THE #1 MAGAZINE FOR ATARI COMPUTER OWNERS

ANALOG COMPUTING

U.S.A. $3.50
CANADA $4.75

ISSUE 60 MAY 1988

INCLUDING
Video Game Digest

DOS-CD:
Double your disc capacity!

APAC System Cloudhopper

ANALOG'S BEST!

Over 88 of ANALOG Computing's best and most popular programs available.

GAMES 1
Stuntman
Fill 'Er Up!
Adventure in the Fifth Dimension
Lumberjack
Space Assault
Darts
Harvey Wallbanger
Supereversion

GAMES 2
Retrofire
Roundup
Livewire
Bricklayer's Nightmare
Knights and Chalices
Air Attack
Avalanche

GAMES 3
Planetary Defense
Crash Dive!
Battle in the B-Ring
Bacterion!
Climber
Money Hungry
Buzz-zap!

GAMES 4
Race in Space
Bopotron!
Bopotron Construction Set
Fire Bug
Basic Burger
Cosmic Defender
Munch'In Clim'In
Shooting Stars

GAMES 5
Adventure at Vandenburg
Popcorn
Demon Birds (Action!)
R.O.T.O. (Action!)
Boulder Bombers
TwoGun
Dragonlord
Lunar Patrol

DISK UTILITIES
Disk Tool (1 and 2)
Burp!
Black Rabbit
Snail
Disk Cataloging Utility
Disk Directory Dump
BASIC Disk Utilities
AlterDOS
Disk Miser

GRAPHICS
Sketch Pad
Graphic Violence!
Multicolor Screen Generator
Moving Missiles in BASIC
Stars 3-D
Bar Chart Subroutine
Solid States
Scredit
Graph E's
P/M Creator/Animator

EDUCATION
Typing Trainer
Observational Astronomy
What Is It?
Spanish Study Guide
Math Attack
Word Scramble
Typing Evaluator
The Reading Program

UTILITIES 1
System Status
Buncrush
Unicheck
Creating an AUTORUN.SYS
NOREM
Hexpad
mUSE
Sound FX
Disassembler in BASIC
ConTEXT
Default

UTILITIES 2
Create-A-Font
Create-A-Font Datamaker
Fast Repeat Key
H:BUG
Auto Line Numbering
Binary File Menu Loader
Minicomp
Saturday Night Special
Touch-Tone™ Dialer
Stopwatch

To order, send $9.95 per disk
VISA (plus $2.50 per order shipping and handling) MasterCard
NO CODs ACCEPTED
to:

P.O. BOX 16927
N. HOLLYWOOD, CA 91615

THE #1 MAGAZINE FOR ATARI COMPUTER OWNERS
ANALOG COMPUTING

CONTENTS

FEATURES

REVIEWS

COLUMNS

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.

Editorial

Picture this:

A man (or a woman, if you like) is flipping through the newspaper when an advertisement from the local computer shop catches his eye. Dazzling graphics! the ad screams. Play arcade quality games in your own living room! A great entertainment value for the whole family!

The old zip and zap glimmers in his eye, and he leans forward, reads more carefully, so as not to miss anything important. He's been thinking about buying a computer. He wants one—wants one badly. But he needs a worthwhile reason, *any* worthwhile reason.

Not games, though. Games are frivolous. Games are time wasters. Games are something you play late Saturday night, when it's dark, when no one is looking. (He arrives at these conclusions as he drools over the screen shots of Flight Simulator II.)

He continues reading, a man on a quest, searching for anything that will allow him to lighten his wallet's green burden, guiltlessly. And then, like a message from the gods, he finds it.

Advanced spreadsheet and word processing capabilities!

"Yes!" the man shrieks, as he tumbles into his car and heads for the shop. "I can work at home!"

Sound familiar? Yeah, I thought it might. More computer purchases have been rationalized in this way than there are leaves on the ground during a New England October. But did you really buy that computer so you could spend your weeknights and weekends slaving over a spreadsheet? (Maybe a couple of you did; stay away from *my* parties.)

Let's be completely honest with each other. What does your computer spend most of its time doing? Playing games, right? Come on, say it. We're all friends here.

Games, games, games.

Didn't that feel good? Aren't you glad we got this all out in the open?

There's nothing to be ashamed of. Games are good, a great way to get the family together some evening, just to share a few hours of joystick jockeying.

But somewhere along the line, entertainment software got a bad name, became the black sheep in the software catalog.

Does it make sense to you? I can't figure it out.

But I *do* have a theory.

I'll bet that somewhere, tucked away in one of those huge glass skyscrapers you see in every city's skyline, there's a group of executives (you know; those people who wear Gucci shoes and drive BMWs) who came up with an idea: Convince people that entertainment software is a bane to modern society and productivity software is the best thing since hot fudge sundaes and—presto!—you've got a crew who'll not only put in their time at work, but get a little extra in at home too!

Call me paranoid, if you like, but sometimes I wonder.

Luckily, you and I know better. That's one of the reasons you read **ANALOG Computing**. You know that, when it comes to games, no one can keep up with us. And, as Greg Knauss's "Cloudhopper" proves, this month is no exception. Arcade quality software tucked neatly into the pages of a magazine. What a bargain!

And we haven't forgotten the younger members of your family. "Money Pouch" by Chuck Rosko is a charming coin counting game that will keep the younger children entertained, while it sneaks in a little education on the side.

When there's a lull in the fun, don't forget to check out Charles F. Johnson's unusual and useful utility found with the article "Binary Load Pictures." Now you can take those Micro Illustrator or Micro Painter picture files and convert them to a binary form that can be loaded and viewed directly from DOS, enabling you to share your masterpieces with friends who may not have the necessary software to view them otherwise.

Or how about Matt Ratcliff's interesting "Busy Buddy Express"? The next time you get distracted for a minute or two from your local BBS, you don't need to fear the dreaded Time-out Syndrome. "Busy Buddy" will help keep that BBS occupied until you get back to it.

For you assembly language programmers, we've got Charles Bachand's "MAC/65 Detokenizer," which will take tokenized MAC/65 files and convert them back to text form. (Charlie spent many an hour laboriously working his way through tokenized files, decoding their secrets byte by byte. It was a horrible thing to watch.) If you've got an assembler other than MAC/65, you'll find this program invaluable.

And, of course, there's more—much more.

But you don't need me to tell you that. Place your fingers at the edges of this page and give it a little flick. It's time for the fun to begin.

Clayton Walnum
Technical Editor
ANALOG Computing

American Techna-Vision

For Orders Only - 1-800-551-9995
CA. Orders / Information 415-352-3787

"Providing 8 Bit support with one of the Worlds largest inventories of Atari replacement parts"

- No surcharge for VISA/MasterCard
- Your card is not charged until we ship

800 4 PIECE BOARD SET

Includes Main Board, Power Supply Assembly, CPU Module and 10K Revision B Operating System Module. All boards are new, tested and complete with all components. $28.50

1050 MECHANISM

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

2400 BAUD MODEM

Mitak Hayes compatible. Auto dial, auto answer. 0-300/1200/2400 baud Internal speaker. Auto baud rate and format adjustment. Touch tone and pulse dialing. Auto busy redial.
Works with All Atari's
8 bit requires interface. $159.95
ST Cable sold seperately.

POWER PACKS

Exact replacement transformer for 800/400, 1050, 810, 1200XL, 850 and 1020 units. Replaces older "weaker" units. Atari part #CO17945. $14.50

400 3 PIECE BOARD SET

Includes Main Board, Power Supply Assembly and CPU Module. All boards are new, tested and complete with all components. $19.50

JOYSTICK

Works with all Atari Computers
ORIGINAL STYLE $7.00

ATARIWRITER CARTRIDGE

Popular cartridge version turns any 8 bit Atari into a powerful word processor. Written by Atari. Disk drive supported but not required.
For all Atari's except ST $29.95

600XL 64K UPGRADE

Easy to install internal modification allows you to hook up a disk drive and run all 800XL software. Kit includes all parts and detailed instructions. Soldering required to install 3 jumpers. $29.95

800 10K "B" O.S. Module

Older 800 units need the revision "B" Operating system to run newer software. Type the following peek in BASIC to see which revision you have. PRINT PEEK(58383).
If the result is 56 order now! $9.50

PILOT PROGRAMMING LANGUAGE PACKAGE

Includes PILOT cart. with "Turtle Graphics", Pilot Primer and Student Pilot manuals. PILOT is an excellent learning or teaching tool.
Works with all Atari's except ST. $17.50

PADDLE CONTROLLERS

(Pair). Required for numerous 8 bit programs and applications. Use these to add two changeable variables to your BASIC or machine language programs. $6.50

SERIAL I/O CABLE

High quality 13 pin cable used to connect 8 bit Atari's to disk drives, interfaces, etc.
New low price $5.95

800/400 MODULES

NEW PARTS COMPLETE WITH IC'S

$9.50 EACH

- 800 Main Board
- 800/400 CPU with GTIA
- 800 10K "B" O.S. Module
- 400 Main Board
- 800 Power Supply Board
- 400 Power Supply Board

16K Ram Module $14.50

INTEGRATED CIRCUITS

$4.50 EACH

- CPU..... CO14806
- POKEY..... CO12294
- PIA..... CO14795
- GTIA..... CO14805
- ANTIC..... CO12296
- CPU..... CO10745
- PIA..... CO10750
- CPU..... CO14377
- DELAY..... CO60472

MORE IC'S

CO60302 XL BASIC ROM	$13.50
1050 O.S. ROM	$13.50
2793 1050 FDC	$19.50
CO10444 2600 TIA	$4.50
1771 810 FDC	$10.00
1050 5713 STEP DRIVER	$5.25

REPAIR MANUALS

SAMS Service Manuals for the following units contain schematics, parts listings, labelled photographs showing the location of checkpoints and more! A special section gives oscilloscope and logic probe readings allowing you to narrow the malfunction down to a specific chip or transistor!
800, 800XL, 130XE, 400, 1025 and 1050 ..... $19.50 each
520ST Service Manual. $37.50

MISC. HARDWARE

1050 Track 0 Sensor	$6.50
1050 Stepper Motor	$14.50
1030 Power Pack	$9.50
Fastchip for 800/400	$15.50
Atari Joystick	$7.00
850 or PR Modem Cable	$14.50
850 or PR Printer Cable	$12.50
P:R: Connection	$59.95
Printer Interface	$39.95
I/O 13 Pin PC mount	$4.50
I/O 13 Pin Plug Kit	$4.50
ST 6' Drive Cable	$14.00
820 Printer Mechanics	$9.50
Joystick Extension Cable	$5.00
EPROM Eraser	$34.95
810 Door Latch Assy.	$15.00
Serial I/O Cable	$5.95
1027 Transformer	CALL
ICD Multi I/O	CALL
810 Tandon Drive Mech.	$49.95

GORF SPACE GAME

Well done cart. vers. of famous arcade game. 800/400 only. $4.00

COMPUTER BOOKS

Inside Atari Basic	$5.00
Atari Basic Ref. manual.	$5.00
Assembly Language Guide	$19.95
XE Users Handbook	$17.95
XL Users Handbook	$17.95
Advanced Programming	$19.50

ATARI 850 INTERFACE

Bare PC Board with parts list and crystal allows you to build your own serial/parallel interface for attaching modems and printers to all 8 bit Atari computers... $7.50
Bare Board & all plug in IC's $39.50

ATARI XM301 MODEM

Direct connect 300 Baud modem works with all 8 bit Atari's. No seperate interface required. $44.95

P:R: CONNECTION

Serial & Parallel interface allows you to attach standard modems and printers. For all 8 bit. (1200XL requires modification)... $59.95

BASIC CARTRIDGE

Basic Rev. "A" Cart. works with all Atari Computers except ST. 800XL Owners Note! Use this cartridge while programming to eliminate the severe errors in the built in "B" Basic..... $10.00

ADDITIONAL SOFTWARE

Pac-Man cartridge	$4.00
Deluxe Invaders Cart.	$4.00
Journey to the Planets	$4.00
Edt/Asm cart. w/o man.	$10.00
Q*bert cartridge	$10.00
Donkey Kong cart.	$5.00
Eastern Front Cart.	$5.00
Springer Cart.	$5.00
Hard Hat Mack disk	$5.00
D-Bug childware disk.	$5.00
Home filing manager	$7.50
Musical Pilot Ed. Disk	$5.00
Big Math Attack Disk	$5.00
Pathfinder disk.	$5.00
O.S.S. Action Cart.	$57.50
O.S.S. Mac-65 Cart.	$57.50
O.S.S. Basic XE Cart.	$57.50
O.S.S. Basic XL Cart.	$47.50

SERVICE RATES

Flat Service Rates below include Parts & Labor, 60 Day Warranty.

800 Computer	$39.50
850 Interface	$39.50
810 Disk Drive	$69.50
1050 Disk Drive	$75.00
800 Keyboard only.	$25.00

Include $7.00 return shipping and insurance. Include $4.00 shipping for 800 keyboard repair only.

CALL TOLL FREE
1-800-551-9995
IN CALIF. OR OUTSIDE U.S.
CALL 415-352-3787

AMERICAN TECHNA-VISION

(Formerly American T.V.)

Mail Order: 15338 Inverness St., San Leandro, Ca. 94579
Repair Center: 2098 Pike Ave., San Leandro, Ca. 94577

Terms: NO MINIMUM ORDER. We accept money orders, personal checks or C.O.D.s. VISA, Master/Card okay. Credit cards restricted to orders 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. All items guaranteed 30 days from date of delivery. No refunds or exchanges.

Prices subject to change without notice. Send SASE for free price list. Atari is a reg. trademark of Atari Corp.

Circle #101 on reader service card.

ANALOG COMPUTING STAFF

Publisher
LEE H. PAPPAS

Executive Editor
CLAYTON WALNUM

Art Director
KATHY WIESNER

Managing Editor
DEAN BRIERLY

East Coast Editor
ARTHUR LEYENBERGER

Midwest Editor
MATTHEW J. W. RATCLIFF

West Coast Editor
CHARLES F. JOHNSON

Contributing Editors
LEE S. BRILLIANT, M.D.; MICHAEL BANKS; ANDY EDDY; STEVE PANAK KARL E. WIEGERS

Entertainment Editors
ARNIE KATZ; BILL KUNKEL; JOYCE WORLEY

Copy Chief
KATRINA VEIT

Copy Editors
ANNE DENBOK
SARA BELLUM

TYPOGRAPHERS
KLARISSA CURTIS; JUDY VILLANUEVA; DAVID BUCHANAN

Contributers
CHARLES BACHAND; EARL DAVIDSON; ANGELO GIAMBRA; ERIC HUFFMAN; GREG KNAUSS; JOHN W. LITTLE; REGINA; CHUCK ROSCO; THOMAS TANIBA

Production Director
DONNA HAHNER

Production Assistant
STEVE HOPKINS

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

Advertising Production Director
JANICE ROSENBLUM

Subscriptions Director
IRENE GRADSTEIN

Vice-President-Sales
JAMES GUSTAFSON

U.S. newsstand distribution by Eastern News Distributors, Inc., 1130 Cleveland Rd., Sandusky, OH 44870

ANALOG Computing magazine (A.N.A.L.O.G. 400/800 Magazine Corp.) is in no way affiliated with Atari. Atari is a trademark of Atari Corp.

