

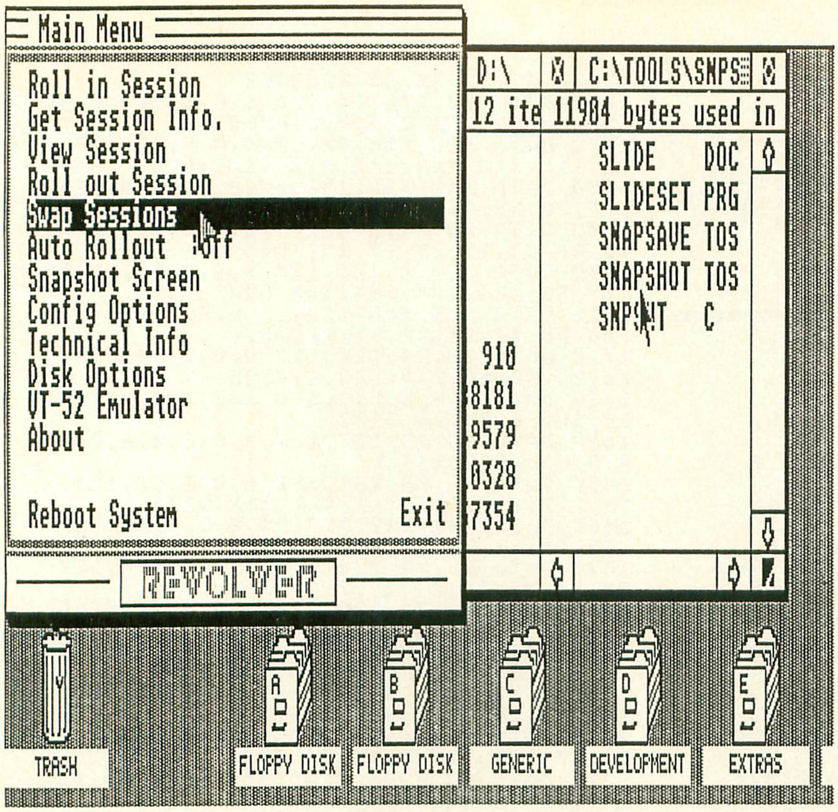
So far, only one virus has been identified in the United States. Another has been described by users, but no one has found a way of detecting it. The virus seemed to have been created in Europe, and gradually found its way to the U.S. through the worldwide series of computer networks and telecommunications.

Virus killing has become a popular cause. George Woodside, the author of *Turtle*, a popular hard-disk backup utility, has been working on a virus immunization program for the ST. George recently posted an anti-virus program that detects and erases a virus program from your ST disks. The program has been made available through DELPHI, CompuServe and GEnie, is also available through Usenet, the worldwide Unix users network.

An interesting twist on the virus issue occurred last month with the release of an imunization program called *FluShot*. Apparently the same people who developed a virus received a copy of FluShot, added a new virus to it, and re-released FluShot back into the public. FluShot 3 has the virus attached! Later versions of Flu-Shot do not have the virus.

Atari-FAX

Atari is pushing hard to make its Mega ST/SLM804 Laser printer combination a viable desktop publishing system for small businesses. Atari has met with limited sucess in trying to sell the ST and Mega as a general business computer, and has now aimed for the smaller "niche"markets (eg., MIDI music, desktop publishing, eduction, etc.). At the same time, the real world of business computing is finding the modern office equipped with computers able to network, communicate with



mainframes, and send and receive facsimile transmissions, commonly called FAXs.

Most business can buy a dedicated FAX machine for under $1,500. FAX is different than its predecessor TELEX, because the recipient of a FAX sees a photocopied image of the original instead of a teletype letter of text. FAX gives you instant copies of business correspondance, invoices and other written reports developed on your computer.

Microfantasy Corp., the company responsible for *Atariwriter 80,* is now working with a small development group on adding FAX capabilities to the ST computer. A new device will plug into the cartridge port of the Mega to interface your computer with a FAX interface. When you want to send a FAX to another party, you specify a text or graphic file in the custom FAX program. The document is converted into a standard FAX transmission and sent to the specified FAX machine. The FAX device has built-in Hayes compatible 9600 baud modem. By connecting the FAX device to your phone line, the FAX program will automatically dial and transmit the document to the destination FAX machine. On the receiving end, the scanner device can be set up to automatically retrieve a FAX document in FAX Group 1,2 or 3 format.

When a FAX transmission is received, the document image can be stored as a GEM image file, DEGAS picture or standard data file. A special printing utility will print the document on any dot-matrix or laser printer (including the Atari SLM804.)

And now for the kicker, the suggested list price will be under $500. The FAX unit will be sold on a "direct" basis and should be available at the end of this year.

Multi-Finder ST

The other 68000-based microcomputers (Mac & Amiga) have built their new operating systems around a multi-tasking environment. You can run a word processor at the same time that a telecommunications program is downloading a file. Since the operating system supports multi-tasking, programs for the Mac and Amiga can be run at the same time. The ST's GEM system doesn't support multi-tasking. However, several programs have been popping up that let you run more than one program in your ST's memory at once.

Juggler (Michtron, $49.95 list) is a program that lets you load up to seven GEM-based programs into your ST. Each application is loaded into a certain portion of memory. A switchbox allows you to select which application will be the "live" program. The ability to switch instantly between applications can greatly improve your productivity. Michtron just released its new *Juggler II,* an improved version which lets you partition your ST's memory into two, four or eight sections. The new software handles TOS and non-GEM applications too! So you can switch between a video game and word processor, quicker than your boss can find you playing games when you should be working.

Revolver (Intersect, $49.95 list) is another program that, among other things, switches between various applications. Revolver can save a copy of the entire one megabyte of memory in your 1040 ST to a compressed disk file. At a later time, you can reload the disk-file data into your ST's memory and begin running the program where you left off.

Revolver is a TOS application that is put into the AUTO folder of your boot disk. When you turn on your ST computer, Revolver loads itself into your ST's memory and waits for the user to activate its main menu. Revolver can be accessed in any screen resolution and within any program, GEM or non-GEM. Therefore, you could save a video game during a difficult game playing section, and later reload the section to replay the fun parts. Revolver is also an incredible utility for your ST. In addition to its application-switching functions, it has an impressive list of disk utilities, printer spoolers, reset-button-proof RAM disks, automatic back-up utilities and other functions for which you might otherwise depend on a desk accessory.

Switch Back (Alpha Systems, $69.95 list) is a simple utility which allows a running application to be saved as a compressed binary disk file. You can reload saved applications at any time, giving you the same utility as Revolver's switching function. Switch Back is being promoted as an "unprotect" program which can remove the copy-protection method built into a game or GEM application, so its relative merits dim compared with Revolver and Juggler.

Companies mentioned:

Microfantasy, Inc.
5811 Cardoza Drive
Westlake Village, CA 91362

Michtron
576 Telegraph
Pontiac, MI 48053
(313) 334-5700

Intersect Software
3951 Sawyer Road, #108
Sarasota, FL 33583
(800) 826-0130

Alpha Systems
1012 Skyland
Macedonia, OH 44056
(216) 374-7469

Snowplow

continued from page 17

LISTING 1: M/L EDITOR DATA

```
1000 DATA 255,255,0,128,10,128,72,169,
2,141,10,212,141,26,208,104,6014
1010 DATA 64,0,140,255,143,0,0,0,0,0
,0,0,236,226,202,3155
1020 DATA 42,162,162,170,170,63,143,16
3,168,10,10,10,10,255,255,0,5452
1030 DATA 170,128,128,128,128,255,255,
0,170,10,10,10,10,253,245,213,8922
1040 DATA 213,213,245,254,254,127,95,8
7,87,87,95,191,191,255,255,255,5771
1050 DATA 255,255,255,255,255,250,234,
160,162,160,162,226,250,250,234,170,98
25
1060 DATA 170,170,170,234,250,191,175,
11,171,11,171,175,191,191,175,171,2873
1070 DATA 171,171,171,175,191,0,0,0,0,
0,24,24,48,255,255,255,6402
1080 DATA 251,234,251,234,255,254,234,
239,234,254,254,254,254,191,171,251,28
82
1090 DATA 171,191,191,191,191,127,99,9
9,99,99,99,127,0,56,24,24,2204
1100 DATA 24,60,60,60,0,127,99,3,127,9
6,96,127,0,126,6,6,9776
1110 DATA 127,7,7,127,0,112,112,112,11
9,119,127,7,0,127,96,96,2628
1120 DATA 127,7,7,127,0,124,108,96,127
,99,99,127,0,127,3,3,675
1130 DATA 31,24,24,24,0,62,54,54,127,1
19,119,127,0,127,99,99,2572
1140 DATA 127,7,7,7,0,0,0,24,24,0,24,2
4,0,255,255,254,3749
1150 DATA 248,170,170,207,255,255,255,
175,171,170,170,243,255,0,0,1,422
1160 DATA 4,85,85,48,0,0,0,80,84,85,85
,12,0,0,60,102,7638
1170 DATA 12,24,0,24,0,0,60,102,110,11
0,96,62,0,0,0,63,7460
1180 DATA 3,127,103,127,0,0,96,96,127,
115,115,127,0,0,0,127,808
1190 DATA 96,96,96,127,0,0,3,3,127,99,
99,127,0,0,0,127,9097
1200 DATA 99,127,112,127,0,0,30,24,126
,24,56,56,0,0,0,127,7493
1210 DATA 99,99,127,7,127,0,96,96,127,
115,115,115,0,0,12,0,9109
1220 DATA 12,12,28,28,0,0,12,0,12,12,1
4,14,126,0,48,48,5212
1230 DATA 118,124,118,115,0,0,24,24,24
,56,56,56,0,0,0,102,6466
1240 DATA 127,127,107,99,0,0,0,63,51,1
15,115,115,0,0,0,63,8104
1250 DATA 51,115,115,127,0,0,0,63,51,1
15,127,112,112,0,0,127,726
1260 DATA 99,99,127,7,7,0,0,63,51,112,
112,112,0,0,0,127,8692
1270 DATA 96,127,7,127,0,0,12,127,12,2
8,28,28,0,0,0,51,5097
1280 DATA 51,115,115,127,0,0,0,99,99,9
9,54,28,0,0,0,99,7601
1290 DATA 107,127,62,54,0,0,0,102,60,2
4,60,102,0,0,0,51,6349
1300 DATA 51,115,127,3,15,0,0,126,12,2
4,48,126,0,0,30,24,6279
1310 DATA 24,24,24,30,0,0,64,96,48,24,
12,6,0,0,120,24,5850
1320 DATA 24,24,24,120,0,0,8,28,54,99,
0,0,0,0,0,0,3700
1330 DATA 0,0,0,255,0,0,0,7,15,28,28,2
8,31,15,0,0,4078
1340 DATA 0,24,31,31,0,0,0,252,252,12,
0,0,252,254,14,14,3275
1350 DATA 14,30,252,248,0,0,0,124,124,
30,30,31,31,31,29,28,8002
1360 DATA 28,28,124,124,0,0,0,62,62,56
,56,56,56,184,248,248,6206
1370 DATA 120,120,62,62,0,0,0,63,127,1
12,112,112,112,112,112,112,4003
1380 DATA 112,112,127,63,0,0,0,248,252
,28,28,28,28,28,28,28,9149
1390 DATA 28,28,252,248,0,0,0,124,124,
28,28,28,29,29,29,29,7936
1400 DATA 31,31,126,124,0,0,0,62,62,56
,56,56,184,184,184,184,5941
1410 DATA 248,248,126,62,0,0,0,127,127
,28,28,28,28,31,31,28,7574
1420 DATA 28,28,127,127,0,0,0,240,248,
28,28,28,28,248,240,0,4905
1430 DATA 0,0,0,0,0,0,0,127,127,28,28,
28,28,28,28,28,6137
1440 DATA 28,28,127,127,0,0,0,0,0,0,0,
0,0,0,0,0,2413
1450 DATA 28,28,252,252,0,0,0,63,127,1
12,112,112,112,112,112,112,5137
1460 DATA 112,112,127,63,0,0,0,248,252
,28,28,28,28,28,28,28,9229
1470 DATA 28,28,252,248,0,0,0,124,124,
28,28,28,29,29,29,29,8016
1480 DATA 31,31,126,124,0,0,0,62,62,56
,56,56,184,184,184,184,6021
1490 DATA 248,248,126,62,0,255,255,255
,215,255,255,255,255,0,0,0,1880
1500 DATA 20,0,0,0,0,255,253,255,253,2
55,253,255,253,0,1,0,835
1510 DATA 1,0,1,0,1,255,127,255,127,25
5,127,255,127,0,64,0,6739
1520 DATA 64,0,64,0,64,255,253,255,93,
255,253,255,253,0,1,0,9971
1530 DATA 81,0,1,0,1,255,127,255,117,2
55,127,255,127,0,64,0,6749
1540 DATA 69,0,64,0,64,255,253,255,93,
255,255,255,255,0,1,0,44
1550 DATA 81,0,0,0,0,255,255,255,93,25
5,253,255,253,0,0,0,9505
1560 DATA 81,0,1,0,1,255,255,255,117,2
55,127,255,127,0,0,0,6715
1570 DATA 69,0,64,0,64,255,127,255,117
,255,255,255,255,0,64,0,353
1580 DATA 69,0,0,0,0,255,255,255,253,2
55,253,255,253,0,0,0,963
1590 DATA 1,0,1,0,1,255,255,255,127,25
5,127,255,127,0,0,0,6755
1600 DATA 64,0,64,0,64,255,127,255,127
,255,255,255,255,0,64,0,468
1610 DATA 64,0,0,0,0,255,253,255,253,2
55,255,255,255,0,1,0,1037
1620 DATA 1,0,0,0,0,255,239,255,85,255
,255,255,255,0,8,0,9479
1630 DATA 85,0,0,0,0,24,24,24,24,24,24
,24,24,0,126,120,7349
1640 DATA 124,110,102,6,0,8,24,56,120,
56,24,8,0,16,24,28,6010
1650 DATA 30,28,24,16,0,0,139,39,139,0
,38,53,37,44,0,17,6831
1660 DATA 21,16,0,48,44,47,55,51,0,19,
0,0,0,0,0,0,3390
1670 DATA 51,35,47,50,37,0,16,16,16,16
,16,16,0,0,0,0,3229
1680 DATA 0,0,155,39,155,0,0,179,174,1
75,183,166,172,161,171,165,1524
1690 DATA 179,0,183,169,174,0,0,0,0,0,
0,0,112,114,101,115,371
1700 DATA 115,0,51,52,33,50,52,0,0,0,0
,0,0,0,127,32,5422
1710 DATA 127,112,112,112,112,112,112,
112,66,0,123,2,112,112,112,2,2464
1720 DATA 2,112,112,112,112,87,0,124,1
12,112,112,112,70,160,123,65,5543
1730 DATA 0,127,128,127,174,127,112,11
2,112,66,0,123,2,117,0,144,3300
1740 DATA 117,128,144,117,0,145,117,12
8,145,117,0,146,117,128,146,117,7328
1750 DATA 0,147,117,128,147,117,0,148,
117,128,148,213,0,149,70,0,5181
1760 DATA 139,6,65,128,127,0,64,165,79
,160,50,185,128,127,153,192,8911
1770 DATA 127,136,16,247,169,140,141,2
44,2,32,251,64,162,74,160,158,9766
1780 DATA 169,7,32,92,228,169,0,133,19
8,169,0,133,183,133,178,32,8136
1790 DATA 39,76,76,47,64,32,73,76,32,9
0,66,32,215,74,169,3,2740
```

```
1800 DATA 133,182,169,0,133,186,32,225
,73,32,81,78,32,36,75,32,1970
1810 DATA 48,71,32,130,70,32,215,74,32
,50,65,165,183,208,14,165,6879
1820 DATA 178,208,10,169,1,133,178,32,
166,79,76,106,64,32,84,76,3573
1830 DATA 32,156,70,32,61,71,32,4,72,3
2,239,70,32,36,71,32,433
1840 DATA 242,77,169,11,133,184,32,75,
65,169,1,133,146,173,31,208,7375
1850 DATA 201,6,208,3,76,232,77,173,12
0,2,133,184,32,75,65,141,5862
1860 DATA 30,208,32,0,68,32,145,69,173
,132,2,208,3,32,166,76,4089
1870 DATA 32,213,77,173,12,208,201,12,
176,33,173,13,208,201,12,176,8549
1880 DATA 26,165,175,197,173,208,12,16
5,174,197,172,208,6,32,143,70,8781
1890 DATA 76,64,64,165,179,5,180,5,181
,208,24,141,30,208,32,241,8474
1900 DATA 73,32,103,75,32,213,73,141,3
0,208,32,81,66,76,234,64,5853
1910 DATA 76,14,74,165,182,240,249,173
,252,2,201,255,240,3,32,131,1670
1920 DATA 74,76,132,64,169,62,141,47,2
,169,17,141,111,2,169,3,3019
1930 DATA 141,29,208,169,128,141,7,212
,169,40,141,192,2,169,6,141,7174
1940 DATA 193,2,169,166,141,194,2,141,
195,2,169,128,133,131,169,112,9379
1950 DATA 133,133,169,1,141,10,208,141
,11,208,96,160,255,169,0,153,9493
1960 DATA 0,132,153,0,133,153,0,134,15
3,0,135,153,0,131,136,192,6982
1970 DATA 255,208,236,96,162,3,165,184
,221,177,65,240,4,202,16,248,1630
1980 DATA 96,189,189,65,133,205,189,19
3,65,133,206,189,197,65,133,203,3206
1990 DATA 189,201,65,133,204,138,72,16
5,133,133,187,24,125,185,65,133,9170
2000 DATA 134,133,188,168,169,0,153,25
5,132,160,0,177,205,166,187,157,2810
2010 DATA 0,132,177,203,166,188,157,0,
133,200,230,187,230,188,192,16,3403
2020 DATA 208,233,166,188,157,0,133,16
6,131,142,0,208,104,170,165,131,386
2030 DATA 24,125,181,65,133,132,141,1,
208,96,11,7,14,13,255,1,2801
2040 DATA 0,0,0,0,255,1,205,221,237,25
3,65,65,65,65,13,29,5096
2050 DATA 45,61,66,66,66,66,0,0,0,255,
255,118,118,118,118,118,7020
2060 DATA 118,255,255,0,0,0,0,0,0,255,
255,110,110,110,110,110,6508
2070 DATA 110,255,255,0,0,0,0,102,102,
126,126,126,126,126,126,102,6271
2080 DATA 102,102,126,126,102,0,0,102,
126,126,102,102,102,126,126,126,6330
2090 DATA 126,126,126,102,102,0,0,170,
170,128,128,132,132,132,132,132,8582
2100 DATA 132,128,128,170,170,0,0,253,
253,1,1,97,97,97,97,97,5514
2110 DATA 97,1,1,253,253,0,0,255,255,0
,153,153,24,24,153,153,7734
2120 DATA 24,24,153,153,0,0,0,0,129,15
3,24,24,153,153,24,24,1381
2130 DATA 153,153,0,255,255,0,160,19,2
08,2,160,7,32,247,67,136,6947
2140 DATA 208,250,96,160,0,169,0,141,8
9,86,169,144,141,125,86,185,8999
2150 DATA 89,86,24,105,128,153,90,86,1
85,125,86,105,0,153,126,86,6308
2160 DATA 200,192,33,208,234,162,7,169
,16,149,212,202,16,251,133,196,2461
2170 DATA 32,48,71,169,3,133,182,169,1
9,141,16,139,169,96,133,211,8963
2180 DATA 169,255,133,191,96,216,68,10
,70,0,169,127,133,206,169,136,9691
2190 DATA 133,205,96,164,130,185,89,86
,133,205,185,125,86,133,206,96,1136
2200 DATA 22,4,165,184,201,7,208,62,32
,86,69,144,1,96,165,243,7815
2210 DATA 201,83,144,1,96,169,1,133,13
8,32,197,74,169,1,133,135,6561
2220 DATA 32,61,67,238,183,66,230,243,
162,3,142,4,212,32,247,67,9471
2230 DATA 32,165,75,202,16,244,198,138
,165,138,16,224,32,96,70,32,6710
2240 DATA 69,74,32,117,71,96,201,11,20
8,79,32,105,69,144,1,96,4185
2250 DATA 165,243,208,1,96,169,1,133,1
38,32,197,74,162,0,142,4,5011
2260 DATA 212,32,247,67,32,200,75,232,
224,4,208,242,169,255,133,135,4456
2270 DATA 162,0,142,4,212,32,61,67,198
,243,206,183,66,198,138,165,2103
2280 DATA 138,16,217,76,243,66,160,0,1
85,136,127,24,101,135,153,136,8520
2290 DATA 127,200,200,200,192,33,208,2
40,96,201,14,208,48,32,132,69,8431
2300 DATA 144,1,96,165,244,240,94,32,1
97,74,169,255,133,135,32,189,1523
2310 DATA 67,198,244,206,184,66,162,15
,142,5,212,32,247,67,32,217,9044
2320 DATA 75,202,16,244,169,0,141,5,21
2,76,243,66,96,201,13,208,9413
2330 DATA 52,32,119,69,144,1,96,165,24
4,201,9,144,1,96,32,197,6819
2340 DATA 74,162,0,142,5,212,32,247,67
,32,217,75,232,224,16,208,733
2350 DATA 242,162,0,142,5,212,169,1,13
3,135,32,189,67,230,244,238,2698
2360 DATA 184,66,76,243,66,96,165,135,
48,27,160,0,185,136,127,24,6077
2370 DATA 105,128,153,136,127,200,185,
136,127,105,0,153,136,127,200,200,1727
2380 DATA 192,33,208,232,96,160,0,185,
136,127,56,233,128,153,136,127,894
2390 DATA 200,185,136,127,233,0,153,13
6,127,200,200,192,33,208,232,96,3204
2400 DATA 169,0,133,20,165,20,240,252,
96,165,184,201,7,208,68,165,1302
2410 DATA 131,201,124,144,9,165,243,20
1,83,176,3,76 70,68,165,131,8120
2420 DATA 201,200,176,46,32,86,69,144,
1,96,32,197,74,160,7,230,6716
2430 DATA 131,166,131,142,0,208,230,13
2,166,132,142,1,208,32,247,67,85
2440 DATA 32,165,75,136,16,233,238,183
,66,238,183,66,76,243,66,32,9850
2450 DATA 185,66,96,201,11,208,61,32,1
05,69,144,1,96,165,131,201,7815
2460 DATA 132,176,12,165,243,208,232,1
65,131,201,48,240,229,144,227,32,4554
2470 DATA 197,74,160,7,198,131,166,131
,142,0,208,198,132,166,132,142,1543
2480 DATA 1,208,32,247,67,32,200,75,13
6,16,233,206,183,66,206,183,2248
2490 DATA 66,76,243,66,201,13,208,106,
32,119,69,144,1,96,165,133,7013
2500 DATA 201,96,144,9,165,244,201,9,1
76,3,76,70,68,165,133,201,8824
2510 DATA 192,176,159,169,15,133,138,3
2,197,74,165,133,133,187,165,134,1192
2520 DATA 133,188,168,169,0,153,0,133,
162,15,164,187,185,15,132,153,8890
2530 DATA 16,132,164,188,185,15,133,20
1,24,208,4,169,153,208,6,201,183
2540 DATA 153,208,2,169,24,153,16,133,
198,187,198,188,202,16,219,230,3906
2550 DATA 133,230,134,32,247,67,198,13
8,165,138,16,190,238,184,66,76,997
2560 DATA 243,66,201,14,240,3,76,73,68
,32,132,69,144,1,96,165,5106
2570 DATA 133,201,112,176,7,165,244,24
0,3,76,70,68,165,133,201,48,8961
2580 DATA 208,3,76,70,68,32,197,74,169
,15,133,138,164,133,162,15,7259
2590 DATA 185,0,132,153,255,131,185,0,
133,201,24,208,4,169,153,208,1147
2600 DATA 6,201,153,208,2,169,24,153,2
55,132,200,202,16,226,32,247,2758
2610 DATA 67,198,133,198,138,165,138,1
6,211,206,184,66,76,243,66,172,1945
2620 DATA 184,66,32,172,66,172,183,66,
200,200,177,205,133,137,32,82,537
2630 DATA 70,96,172,184,66,32,172,66,1
72,183,66,136,136,76,97,69,7525
2640 DATA 172,184,66,200,32,172,66,172
,183,66,76,97,69,172,184,66,8636
2650 DATA 136,32,172,66,172,183,66,76,
97,69,172,184,66,32,172,66,7263
2660 DATA 172,183,66,177,205,162,15,22
1,53,70,240,6,202,16,248,76,9649
2670 DATA 224,69,201,27,240,4,201,29,2
08,21,72,169,97,145,205,200,1074
2680 DATA 145,205,32,71,72,32,93,74,10
4,201,27,240,19,208,26,170,7802
```

```
2690 DATA 232,138,145,205,200,177,205,
170,232,138,145,205,201,11,240,9,3344
2700 DATA 230,174,208,2,230,175,32,75,
72,165,137,201,8,208,44,230,507
2710 DATA 137,172,184,66,32,172,66,166
,207,165,206,213,224,240,4,202,4888
2720 DATA 16,247,96,173,183,66,24,101,
205,213,218,208,242,169,100,149,4762
2730 DATA 153,32,254,75,32,48,71,32,13
0,70,96,172,184,66,32,172,6748
2740 DATA 66,166,185,240,245,165,206,2
13,234,240,4,202,16,247,96,165,4734
2750 DATA 205,24,109,183,66,213,230,20
8,242,169,50,149,160,96,96,98,1582
2760 DATA 100,102,104,106,108,110,112,
114,116,118,120,8,27,29,97,99,4132
2770 DATA 101,103,105,107,109,111,113,
115,117,119,121,162,28,221,53,70,7633
2780 DATA 240,5,202,16,248,56,96,24,96
,198,181,165,181,5,180,5,8158
2790 DATA 179,240,20,165,181,16,16,169
,9,133,181,198,180,165,180,16,18
2800 DATA 6,169,9,133,180,198,179,32,1
30,70,96,162,2,181,179,9,7559
2810 DATA 16,157,6,139,202,16,246,96,1
69,0,162,15,133,146,32,77,6278
2820 DATA 66,202,16,250,96,162,16,169,
12,157,66,3,32,86,228,162,7326
2830 DATA 16,169,3,157,66,3,169,70,157
,69,3,169,227,157,68,3,6293
2840 DATA 169,4,157,74,3,169,0,157,75,
3,32,86,228,16,1,96,2897
2850 DATA 162,16,169,144,157,69,3,169,
0,157,68,3,169,0,169,10,3945
2860 DATA 157,73,3,169,7,157,66,3,32,8
6,228,96,68,49,58,83,3887
2870 DATA 77,65,80,46,32,32,32,155,160
,0,132,172,230,172,132,173,419
2880 DATA 132,176,32,172,66,160,0,177,
205,162,12,221,53,70,240,20,8692
2890 DATA 202,16,248,200,200,192,0,208
,238,230,206,230,176,164,176,192,8248
2900 DATA 10,208,226,96,230,172,208,2,
230,173,76,10,71,160,50,185,9671
2910 DATA 192,127,153,128,127,136,16,2
47,96,169,0,133,181,169,1,133,8878
2920 DATA 179,165,192,133,180,96,160,0
,132,176,132,207,164,176,192,19,1797
2930 DATA 208,1,96,32,172,66,160,0,177
,205,201,8,240,11,200,200,1356
2940 DATA 192,126,208,244,230,176,76,6
7,71,166,207,165,206,149,224,152,5370
2950 DATA 24,101,205,149,218,230,207,1
65,207,201,6,208,225,96,166,207,6132
2960 DATA 240,32,181,153,240,25,214,15
3,181,153,208,19,181,224,133,206,4946
2970 DATA 181,218,133,205,160,0,169,8,
145,205,200,169,10,145,205,202,2903
2980 DATA 16,224,166,185,208,1,96,181,
160,240,38,214,160,181,160,208,5016
2990 DATA 32,181,234,133,206,181,230,1
33,205,160,0,177,205,201,96,208,5224
3000 DATA 67,169,27,145,205,200,169,28
,145,205,169,80,149,166,32,180,1493
3010 DATA 77,202,16,211,166,185,181,16
6,240,33,214,166,181,166,208,27,3983
3020 DATA 181,234,133,206,181,230,133,
205,160,0,177,205,201,27,208,15,1946
3030 DATA 169,96,145,205,200,145,205,1
69,100,149,160,202,16,216,96,169,3253
3040 DATA 97,208,239,96,169,29,145,205
,200,169,30,208,187,160,0,132,1427
3050 DATA 176,132,185,164,176,192,19,2
08,1,96,32,172,66,160,0,177,7845
3060 DATA 205,201,27,240,11,200,200,19
2,126,208,244,230,176,76,10,72,2211
3070 DATA 166,185,165,206,149,234,152,
24,101,205,149,230,152,72,169,96,2743
3080 DATA 145,205,200,145,205,104,168,
230,185,165,185,201,4,208,214,96,4952
3090 DATA 162,3,208,2,162,4,181,212,24
,105,1,149,212,201,26,144,9016
3100 DATA 14,169,16,149,212,202,48,7,2
46,212,181,212,76,84,72,162,1465
3110 DATA 5,181,212,157,29,139,202,16,
248,165,213,197,196,240,7,133,3992
3120 DATA 196,230,182,32,225,73,96,165
,211,240,26,198,211,166,189,189,5892
3130 DATA 205,72,141,151,72,189,206,72
,141,152,72,169,1,133,186,32,8804
3140 DATA 255,255,76,98,228,173,10,210
,41,3,170,10,133,189,169,107,9464
3150 DATA 133,211,189,63,73,133,143,18
9,67,73,133,144,24,105,12,133,6814
3160 DATA 141,32,115,73,32,79,73,165,1
93,133,238,165,194,133,239,169,4805
3170 DATA 0,133,186,76,98,228,213,72,2
20,72,227,72,234,72,32,102,446
3180 DATA 73,32,241,72,96,32,71,73,32,
22,73,96,32,102,73,32,1995
3190 DATA 22,73,96,32,71,73,32,241,72,
96,164,143,162,31,185,0,7162
3200 DATA 134,153,255,133,185,0,135,15
3,255,134,200,202,16,240,198,143,5116
3210 DATA 169,0,164,142,153,1,131,169,
195,153,255,130,198,142,96,165,3771
3220 DATA 143,24,105,31,168,162,31,185
,0,134,153,1,134,185,0,135,6886
3230 DATA 153,1,135,136,202,16,240,230
,143,169,0,164,142,153,0,131,9989
3240 DATA 169,195,153,2,131,230,142,96
,192,48,48,192,10,224,10,224,103
3250 DATA 198,144,198,144,198,141,198,
141,165,144,141,2,208,24,105,16,8627
3260 DATA 141,3,208,165,141,141,4,208,
24,105,6,141,7,208,96,230,9081
3270 DATA 144,230,144,230,141,230,141,
165,141,76,79,73,32,197,73,162,253
3280 DATA 0,164,143,189,165,73,153,0,1
34,153,1,134,189,181,73,153,16
3290 DATA 0,135,153,1,135,200,200,232,
224,16,208,231,165,143,24,105,2577
3300 DATA 12,133,142,164,142,169,195,1
53,0,131,153,1,131,96,0,4,5089
3310 DATA 2,50,11,7,61,79,12,61,71,11,
18,18,1,0,0,128,8048
3320 DATA 72,72,208,230,188,240,50,188
,224,208,76,64,32,0,162,0,7860
3330 DATA 138,157,0,135,157,0,134,157,
0,131,232,208,244,96,198,182,4057
3340 DATA 32,225,73,32,48,71,32,130,70
,96,165,182,201,10,144,4,6665
3350 DATA 169,9,133,182,9,16,141,16,13
9,96,169,0,141,2,208,141,7227
3360 DATA 3,208,141,4,208,141,7,208,13
3,211,165,193,133,238,165,194,5895
3370 DATA 133,239,169,0,133,186,96,169
,155,141,170,127,169,0,141,169,1508
3380 DATA 127,169,0,133,146,32,41,74,1
73,31,208,201,6,208,249,76,686
3390 DATA 32,64,169,0,141,8,210,162,3,
142,15,210,162,7,157,0,6267
3400 DATA 210,202,16,250,96,169,0,141,
0,210,141,1,210,96,169,0,7956
3410 DATA 141,3,210,141,2,210,96,169,0
,141,4,210,141,5,210,141,9328
3420 DATA 6,210,141,7,210,96,32,41,74,
162,1,169,121,141,4,210,7767
3430 DATA 169,166,141,5,210,169,10,141
,6,210,169,36,141,7,210,32,7674
3440 DATA 77,66,32,78,74,32,77,66,202,
16,224,96,162,255,142,252,3118
3450 DATA 2,169,0,133,146,32,247,67,17
3,252,2,201,255,240,249,142,6702
3460 DATA 252,2,169,1,133,146,96,165,1
46,208,3,76,98,228,165,238,2848
3470 DATA 5,239,240,13,198,238,165,238
,201,255,208,239,198,239,76,162,9369
3480 DATA 74,198,145,165,145,208,228,1
65,195,133,145,76,126,72,165,186,3623
3490 DATA 208,13,169,50,141,2,210,169,
70,141,3,210,32,247,67,96,8978
3500 DATA 169,128,133,131,169,112,133,
133,169,4,141,184,66,169,24,141,9520
3510 DATA 183,66,169,0,133,243,133,244
,162,5,169,0,149,153,202,16,70
3520 DATA 251,162,3,189,32,75,149,160,
169,0,149,166,202,16,244,169,2159
3530 DATA 0,133,174,133,175,230,174,13
3,211,165,195,133,145,32,41,74,809
3540 DATA 165,193,133,238,165,194,133,
239,96,48,96,128,176,166,191,224,5271
3550 DATA 5,240,2,230,191,166,191,189,
75,75,133,192,189,81,75,133,1797
3560 DATA 193,189,87,75,133,194,189,93
,75,133,195,169,1,141,88,86,9449
3570 DATA 32,41,74,96,8,7,6,5,4,3,128,
128,0,128,0,128,1304
```

```
3580 DATA 3,2,2,1,1,0,10,8,6,4,2,1,0,2
,1,3,3955
3590 DATA 165,184,72,160,4,132,148,169
,0,133,197,162,3,134,147,189,764
3600 DATA 99,75,170,32,88,65,169,134,1
41,3,210,165,197,24,105,9,7777
3610 DATA 133,197,141,2,210,32,81,66,1
98,147,166,147,208,225,198,148,4939
3620 DATA 164,148,208,215,104,133,184,
32,88,65,32,69,74,96,132,208,8662
3630 DATA 134,210,165,209,41,1,170,189
,198,75,164,134,153,1,133,153,818
3640 DATA 2,133,153,13,133,153,14,133,
230,209,164,208,166,210,96,85,3522
3650 DATA 169,132,208,134,210,165,209,
41,1,170,189,215,75,76,177,75,1336
3660 DATA 170,149,132,208,134,210,164,
134,162,7,185,4,133,201,153,208,3283
3670 DATA 4,169,24,208,6,201,24,208,2,
169,153,153,4,133,200,202,1357
3680 DATA 16,232,166,210,164,208,96,32
,69,74,169,164,141,3,210,162,1299
3690 DATA 160,142,2,210,32,30,76,224,4
5,208,5,169,162,141,3,210,9697
3700 DATA 202,208,238,32,69,74,96,160,
200,32,247,67,136,16,250,96,820
3710 DATA 32,73,76,162,96,169,3,157,66
,3,169,76,157,69,3,169,6686
3720 DATA 156,157,68,3,169,6,157,74,3,
169,0,157,75,3,32,86,3452
3730 DATA 228,96,162,96,169,12,157,66,
3,32,86,228,96,169,5,162,7874
3740 DATA 96,157,66,3,169,86,157,69,3,
169,68,157,68,3,169,20,5502
3750 DATA 157,72,3,169,0,157,73,3,32,8
6,228,48,7,173,72,86,5414
3760 DATA 201,70,208,12,32,39,76,169,0
,133,178,104,104,76,86,64,6317
3770 DATA 162,10,189,68,86,157,225,70,
201,32,240,3,232,208,243,169,5380
3780 DATA 155,157,225,70,96,68,49,58,8
3,77,65,80,46,42,155,173,6370
3790 DATA 88,86,208,1,96,165,184,201,7
,208,61,173,183,66,133,140,1472
3800 DATA 169,200,56,229,131,74,74,133
,139,24,109,183,66,141,183,66,9653
3810 DATA 206,183,66,206,183,66,32,86,
69,144,6,165,140,141,183,66,9329
3820 DATA 96,162,200,142,0,208,134,131
,232,142,1,208,134,132,238,183,4745
3830 DATA 66,238,183,66,206,88,86,96,2
01,11,208,63,173,183,66,133,1005
3840 DATA 140,165,131,56,233,48,74,74,
133,139,173,183,66,56,229,139,1477
3850 DATA 141,183,66,238,183,66,238,18
3,66,32,105,69,144,6,165,140,9516
3860 DATA 141,183,66,96,162,48,142,0,2
08,134,131,202,142,1,208,134,1242
3870 DATA 132,206,183,66,206,183,66,20
6,88,86,96,201,14,208,70,173,1497
3880 DATA 184,66,133,140,165,133,56,23
3,48,74,74,74,74,133,139,173,9585
3890 DATA 184,66,56,229,139,141,184,66
,238,184,66,32,132,69,144,6,8677
3900 DATA 165,140,141,184,66,96,206,18
4,66,32,110,77,169,48,133,133,9364
3910 DATA 32,75,65,206,88,86,96,160,25
5,169,0,153,0,132,153,0,7983
3920 DATA 133,136,208,247,96,201,13,20
8,34,173,184,66,133,140,169,192,3526
3930 DATA 56,229,133,74,74,74,74,24,10
9,184,66,141,184,66,206,184,1252
3940 DATA 66,32,119,69,144,6,165,140,1
41,184,66,96,32,110,77,169,8536
3950 DATA 192,133,133,32,75,65,238,184
,66,206,88,86,96,32,69,74,7407
3960 DATA 169,12,141,2,210,169,169,133
,177,198,177,165,177,201,160,144,6214
3970 DATA 9,141,3,210,32,81,66,76,192,
77,32,69,74,96,165,186,8261
3980 DATA 240,11,169,134,141,1,210,169
,11,141,0,210,96,32,60,74,6627
3990 DATA 96,173,31,208,201,6,240,249,
76,14,74,169,127,141,49,2,8128
4000 DATA 169,128,141,48,2,169,192,141
,14,212,169,128,141,1,2,169,8758
4010 DATA 0,141,0,2,162,4,189,156,66,1
57,196,2,202,16,247,96,140
4020 DATA 146,10,0,66,0,238,225,244,23
3,239,238,225,236,128,247,229,1439
4030 DATA 225,244,232,229,242,128,243,
229,242,246,233,227,229,128,226,245,38
70
4040 DATA 236,236,229,244,233,238,0,0,
0,0,0,0,0,128,243,238,8249
4050 DATA 239,247,128,247,225,242,238,
233,238,231,160,0,132,146,132,186,7190
4060 DATA 32,41,74,166,198,208,26,169,
8,141,31,208,169,90,133,240,2443
4070 DATA 185,0,142,153,0,125,169,0,15
3,0,142,153,0,124,200,208,65
4080 DATA 239,162,5,181,212,157,172,12
3,202,16,248,162,7,169,0,157,1191
4090 DATA 0,208,202,16,250,168,153,0,1
24,136,208,250,160,53,185,28,2314
4100 DATA 78,153,22,124,136,16,247,160
,4,185,23,78,153,196,2,136,8845
4110 DATA 16,247,169,0,133,205,133,203
,141,198,2,169,125,133,204,169,4127
4120 DATA 142,133,206,169,0,141,48,2,1
69,127,141,49,2,165,198,208,584
4130 DATA 43,230,198,32,110,79,169,15,
133,200,169,15,133,150,164,150,1607
4140 DATA 185,52,86,168,185,150,79,133
,203,133,205,164,200,177,203,32,4816
4150 DATA 61,79,176,8,198,150,16,230,1
98,200,16,222,160,0,140,21,9909
4160 DATA 127,185,0,125,153,0,142,200,
208,247,162,7,142,4,212,169,2510
4170 DATA 0,133,20,173,31,208,240,38,2
01,6,240,23,165,20,240,243,3273
4180 DATA 202,16,233,172,21,127,200,19
6,240,208,2,160,0,140,21,127,125
4190 DATA 76,1,79,169,0,133,183,141,21
,127,141,4,212,96,169,1,8097
4200 DATA 133,183,169,0,240,241,132,19
9,201,0,240,32,160,0,145,205,2736
4210 DATA 196,199,240,24,162,0,134,20,
174,252,2,224,255,208,17,166,3462
4220 DATA 20,240,245,72,169,0,145,205,
104,200,208,226,164,199,24,96,3993
4230 DATA 162,255,142,252,2,56,96,169,
1,133,149,173,10,210,41,15,7685
4240 DATA 141,52,86,173,10,210,41,15,1
60,0,217,52,86,240,244,200,2941
4250 DATA 196,149,208,246,153,52,86,20
0,132,149,192,16,208,229,96,0,1963
4260 DATA 16,32,48,64,80,96,112,128,14
4,160,176,192,208,224,240,0,4100
4270 DATA 123,179,123,0,0,0,0,0,0,0,
0,0,0,0,64,6144
4280 DATA 66,68,70,72,74,76,78,80,82,8
4,86,88,90,92,94,0,4440
4290 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,4290
4360 DATA 0,0,0,0,0,0,0,65,67,69,71,73
,75,77,79,81,2304
4310 DATA 83,85,87,89,91,93,95,0,0,0,0
,0,0,0,0,0,6858
4320 DATA 0,0,0,0,0,0,34,57,26,0,34,33
,50,50,57,0,8223
4330 DATA 43,47,44,34,37,0,33,46,36,0,
34,50,57,33,46,0,8710
4340 DATA 51,35,40,33,48,48,37,44,0,0,
0,0,0,0,0,0,5852
4350 DATA 35,47,48,57,50,41,39,40,52,0
,17,25,24,24,0,34,8087
4360 DATA 34,43,0,37,46,52,37,50,48,50
,41,51,37,51,0,0,9019
4370 DATA 0,0,0,0,0,116,111,112,0,115,
99,111,114,101,0,16,3462
4380 DATA 16,16,16,16,16,0,0,166,79,51
,86,169,144,133,206,169,9671
4390 DATA 0,133,205,169,42,133,203,169
,80,133,204,169,1,133,241,169,4244
4400 DATA 154,133,242,160,0,132,152,13
2,202,32,33,80,24,42,38,152,6461
4410 DATA 74,133,201,208,10,32,33,80,1
33,202,32,33,80,133,201,165,9820
4420 DATA 152,240,21,32,33,80,145,205,
32,12,80,198,201,208,244,165,4032
4430 DATA 202,240,210,198,202,76,222,7
9,32,33,80,133,151,165,151,145,2138
4440 DATA 205,32,12,80,198,201,208,245
,165,202,240,185,198,202,76,248,9552
4450 DATA 79,230,205,208,2,230,206,165
,206,197,242,208,8,165,205,197,8211
4460 DATA 241,208,2,104,104,96,177,203
```

```
,230,203,208,2,230,204,96,4,3260
4470 DATA 0,30,96,130,108,110,14,96,13
0,108,116,6,7,130,114,110,6383
4480 DATA 28,96,130,27,28,12,96,130,10
8,110,22,96,130,108,116,4,5594
4490 DATA 0,130,14,15,3,7,135,1,2,3,4,
7,5,6,2,7,6329
4500 DATA 130,5,6,3,7,130,5,6,9,7,130,
98,100,2,7,130,1838
4510 DATA 5,6,2,7,130,5,6,6,7,130,98,1
00,6,7,130,98,2666
4520 DATA 100,2,7,137,13,7,1,2,3,4,7,5
,6,27,7,130,8168
4530 DATA 5,6,2,7,130,98,100,5,7,130,1
4,15,15,7,130,98,2067
4540 DATA 100,4,0,2,7,130,114,110,12,9
6,130,108,116,6,7,130,4720
4550 DATA 5,6,4,7,130,98,100,4,7,130,5
,6,2,7,130,5,221
4560 DATA 6,4,7,132,98,100,5,6,2,7,132
,5,6,98,104,10,1066
4570 DATA 96,130,108,116,6,7,135,1,2,3
,4,7,5,6,5,7,7251
4580 DATA 130,5,6,8,7,132,5,6,98,100,2
,7,130,114,110,14,2828
4590 DATA 96,130,108,116,2,7,130,98,10
0,4,0,135,5,6,98,100,3259
4600 DATA 7,5,6,9,7,133,98,100,7,5,6,5
,7,130,5,6,9311
4610 DATA 2,7,130,98,100,14,7,130,98,1
00,2,7,130,5,6,2,951
4620 DATA 7,130,98,100,10,7,133,98,100
,7,5,6,5,7,4,96,244
4630 DATA 130,108,116,2,7,130,14,15,11
,7,131,13,98,100,2,7,947
4640 DATA 130,98,100,14,7,130,98,100,2
,7,130,98,100,4,0,2,1705
4650 DATA 7,130,98,100,7,7,135,1,2,3,4
,7,120,112,12,96,1661
4660 DATA 130,106,118,14,7,130,98,104,
6,96,130,102,100,8,7,135,5090
4670 DATA 1,2,3,4,7,5,6,4,7,130,5,6,3,
7,130,120,336
4680 DATA 112,12,96,130,108,110,2,96,1
30,106,118,2,7,130,98,100,6139
4690 DATA 2,7,130,114,110,6,96,130,108
,116,2,7,130,98,100,2,4682
4700 DATA 7,130,98,100,4,0,2,7,130,98,
100,4,7,130,5,6,1131
4710 DATA 3,7,130,5,6,5,7,130,5,6,2,7,
130,5,6,6,8443
4720 DATA 7,130,5,6,2,7,130,5,6,8,7,13
0,98,100,6,7,675
4730 DATA 130,98,100,2,7,130,5,6,3,7,1
30,5,6,4,7,133,216
4740 DATA 5,6,7,5,6,16,7,130,8,10,2,7,
130,98,100,2,885
4750 DATA 7,130,5,6,2,7,130,98,100,2,7
,138,98,100,13,7,2436
4760 DATA 1,2,3,4,98,100,2,7,130,98,10
0,2,7,130,98,100,4205
4770 DATA 4,0,2,7,130,120,112,30,96,13
0,27,28,4,96,130,108,5073
4780 DATA 116,4,7,130,98,100,6,7,130,9
8,100,4,7,130,114,110,5312
4790 DATA 2,96,130,108,116,4,7,130,114
,110,2,96,130,108,116,4,5805
4800 DATA 7,130,114,110,4,96,130,108,1
16,6,7,130,98,100,4,7,3806
4810 DATA 132,5,6,98,100,2,7,130,120,1
12,4,96,2,7,130,98,4001
4820 DATA 100,2,7,130,98,100,2,7,130,9
8,100,4,0,13,7,130,2290
4830 DATA 5,6,21,7,130,5,6,2,7,130,98,
100,4,7,132,98,3015
4840 DATA 100,8,10,4,7,130,98,100,4,7,
134,98,100,5,6,98,3087
4850 DATA 100,4,7,134,98,100,5,6,98,10
0,4,7,130,98,100,2,3292
4860 DATA 7,132,14,15,98,100,6,7,130,9
8,100,6,7,130,98,100,4724
4870 DATA 10,7,130,98,100,2,7,130,98,1
00,2,7,130,98,100,4,3891
4880 DATA 0,130,114,110,28,96,130,108,
116,6,7,134,14,15,98,100,4663
4890 DATA 5,6,2,7,130,98,100,6,7,130,1
20,112,4,96,130,106,5996
4900 DATA 118,2,7,130,120,112,4,96,130
,106,118,2,7,130,120,112,6686
4910 DATA 4,96,130,106,118,4,7,130,120
,112,2,96,130,108,110,2,5881
4920 DATA 96,130,106,118,6,7,130,120,1
12,10,96,130,106,118,2,7,4904
4930 DATA 130,98,100,2,7,130,98,100,4,
0,139,98,100,7,5,6,2175
4940 DATA 5,6,5,6,5,6,7,7,130,5,6,3,7,
130,5,6,8566
4950 DATA 5,7,130,98,100,4,7,130,114,1
10,2,96,130,106,118,4,5672
4960 DATA 7,130,98,100,6,7,137,13,7,1,
2,3,4,7,5,6,7508
4970 DATA 27,7,131,98,100,13,25,7,130,
98,100,2,7,130,98,100,4860
4980 DATA 4,0,130,120,112,10,96,130,8,
10,6,7,130,5,6,8,486
4990 DATA 7,130,98,100,4,7,130,98,100,
8,7,130,98,100,8,7,3230
5000 DATA 130,114,110,18,96,130,108,11
6,4,7,130,114,110,6,96,130,6642
5010 DATA 106,112,4,96,130,27,28,20,96
,130,102,100,2,7,130,98,5032
5020 DATA 100,4,0,5,7,130,5,6,16,7,130
,5,6,5,7,130,83
5030 DATA 98,100,4,7,130,98,100,8,7,13
2,98,100,5,6,2,7,1322
5040 DATA 130,5,6,2,7,130,98,100,6,7,1
38,5,6,7,1,2,9432
5050 DATA 3,4,7,5,6,2,7,130,98,100,4,7
,130,98,100,24,3189
5060 DATA 7,130,5,6,8,7,130,98,100,2,7
,130,98,100,4,0,2433
5070 DATA 130,114,110,6,96,130,108,116
,4,7,130,114,110,6,96,130,6664
5080 DATA 108,116,6,7,130,98,100,4,7,1
30,120,112,2,96,130,108,6511
5090 DATA 116,4,7,130,98,100,2,7,130,5
,6,2,7,132,5,6,335
5100 DATA 98,100,2,7,130,5,6,4,7,130,5
,6,8,7,132,98,1426
5110 DATA 100,8,10,2,7,130,98,104,10,9
6,130,108,116,2,7,136,5190
5120 DATA 13,7,1,2,3,4,1,2,3,7,132,5,6
,5,6,5,7147
5130 DATA 7,130,98,100,2,7,130,98,100,
4,0,130,98,100,6,7,3213
5140 DATA 130,98,100,2,7,132,5,6,98,10
0,6,7,130,98,100,6,3374
5150 DATA 7,130,98,100,8,7,130,98,100,
4,7,130,98,100,8,7,3370
5160 DATA 130,98,100,4,7,130,5,6,12,7,
130,98,100,4,7,130,3025
5170 DATA 98,100,2,7,130,5,6,6,7,130,9
8,100,6,7,130,98,3607
5180 DATA 100,2,7,130,5,6,10,7,130,98,
100,2,7,130,98,100,4267
5190 DATA 4,0,130,98,100,2,7,130,5,6,2
,7,130,120,112,4,2902
5200 DATA 96,132,106,118,5,6,2,7,134,5
,6,98,100,5,6,2,471
5210 DATA 7,135,5,6,98,100,7,5,6,5,7,1
30,98,100,4,7,1292
5220 DATA 130,98,100,8,7,130,120,112,8
,96,130,108,110,8,96,130,7249
5230 DATA 106,118,4,7,130,98,100,5,7,1
30,5,6,3,7,130,120,3087
5240 DATA 112,6,96,130,106,118,5,7,130
,5,6,7,7,134,98,100,3908
5250 DATA 14,15,98,100,4,0,130,98,100,
4,7,130,5,6,10,7,690
5260 DATA 130,5,6,2,7,130,98,100,2,7,1
30,5,6,2,7,130,1596
5270 DATA 120,112,8,96,130,106,112,4,9
6,130,102,100,4,7,130,5,4790
5280 DATA 6,3,7,130,5,6,2,7,130,5,6,3,
7,130,98,100,2267
5290 DATA 2,7,130,5,6,3,7,130,5,6,5,7,
130,98,104,4,1783
5300 DATA 96,130,108,116,11,7,130,5,6,
14,7,131,13,98,100,2,2407
5310 DATA 7,130,98,100,4,0,130,98,100,
6,7,130,114,110,6,96,5230
5320 DATA 132,1,2,3,4,2,7,130,98,100,9
,7,130,5,6,11,684
5330 DATA 7,130,98,100,6,7,130,5,6,10,
7,130,98,100,4,7,1950
5340 DATA 130,5,6,8,7,130,98,100,2,7,1
32,3,4,98,100,2,2363
5350 DATA 7,130,8,10,24,96,130,106,118
```

```
,2,7,130,98,100,4,0,3588
5360 DATA 130,98,100,3,7,133,5,6,7,98,
100,12,7,130,98,100,4182
5370 DATA 13,7,130,14,15,7,7,130,98,10
0,9,7,130,5,6,2,996
5380 DATA 7,130,5,6,3,7,130,98,100,7,7
,130,5,6,5,7,380
5390 DATA 130,98,100,5,7,130,14,15,23,
7,132,1,2,3,4,4,9002
5400 DATA 7,130,98,100,4,0,132,1,2,3,4
,4,7,130,120,112,2956
5410 DATA 12,96,130,106,112,12,96,130,
27,28,8,96,130,106,112,18,5677
5420 DATA 96,132,106,118,14,15,2,7,134
,1,2,3,4,3,4,4,8292
5430 DATA 96,130,106,112,38,96,130,106
,118,88,86,88,86,1,226,2,7574
5440 DATA 227,2,0,64,0,0,0,0,0,0,0,0,0
,0,0,0,5927
```