WHERE TO WRITE

Editor, **ANALOG Computing,** 9171 Wilshire Blvd., Suite 300, Beverly Hills, CA 90210

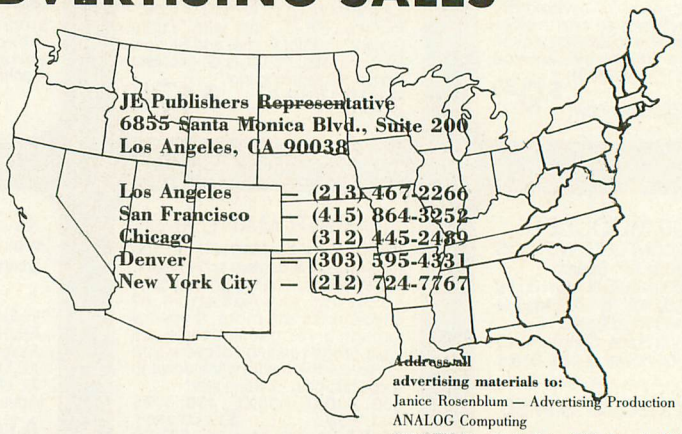
Correspondence regarding subscriptions, including problems and changes of address, should be sent to: **ANALOG Computing,** P.O. Box 16927, North Hollywood, CA 91615, or call 1-818-760-8983.

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:
Janice Rosenblum — 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 further 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.

In addition, any programs used must state that they are taken from **ANALOG Computing.** For further information, write to **ANALOG Computing,** P.O. Box 1413-MO, Manchester, CT 06040-1413

Reader comment

Plaudits...

Thank you for including my program, Creative Process, among your "Four Star Software Picks." I felt proud that I have somehow contributed to the Atari community. The honor meant even more when I realized that Ian Chadwick had picked Creative Process. His book, *Mapping the Atari*, is the best reference for Atari computers that I've used, and Mr. Chadwick is one of the people I respect most in the Atari world. With your selections of CP and RAMbrandt, both from the Antic Catalog, you have risen above the rivalry expected of competing publishers.

Cheers for bringing back memories of some good software, and for making the future seem a bit brighter for Atari's 8-bit line. A computer with the quality software you highlighted will not become obsolete simply because newer systems are available!

Sincerely,
Dave Thorson
Phoenix, AZ
An avid reader for 48 issues

...and more plaudits

I want to thank you and your staff, as well as Alan Glick and John Faber of A-BUG (the Atari Boston Users' Group of the Boston Computer Society) and all the others who helped make the Northeast Atari Fair at the Worcester, MA Centrum such a success.

I spent several hours there on Saturday, and was greatly impressed by the crowds (which I presume were evident on Sunday as well), by the corporate presence, and by the enthusiasm shown by Fair staff, visitors and boothholders alike. The displays and demonstrations of both hardware and software were exciting and well presented, and I was pleased to note that 8-bit Ataris still have a place in the scheme of things.

The show made me proud to be an Atarian, and confirmed my belief that those who look down on Atari don't know what they are missing, and are unconsciously manifesting "sour grapes."

Atari-ally yours,
Miss Dorothy Nash
Dover, NH

A difference of opinion

This letter is in response to an article in issue 58 of **ANALOG Computing**, entitled "Artificial Intelligence." I think the author, Dr. Ron Schaefer, has the wrong idea of what artificial intelligence really is.

I think the title for his article should have been something along the lines of "Deductive Reasoning" or "Question and Answer." His program follows a series of questions that have a yes or no answer until the questions come to an end and a result can be given to the user. The user has to make *and* answer his own questions. This program, in my opinion, does not display artificial intelligence.

Artificial intelligence can be described in terms of human intelligence. Artificial intelligence is something done by a computer that would be considered intelligent if a human did it. For example, if it's cold in your house, you turn up the thermostat. It would be considered artificially intelligent for a computer to decide that it's cold, discover a solution, and carry out that procedure *without* being told specifically how or when to do it.

A truly intelligent program—one that would work similarly to Dr. Schaefer's program—should be able to form questions *on its own*, based on information that it needs and information that it already has. Then it should decide what it needs do with the results, and how to continue after that.

One example of artificial intelligence is a spelling checker. A normal spelling checker takes each word and compares it with its dictionary. While this method catches most errors, it doesn't work well if you have a word that's spelled right, but in the wrong context. (Example: "Go *two* the store.") An intelligent computer would see that this doesn't make sense and report an error. A "stupid" computer, on the other hand, would only see the word as being spelled correctly.

I do not want your readers to get the impression that Dr. Schaefer's program is an example of artificial intelligence. It's very hard—if not impossible—to write a program that exhibits intelligence on a computer like the Atari 520ST, in a language such as BASIC, or even higher level languages like C or Pascal. I think his program has some application (such as identification of certain objects, using descriptions as questions), but the title is an exaggeration of its capabilities.

Sincerely,
David Martin
Columbia Station, OH

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.

Gear up Your Disk Drive For Big Savings!
Save $14 Off The Cover Price.

SUBSCRIBE TO

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

ANALOG COMPUTING

☐ 1 Year.....$28.....Save $14
MCEYY

☐ 1 Year with Disk.....$105
DCEYY

Name

Address

City State Zip

Make checks payable to: L.F.P. Inc. Allow 4-6 weeks for delivery.

☐ Payment Enclosed ☐ Bill Me
☐ Charge My ☐ Visa ☐ MC
_____ Exp ____

Signature

FOREIGN—Add $7 per year
MONEY BACK if not delighted
Analog
P.O. Box 16927
N. Hollywood, CA 91615

Nobody's perfect

And speaking of "Artificial Intelligence" . . . Looks like we goofed. We inadvertently omitted the last portion of Listing 1 from that program. Our apologies to the author and our readers. The following lines should be added to the end of Listing 1 (page 74):

```
2080 linef 409,150,158,150
:linef 158,150,158,90
2090 linef 155,87,412,87:l
inef 412,87,412,153
2100 linef 412,153,155,153
:linef 155,153,155,87
2110 gotoxy 20,10:?"Enter
the new answer"
2120 case(rule,2)=flen:fle
n=flen+1
2140 case(flen,1)=0:case(f
len,2)=-1
2210 a=flen:b=2:gosub INLI
NES
2220 clearw 2:return
```

NFL update

In issue 57 of **ANALOG** was an excellent program called "NFL Game Analyzer." I read in the Listing and ran the program about two hours after the magazine showed up in my mailbox.

When viewing option 4 (the only non-entry option that works in preseason), I noticed a slight discrepancy in the ratings shown from those listed on page 8. (Not that I needed the printed listing to tell that the St. Louis Cardinals shouldn't be listed at number 1.) After about an hour of gazing blankly at the code (it was 2:30 a.m.), I realized that the problem lies in the data in Listing 2. It seems that the rates for the Detroit Lions are left completely out of the listing, and three strange rates are tacked on to the end of the code. In order to rectify the problem, Lines 80 and 140 of Listing 2 should be changed to the following:

```
80 DATA 100.0,100.8,100.4,
103.7,111.1,107.4,101.1,98
.3,99.7,98.1,93.1,95.6,95.
7,87.1,91.6,105.2,98.8,102
.3
140 DATA 102.5,95.4,99.2,1
07.0,97.5,102.2
```

After the lines are changed, Listing 2 must be run again to put the correct rates on file. This will, however, destroy the previous rates. If you have already entered the scores for the preseason, you must erase SCORES.DAT from the disk and reenter them for the program to be completely accurate. Don't panic if you don't have the scores, though. The only thing that can happen is that the predictions for the first part of the season might be a little bit off.

Despite this rather annoying glitch, this program is well done and will surely get a whole load of use. My hat's off to Mr. Genson for creating a program that every football fan can appreciate.

Sincerely,
Keven Mizera
Johnson City, NY

Stay tuned for the exciting conclusion

I've been a reader of **ANALOG** for a few years and have always enjoyed the magazine. Many programs, "Multicopy" in particular, have made my relationship with my 130XE a happy one.

The length of a program has never been of great concern to me. I may be a bit demented, but the longer it takes to type in a program, the more enjoyment I derive from seeing the finished product. It was then with great chagrin that I found the listings for "Troll War II" (issue 57) came in two parts. I've always hated installments, series or anything of the sort. I would have preferred to see all of the data printed in one issue. Then I could make the choice as to how long I wanted to spend typing it in. Now I have to wait for part II.

In the past, **ANALOG** has published other lengthy programs, such as "Treasures of Barboz" (480 lines). The length of a program does not daunt Atari addicts like myself.

You did want readers' opinions, so you have mine.

Thank you for a great magazine.
Donald Zelaya

*Thank you for your comments. Unfortunately, there's more involved in printing a long listing like "Troll War II" than whether or not a person would want to type it in one sitting. Believe it or not, putting together a magazine like **ANALOG Computing** is a delicate operation, and in order to keep the magazine's balance, we only have a certain number of pages we can allot to any particular program. When one article or program takes up more room than it should, something else gets bumped out, making some other reader unhappy. We thought splitting a long listing like "Troll War II" was a good compromise.* —*Ed.*

MOVING?

DON'T MISS A SINGLE ISSUE

Let us know your new address right away. Attach an old mailing label in the space provided below and print your new address where indicated.

DO YOU HAVE A QUESTION ABOUT YOUR SUBSCRIPTION?

Check the appropriate boxes below:

☐ New subscription. Please allow 4 to 8 weeks for your first copy to be mailed.

☐ Renewal subscription. Please include a current address label to insure prompt and proper extension.

☐ 1 year — $28.00. This rate limited to the U.S. and its possessions.

☐ Payment enclosed. ☐ Bill me.

MAIL TO: **ANALOG Computing,** P.O. BOX 625, HOLMES, PA 19043

Name _____

Street Address _____

City _____ State ___ Zip ___

ATTACH LABEL HERE

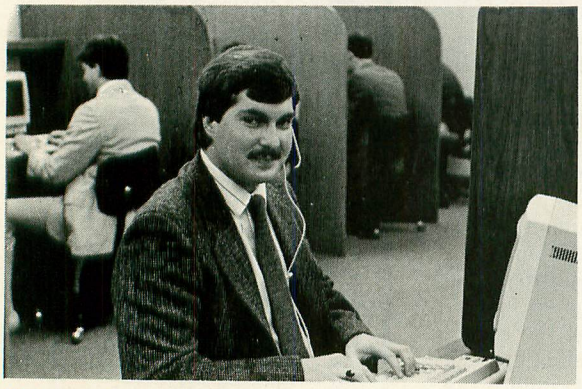
(IF LABEL IS NOT HANDY, PRINT OLD ADDRESS IN THIS SPACE.)

Lyco Computer

Marketing & Consultants

Since 1981

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

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 • compatability not guaranteed • return authorization required • price/availability subject to change • Prepaid orders under $50 in con., U.S. add $3.00.

Monitors

Thomson:

230 Amber TTL/12"	$79.95
4120 CGA	$225.95
4160 CGA	$259.95
4460 EGA	$319.95
4375 UltraScan	$389.95
GB 100 EGA Card	$129.95
GB 200 Super Card	$219.95

***Quantities Limited**

THOMSON 4120 Monitor

- 14" RGBI/video composite/analog
- Compatible with IBM and Commodore
- RGB data cable included

$225.95

Blue Chip:

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

NEC:

Multisync II	$599

Save $210 over NEC Multisync with Thomson 4375 UltraScan $389.95

Modems

Avatex:

1200e	$69.95
1200i PC Card	$69.95
1200hc Modem	$89.95
2400	$179.95
2400i PC Card	$169.95

Hayes:

Smartmodem 300	$149.95
Smartmodem 1200	$285.95
Smartmodem 2400	$425.95

Smarteam 1200 Baud Modem

OUR PRICE **$89.95**

Hayes Compatible

ATARI

Access:

Triple Pack	$11.95
Leader Board Pack	$14.95

Activision:

Hitch Hikers	$13.95
Music Studio	$19.95

Broderbund:

Print Shop	$25.49
Print Shop Compan.	$22.95
Graphic Lib. I, II, III	$13.49
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

Firebird:

The Pawn	$22.95

Microleague:

Microleag. Baseball	$22.95
General Manager	$16.95
Stat Disk	$13.95

Microprose:

Conflict in Vietnam	$22.95
F-15 Strike Eagle	$19.95
Kennedy Approach	$13.95
Silent Service	$19.95
Top Gunner	$13.95

Strategic Simulations:

Battle of Antetiem	$28.95
Battlecruiser	$33.95
Nam	$22.95
Phantasie	$22.95
Wargame Construc.	$16.95
Warship	$33.95
Wizards Crown	$22.95

Sublogic:

Flight Simulator II	$31.49
Night Mission Pinball	$18.95
Scenery #1-#6 ea.	$12.95
Scenery #7	$14.95

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
Road War 2000	$22.95
Colonial Conquest	$22.95

Sublogic:

Flight Simulator II	$31.49
Scenery Disk	$14.95

Timeworks:

Wordwriter ST	$44.95
Partner ST	$39.95
Data Manager ST	$44.95

Unison World:

Art Gallery 1 or 2	$14.95
Print Master	$19.95
Fonts & Borders	$17.95
Music Studio	$27.95
Bureaucracy	$22.95

Electronic Arts:

Arctic Fox	$25.95
Empire	$32.95
Starfleet I	$32.95
Chess Master 2000	$25.95
Gridiron	$32.95

Epyx:

Sub Battle Simulator	$22.95
World Games	$22.95
Wrestling	$22.95
Winter Games	$11.95

Firebird:

Pawn	$25.95
Starglider	$25.95
Golden Path	$25.95
Guild of Thieves	$25.95
Tracker	$25.95

ATARI ST

Access:

Leader Board	$22.95
Tournament #1	$11.95
10th Frame	$22.95

Activision:

Champion. Baseball	$22.95
Champion. Basketball	$22.95
Championship Golf	$New
GFL Football	$22.95

Joysticks

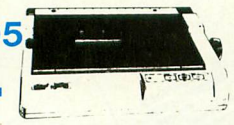
Tac 3	$9.95
Tac 2	$10.95
Tac 5	$12.95
Tac 1 + IBM/AP	$26.95
Economy	~~$5.95~~
Slik Stick	$6.95
Black Max	$10.95
Boss	$11.99
3-Way	$19.99

1-800-233-8760

star micronics · inc NX-1000

- 144 cps Draft
- 36 cps NLQ
- EZ Operation Front Panel Control

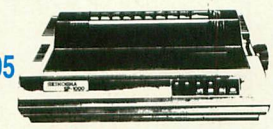
$174.95



SEIKOSHA Sp 180Ai

- 100 cps draft
- 20 cps NLQ

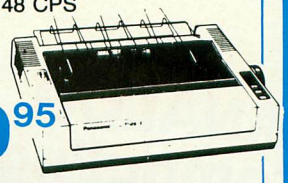
$129.95



Panasonic 1092i

- 240 CPS/12 Character Mode
- Push Feed Tractor
- NLQ Mode 48 CPS

$319.95



PRINTERS

star micronics · inc

NL-10 w/o Cart	$174.95
NX-1000	$165.95
NX-1000C	$179.95
NX-1000 Color	$225.95
NX-1000C Color	$229.95
NX-15	$309.95
ND-10	$265.95
ND-15	$379.95
NR-10	$339.95
NR-15	$425.95
NB-15 24 Pin	$699.95
NB24-10 24 Pin	$425.95
NB24-15 24 Pin	$579.95

BROTHER

M1109	$195
M1409	$299
M1509	$335
M1709	$475
Twinwriter 6 Dot & Daisy	$899
M1724L	$599
HR20	$339
HR40	$569
HR60	$709.95