LISTING 2: ASSEMBLY

```
0100 ;SAVE#D:SNOW.PT1
0110 ;-------------------
0120 ; part 1 of the game
0130 ;
0140 ; by: Barry Kolbe
0150 ;
0160 ;-------------------
0170       .OPT NO LIST
0180 ;
0190 ;page zero variables
0200 ;
0210 DRY =    $82
0220 PXP =    $83         ;2byter
0230 PYP =    $85         ; "
0240 ADD =    $87
0250 HMV =    $88
0260 CURCHR = $89
0270 VERT =   $8A
0280 EADD =   $8B
0290 EHOLD =  $8C
0300 EYX =    $8D
0310 EYY =    $8E
0320 ICY =    $8F
0330 ICX =    $90
0340 VTIME =  $91
0350 VFLG =   $92
0360 XSP =    $93
0370 YSP =    $94
0380 RAND =   $95
0390 ICNT =   $96
0400 DATA =   $97
0410 UNIQUE = $98
0420 FTIME = $99          ;7 bytes
0430 CARTIM = $A0         ;6 bytes
0440 RMTIM = $A6          ;6 bytes
0450 ROADC = $AC          ;2 bytes
0460 ROADG = $AE          ;  "
0470 CNT =    $B0
0480 SHD =    $B1
0490 MAPFLG = $B2
0500 FUEL =   $B3         ;3 bytes
0510 LIVES =  $B6
0520 DIRF =   $B7
0530 DIRECT = $B8
0540 CARCNT = $B9
0550 ICEON = $BA
0560 Y1  =    $BB
0570 Y2  =    $BC
0580 ICEDIR = $BD
0590 IDECNT = $BE
0600 LEVEL = $BF
0610 FULK =   $C0
0620 ICTL =   $C1
0630 ICTH =   $C2
0640 ICESPEED = $C3
0650 THOUS = $C4
0660 SPSND = $C5
0670 IFLAG = $C6
0680 IISY =   $C7
0690 IIY3 =   $C8
0700 COUNT = $C9          ;these are
0710 IL  =    $CB
0720 TL  =    $CB         ;all
0730 JL  =    $CD
0740 BFL =    $CD
0750 INDR =   $CB         ;2 bytes
0760 IND2 =   $CD
0770 FOFS =   $CF
0780 YH  =    $D0
0790 XD1 =    $D1
0800 XH  =    $D2
0810 ICECNT = $D3
0820 SCRS =   $D4         ;6 bytes
0830 FULX =   $DA         ;6 bytes
0840 FULY =   $E0         ;6 bytes
0850 CARX =   $E6         ;4 bytes
0860 CARY =   $EA         ;4 bytes
0870 OFFSCN = $EE         ;2 bytes
0880 SLENGTH = $F0
0890 EDM =    $F1         ;2
0900 SXP =    $F3
0910 SYP =    $F4
0920 ;
0930 ;OS equates
0940 ;
0950 CIOV =  $E456
0960 ICCOM = $0342
0970 ICBAL = $0344
0980 ICBAH = $0345
0990 ICBLL = $0348
1000 ICBLH = $0349
1010 AUX1 = $034A
1020 AUX2 =  $034B
1030 SETVBV = $E45C
1040 XITVBV = $E462
1050 RANDOM = $D20A
1060 RTCLOK = $14
1070 CONSOL = $D01F
1080 EOL = $9B
1090 SEOL =  $BB
1100 PCHR =  54
1110 ;
1120 COLOR0 = $02C4
1130 COLOR1 = $02C5
1140 COLOR2 = $02C6
1150 COLOR3 = $02C7
1160 COLOR4 = $02C8
1170 SDLSTL = $0230
1180 CH   =  $02FC
1190 ;
1200 STICK = $0278
1210 STRIG = $0284
1220 HPOSP0 = $D000
1230 HPOSM0 = $D004
1240 HSCROL = $D404
1250 VSCROL = $D405
1260 HITCLR = $D01E
1270 P0PL =  $D00C
1280 P1PL =  $D00D
1290 ;
1300 ;memory usage
1310 ;
1320 SCNMEM = $9000
1330 PMB =    $8000
1340 SET =    $8C00
1350 MYPMB = $8400
1360 MSMEM = PMB+$0300
1370 P0MEM = MYPMB
1380 P1MEM = MYPMB+$0100
1390 P2MEM = MYPMB+$0200
1400 P3MEM = MYPMB+$0300
1410 CHSET = $8C00
1420 GOVER = $9B00
1430 TXTWIN = $8B00
1440 SCLN =   TXTWIN+20
1450 CHRLN1 = TXTWIN+40
1460 SC2 =    SCNMEM+128
1470 SC3 =    SC2+128
1480 SC4 =    SC3+128
1490 SC5 =    SC4+128
1500 SC6 =    SC5+128
1510 SC7 =    SC6+128
1520 SC8 =    SC7+128
1530 SC9 =    SC8+128
1540 SC10 =   SC9+128
1550 SC11 =   SC10+128
1560 BCKUP = $7FC0
1570 SET2 = $7D00
1580 SCRLMEM = $7C00
1590 ;credits @ $7b00
1600 ;
1610 ;display list interrupt
1620 ;
1630       *=  PMB
1640 DLI PHA
```

```
1650 DIC LDA #2
1660      STA $D40A
1670      STA $D01A
1680      PLA
1690      RTI
1700 ;
1710 ;character set
1720 ;
1730      *=  $8C00
1740      .INCLUDE #D:SNOW.PT5
1750 ;
1760      *=  TXTWIN
1770      .SBYTE " FUEL 150 PLOWS "
1780      .SBYTE "3         SCORE "
1790      .SBYTE "000000          "
1800 ;
1810      *=  GOVER
1820      .SBYTE "   SNOWFLAKES "
1830      .SBYTE "WIN          press "
1840      .SBYTE "START           "
1850 ;
1860 ;display lists ;intro first
1870 ;then game board
1880 ;
1890      *=  $7F00
1900 IDLST .BYTE $70,$70,$70,$70
1910      .BYTE $70,$70,$70,$42
1920      .WORD SNOWMEM
1930      .BYTE $02,$70,$70,$70,$02
1940      .BYTE $02,$70,$70,$70,$70
1950      .BYTE $57
1960 LMS .WORD SCRLMEM
1970      .BYTE $70,$70,$70,$70,$46
1980      .WORD TOPSCORE
1990      .BYTE $41
2000      .WORD IDLST
2010 ;
2020      *=  $7F80
2030 DL1 .BYTE $70,$70,$70
2040      .BYTE $42
2050      .WORD SNOWMEM
2060      .BYTE 2
2070      .BYTE $75
2080 SL1 .WORD SCNMEM
2090      .BYTE $75
2100      .WORD SC2
2110      .BYTE $75
2120      .WORD SC3
2130      .BYTE $75
2140      .WORD SC4
2150      .BYTE $75
2160      .WORD SC5
2170      .BYTE $75
2180      .WORD SC6
2190      .BYTE $75
2200      .WORD SC7
2210      .BYTE $75
2220      .WORD SC8
2230      .BYTE $75
2240      .WORD SC9
2250      .BYTE $75
2260      .WORD SC10
2270      .BYTE $D5
2280      .WORD SC11
2290      .BYTE $46
2300 TMESS .WORD TXTWIN
2310      .BYTE 6,$41
2320      .WORD DL1
2330 ;
2340 ;start of program
2350 ;
2360      *=  $4000
2370 ;
2380 ;save the display list
2390 ;for next levels
2400 ;
2410 BEGIN LDY #50
2420 MDL LDA DL1,Y
2430      STA BCKUP,Y
2440      DEY
2450      BPL MDL
2460      LDA # >CHSET ;new chrset
2470      STA 756
2480      JSR SETPMG  ;player init
2490 ;
2500      LDX # >VBLNK ;set vbi
2510      LDY # <VBLNK
2520      LDA #7
2530      JSR SETVBV
2540      LDA #0
2550      STA IFLAG   ;intro flag
2560 ;
2570 ;begin new level
2580 ;
2590 NEWBEG LDA #0
2600      STA DIRF
2610      STA MAPFLG
2620      JSR GETDIR
2630      JMP NEWB
2640 NDIRC JSR CLOSE6
2650 ;
2660 NEWB JSR INIT
2670      JSR REPLAY
2680      LDA #3
2690      STA LIVES
2700      LDA #0
2710      STA ICEON
2720      JSR SHOLIV
2730 NLEVL JSR STARTI
2740      JSR FIGLEV
2750      JSR RESFUL
2760      JSR DSPFUL
2770      JSR REPLAY
2780      JSR CLRPM
2790      LDA DIRF
2800      BNE DROK
2810 MAP2 LDA MAPFLG ;intern map
2820      BNE DROK
2830      LDA #1
2840      STA MAPFLG
2850      JSR UNCOM
2860      JMP SKP
2870 DROK JSR GETFIL
2880      JSR LOADMP
2890 SKP JSR FNDFUL
2900      JSR FNDCRS
2910      JSR CNTRDS
2920      JSR COPYDL
2930      JSR SETSCN
2940      LDA #$0B
2950      STA DIRECT
2960      JSR DEFPLR
2970      LDA #1
2980      STA VFLG
2990 ;
3000 ;main loop
3010 ;
3020 MAIN LDA CONSOL
3030      CMP #6
3040      BNE MN2
3050      JMP STKEY
3060 MN2 LDA STICK
3070      STA DIRECT
3080      JSR DEFPLR
3090      STA HITCLR
3100      JSR MOVPLR
3110      JSR REMOVE
3120      LDA 644
3130      BNE NTRG
3140      JSR JMPEDG
3150 NTRG JSR ICESND
3160      LDA P0PL
3170      CMP #12
3180      BCS OUCH
3190      LDA P1PL
3200      CMP #12
3210      BCS OUCH
3220      LDA ROADG+1
3230      CMP ROADC+1
3240      BNE KL
3250      LDA ROADG
3260      CMP ROADC
3270      BNE KL
3280      JSR WAITSM
3290      JMP NLEVL
3300 KL   LDA FUEL
3310      ORA FUEL+1
3320      ORA FUEL+2
3330      BNE KK
3340 OUCH STA HITCLR
3350      JSR REMP23
3360      JSR SPIN
3370      JSR GRESET
3380      STA HITCLR
3390      JSR DELAY
3400      JMP KK
3410 OOPS JMP GAMOVR
3420 KK   LDA LIVES
```

```
3430      BEQ OOPS
3440      LDA CH
3450      CMP #$FF
3460      BEQ NOPAUS
3470      JSR PAUSE
3480 NOPAUS JMP MAIN
3490 ;
3500 ;set pmg
3510 ;
3520 SETPMG LDA #62
3530      STA 559
3540      LDA #17       ;5th enable
3550      STA 623
3560      LDA #3
3570      STA 53277
3580      LDA # >PMB
3590      STA 54279
3600      LDA #$28      ;yellow
3610      STA 704
3620      LDA #6        ;black
3630      STA 705
3640      LDA #$A6
3650      STA 706
3660      STA 707
3670      LDA #$80
3680      STA PXP
3690      LDA #$70
3700      STA PYP
3710      LDA #1
3720      STA 53258
3730      STA 53259
3740      RTS
3750 ;
3760 ;clear pm area
3770 ;
3780 CLRPM LDY #$FF
3790      LDA #0
3800 CPM STA P0MEM,Y
3810      STA P1MEM,Y
3820      STA P2MEM,Y
3830      STA P3MEM,Y
3840      STA MSMEM,Y
3850      DEY
3860      CPY #$FF
3870      BNE CPM
3880      RTS
3890 ;
3900 ;define plrs
3910 ;
3920 DEFPLR LDX #3
3930      LDA DIRECT
3940 DF1 CMP DIRTAB,X
3950      BEQ DFOK
3960      DEX
3970      BPL DF1
3980      RTS
3990 DFOK LDA PD0L,X
4000      STA JL
4010      LDA PD0H,X
4020      STA JL+1
4030      LDA PD1L,X
4040      STA IL
4050      LDA PD1H,X
4060      STA IL+1
4070      TXA
4080      PHA
4090      LDA PYP
4100      STA Y1
4110      CLC
4120      ADC YTB,X
4130      STA PYP+1
4140      STA Y2
4150      TAY
4160      LDA #0
4170      STA P1MEM-1,Y
4180      LDY #0
4190 DFP LDA (JL),Y
4200      LDX Y1
4210      STA P0MEM,X
4220      LDA (IL),Y
4230      LDX Y2
4240      STA P1MEM,X
4250      INY
4260      INC Y1
4270      INC Y2
4280      CPY #16
4290      BNE DFP
4300      LDX Y2
4310      STA P1MEM,X
```

```
4320      LDX PXP
4330      STX HPOSP0
4340      PLA
4350      TAX
4360      LDA PXP
4370      CLC
4380      ADC XTB,X
4390      STA PXP+1
4400      STA HPOSP0+1
4410      RTS
4420 ;
4430 DIRTAB .BYTE 11,7,14,13
4440 XTB .BYTE $FF,$01,$00,$00
4450 YTB .BYTE $00,$00,$FF,$01
4460 ;
4470 ;player definitions & tables
4480 ;
4490 PD0L .BYTE  <P0D1, <P0D2, <P0D3,
<P0D4
4500 PD0H .BYTE  >P0D1, >P0D2, >P0D3,
>P0D4
4510 PD1L .BYTE  <P1D1, <P1D2, <P1D3,
<P1D4
4520 PD1H .BYTE  >P1D1, >P1D2, >P1D3,
>P1D4
4530 P0D1 .BYTE $00,$00,$00,$FF
4540      .BYTE $FF,$76,$76,$76
4550      .BYTE $76,$76,$76,$FF
4560      .BYTE $FF,$00,$00,$00
4570 P0D2 .BYTE $00,$00,$00,$FF
4580      .BYTE $FF,$6E,$6E,$6E
4590      .BYTE $6E,$6E,$6E,$FF
4600      .BYTE $FF,$00,$00,$00
4610 P0D3 .BYTE $00,$66,$66,$7E
4620      .BYTE $7E,$7E,$7E,$7E
4630      .BYTE $7E,$66,$66,$66
4640      .BYTE $7E,$7E,$66,$00
4650 P0D4 .BYTE $00,$66,$7E,$7E
4660      .BYTE $66,$66,$66,$7E
4670      .BYTE $7E,$7E,$7E,$7E
4680      .BYTE $7E,$66,$66,0
4690 ;
4700 P1D1 .BYTE $00,$AA,$AA,$80
4710      .BYTE $80,$84,$84,$84
4720      .BYTE $84,$84,$84,$80
4730      .BYTE $80,$AA,$AA,$00
4740 P1D2 .BYTE $00,$FD,$FD,$01
4750      .BYTE $01,$61,$61,$61
4760      .BYTE $61,$61,$61,$01
4770      .BYTE $01,$FD,$FD,$00
4780 P1D3 .BYTE 0,$FF,$FF,$00
4790      .BYTE $99,$99,$18,$18
4800      .BYTE $99,$99,$18,$18
4810      .BYTE $99,$99,$00,$00
4820 P1D4 .BYTE $00,$00,$81,$99
4830      .BYTE $18,$18,$99,$99
4840      .BYTE $18,$18,$99,$99
4850      .BYTE $00,$FF,$FF,$00
4860 ;
4870 ;delay
4880 ;
4890 LDL LDY #19
4900      BNE DLN
4910 DELAY LDY #7
4920 DLN JSR WAIT
4930      DEY
4940      BNE DLN
4950      RTS
4960 ;
4970 ;initialize
4980 ;
4990 ;set up 128 table
5000 INIT LDY #0
5010      LDA # <SCNMEM
5020      STA M128L
5030      LDA # >SCNMEM
5040      STA M128H
5050 TLP LDA M128L,Y
5060      CLC
5070      ADC #128
5080      STA M128L+1,Y
5090      LDA M128H,Y
5100      ADC #0
5110      STA M128H+1,Y
5120      INY
5130      CPY #33
5140      BNE TLP
5150 ;
5160      LDX #7        ;score & fuel
```

```
5170       LDA #$10      ;IC $10
5180 SCF   STA SCRS,X
5190       DEX
5200       BPL SCF
5210       STA THOUS
5220       JSR RESFUL
5230       LDA #3        ;set lives
5240       STA LIVES
5250       LDA #$13
5260       STA TXTWIN+16
5270       LDA #$60
5280       STA ICECNT
5290       LDA #$FF
5300       STA LEVEL
5310       RTS
5320 ;
5330 CLRTAB .BYTE $D8,$44,$0A,$46,$00
5340 ;
5350 STSCRL LDA # >SL1
5360       STA JL+1
5370       LDA # <SL1
5380       STA JL
5390       RTS
5400 ;
5410 ;get scn pos
5420 ;
5430 WHER LDY DRY
5440 WHERE LDA M128L,Y
5450       STA JL
5460       LDA M128H,Y
5470       STA JL+1
5480       RTS
5490 ;
5500 ;shadow scrols
5510 ;
5520 SMX .BYTE 22      ;pos on scrn
5530 SMY .BYTE 4
5540 ;
5550 ;move
5560 ;
5570 SCROLL LDA DIRECT
5580 ;
5590 MMR  CMP #7       ;rt?
5600      BNE MML
5610      JSR LOOKR
5620      BCC RA1
5630      RTS
5640 RA1  LDA SXP
5650      CMP #83
5660      BCC HO
5670      RTS
5680 HO   LDA #1
5690      STA VERT
5700      JSR PLWSND
5710 KB   LDA #1
5720      STA ADD
5730      JSR CORSH
5740      INC SMX
5750      INC SXP
5760      LDX #3
5770 KA   STX HSCROL
5780      JSR WAIT
5790      JSR TRACKR
5800      DEX
5810      BPL KA
5820      DEC VERT
5830      LDA VERT
5840      BPL KB
5850 FULOUT JSR SFUEL
5860      JSR OFF2
5870      JSR CKFTIM
5880      RTS
5890 ;
5900 MML  CMP #11
5910      BNE MUP
5920      JSR LOOKL
5930      BCC LA1
5940      RTS
5950 LA1  LDA SXP
5960      BNE HQ
5970      RTS
5980 HQ   LDA #1
5990      STA VERT
6000      JSR PLWSND
6010 KD   LDX #0
6020 KC   STX HSCROL
6030      JSR WAIT
6040      JSR TRACKL
6050      INX
```

```
6060      CPX #4
6070      BNE KC
6080      LDA #$FF
6090      STA ADD
6100      LDX #0
6110      STX HSCROL
6120      JSR CORSH
6130      DEC SXP
6140      DEC SMX
6150      DEC VERT
6160      LDA VERT
6170      BPL KD
6180      JMP FULOUT
6190 ;
6200 ;coarse h scrol
6210 ;
6220 CORSH LDY #0
6230 CSA  LDA SL1,Y
6240      CLC
6250      ADC ADD
6260      STA SL1,Y
6270      INY
6280      INY
6290      INY
6300      CPY #33
6310      BNE CSA
6320      RTS
6330 ;
6340 ;up
6350 ;
6360 MUP  CMP #14
6370      BNE MDN
6380      JSR LOOKU
6390      BCC UA1
6400      RTS
6410 UA1  LDA SYP
6420      BEQ NOM
6430      JSR PLWSND
6440      LDA #$FF
6450      STA ADD
6460      JSR CORSV
6470      DEC SYP
6480      DEC SMY
6490      LDX #15
6500 U1   STX VSCROL
6510      JSR WAIT
6520      JSR TRACKU
6530      DEX
6540      BPL U1
6550      LDA #0
6560      STA VSCROL
6570      JMP FULOUT
6580      RTS
6590 ;
6600 ;down
6610 ;
6620 MDN  CMP #13
6630      BNE NOM
6640      JSR LOOKD
6650      BCC DA1
6660      RTS
6670 DA1  LDA SYP
6680      CMP #9
6690      BCC D1
6700      RTS
6710 D1   JSR PLWSND
6720      LDX #0
6730 D2   STX VSCROL
6740      JSR WAIT
6750      JSR TRACKU
6760      INX
6770      CPX #16
6780      BNE D2
6790      LDX #0
6800      STX VSCROL
6810      LDA #1
6820      STA ADD
6830      JSR CORSV
6840      INC SYP
6850      INC SMY
6860      JMP FULOUT
6870 NOM  RTS
6880 ;
6890 ;coarse vrt scrl
6900 ;
6910 CORSV LDA ADD
6920      BMI CSV
6930      LDY #0
6940 CVL  LDA SL1,Y
```

```
6950      CLC
6960      ADC #$80
6970      STA SL1,Y
6980      INY
6990      LDA SL1,Y
7000      ADC #0
7010      STA SL1,Y
7020      INY
7030      INY
7040      CPY #33
7050      BNE CVL
7060      RTS
7070 ;
7080 CSV LDY #0
7090 CVP LDA SL1,Y
7100      SEC
7110      SBC #$80
7120      STA SL1,Y
7130      INY
7140      LDA SL1,Y
7150      SBC #0
7160      STA SL1,Y
7170      INY
7180      INY
7190      CPY #33
7200      BNE CVP
7210      RTS
7220 ;
7230 ;wait a jiffy
7240 ;
7250 WAIT LDA #0
7260      STA $14
7270 WW2 LDA $14
7280      BEQ WW2
7290      RTS
7300 ;
7310 ;fine scrl plr
7320 ;
7330 MOVPLR LDA DIRECT
7340      CMP #7
7350      BNE MLL
7360      LDA PXP
7370      CMP #$7C
7380      BCC GOR
7390      LDA SXP
7400      CMP #83
7410      BCS GOR
7420      JMP MOUT
7430 GOR LDA PXP
7440      CMP #$C8
7450      BCS MVRET
7460      JSR LOOKR
7470      BCC GOR1
7480      RTS              ;nope
7490 GOR1 JSR PLWSND
7500      LDY #7
7510 RL1 INC PXP
7520      LDX PXP
7530      STX HPOSP0
7540      INC PXP+1
7550      LDX PXP+1
7560      STX HPOSP0+1
7570      JSR WAIT
7580      JSR TRACKR
7590      DEY
7600      BPL RL1
7610      INC SMX
7620      INC SMX
7630      JMP FULOUT
7640 MOUT JSR SCROLL
7650 MVRET RTS
7660 ;
7670 MLL CMP #11
7680      BNE MDD
7690      JSR LOOKL
7700      BCC GOL2
7710      RTS
7720 GOL2 LDA PXP
7730      CMP #132         ;124+8
7740      BCS GOL1
7750      LDA SXP
7760      BNE MOUT
7770 GOL LDA PXP
7780      CMP #$30
7790      BEQ MVRET
7800      BCC MVRET
7810 GOL1 JSR PLWSND
7820      LDY #7
7830 LL1 DEC PXP
```

```
7840      LDX PXP
7850      STX HPOSP0
7860      DEC PXP+1
7870      LDX PXP+1
7880      STX HPOSP0+1
7890      JSR WAIT
7900      JSR TRACKL
7910      DEY
7920      BPL LL1
7930      DEC SMX
7940      DEC SMX
7950      JMP FULOUT
7960 ;
7970 MDD CMP #13
7980      BNE MUU
7990      JSR LOOKD
8000      BCC GD3
8010      RTS
8020 GD3 LDA PYP
8030      CMP #$60
8040      BCC GDN
8050      LDA SYP
8060      CMP #9
8070      BCS GDN
8080      JMP MOUT
8090 GDN LDA PYP
8100      CMP #$C0
8110      BCS MVRET
8120      LDA #15
8130      STA VERT
8140      JSR PLWSND
8150 GD2 LDA PYP
8160      STA Y1
8170      LDA PYP+1
8180      STA Y2
8190      TAY
8200      LDA #0
8210      STA P1MEM,Y ;eras top
8220      LDX #15
8230 GD1 LDY Y1
8240      LDA P0MEM+15,Y
8250      STA P0MEM+16,Y
8260      LDY Y2
8270      LDA P1MEM+15,Y
8280      CMP #$18
8290      BNE FOA
8300      LDA #$99
8310      BNE FOB
8320 FOA CMP #$99
8330      BNE FOB
8340      LDA #$18
8350 FOB STA P1MEM+16,Y
8360      DEC Y1
8370      DEC Y2
8380      DEX
8390      BPL GD1
8400      INC PYP
8410      INC PYP+1
8420      JSR WAIT
8430      DEC VERT
8440      LDA VERT
8450      BPL GD2
8460      INC SMY
8470      JMP FULOUT
8480 ;
8490 ;
8500 MUU CMP #14
8510      BEQ MU3
8520 GU4 JMP MVRET
8530 MU3 JSR LOOKU
8540      BCC MU4
8550      RTS
8560 MU4 LDA PYP
8570      CMP #$70
8580      BCS GUU
8590      LDA SYP
8600      BEQ GUU
8610      JMP MOUT
8620 GUU LDA PYP
8630      CMP #$30
8640      BNE GU5
8650      JMP MOUT
8660 GU5 JSR PLWSND
8670      LDA #15
8680      STA VERT
8690 GU2 LDY PYP
8700      LDX #15
8710 GU1 LDA P0MEM,Y
8720      STA P0MEM-1,Y
```

```
8730      LDA P1MEM,Y
8740      CMP #$18
8750      BNE FOC
8760      LDA #$99
8770      BNE FOD
8780 FOC  CMP #$99
8790      BNE FOD
8800      LDA #$18
8810 FOD  STA P1MEM-1,Y
8820      INY
8830      DEX
8840      BPL GU1
8850      JSR WAIT
8860      DEC PYP
8870      DEC VERT
8880      LDA VERT
8890      BPL GU2
8900      DEC SMY
8910      JMP FULOUT
8920 ;
8930 ;locate :result in curchr
8940 ;
8950 LOOKR LDY SMY
8960      JSR WHERE
8970      LDY SMX
8980      INY
8990      INY
9000 LKRET LDA (JL),Y
9010      STA CURCHR
9020      JSR CKCHAR
9030      RTS
9040 ;
9050 LOOKL LDY SMY
9060      JSR WHERE
9070      LDY SMX
9080      DEY
9090      DEY
9100      JMP LKRET
9110 ;
9120 LOOKD LDY SMY
9130      INY
9140      JSR WHERE
9150      LDY SMX
9160      JMP LKRET
9170 ;
9180 LOOKU LDY SMY
9190      DEY
9200      JSR WHERE
9210      LDY SMX
9220      JMP LKRET
9230 ;
9240 ;erase chr beneath dozer
9250 ;repalce with next char.
9260 ;which is a road w/o snow
9270 ;
9280 REMOVE LDY SMY
9290      JSR WHERE
9300      LDY SMX
9310      LDA (JL),Y
9320      LDX #15
9330 RB1  CMP ROADS,X
9340      BEQ SF5
9350      DEX
9360      BPL RB1
9370      JMP SF6
9380 SF5  CMP #$1B        ;cng car to road
9390      BEQ SF8
9400      CMP #$1D
9410      BNE SF7
9420 SF8  PHA             ;save car
9430      LDA #$61
9440      STA (JL),Y
9450      INY
9460      STA (JL),Y
9470      JSR BONSCR
9480      JSR HORN
9490      PLA
9500      CMP #$1B
9510      BEQ SFA
9520      BNE SF6
9530 SF7  TAX
9540      INX
9550      TXA
9560      STA (JL),Y
9570      INY
9580      LDA (JL),Y
9590      TAX
9600      INX
9610      TXA
9620      STA (JL),Y
9630      CMP #$0B        ;     fuel?
9640      BEQ SF6
9650 SFA  INC ROADG
9660      BNE SH1
9670      INC ROADG+1
9680 ;
9690 SH1  JSR DOSCOR      ;show score
9700 ;
9710 SF6  LDA CURCHR
9720      CMP #$08        ;fuel?
9730      BNE CAR?
9740 ;
9750      INC CURCHR      ;empty fuel
9760      LDY SMY
9770      JSR WHERE
9780      LDX FOFS        ;get pos.
9790 FFA  LDA JL+1
9800      CMP FULY,X
9810      BEQ FFB
9820 FFC  DEX
9830      BPL FFA
9840      RTS             ;shouldnt happen
9850 FFB  LDA SMX         ;put it on
9860      CLC             ;screen
9870      ADC JL
9880      CMP FULX,X
9890      BNE FFC
9900      LDA #100        ;got it
9910      STA FTIME,X     ;set timer
9920      JSR FILSND      ;sound
9930      JSR RESFUL      ;show it
9940      JSR DSPFUL
9950 SF2  RTS
9960 ;
9970 ;hit a car
9980 ;
9990 CAR? LDY SMY         ;find out
010000    JSR WHERE       ;which one
010010    LDX CARCNT
010020    BEQ SF2
010030 CB1 LDA JL+1
010040    CMP CARY,X
010050    BEQ CB2
010060 CB3 DEX
010070    BPL CB1
010080    RTS             ;nope
010090 CB2 LDA JL
010100    CLC
010110    ADC SMX
010120    CMP CARX,X
010130    BNE CB3
010140    LDA #50         ;set timer
010150    STA CARTIM,X
010160    RTS
010170 ;
010180 ;road,fuel,car chrs
010190 ; 8 = fuel $1b= car
010200 ;
010210 ROADS .BYTE $60,$62,$64,$66
010220    .BYTE $68,$6A,$6C,$6E
010230    .BYTE $70,$72,$74,$76
010240    .BYTE $78,$08,$1B,$1D
010250    .BYTE $61,$63,$65,$67,$69
010260    .BYTE $6B,$6D,$6F,$71
010270    .BYTE $73,$75,$77,$79
010280 ;
010290 ;check if about to move
010300 ;onto a valid char.
010310 ;
010320 CKCHAR LDX #28
010330 CK1 CMP ROADS,X
010340    BEQ CROK
010350    DEX
010360    BPL CK1
010370    SEC             ;no move
010380    RTS
010390 CROK CLC           ;move ok
010400    RTS
010410 ;
010420 ;decrease fuel
010430 ;
010440 SFUEL DEC FUEL+2
010450    LDA FUEL+2
010460    ORA FUEL+1
010470    ORA FUEL
010480    BEQ SF3
010490    LDA FUEL+2
010500    BPL SF3
```

```
010510    LDA #9
010520    STA FUEL+2
010530    DEC FUEL+1
010540    LDA FUEL+1
010550    BPL SF3
010560    LDA #9
010570    STA FUEL+1
010580    DEC FUEL
010590 SF3 JSR DSPFUL
010600 FRET RTS
010610 ;
010620 ;show fuel left
010630 ;
010640 DSPFUL LDX #2
010650 DS LDA FUEL,X
010660    ORA #$10
010670    STA TXTWIN+6,X
010680    DEX
010690    BPL DS
010700    RTS
010710 ;
010720 ;wait some
010730 ;
010740 WAITSM LDA #0
010750    LDX #15      ;15 secs
010760    STA VFLG
010770 WSM JSR LDL
010780    DEX
010790    BPL WSM
010800    RTS
010810 ;
010820 ;pt2 is rest of game
010830 ;pt3 is the screen maker
010840 ;pt4 is the introduction
010850 ;
010860    .INCLUDE #D:SNOW.PT2
010870    .INCLUDE #D:SNOW.PT4
010880    .INCLUDE #D:SNOW.PT3
010890 ;
010900 ;variables
010910 ;
010920 RANDS .DS 16
010930 DBUF .DS 20
010940 EDGFLG .BYTE 1
010950 M128L .DS 36
010960 M128H .DS 36
010970    *=  $02E0
010980    .WORD BEGIN
010990    .END
```

LISTING 3: ASSEMBLY

```
0100 ;SAVE#D:SNOW.PT2
0110 ;----------------
0120 ; part 2 of game
0130 ;
0140 ; by:Barry Kolbe
0150 ;
0160 ;----------------
0170 ;
0180 ;load a map from disk
0190 ;
0200 LOADMP LDX #$10
0210    LDA #$0C
0220    STA ICCOM,X
0230    JSR CIOV
0240    LDX #$10
0250    LDA #3
0260    STA ICCOM,X
0270    LDA # >MAPNAM
0280    STA ICBAH,X
0290    LDA # <MAPNAM
0300    STA ICBAL,X
0310    LDA #4
0320    STA AUX1,X
0330    LDA #0
0340    STA AUX2,X
0350    JSR CIOV
0360    BPL RDOK
0370    RTS
0380 RDOK LDX #$10
0390    LDA # >SCNMEM
0400    STA ICBAH,X
0410    LDA # <SCNMEM
0420    STA ICBAL,X
0430    LDA #0
0440    LDA #10      ;10 pages
0450    STA ICBLH,X
0460    LDA #7          ;read it
0470    STA ICCOM,X
0480    JSR CIOV
0490    RTS
0500 ;
0510 MAPNAM .BYTE "D1:SMAP.   ",EOL
0520 ;
0530 ;count all the roads
0540 ;each pair of road bytes
0550 ;counts as 1 point
0560 ;
0570 CNTRDS LDY #0
0580    STY ROADC
0590    INC ROADC
0600    STY ROADC+1
0610    STY CNT         ;lines
0620    JSR WHERE
0630 CR4 LDY #0
0640 CR1 LDA (JL),Y
0650    LDX #12
0660 CR3 CMP ROADS,X ;is it road?
0670    BEQ CR2
0680    DEX
0690    BPL CR3
0700 CR6 INY            ;skip over 2
0710    INY
0720    CPY #0
0730    BNE CR1
0740    INC JL+1
0750    INC CNT
0760    LDY CNT
0770    CPY #10
0780    BNE CR4
0790    RTS
0800 CR2 INC ROADC
0810    BNE CR5
0820    INC ROADC+1
0830 CR5 JMP CR6
0840 ;
0850 ;copy the game dlist
0860 ;back for next board
0870 ;
0880 COPYDL LDY #50
0890 CC1 LDA BCKUP,Y
0900    STA DL1,Y
0910    DEY
0920    BPL CC1
0930    RTS
0940 ;
0950 ;reset fuel in text window
0960 ;
0970 RESFUL LDA #0
0980    STA FUEL+2
0990    LDA #1
1000    STA FUEL
1010    LDA FULK
1020    STA FUEL+1
1030    RTS
1040 ;
1050 ;find fuel containers
1060 ;in map-store their
1070 ;positions
1080 ;
1090 FNDFUL LDY #0
1100    STY CNT
1110    STY FOFS
1120 SS4 LDY CNT
1130    CPY #19         ;20 lines
1140    BNE SS3
1150    RTS
1160 SS3 JSR WHERE
1170    LDY #0
1180 SS2 LDA (JL),Y
1190    CMP #8          ;fuel chr
1200    BEQ SS1
1210 SS5 INY
1220    INY
1230    CPY #126        ;end of line?
1240    BNE SS2
1250    INC CNT
1260    JMP SS4
1270 ;
1280 SS1 LDX FOFS       ;store x,y
1290    LDA JL+1        ;positions
1300    STA FULY,X
1310    TYA
1320    CLC
1330    ADC JL
1340    STA FULX,X
```

```
1350       INC FOFS
1360       LDA FOFS
1370       CMP #6        ;only 6 allowed
1380       BNE SSS
1390       RTS
1400 ;
1410 ;check timers-fuel first
1420 ;
1430 CKFTIM LDX FOFS
1440       BEQ CCB       ;no fuel
1450 FTC LDA FTIME,X
1460       BEQ FTB
1470       DEC FTIME,X
1480       LDA FTIME,X
1490       BNE FTB
1500       LDA FULY,X
1510       STA JL+1
1520       LDA FULX,X
1530       STA JL
1540       LDY #0
1550       LDA #8
1560       STA (JL),Y
1570       INY
1580       LDA #$0A
1590       STA (JL),Y
1600 FTB DEX
1610       BPL FTC
1620 ;
1630 ;check bonus car timers
1640 ;
1650 CCB LDX CARCNT
1660       BNE CTC       ;yes
1670       RTS           ;no cars
1680 CTC LDA CARTIM,X
1690       BEQ CTD
1700       DEC CARTIM,X
1710       LDA CARTIM,X
1720       BNE CTD
1730       LDA CARY,X
1740       STA JL+1
1750       LDA CARX,X
1760       STA JL
1770       LDY #0        ;put bonus
1780       LDA (JL),Y    ;car on scn
1790       CMP #$60      ;snow bckgrnd?
1800       BNE CTF
1810       LDA #$1B      ;yes
1820       STA (JL),Y
1830       INY
1840       LDA #$1C
1850 CTG STA (JL),Y
1860       LDA #$50
1870       STA RMTIM,X
1880       JSR BELL
1890 CTD DEX
1900       BPL CTC
1910 ;
1920 ;timers to remove cars
1930 ;
1940       LDX CARCNT
1950 CTK LDA RMTIM,X
1960       BEQ RRA
1970       DEC RMTIM,X
1980       LDA RMTIM,X
1990       BNE RRA
2000       LDA CARY,X
2010       STA JL+1
2020       LDA CARX,X
2030       STA JL
2040       LDY #0
2050       LDA (JL),Y
2060       CMP #$1B
2070       BNE CTI
2080       LDA #$60      ;snow road
2090 CTM STA (JL),Y
2100       INY
2110       STA (JL),Y
2120       LDA #100      ;reset timer
2130       STA CARTIM,X
2140 RRA DEX
2150       BPL CTK
2160       RTS
2170 ;
2180 CTI LDA #$61      ;plain road
2190       BNE CTM
2200       RTS
2210 CTF LDA #$1D
2220       STA (JL),Y
2230       INY
```

```
2240       LDA #$1E
2250       BNE CTG
2260 ;
2270 ;find bonus cars on map
2280 ;
2290 FNDCRS LDY #0
2300       STY CNT
2310       STY CARCNT
2320 FCA LDY CNT
2330       CPY #19       ;20 lines
2340       BNE FCE
2350       RTS
2360 FCE JSR WHERE
2370       LDY #0
2380 FCC LDA (JL),Y
2390       CMP #$1B      ;cars
2400       BEQ FCB
2410 FCD INY
2420       INY
2430       CPY #126      ;end of line?
2440       BNE FCC
2450       INC CNT
2460       JMP FCA
2470 FCB LDX CARCNT
2480       LDA JL+1
2490       STA CARY,X
2500       TYA
2510       CLC
2520       ADC JL
2530       STA CARX,X
2540       TYA
2550       PHA
2560       LDA #$60      ;repl w/road
2570       STA (JL),Y
2580       INY
2590       STA (JL),Y
2600       PLA
2610       TAY
2620       INC CARCNT
2630       LDA CARCNT
2640       CMP #4        ;only 4 cars
2650       BNE FCD
2660       RTS
2670 ;
2680 ;show bonus & regular score
2690 ;
2700 BONSCR LDX #3    ;100 bonus
2710       BNE SCD
2720 ;
2730 DOSCOR LDX #4    ;reg. score
2740 SCD LDA SCRS,X
2750       CLC
2760       ADC #1
2770       STA SCRS,X
2780 SCB CMP #$1A
2790       BCC SCE
2800       LDA #$10
2810       STA SCRS,X
2820       DEX
2830       BMI SCE
2840       INC SCRS,X
2850       LDA SCRS,X
2860       JMP SCB
2870 SCE LDX #5
2880 SCC LDA SCRS,X
2890       STA SCLN+9,X
2900       DEX
2910       BPL SCC
2920       LDA SCRS+1
2930       CMP THOUS
2940       BEQ CRET
2950       STA THOUS
2960       INC LIVES
2970       JSR SHOLIV
2980 CRET RTS
2990 ;
3000 ;move the storm
3010 ;
3020 MOVICE LDA ICECNT ;time for
3030       BEQ NEWD      ;new direction?
3040       DEC ICECNT    ;no
3050       LDX ICEDIR    ;get direction
3060       LDA ICETB,X
3070       STA IJMP+1
3080       LDA ICETB+1,X
3090       STA IJMP+2
3100       LDA #1
3110       STA ICEON
3120 IJMP JSR $FFFF   ;move it
```

```
3130      JMP XITVBV
3140 ;
3150 ;get a new direction
3160 ;and set flags
3170 ;
3180 NEWD LDA RANDOM
3190      AND #3
3200      TAX
3210      ASL A
3220      STA ICEDIR
3230      LDA #$6B      ;time on
3240      STA ICECNT    ;screen
3250      LDA IYP,X     ;starting
3260      STA ICY       ;position
3270      LDA IXP,X
3280      STA ICX
3290      CLC
3300      ADC #$0C      ;eyes
3310      STA EYX
3320      JSR PUTICE    ;put definition
3330      JSR ZB3       ;in memory
3340      LDA ICTL      ;time spent
3350      STA OFFSCN    ;off screen
3360      LDA ICTH
3370      STA OFFSCN+1
3380      LDA #0        ;off yet
3390      STA ICEON
3400      JMP XITVBV
3410 ;
3420 ;move table
3430 ;
3440 ICETB .WORD ZRU
3450      .WORD ZLD
3460      .WORD ZRD
3470      .WORD ZLU
3480 ;
3490 ;diagonal movement
3500 ;
3510 ZRU JSR ZIRT
3520      JSR ZIUP
3530      RTS
3540 ZLD JSR ZILF
3550      JSR ZIDN
3560      RTS
3570 ZRD JSR ZIRT
3580      JSR ZIDN
3590      RTS
3600 ZLU JSR ZILF
3610      JSR ZIUP
3620      RTS
3630 ;
3640 ;slide it up
3650 ;
3660 ZIUP LDY ICY
3670      LDX #31
3680 ZIUA LDA P2MEM,Y
3690      STA P2MEM-1,Y
3700      LDA P3MEM,Y
3710      STA P3MEM-1,Y
3720      INY
3730      DEX
3740      BPL ZIUA
3750      DEC ICY
3760      LDA #0
3770      LDY EYY
3780      STA M5MEM+1,Y
3790      LDA #$C3
3800      STA M5MEM-1,Y
3810      DEC EYY
3820      RTS
3830 ;
3840 ;go down
3850 ;
3860 ZIDN LDA ICY
3870      CLC
3880      ADC #31
3890      TAY
3900      LDX #31
3910 ZIDA LDA P2MEM,Y
3920      STA P2MEM+1,Y
3930      LDA P3MEM,Y
3940      STA P3MEM+1,Y
3950      DEY
3960      DEX
3970      BPL ZIDA
3980      INC ICY
3990      LDA #0
4000      LDY EYY
4010      STA M5MEM,Y
4020      LDA #$C3
4030      STA M5MEM+2,Y
4040      INC EYY
4050      RTS
4060 ;
4070 IYP .BYTE $C0,$30,$30,$C0
4080 IXP .BYTE $0A,$E0,$0A,$E0
4090 ;
4100 ;move storm left
4110 ;
4120 ZILF DEC ICX
4130      DEC ICX
4140      DEC EYX
4150      DEC EYX
4160 ZB3 LDA ICX
4170      STA HPOSP0+2
4180      CLC
4190      ADC #16
4200      STA HPOSP0+3
4210      LDA EYX
4220      STA HPOSM0
4230      CLC
4240      ADC #6
4250      STA HPOSM0+3
4260      RTS
4270 ;
4280 ;move stormy right
4290 ;
4300 ZIRT INC ICX
4310      INC ICX
4320      INC EYX
4330      INC EYX
4340      LDA EYX
4350      JMP ZB3
4360 ;
4370 ;put snow storm on screen
4380 ;
4390 PUTICE JSR CLR23
4400      LDX #0
4410      LDY ICY
4420 ICA LDA ICEDAT,X
4430      STA P2MEM,Y
4440      STA P2MEM+1,Y
4450      LDA ICDT2,X
4460      STA P3MEM,Y
4470      STA P3MEM+1,Y
4480      INY
4490      INY
4500      INX
4510      CPX #16
4520      BNE ICA
4530      LDA ICY
4540      CLC
4550      ADC #$0C
4560      STA EYY
4570      LDY EYY
4580      LDA #$C3
4590      STA M5MEM,Y
4600      STA M5MEM+1,Y
4610      RTS
4620 ;
4630 ;stormy's definition
4640 ;
4650 ICEDAT .BYTE $00,$04,$02,$32
4660      .BYTE $0B,$07,$3D,$4F
4670      .BYTE $0C,$3D,$47,$0B
4680      .BYTE $12,$12,$01,$00
4690 ICDT2 .BYTE $00,$80,$48,$48
4700      .BYTE $D0,$E6,$BC,$F0
4710      .BYTE $32,$BC,$E0,$D0
4720      .BYTE $4C,$40,$20,$00
4730 ;
4740 ;erase plyrs 2,3 & missiles
4750 ;
4760 CLR23 LDX #0
4770      TXA
4780 C23 STA P3MEM,X
4790      STA P2MEM,X
4800      STA M5MEM,X
4810      INX
4820      BNE C23
4830      RTS
4840 ;
4850 ;reset game due to
4860 ;lost lives
4870 ;
4880 GRESET DEC LIVES
4890      JSR SHOLIV
4900      JSR RESFUL
```

```
4910      JSR DSPFUL
4920      RTS
4930 ;
4940 ;show # of lives
4950 ;
4960 SHOLIV LDA LIVES
4970      CMP #10
4980      BCC GRT
4990      LDA #9
5000      STA LIVES
5010 GRT ORA #$10
5020      STA TXTWIN+16
5030      RTS
5040 ;
5050 ;let the storm exit stage
5060 ;
5070 REMP23 LDA #0
5080      STA HPOSP0+2
5090      STA HPOSP0+3
5100      STA HPOSM0
5110      STA HPOSM0+3
5120      STA ICECNT
5130      LDA ICTL
5140      STA OFFSCN
5150      LDA ICTH
5160      STA OFFSCN+1
5170      LDA #0
5180      STA ICEON
5190      RTS
5200 ;
5210 ;game over-snow guys win
5220 ;
5230 GAMOVR LDA # >GOVER
5240      STA TMESS+1
5250      LDA # <GOVER
5260      STA TMESS
5270      LDA #0       ;turn off vbi
5280      STA VFLG
5290      JSR SNDOFF
5300 GAM LDA CONSOL  ;chk for START
5310      CMP #6
5320      BNE GAM
5330      JMP NEWBEG
5340 ;
5350 ;initialize sound
5360 ;
5370 SNDOFF LDA #0
5380      STA $D208
5390      LDX #3
5400      STX $D20F
5410      LDX #7
5420 SNL STA $D200,X
5430      DEX
5440      BPL SNL
5450      RTS
5460 ;
5470 ;turn off individ. snds
5480 ;
5490 OFF1 LDA #0
5500      STA $D200
5510      STA $D201
5520      RTS
5530 ;
5540 OFF2 LDA #0
5550      STA $D203
5560      STA $D202
5570      RTS
5580 ;
5590 OFF34 LDA #0
5600      STA $D204
5610      STA $D205
5620      STA $D206
5630      STA $D207
5640      RTS
5650 ;
5660 ;honk horn when dozer hits car
5670 ;
5680 HORN JSR SNDOFF
5690      LDX #1        ;twice!
5700 HN1 LDA #121
5710      STA $D204
5720      LDA #$A6
5730      STA $D205
5740      LDA #10
5750      STA $D206
5760      LDA #$24
5770      STA $D207
5780      JSR LDL
5790      JSR OFF34
5800      JSR LDL
5810      DEX
5820      BPL HN1
5830      RTS
5840 ;
5850 ;pause the game
5860 ;
5870 PAUSE LDX #$FF
5880      STX CH
5890      LDA #0        ;hold on vbi
5900      STA VFLG
5910      JSR WAIT
5920 PAUS LDA CH
5930      CMP #$FF
5940      BEQ PAUS
5950      STX CH
5960      LDA #1
5970      STA VFLG
5980      RTS
5990 ;
6000 ;the vbi
6010 ;
6020 VBLNK LDA VFLG ;running?
6030      BNE VBC     ;yes
6040 VBA JMP XITVBV
6050 VBC LDA OFFSCN ;coming on?
6060      ORA OFFSCN+1
6070      BEQ VBB     ;yes
6080      DEC OFFSCN  ;countdown
6090      LDA OFFSCN
6100      CMP #$FF
6110      BNE VBA
6120      DEC OFFSCN+1
6130      JMP VBA
6140 VBB DEC VTIME   ;vbi speed
6150      LDA VTIME
6160      BNE VBA
6170      LDA ICESPEED
6180      STA VTIME
6190      JMP MOVICE ;do the move
6200 ;
6210 ;make a plowing sound
6220 ;
6230 PLWSND LDA ICEON ;unless
6240      BNE NOPL     ;the storm is
6250      LDA #$32     ;making noise
6260      STA $D202
6270      LDA #$46
6280      STA $D203
6290      JSR WAIT
6300 NOPL RTS
6310 ;
6320 ;reset some playing stuff
6330 ;
6340 REPLAY LDA #$80 ;dozer's
6350      STA PXP      ;position
6360      LDA #$70
6370      STA PYP
6380      LDA #4       ;scroll shadows
6390      STA SMY
6400      LDA #24
6410      STA SMX
6420      LDA #0       ;screen pos.
6430      STA SXP
6440      STA SYP
6450      LDX #5       ;fuel timers
6460      LDA #0
6470 FT1 STA FTIME,X
6480      DEX
6490      BPL FT1
6500      LDX #3       ;car timers
6510 CT1 LDA CARSHD,X
6520      STA CARTIM,X
6530      LDA #0
6540      STA RMTIM,X
6550      DEX
6560      BPL CT1
6570      LDA #0       ;reset road
6580      STA ROADG    ;counters
6590      STA ROADG+1
6600      INC ROADG
6610      STA ICECNT
6620      LDA ICESPEED ;storm'S speed
6630      STA VTIME
6640      JSR SNDOFF  ;sound off
6650      LDA ICTL
6660      STA OFFSCN
6670      LDA ICTH
6680      STA OFFSCN+1
```

```
6690      RTS
6700 ;
6710 ;interval between cars
6720 ;
6730 CARSHD .BYTE $30,$60,$80,$B0
6740 ;
6750 ;speed up for next level
6760 ;fuel down by 10
6770 ;storm is off screen less
6780 ;storm moves faster
6790 ;
6800 FIGLEV LDX LEVEL
6810      CPX #5
6820      BEQ LEA
6830      INC LEVEL
6840 LEA LDX LEVEL
6850      LDA FULM,X
6860      STA FULK
6870      LDA ICOFFL,X
6880      STA ICTL
6890      LDA ICOFFH,X
6900      STA ICTH
6910      LDA ICESPDT,X
6920      STA ICESPEED
6930      LDA #1
6940      STA EDGFLG
6950      JSR SNDOFF
6960      RTS
6970 ;
6980 ;tables for stormy
6990 ;
7000 FULM .BYTE 8,7,6,5,4,3
7010 ICOFFL .BYTE $80,$80,0,$80,0,$80
7020 ICOFFH .BYTE 3,2,2,1,1,0
7030 ICESPDT .BYTE 10,8,6,4,2,1
7040 SPTAB .BYTE 0,2,1,3
7050 ;
7060 ;spin dozer if hit by storm
7070 ;or out of fuel
7080 ;
7090 SPIN LDA DIRECT
7100      PHA
7110      LDY #4
7120      STY YSP
7130      LDA #0
7140      STA SPSND
7150 SPB LDX #3
7160      STX XSP
7170 SPA LDA SPTAB,X
7180      TAX
7190      JSR DFOK
7200      LDA #$86
7210      STA $D203
7220      LDA SPSND
7230      CLC
7240      ADC #9
7250      STA SPSND
7260      STA $D202
7270      JSR DELAY
7280      DEC XSP
7290      LDX XSP
7300      BNE SPA
7310      DEC YSP
7320      LDY YSP
7330      BNE SPB
7340      PLA
7350      STA DIRECT
7360      JSR DFOK
7370      JSR OFF2
7380      RTS
7390 ;
7400 ;move the tracks on the dozer
7410 ;
7420 TRACKR STY YH
7430      STX XH
7440      LDA XD1
7450      AND #1
7460      TAX
7470      LDA TRKR,X
7480 TRKJMP LDY PYP+1
7490      STA P1MEM+1,Y
7500      STA P1MEM+2,Y
7510      STA P1MEM+13,Y
7520      STA P1MEM+14,Y
7530      INC XD1
7540      LDY YH
7550      LDX XH
7560      RTS
7570 ;
```

```
7580 TRKR .BYTE $55,$A9 ;masks
7590 ;
7600 TRACKL STY YH
7610      STX XH
7620      LDA XD1
7630      AND #1
7640      TAX
7650      LDA TRKL,X
7660      JMP TRKJMP
7670 ;
7680 TRKL .BYTE $AA,$95 ;masks
7690 ;
7700 TRACKU STY YH
7710      STX XH
7720      LDY PYP+1
7730      LDX #7
7740 TRU1 LDA P1MEM+4,Y
7750      CMP #$99
7760      BNE TRU2
7770      LDA #$18
7780      BNE TRU3
7790 TRU2 CMP #$18
7800      BNE TRU3
7810      LDA #$99
7820 TRU3 STA P1MEM+4,Y
7830      INY
7840      DEX
7850      BPL TRU1
7860      LDX XH
7870      LDY YH
7880      RTS
7890 ;
7900 ;filling up with fuel sound
7910 ;
7920 FILSND JSR OFF2
7930      LDA #$A4
7940      STA $D203
7950      LDX #160
7960 FIL1 STX $D202
7970      JSR FDEL
7980      CPX #45
7990      BNE FIL2
8000      LDA #$A2
8010      STA $D203
8020 FIL2 DEX
8030      BNE FIL1
8040      JSR OFF2
8050      RTS
8060 ;
8070 FDEL LDY #200
8080 FD1 JSR WAIT
8090      DEY
8100      BPL FD1
8110      RTS
8120 ;
8130 ;get the directory
8140 ;search for SMAP.???
8150 ;
8160 GETDIR JSR CLOSE6
8170      LDX #$60
8180      LDA #3
8190      STA ICCOM,X
8200      LDA # >DIRNAM
8210      STA ICBAH,X
8220      LDA # <DIRNAM
8230      STA ICBAL,X
8240      LDA #6
8250      STA AUX1,X
8260      LDA #0
8270      STA AUX2,X
8280      JSR CIOV
8290      RTS
8300 ;
8310 CLOSE6 LDX #$60
8320      LDA #$0C
8330      STA ICCOM,X
8340      JSR CIOV
8350      RTS
8360 ;
8370 ;read ina map from disk
8380 ;
8390 GETFIL LDA #5
8400      LDX #$60
8410      STA ICCOM,X
8420      LDA # >DBUF
8430      STA ICBAH,X
8440      LDA # <DBUF
8450      STA ICBAL,X
8460      LDA #20
```

```
8470      STA ICBLL,X
8480      LDA #0
8490      STA ICBLH,X
8500      JSR CIOV
8510      BMI DOV
8520      LDA DBUF+4
8530      CMP #'F
8540      BNE FLP
8550 DOV  JSR GETDIR
8560      LDA #0
8570      STA MAPFLG
8580      PLA
8590      PLA
8600      JMP MAP2
8610 ;
8620 FLP  LDX #10
8630 DLP  LDA DBUF,X
8640      STA MAPNAM-2,X
8650      CMP #$20
8660      BEQ DRET
8670      INX
8680      BNE DLP
8690 DRET LDA #EOL
8700      STA MAPNAM-2,X
8710      RTS
8720 ;
8730 DIRNAM .BYTE "D1:SMAP.*",EOL
8740 ;
8750 ;try jumping to edge
8760 ;once per level only
8770 ;to road
8780 ;
8790 JMPEDG LDA EDGFLG
8800      BNE JEA
8810      RTS
8820 JEA  LDA DIRECT
8830      CMP #7
8840      BNE JLF
8850      LDA SMX
8860      STA EHOLD
8870      LDA #$C8
8880      SEC
8890      SBC PXP
8900      LSR A
8910      LSR A
8920      STA EADD
8930      CLC
8940      ADC SMX
8950      STA SMX
8960      DEC SMX
8970      DEC SMX
8980      JSR LOOKR
8990      BCC JYES
9000      LDA EHOLD
9010      STA SMX
9020      RTS
9030 JYES LDX #$C8
9040      STX HPOSP0
9050      STX PXP
9060      INX
9070      STX HPOSP0+1
9080      STX PXP+1
9090      INC SMX
9100      INC SMX
9110      DEC EDGFLG
9120      RTS
9130 ;
9140 JLF  CMP #11
9150      BNE JUP
9160      LDA SMX
9170      STA EHOLD
9180      LDA PXP
9190      SEC
9200      SBC #$30
9210      LSR A
9220      LSR A
9230      STA EADD
9240      LDA SMX
9250      SEC
9260      SBC EADD
9270      STA SMX
9280      INC SMX
9290      INC SMX
9300      JSR LOOKL
9310      BCC JEB
9320      LDA EHOLD
9330      STA SMX
9340      RTS
9350 JEB  LDX #$30
9360      STX HPOSP0
9370      STX PXP
9380      DEX
9390      STX HPOSP0+1
9400      STX PXP+1
9410      DEC SMX
9420      DEC SMX
9430      DEC EDGFLG
9440      RTS
9450 ;
9460 JUP  CMP #14
9470      BNE JDN
9480      LDA SMY
9490      STA EHOLD
9500      LDA PYP
9510      SEC
9520      SBC #$30
9530      LSR A
9540      LSR A
9550      LSR A
9560      LSR A          ;/16
9570      STA EADD
9580      LDA SMY
9590      SEC
9600      SBC EADD
9610      STA SMY
9620      INC SMY
9630      JSR LOOKU
9640      BCC JEC
9650      LDA EHOLD
9660      STA SMY
9670      RTS
9680 JEC  DEC SMY
9690      JSR ERS01
9700      LDA #$30
9710      STA PYP
9720      JSR DEFPLR
9730      DEC EDGFLG
9740      RTS
9750 ;
9760 ;erase players 0 and 1
9770 ;
9780 ERS01 LDY #$FF
9790      LDA #0
9800 ERSA STA P0MEM,Y
9810      STA P1MEM,Y
9820      DEY
9830      BNE ERSA
9840      RTS
9850 ;
9860 JDN  CMP #13
9870      BNE JRET
9880      LDA SMY
9890      STA EHOLD
9900      LDA #$C0
9910      SEC
9920      SBC PYP
9930      LSR A
9940      LSR A
9950      LSR A
9960      LSR A
9970      CLC
9980      ADC SMY
9990      STA SMY
010000    DEC SMY
010010    JSR LOOKD
010020    BCC JED
010030    LDA EHOLD
010040    STA SMY
010050 JRET RTS
010060 JED JSR ERS01
010070    LDA #$C0
010080    STA PYP
010090    JSR DEFPLR
010100    INC SMY
010110    DEC EDGFLG
010120    RTS
010130 ;
010140 ;bonus car bell sound
010150 ;
010160 BELL JSR OFF2
010170    LDA #12
010180    STA $D202
010190 BL3 LDA #$A9
010200    STA SHD
010210 BL1 DEC SHD
010220    LDA SHD
010230    CMP #$A0
```