SEIKOSHA

SP 180Ai	$129.95
SP 180VC	$129.95
SP 1000VC	$139.95
SP 1000AP	$169.95
SP 1200VC	$155.95
SP 1200Ai	$165.95
SP 1200AS RS232	$165.95
SL 80Ai	$299.95
MP1300Ai	$269.95
MP5300Ai	$399.95
MP5420Ai	$879.95
SP Series Ribbon	$7.95
SK3000 Ai	$339.95
SK3005 Ai	$419.95
SPB 10	$CALL
SL 130Ai	$599.95

Toshiba

321SL	$489
341 SL	$659
P351 Model II	$899
351 SX 400 cps	$1019

EPSON

LX800	$179.95
FX86E	$279.95
FX286E	$424.95
EX800	$399.95
EX1000	$469.95
LQ500	$309.95
LQ1000 w/tractor	$549.95
LQ2500	$819.95
GQ3500	$LOW
LQ850	$489.95
LQ1050	$659.95

CITIZEN

120 D	$169.95
180 D	$189.95
MSP-10	$259.95
MSP-40	$309.95
MSP-15	$349.95
MSP-50	$399.95
MSP-45	$459.95
MSP-55	$539.95
Premiere 35	$499.95
Tribute 224	$649.95

Panasonic

1080i Model II	$179.95
1091i Model II	$199.95
1092i	$319.95
1592	$409.95
1595	$459.95
3131	$299.95
3151	$479.95
KXP 4450 Laser	$CALL
1524 24 Pin	$559.95
Fax Partner	$589.95

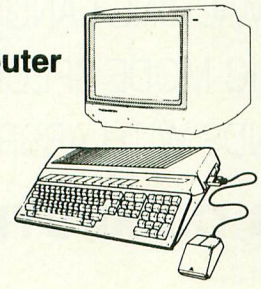
OKIDATA an OKI AMERICA company

Okimate 20	$119
Okimate 20 w/cart	$179.95
120	$189.95
180	$219.95
182+	$225.95
183	$249.95
192+	$309.95
193+	$449.95
292 w/interface	$449.95
293 w/interface	$585.95
294 w/interface	$819.95
393	$955.95

ATARI 520 ST Computer

- Built-in Drive
- Thomson 4120 Monitor

$769.95



ATARI HARDWARE

520 ST FM Mono	$675.95
520 ST FM Color	$819.95
1040 ST Mono	$789.95
1040 ST Color	$975.95
130XE Computer	$135.95
SX551 Drive	$CALL
SF 314 Disk Drive	$219.95
Indus GT Atari Drive	$175.95
SHD 204 20 MEG Drive	$579.95
XM301 Modem	$42.95
SX212 Modem	$89.95
GTS 100 (3.5" DSDD ST)	$195.95

ATARI 520 ST-FM Monochrome System

$675.95

Internal drive included

ATARI 1040 ST Color System

$975.95

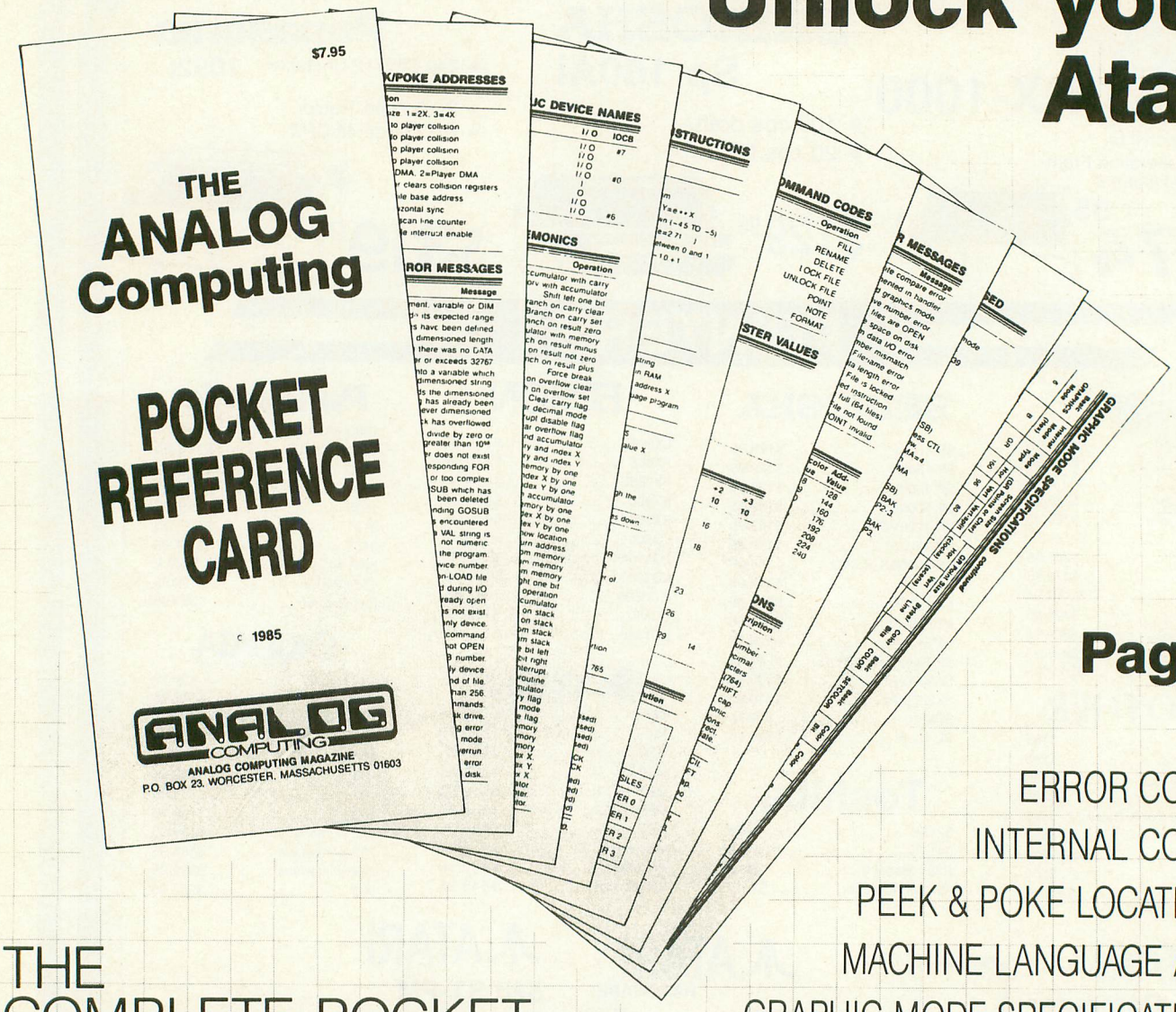
Attention Educational Institutions:
If you are not currently using our educational service program, please call our representatives for details.

ATARI 1040 Monochrome System

$789.95

Circle #104 on reader service card.

Unlock your Atari



16 Pages

ERROR CODES
INTERNAL CODES
PEEK & POKE LOCATIONS
MACHINE LANGUAGE AIDS
GRAPHIC MODE SPECIFICATIONS
BASIC COMMANDS WITH ABBREVIATIONS

THE
COMPLETE POCKET
PROGRAMMING AID

ANALOG COMPUTING

ONLY $7.95 ea.

THE #1 MAGAZINE FOR ATARI COMPUTER OWNERS
ANALOG COMPUTING

P.O. Box 16927
N. Hollywood, CA 91615

YES! Please send me _____ **ANALOG Computing Pocket Reference Cards.** I am enclosing $7.95 per copy.

☐ CASH ☐ CHECK ☐ CHARGE

Name _____

Address _____

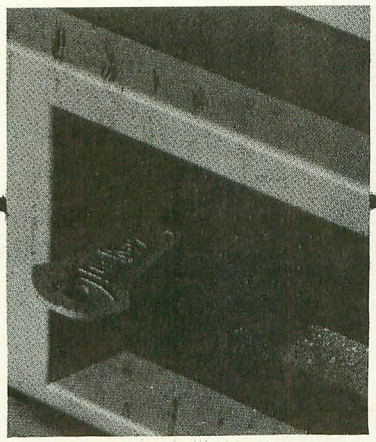
City _____ State ____ Zip _____

Card # _____

Exp. date _____

Signature _____

8-bit news!



THE KEY TO WRITE PROTECTION

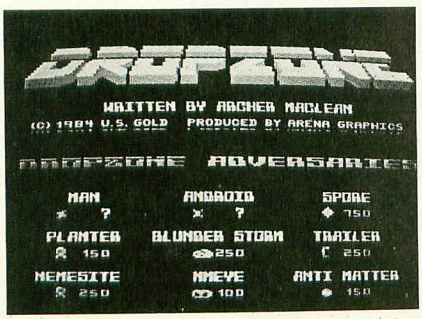
DisKey from CroResearch provides a quick and handy alternative to adhesive-backed write protect tabs. Designed to fit most 5¼-inch disk drives, **DisKey** not only write protects your data when inserted into the left side of the disk drive opening, but it also provides a highly visible indicator as well—and only one **DisKey** is needed for each computer system.

With a suggested retail price of $1.69, the **DisKey** protection system is available from CroResearch, 100 Meadow Court, P.O. Box 129, Lyons, CO 80540 — (303) 823-5088.

CIRCLE #130 ON READER SERVICE CARD

DROPPING IN ON MICRODAFT

People looking for a Defender-like game to help get rid of frustrations after a long day at the office should check into **Dropzone** from Microdaft. Player controls include smart bombs and a cloaking device to make you invisible to the enemy craft (if only for a short time).



With its outstanding graphics and an arcade-style title screen that has high score readout, all that seems to be missing from this game—which retails for $24.95—is a coin slot.

For more information about **Dropzone**, contact Microdaft, 19 Harbor Drive, Lake Hopatcong, NJ 07849 — (201) 663-0202.

CIRCLE #131 ON READER SERVICE CARD

RE-MEMORY'ING THE GOOD OLD 800

Those of us who are still in love with our Atari 800 computers, but haven't seen anything in the way of memory upgrades in some years, will want to look into Magna Systems. This company produces three different versions of their **Ramcharger memory boards** that up your computer's RAM capacity by 256K ($149.95), 512K ($199.95) and a mind boggling 1MEG ($299.95).

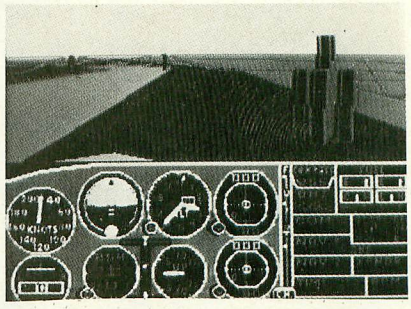
Also included is a specially configured version of MYDOS (version 4.3a) that supports the memory board as a giant RAMdisk, simulating a floppy with 2,000, 4,000 or 8,000 free single-density sectors.

Magna Systems is located at 147-05 Sanford Ave., Suite 4E, Flushing, NY 11355 — (718) 939-0908.

CIRCLE #132 ON READER SERVICE CARD

FALLING IN WITH FLIGHT SIMULATOR

subLOGIC has just released **Scenery Disk II** for its Flight Simulator program, featuring highly detailed views of Detroit, Pittsburgh and Niagara Falls. You can actually fly below the rim of the Niagara River canyon right up to the Falls! (Oh, Stewardess! Why am I perspiring?)



Scenery Disk II also introduces a new default ground pattern that simulates fields and other areas of varying color on the ground below, thus eliminating the monotony of flying between cities. The new ground pattern also improves the user's perspective and makes the scenery much more interesting to look at.

Scenery Disk II lists for $24.95 and is available from subLOGIC Corp., 713 Edgebrook Drive, Champaign, IL 61820 — (217) 359-8482.

CIRCLE #133 ON READER SERVICE CARD

ACCESS DATA ON IBM PC DISKS WITH YOUR 8-BIT

Atari 1050 disk drive owners who have installed an enhancement board from HAPPY Computers can now share data with an IBM PC (or compatible) by using version 7.1 of HAPPY's **Warp Speed Software**. The HAPPY-equipped 1050 is automatically reprogrammed to handle the different sector sizes and file structures of the PC disk. A built-in text conversion feature allows automatic bi-directional translation between ASCII used on the PC and ATASCII used on the Atari.

Version 7.1 can be acquired for $10 by previous owners of the enhancement hardware, whereas first-time users can buy the hardware and software for $99.95 from HAPPY Computers, Inc., P.O. Box 1268, Morgan Hill, CA 95037 — (408) 779-3830.

CIRCLE #134 ON READER SERVICE CARD

"Your wife is back in the hotel room playing with your new computer..."

Quick Screen

A fast and hassle-free technique to produce instant screen displays

by Earl Davidson

Can Atari BASIC produce instant screen displays without complicated page flipping? Yes! **Quick Screen** provides the setup and an easy-to-use routine to include in your BASIC programs. Your screens will "pop" onto your monitor instantly.

The Quick Screen Maker program lets you create up to ten graphic 0 screens using the full screen editor. You can include borders, graphics characters or inverse characters, and once the screens are created, you can save them to a disk file. Write your BASIC program around the skeleton program, **Quick Screen**—which contains routines to load the screen file from disk—and display any screen instantly with a simple GOSUB USEMENU statement. BASIC can now handle your screens almost as fast as machine language—with none of the hassle!

Typing in the programs

There are three program listings. Listing 1 is a program that creates the screen file for Quick Screen Maker. Type it in using "Basic Editor II" and save it to disk as QSMENU.BAS. Run Listing 1. A file named QSMENU.PGE will be created on drive 1. This is simply a data file that will be used by Quick Screen Maker.

Listing 2 should be typed in using "Basic Editor II." Save it to disk as SQUEEZE.BAS before running it. Run the program and a file named SQUEEZE.DAT will be created on drive 1. This file will be used with Listing 3 below.

Listing 3 will be used to prepare two programs. First type in the following lines from Listing 3:

Line 0
Line 120
Line 150
Lines 10000 to 10070
Line 10140

Now type *ENTER "D:SQUEEZE.DAT"* and Lines 10080 through 10120 will be added to your program. At this point, list the program to disk as QS.LST. This program, called **Quick Screen**, will be used when writing your own BASIC programs.

With **Quick Screen** still in memory, type in the remaining lines of Listing 3 to create Quick Screen Maker. List the program to disk as QSMAKER.LST. Listing these two programs to disk is *necessary* so that the variable tables will not be written to disk with the program. Saving the programs at this point will save variables used by "Basic Editor II." Now type *NEW*. Enter QSMAKER.LST and save it to disk as QSMAKER.BAS. Now run the program.

Using Quick Screen Maker

Quick Screen Maker uses nine screens to present menus, allow user input, and to display help information. The screens provide most of the instructions necessary to use the program.

Before choosing main menu options *0* through 9, you should view the help screen. Press *H* from the main menu. The help screen contains the commands used while in the edit mode. Note that all editing commands are SHIFT-CTRL functions: This means you must press and hold the SHIFT and CTRL keys while pressing the appropriate key. You should try to study the editing functions available. When in the edit mode, the help screen is available by hitting SHIFT-CTRL-H.

The screens used by Quick Screen Maker were originally designed with graphic-character borders, boxes, and so on. In order to shorten Listing 1, these have been removed from most screens. You may dress them up yourself, and become familar with the program, by loading (main menu option "L") the QSMENU.PGE file and using Quick Screen Maker to modify the screens. Do not insert or delete any screens while modifying the QSMENU.PGE file. From the

main menu, press a number from 0 to 9 to view and edit a screen.

When editing a screen, press SHIFT-CTRL-n (where *n* is a number from 0-9) to draw one of nine borders available around it. Press SHIFT-CTRL-T to toggle the cursor on and off, and SHIFT-CTRL-R to "record" the screen. (This does not save the file to disk.) Press SHIFT-CTRL-Q to return to the main menu. Save the file to disk (main menu option "S") as QSMENU.PGE. Run Quick Screen Maker again to see the results of your creativity.