B&C ComputerVisions

3237 Kifer Road
Santa Clara, CA 95051
(408) 749-1003

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

$8.95 each

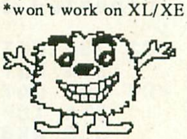
CBS SOFTWARE
(ages 3-6 on disk)
Ernie's Magic Shapes
Astrogrover
Big Bird's Special Delivery
Match Wits

$8.95 each
FISHER PRICE SOFTWARE
(on cartridge)
Memory Manor
Linking Logic
Dance Fantasy
Logic Levels

Perfect for Pre-schoolers

$9.95 each

SPINNAKER SOFTWARE
(all on cartridge)
Alf in the Color Caves (age 3-6)
Alphabet Zoo (age 3-8)
Adventure Creator (age 8-up)
Delta Drawing (age 4-up)
Face Maker (age 3-8)
Kids on Keys (age 3-9)
Kindercomp (age 3-8)
Story Machine (age 3-8)*
*won't work on XL/XE



$8.95 each

TINK! TONK!
(age 4-8 on disk)

Count and Add
(formerly Tinks'a Mazes)
Being a Smart Thinker
(formerly Land of the buddy Bots)
ABC's
(formerly Tink's Adventure)
Subtraction
(formerly Subtraction Fair)
Spelling
(formerly Tuk Goes to Town)
Developing Thinking Skills
(formerly Castle Clobber)
Get all 6 titles for only $39.95

$9.95 each

HAYDEN SOFTWARE
(all on disk)
Monkey See, Spell (age 4-9)
Don't shot that Word (age 6+)

(age 4-10)
Micro Add/subtract
Micro Subtraction
Micro Multiplication
Micro Division

(Pre-School)
Shape Up!
Match Up!

$5.00 each
(on cartridge)

Assembler Rev. A
Basic Rev. C
Donkey Kong
Gorf (won't run on XL/XE)
Missile Command
Pac Man
Star Raiders
Wizard of Wor

$9.95each
(on cartridge)

E.T.
Alien Ambush
Mr. Cool
Cloudburst
Atari Tennis
Touch Typing (disk/cass)

$12.95 each
(on disk)
Atari Paint
Home Filing Manager
Wombats Adventure

14.95 each
(on cartridge)
Popeye
Cobra
Q-Bert
Frogger
Frogger II
Star Wars

$15.95 each
(on cartridge)
Asteroids
Centipede
Football
Sky Writer
Space Invaders
Qix

$19.95 each
(on cartridge)

Archon
Ball Blazer
Battle Zone
Blue Max
David's Midnight Magic
Defender
Donkey Kong Jr.
Fight Night
Final Legacy
Joust
Hardball
Jungle Hunt
Moon Patrol
Ms. Pac Man
One on One (XL/XE only)
Rescue on Fractalas
Robotron:2084
Star Raiders II

$19.95 each
(on cartridge)

Pole Position
Galaxian
Millipede
Pengo
Dig Dug

$24.95 each
(on cartridge)
Gato
Lode Runner
Barnyard Blaster (req. light gun)
Pastfinder

SUPER SPECIALS

RECONDITIONED ATARI MERCHANDISE

All merchandise has been tested and reconditioned and is in like-new condition except where noted by the letter "B" after the price. The "B" price indicates product may have scratches or other superficial surface marks. 30 day warranty. NO free shipping on this merchandise.

ATARI **TRAK BALL** **$9.95** SPICE UP THE ACTION IN YOUR ARCADE GAMES!	ATARI **SPACE AGE JOYSTICK** **$5.00** 	**1020 COLOR PLOTTER/PRINTER** **$29.95** A 40 Columns wide Includes paper and color pen set	**850 INTERFACE** **$89.95** LIMITED SUPPLY	ATARI **BOOKKEEPER** **$14.95 - NO BOX** ($19.95 WITH RECON KEYPAD) **$24.95 - IN BOX** (29.95 WITH RECON KEYPAD)
400 (16K) **COMPUTER** **$29.95** 48K UPGRADE KIT **$25.00** 	**1030 MODEM** WITH **EXPRESS!** **$29.95** GET ONLINE TODAY	**800** (48K) **COMPUTER** **$79.95** INCL. BASIC CART & MANUAL 	**NUMERIC KEYPAD** **$7.95** INCL HANDLER DISK USE WITH BASIC & BOOKKEEPER	**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. 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 complete line of ATARI products and have a large public domain library. Write or call for free catalogue. (408) 749-1003 TUE - FRI 10AM - 6 PM

PRICES SUBJECT TO CHANGE WITHOUT NOTICE - ALL SALES ARE FINAL

CIRCLE #106 ON READER SERVICE CARD.

```
010240    BCC BL2
010250    STA $D203
010260    JSR DELAY
010270    JMP BL1
010280 BL2 JSR OFF2
010290    RTS
010300 ;
010310 ;storm sound
010320 ;
010330 ICESND LDA ICEON
010340    BEQ ICOFF
010350    LDA #$86
010360    STA $D201
010370    LDA #11
010380    STA $D200
010390    RTS
010400 ICOFF JSR OFF1
010410    RTS
010420 ;
010430 ;check if START was pressed
010440 ;
010450 STKEY LDA CONSOL
010460    CMP #6
010470    BEQ STKEY
010480    JMP GAMOVR
010490 ;
010500 ;setup game board and colors
010510 ;
010520 SETSCN LDA # >DL1
010530    STA SDLSTL+1
010540    LDA # <DL1
010550    STA SDLSTL
010560    LDA #$C0
010570    STA $D40E
010580    LDA # >DLI
010590    STA 513
010600    LDA # <DLI
010610    STA 512
010620    LDX #4
010630 KLR LDA CLRTAB,X
010640    STA COLOR0,X
010650    DEX
010660    BPL KLR
010670    RTS
```

LISTING 4: ASSEMBLY

```
0100 ;SAVE#D:SNOW.PT3
0110 ;
0120 ;-----------------
0130 ; screen data file
0140 ; & uncompacter
0150 ; for snowplow
0160 ;
0170 ; by: Barry Kolbe
0180 ;
0190 ;-----------------
0200 ;
0210 ;uncompact screen data
0220 ;
0230 ;set up pointers &
0240 ;end of memory
0250 ;
0260 UNCOM LDA #$90
0270      STA BFL+1
0280      LDA #0
0290      STA BFL
0300      LDA # <MAPDATA
0310      STA TL
0320      LDA # >MAPDATA
0330      STA TL+1
0340      LDA #1
0350      STA EDM
0360      LDA #$9A
0370      STA EDM+1
0380 ;
0390 ;test for unique or
0400 ;repeated data
0410 ;
0420      LDY #0
0430 UC1 STY UNIQUE
0440      STY COUNT+1
0450      JSR GETAB    ;in A
0460      CLC
0470      ROL A
0480      ROL UNIQUE   ;bit 7 test
0490      LSR A
0500      STA COUNT    ;lsb
0510      BNE CKU      ;if 0 long count
0520      JSR GETAB    ;msb
0530      STA COUNT+1
0540      JSR GETAB
0550      STA COUNT    ;lsb of long cnt
0560 CKU LDA UNIQUE
0570      BEQ UC2
0580 UC3 JSR GETAB     ;unique data
0590      STA (BFL),Y
0600      JSR NXBFL
0610      DEC COUNT
0620      BNE UC3
0630      LDA COUNT+1
0640      BEQ UC1
0650      DEC COUNT+1
0660      JMP UC3
0670 ;
0680 ;repeated data
0690 ;
0700 UC2 JSR GETAB
0710      STA DATA
0720 UC4 LDA DATA
0730      STA (BFL),Y
0740      JSR NXBFL
0750      DEC COUNT
0760      BNE UC4
0770      LDA COUNT+1
0780      BEQ UC1      ;next
0790      DEC COUNT+1
0800      JMP UC4
0810 ;
0820 ;increment dest. ptr &
0830 ;check for end of screen
0840 ;memory
0850 ;
0860 NXBFL INC BFL
0870      BNE NIN
0880      INC BFL+1
0890 NIN LDA BFL+1
0900      CMP EDM+1
0910      BNE NRT
0920      LDA BFL
0930      CMP EDM
0940      BNE NRT
0950      PLA          ;done so get out
0960      PLA
0970 NRT RTS
0980 ;
0990 GETAB LDA (TL),Y ;get 1 byte
1000      INC TL       ;&inc. ptr of
1010      BNE GTZ      ;source
1020      INC TL+1
1030 GTZ RTS
1040 ;
1050 ;the actual screen compacted
1060 ;
1070 MAPDATA .BYTE 4,0,30,96,130,108
1080      .BYTE 110,14,96,130,108,116
1090      .BYTE 6,7,130,114,110,28
1100      .BYTE 96,130,27,28,12,96
1110      .BYTE 130,108,110,22,96,130
1120      .BYTE 108,116,4,0,130,14
1130      .BYTE 15,3,7,135,1,2
1140      .BYTE 3,4,7,5,6,2
1150      .BYTE 7,130,5,6,3,7
1160      .BYTE 130,5,6,9,7,130
1170      .BYTE 98,100,2,7,130,5
1180      .BYTE 6,2,7,130,5,6
1190      .BYTE 6,7,130,98,100,6
1200      .BYTE 7,130,98,100,2,7
1210      .BYTE 137,13,7,1,2,3
1220      .BYTE 4,7,5,6,27,7
1230      .BYTE 130,5,6,2,7,130
1240      .BYTE 98,100,5,7,130,14
1250      .BYTE 15,15,7,130,98,100
1260      .BYTE 4,0,2,7,130,114
1270      .BYTE 110,12,96,130,108,116
1280      .BYTE 6,7,130,5,6,4
1290      .BYTE 7,130,98,100,4,7
1300      .BYTE 130,5,6,2,7,130
1310      .BYTE 5,6,4,7,132,98
1320      .BYTE 100,5,6,2,7,132
1330      .BYTE 5,6,98,104,10,96
1340      .BYTE 130,108,116,6,7,135
1350      .BYTE 1,2,3,4,7,5
```

```
1360        .BYTE 6,5,7,130,5,6
1370        .BYTE 8,7,132,5,6,98
1380        .BYTE 100,2,7,130,114,110
1390        .BYTE 14,96,130,108,116,2
1400        .BYTE 7,130,98,100,4,0
1410        .BYTE 135,5,6,98,100,7
1420        .BYTE 5,6,9,7,133,98
1430        .BYTE 100,7,5,6,5,7
1440        .BYTE 130,5,6,2,7,130
1450        .BYTE 98,100,14,7,130,98
1460        .BYTE 100,2,7,130,5,6
1470        .BYTE 2,7,130,98,100,10
1480        .BYTE 7,133,98,100,7,5
1490        .BYTE 6,5,7,4,96,130
1500        .BYTE 108,116,2,7,130,14
1510        .BYTE 15,11,7,131,13,98
1520        .BYTE 100,2,7,130,98,100
1530        .BYTE 14,7,130,98,100,2
1540        .BYTE 7,130,98,100,4,0
1550        .BYTE 2,7,130,98,100,7
1560        .BYTE 7,135,1,2,3,4
1570        .BYTE 7,120,112,12,96,130
1580        .BYTE 106,118,14,7,130,98
1590        .BYTE 104,6,96,130,102,100
1600        .BYTE 8,7,135,1,2,3
1610        .BYTE 4,7,5,6,4,7
1620        .BYTE 130,5,6,3,7,130
1630        .BYTE 120,112,12,96,130,108
1640        .BYTE 110,2,96,130,106,118
1650        .BYTE 2,7,130,98,100,2
1660        .BYTE 7,130,114,110,6,96
1670        .BYTE 130,108,116,2,7,130
1680        .BYTE 98,100,2,7,130,98
1690        .BYTE 100,4,0,2,7,130
1700        .BYTE 98,100,4,7,130,5
1710        .BYTE 6,3,7,130,5,6
1720        .BYTE 5,7,130,5,6,2
1730        .BYTE 7,130,5,6,6,7
1740        .BYTE 130,5,6,2,7,130
1750        .BYTE 5,6,8,7,130,98
1760        .BYTE 100,6,7,130,98,100
1770        .BYTE 2,7,130,5,6,3
1780        .BYTE 7,130,5,6,4,7
1790        .BYTE 133,5,6,7,5,6
1800        .BYTE 16,7,130,8,10,2
```

```
1810        .BYTE 7,130,98,100,2,7
1820        .BYTE 130,5,6,2,7,130
1830        .BYTE 98,100,2,7,138,98
1840        .BYTE 100,13,7,1,2,3
1850        .BYTE 4,98,100,2,7,130
1860        .BYTE 98,100,2,7,130,98
1870        .BYTE 100,4,0,2,7,130
1880        .BYTE 120,112,30,96,130,27
1890        .BYTE 28,4,96,130,108,116
1900        .BYTE 4,7,130,98,100,6
1910        .BYTE 7,130,98,100,4,7
1920        .BYTE 130,114,110,2,96,130
1930        .BYTE 108,116,4,7,130,114
1940        .BYTE 110,2,96,130,108,116
1950        .BYTE 4,7,130,114,110,4
1960        .BYTE 96,130,108,116,6,7
1970        .BYTE 130,98,100,4,7,132
1980        .BYTE 5,6,98,100,2,7
1990        .BYTE 130,120,112,4,96,2
2000        .BYTE 7,130,98,100,2,7
2010        .BYTE 130,98,100,2,7,130
2020        .BYTE 98,100,4,0,13,7
2030        .BYTE 130,5,6,21,7,130
2040        .BYTE 5,6,2,7,130,98
2050        .BYTE 100,4,7,132,98,100
2060        .BYTE 8,10,4,7,130,98
2070        .BYTE 100,4,7,134,98,100
2080        .BYTE 5,6,98,100,4,7
2090        .BYTE 134,98,100,5,6,98
2100        .BYTE 100,4,7,130,98,100
2110        .BYTE 2,7,132,14,15,98
2120        .BYTE 100,6,7,130,98,100
2130        .BYTE 6,7,130,98,100,10
2140        .BYTE 7,130,98,100,2,7
2150        .BYTE 130,98,100,2,7,130
2160        .BYTE 98,100,4,0,130,114
2170        .BYTE 110,28,96,130,108,116
2180        .BYTE 6,7,134,14,15,98
2190        .BYTE 100,5,6,2,7,130
2200        .BYTE 98,100,6,7,130,120
2210        .BYTE 112,4,96,130,106,118
2220        .BYTE 2,7,130,120,112,4
2230        .BYTE 96,130,106,118,2,7
2240        .BYTE 130,120,112,4,96,130
2250        .BYTE 106,118,4,7,130,120
```

CIRCLE #107 ON READER SERVICE CARD.

CIRCLE #108 ON READER SERVICE CARD.

```
2260         .BYTE 112,2,96,130,108,110
2270         .BYTE 2,96,130,106,118,6
2280         .BYTE 7,130,120,112,10,96
2290         .BYTE 130,106,118,2,7,130
2300         .BYTE 98,100,2,7,130,98
2310         .BYTE 100,4,0,139,98,100
2320         .BYTE 7,5,6,5,6,5
2330         .BYTE 6,5,6,7,7,130
2340         .BYTE 5,6,3,7,130,5
2350         .BYTE 6,5,7,130,98,100
2360         .BYTE 4,7,130,114,110,2
2370         .BYTE 96,130,106,118,4,7
2380         .BYTE 130,98,100,6,7,137
2390         .BYTE 13,7,1,2,3,4
2400         .BYTE 7,5,6,27,7,131
2410         .BYTE 98,100,13,25,7,130
2420         .BYTE 98,100,2,7,130,98
2430         .BYTE 100,4,0,130,120,112
2440         .BYTE 10,96,130,8,10,6
2450         .BYTE 7,130,5,6,8,7
2460         .BYTE 130,98,100,4,7,130
2470         .BYTE 98,100,8,7,130,98
2480         .BYTE 100,8,7,130,114,110
2490         .BYTE 18,96,130,108,116,4
2500         .BYTE 7,130,114,110,6,96
2510         .BYTE 130,106,112,4,96,130
2520         .BYTE 27,28,20,96,130,102
2530         .BYTE 100,2,7,130,98,100
2540         .BYTE 4,0,5,7,130,5
2550         .BYTE 6,16,7,130,5,6
2560         .BYTE 5,7,130,98,100,4
2570         .BYTE 7,130,98,100,8,7
2580         .BYTE 132,98,100,5,6,2
2590         .BYTE 7,130,5,6,2,7
2600         .BYTE 130,98,100,6,7,138
2610         .BYTE 5,6,7,1,2,3
2620         .BYTE 4,7,5,6,2,7
2630         .BYTE 130,98,100,4,7,130
2640         .BYTE 98,100,24,7,130,5
2650         .BYTE 6,8,7,130,98,100
2660         .BYTE 2,7,130,98,100,4
2670         .BYTE 0,130,114,110,6,96
2680         .BYTE 130,108,116,4,7,130
2690         .BYTE 114,110,6,96,130,108
2700         .BYTE 116,6,7,130,98,100
2710         .BYTE 4,7,130,120,112,2
2720         .BYTE 96,130,108,116,4,7
2730         .BYTE 130,98,100,2,7,130
2740         .BYTE 5,6,2,7,132,5
2750         .BYTE 6,98,100,2,7,130
2760         .BYTE 5,6,4,7,130,5
2770         .BYTE 6,8,7,132,98,100
2780         .BYTE 8,10,2,7,130,98
2790         .BYTE 104,10,96,130,108,116
2800         .BYTE 2,7,136,13,7,1
2810         .BYTE 2,3,4,1,2,3
2820         .BYTE 7,132,5,6,5,6
2830         .BYTE 5,7,130,98,100,2
2840         .BYTE 7,130,98,100,4,0
2850         .BYTE 130,98,100,6,7,130
2860         .BYTE 98,100,2,7,132,5
2870         .BYTE 6,98,100,6,7,130
2880         .BYTE 98,100,6,7,130,98
2890         .BYTE 100,8,7,130,98,100
2900         .BYTE 4,7,130,98,100,8
2910         .BYTE 7,130,98,100,4,7
2920         .BYTE 130,5,6,12,7,130
2930         .BYTE 98,100,4,7,130,98
2940         .BYTE 100,2,7,130,5,6
2950         .BYTE 6,7,130,98,100,6
2960         .BYTE 7,130,98,100,2,7
2970         .BYTE 130,5,6,10,7,130
2980         .BYTE 98,100,2,7,130,98
2990         .BYTE 100,4,0,130,98,100
3000         .BYTE 2,7,130,5,6,2
3010         .BYTE 7,130,120,112,4,96
3020         .BYTE 132,106,118,5,6,2
3030         .BYTE 7,134,5,6,98,100
3040         .BYTE 5,6,2,7,135,5
3050         .BYTE 6,98,100,7,5,6
3060         .BYTE 5,7,130,98,100,4
3070         .BYTE 7,130,98,100,8,7
3080         .BYTE 130,120,112,8,96,130
3090         .BYTE 108,110,8,96,130,106
3100         .BYTE 118,4,7,130,98,100
3110         .BYTE 5,7,130,5,6,3
3120         .BYTE 7,130,120,112,6,96
3130         .BYTE 130,106,118,5,7,130
3140         .BYTE 5,6,7,7,134,98
3150         .BYTE 100,14,15,98,100,4
3160         .BYTE 0,130,98,100,4,7
3170         .BYTE 130,5,6,10,7,130
3180         .BYTE 5,6,2,7,130,98
3190         .BYTE 100,2,7,130,5,6
3200         .BYTE 2,7,130,120,112,8
3210         .BYTE 96,130,106,112,4,96
3220         .BYTE 130,102,100,4,7,130
3230         .BYTE 5,6,3,7,130,5
3240         .BYTE 6,2,7,130,5,6
3250         .BYTE 3,7,130,98,100,2
3260         .BYTE 7,130,5,6,3,7
3270         .BYTE 130,5,6,5,7,130
3280         .BYTE 98,104,4,96,130,108
3290         .BYTE 116,11,7,130,5,6
3300         .BYTE 14,7,131,13,98,100
3310         .BYTE 2,7,130,98,100,4
3320         .BYTE 0,130,98,100,6,7
3330         .BYTE 130,114,110,6,96,132
3340         .BYTE 1,2,3,4,2,7
3350         .BYTE 130,98,100,9,7,130
3360         .BYTE 5,6,11,7,130,98
3370         .BYTE 100,6,7,130,5,6
3380         .BYTE 10,7,130,98,100,4
3390         .BYTE 7,130,5,6,8,7
3400         .BYTE 130,98,100,2,7,132
3410         .BYTE 3,4,98,100,2,7
3420         .BYTE 130,8,10,24,96,130
3430         .BYTE 106,118,2,7,130,98
3440         .BYTE 100,4,0,130,98,100
3450         .BYTE 3,7,133,5,6,7
3460         .BYTE 98,100,12,7,130,98
3470         .BYTE 100,13,7,130,14,15
3480         .BYTE 7,7,130,98,100,9
3490         .BYTE 7,130,5,6,2,7
3500         .BYTE 130,5,6,3,7,130
3510         .BYTE 98,100,7,7,130,5
3520         .BYTE 6,5,7,130,98,100
3530         .BYTE 5,7,130,14,15,23
3540         .BYTE 7,132,1,2,3,4
3550         .BYTE 4,7,130,98,100,4
3560         .BYTE 0,132,1,2,3,4
3570         .BYTE 4,7,130,120,112,12
3580         .BYTE 96,130,106,112,12,96
3590         .BYTE 130,27,28,8,96,130
3600         .BYTE 106,112,18,96,132,106
3610         .BYTE 118,14,15,2,7,134
3620         .BYTE 1,2,3,4,3,4
3630         .BYTE 4,96,130,106,112,38
3640         .BYTE 96,130,106,118
```

LISTING 5: ASSEMBLY

```
0100 ; SAVE#D:SNOW.PT4
0110 ;
0120 ;-------------------------
0130 ;Intro Screen for SNOWPLOW
0140 ;
0150 ;by: Bryan Schappel
0160 ;
0170 ;-------------------------
0180 ;
0190 CLS .BYTE $92,$0A,$00,$42,$00
0200 WEATHER .SBYTE "national weat"
0210      .SBYTE "her service bulle"
0220      .SBYTE "tin        snow w"
0230      .SBYTE "arning"
0240 ;
0250 STARTI LDY #0    ;turn off vbi
0260      STY VFLG
0270      STY ICEON   ;storm off
0280      JSR SNDOFF
0290      LDX IFLAG   ;falling
0300      BNE INTRO   ;letters?
0310      LDA #8
0320      STA CONSOL
0330      LDA #90     ;scroll len
0340      STA SLENGTH
0350 CP1 LDA SET+$0200,Y ;copy chrset
0360      STA SET2,Y  ;out
0370      LDA #0
0380      STA SET+$0200,Y
0390      STA SCRLMEM,Y
0400      INY
0410      BNE CP1
0420 ;
0430 INTRO LDX #5     ;copy high score
0440 TSLP LDA SCRS,X ;to intro
```

```
0450       STA TOPSCORE+12,X ;screen
0460       DEX
0470       BPL TSLP
0480       LDX #7
0490       LDA #0        ;players off
0500 WWLP STA HPOSP0,X
0510       DEX
0520       BPL WWLP
0530 ;
0540       TAY          ;set up scroll
0550 CWLP STA SCRLMEM,Y ;message
0560       DEY
0570       BNE CWLP
0580       LDY #53
0590 WCP LDA WEATHER,Y
0600       STA SCRLMEM+22,Y
0610       DEY
0620       BPL WCP
0630 ;
0640       LDY #4        ;put in colors
0650 GCL LDA CLS,Y
0660       STA COLOR0,Y
0670       DEY
0680       BPL GCL
0690 ;
0700       LDA #0        ;ptrs for chset
0710       STA IND2      ;move
0720       STA INDR
0730       STA COLOR2
0740       LDA # >SET2
0750       STA INDR+1
0760       LDA # >[SET+$0200]
0770       STA IND2+1
0780       LDA # <IDLST ;intro dlist
0790       STA SDLSTL
0800       LDA # >IDLST
0810       STA SDLSTL+1
0820 ;
0830       LDA IFLAG    ;first time?
0840       BNE SKIPSNOW ;for snow
0850       INC IFLAG    ;letters?
0860       JSR GETRAND
0870       LDA #15
0880       STA IIY3
0890 LP1 LDA #15
0900       STA ICNT
0910 LOOP LDY ICNT
0920       LDA RANDS,Y
0930       TAY
0940       LDA TAB16,Y
0950       STA INDR
0960       STA IND2
0970       LDY IIY3
0980       LDA (INDR),Y
0990       JSR MOVEDN
1000       BCS SKIPSNOW
1010       DEC ICNT
1020       BPL LOOP
1030       DEC IIY3
1040       BPL LP1
1050 ;
1060 SKIPSNOW LDY #0
1070       STY LMS
1080 SK1 LDA SET2,Y
1090       STA SET+$0200,Y
1100       INY
1110       BNE SK1
1120 ;
1130 ;Scroll Weather Message
1140 ;
1150 ISCRL LDX #7
1160 ISC STX HSCROL
1170       LDA #0
1180       STA RTCLOK
1190 WT1 LDA CONSOL
1200       BEQ SKPPER
1210       CMP #6
1220       BEQ GSTART
1230       LDA RTCLOK
1240       BEQ WT1
1250       DEX
1260       BPL ISC
1270       LDY LMS
1280       INY
1290       CPY SLENGTH
1300       BNE ISK
1310       LDY #0
1320 ISK STY LMS
1330       JMP ISCRL
1340 ;
1350 GSTART LDA #0
1360       STA DIRF
1370 GST STA LMS
1380       STA HSCROL
1390       RTS
1400 ;
1410 SKPPER LDA #1
1420       STA DIRF
1430       LDA #0
1440       BEQ GST
1450 ;
1460 ;Move byte down
1470 ;
1480 MOVEDN STY IISY
1490       CMP #0
1500       BEQ MRTS
1510       LDY #0
1520 MLP STA (IND2),Y
1530       CPY IISY
1540       BEQ MRTS
1550       LDX #0
1560       STX RTCLOK
1570 WL    LDX CH
1580       CPX #$FF
1590       BNE BRTS
1600       LDX RTCLOK
1610       BEQ WL
1620       PHA
1630       LDA #0
1640       STA (IND2),Y
1650       PLA
1660       INY
1670       BNE MLP
1680 MRTS LDY IISY
1690       CLC
1700       RTS
1710 BRTS LDX #$FF
1720       STX CH
1730       SEC
1740       RTS
1750 ;
1760 ;Get 16 Random Numbers
1770 ;
1780 GETRAND LDA #1
```

```
1790      STA RAND
1800      LDA RANDOM
1810      AND #$0F
1820      STA RANDS
1830 RLOOP LDA RANDOM
1840      AND #$0F
1850      LDY #0
1860 RSRCH CMP RANDS,Y
1870      BEQ RLOOP
1880      INY
1890      CPY RAND
1900      BNE RSRCH
1910      STA RANDS,Y
1920      INY
1930      STY RAND
1940      CPY #16
1950      BNE RLOOP
1960      RTS
1970 ;
1980 TAB16 .BYTE 0,16,32,48
1990      .BYTE 64,80,96,112
2000      .BYTE 128,144,160,176
2010      .BYTE 192,208,224,240
2020 ;
2030 STAR =  *
2040     *=  $7B00
2050 ;
2060 SNOWMEM .SBYTE "                "
2070      .BYTE "@BDFHJLNPRTVXZ\^"
2080      .SBYTE "                "
2090      .SBYTE "                "
2100      .BYTE "ACEGIKMOQSUWY[]_"
2110      .SBYTE "                "
2120      .SBYTE "   BY: BARRY KOL"
2130      .SBYTE "BE AND BRYAN SCH"
2140      .SBYTE "APPEL           "
2150      .SBYTE "     COPYRIGHT 1"
2160      .SBYTE "988 BBK ENTERPRI"
2170      .SBYTE "SES             "
2180 ;
2190 TOPSCORE .SBYTE "  top score "
2200      .SBYTE "000000  "
2210     *=  STAR
```

LISTING 6: ASSEMBLY

```
0100 ;--------------------------
0110 ;
0120 ;SNOWPLOW Character Set
0130 ;
0140 ;by: Barry Kolbe (graphics)
0150 ;and Bryan Schappel (text)
0160 ;
0170 ;--------------------------
0180 ;
0190      .BYTE $00,$00,$00,$00
0200      .BYTE $00,$00,$00,$00
0210      .BYTE $EC,$E2,$CA,$2A
0220      .BYTE $A2,$A2,$AA,$AA
0230      .BYTE $3F,$8F,$A3,$A8
0240      .BYTE $0A,$0A,$0A,$0A
0250      .BYTE $FF,$FF,$00,$AA
0260      .BYTE $80,$80,$80,$80
0270      .BYTE $FF,$FF,$00,$AA
0280      .BYTE $0A,$0A,$0A,$0A
0290      .BYTE $FD,$F5,$D5,$D5
0300      .BYTE $D5,$F5,$FE,$FE
0310      .BYTE $7F,$5F,$57,$57
0320      .BYTE $57,$5F,$BF,$BF
0330      .BYTE $FF,$FF,$FF,$FF
0340      .BYTE $FF,$FF,$FF,$FF
0350      .BYTE $FA,$EA,$A0,$A2
0360      .BYTE $A0,$A2,$E2,$FA
0370      .BYTE $FA,$EA,$AA,$AA
0380      .BYTE $AA,$AA,$EA,$FA
0390      .BYTE $BF,$AF,$0B,$AB
0400      .BYTE $0B,$AB,$AF,$BF
0410      .BYTE $BF,$AF,$AB,$AB
0420      .BYTE $AB,$AB,$AF,$BF
0430      .BYTE $00,$00,$00,$00
0440      .BYTE $00,$18,$18,$30
0450      .BYTE $FF,$FF,$FF,$FB
0460      .BYTE $EA,$FB,$EA,$FF
0470      .BYTE $FE,$EA,$EF,$EA
0480      .BYTE $FE,$FE,$FE,$FE
0490      .BYTE $BF,$AB,$FB,$AB
0500      .BYTE $BF,$BF,$BF,$BF
0510      .BYTE $7F,$63,$63,$63
0520      .BYTE $63,$63,$7F,$00
0530      .BYTE $38,$18,$18,$18
0540      .BYTE $3C,$3C,$3C,$00
0550      .BYTE $7F,$63,$03,$7F
0560      .BYTE $60,$60,$7F,$00
0570      .BYTE $7E,$06,$06,$7F
0580      .BYTE $07,$07,$7F,$00
0590      .BYTE $70,$70,$70,$77
0600      .BYTE $77,$7F,$07,$00
0610      .BYTE $7F,$60,$60,$7F
0620      .BYTE $07,$07,$7F,$00
0630      .BYTE $7C,$6C,$60,$7F
0640      .BYTE $63,$63,$7F,$00
0650      .BYTE $7F,$03,$03,$1F
0660      .BYTE $18,$18,$18,$00
0670      .BYTE $3E,$36,$36,$7F
0680      .BYTE $77,$77,$7F,$00
0690      .BYTE $7F,$63,$63,$7F
0700      .BYTE $07,$07,$07,$00
0710      .BYTE $00,$00,$18,$18
0720      .BYTE $00,$18,$18,$00
0730      .BYTE $FF,$FF,$FE,$F8
0740      .BYTE $AA,$AA,$CF,$FF
0750      .BYTE $FF,$FF,$AF,$AB
0760      .BYTE $AA,$AA,$F3,$FF
0770      .BYTE $00,$00,$01,$04
0780      .BYTE $55,$55,$30,$00
0790      .BYTE $00,$00,$50,$54
0800      .BYTE $55,$55,$0C,$00
0810      .BYTE $00,$3C,$66,$0C
0820      .BYTE $18,$00,$18,$00
0830      .BYTE $00,$3C,$66,$6E
0840      .BYTE $6E,$60,$3E,$00
0850      .BYTE $00,$00,$3F,$03
0860      .BYTE $7F,$67,$7F,$00
0870      .BYTE $00,$60,$60,$7F
0880      .BYTE $73,$73,$7F,$00
0890      .BYTE $00,$00,$7F,$60
0900      .BYTE $60,$60,$7F,$00
0910      .BYTE $00,$03,$03,$7F
0920      .BYTE $63,$63,$7F,$00
0930      .BYTE $00,$00,$7F,$63
0940      .BYTE $7F,$70,$7F,$00
0950      .BYTE $00,$1E,$18,$7E
0960      .BYTE $18,$38,$38,$00
0970      .BYTE $00,$00,$7F,$63
0980      .BYTE $63,$7F,$07,$7F
0990      .BYTE $00,$60,$60,$7F
1000      .BYTE $73,$73,$73,$00
1010      .BYTE $00,$0C,$00,$0C
1020      .BYTE $0C,$1C,$1C,$00
1030      .BYTE $00,$0C,$00,$0C
1040      .BYTE $0C,$0E,$0E,$7E
1050      .BYTE $00,$30,$30,$76
1060      .BYTE $7C,$76,$73,$00
1070      .BYTE $00,$18,$18,$18
1080      .BYTE $38,$38,$38,$00
1090      .BYTE $00,$00,$66,$7F
1100      .BYTE $7F,$6B,$63,$00
1110      .BYTE $00,$00,$3F,$33
1120      .BYTE $73,$73,$73,$00
1130      .BYTE $00,$00,$3F,$33
1140      .BYTE $73,$73,$7F,$00
1150      .BYTE $00,$00,$3F,$33
1160      .BYTE $73,$7F,$70,$70
1170      .BYTE $00,$00,$7F,$63
1180      .BYTE $63,$7F,$07,$07
1190      .BYTE $00,$00,$3F,$33
1200      .BYTE $70,$70,$70,$00
1210      .BYTE $00,$00,$7F,$60
1220      .BYTE $7F,$07,$7F,$00
1230      .BYTE $00,$0C,$7F,$0C
1240      .BYTE $1C,$1C,$1C,$00
1250      .BYTE $00,$00,$33,$33
1260      .BYTE $73,$73,$7F,$00
1270      .BYTE $00,$00,$63,$63
1280      .BYTE $63,$36,$1C,$00
1290      .BYTE $00,$00,$63,$6B
1300      .BYTE $7F,$3E,$36,$00
1310      .BYTE $00,$00,$66,$3C
1320      .BYTE $18,$3C,$66,$00
1330      .BYTE $00,$00,$33,$33
1340      .BYTE $73,$7F,$03,$0F
1350      .BYTE $00,$00,$7E,$0C
1360      .BYTE $18,$30,$7E,$00
1370      .BYTE $00,$1E,$18,$18
1380      .BYTE $18,$18,$1E,$00
1390      .BYTE $00,$40,$60,$30
1400      .BYTE $18,$0C,$06,$00
```

```
1410        .BYTE $00,$78,$18,$18
1420        .BYTE $18,$18,$78,$00
1430        .BYTE $00,$08,$1C,$36
1440        .BYTE $63,$00,$00,$00
1450        .BYTE $00,$00,$00,$00
1460        .BYTE $00,$00,$FF,$00
1470        .BYTE $00,$00,$07,$0F
1480        .BYTE $1C,$1C,$1C,$1F
1490        .BYTE $0F,$00,$00,$00
1500        .BYTE $18,$1F,$1F,$00
1510        .BYTE $00,$00,$FC,$FC
1520        .BYTE $0C,$00,$00,$FC
1530        .BYTE $FE,$0E,$0E,$0E
1540        .BYTE $1E,$FC,$F8,$00
1550        .BYTE $00,$00,$7C,$7C
1560        .BYTE $1E,$1E,$1F,$1F
1570        .BYTE $1F,$1D,$1C,$1C
1580        .BYTE $1C,$7C,$7C,$00
1590        .BYTE $00,$00,$3E,$3E
1600        .BYTE $38,$38,$38,$38
1610        .BYTE $B8,$F8,$F8,$78
1620        .BYTE $78,$3E,$3E,$00
1630        .BYTE $00,$00,$3F,$7F
1640        .BYTE $70,$70,$70,$70
1650        .BYTE $70,$70,$70,$70
1660        .BYTE $70,$7F,$3F,$00
1670        .BYTE $00,$00,$F8,$FC
1680        .BYTE $1C,$1C,$1C,$1C
1690        .BYTE $1C,$1C,$1C,$1C
1700        .BYTE $1C,$FC,$F8,$00
1710        .BYTE $00,$00,$7C,$7C
1720        .BYTE $1C,$1C,$1C,$1D
1730        .BYTE $1D,$1D,$1D,$1F
1740        .BYTE $1F,$7E,$7C,$00
1750        .BYTE $00,$00,$3E,$3E
1760        .BYTE $38,$38,$38,$B8
1770        .BYTE $B8,$B8,$B8,$F8
1780        .BYTE $F8,$7E,$3E,$00
1790        .BYTE $00,$00,$7F,$7F
1800        .BYTE $1C,$1C,$1C,$1C
1810        .BYTE $1F,$1F,$1C,$1C
1820        .BYTE $1C,$7F,$7F,$00
1830        .BYTE $00,$00,$F0,$F8
1840        .BYTE $1C,$1C,$1C,$1C
1850        .BYTE $F8,$F0,$00,$00
1860        .BYTE $00,$00,$00,$00
1870        .BYTE $00,$00,$7F,$7F
1880        .BYTE $1C,$1C,$1C,$1C
1890        .BYTE $1C,$1C,$1C,$1C
1900        .BYTE $1C,$7F,$7F,$00
1910        .BYTE $00,$00,$00,$00
1920        .BYTE $00,$00,$00,$00
1930        .BYTE $00,$00,$00,$1C
1940        .BYTE $1C,$FC,$FC,$00
1950        .BYTE $00,$00,$3F,$7F
1960        .BYTE $70,$70,$70,$70
1970        .BYTE $70,$70,$70,$70
1980        .BYTE $70,$7F,$3F,$00
1990        .BYTE $00,$00,$F8,$FC
2000        .BYTE $1C,$1C,$1C,$1C
2010        .BYTE $1C,$1C,$1C,$1C
2020        .BYTE $1C,$FC,$F8,$00
2030        .BYTE $00,$00,$7C,$7C
2040        .BYTE $1C,$1C,$1C,$1D
2050        .BYTE $1D,$1D,$1D,$1F
2060        .BYTE $1F,$7E,$7C,$00
2070        .BYTE $00,$00,$3E,$3E
2080        .BYTE $38,$38,$38,$B8
2090        .BYTE $B8,$B8,$B8,$F8
2100        .BYTE $F8,$7E,$3E,$00
2110        .BYTE $FF,$FF,$FF,$D7
2120        .BYTE $FF,$FF,$FF,$FF
2130        .BYTE $00,$00,$00,$14
2140        .BYTE $00,$00,$00,$00
2150        .BYTE $FF,$FD,$FF,$FD
2160        .BYTE $FF,$FD,$FF,$FD
2170        .BYTE $00,$01,$00,$01
2180        .BYTE $00,$01,$00,$01
2190        .BYTE $FF,$7F,$FF,$7F
2200        .BYTE $FF,$7F,$FF,$7F
2210        .BYTE $00,$40,$00,$40
2220        .BYTE $00,$40,$00,$40
2230        .BYTE $FF,$FD,$FF,$5D
2240        .BYTE $FF,$FD,$FF,$FD
2250        .BYTE $00,$01,$00,$51
2260        .BYTE $00,$01,$00,$01
2270        .BYTE $FF,$7F,$FF,$75
2280        .BYTE $FF,$7F,$FF,$7F
2290        .BYTE $00,$40,$00,$45
2300        .BYTE $00,$40,$00,$40
2310        .BYTE $FF,$FD,$FF,$5D
2320        .BYTE $FF,$FF,$FF,$FF
2330        .BYTE $00,$01,$00,$51
2340        .BYTE $00,$00,$00,$00
2350        .BYTE $FF,$FF,$FF,$5D
2360        .BYTE $FF,$FD,$FF,$FD
2370        .BYTE $00,$00,$00,$51
2380        .BYTE $00,$01,$00,$01
2390        .BYTE $FF,$FF,$FF,$75
2400        .BYTE $FF,$7F,$FF,$7F
2410        .BYTE $00,$00,$00,$45
2420        .BYTE $00,$40,$00,$40
2430        .BYTE $FF,$7F,$FF,$75
2440        .BYTE $FF,$FF,$FF,$FF
2450        .BYTE $00,$40,$00,$45
2460        .BYTE $00,$00,$00,$00
2470        .BYTE $FF,$FF,$FF,$FD
2480        .BYTE $FF,$FD,$FF,$FD
2490        .BYTE $00,$00,$00,$01
2500        .BYTE $00,$01,$00,$01
2510        .BYTE $FF,$FF,$FF,$7F
2520        .BYTE $FF,$7F,$FF,$7F
2530        .BYTE $00,$00,$00,$40
2540        .BYTE $00,$40,$00,$40
2550        .BYTE $FF,$7F,$FF,$7F
2560        .BYTE $FF,$FF,$FF,$FF
2570        .BYTE $00,$40,$00,$40
2580        .BYTE $00,$00,$00,$00
2590        .BYTE $FF,$FD,$FF,$FD
2600        .BYTE $FF,$FF,$FF,$FF
2610        .BYTE $00,$01,$00,$01
2620        .BYTE $00,$00,$00,$00
2630        .BYTE $FF,$EF,$FF,$55
2640        .BYTE $FF,$FF,$FF,$FF
2650        .BYTE $00,$08,$00,$55
2660        .BYTE $00,$00,$00,$00
2670        .BYTE $18,$18,$18,$18
2680        .BYTE $18,$18,$18,$18
2690        .BYTE $00,$7E,$78,$7C
2700        .BYTE $6E,$66,$06,$00
2710        .BYTE $08,$18,$38,$78
2720        .BYTE $38,$18,$08,$00
2730        .BYTE $10,$18,$1C,$1E
2740        .BYTE $1C,$18,$10,$00
```

END · USER

Software piracy continues to choke our industry, hurting both software companies and ultimately end users like you and me.

Arthur Leyenberger is a human factors psychologist and freelance writer living in New Jersey. He has written over 100 articles about computers in the last five years and continues to be an Atari enthusiast. When not computing he enjoys playing with robotic toys.

by Arthur Leyenberger

Piracy. It has become a major problem over the years for software manufacturers with every computer. ANALOG magazine has discussed the issue many times before. Readers have responded time and time again. But it's still a problem. Finally, there may be some hope for thwarting the efforts of software pirates.

The hope comes from John Weaver, who has written several programs for the ST. His programs are distributed by Michtron, Inc., the most prolific ST software vendor on the face of the earth. The specifics of the legal case are that a teenager allegedly operated a pirate bulletin board system from which users could download copyrighted programs. One of the programs was *Cards,* Weaver's card playing simulation for the ST.

John Weaver, who owns the copyright of Cards, is suing not just the teenager, but also his parents. Although pirates have been sued by software companies before, this may be the first case in which the pirate's parents have also been sued. According to Jonathan D. Wallace, Esq., the computer lawyer representing Weaver and a partner in the New York City law firm of Meatto, Russo, Burke & Wallace, the case raises a question of first impression under the copyright law. "Our argument is that a parent who supplies the computer equipment and telephone line which is used to operate a pirate bulletin board, and who then tolerates the trading of pirated software, contributes to the copyright infringement," Wallace said. "Since teenagers usually have no assets with which to pay a judgment, holding the parents responsible will give a strong incentive to families not to condone this type of behavior." At the time of this writing, the case is pending in federal court in New York.

As far as I am concerned, more power to Weaver, et. al. I strongly believe that parents have increasingly refused to take responsibility regarding their children for a number of things from sex education to manners to teaching right from wrong. I also believe that software piracy continues to choke our industry, hurting both software companies and ultimately end users like you and me. Hopefully the judge and/or jury will decide that parents *can* indeed be legally responsible for acts of software piracy by their teen-age children. I'll keep you posted on the outcome of this important case.

In discussing this case and the general problem of software theft with Gordon Monnier, president of Michtron, I learned that Michtron has gone after other software thieves as well. Gordon told me that most of the people caught in the act settle out of court or even on the spot. In fact, they caught this guy in Florida in the process of printing a catalog. As a result, they seized his computer equipment and he settled on the spot. In the past year, Michtron has closed down five bulletin board systems (BBS) and one person that was selling illegal copies of their software outright.

Many of these so called pirate BBS also have stolen telephone and bank credit card numbers. Few people realize that the telephone company is constantly looking for this type of illegal activity. Further, the secret service keeps a database of stolen credit card users and illegal BBS operators. That hot shot computer hacker who runs a pirate BBS may be surprised later in life when he applies for a top secret clearance for a programmer position and is denied the classification.

The Ah-haa phenomenon

Ever have the experience where you are trying to figure something out and then the answer suddenly dawns on you. When it happens, you say to yourself, "Ah-haa". You're not alone, this experience happens to everybody. Recently it happened to me.

For years, at least the several years that Jack Tramiel and family have been running Atari, I have wondered why they have not been more active in the U.S. computer market. You probably know the story already—Atari talks a good line about advertising computers, shows commercials at trade shows and then nothing substantial appears in the media. If anything, Atari stresses their game machines in the U.S. market.

At the same time, I keep hearing about Atari in the European computer market. Well folks, ah-haa. It now seems clear to me why Atari is more active overseas. First of all, with the ever-declining value of the U.S. dollar, the dollar is worth more in Europe. As a result, Atari gets more bang for the buck when it spends money promoting its products over there.

Second, Atari is the number one selling computer, at least in France and Germany. Understandably, they do not want to lose their sales lead in those markets. Finally, Atari is still a relatively small company with limited financial resources. They must carefully choose where they spend their advertising and promotion dollars. Whatever they spend in one area means that much less they can spend somewhere else.

Given the resurgence of computer games in the United States and Atari's strength in that market as well as their strength in the European computer market, their marketing policies make sense. However, the ST has not done as well as expected in the US and continued marketing emphasis elsewhere may doom the ST and make it an orphan computer. None of us wants that to happen but Atari will have to do more than make idle promises and rely solely on the hobbyist market is they want the ST to succeed in this country.