Using *Quick Screen*

Quick Screen is the short program you created from Listing 3. Whenever you begin a new BASIC program you should enter QS.LST then write your program around it. Line 0 should not be renumbered—it must be the first line in your program. All other lines can be renumbered. (Don't forget to change any references to the line numbers you change.) Change the value assigned to the variable NUMPAGES in Line 10050 to the number of screens in your file, and change the filename in Line 10140 to the filename containing your screens. You should place a line immediately following Line 10140 with the statement GOTO 200 (or whatever line your main program will start with). All your own program lines should go between Line 150 and Line 10000.

Save $14 Off The Cover Price

ANALOG

☐ 1 Year.....$28.00 Save $14!
MCEYY

☐ 1 Year with Disk.....$105.00
DCEYY

FOREIGN: Add $7 Per Year
Money Back If Not Delighted!

☐ Payment Enclosed ☐ Bill Me
☐ Charge My ☐ Visa ☐ MC
_____ Exp. __

Signature

Name

City State Zip

Make Checks Payable to: L.F.P. Inc. Allow 4-6 weeks for delivery of first issue.

Analog
P.O. Box 16927
N. Hollywood, CA 91615

To display a screen you need a statement to identify the screen to be used: MENUNO=1 will call the second screen in the file. (Screen zero is the first.) Then a GOSUB to the routine USEMENU will display the screen instantly. You may then print to the screen as usual.

A program explanation

The following explanation will refer to Listing 3: **Quick Screen** and Quick Screen Maker.

Line 0 — The screen-handling techniques used by **Quick Screen** utilize a string as screen memory. However, instead of relocating screen memory, a string is modified by changing values in the variable value table. Although any string could be modified, the easiest method is to modify the first string in the table. The first string is always the first variable encountered by the system when a program is entered from disk or the keyboard and may be a constant, array or string.

Using Line 0 to dimension SCREEN$ insures that it will be the first variable in the variable value table. Note that SCREEN$ is dimensioned to a length of one character and assigned a value of " ". Control is then passed to the initialization routine starting at Line 10000.

Line 10020 — VVTP is set to the value in locations 134 and 135, which contain a pointer to the address of the variable value table. This table contains eight bytes of information about each variable. Byte 1 is the variable type. SCREEN$ has been established as the first variable, and VVTP is the address of the first byte in the table. Therefore, VVTP is the location of the byte which identifies the type of variable SCREEN$ will be. On page 67 of *The Atari BASIC Source Book* (COMPUTE! Publications) the different types of variables and their type codes are listed. Type 131 ($83) is defined as "a dimensioned string with an absolute address pointer." Poking location VVTP with a value of 131 changes the variable type of the first variable, which is SCREEN$.

Line 10030 — The second byte in the variable value table is the variable number, which we do not change. The third and fourth bytes (VVTP + 2 and VVTP + 3) contain the address of the start of the string space or data. POKE these locations with the address of screen memory, as found in locations 88 and 89.

Line 10040 — The fifth and sixth bytes (VVTP + 4 and VVTP + 5) contain the actual length of the string. SCREEN$ was dimensioned to a length of one byte. However, changing these bytes to 192 and 3, respectively, changes the length of SCREEN$ to 960 bytes (192 + 3 * 256) which is the length of the graphics 0 screen memory.

The seventh and eighth bytes (VVTP + 6 and VVTP + 7) contain the dimensioned length of SCREEN$. This is changed from one to 960 bytes also.

At this point, SCREEN$ occupies the same memory that has been allocated by BASIC as screen memory. BASIC will not move SCREEN$ around, even if you insert or delete BASIC lines. If your program needs to relocate screen memory for some reason, you should insert your routine to do so *before* Line 10020. Lines 10020 to 10040 will then position SCREEN$ over screen memory, no matter where you locate it.

SCREEN$, as modified above, is the key to **Quick Screen** and Quick Screen Maker. If a change is made to SCREEN$, then the display on the monitor changes instantly. All string-handling functions of Atari BASIC may now be executed directly on screen memory.

Line 10050 — USEMENU is assigned a value of 130, which represents a line number for a routine to be explained later. MENUPAGES=N10 specifies that MENU$ may contain up to ten screens. MENUL is set to the value of MENUPAGES * 960.

When using **Quick Screen** in your own program, you should assign MENUPAGES with the number of screens you'll be using. This takes care of dimensioning MENU$ to the correct length and allows the correct number of bytes to be read from your PGE file by SQUEEZE$.

Line 10060 — MENU$ is dimensioned to a length of MENUL and filled with zeros.

Lines 10070-10120 — SQUEEZE$ is created and the only machine language routine in **Quick Screen** is set up. This routine reads a disk file containing previously designed screens and stores them in MENU$. More on this later.

Line 10140 — Channel 1 is opened to read the file D:QSMENU.PGE, which contains the screens for Quick Screen Maker. The routine in SQUEEZE$ is called by USR.

Line 10150 — The USR call to SQUEEZE$ will return a value to the variable A. If the value of A is 1, 3 or 136, then the file was read properly. If not, the error number is printed by Line 10160 and execution is stopped.

MENUNO is set to 0 so that the the first screen in MENU$ can be displayed by the routine named USEMENU at Line 130.

Line 150 — To display the chosen portion of MENU$, the cursor is turned off with the POKE 752,1. SCREEN$ is set equal to a portion of MENU$ as follows:

```
SCREEN$=MENU$(1+MENUNO*960,960+MENUNO*960)
```

or, in this case:

```
SCREEN$=MENU$(1,960)
```

Remember, MENUNO was set to 0 at this point.

The screen display has been changed to the first page of MENU$. The routine at Line 120 is called.

Line 120 — Several locations are maintained by the operating system to handle the cursor. Since the screen has been changed, these locations must also be changed. Locations 94 and 95 (OLDADR) contain the actual address of the cursor. A PEEK into the value at this location returns the screen code for the character at the cursor location. Poking this value into location 193 (OLDCHR) allows the operating system to properly replace the correct character when the cursor is moved.

The following is an explanation of significant portions of Quick Screen Maker. You may use portions of this in your own BASIC programs.

Line 11060 — PAGE$ is dimensioned to 9,600 bytes (ten screens) and filled with zeros. PAGE$ will be used to hold your screens as you create and edit them. It may be saved and loaded from disk.

The main routine at Line 400 is called by Line 11130.

Line 400 — The main routine begins by setting MENUNO to 1 and displaying the main menu. Do not renumber Lines 427 through 486, as they are accessed by the GOTO in Line 420, based on the value of the key pressed for the menu choice. This method is faster and more memory efficient than several IF statements, but it does result in strange line numbering.

Line 70 — The GKEY routine, beginning at Line 70, is used when creating or editing a screen. The variable A is set to the value of location 764. If A is 255, then no key has been pressed and Line 70 is executed again. If A is less than 200, Line 90 is executed. If A is 200 or greater, but not 255, then a SHIFT-CTRL function is being called.

In order to allow all graphics characters to be entered on the screen, it was necessary to use the SHIFT-CTRL-KEY combination method to allow functions to be executed. Location 764 contains the *internal* hardware value, or keyboard internal code, of the last key pressed. This is not the ATASCII or screen code. The **ANALOG Computing** Pocket Reference Card contains a complete list of the codes returned by PEEK(764).

Be careful to notice the *change* in the value when a key is pressed, as all key combinations do not change the value in location 764. For instance, pressing SHIFT and CTRL and any of the following characters will not change the value: 1, Z, X, C, V, B, J, K, L, ;, +, or *.

Line 80 — Holding the SHIFT and CTRL keys and pressing another key (except those listed above) produces a code above 200 in location 764. This code is used as a line number for the GOTO statement in Line 80. If the line is not found, the TRAP in Line 70 causes the key press to be ignored.

Line 130 — When editing a screen, the string SCREEN$ holds the screen memory as described above. When you press SHIFT-CTRL-R to record the screen, SCREEN$ is copied to the appropriate portion of PAGE$, as determined by MENUNO. The BEEP routine confirms the success of the operation.

Line 140 — This line is used to display a portion of PAGE$, as determined by MENUNO, upon entering the edit mode from the main menu. The BEEP routine is used here also. Note that this line is functionally equivalent to Line 150 (discussed above), except PAGE$ is used instead of MENU$.

Line 160 — This is the BEEP routine which sounds the tone to confirm the successful completion of a function or operation.

Lines 205-255 — These lines execute the various functions available while editing a screen. They are accessed by Line 80 based on the value of A. These lines should not be renumbered.

Lines 900-1230 — These lines contain three routines allowing a screen to be copied, inserted, or deleted to or from PAGE$. After getting input about which screen to change, each routine uses normal string manipulation techniques to modify PAGE$.

The Squeeze routine

This machine language routine is used for two reasons: speed when reading and writing to or from the disk, and to conserve disk space. When a set of pages (PAGE$) is saved to disk without being squeezed, or compacted, the disk file

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 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' checksum 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's correct, press RETURN.

If you find an error, make the correction. When all data's valid, the screen will return to grey, 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).

That's **M/L Editor**. Use it in good health.

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

Listing 1.
BASIC listing.

```
AZ 10 DIM BF(16),H$(4),A$(1),B$(1),F$(15)
   ,F1$(15)
LF 11 DIM MOD$(4)
BN 20 LINE=1000:RETRN=155:BACK5P=126:CHK5
   UM=0:EDIT=0
GO 30 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:? "MOM ON
   LINE: ";LINE:CHK5UM=0
GH 180 L1=3:FOR X=1 TO 16:POSITION 13*(X<
   10)+12*(X>9),X+2:POKE 752,0:? "BYTE #"
   ;X;"! ";: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:CHK5UM=CHK5UM+BYTE*X:IF
    CHK5UM>9999 THEN CHK5UM=CHK5UM-10000
MS 230 NEXT X:CHK5UM=CHK5UM+LINE:IF CHK5U
   M>9999 THEN CHK5UM=CHK5UM-10000
IG 240 POSITION 12,X+2:POKE 752,0:? "CHEC
   K5UM: ";:L1=4:GOSUB 310
EW 250 IF EDIT AND L=0 THEN 270
QM 260 C=VAL(N$)
SY 270 POSITION 22,X+2:? C;"   "
IL 280 IF C=CHK5UM THEN 300
DI 290 GOSUB 440:EDIT=1:CHK5UM=0:GOTO 180
LW 300 FOR X=1 TO 16:PUT #2,BF(X):NEXT X:
   LINE=LINE+10:EDIT=0:GOTO 170
FV 310 L=0
KZ 320 GOSUB 500:IF (A=ASC("Q") OR A=ASC(
   "q")) AND X=1 AND  NOT EDIT THEN 420
PO 330 IF A<>RETRN AND A<>BACK5P AND (A<4
   8 OR A>57) THEN 320
DX 331 IF A=RETRN AND N$="" THEN N$=MOD$
TD 335 IF A=RETRN AND L=0 AND X>1 THEN 35
   0
JR 340 IF ((A=RETRN AND  NOT EDIT) OR A=B
   ACK5P) AND L=0 THEN 320
DH 350 IF A=RETRN THEN POKE 752,1:? " ";R
   ETURN
GG 360 IF A<>BACK5P THEN 400
5A 370 IF L>1 THEN N$=N$(1,L-1):GOTO 390
AS 380 N$=""
RE 390 ? CHR$(BACK5P);:L=L-1:GOTO 320
BB 400 L=L+1:IF L>L1 THEN A=RETRN:GOTO 35
   0
WK 410 N$(L)=CHR$(A):? CHR$(A);:GOTO 320
KN 420 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
```

will be 77 sectors long. In the squeezed format the file length will vary with each set of pages.

The squeeze routine looks for repeating characters, and counts the number of repetitions. For example, suppose the routine finds a series of 320 spaces (eight blank lines). Only four characters are written to the disk: 27, 0, 64, 1. The 27 is the escape character which is used as a flag to indicate the compression. The 0 is the space character internal screen code that is to be repeated. The 64 and 1 represent 320, in least significant/most significant order (64+1*256), which is the number of times to repeat the space. When the file is read from disk, the routine looks for the escape character flag and repeats the next character the number of times necessary before reading the next byte from disk. The routine will read files that are not squeezed without modification because it does not encounter the escape character.

The routine is accessed by a USR call in the BASIC program. Several parameters must be passed to the routine. The first parameter is the address of SQUEEZE$. Next is the channel number to be used. Third is the name of the string that is to hold the file being read from disk, or that holds the file to be written to disk. Fourth is the number of bytes to read. Finally, a 1 is passed if the file is to be read from disk, or a 0 is passed if the file is to be written to disk.

Programming hints

Quick Screen Maker uses all three character codes in the Atari operating system. ATASCII is used for all printing to the screen and normal character input. The internal screen code is used for all direct modifications to SCREEN$. The internal keyboard code is used by the routine at Line 50 to detect the SHIFT-CTRL-KEY combinations. Be careful to use the correct code in the appropriate places of your program.

If you get strange results during initialization of your program you may have a variable in the variable value table before SCREEN$. Try listing your program to disk, type *NEW*, and enter the program back into memory.

Try **Quick Screen** the next time you write a program. It gives a more professional look and feel to your BASIC programs.

Earl R. Davidson has been an avid Atari enthusiast for many years and is president of the Atari Users' Group of Albany, GA. As one of the owners of SoSoft, he wrote the user's manual for InSyst!, a small business inventory program for 8-bit Ataris.

(Listing starts on next page)

COMPUTER SOFTWARE SERVICES
P.O. BOX 17660, ROCHESTER, N.Y. 14617
ATARI® PHONE (716) 467-9326

DEALER/DISTRIBUTOR/USER GROUP Discounts available call for info.
Phone Orders - MASTER CARD, VISA
Mail - Money Orders, Check

ATARI®

THE
$69.95 "SUPER ARCHIVER"!® NOW "BEST BUY" RATED!
(for ATARI 1050 drives)

The new **SUPER ARCHIVER,** obsoletes **all copying devices** currently available for the **ATARI 1050!** It **eliminates** the need for Patches, **PDB files,** Computer Hardware, etc. Copies are **exact duplicates** of originals and **will run on any drive;** without exaggeration, the **SUPER ARCHIVER** is the **most powerful PROGRAMMING/COPYING** device available for the 1050! Installation consists of a plug-in chip and 6 **simple** solder connections. Softwares included. Features are:

- **TRUE DOUBLE DENSITY**
- **ULTRA-SPEED** read/write
- **FULLY AUTOMATIC COPYING**
- **SUPPORTS EXTRA MEMORY**
- **SCREEN DUMP** to printer
- **TOGGLE HEX/DEC DISPLAY**
- **SECTOR** or **TRACK TRACING**
- **AUTOMATIC DIAGNOSTICS**
- **DISPLAYS HIDDEN PROTECTION**
- **ADJUSTABLE/CUSTOM SKEWING**
- **AUTOMATIC SPEED COMPENSATION**
- **AUTOMATIC/PROGRAMMABLE PHANTOM SECTOR MAKER**
- **ARCHIVER/HAPPY ARCHIVER COMPATIBLE**
- **BUILT-IN EDITOR**-reads, writes, displays upto 35 sectors/track (short)
- **BUILT-IN CUSTOM FORMATTER** - upto 40 sectors/track
- **BUILT-IN DISASSEMBLER**
- **BUILT-IN MAPPER** - upto 42 sectors/track
- **DISPLAYS/COPIES** Double Density **HEADERS**
- **AUTOMATIC FORMAT LENGTH CORRECTION**
- **SIMPLE INSTALLATION**

The **SUPER ARCHIVER** is so **POWERFUL** that the only programs we know of that can't be copied are the newer **ELECTRONIC ARTS** and **SYNFILE/SYNCALC** (34 **FULL** sectors/track). If you want it **ALL**. . . . buy the **"BIT—WRITER"!** also. . . . then you'll **be able to copy** even these programs!