Happy anniversary

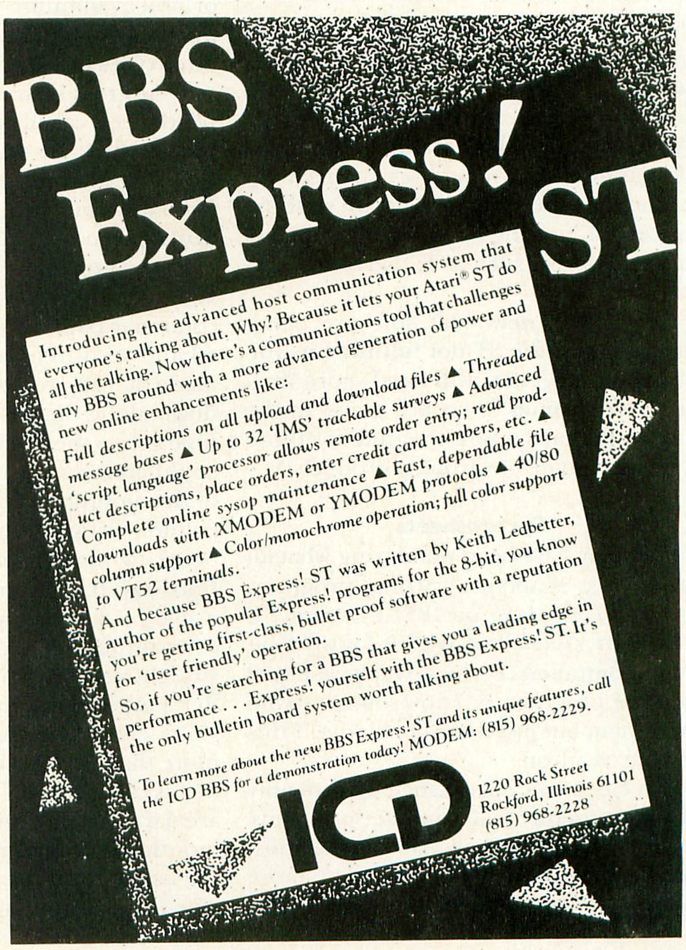
It has been about a year now since Atari acquired the Federated Group, a southern California based retail consumer electronics chain. Atari originally said they would buy the 67-store chain for about 67 million dollars in order to increase distribution of their computers. For over two years, Atari has had difficulty trying to persuade retailers to carry its wares.

Although the Federated Group of stores had been losing money for almost a year prior to the purchase, Atari believed that its financial backing would put the retail chain back in the black and perhaps allow it to begin expanding again. Moreover, Atari was really looking for better distribution. The acquisition was also said to help make Atari a vertically integrated company and give Atari an outlet for new non- computer electronics products that were to be introduced within the year.

It's been a year. Little is heard from the Federated Stores. More importantly, little is heard from Atari *about* the Federated Stores which probably means they are not fulfilling their initial purpose. And what about those new "noncomputer electronics products" that Atari said they would be introducing? I have not heard about nor seen them. Another smoke screen?

What of Atari's distribution? Has it increased? Have those of us concerned about Atari's future been asking this same question for years? According to a recent *New York Times* article, some computer retailers such as Computerland have decided not to carry Atari machines

CIRCLE #111 ON READER SERVICE CARD.



CIRCLE #112 ON READER SERVICE CARD.

END · USER

An electronic spreadsheet is probably the most useful of all programs. Unfortunately, it may also be the most misunderstood.

"partly because Atari has an image as a video game company whose machines would not appeal to corporate customers." Other retailers "are wary of Atari's chairman Jack Tramiel, who in his days as head of Commodore International, undermined his dealers by slashing prices and moving his computers to mass merchandisers such as K mart", the article said.

With little new 8-bit software being released and the ST not fulfilling its initial promises, Atari will need more than just video games to keep it going in the future. Perhaps now is the time for those "noncomputer electronics products."

Spreadsheets

You may have been wondering what all of the fuss is about regarding programs such as *Lotus 1-2-3* on IBM PCs and PC clones or *Visicalc* and *SynCalc* for the 8-bit Atari computers. Perhaps you have heard of these programs or know someone who uses them but just don't know why. If this is true, read on.

An electronic spreadsheet is probably the most useful of all computer programs. Unfortunately, it may also be the most misunderstood. It's a shame that more people don't understand and appreciate the fundamental simplicity of a spreadsheet and therefore the tremendous power that is available in this type of program.

Electronic spreadsheets did not just appear out of thin air. Like many useful categories of computer programs, they are modeled very closely on their manual counterpart. Before computers came along, accountants, bookkeepers, statisticians and even families have used spreadsheets to keep track of everything from depreciation to profit and loss to household budgets. A spreadsheet is nothing more than a two-dimensional set of names and numbers. The key ingredient is rows and columns! If you can understand rows and columns then you understand the underlying principle of a spreadsheet.

Anyone who has ever filled out an income tax form has used a form of a spreadsheet. An income tax form has a vertical list of items such as gross income, number of dependents, deductions, taxable income and tax due. Next to this column is another one for the numbers or dollar amounts. In a spreadsheet, the numerical column may contain one of three types of entries: an input value such as your gross income and number of dependents; a calculation such as taxable income which in this simple example is gross income minus deductions; and a fixed amount such as the amount of tax (for a given taxable income level).

In the precomputer age (not that long ago) spreadsheets were manually done on columnar paper. This paper had horizontal and vertical lines printed on it which made it easy to write the item names down along the left column and units of time across the top line. For example, one could create a home budget that had all of the monthly expenses listed in the first column with all remaining columns labeled by months for one year. Then to find, say the electric bill for July, you would read down the page to find the row containing the electric bill item and straight across to the July column. Remember, rows and columns.

An electronic spreadsheet is nothing more than an old fashioned spreadsheet that is calculated on a computer. Columns are labeled with units of time such as months, quarters, or years and line items are listed down the left side of the form. The major advantage offered by a computerized spreadsheet is automatic recalculation of results. Make one change on an electronic spreadsheet and all calculations that use that value will instantly change. This happens automatically compared to the erase-recalculate-rewrite procedure necessary when using a manual spreadsheet.

A bargain

When I first bought my Atari 800 in 1982, I spent $200 on Visicalc, the very first electronic spreadsheet. Visicalc was originally written for the Apple II computer and in fact was the sole reason that many people bought an Apple back in those days. In June 1983, Synapse (no longer in business) announced the Syn series of software, three separate programs for database, business graphics and spreadsheet applications. There were several other programs in the series as well.

By January 1984, Synapse struck a deal with the old Atari for Atari to distribute *SynFile+*, *SynTrend* and *SynCalc*. Just as these products were being shipped out the door, Jack Tramiel and company bought Atari and promptly canceled the Synapse arrangement. Synapse never got paid by Atari for their effort and material, they had to lower the price in order to move as many as possible and Synapse ultimately went out of business as a result. The entire matter is still waiting to be settled in court.

However, SynCalc was an excellent product. To this day, it has features that even the programs running on the big rigs don't have, such as the use of menus that build the command and display it as you type. That way you learn what the command is and are eventually weaned from the menus. Other features of SynCalc include variable- width columns, sorting data either numerically or alphabetically, compatibility with AtariWriter, a 255 row by 128 column maximum matrix and operation with one or two disk drives.

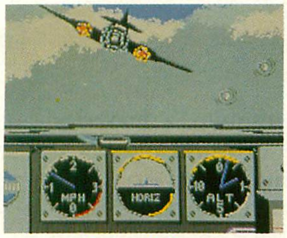
SynCalc was originally scheduled to sell for $99. Within a year after it was released it was selling at a street price of about $35. Recently I saw it for $20 at a megamall toy store. If you have an Atari 8-bit computer and you don't have a spreadsheet for it, SynCalc is the best. If you see it, pick it up. You will be glad you did.

PANAK STRIKES

by Steve Panak

This month we look at the best in fantasy games. This is without a doubt my favorite genre; a logical progression of my affinity for the classic dungeon and dragon games. And as we saw with war-game simulations, we again find that the computerized fantasy games open the D & D genre to a much larger audience, with a mighty microprocessor replacing complex result tables and multisided dice. In the category of fantasy games, there are three lines of evolution which are the best.

The first, and what I consider the best line to follow, is the *Phantasie* series from SSI. This trilogy follows a brave band of adventurers through their trials and tribulations as they tackle the evil dark lord Nickademus. These games were the first fantasy games to tap the vast graphic power and menuing capabilities of the ST. The 8-bit versions, while lacking the sensational graphics, still exhibit the engaging story lines and puzzles that boost Phantasie above all the competition. The only drawback is a lengthy and needlessly involved setup procedure. Unfortunately for 8-bit owners, as of this writing, only the first two installments are available for our machines.

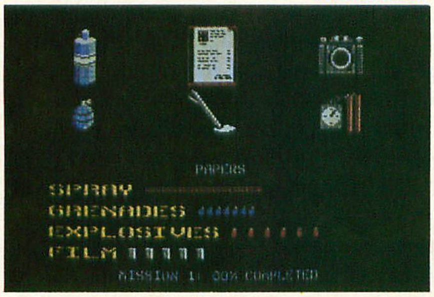


Ace of Aces

For those who want a little more difficulty, Datasoft offers up *Alternate Reality*. The premise is abduction based—you've suddenly been transported away from your comfortable home and into an alternate reality, a medieval world full of adventure and danger. So far there are two such worlds, *The City* and *The Dungeon*. Both play pretty much alike, allowing simple interaction between you and each world's many inhabitants. The main drawback of the game and a contributing factor to its difficulty, is the fact that you cannot save your position before attempting to survive a dangerous situation. Saving a game does just that—saves the game and terminates the program. The save-game disk is then used to restart. This operational aberration calls for an altogether different play strategy.

Using your arsenal of explosives, sleeping gas, fake I.D. papers and a camera and mine detector, you infiltrate the enemy compound.

Infiltrator



Beginners may find refuge in the *Wizard's Crown* series. These simple games are the perfect introduction to computer-based D & D for experienced gamers, due to their moderate level of difficulty. The goal is to search for the wizard's crown, using eight characters which you design. Wizard's Crown and its sequel, *The Eternal Dagger* require 50 hours of play to complete, and it is 50 hours well spent indeed, with the only drawbacks being an overly complex and poorly designed setup procedure and difficult-to-use command structure.

Unfortunately I have no new fantasies to share with you this month. I hope that *Phantasie III* is soon converted to 8-bit, but until that time we'll just have to make do with one of these.

Infiltrator
Mindscape
3444 Dundee Road
Northbrook, IL 60062
64K disk, $29.95

Ace of Aces
Accolade
20813 Stevens Creek Blvd.
Cupertino, CA 95014
64K disk, $29.95

This month we have not one, but two—count them—two flight simulators. Yes,

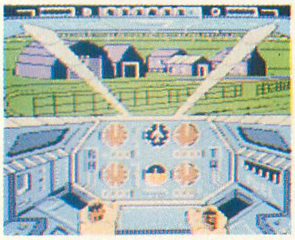
Ace of Aces

PANAK STRIKES

Plundered Hearts

into a market saturated with similar games, Accolade and Mindscape have chosen to launch their own air-based battle and strategy games. And while I tired long ago of flight simulators, I'll try to hang onto my sanity just long enough to tell you if they're air worthy.

In Mindscape's new game, *Infiltrator*, you are Captain Johnny "Jimbo Baby" McGibbits, the Infiltrator. Your mission, should you choose to accept it, is to complete three separate assignments, each comprised of a flight into enemy territory, a ground mission and the flight out. As an Infiltrator, success will hinge on your ability to remain hidden from the enemy, sneaking in and out of hostile territory, fighting only when forced to. Your craft is the Gizmo DHX-1 Attack Chopper, nicknamed "The Snuffmaster," and before you hit the skies you'll have to take a little time out to learn at least some of this aircaft's incredible and diverse capabilities.



Infiltrator

The features found in this chopper read like Rambo's Christmas wish list: air-to-air heat-seeking missiles, a pair of rapid-fire 20mm cannons, a turbo booster to get you out of tight situations, flares and chaff to decoy enemy fire and sophisticated communications, guidance, control and surveillance systems. In the spirit of a true simulation, nearly every system is present and accounted for, and you are required to step through as many procedures to get this chopper off the ground as you would find yourself doing with the real thing.

Start by turning on the battery and initializing the computer system. Pressing I starts your engine. Use the joystick to control movement, while the keyboard arms your various offensive and defensive weapons systems. Pressing the fire button launches your attacks. And if this sounds complicated, wait until you see the cockpit. Once you learn what every dial, read out and warning light in the ultrasophisticated cockpit is trying to tell you, you'll never be in the dark again. In addition to the expected compass, artificial horizon and altimeter and airspeed indicator, warning lights and gauges keep you apprised of critical fuel levels, engine and battery temperatures and engine damage. Sensors detect incoming missiles, while your computer terminal displays craft status and a tactical map. A communications facility allows limited contact with other aircraft, a correct response to messages being required to avoid an attack. And you'll want to avoid a lot of battles if you hope to reach your destination.

Upon arrival, you exit your cockpit and proceed on foot. Using your arsenal of explosives, sleeping gas, fake ID papers and a camera and mine detector, you infiltrate the enemy compound, searching rooms, photographing secret documents. You will be informed when you have completed the mission, so you can hightail it back to your chopper, having saved the world yet again.

As if this were not excitement enough for a worldclass hero, we have yet another flight simulator this month. Into the oversaturated market Accolade launches *Ace of Aces,* a combat flight simulator patterned after the Mosquito, a maverick RAF fighter bomber of World War II. On your way to becoming "Ace of Aces," you work your way through four missions, involving air battles, train bombings, sub sinkings and the eradication of VI rockets before they reach mother England. In each mission, the threat of aerial dogfights with Nazi fighters is always present.

After booting up this game, you might want to start in practice mode to familiarize yourself with the controls. The opening sequence (a series of photographs depicting an aircraft scramble as an air-raid siren blares annoyingly in the background) is best skipped by pressing the joystick button. Lacking the complexity of Infiltrator, your cockpit in Ace contains only the bare essentials—not unexpected, as this is a primitive WWII aircraft. The pilot's view contains dials indicating airspeed and altitude, while the engineer's view allows you set the throttle and flaps and monitor fuel and engine speed.

The navigator's map shows your position relative to enemy installations, and using the bombardier's view you drop bombs and inventory your armaments. You move through these views by pressing a keyboard number or clicking the joystick button twice and manipulating the stick. An on-screen figure of your plane reminds you which stick direction activates each view, and the figure doubles

On your way to becoming "Ace of Aces," you work your way through air battles, train bombings, sub sinkings and the eradication of V1 rockets.

as a trouble indicator. For example, the rear of the plane lights up when the tail-based navigator should be consulted.

Once you gather up enough courage to chance a real mission, you'll be presented with a menu allowing you to designate what types of targets you'll pursue. Next, load your plane with the correct mix of weapons and fuel.

The manual reminds you that bombs are a must for trains and subs, while lots of fuel will be needed for long missions and engaging enemy fighters in dogfights. And once the enemy is vanquished, the mission is over. Unlike Infiltrator, Ace never requires you to leave the cockpit. The complexity in each of these games makes control hard to learn. I found Ace to be more intuitive, thus a little easier to get a grip on. On the other hand, while Infiltrator was a little harder to play, it was also more challenging. However, I did feel that Infiltrator went a little too far in trying to make you feel you were in the pilot's seat. For instance, there's a lot of foolishness as you start each game; the program requiring you to get a number from one console and input it into another to set your guidance system. One would hope a craft as sophisticated as the Snuffmaster would have peripherals that can communicate with one another. The controls on Ace seemed a little more responsive, although both aircrafts were sluggish, lacking the instant response of arcade and ST-based simulators. To make things worse, Ace occasionally accessed the drive before displaying a new screen. Graphics again were very similar, although I'd have to award this skirmish to Infiltrator. Its cockpit was just a little more detailed. And while both featured a vast number (for 8-bit games) of gauges and indicators, all highly detailed, enemy planes were crudely drawn, explosions less than spectacular. To round out the graphics area, a special mention for Ace is in order for its creative views looking out over each wing.

The manual for Infiltrator is thick and quite complete. It sets up the scenario of the game, then goes on to describe the control and design of the Gizmo DHX-1 in great detail. Numerous illustrations help you quickly identify the various systems in the cockpit and the entire manual is written in a lighthearted way. While I found that the constant "cuteness" of the prose became annoying, it was, at least, not boring. A handy reference card is also included. The documentation for Ace of Aces is much less elaborate (and less cute) than Infiltrator's. However, it does contain most of what you'll need to know about the game along with ample diagrams, so you won't be wondering what the various displays will look like. Unfortunately for 800 owners, both games require 64K and will not settle for anything less.

It's kind of hard to pick a winner here; I feel like a voter—powerless and forced to choose the lesser of two evils. In this case, that would have to be Ace of Aces. It was easier to learn and offered most of what Infiltrator did, save the ground mission, which I could have done without anyway. But either of these games will let the prospective pilot take to the skies.

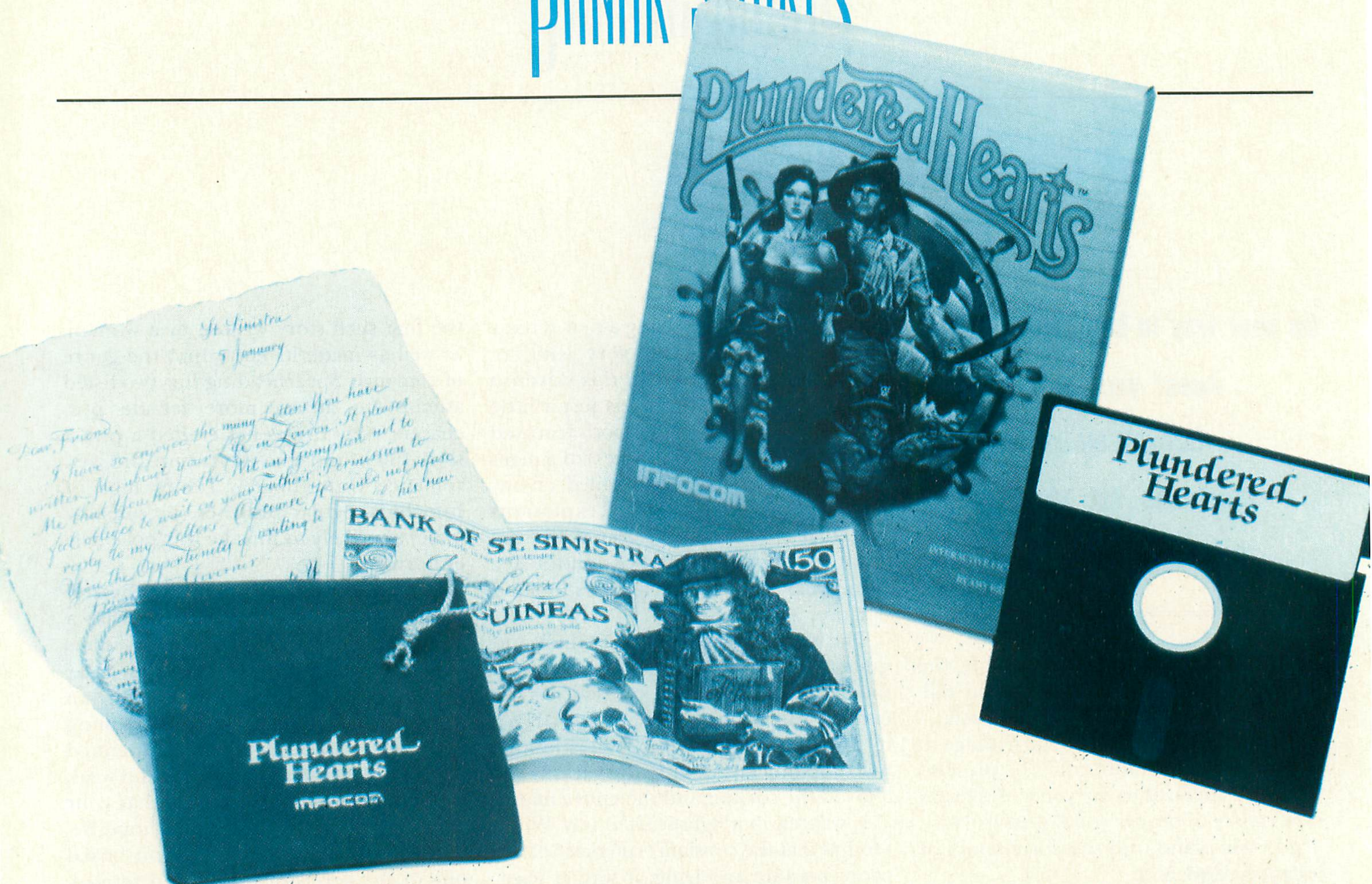
Plundered Hearts
Infocom
125 Cambridge Park Drive
Cambridge, MA 02140
48K disk, $34.95

With the release of their latest work of interactive fiction, Infocom has produced the first such story written by a woman, as well as made its entry into the genre of romance. Spearheading this two-fisted attempt to attract more female purchasers, Amy Briggs has crafted a pirate story full of intrigue, adventure and, yes, romance. Unfortunately, I think most of Infocom's regular audience (presumably male) are likely to forsake this bold new endeavor.

In *Plundered Hearts,* you are a beautiful young woman who has just received a message detailing how a grave illness has befallen your father. He now lies near death on a tropical island, and an unknown friend, Jean Lafond, claiming to the governor of the island, has penned the note, because your father is too weak to even lift a hand. He pleads that your encouragement may be his only hope. Being a loyal, loving daughter, you board one of the governor's ships and set sail.

But two days into the voyage, pirates attack. The captain of your ship, Bartholomew Davis, who might be considered less than heroic, immediately sees in you a way to divert the pirates, saving his own skin. You are locked in his cabin. Moments later, the door breaks in, and a vile piece of humanity grasps you in his arms. He clutches you tightly, fouling your face with a breath reeking of rum. Just moments before he has his way with you, he slumps, having been knocked unconscious by his leader, one Captain Nicholas Jamison, also known as the Falcon. He too has a note from your father,

Spearheading this two-fisted attempt to attract more female purchasers, Amy Briggs has crafted a pirate story full of intrigue, adventure and, yes, romance.



labeling Lafond a traitor, and telling you to trust and accompany the Falcon. Does the fact that you feel so attracted to this fine specimen of a man make it a little too easy to trust him? But then, what choice do you have, as your ship burns, the cowardly Davis lying slain at your feet? You go with the Falcon and begin a journey unlike any you've ever imagined.

Since this game is targeted, presumably, at first-time Infocom customers, I'll take a minute to cover some program specifics. (Regular users will want to jump to the next paragraph.) In interactive fiction, you control the game by issuing commands to the main character in a story. In theory, as in a good book, you project yourself into this character. What distinguishes Infocom stories is the complexity of the program's parser and the power of its prose. The program understands (and occasionally demands) complete sentences, containing subject, verb, direct object and adjectives. Abbreviations speed you through often-used commands, while special commands allow you to save your place in the novel, control verbosity and print a journal. And the simplicity with which you communicate with the program allows you to fully enjoy the rich descriptions and action. Just relax and let the story unfold.

Getting back to Plundered Hearts, included in every package is a velvet reticule (a pouch) containing a 50 guinea note and a letter from Jean Lafond. Also included is the standard Infocom instruction manual, explaining how to play the game, giving helpful tips and highlighting common problems. Finally, a special coupon will allow purchasers to enjoy *Cutthroats,* another Infocom pirate story, at a special price. But still, I have a lot of trouble recommending this game to everyone.

This is because romance is my least favorite literary form. My personal tastes, firmly ingrained by a childhood filled with *Twilight Zone, Star Trek* and bad B movies, run toward horror, fantasy and science fiction. So, it should not come as a surprise that I didn't care much for the story. Newcomers will find the puzzles to be standard Infocom fare, and advanced players will find it relatively easy.

Plundered Hearts should be looked at as just what it is, a romantic work of fiction. One trip to the bookstore will convince you that romance is a viable, popular and profitable literary genre. And with the game's program design and prose up to Infocom's traditional high standards, romance readers might just find it worth their while to turn on to these pages.

That's a wrap for this month. But before I go, some old business. I took a look at *Bridge 5.0* last month, and in the interim a new version has been released. A cursory test has revealed that although the last version's weak bidding has been strengthened, the auto-play mode still needs some work. Whether you want to hold out for yet a better version is dependent on how much you want to play bridge. Next month: the latest simulation from SSI and yet another flight simulator. Until then, good gaming.

In the six years since Steve joined the Atari community, he has spent thousands of hours playing hundreds of games. Between games, he is an attorney and trust officer in a large Ohio bank. Steve is still searching for the perfect game.

REVIEW

Awardware
Hi-Tech Expressions
1700 N.W. 65th Ave.
Suite 9
Plantation, FL 33313
48K disk, $14.95

Reviewed by Matthew J.W. Ratcliff

AwardWare is a super new printer graphic art program from Hi-Tech Expressions, another in their progressive line of products that consistently support the Atari home computers. Design and print your own awards, certificates, ribbons, tickets, coupons, checks and more.

AwardWare is like a specialized version of *Print Shop.* Its functions are similar in many ways. You may choose from 20 different borders, five different fonts, and 25 different graphics in building your awards. What makes AwardWare specialized is that all the page layouts are done for you.

When run, AwardWare presents an impressive graphics introduction display, followed by the main menu. From here you can move on to "Printer Setup," "Create a QuickAward," "Create Awards and More," "Create an AwardDisk" or "Exit." AwardWare exits the program by forcing the system to reboot instead of your having to power cycle the computer: a nice touch. Creating a QuickAward simply allows you to output an award from a layout that has been all or partially completed for you. To create an AwardDisk, you create an award then copy the award disk. You can then send it to a friend with instructions on how to print the last award. I found it odd that the manual would actually instruct you to copy the program and give it to a friend. However, selection of AwardWare's page designs are difficult without documentation. (This is by no means an approval by the manufacturer to make illegal copies of the software, just permission for limited copying of the software for a specific purpose. This policy is quite unusual, and misleading to the uninformed.)

The manual is well written. It begins with a tutorial on creating your first award, explaining all the parts of the display. Generally you will select the option to "Create Awards and More." This will bring up another menu at the bottom of the screen with the options to edit the last template (the last award created is always retrievable from disk), create Awards/Licenses, Letterhead/Memos, Checks/Tickets/Coupons, Ribbons or Miscellaneous.

After a subheading, such as Ribbons, is selected, a smaller menu box pops up showing a number. Press the up and down arrows to change this value, and RETURN to select. You will have to refer to the manual at this point. Every menu selection has its own section in the manual, showing in complete detail all the awards you can choose from, along with the reference number you need to enter.

Next the graphic is shown at the top left of the screen, with the first editable area displayed. At the top right, a template menu is shown, along with a description of the current field type. Pressing RETURN brings up a menu at the bottom left of the display, showing all the graphics, borders, fonts, or whatever is appropriate for this area, that you can choose from.

You can move freely from one field on the graphic to the next, and back again. Changes are quick and easy to make. When finished, select PRINT to make a hardcopy of your award, in either final or draft mode.

While printing, you will see a lot of graphics junk on the display. The program uses the screen RAM as a work area while building the printer graphics output. This is done just to let you see the program working. It's a bit more interesting than a simple "Working, Please Wait" prompt.

Printer setup can be selected from the main menu, a wide variety of printers is supported. If AwardWare doesn't support your printer, a printer-driver construction utility is provided. You will need a good printer reference manual and some understanding of special printer codes. You probably won't have to use the printer-driver editor, however, since 19 different printers are supported, with drivers included for Panasonic, Okidata and Star printers. You will also find some unexpected drivers for the Star NB-24 (24 pin printer), Apple Imagewriter and even the Hewlett Packard QuietJet.

There are 60 different award, license layouts to choose from. Three are five different letterhead and memo designs, each having five border layouts to select from. There are two checks, one ticket, four coupons and four ribbon designs available. Under miscellaneous you will find a scroll, key, trophy, newspaper and more. In all, there are nearly 100 unique basic awards you can create. I have certainly had a lot of fun creating awards for my friends at work, ribbons for my sons and "kiss" coupons for my wife. I haven't come across any apparent bugs in the program at all. AwardWare is an impressive little program for the price. I was pleased to find that AwardWare is not copy-protected, allowing me to make a backup copy, as well as install it on my ICD FA-ST hard drive for faster operation (running under SpartaDOS).

M/L Editor

For use in machine language entry.

by Clayton Walnum

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The two-letter checksum code preceding the line numbers here is *not* a part of the BASIC program. For more information, see the "BASIC Editor II" in issue 47.