$79.95 THE SUPER ARCHIVER "BIT-WRITER"! $79.95

The Super Archiver **"BIT-WRITER"!** is capable of **duplicating even** the "uncopyable" **EA** and **SYN** series which employ 34 **FULL** sectors/track. **"BIT-WRITER"!** is capable of reproducing these and **FUTURE** protection schemes of non physically damaged disks. **PLUG-IN** circuit board and 4 **simple** solder connections. The **SUPER ARCHIVER** with **"BIT-WRITER"!** is the ultimate **PROGRAMMING/COPYING** device for Atari 1050's **EXACT DUPLICATES** of originals are made! Copies run on **ANY** drive.

$69.95 "ULTRA SPEED PLUS!" $69.95

Imagine a universal XL/XE Operating System so easy to use that anyone can operate it instantly, yet so versatile and powerful that every Hacker, Programer and Ramdisk owner will wonder how they ever got along without it! Ultra Speed Plus puts unbelievable speed and convenience at your fingertips. Use **ANY DOS** to place an **ULTRA SPEED** format on your disks, boot any drive (1-9) upon power-up, format your **RAMDISK** in Double Density, activate a built-in **400/800 OS** for software compatibility, plus dozens of other features to numerous to mention! Below are just a **FEW** features you'll find in the amazing **OS:**

- **ULTRA** Speed S10 for most modified drives
- **ULTRA** Speed is toggleable
- Boot directly from **RAMDISK**
- Special timer circuits not required for 1 or 2 Meg upgrades
- Background colors adjustable
- Reverse use of **OPTION** key
- Cold-start without memory loss
- Built in floppy disk configuration editor (1-9)
- Built in **RAMDISK** configuration editor (1-9)
- **RAMDISK** exactly duplicates floppy drive so sector copying and sector editing are now possible
- Built in **MINI** Sector Copier
- Toggle **SCREEN OFF** for up to 40% increase of processing speed
- Toggle internal **BASIC**
- Rom resident disk loader program **(MACH 10** menu)
- **DOUBLE DENSITY RAMDISK** capable
- Entire **MEMORY** test that pinpoints defective **RAM** chip
- Boot any drive (1-9) upon power-up or cold-start
- Supports memory upgrades up to **TWO MEGABYTES**
- **THREE** Operating Systems in one **(XL/XE, 400/800, ULTRA SPEED PLUS)**

$29.95 RAMDISK "WRITE-PROTECTOR!" $29.95

Hackers, Programers, or BBS users. . . if you own a **RAMDISK** (memory upgrades for your XL or XE computer), think about this: Every disk drive ever manufactured has **WRITE-PROTECT** capabilities. . . .except your **RAMDISK.** Without it, your valuable stored data/program lie naked, awaiting that one mistake that will wipe out hours, maybe weeks of precious programming efforts. End the fears of accidental formatting or overwriting by installing our universal **RAMDISK** "Write-Protector!" Works on all memory upgrades up to 2 megabytes. Simple installation. Only $29.95.

$29.95 "XF551 ENHANCER!" $29.95

The XF551 Atari drive is a fine product with one major flaw. . . it writes to side **TWO** of your floppy disks **BACKWARDS.** This causes read/write incompatibility problems with all other single sided drives made for Atari such as Indus, Trak, Rana, Percom, Astra, Atari 1050, Atari 810, etc. Add the **XF551 ENHANCER** to the new XF551 drive and your problems are over! This device will restore 100% compatibility between all drives while retaining all of the original design qualities of Ataris' super new drive. The **XF551 ENHANCER** is a **MUST** for all XF551 owners. Installation is simple. Only $29.95.

Circle #103 on reader service card.

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

Listing 1
BASIC listing

```
KD 10 REM Q5MENU MAKER by Earl Davidson
GC 20 REM Creates the file Q5MENU.PGE
WP 30 REM to be used with
LL 40 REM Quick Screen Maker
MC 50 ? CHR$(125):? "CREATING Q5MENU.PGE"
   :? :? "PLEASE WAIT"
AJ 60 RESTORE
ZH 70 OPEN #1,8,0,"D:Q5MENU.PGE"
KJ 80 TRAP 100
CN 90 READ DAT:PUT #1,DAT:GOTO 90
CZ 100 CLOSE #1:IF PEEK(195)=6 THEN ? "DO
   NE":? :? :END
LD 110 ? "ERROR ";PEEK(195);" AT LINE ";P
   EEK(186)+256*PEEK(187):END
OL 1000 DATA 27,128,38,0,180,173,128,27,0
   ,38,0,128,128,0,72,27,128,4,0,0,128,12
   8,74,0,0
JY 1010 DATA 128,0,72,27,128,4,0,0,128,27
   ,0,5,0,27,128,5,0,0,27,128,5,0,0,128,1
   28
FY 1020 DATA 0,128,74,0,0,128,0,128,74,20
   2,74,0,128,0,128,74,0,0,128,0,128,74,2
   7,0,4
YJ 1030 DATA 0,128,74,0,0,128,0,128,74,27
   ,0,4,0,128,128,0,128,128,128,194,128,0
   ,128,128,74
KA 1040 DATA 202,74,128,0,128,128,128,194
   ,128,0,128,128,74,0,0,0,128,128,74,0,1
   28,0,128,128,74
RT 1050 DATA 202,128,0,128,128,0,128,128,
   128,74,212,0,128,128,128,0,202,212,0,1
   28,128,128,74,212,0
OW 1060 DATA 27,128,4,0,212,0,27,128,4,0,
   212,0,27,128,4,0,212,0,128,128,27,0,38
   ,0,27
ZJ 1070 DATA 128,11,0,89,201,76,201,89,20
   1,201,89,201,89,89,89,207,76,89,201,89
   ,201,76,27,128,21
UO 1080 DATA 0,89,203,79,203,27,89,4,0,20
   1,76,203,89,217,0,89,89,89,203,89,27,1
   28,12,0,27
OY 1090 DATA 0,38,0,128,128,27,0,15,0,112
   ,114,101,115,101,110,116,115,27,0,15,0
   ,128,128,85,85
EY 1100 DATA 79,85,85,79,79,0,79,85,79,85
   ,85,73,0,79,0,0,85,85,79,85,85,79,85,8
   5,79
DX 1110 DATA 85,85,79,85,85,79,79,0,79,85
   ,85,27,128,4,0,89,89,0,89,89,0,89,217,
   0,89
VP 1120 DATA 75,217,73,76,0,0,89,0,76,89,
   0,76,89,0,89,89,0,0,89,0,0,201,79,89,2
   7
IS 1130 DATA 128,6,0,89,89,0,89,89,0,89,2
   17,0,89,0,217,213,89,0,0,213,213,89,89
   ,0,0
BD 1140 DATA 201,207,76,203,85,0,203,85,0
   ,89,75,89,27,128,6,0,89,89,71,89,89,0,
   89,217,0
MO 1150 DATA 89,73,217,0,89,0,0,79,0,89,8
   9,0,79,89,0,89,89,0,0,89,0,0,89,0,89
ES 1160 DATA 27,128,4,0,213,213,76,213,21
   3,76,213,213,76,213,76,213,213,75,0,76
   ,0,0,213,213,76
BY 1170 DATA 213,213,76,76,0,76,213,213,7
   6,213,213,76,76,0,76,213,213,128,128,2
   7,78,38,0,27,128
HC 1180 DATA 4,0,89,98,121,0,37,97,114,10
   8,0,36,97,118,105,100,115,111,110,0,97
   ,110,100,0,42
WT 1190 DATA 111,104,110,0,47,97,107,108,
   101,121,217,27,128,4,0,27,77,38,0,128,
   128,27,0,10,0
EM 1200 DATA 35,111,112,121,114,105,103,1
   04,116,0,8,99,9,0,17,25,24,22,27,0,10,
   0,128,128,27
MC 1210 DATA 0,38,0,128,128,27,0,13,0,48,
   114,101,115,115,0,97,110,121,0,107,101
   ,121,27,0,12
TX 1220 DATA 0,128,128,27,0,38,0,27,128,4
   1,0,27,0,53,0,66,177,181,169,163,171,1
   28,179,163,178
VP 1230 DATA 165,165,174,86,27,0,107,0,66
   ,173,161,169,174,128,128,173,165,174,1
   81,86,27,0,98,0,144
ED 1240 DATA 141,153,0,37,100,105,116,15,
   35,114,101,97,116,101,15,54,105,101,11
   9,0,97,0,51,99,114
GX 1250 DATA 101,101,110,27,0,12,0,163,0,
   0,35,111,112,121,0,97,0,51,99,114,101,
   101,110,27,0
NN 1260 DATA 24,0,164,0,0,36,101,108,101,
   116,101,0,97,0,51,99,114,101,101,110,2
   7,0,22,0,168
XT 1270 DATA 0,0,40,101,108,112,27,0,33,0
   ,169,0,0,41,110,115,101,114,116,0,97,0
   ,51,99,114
VT 1280 DATA 101,101,110,27,0,22,0,172,0,
   0,44,111,97,100,0,102,114,111,109,0,36
   ,105,115,107,27
RK 1290 DATA 0,23,0,177,0,0,49,117,105,11
   6,27,0,33,0,179,0,0,51,97,118,101,0,11
   6,111,0
AR 1300 DATA 36,105,115,107,0,13,0,99,111
   ,109,112,114,101,115,115,101,100,27,0,
   12,0,181,0,0,53
EY 1310 DATA 110,99,111,109,112,114,101,1
   15,115,101,100,0,51,97,118,101,0,116,1
   11,0,36,105,115,107,27
RY 1320 DATA 0,12,0,182,0,0,54,105,101,11
   9,0,51,99,114,101,101,110,115,27,0,24,
   0,165,179,163
PE 1330 DATA 0,36,47,51,27,0,162,0,37,110
   ,116,101,114,0,99,104,111,105,99,101,2
   6,27,0,81,1
ZJ 1340 DATA 165,238,244,229,242,128,166,
   233,236,229,243,240,229,227,154,27,0,4
   0,0,27,77,15,0,27,0
HO 1350 DATA 10,0,37,120,97,109,112,108,1
   01,115,26,27,0,73,0,36,26,57,47,53,50,
   38,41,44,37
PY 1360 DATA 14,48,39,37,27,0,26,0,36,18,
   26,57,47,53,50,38,41,44,37,14,48,39,37
   ,27,0
PV 1370 DATA 103,0,36,105,115,107,0,36,10
   5,114,101,99,116,111,114,121,26,0,101,
   110,116,101,114,0,100
UK 1380 DATA 105,115,107,0,100,114,105,11
   8,101,27,0,10,0,110,117,109,98,101,114
   ,0,8,17,13,24,9
UV 1390 DATA 27,0,66,0,48,114,101,115,115
   ,0,28,50,37,52,53,50,46,30,0,116,111,0
   ,101,120,105
QP 1400 DATA 116,0,116,111,0,45,97,105,11
   0,27,0,12,0,45,101,110,117,27,0,21,1,6
   6,179,163,178
WR 1410 DATA 165,165,174,128,165,164,169,
   180,175,178,128,166,181,174,163,180,16
   9,175,174,128,171,165,185,179,86
KX 1420 DATA 27,0,47,0,66,168,239,236,228
   ,128,179,168,169,166,180,128,134,128,1
   63,175,174,180,178,175,172
FH 1430 DATA 128,225,238,228,128,240,242,
   229,243,243,128,235,229,249,86,27,0,44
   ,0,66,168,86,40,101,108
```

```
KC 1440 DATA 112,0,13,0,100,105,115,112,1
   08,97,121,0,116,104,105,115,0,115,99,1
   14,101,101,110,27,0
WD 1450 DATA 11,0,66,173,86,45,97,114,103
   ,105,110,0,13,0,116,111,103,103,108,10
   1,0,8,0,16,0
MK 1460 DATA 111,114,0,18,0,9,27,0,11,0,6
   6,177,86,49,117,105,116,0,116,111,0,45
   ,97,105,110
JG 1470 DATA 0,45,101,110,117,0,8,50,101,
   99,111,114,100,101,100,31,9,27,0,8,0,6
   6,178,86,50
WH 1480 DATA 101,99,111,114,100,0,115,99,
   114,101,101,110,0,116,111,0,50,33,45,0
   ,8,46,47,52,0
CR 1490 DATA 36,41,51,43,1,9,27,0,5,0,66,
   180,86,52,111,103,103,108,101,0,35,117
   ,114,115,111
AN 1500 DATA 114,0,8,111,110,15,111,102,1
   02,9,27,0,15,0,66,169,86,41,110,118,10
   1,114,115,101,0
OY 1510 DATA 116,111,103,103,108,101,0,99
   ,104,97,114,14,0,117,110,100,101,114,0
   ,99,117,114,115,111,114
QO 1520 DATA 27,0,4,0,66,220,86,53,48,0,1
   14,101,112,101,97,116,0,99,104,97,114,
   14,0,117,110
CD 1530 DATA 100,101,114,0,99,117,114,115
   ,111,114,27,0,9,0,66,221,86,36,47,55,4
   6,0,114,101,112
IU 1540 DATA 101,97,116,0,99,104,97,114,1
   4,0,117,110,100,101,114,0,99,117,114,1
   15,111,114,27,0,7
RX 1550 DATA 0,66,156,86,44,37,38,52,0,11
   4,101,112,101,97,116,0,99,104,97,114,1
   4,0,117,110,100
VZ 1560 DATA 101,114,0,99,117,114,115,111
   ,114,27,0,7,0,66,158,86,50,41,39,40,52
   ,0,114,101,112
FO 1570 DATA 101,97,116,0,99,104,97,114,1
   4,0,117,110,100,101,114,0,99,117,114,1
   15,111,114,27,0,6
OT 1580 DATA 6,66,144,141,153,86,34,111,1
   14,100,101,114,115,0,8,16,0,101,114,97
   ,115,101,115,0,98
HN 1590 DATA 111,114,100,101,114,9,27,0,5
   0,0,84,51,99,114,101,101,110,0,109,117
   ,115,116,0,98,101
NX 1600 DATA 0,2,50,37,35,47,50,36,37,36,
   2,0,98,101,102,111,114,101,27,0,8,0,11
   4,101,116
WV 1610 DATA 117,114,110,105,110,103,0,11
   6,111,0,45,97,105,110,0,45,101,110,117
   ,0,111,114,0,105,116
GI 1620 DATA 0,119,105,108,108,27,0,7,0,9
   8,101,0,2,108,111,115,116,2,14,27,0,29
   ,0,84,38
SP 1630 DATA 117,108,108,0,115,99,114,101
   ,101,110,0,101,100,105,116,111,114,0,1
   3,0,106,117,115,116,0
XZ 1640 DATA 108,105,107,101,0,116,104,10
   1,27,0,6,0,34,33,51,41,35,0,101,100,10
   5,116,111,114,14
VY 1650 DATA 27,0,72,0,48,114,101,115,115
   ,0,97,110,121,0,107,101,121,0,116,111,
   0,99,111,110,116
QC 1660 DATA 105,110,117,101,27,0,8,0,209
   ,27,210,37,0,197,0,252,27,128,6,0,164,
   169,178,165,163
HU 1670 DATA 180,175,178,185,128,175,166,
   128,164,178,169,182,165,128,164,128,15
   4,138,142,138,27,128,6,0,252
IW 1680 DATA 0,193,27,210,18,0,215,27,210
   ,18,0,196,0,252,172,128,174,161,173,16
   5,27,128,4,0,165
UJ 1690 DATA 184,180,128,172,165,174,128,
   252,172,128,174,161,173,165,27,128,4,0
   ,165,184,180,128,172,165,174
KW 1700 DATA 128,252,0,193,27,210,18,0,21
   1,27,210,18,0,196,0,252,27,0,18,0,252,
   27,0,18,0
DK 1710 DATA 252,0,252,27,0,18,0,252,27,0
   ,18,0,252,0,252,27,0,18,0,252,27,0,18,
   0,252
ZB 1720 DATA 0,252,27,0,18,0,252,27,0,18,
   0,252,0,252,27,0,18,0,252,27,0,18,0,25
   2,0
ZI 1730 DATA 252,27,0,18,0,252,27,0,18,0,
   252,0,252,27,0,18,0,252,27,0,18,0,252,
   0,252
OH 1740 DATA 27,0,18,0,252,27,0,18,0,252,
   0,252,27,0,18,0,252,27,0,18,0,252,0,25
   2,27
ES 1750 DATA 0,18,0,252,27,0,18,0,252,0,2
   52,27,0,18,0,252,27,0,18,0,252,0,252,2
   7,0
LK 1760 DATA 18,0,252,27,0,18,0,252,0,252
   ,27,0,18,0,252,27,0,18,0,252,0,252,27,
   0,18
BU 1770 DATA 0,252,27,0,18,0,252,0,252,27
   ,0,18,0,252,27,0,18,0,252,0,252,27,0,1
   8,0
WJ 1780 DATA 252,27,0,18,0,252,0,193,27,2
   10,18,0,216,27,210,18,0,196,0,252,27,1
   28,37,0,252
CQ 1790 DATA 0,218,27,210,37,0,195,27,0,8
   8,0,128,163,128,175,128,176,128,185,27
   ,128,4,0,179,128
BX 1800 DATA 163,128,178,128,165,128,165,
   128,174,128,27,0,240,1,128,176,242,229
   ,243,243,128,156,178,165,180
IT 1810 DATA 181,178,174,158,128,244,239,
   128,229,248,233,244,128,27,0,91,0,46,4
   7,52,37,26,0,0,52
LU 1820 DATA 104,101,0,36,37,51,52,41,46,
   33,52,41,47,46,0,115,99,114,101,101,11
   0,0,119,105,108
OM 1830 DATA 108,27,0,6,0,98,101,0,114,10
   1,112,108,97,99,101,100,0,98,121,0,116
   ,104,101,0,115
HM 1840 DATA 111,117,114,99,101,0,115,99,
   114,101,101,110,14,27,0,7,0,52,104,101
   ,0,51,47,53,50
TX 1850 DATA 35,37,0,115,99,114,101,101,1
   10,0,119,105,108,108,0,114,101,109,97,
   105,110,0,117,110,13
BK 1860 DATA 27,0,7,0,99,104,97,110,103,1
   01,100,14,27,0,196,0,128,169,128,174,1
   28,179,128,165,128
GF 1870 DATA 178,128,180,128,128,128,179,
   128,163,128,178,128,165,128,165,128,17
   4,128,27,0,249,0,37,110,116
JA 1880 DATA 101,114,0,116,104,101,0,115,
   99,114,101,101,110,0,110,117,109,98,10
   1,114,0,8,16,13,25
YU 1890 DATA 9,27,0,11,0,119,104,101,114,
   101,0,121,111,117,0,119,105,115,104,0,
   116,111,0,66,169
RQ 1900 DATA 174,179,165,178,180,86,0,97,
   0,98,108,97,110,107,27,0,6,0,115,99,11
   4,101,101,110,26
SD 1910 DATA 63,27,0,157,0,128,176,242,22
   9,243,243,128,156,178,165,180,181,178,
   174,158,128,244,239,128,229
PO 1920 DATA 248,233,244,128,27,0,91,0,46
   ,47,52,37,26,0,0,51,99,114,101,101,110
   ,0,3,25,0
EF 1930 DATA 119,105,108,108,0,98,101,0,7
   ,112,117,115,104,101,100,7,27,0,7,0,11
   1,102,102,0,116
XV 1940 DATA 104,101,0,102,105,108,101,14
   ,0,0,35,111,112,121,0,3,25,0,116,111,0
   ,97,110,111,116
CW 1950 DATA 104,101,114,27,0,7,0,115,99,
   114,101,101,110,0,34,37,38,47,50,37,0,
   121,111,117,0
```

```
JN 1960 DATA 41,46,51,37,50,52,0,105,102,
   0,121,111,117,27,0,9,0,119,105,115,104
   ,0,116,111,0
OV 1970 DATA 114,101,116,97,105,110,0,105
   ,116,14,27,0,16,1,164,165,172,165,180,
   165,128,161,128,179,163
VV 1980 DATA 178,165,165,174,27,0,178,0,3
   7,110,116,101,114,0,116,104,101,0,115,
   99,114,101,101,110,0
LL 1990 DATA 110,117,109,98,101,114,0,8,1
   6,13,25,9,27,0,17,0,116,111,0,98,101,6
   6,164,165,172
XC 2000 DATA 165,180,165,164,86,26,63,27,
   0,141,0,48,114,101,115,115,0,28,50,37,
   52,53,50,46,30
FX 2010 DATA 0,116,111,0,101,120,105,116,
   27,0,92,0,46,47,52,37,26,0,0,33,108,10
   8,0,115,99
WM 2020 DATA 114,101,101,110,115,0,110,11
   7,109,98,101,114,101,100,0,108,97,114,
   103,101,114,27,0,6,0
VJ 2030 DATA 116,104,97,110,0,116,104,101
   ,0,36,37,44,37,52,37,36,0,115,99,114,1
   01,101,110,0,119
MW 2040 DATA 105,108,108,0,98,101,27,0,9,
   0,109,111,118,101,100,0,111,110,101,0,
   115,99,114,101,101
AB 2050 DATA 110,0,108,111,119,101,114,0,
   119,104,101,110,0,116,104,105,115,27,0
   ,8,0,99,111,109,109
BP 2060 DATA 97,110,100,0,105,115,0,117,1
   15,101,100,14,0,0,51,99,114,101,101,11
   0,0,110,117,109,98
YB 2070 DATA 101,114,0,25,27,0,7,0,119,10
   5,108,108,0,98,101,0,98,108,97,110,107
   ,14,27,0,231
MT 2080 DATA 7
```