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:BACKSP=126:CHKS
   UM=0:EDIT=0
GO 30 GOSUB 450:POSITION 10,6:? "Start or
    Continue? ";:GOSUB 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$(A)="S" THEN 120
FD 90 TRAP 430:OPEN #2,4,0,F1$: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$:GOTO 170
VT 120 TRAP 160:OPEN #2,4,0,F1$:GOSUB 440
   :POSITION 10,10:? "FILE ALREADY EXISTS
   !!":POKE 752,0
ZU 130 POSITION 10,12:? "ERASE IT? ";:GOS
   UB 500:POKE 752,1:? CHR$(A)
VH 140 IF CHR$(A)="N" OR CHR$(A)="n" THEN
    CLOSE #2:GOTO 30
QG 150 IF CHR$(A)<>"Y" AND CHR$(A)<>"y" T
   HEN 130
BH 160 CLOSE #2:OPEN #2,8,0,F1$
IE 170 GOSUB 450:POSITION 10,1:? "NOW ON 
   LINE: ";LINE:CHKSUM=0
GH 180 L1=3:FOR X=1 TO 16:POSITION 13*(X<
   10)+12*(X>9),X+2:POKE 752,0:? "BYTE #"
   ;X;": ";:GOSUB 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:CHKSUM=CHKSUM+BYTE*X:IF
    CHKSUM>9999 THEN CHKSUM=CHKSUM-10000
MS 230 NEXT X:CHKSUM=CHKSUM+LINE:IF CHKSU
   M>9999 THEN CHKSUM=CHKSUM-10000
IG 240 POSITION 12,X+2:POKE 752,0:? "CHEC
   KSUM: ";:L1=4:GOSUB 310
EW 250 IF EDIT AND L=0 THEN 270
QM 260 C=VAL(N$)
SY 270 POSITION 22,X+2:? C;"   "
IL 280 IF C=CHKSUM THEN 300
DI 290 GOSUB 440:EDIT=1:CHKSUM=0:GOTO 180
LW 300 FOR X=1 TO 16:PUT #2,BF(X):NEXT X:
   LINE=LINE+10:EDIT=0:GOTO 170
FV 310 L=0
KZ 320 GOSUB 500:IF (A=ASC("Q") OR A=ASC(
   "q")) AND X=1 AND  NOT EDIT THEN 420
PO 330 IF A<>RETRN AND A<>BACKSP AND (A<4
   8 OR A>57) THEN 320
DX 331 IF A=RETRN AND N$="" 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
   ACKSP) AND L=0 THEN 320
DW 350 IF A=RETRN THEN POKE 752,1:? " ":R
   ETURN
GG 360 IF A<>BACKSP THEN 400
SA 370 IF L>1 THEN N$=N$(1,L-1):GOTO 390
AS 380 N$=""
RE 390 ? CHR$(BACKSP);:L=L-1:GOTO 320
BB 400 L=L+1:IF L>L1 THEN A=RETRN:GOTO 35
   0
WX 410 N$(L)=CHR$(A):? CHR$(A);:GOTO 320
KN 420 GRAPHICS 0:END
YT 430 GOSUB 440:POSITION 10,10:? "NO SUC
   H FILE!":FOR X=1 TO 1000:NEXT X:CLOSE
   #2:GOTO 30
FD 440 POKE 710,48:SOUND 0,100,12,8:FOR X
   =1 TO 50:NEXT X:SOUND 0,0,0,0:RETURN
MY 450 GRAPHICS 23:POKE 16,112:POKE 53774
   ,112:POKE 559,0:POKE 710,4
XR 460 DL=PEEK(560)+256*PEEK(561)+4:POKE
   DL-1,70:POKE DL+2,6
HW 470 FOR X=3 TO 39 STEP 2:POKE DL+X,2:N
   EXT X:FOR X=4 TO 40 STEP 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
WZ 500 OPEN #1,4,0,"K:":GET #1,A:CLOSE #1
   :RETURN
```

REVIEW

DELPHI, The Official Guide
by Michael A. Banks
Brady Books/Simon & Schuster
488 pages, $19.95

Reviewed by Clayton Walnum

There's no arguing the fact that the major online services are complex systems that can be daunting to even the most experienced user. To the new subscriber, however, the numerous commands required to navigate the network can be intimidating to the point of frustration. Although DELPHI is more user-friendly than most systems of its type, there's no avoiding the fact that to provide the greatest "bang for the buck" a certain amount of complexity must exist. DELPHI is an immense and labyrinthine web. After all, there are literally *hundreds* of areas the subscriber may access (everything from an online encyclopedia to more esoteric areas such as AMSEX [American Sexology,] and the Hearing Impaired Forum), and each area has unique features the user must become familiar with.

Brady
DELPHI™
The Official Guide
Your Complete Resource to the World's Premier On-Line Information Service
INSTANT SIGN-UP OFFER INSIDE
Michael A. Banks with the cooperation of General Videotex Corporation

What to do?

If you're a subscriber to DELPHI, I've got great news. There's a new book by Michael A. Banks that absolutely has to be added to your library. *DELPHI, The Official Guide* will not only escort the beginners among you effortlessly through your inaugural DELPHI wanderings, but will also surprise you old masters with myriad tidbits that will make your online excursions even more fruitful than they were before. In fact, this book is so complete that DELPHI abandoned their own manual and took on *DELPHI, The Official Guide* as the guide provided to new subscribers at sign-up time. (Could be why it's called the *official* guide, eh?)

Almost 500 pages in length, the book is loaded with "screen shots" that illustrate exactly what you'll see on-screen during your DELPHI sessions. Command line examples, showing what should be typed at the various prompts, are also included. When you combine the sample screens with the command illustrations, you find that reading the book is almost like being online. You could easily learn the basics of the network without ever touching your computer.

Part 1 of the book, "Getting Started," begins with chapters that describe DELPHI in a general way and explain what is required to access the services. The basics of communicating with DELPHI are then discussed, including the use of the various types of menus and the entry of the control key and immediate (global) commands.

These introductory chapters are followed by a description of DELPHI's main menu. In this section of the book, each of the primary areas is briefly described, preparing you for the more detailed chapters to come.

The real "meat" of the book lies in Part 2, the "DELPHI Members Handbook," where each of the primary areas gets a chapter unto itself. This 340-page section is where you'll spend most of your time, where you'll learn how to send E-mail, how to participate in a CO (conference), how to join and steer your way through SIGs (Special Interest Groups), how to upload and download files, how to manage your work space, how to use DELPHI's editors and so on. In short, everything you need to know is described in careful detail, with plenty of examples to ensure understanding.

Part 3, the "DELPHI User's Guide," offers many tips to help you use your time on DELPHI more efficiently, and Part 4, "Reference," includes a DELPHI index, the DELPHI membership agreement, a troubleshooting section and a list of access numbers for Telenet, Tymnet and DATAPAC. Finally wrapping up the book is a lengthy glossary and an equally lengthy index. An extra bonus is the quick reference card bound into the back of the book. After removal (no sweat; it has a perforated edge), it will reside right next to your keyboard, where you can Grab it the next time you need a quick answer to a question regarding DELPHI.

The book's author, Michael Banks, is not your average computer-hacker-become-writer. He is a seasoned professional who has to his credit many non-fiction books and science-fiction novels. He also has monthly telecommunications columns in several magazines (not the least of which is his "Database Delphi" column in ANALOG Computing) and has published articles and short stories almost beyond counting. To further substantiate his credentials, I should mention that he is the primary manager of DELPHI's Science Fiction and Fantasy SIG.

If you're already a subscriber to DELPHI, you may order a copy of *DELPHI, The Official Guide* right online. The book is also available in bookstores throughout the country or by direct order from the publisher. New subscribers to DELPHI will receive the book as part of their sign-up package, a bargain that's hard to ignore.

DELPHI, The Official Guide is a complete, carefully organized and well-written book that provides much more information than one has a right to expect for a measly $19.95. (Equivalent computer-related handbooks may run as high as $35.) An immense amount of labor went into its creation, and it is you and I who gain the fruits of that labor—all the fruits except the royalties.

Game D
Wo

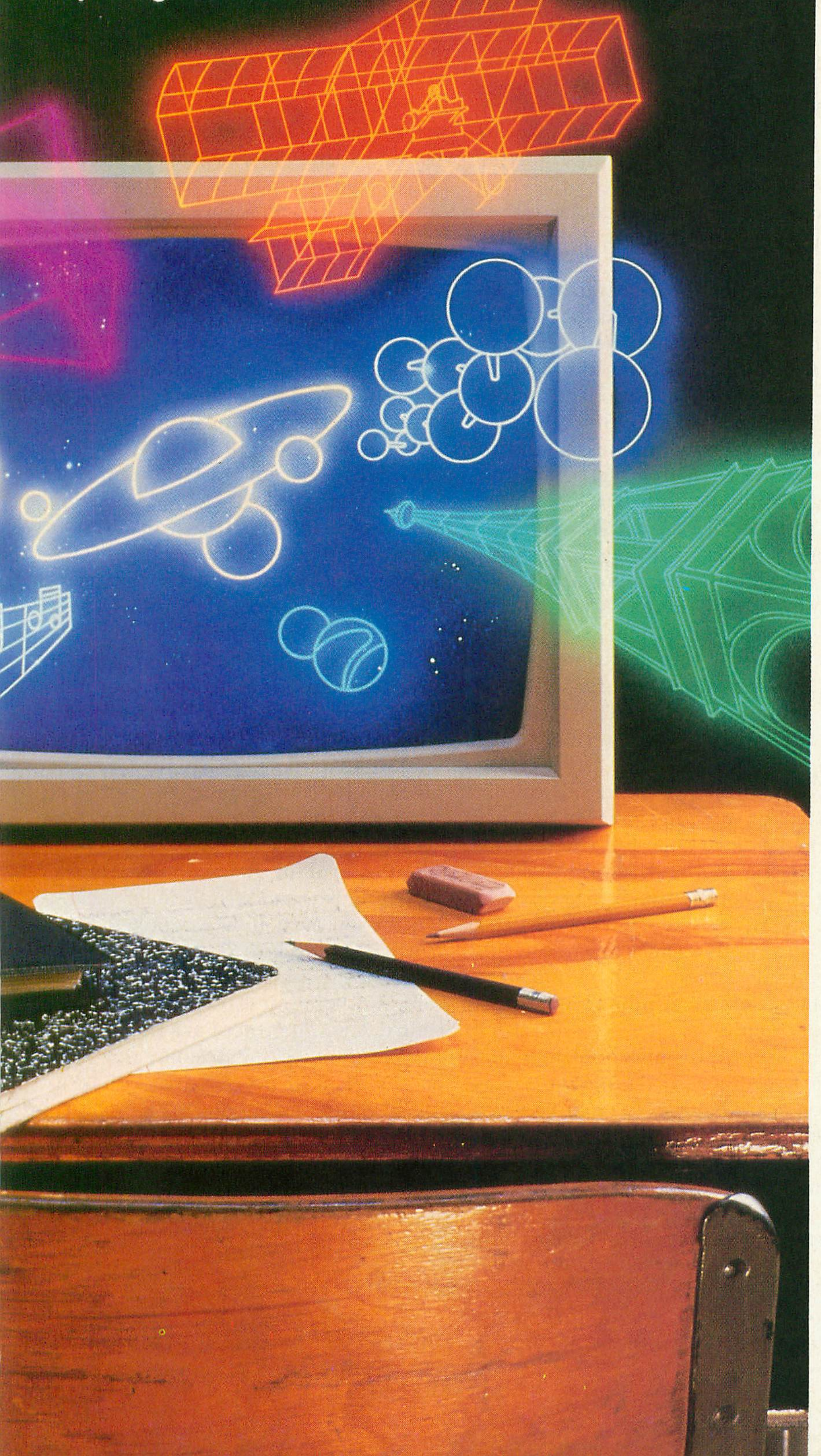
With small programs, it's very easy to just jump right into things and start programming. Unfortunately, it's not so easy once they get larger; there are so many things to take care of, that you can get totally lost and confused very quickly. The solution is to take an intermediate step between your mind and the program; something that makes sense to you and is easily converted into a program. This step is called a flowchart.

Despite the value of using a flowchart, very few people actually use one, especially in the world of microcomputers. And, if you promise not to tell anyone, I'll let you in on a little secret. Up until this column, I had never used a flowchart either! And I've been programming for seven years now, including some very complicated video games. So I'm not going to come at you and say, "Well, you should use flowcharts because that's what I was taught to do, and it's worked for me." Instead, I'll explain the advantages and disadvantages that I ran across in using my very first flowchart.

First of all, let's take a look at a simple example of a flowchart. This is for the part of a game that updates the score. In

esign rkshop

by Craig Patchett



this particular game, which is actually the BASIC Invaders game that we'll be developing together, a bonus base is given at 10,000 points. The sample flowchart is shown in Figure 1.

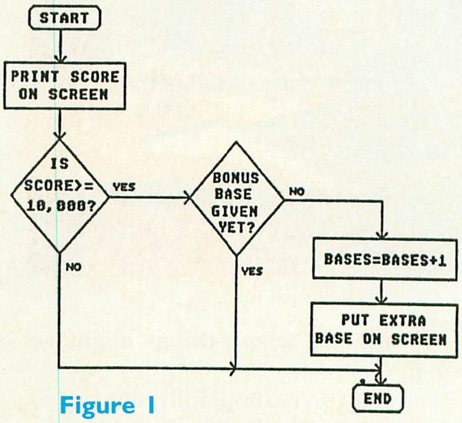
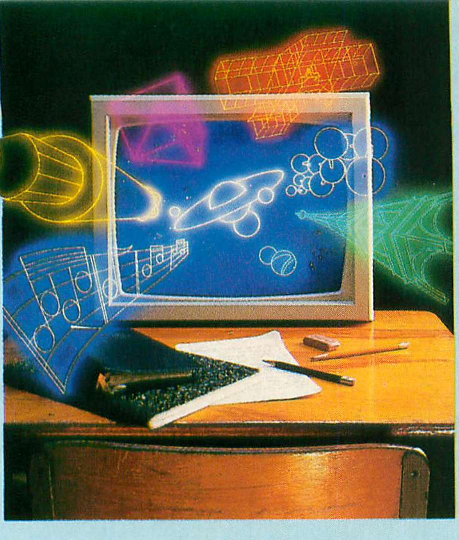


Figure 1

So what does this all mean? First of all, notice how easy it is to understand what's going on. That's because, apart from the funny squares and diamonds, everything is written in English, not BASIC. If you were to take a look at the BASIC program that accomplished the same thing, chances are it would be much more difficult to understand what it was doing. At the same time, it's now quite easy to take this flowchart and make it into a program; a lot easier than taking just the original idea.

What is it exactly that a flowchart does to make programming easier? When you go to write a program, you start off with an idea of what you want the program to do. Unfortunately, computers don't run on ideas. You have to be able to break this idea down into a series of very concrete steps, and then write these steps in a language that the computer will understand. Most people (including myself), try to go straight from the idea to the computer, taking care of the intermediate steps in their head. As I said before, this works fine if the resulting program is very small, but most people don't have the capacity to keep a lot of precise steps organized in their head for a larger program. The result is a program that takes a long time to write, and even longer time to debug, and ends up looking like a mess. (Be honest, when was the last time you wrote a program that looked as neat and orderly as the ones in the magazines?) Believe me, I know from experience! Anyway, the point I'm taking too long to make is that a flowchart organizes your thoughts for you. It breaks down your terrific idea into a series of concrete steps that can then be easily translated into a program. It also often has the added benefit of letting you



see in advance where things might go wrong.

Now that you're (hopefully) convinced about the benefits of a flowchart, let's take a look at how to create one. Of course, there is nothing to say that you have to follow these rules. Whatever works best for you is fine, but the following guidelines are a good place to start.

As you saw in our example, flowcharts are made up of a whole bunch of shapes connected by arrows. Inside these shapes are descriptions of each step. Why are there different kinds of shapes? Because there are different kinds of steps. Figure 2 is a summary of the shapes and the kinds of steps they represent.

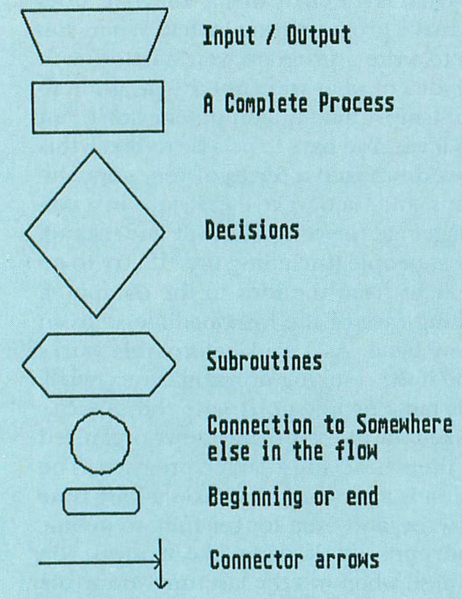


Figure 2

You'll see these shapes or symbols throughout future columns, and at the end of this column when we present the complete flowchart for BASIC Invaders. Actually, that's not quite true. You won't see the Input/Output symbol, largely because there is no I/O in the program. How should you use this symbol? In whatever way makes sense to you. As I said before, a flowchart is meant to make things easier for you, so you should use it in whichever way you're most comfortable with.

Now that you know what a flowchart is and how to make one, you're probably wondering whether or not it's worth the effort to use one. After all, it does take time to do a flowchart, and that time could otherwise be spent programming. Well, we've already seen most of the advantages of flowcharts. They break down a program into small steps that can then be easily programmed; which means that it takes less time to do the programming, which makes up for the time it takes to do the flowchart. Another advantage is that it's often easy to look over a flowchart and see where problems might arise, thus helping you to get rid of bugs before they occur.

But what about the disadvantages? After all, I've already told you that I lived without flowcharts for seven years; so there must be some disadvantages to them, right? The big disadvantage to flowcharts is the fact that they aren't the easiest things to create, and especially to change. If you're writing a program and you make a mistake or forget something, then it's easy to take out a line or add one. But with a flowchart, things start to get messy. Take it from me, never do a flowchart in pen! And, there's no way around it, flowcharts do take time. Even though they'll eventually save you time on the programming, that's no consolation while you're spending hours with a piece of paper and not getting any results on the screen.

So what's the verdict? Should you use flowcharts, or shouldn't you? My advice is to try them at least once and see what you think. Maybe they'll work for you and maybe they won't. (Just think, you paid money for advice like this!) Personally, I plan on using flowcharts again, but not for everything. I'm one of those impatient souls that needs to see immediate results on the screen.

So much for our philosophizing, now it's time to get into a real program. As I'm sure you know by now, we're going to be developing a BASIC version of the popular Invaders-type program. Appropriately enough, we're going to be calling it BASIC Invaders. In any case, we'll start off by presenting the complete flowchart for the game. As we go through the game piece by piece, it will help you to look at the flowchart and see how the BASIC code relates to it. So, without any further ado, Figure 3 is my flowchart for the BASIC Invaders game.

Look it over carefully and then keep it in mind as we write the program. Although we won't be referring to it anymore, it will be used implicitly as we put things together.

Another introduction

Don't worry, this will be relatively short and painless. It is an introduction to the program examples that you will be coming across throughout the rest of the text. These examples serve two purposes. First of all, they are examples of the techniques that we will be covering. In this sense, I will do as much as possible to see that it is obvious how similar sections would be written for games other than the one we will be writing here. Second of all, they are, of course, a part of the final game, our BASIC Invaders. Thus they will eventually all fit together to create the game. Because of this, our line numbers are going to be a little off the wall. This is to save you time, since you will eventually be able to merge all the segments together to make a complete program. Thus the line numbers in the various segments are those from the final program.

And now, our first program. What! How can this be? Well, I mentioned before that I'll be giving you a lot of machine-language routines. If you've seen machine-language routines before, you know that they are made up of either a lot of numbers or a lot of funny characters, depending on which technique the author uses. And if you've tried typing in any of these routines, you know it can be a real pain. This column's routines have a grand total of 997 such funny characters. So what do we do? My answer to this problem is the program shown in Listing 1. As you can see, it has numbers, not characters, to make life easier for you. But when you run it, the computer will take these numbers and turn them into characters for you. Neat, huh? Not only that, but it will also check to make sure that you typed in the numbers correctly, and will tell you where you made a mistake if you didn't. Assuming there are no mistakes, the program will create some new lines, because they are the ones that we'll be using in our game. To do this and get rid of the other lines, use one of the following:

```
LIST "D:MACHINE", 29000, 32510
LIST "C:", 29000, 32510
```

Of course, which one you use depends on whether you have a disk or cassette.

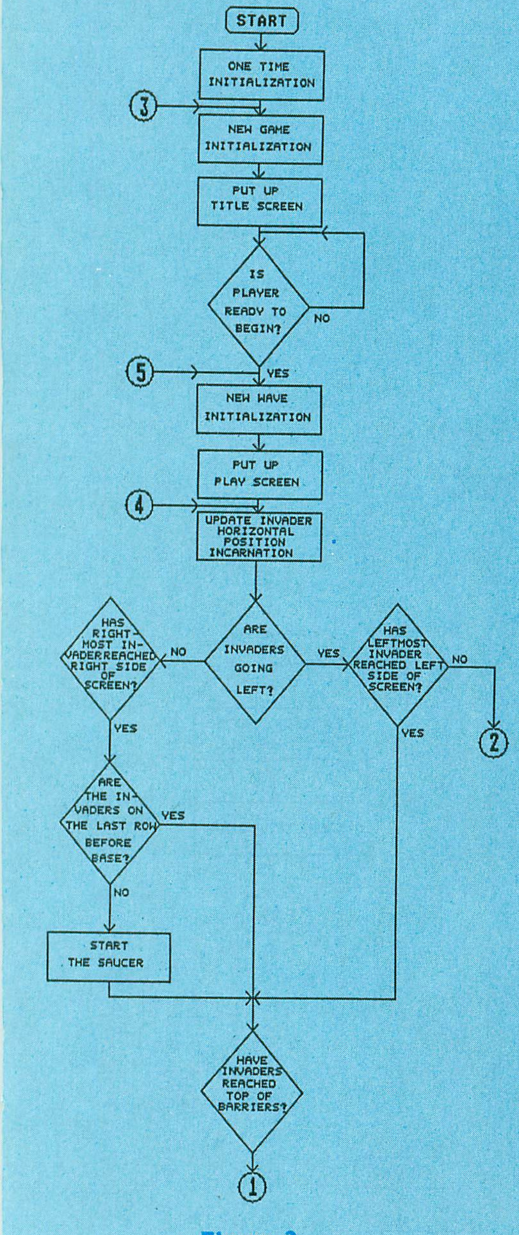

START
ONE TIME INITIALIZATION
3
NEW GAME INITIALIZATION
PUT UP TITLE SCREEN
IS PLAYER READY TO BEGIN?
NO
YES
5
NEW WAVE INITIALIZATION
PUT UP PLAY SCREEN
4
UPDATE INVADER HORIZONTAL POSITION INCARNATION
ARE INVADERS GOING LEFT?
NO
YES
HAS RIGHT-MOST IN-VADER REACHED RIGHT SIDE OF SCREEN?
HAS LEFTMOST INVADER REACHED LEFT SIDE OF SCREEN?
NO
2
YES
YES
ARE THE IN-VADERS ON THE LAST ROW BEFORE BASE?
YES
NO
START THE SAUCER
HAVE INVADERS REACHED TOP OF BARRIERS?
1

Figure 3

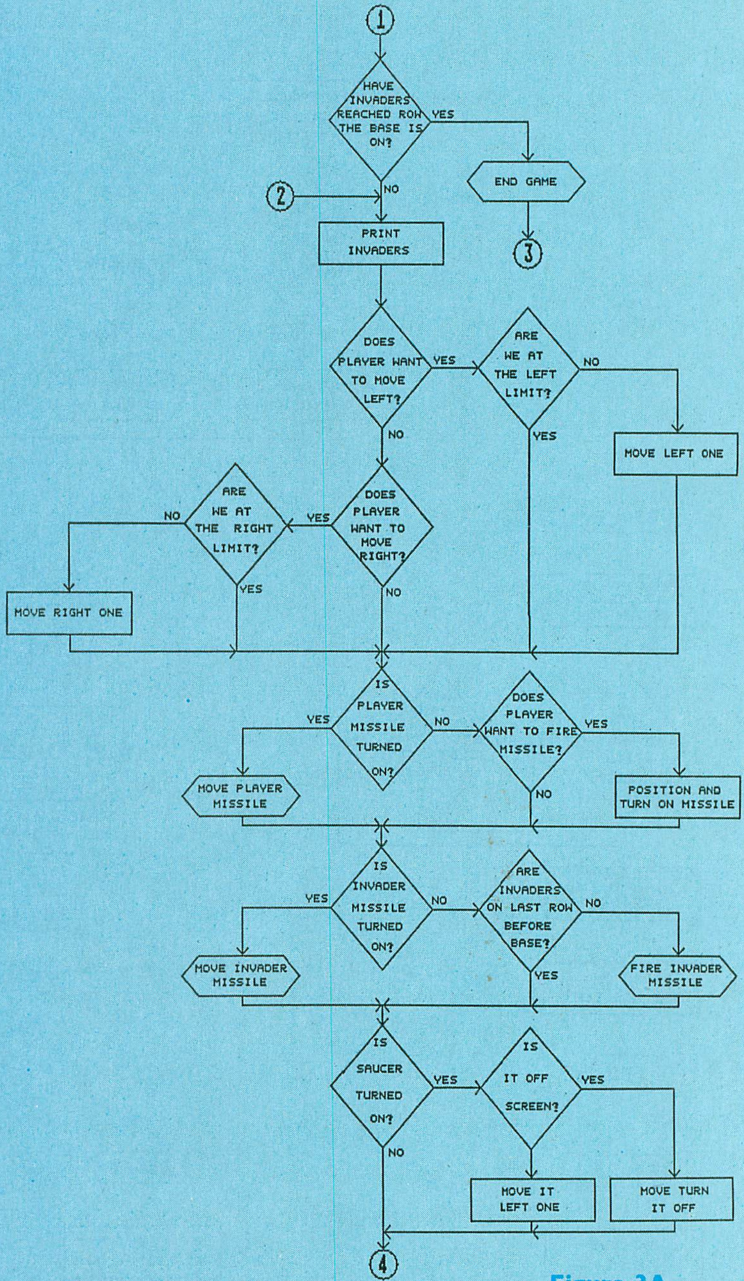

1
HAVE INVADERS REACHED ROW THE BASE IS ON?
YES
END GAME
3
NO
2
PRINT INVADERS
DOES PLAYER WANT TO MOVE LEFT?
YES
ARE WE AT THE LEFT LIMIT?
NO
YES
MOVE LEFT ONE
NO
DOES PLAYER WANT TO MOVE RIGHT?
YES
ARE WE AT THE RIGHT LIMIT?
NO
MOVE RIGHT ONE
YES
NO
IS PLAYER MISSILE TURNED ON?
YES
NO
DOES PLAYER WANT TO FIRE MISSILE?
YES
NO
MOVE PLAYER MISSILE
POSITION AND TURN ON MISSILE
IS INVADER MISSILE TURNED ON?
YES
NO
ARE INVADERS ON LAST ROW BEFORE BASE?
NO
YES
MOVE INVADER MISSILE
FIRE INVADER MISSILE
IS SAUCER TURNED ON?
YES
IS IT OFF SCREEN?
YES
NO
MOVE IT LEFT ONE
MOVE TURN IT OFF
4

Figure 3A

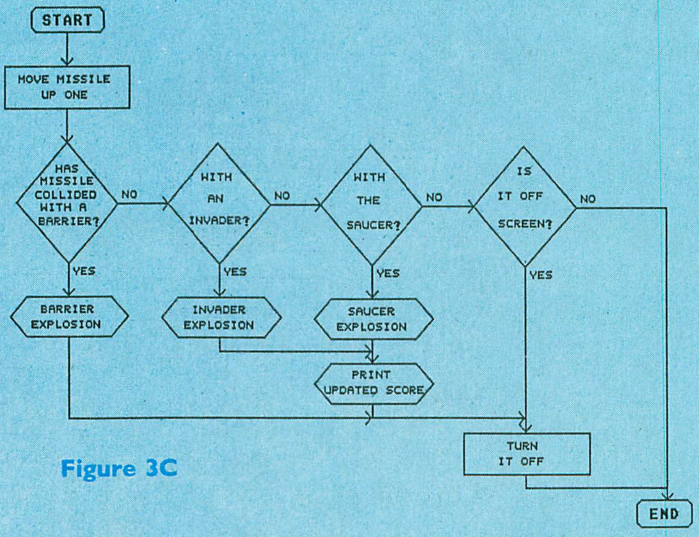


Figure 3C

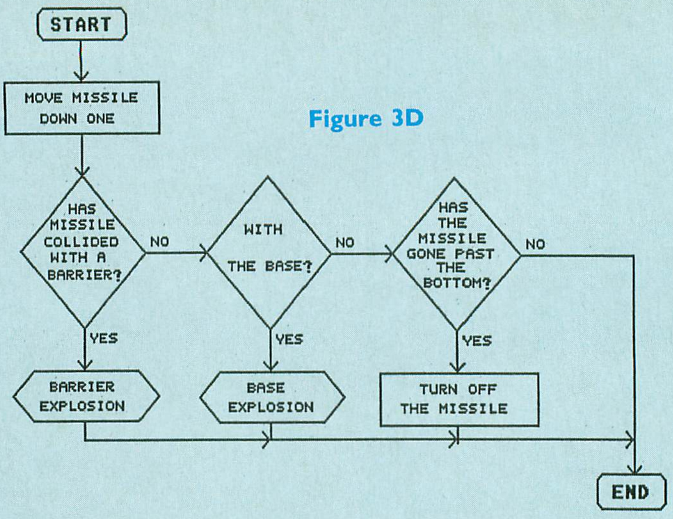


Figure 3D

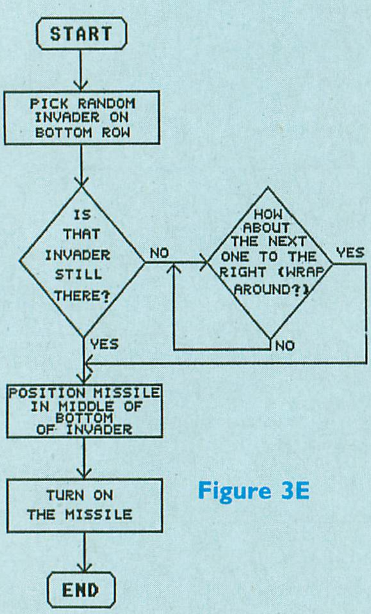

START
PICK RANDOM INVADER ON BOTTOM ROW
IS THAT INVADER STILL THERE?
NO
HOW ABOUT THE NEXT ONE TO THE RIGHT (WRAP AROUND?)
YES
YES
NO
POSITION MISSILE IN MIDDLE OF BOTTOM OF INVADER
TURN ON THE MISSILE
END

Figure 3E

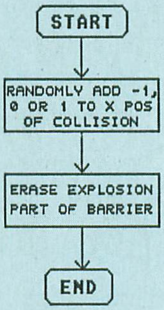

START
RANDOMLY ADD -1, 0 OR 1 TO X POS OF COLLISION
ERASE EXPLOSION PART OF BARRIER
END

Figure 3F

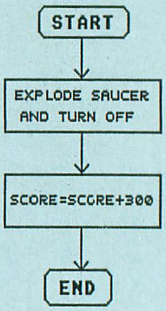


Figure 3H

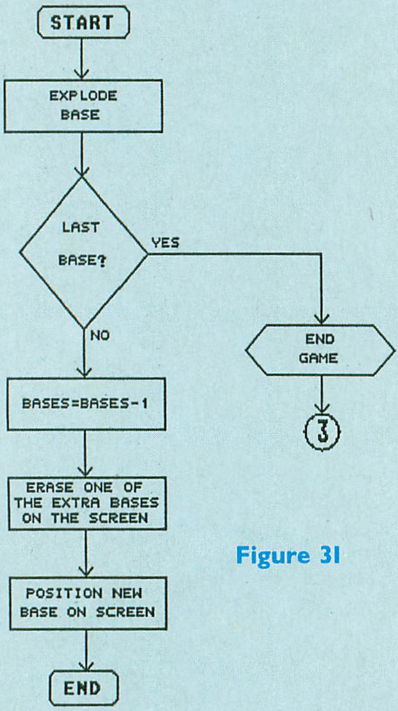

START
EXPLODE BASE
LAST BASE?
YES
END GAME
3
NO
BASES=BASES-1
ERASE ONE OF THE EXTRA BASES ON THE SCREEN
POSITION NEW BASE ON SCREEN
END

Figure 3I

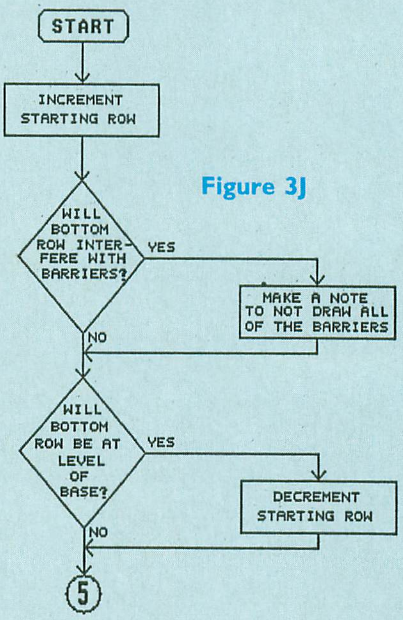

START
INCREMENT STARTING ROW
WILL BOTTOM ROW INTER-FERE WITH BARRIERS?
YES
MAKE A NOTE TO NOT DRAW ALL OF THE BARRIERS
NO
WILL BOTTOM ROW BE AT LEVEL OF BASE?
YES
DECREMENT STARTING ROW
NO
5

Figure 3J

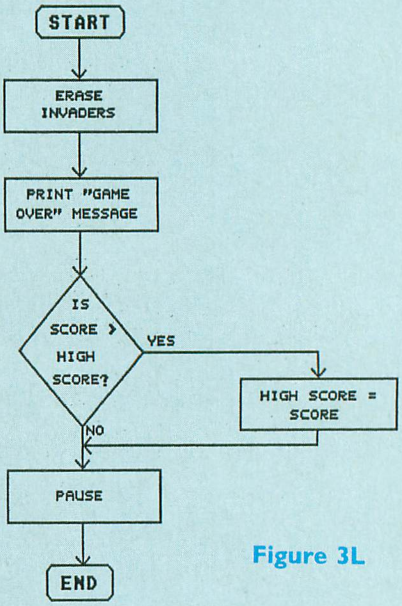


Figure 3L

Now for a summary of each of the routines stored in the lines the program creates:

29000—VBLOFF turns off any VBLANK routines you use.

29500—MOVMEM moves things around in memory.

30000—MISCLR clears one or more of the missiles.

30500—MEMCLR clears memory.

31000—SCROLL takes care of fine and coarse scrolling during VBLANK.

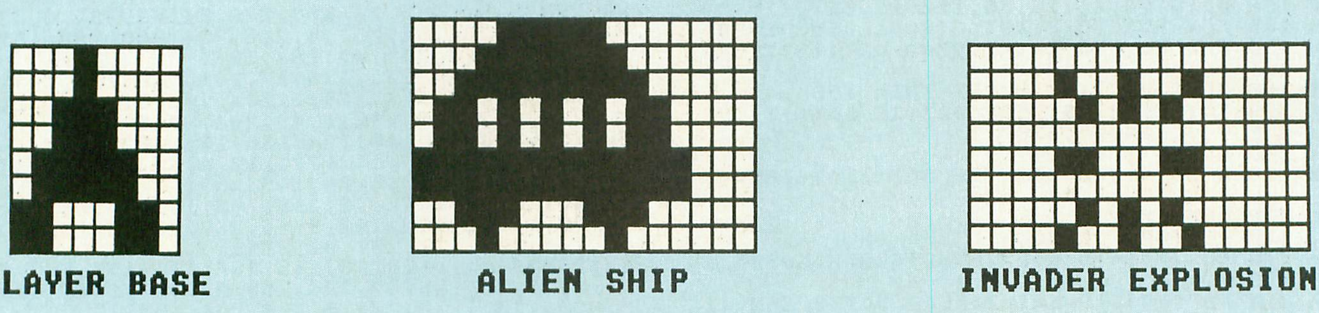


Figure 4

31500—SCRLON gets SCROLL going.

32000-32070—PMOVE lets you move players and missiles around easily during VBLANK, which means that you don't have to worry about it from BASIC.

32500-32510—this isn't a routine, but rather the data for the redefined characters we'll be using in BASIC Invaders.

Throughout the rest of the columns, you should make sure that these lines are included in any segment that uses one of the above routines. Do this by ENTERing the lines back in before or after typing in the segment.

Now, after all this hassle, we're finally ready to start programming a game.

Looks aren't everything (but they're a start)

The first step to writing a game, obviously, is deciding what kind of game you want to write and exactly how things are going to work in it. That's what we did with the flowchart. The next step is deciding how you want the game to look. Perhaps one of the hardest things for a programmer to do is design a game's graphics. Notice that I said "design," not "program." Before all the dazzling details make it to the television screen, they have to be drawn on paper, and there aren't too many programmers that are also artists. Therefore it often takes more time to get the screen looking just right than it does to actually program it. So let's take a look at what goes into getting a good-looking game.

We'll start with the obvious. What are the various kinds of shapes that have to be designed for BASIC Invaders? Well, there are three types of aliens and two versions of each (so that they appear to be moving). There's an alien ship and the player's base. We also have the barriers that protect the player. Did we miss anything? How about the explosion that occurs when the player shoots an alien? I bet you didn't think of that. That's about it, though, as far as the shapes are concerned. Of course, there is also the text, such as "SCORE" and so forth, but that's already been created for us. So we're left with a total of nine shapes that have to be designed. Remembering that these shapes have to be made up of dots, let's go! The shapes we'll be using are shown in Figures 4 and 5.

Of course, it's real easy to look at these shapes and say, "Yup, that's how they look," but what if you were designing an original game? How do you go about coming up with your own shapes? To start with, you should decide how big you want them to be. In making this decision, you should keep in mind how you're going to put the shapes on the screen. For example, anything that moves is either going to be stored in characters or players; in which case you'll have some multiple of eight dots available for width and height (any number up to 256 for player height). So if you end up with a ship that's, say, nine dots wide, you may want to consider shortening it to eight, or taking advantage

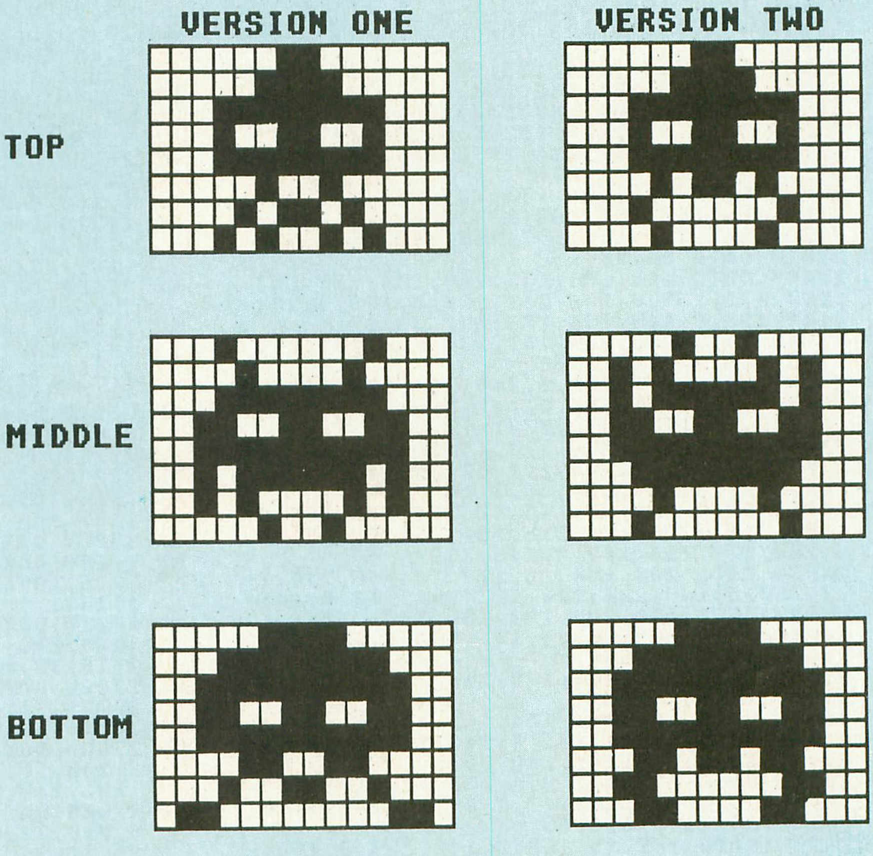


Figure 5

LISTING 1: BASIC

```
VW 100 GRAPHICS 0:? "Make sure you have s
   aved a copy of":? "this program before
    RUNning it":FOR X=1 TO 1050:NEXT X
SQ 110 ? :?
RO 120 DIM LN(8):FOR X=1 TO 8:READ DAT:LN
   (X)=DAT:NEXT X
PE 130 DATA 20,41,26,36,112,11,657,128
OJ 140 FOR X=1 TO 8:TOT=0:N=0:GOSUB 1000
NH 150 FOR N=1 TO LN(X):READ DAT:TOT=TOT+
   DAT
HP 160 IF N/25<>INT(N/25) THEN 190
QP 170 T=TOT:TOT=0:READ DAT:IF DAT<>T THE
   N ? "...ERROR":STOP
QY 180 GOSUB 1000
JW 190 NEXT N:READ DAT:IF DAT<>TOT THEN ?
    "...ERROR":STOP
LM 200 NEXT X
AJ 210 RESTORE 20000
OV 220 FOR X=1 TO 8:L=28500+500*X:GOSUB 1
   010
BP 230 FOR N=1 TO LN(X):READ DAT:? CHR$(2
   7);CHR$(DAT);
TJ 240 IF N/25=INT(N/25) THEN READ DAT
NF 250 IF N/90=INT(N/90) THEN GOSUB 1020:
   L=L+10:GOSUB 1010
RQ 260 NEXT N:READ DAT:GOSUB 1020
MA 270 NEXT X
OH 280 END
LW 1000 ? :? "CHECKING LINE ";19000+1000*
   X+10*INT(N/25);:RETURN
DJ 1010 GRAPHICS 0:POSITION 2,4:? L;" MLA
   NG$=";CHR$(34);:RETURN
CU 1020 ? CHR$(34);":RETURN":? "CONT":POS
   ITION 0,0:POKE 842,13:STOP
UF 1030 POKE 842,12:RETURN
UG 20000 DATA 104,162,228,160,95,169,6,32
   ,92,228,162,228,160,98,169,7,32,92,228
   ,96,2548
QY 21000 DATA 104,104,133,207,104,133,206
   ,104,133,209,104,133,208,104,170,160,2
   55,138,208,2,104,168,177,206,145,3719
EW 21010 DATA 208,136,192,255,208,247,230
   ,207,230,209,202,224,255,208,233,96,33
   40
TH 22000 DATA 104,104,133,207,104,133,206
   ,104,104,168,104,104,133,208,177,206,3
   7,208,145,206,136,192,255,208,245,3931
YN 22010 DATA 96,96
JJ 23000 DATA 104,104,133,204,104,133,203
   ,104,170,169,0,160,255,224,0,208,4,104
   ,168,169,0,145,203,136,192,3396
PM 23010 DATA 255,208,249,230,204,202,224
   ,255,208,234,96,2365
FT 24000 DATA 173,251,6,240,104,173,252,6
   ,141,4,212,173,253,6,141,5,212,173,254
   ,6,240,79,173,48,2,3327
JY 24010 DATA 133,204,173,49,2,133,205,16
   0,3,177,204,201,65,240,61,201,1,240,52
   ,41,112,201,64,144,48,3114
IX 24020 DATA 201,80,144,42,200,173,255,6
   ,48,18,177,204,24,216,109,254,6,145,20
   4,200,177,204,105,0,145,3337
SL 24030 DATA 204,144,20,177,204,56,216,2
   37,254,6,145,204,200,177,204,233,0,145
   ,204,144,2,200,200,200,208,3984
NY 24040 DATA 189,169,0,141,254,6,141,251
   ,6,76,95,228,1556
IE 25000 DATA 104,104,170,104,168,169,6,3
   2,92,228,96,1273
HM 26000 DATA 104,104,104,141,188,6,104,1
   04,141,228,6,141,231,6,141,234,6,141,2
   37,6,238,237,6,141,240,3235
WO 26010 DATA 6,238,240,6,169,127,141,199
   ,6,162,9,160,4,173,47,2,41,16,240,9,16
   9,255,141,199,6,2765
FA 26020 DATA 162,19,160,8,140,200,6,160,
   9,189,206,6,153,189,6,202,136,16,246,1
   69,7,174,240,6,160,2969
OH 26030 DATA 108,32,92,228,96,32,238,6,1
   89,152,6,24,109,200,6,168,205,199,6,14
   4,3,172,199,6,189,2809
BK 26040 DATA 152,6,56,237,200,6,141,201,
   6,136,177,204,200,145,204,136,240,5,20
   4,201,6,176,242,169,0,3450
BE 26050 DATA 145,204,96,32,238,6,189,152
   ,6,56,237,200,6,168,176,2,160,0,189,15
   2,6,24,109,200,6,2759
MY 26060 DATA 141,201,6,200,177,204,136,1
   45,204,200,204,199,6,240,7,204,201,6,1
   44,239,240,237,169,0,145,3855
TM 26070 DATA 204,96,138,72,162,4,32,238,
   6,104,170,189,160,6,56,237,200,6,168,1
   76,2,160,0,189,160,2935
HO 26080 DATA 6,24,109,200,6,141,201,6,13
   6,177,204,61,202,6,145,204,200,200,189
   ,202,6,73,255,49,204,3206
OF 26090 DATA 136,136,17,204,145,204,200,
   200,204,199,6,176,7,204,201,6,144,221,
   240,219,189,202,6,49,204,3719
UU 26100 DATA 145,204,136,189,202,6,49,20
   4,145,204,96,138,72,162,4,32,238,6,104
   ,170,189,160,6,24,109,2994
IH 26110 DATA 200,6,168,205,199,6,144,3,1
   72,199,6,189,160,6,56,237,200,6,141,20
   1,6,200,177,204,61,3152
BK 26120 DATA 202,6,145,204,136,136,189,2
   02,6,73,255,49,204,200,200,17,204,145,
   204,136,136,240,5,204,201,3699
CO 26130 DATA 6,176,224,189,202,6,49,204,
   145,204,200,189,202,6,49,204,145,204,9
   6,189,189,6,133,204,24,3445
GY 26140 DATA 216,173,188,6,125,194,6,133
   ,205,169,0,133,77,96,162,0,188,128,6,4
   8,106,185,120,2,41,2707
MS 26150 DATA 8,208,23,189,148,6,221,136,
   6,240,43,169,0,133,77,254,148,6,189,14
   8,6,157,0,208,208,2931
WH 26160 DATA 28,185,120,2,41,4,208,21,16
   9,0,133,77,189,148,6,221,132,6,240,9,2
   22,148,6,189,148,2652
WY 26170 DATA 6,157,0,208,188,128,6,185,1
   20,2,41,2,208,17,189,152,6,221,144,6,2
   40,30,254,152,6,2668
UX 26180 DATA 32,229,6,138,16,21,185,120,
   2,41,1,208,14,189,152,6,221,140,6,240,
   6,222,152,6,32,2385
EX 26190 DATA 226,6,232,224,4,208,140,162
   ,0,189,164,6,240,83,189,168,6,240,50,1
   6,23,222,156,6,222,3182
NF 26200 DATA 156,6,189,156,6,157,4,208,2
   01,47,176,32,169,0,157,164,6,240,53,25
   4,156,6,254,156,6,2959
EL 26210 DATA 189,156,6,157,4,208,201,208
   ,144,9,169,0,157,164,6,240,106,208,196
   ,189,172,6,240,57,16,3208
XO 26220 DATA 23,222,160,6,222,160,6,32,2
   32,6,189,160,6,201,16,176,39,169,0,157
   ,164,6,240,74,254,2920
AI 26230 DATA 160,6,254,160,6,32,235,6,18
   9,160,6,24,216,105,16,205,199,6,176,4,
   41,240,208,7,169,2830
AX 26240 DATA 0,157,164,6,240,42,189,176,
   6,61,0,208,240,13,169,255,157,176,6,15
   7,184,6,169,0,157,2938
NV 26250 DATA 164,6,189,180,6,61,8,208,24
   0,13,169,255,157,180,6,157,184,6,169,0
   ,157,164,6,232,224,3141
KA 26260 DATA 4,208,145,76,98,228,0,759
NF 27000 DATA 0,0,0,0,0,0,0,0,1,3,7,13,15
   ,2,5,10,128,192,224,176,240,64,160,80,
   1,1321
MD 27010 DATA 3,7,13,15,5,8,4,128,192,224
   ,176,240,160,16,32,8,4,15,29,31,23,20,
   2,16,32,1403
PT 27020 DATA 240,184,248,232,40,64,2,20,
   23,29,31,15,4,8,64,40,232,184,248,240,
   32,16,3,15,31,2245
CW 27030 DATA 25,31,6,9,48,192,240,248,15
   2,248,96,144,12,3,15,31,25,31,13,24,12
   ,192,240,248,152,2437
XA 27040 DATA 248,176,24,48,0,9,5,0,12,0,
   5,9,0,32,64,0,96,0,64,32,16,16,56,56,1
   24,1092
UU 27050 DATA 124,198,198,520
```

of the other seven dots if you're going to use two characters or players. Things that won't be moving, such as the barriers in BASIC Invaders, will be drawn with bit-mapped graphics; in which case they can be any size you want (as long as they don't overlap into the part of the screen that has character graphics).

Once size has been determined, the next step is to come up with the actual shape. The best way to do this is to get some graph paper with reasonably small squares, and block off a section with the number of squares you'll be using for the shape (each square represents a dot). Then sketch a rough version of how you want the shape to look within this area, and color in the squares that your sketch passes through.

You now have your first version, with heavy emphasis on the word "first." This is the stage where somebody will look at your brilliantly designed alien and say, "Hey, nice-looking rock." Don't despair, now is the time to experiment by erasing and filling in dots until you arrive at something that looks good. Luckily, there is a limited number of possible dot combinations; so you're bound to arrive at something that looks right sooner or later. Of course, erasing and filling in dots can be a pain in the you-know-what. The alternative is to use a character editor, which allows you to make these changes on the screen instead of on paper.

Once you get some shapes that, hopefully, you're satisfied with, what's the next step? Are you finally ready to put your creations up on the screen? Not quite. The final step in designing the graphics is to decide how everything is going to be laid out on the screen. Again, this sounds rather obvious, but nothing is obvious to a computer. Everything must be precisely specified. This step involves deciding where the alien saucer is going to fly, where the barriers will be placed, where the player and the score will go and how far across the screen the aliens and player can travel. A lot of these choices will have to do with the display list, which we'll cover later, but the basics can be decided upon without it. The main thing to keep in mind as far as the display list is concerned is that you can't have character graphics and bit-mapped graphics on the same line. Other than this one restriction, you should lay things out in the way that looks best to you. Figure 6 shows the way we're going to do things for BASIC Invaders.

Of course, there's no reason why you can't change any of this to suit your own tastes. As a matter of fact, that's a good point to bring up at this stage. Nothing that I do here, with the exception of the programming techniques, has to be done the way it is. If you don't like my aliens, or if you think later that the scoring system should be different, or if you run across anything that you think can be improved, then go ahead and do it. One of the easiest ways to learn how things are done is to make changes. If you're lucky, then your changes won't work right away, and you'll have to go into the program more deeply to find out what's going wrong. I've learned more by doing this than by any other method; so go ahead and play.

Before we get into the actual programming, there's one more think that should be included that I've already touched on, but haven't really explained. We've already decided that we'll use character graphics for the aliens, bit-mapped graphics for the barriers and (as it will turn out), player/missile graphics for the alien ship and the player's base. How exactly do we go about making these decisions though? Is there a set of criteria that we should use in deciding what to use for which, or can we just do whatever suits us? Obviously I wouldn't have brought it up if we could just choose randomly; so let's take a quick (because it is relatively simple) look at the decision process.

Just what's on the screen!

There are two basic types of objects on the screen: those that move and those that don't. There are three basic types of graphics that can be used: character, bit-mapped and player/missile. Each of these three can be used for either type of object, so you can see that there are a lot of possible combinations. Let's start with the objects that stay still, because they're the easiest. The main rule here is that if the object will change during the course of the game, as in the case of the barriers in Invaders, you should use bit-mapping. The reason is simple; it's much more difficult to change characters than it is to PLOT and DRAWTO. So when should you use character graphics? Sometimes you'll have an object that doesn't move, but is nonetheless animated. Perhaps it's a building with a window that opens and closes, or a flag that waves in the breeze. As you'll see in the columns to come, this is much easier to do with character graphics than it is with bit-mapping.

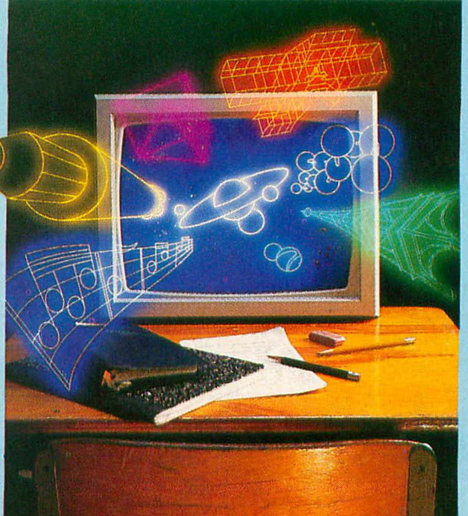
I said that player/missile graphics (PMG) could be used for nonmoving objects as well. Why should you want to use a player for something that doesn't move? After all, the benefit of PMG is that it makes movement easier, right? Right, but it also adds some extra colors to the screen. And, if you're not using all four players, there's no reason why you can't have the ones you're not using sit around and make the screen more colorful.

Things that move

On to the things that move. As long as we're on the topic of PMG, we may as well start there. PMG is best at moving objects over or under other objects. It's the easiest way to move something, period. Of course, you are restricted to objects that are no more than eight dots wide, unless you position two or more players side by side. This means that PMG would not be of help in moving the invaders in our game.

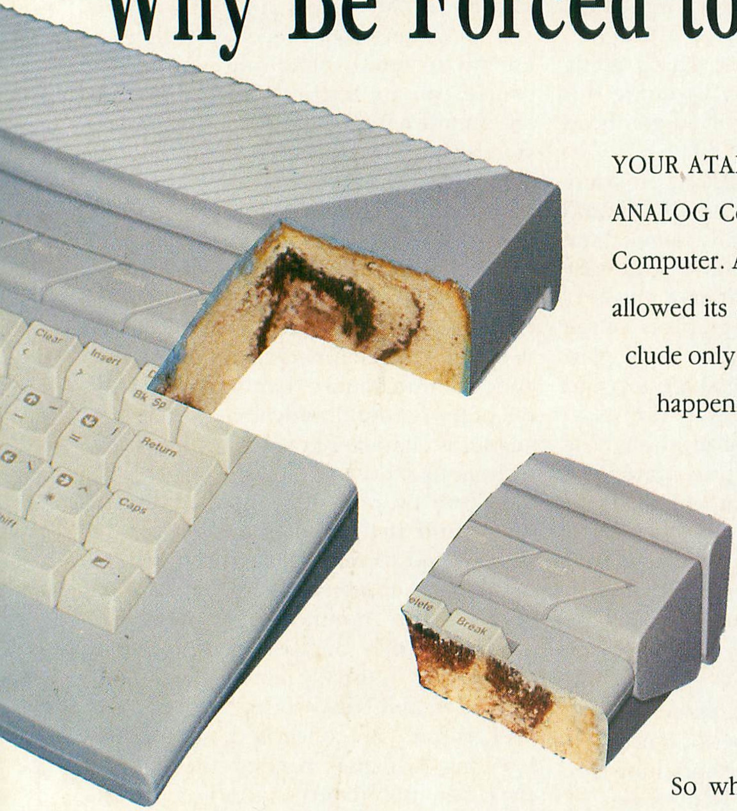
How would we move the invaders? Would we use bit-mapped graphics or character graphics? Because they're relatively slow, bit-mapped graphics are not good for much more than moving a couple of dots around, which means that the answer is character graphics (if it's going to have to move).

Believe or not, that about covers it. You'll find that most games written in BASIC tend to rely a little too much on PMG for movement and bit-mapped graphics for nonmovement or background. Why? Because redefining a character set is usually more difficult than PLOTting and DRAWTOing as far as the background is concerned, and fine-scrolling (which is needed for smooth character movement) is almost impossible to do well from BASIC. Still, we're going to change things a little, by giving you some handy machine-language routines that can be used just as easily as a BASIC statement to get fast professional-looking PMG and fine-scrolling. With the help of these routines, you should be able to break away from the normal and come up with some truly impressive looking games—and all of this without having to learn one bit of machine language. But we're out of time now. See you.



You Own an Atar

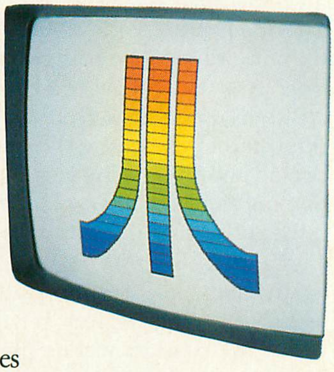
Why Be Forced to Read a Magazine tha



YOUR ATARI RESOURCE CENTER

ANALOG Computing continues to offer exciting products for you and your Atari Computer. And we're the only magazine for the Atari 8-bit computer line that hasn't allowed its content to be virtually taken over by coverage of the Atari ST. We include only a minimal amount of ST material so that you can stay informed of what's happening with the 8-bit computer's brother.

Whether you own a reliable ol' 400 or 800, a shiny XL, new XE or even an XE Game Machine. . . we offer usable utilities, entertaining educational software, dynamite disk programs and great graphics and games. In fact, our readers still use ANALOG programs that were published over five years ago!



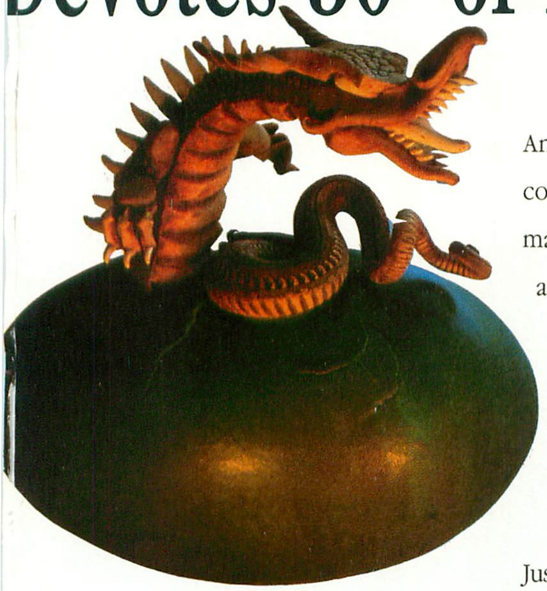
So when software companies turn their heads to other computers, you can turn yours to the one that supports your 8-bit Atari. And that's ANALOG Computing.

ANALOG's Best! Over 88 of ANALOG Computing's best and most requested programs are now available on this series of ten diskettes. The programs are all ready to run and come with complete documentation on the flip side of each floppy diskette. Select from Graphics, Educational, Utilities 1, Utilities 2, Disk Utilities and Games Disks 1, 2, 3, 4 and 5. Only $9.95 each (plus $1.50 shipping per order). Specify disk title when ordering.

Unlock the secrets of your Atari Computer! This handy 16-page pocket reference card covers information you need when programming your 8-bit. Error codes, internal codes, PEEK & POKE locations, machine-language aids, graphic mode specs and BASIC commands with abbreviations are only some of the helpful items at your fingertips. The ANALOG Computing Pocket Reference Card, only $7.95 each! (Plus $1.50 shipping and handling.)

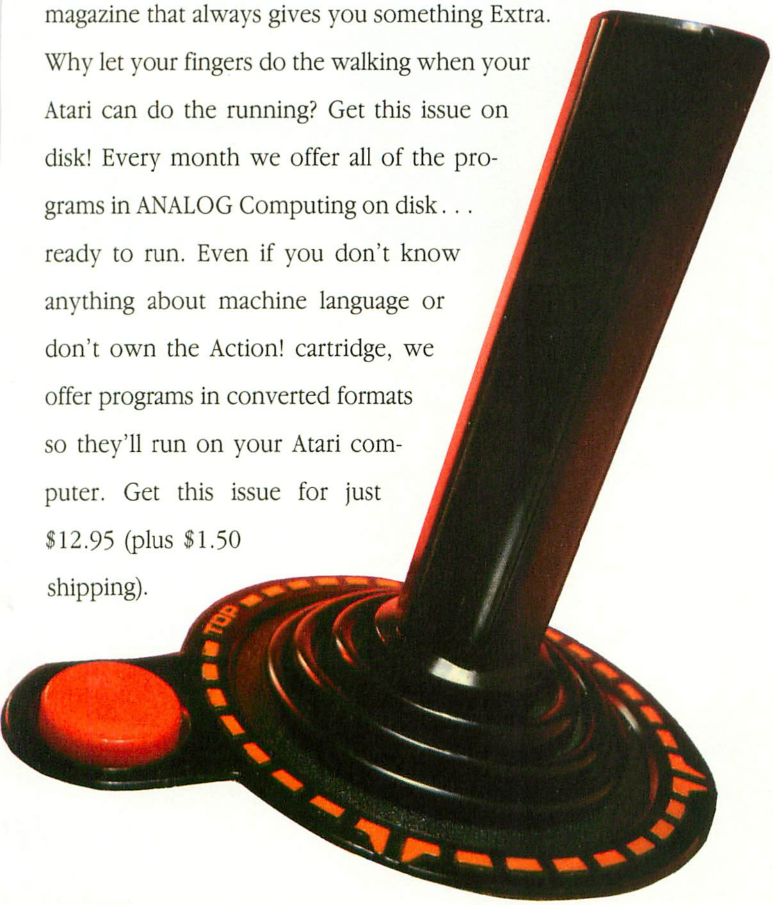
6502 Computer.

Devotes 50% of Its Pages to the Atari ST?



An Atari 8-bit Extra. While other "Atari" 8-bit magazines just make claims on how they cover your machine, we come through! Over 130 pages of new, never before published material. Programs like Easy Type, Dragon Chase, Pastels, Display List Mod, Tactics, Trivia and Create-a-base are all documented and ready to type in and run. . . all for just $8.95! (Add $1.50 for shipping.)

Get the Extra on disk! This special offer for Extra owners gets you all of the programs in an Atari 8-bit Extra on disk. Avoid typing errors, hours of tedious typing and frustration. Just plug in the disk and you are ready to roll! Two, ready-to-run double-sided floppies, $24.95. (Disks only. Atari 8-bit Extra sold separately. Please add $1.50 for shipping.) From the magazine that always gives you something Extra.



Why let your fingers do the walking when your Atari can do the running? Get this issue on disk! Every month we offer all of the programs in ANALOG Computing on disk . . . ready to run. Even if you don't know anything about machine language or don't own the Action! cartridge, we offer programs in converted formats so they'll run on your Atari computer. Get this issue for just $12.95 (plus $1.50 shipping).

ANALOG COMPUTING

ANALOG COMPUTING OFFICIAL ORDER FORM

Use this coupon to order the most complete up-to-date products specifically designed for your ATARI PC!

ANALOG'S BEST—Graphics Disk	**$ 9.95**	$ ______
ANALOG'S BEST—Educational Disk	**$ 9.95**	$ ______
ANALOG'S BEST—Utilities #1	**$ 9.95**	$ ______
ANALOG'S BEST—Utilities #2	**$ 9.95**	$ ______
ANALOG'S BEST—Disk Utilities	**$ 9.95**	$ ______
ANALOG'S BEST—Games #1	**$ 9.95**	$ ______
ANALOG'S BEST—Games #2	**$ 9.95**	$ ______
ANALOG'S BEST—Games #3	**$ 9.95**	$ ______
ANALOG'S BEST—Games #4	**$ 9.95**	$ ______
ANALOG'S BEST—Games #5	**$ 9.95**	$ ______
ANALOG COMPUTING—POCKET REFERENCE CARD	**$ 7.95**	$ ______
ANALOG COMPUTING—8-bit EXTRA	**$ 8.95**	$ ______
ANALOG COMPUTING—8-bit EXTRA (on disk)	**$24.95**	$ ______
ANALOG MAGAZINE ON DISK (please specify issue)	**$12.95**	$ ______
SHIPPING AND HANDLING—add $1.50 for each product ordered		$ ______
TOTAL ORDER		$ ______

□ Payment Enclosed Charge My □ VISA □ Master Card

Card # ______________________ Exp. __________

Signature ______________________

Name ______________________

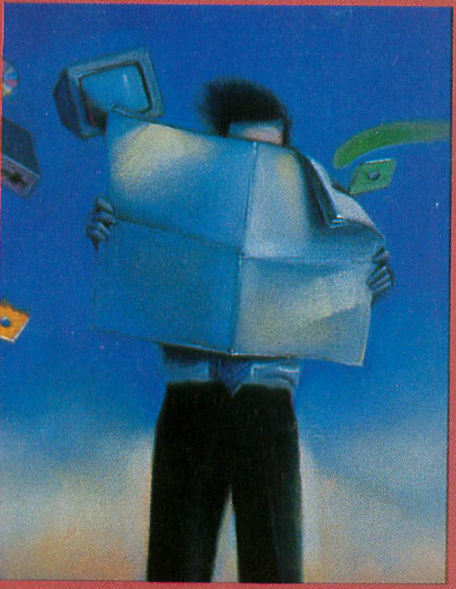
Address ______________________

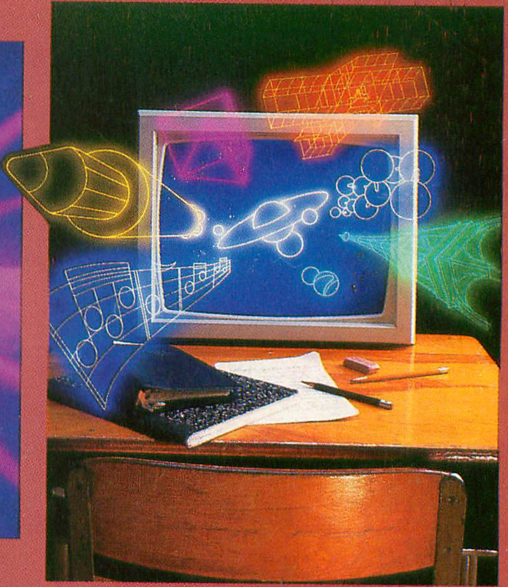
City ______________ State ________ Zip ________

Make checks payable to: **LFP, Inc. P.O. Box 67068, Los Angeles, CA 90067.**
Your order will arrive in 4 to 6 weeks — WATCH FOR IT! ZIHYY

California residents add 6.5% sales tax on all orders except back issues.

INSIDE THIS ISSUE:





MANDLEBROT SET (for 8-bit)

DELPHI GUIDE REVIEW

AWARDWARE REVIEW