•

Listing 2
BASIC listing

```
IE 10 REM STRING MAKER by Earl Davidson
RN 20 REM Creates BASIC lines to assign
HM 30 REM a machine language routine to
KY 40 REM a STRING from DATA statements
SV 50 DIM WORK$(128)
KU 60 DIM DESTFN$(15):DESTFN$="D:SQUEEZE.
   DAT":REM NAME OF FILE TO BE CREATED
XH 70 DIM DEST$(20):DEST$="SQUEEZE$":REM
   NAME OF STRING TO BE CREATED
DL 80 BEGLIN=10080:REM FIRST LINE NUMBER
   TO BE CREATED
YF 90 INC=10:REM INCREMENT LINE NUMBERS B
   Y THIS NUMBER
ZE 100 RESTORE :? CHR$(125):? "CHECKING D
   ATA STATEMENTS";?
LC 110 FOR I=1 TO 387:READ DAT:CHKSUM=CHK
   SUM+DAT:NEXT I
JQ 120 IF CHKSUM<>52443 THEN ? "ERROR IN
   DATA STATEMENTS...":? "CANNOT CREATE S
   TRING":STOP
PD 130 ? "DATA STATEMENTS OK!":? "CREATIN
   G ";DEST$:?
VI 140 RESTORE :TRAP 220:SL=1:OPEN #1,8,0
   ,DESTFN$
OZ 150 WORK$=STR$(BEGLIN):WORK$(LEN(WORK$
   )+1)=" ":WORK$(LEN(WORK$)+1)=DEST$:WOR
   K$(LEN(WORK$)+1)="("
LL 160 WORK$(LEN(WORK$)+1)=STR$(SL):WORK$
   (LEN(WORK$)+1)=")=":WORK$(LEN(WORK$)+1
   )=CHR$(34):LL=LEN(WORK$)
VB 170 READ DAT:WORK$(LL+1)=CHR$(DAT)
IP 180 SL=SL+1:LL=LL+1:IF LL<113 THEN GOT
   O 170
GA 190 WORK$(LL+1)=CHR$(34):LL=LL+1
ND 200 FOR I=1 TO LL:? CHR$(27);WORK$(I,I
   );:NEXT I:? :? #1;WORK$:REM PRINT LINE
   TO SCREEN AND DISK
JQ 210 BEGLIN=BEGLIN+INC:GOTO 150
IP 220 IF PEEK(195)<>6 THEN ? "ERROR ";PE
   EK(195);" AT LINE ";PEEK(186)+256*PEEK
   (187):STOP
FP 230 WORK$(LL+1)=CHR$(34):LL=LL+1
MA 240 FOR I=1 TO LL:? CHR$(27);WORK$(I,I
   );:NEXT I:? #1;WORK$:CLOSE #1
LC 250 ? :? DESTFN$;" now contains the BA
   SIC":? "lines to create ";DEST$
BJ 260 ? :? "To use ";DEST$;" it must be"
   :? "DIMensioned to a length of ";SL-1
HK 270 ? :? "To include the lines in ";DE
   STFN$:? "in your program type:":?
PR 280 ? "     ENTER ";CHR$(34);DESTFN$;CH
   R$(34)
OJ 290 END
SE 10200 DATA 104,208,7,169,0,133,212,133
   ,213,96,201,4,240,8,170,104,104,202,20
   8,251
EZ 10210 DATA 240,237,104,104,10,10,10,10
   ,170,104,133,204,104,133,203,104,133,2
   08,104,133
LI 10220 DATA 207,104,104,208,91,165,207,
   5,208,240,81,160,0,177,203,133,215,230
   ,203,208
G5 10230 DATA 2,230,204,198,207,165,207,2
   01,255,208,2,198,208,165,203,72,165,20
   4,72,165
ZR 10240 DATA 207,72,165,208,72,169,0,133
   ,210,169,1,133,209,165,207,5,208,240,3
   9,160
```


Save $14 Off The Cover Price

ANALOG

☐ 1 Year.....$28.00 Save $14!
MCEYY

☐ 1 Year with Disk.....$105.00
DCEYY

FOREIGN: Add $7 Per Year
Money Back If Not Delighted!

☐ Payment Enclosed ☐ Bill Me

☐ Charge My ☐ Visa ☐ MC

_____ Exp. __

Signature

Name

City State Zip

Make Checks Payable to: L.F.P. Inc. Allow 4-6 weeks for delivery of first issue.

Analog
P.O. Box 16927
N. Hollywood, CA 91615

```
EU 10250 DATA 0,177,203,197,215,208,31,23
   0,203,208,2,230,204,230,209,208,2,230,
   210,198
SG 10260 DATA 207,165,207,201,255,208,2,1
   98,208,56,176,217,240,92,176,165,208,9
   5,165,210
VA 10270 DATA 208,6,165,209,201,4,144,33,
   104,104,104,104,165,209,133,216,165,21
   0,133,217
DA 10280 DATA 169,27,133,214,169,214,157,
   68,3,169,0,157,69,3,169,4,157,72,3,208
LJ 10290 DATA 27,104,133,208,104,133,207,
   104,133,204,104,133,203,169,1,157,72,3
   ,169,215
RX 10300 DATA 157,68,3,169,0,157,69,3,169
   ,0,157,73,3,169,11,157,66,3,32,86
UJ 10310 DATA 228,48,3,56,176,164,132,212
   ,169,0,133,213,96,160,1,165,207,5,208,
   240
DQ 10320 DATA 30,169,215,157,68,3,169,0,1
   57,69,3,169,7,157,66,3,169,1,157,72
ZQ 10330 DATA 3,169,0,157,73,3,32,86,228,
   16,7,132,212,169,0,133,213,96,160,0
LG 10340 DATA 165,215,201,27,240,21,145,2
   03,230,203,208,2,230,204,198,207,165,2
   07,201,255
DJ 10350 DATA 208,2,198,208,56,176,182,16
   9,0,157,73,3,169,3,157,72,3,169,215,15
   7
OI 10360 DATA 68,3,169,0,157,69,3,169,7,1
   57,66,3,32,86,228,48,190,165,216,5
JK 10370 DATA 217,240,41,165,207,5,208,24
   0,35,160,0,165,215,145,203,230,203,208
   ,2,230
TT 10380 DATA 204,198,216,165,216,201,255
   ,208,2,198,217,198,207,165,207,201,255
   ,208,2,198
GX 10390 DATA 208,56,176,209,56,176,174
```

Listing 3
BASIC listing

```
LE 0 GRAPHICS 0:DIM SCREEN$(1):SCREEN$=""
   :GOTO 10000
DA 10 REM ********************************
PO 20 REM *        QUICK SCREEN          *
HF 30 REM *    COPYRIGHT (c) 1986 by     *
MJ 40 REM *       Earl Davidson          *
TV 50 REM *      and John Oakley         *
DF 60 REM ********************************
IM 69 REM Get Key (GKEY)
MS 70 TRAP GKEY:A=PEEK(N764):POKE 99,80:I
   F A=N255 THEN GOTO GKEY
PJ 80 IF A>199 THEN GOTO A
UN 90 GET #N2,KY:KY$=CHR$(KY):POKE 763,N2
   55:IF KY=27 AND ESCFLG THEN POKE 674,N
   0
EK 100 ? KY$;:ESCFLG=0+1*KY=27:GOTO GKEY
JN 109 REM Clear Key (CLKEY)
FI 110 POKE N764,N255:GOTO GKEY
KM 119 REM RESTORE CHARACTER UNDER CURSOR
   (RCH)
YF 120 POKE N93,PEEK(PEEK(N94)+PEEK(N95)*
   N256):? DN$;UP$;:RETURN
WU 129 REM RECORD PAGE
WP 130 PAGE$(N1+PAGENO*N960,N960+PAGENO*N
   960)=SCREEN$:GOSUB BEEP:RETURN
DQ 139 REM USE PAGE
OQ 140 POKE N752,C:SCREEN$=PAGE$(N1+PAGEN
   O*N960,N960+PAGENO*N960):GOSUB RCH:GOS
   UB BEEP:RETURN
SG 149 REM USE MENU
GQ 150 POKE N752,N1:SCREEN$=MENU$(N1+MENU
   NO*N960,N960+MENUNO*N960):GOSUB RCH:RE
   TURN
MA 159 REM BEEP
NE 160 FOR I=16.5 TO N0 STEP -0.75:SOUND
   N0,N10,N12,I:NEXT I:RETURN
DD 169 REM DISPLAY HELP PAGE
KZ 170 GOSUB RCH:MENU$(N1+N9*N960,N960+N9
   *N960)=SCREEN$:MENUNO=N3:GOSUB USEMENU
   :GOSUB ANYK
NE 180 MENUNO=N9:GOSUB USEMENU:POKE 752,C
   :GOSUB RCH:GOTO CLKEY
JE 199 REM SHIFT/CONTROL/FUNCTIONS
DX 205 A=PEEK(N93):POKE N93,A+N128*(A<N12
   8)-N128*(A>N127):? RT$;:GOTO CLKEY
SE 206 A=PEEK(N93):UP=(PEEK(N94)+PEEK(N95
   )*N256)-N40*(UP-SAVMSC>N40):POKE UP,A:
   ? UP$;:GOTO CLKEY
ZS 207 A=PEEK(N93):DWN=(PEEK(N94)+PEEK(N9
   5)*N256)+N40*(DWN-SAVMSC<921):POKE DWN
   ,A:? DN$;:GOTO CLKEY
HX 216 TL=217:TR=207:BL=75:BR=213:H=149:V
   =153:GOTO BORDER
DE 218 TL=N128:TR=TL:BL=TL:BR=TL:H=160:V=
   H:GOTO BORDER
VV 219 TL=84:TR=TL:BL=TL:BR=TL:H=20:V=H:G
   OTO BORDER
MH 221 TL=73:TR=79:BL=75:BR=76:H=18:V=124
   :GOTO BORDER
SK 222 TL=209:TR=197:H=146:V=252:BL=218:B
   R=195:GOTO BORDER
JD 223 TL=81:TR=69:BL=90:BR=67:H=18:V=124
   :GOTO BORDER
LS 229 LM=N0+(N2*(LM=N0)):POKE N82,LM:? C
   R$;UP$;:GOTO CLKEY
BB 232 GOSUB CAPPAGE:GOTO CLKEY
PO 237 C=N0+PEEK(N752)=N0:POKE N752,C:? D
   N$;UP$;:GOTO CLKEY
UQ 239 POKE N764,N255:GOTO MAIN
IU 240 TL=N3:TR=TL:BL=TL:BR=TL:H=35:V=H:G
   OTO BORDER
EH 242 TL=N0:TR=TL:BL=TL:BR=TL:H=32:V=H:G
   OTO BORDER
YS 243 TL=N10:TR=TL:BL=TL:BR=TL:H=42:V=H:
   GOTO BORDER
YN 245 TL=138:TR=TL:BL=TL:BR=TL:H=170:V=H
   :GOTO BORDER
KT 246 A=PEEK(N93):LF=(PEEK(N94)+PEEK(N95
   )*N256)-N1*(LF-SAVMSC>N0 AND PEEK(85)<
   >N0):POKE LF,A:? LT$;:GOTO CLKEY
PC 247 A=PEEK(N93):RT=(PEEK(N94)+PEEK(N95
   )*N256)+N1*(RT-SAVMSC<958 AND PEEK(85)
   <>39):POKE RT,A:? RT$;:GOTO CLKEY
PP 249 GOTO 170
ML 255 GOTO CLKEY:REM 5/C/A not available
IP 299 REM GET ANY CHARACTER (ANYK)
VJ 300 POKE 694,N0:POKE 702,N64:GET #N2,K
   Y:KY$=CHR$(KY):RETURN
LN 310 WORK$="":CHNO=N1
KP 320 GOSUB ANYK:IF KY=59 THEN KY=58:KY$
   =CHR$(KY)
OA 330 IF KY=N155 THEN RETURN
YG 340 IF KY=47 THEN 390
XE 350 IF KY>45 AND KY<59 OR KY>N64 AND K
   Y<91 THEN IF CHNO<16 THEN ? KY$;:WORK$
   (CHNO,CHNO)=KY$:CHNO=CHNO+N1:GOTO N320
CV 360 IF KY<>30 AND KY<>126 AND KY<>254
   AND KY<>156 AND KY<>43 THEN ? CHR$(253
   );:GOTO N320
UF 370 IF LEN(WORK$)>N1 THEN WORK$=WORK$(
   N1,(LEN(WORK$)-N1)):CHNO=CHNO-N1:? CHR
   $(126);:GOTO N320
PD 380 IF CHNO=N2 THEN WORK$="":CHNO=N1:?
    CHR$(126);:GOTO N320
FD 390 ? CHR$(253);:GOTO N320
YW 399 REM MAIN ROUTINE
MW 400 TRAP MAIN:MENUNO=N1:GOSUB USEMENU:
   POSITION 26,21:? "_";LT$;:GOSUB ANYK:?
    KY$;
MI 410 IF KY>47 AND KY<58 THEN PAGENO=KY-
   N48:TRAP GKEY:POSITION N0,N0:GOSUB USE
   PAGE:GOTO N50
VX 420 GOTO 400+KY
ML 427 ? DN$:DOS
KQ 467 GOTO 1000
```

```
LP 468 GOTO 1200
LF 472 MENUNO=N3:GOSUB USEMENU:GOSUB ANYK
   :GOTO MAIN
KR 473 GOTO 1100
WY 476 FUNC=1:MENUNO=N2:GOSUB 800:GOTO 60
   0
QC 481 POKE N752,N0:? CHR$(125):? :? :? :
   ? "GOTO MAIN";UP$;UP$;UP$:STOP
UW 483 FUNC=0:MENUNO=N2:GOSUB 800:GOTO 50
   0
EA 485 FUNC=N2:MENUNO=N2:GOSUB 800:GOTO 5
   50
LY 486 GOTO 1300
GL 499 REM SAVE PAGE$ TO DISK
MP 500 TRAP 530
GZ 510 CLOSE #N1:OPEN #N1,8,N0,FILENAME$:
   A=USR(ADR(SQUEEZE$),N1,ADR(PAGE$),LEN(
   PAGE$),N0)
BX 520 CLOSE #N1:IF A=N1 THEN GOSUB BEEP:
   GOTO MAIN
BR 530 POSITION N3,N23:GOSUB N1500:GOSUB
   ANYK:GOSUB 800:GOTO 500
MZ 550 TRAP 530
NU 560 CLOSE #N1:OPEN #N1,N8,N0,FILENAME$
   :? #N1;PAGE$;
PP 570 CLOSE #N1:GOSUB BEEP:GOTO MAIN
GY 599 REM LOAD PAGE$ FROM DISK
NA 600 TRAP 630
LH 610 CLOSE #N1:OPEN #N1,N4,N0,FILENAME$
   :A=USR(ADR(SQUEEZE$),N1,ADR(PAGE$),LEN
   (PAGE$),N1):POKE 195,A
DB 620 CLOSE #N1:IF A=N1 OR A=N3 THEN GOS
   UB BEEP:GOTO MAIN
DZ 630 POSITION N3,N23:GOSUB N1500:GOSUB
   ANYK:GOSUB 800:GOTO 600
TX 699 REM BORDER
FR 700 GOSUB RCH:POKE N752,N1:SCREEN$(N1,
   N1)=CHR$(TL):COLOR H:PLOT N1,N0:DRAWTO
    38,N0
KC 710 SCREEN$(N40,N40)=CHR$(TR):COLOR V:
   PLOT 39,1:DRAWTO 39,22:SCREEN$(N960,N9
   60)=CHR$(BR):COLOR H:PLOT 38,23
OF 720 DRAWTO N0,N23:SCREEN$(921,921)=CHR
   $(BL):COLOR V:PLOT N0,22:DRAWTO N0,N1
SB 730 POKE 752,C:GOSUB RCH:? RT$;:GOTO C
   LKEY
JW 799 REM GET FILENAME ROUTINE
QN 800 MENUNO=N2:GOSUB USEMENU:POSITION N
   12+N1,N3:IF FUNC=N1 THEN ? "LOAD FROM
   DISK":GOTO 830
PB 810 IF FUNC=N0 THEN ? " SAVE TO DISK "
AV 820 IF FUNC=N2 THEN POSITION N7,N3:? "
   UNCOMPRESSED SAVE TO DISK"
VC 830 POSITION 18,N6:GOSUB 310:FILENAME$
   =WORK$
RB 840 IF FILENAME$="" THEN GOTO MAIN
XQ 850 IF LEN(FILENAME$)=N1 THEN GOSUB 14
   00:GOTO 800
ZQ 860 RETURN
GE 899 REM GET NUMERIC CHARACTER
MM 900 GOSUB ANYK:IF KY=N155 THEN GOTO MA
   IN
SJ 910 IF KY<N48 OR KY>57 OR KY=SOU THEN
   ? KY$;CHR$(253);CHR$(126);:GOTO 900
FG 920 KY=KY-N48:? KY$;:RETURN
SN 999 REM COPY SCREEN ROUTINE
ZO 1000 MENUNO=N5:GOSUB USEMENU:POSITION
   N3,N9:? "Enter SOURCE screen      :_";
   LT$;:GOSUB 900
WZ 1010 SOU=KY
IZ 1020 POSITION N3,11:? "Enter DESTINATI
   ON screen :_";LT$;:GOSUB 900
QL 1030 DES=KY
ZU 1040 PAGE$(N1+DES*N960,N960+DES*N960)=
   PAGE$(N1+SOU*N960,N960+SOU*N960):GOSUB
    BEEP:GOTO MAIN
SB 1099 REM INSERT SCREEN ROUTINE
NQ 1100 MENUNO=N6:GOSUB USEMENU:POSITION
   N10,11:GOSUB 900
ZJ 1110 SOU=KY:DES=8:PNO=N1+8*N960
MI 1120 IF DES>=SOU THEN PAGE$(PNO+N960,P
   NO+1919)=PAGE$(PNO,PNO+959):DES=DES-N1
   :PNO=PNO-N960:GOTO 1120
HA 1130 ? CHR$(125):PAGENO=SOU:GOSUB CAPP
   AGE:GOTO MAIN
BP 1199 REM DELETE SCREEN ROUTINE
NV 1200 MENUNO=7:GOSUB USEMENU:POSITION 2
   7,N10:GOSUB 900
UW 1210 SOU=KY:DES=SOU:PNO=N1+SOU*N960
HC 1220 IF DES<N9 THEN PAGE$(PNO,PNO+959)
   =PAGE$(PNO+N960,PNO+1919):DES=DES+N1:P
   NO=PNO+N960:GOTO 1220
JX 1230 ? CHR$(125):PAGENO=N9:GOSUB CAPPA
   GE:GOTO MAIN
FS 1299 REM VIEW SCREENS
HJ 1300 C=N1:FOR PAGENO=N0 TO N9:? CHR$(1
   25):POSITION 16,N4:? "SCREEN ";PAGENO:
   FOR WAIT=N1 TO N50:NEXT WAIT
CC 1310 GOSUB USEPAGE:GOSUB ANYK:IF KY=N1
   55 THEN POP :GOTO MAIN
YK 1320 NEXT PAGENO:C=N0:GOTO MAIN
MV 1399 REM DISK DIRECTORY ROUTINE
ZX 1400 POKE N752,N1:MENUNO=N4:GOSUB USEM
   ENU:A$=CHR$(ASC(WORK$(N1,N1))+N128):PO
   SITION 28,N1:? A$;:TRAP 1470
RL 1410 FILENAME$="D1:*.*":FILENAME$(N2,N
   2)=WORK$
LY 1420 CLOSE #N1:OPEN #N1,N6,N0,FILENAME
   $:TRAP 1460:POKE N82,N2:I=N1:POSITION
   N2,N5
JI 1430 INPUT #N1,WORK$:? WORK$;CHR$(31);
   CHR$(31);:INPUT #N1,WORK$:? WORK$:I=I+
   N1:IF I<17 THEN GOTO 1430
HC 1440 POSITION N6,22:? "     MORE...Pres
   s any key     ";:GOSUB ANYK:GOSUB USEME
   NU:POSITION 28,N1:? A$;:I=N1
NN 1450 POSITION N2,N5:GOTO 1430
EL 1460 IF PEEK(195)=136 THEN POSITION N6
   ,22:? "Press any key to continue...";:
   GOSUB ANYK:RETURN
UC 1470 POSITION N2,22:GOSUB N1500:GOSUB
   ANYK:GOTO MAIN
LF 1499 REM ERROR MESSAGE
IP 1500 ? " ERROR ";PEEK(195);" AT LINE "
   ;PEEK(186)+256*PEEK(187);:RETURN
MC 9999 REM QUICK SCREEN SETUP
IJ 10000 N1=N0=N0:N2=N1+N1:N3=N2+N1:N4=N2
   +N2:N5=N3+N2:N6=N5+N1:N7=N6+N1:N10=N5+
   N5:N93=93:N94=N93+N1:N95=N94+N1
CB 10010 N100=N95+N5:N256=256:N960=960:N9
   600=N960*N10
UN 10020 VVTP=PEEK(134)+PEEK(135)*N256:PO
   KE VVTP,131
VL 10030 POKE VVTP+N2,PEEK(88):POKE VVTP+
   N3,PEEK(89)
QR 10040 POKE VVTP+N4,192:POKE VVTP+N5,N3
   :POKE VVTP+N6,192:POKE VVTP+N7,N3
XM 10050 USEMENU=150:RCH=120:MENUPAGES=N1
   0:MENUL=MENUPAGES*N960
AY 10060 DIM MENU$(MENUL),UP$(N1),DN$(N1)
   :MENU$=CHR$(0):MENU$(MENUL)=MENU$:MENU
   $(N2)=MENU$:UP$=CHR$(28):DN$=CHR$(29)
FV 10070 DIM SQUEEZE$(387)
TG 10079 REM LINES 10080 TO 10120 are cre
   ated by LISTING 2.  Type: ENTER "D:SQU
   EEZE.LST". Then type: LIST 10080,10120
TR 10129 REM  READ MENUS FROM DISK
NP 10130 POKE 752,1:POSITION N10+N1,N6:?
   "READING QSMENU.PGE"
CV 10140 CLOSE #N1:OPEN #N1,N4,N0,"D:QSME
   NU.PGE":A=USR(ADR(SQUEEZE$),N1,ADR(MEN
   U$),MENUL,N1)
BE 10150 CLOSE #N1:IF A=N1 OR A=N3 OR A=1
   36 THEN MENUNO=N0:GOSUB USEMENU:GOTO 1
   1000
KH 10160 ? "ERROR NUMBER: ";A;" AT LINE 1
   0220":STOP
```

```
RY 10999 REM INITIALIZATION
FM 11000 N8=N4+N4:N9=N10-N1:N12=N10+N2:N2
   3=N12+N10+N1
LV 11010 N32=N8*N4:N33=N32+N1:N48=N12*N4:
   N50=N48+N2:N40=N50-N10:N64=N32+N32:N82
   =82
NI 11020 N96=N95+N1:N127=N95+N32:N128=N12
   7+N1:N155=N100+N50+N5:N255=N155+N100
XV 11030 N320=N155*N2+N10:N752=752:N764=N
   752+N12:N300=N100*N3:N900=N300*N3:N910
   =N900+N10
NX 11040 N752=752:N764=N752+N12:N1500=150
   0
ZX 11050 SAVM5C=PEEK(88)+PEEK(89)*N256
KU 11060 DIM PAGE$(N9600):PAGE$=CHR$(0):P
   AGE$(N9600)=PAGE$:PAGE$(N2)=PAGE$
KW 11070 POKE 730,N1:POKE 729,24:POKE 709
   ,14:POKE 710,N0:POKE 712,N2
LE 11080 GKEY=70:ANYK=N300:CLKEY=110:CAPP
   AGE=130:USEPAGE=140:BEEP=160:OFF=40000
   :LM=N2:MAIN=400:BORDER=700
AI 11090 CLOSE #N2:OPEN #N2,N4,N0,"K:":PO
   KE N82,LM
CC 11100 DIM KY$(N1),FILENAME$(15),A$(N1)
   ,WORK$(N256),ERR$(36),LT$(N1),RT$(N1),
   CR$(N1)
XO 11110 LT$=CHR$(30):RT$=CHR$(31):CR$=CH
   R$(155)
VK 11120 UP$=CHR$(28):DN$=CHR$(29):LT$=CH
   R$(30):RT$=CHR$(31):CR$=CHR$(155)
VY 11130 GOSUB ANYK:GOTO MAIN
```

●

BOOT UP TO BIG SAVINGS SUBSCRIBE

☐ 1 Year.....$28..... Save $14
MCEYY

☐ 1 Year with Disk.....$105
DCEYY

Name

Address

City State Zip

Make checks payable to: L.F.P. Inc. Allow 4-6 weeks for delivery.

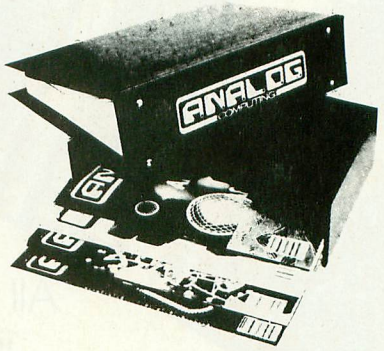
☐ Payment Enclosed ☐ Bill Me
☐ Charge My ☐ Visa ☐ MC
_____ Exp ____

Signature

FOREIGN—Add $7 per year
MONEY BACK if not delighted
Analog
P.O. Box 16927
N. Hollywood, CA 91615

ULTIMATE STORAGE

Here's the perfect way to organize your **ANALOG Computing** library—sturdy, custom-made binders and files in deep blue leatherette with embossed silver lettering. Silver labels are included to index by volume and year. One binder or a box-style file is all you'll need to accommodate 12 issues (1 year) of **ANALOG Computing**—all the games, programs, tutorials and utilities that you want handy.



The ANALOG Computing binder opens flat for easy reading and reference. They're economically priced at only $9.95 each—3 binders for $27.95 or 6 binders for $52.95.

The ANALOG Computing file is attractive and compact, holding 12 issues for easy access. Files are available for only $7.95 each—3 files for $21.95 or 6 files for $39.95.

Add $1.00 per case/binder for postage and handling.
Outside U.S., add $2.50 per case/binder — U.S. funds only.

I enclose my check or money order in the amount of $_____.

Send me: ____ **ANALOG Computing** files
____ **ANALOG Computing** binders.

PLEASE PRINT.

Name: _____

Address: (No P.O. Boxes) _____

City: _____

State: _____ Zip Code: _____

Send your order to:

Jesse Jones Industries

DEPT. ACOM, 499 East Erie Ave., Philadelphia, PA 19134
Call Toll Free 1-800-972-5858 — 7 days, 24 hours

Charge orders only, minimum $15.00
PA residents, add 6% sales tax

VISA MasterCard

Satisfaction guaranteed or money refunded.

Back Issues

All back issues are priced at $4.00 each.

Send your check or money order to
ANALOG Computing Back Issues,
P.O. Box 16927
N. Hollywood, CA 91615

Back issues on 5¼-inch disk
$12.95 each, plus $3.00 shipping and handling.
Issues 35 and up are available in this format.

ISSUE 30 • Loan Shark • Z-Plotter • BASIC Burger • **ANALOG TCS Guide** • Boulder Bombers

ISSUE 31 • Unicheck • R.O.T.O. • Lunar Patrol • ATASCII Animation • Lazer Type • Atari Clock • Personal Planning Calendar

ISSUE 32 • Supereversion • DOS III to DOS 2 conversion • Color the Shapes • Home-made Translator • Cosmic Defender • 520ST

ISSUE 33 • An Intro to MIDI • Note Master • Syntron • BASIC Bug Exterminator • Assemble Some Sound • C.COM • Mince (ST)

ISSUE 34 • Dragon's Breath • Multiple Choice Vocabulary Quiz • Elevator Repairman • Assemble Some Sound Part 2

ISSUE 35 *(also on disk)* • Hide and Seek • Printers Revisited • Bonk • Turtle 1020 • G:

ISSUE 36 *(also on disk)* • Sneak Attack • Maze War • Nightshade • Solid Gold Input Routine • Rafferty Run

ISSUE 37 *(also on disk)* • Speedski • Index to **ANALOG Computing** (15-36) • Master Disk Directory • Halley Hunter • Bank Switching for the 130XE

ISSUE 38 *(also on disk)* • Color Alignment Generator • Incoming! • DLI Maker • Air Hockey • ST Color Palette

ISSUE 39 *(also on disk)* • Super Pong • Unicheck (updated) • C-Manship Part 1 • Program Helper • Adventurous Programming Part 1 • ST Software Guide

ISSUE 40 *(also on disk)* • Clash of Kings • Micro-Mail • Koala Slideshow Program • Adventurous Programming Part 2 • Mouser

ISSUE 44 • RAMcopy! • The 8-Bit Parallel Interface • Arm your Atari • Blast! • D:CHECK in Action! • ST-Log 4

ISSUE 45 • Stencil Graphics • Roll 'Em! • RAM DOS XL • LBASIC • Using BASIC XL's Hidden Memory • ST-Log 5

ISSUE 46 • Magic Spell • Moonlord • Soft Touch • La Machine • June CES • Launch Code • ST-Log 6

ISSUE 47 • DLIs: A minute to learn • Deathzone • BASIC Editor II • • The ANALOG Database • DiskFile • ST-Log 7

ISSUE 48 • M-Windows • Cosmic Glob • DLIs - Part 2 • Modem Chess • Status Report • ST-Log 8

ISSUE 49 • The Atari 8-bit Gift Guide • Brickworks • TechPop • Fortune-Wheel • Smiles and other facial wrinkles • ST-Log 9

ISSUE 50 • Krazy Katerpillars • Atari Picture Storage Techniques • Trails in Action! • Scroll-It • Screen Scroller

Issues 12, 14, 15, 16, 17, 18, 19, 20, 21 and 22 are also still available.

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS™
ANALOG COMPUTING

The MAC/65 De-Tokenizer

Convert your tokenized MAC/65 files into plain ol' English

by Charles Bachand

MAC/65 is an outstanding 6502 macro assembler for the Atari 8-bit computer from Optimized Systems Software, Inc. Our staff, and most of our regular contributors, use it whenever they do any assembly language programming. The cartridge contains one of the fastest assemblers on the market today, and the tokenized source-code files that it generates can save quite a bit of disk space. We highly recommend MAC/65 to anyone doing 6502 machine language software development.

However, (you knew that this was coming, didn't you?) the Catch-22 with MAC/65 is that the file format used to save code is not compatible with anything; if you want to use MAC/65 files, you need MAC/65. Or, at least until now, you did. **The MAC/65 De-Tokenizer** presented here will free you from this requirement.

The De-Tokenizer is a small Atari BASIC program that will convert a tokenized MAC/65 file back into readable English—or at least what passes for English from some of the programmers I know.

How *The MAC/65 De-Tokenizer* works

To understand the internal workings of **The De-Tokenizer** program, one must understand the rules by which MAC/65 compresses its files. There are actually two parts to a MAC/65 tokenized file. The first four bytes comprise the file header, which is broken into two parts.

```
Byte  1  2  3  4
      -- -- -- --
      FE FE xx xx ...
      \___/ \___/
        |     |
        |     Byte count (low-byte/high-byte)
        MAC/65 file identifier bytes
```

Bytes 1 and 2 each have the value of 254 (or $FE in hex notation.) If, when you try to load a file, MAC/65 sees something other than a 254 in these positions, it knows that the file is the wrong format and the editor will display a *File Type Error #23.*

Now, assuming that the load process gets you past this check, MAC/65 reads in the next 2 bytes (3 and 4), which specify the number of bytes of actual tokenized data that the file contains. This is a 16-bit value which is stored in the classic "low-byte/high-byte" format. Adding 4—the number of bytes we've just read—to this number, will give us the total number of bytes in the file.

Once we get past these first 4 bytes, we begin to encounter the tokenized data—the place where all the real fun begins.

Getting on the token express

Atari BASIC is tokenized. This means that the programmers of this language devised a scheme to compress the

data, in order to reduce memory usage and to increase the execution speed of its programs. A similar savings is also realized when storing tokenized programs out to cassette or disk.

This concept of file tokenization has been carried over into the design of OSS's MAC/65 as illustrated in the following example:

Assembly listing		Hex data
10 LOOP	LDA $0600	0A 00 0C 84 4C 4F 4F 50 51 05 00 06
20	CLC	14 00 04 2C
30	ADC -10	1E 00 07 4F 3E 08 0A
40	STA $0600	28 00 07 50 05 00 06
50	JMP LOOP	32 00 09 21 84 4C 4F 4F 50

What do the five lines of assembly code and the five lines of hex data in the above example have in common? They happen to be one and the same as far as MAC/65 is concerned! The listing takes 78 bytes as text, but only 39 bytes to store it in the computer's memory or out to a disk file. In this example alone, we've cut our storage requirements in half, and this degree of savings is not an uncommon occurrence when using MAC/65.

One line at a time

Just as every MAC/65 file has a file header, so does every tokenized line in the file have a 3-byte line header associated with it. For instance, the first three bytes in Line 10 from our example above are $0A, $00 and $0C in hex. This is further broken down as follows:

```
0A 00 0C 84 4C 4F 4F 50 51 05 00 06
\_____/  |  \_____Token data_____/
   |     Line length
Line number (low-byte/high-byte)
```

The first two bytes represent the line number—again in the standard low-byte/high-byte format. It's easy enough to convert these two bytes back to a number that can be printed by multiplying the value of the second byte by 256 and adding the value of the first byte to it. Using our example, we have $00 (or 0 decimal), multiplied by 256, giving us 0 (so it's not an exceptionally exciting example) and then adding $0A (or 10 decimal) to it, which finally gives us a line number of 10. Wasn't that easy?

The third byte in the line tells us the number of bytes of tokenized data associated with that line. This count also includes the two line number bytes and the count byte. Our example of $0C (12 decimal) in this position signifies that the line contains 9 bytes of data, plus the 3 bytes of header data, giving us a total of 12 bytes.

A token for your thoughts

Now that we're finally past all this header stuff, we get to look at what you've all come here to see—namely MAC/65 tokens.

An assembly source statement line is in the form:

[label] [(6502 instruction) or (assembly directive)] [comment]

A string of characters, such as a label or a string in quotes, is identified by a token made up of the string's character count, with the high bit set, followed by the ASCII values of the string. So, if we have a 4-character string like "TEST" it will have a token of 132 ($84) or 128+4, followed by the ASCII values of the four characters *T, E, S* and *T*. String tokens are always greater than 128 in value.

If the value of the token is less than 128, then it's either a MAC/65 assembler directive or a 6502 instruction. Table 1 shows the allowed token substitution values.

Token	Replacement				
0*	"ERROR –"	32	"JSR"	64	"TSX"
1	".IF"	33	"JMP"	65	"TXA"
2	".ELSE"	34	"DEC"	66	"TXS"
3	".ENDIF"	35	"INC"	67	"TYA"
4	".MACRO"	36	"LDX"	68	"BCC"
5	".ENDM"	37	"LDY"	69	"BCS"
6	".TITLE"	38	"STX"	70	"BEQ"
7*	" "	39	"STY"	71	"BMI"
8	".PAGE"	40	"CPX"	72	"BNE"
9	".WORD"	41	"CPY"	73	"BPL"
10	".ERROR"	42	"BIT"	74	"BVC"
11	".BYTE"	43	"BRK"	75	"BVS"
12	".SBYTE"	44	"CLC"	76	"ORA"
13	".DBYTE"	45	"CLD"	77	"AND"
14	".END"	46	"CLI"	78	"EOR"
15	".OPT"	47	"CLV"	79	"ADC"
16	".TAB"	48	"DEX"	80	"STA"
17	".INCLUDE"	49	"DEY"	81	"LDA"
18	".DS"	50	"INX"	82	"CMP"
19	".ORG"	51	"INY"	83	"SBC"
20	".EQU"	52	"NOP"	84	"ASL"
21	"BRA"	53	"PHA"	85	"ROL"
22	"TRB"	54	"PHP"	86	"LSR"
23	"TSB"	55	"PLA"	87	"ROR"
24	".FLOAT"	56	"PLP"	88*	Comment line
25	".CBYTE"	57	"RTI"	89	"STZ"
26	";"	58	"RTS"	90	"DEA"
27	".LOCAL"	59	"SEC"	91	"INA"
28	".SET"	60	"SED"	92	"PHX"
29	"*="	61	"SEI"	93	"PHY"
30	"="	62	"TAX"	94	"PLX"
31	":="	63	"TAY"	95	"PLY"

(* = see text)

Table 1

Looks simple, doesn't it? If a token value is 33, then it is a 6502 "JMP" instruction. If the value is 17, it's a MAC/65 ".INCLUDE" directive.

Of course, there are exceptions. A line that could not be tokenized properly, due to an error in syntax, has a 0 as its token byte. A comment line—one that does not contain any label or instructions—has a token value of 88. The rest of the line in both cases is stored as pure ASCII text.

If the token has a value of 7, then the label following it identifies the name of a macro that is to be called at this point in an assembly. Macro names are usually indented more than the regular mnemonics, so we output a few extra spaces before printing the macro name.

Now, once we get past the initial label and instruction/directive phase of the tokenization process, we encounter a second set of tokens for the MAC/65 operands which are listed in Table 2. These are the ones that we'll be using until we reach the end of the tokenized line.

If we bump into a token that is greater than 128, it is a label and it follows the same rules we outlined previously in handling labels. All the other operand tokens—those with values less than 128—are handled with a simple substitution for their corresponding text values. The exceptions are for the tokens with values of 5, 6, 7, 8, 10 and 59.

Token	Replacement				
5*	"$"	28	">"	56	"Y"
6*	"$"	29	"<"	57	",X"
7*		30	"_"	58	")"
8*		31	"["	59*	" "
10*	" ' "	32	"]"	61	" , "
11	"%$"	36	"!"	62	"#"
12	"%"	37	"^"	63	"A"
13	"*"	39	"\"	64	"("
18	"+"	47	".REF"	65	" "
19	"-"	48	".DEF"	69	"NO"
20	"*"	49	".NOT"	70	"OBJ"
21	"/"	50	".AND"	71	"ERR"
22	"&"	51	".OR"	72	"EJECT"
24	"="	52	"<"	73	"LIST"
25	"<="	53	">"	74	"XREF"
26	">="	54	",X)"	75	"MLIST"
27	"<>"	55	"),Y"	76	"CLIST"
				77	"NUM"

(* = see text)

Table 2

Token 5 is used for word-length hex constants and is followed by two bytes containing the value of the constant in low-byte/high-byte format. For example, a hex constant like $0600 is tokenized as the three bytes 5, 0 and 6.

Token 6 is very similar to 5, but is used for 1-byte-long hex constants. A value like $80 would tokenize as the two bytes 6 and 128.

Tokens 7 and 8 are used for word and byte decimal constants and follow the same structure as the hex constant tokens 5 and 6. Thus, a decimal word constant like 256 stores as 7, 0, 1, while a byte-sized constant like 255 ends up as 8, 255.

Token 10 defines a character constant, which in MAC/65 is an apostrophe, followed by any displayable character. Thus, a reference to the ATASCII value of the capital letter A would appear in MAC/65 as 'A and be stored as 10, 65.

That just leaves us with the value of 59, which signifies the beginning of the comment field, to round off our oddball token list. All the bytes that follow the token, up to the end of the line, are part of the comment field and are output as ATASCII text.

How the *MAC/65 De-Tokenizer* works—really!

Sorry about suckering you into reading about MAC/65 tokenization methods with that false "How the program works" title near the beginning of the article, but you might have skipped over some wonderful pieces of information. I promise, the following paragraphs do in fact describe how the program works. Honest!

The De-Tokenizer is a bare-bones utility program. All you need do to make it run is supply it with the names of the MAC/65 file and the text file or device that you want the output sent to. If you're not sure of the name of the MAC/65 file, just hit RETURN and the program will list the directory of the disk in drive 1.

The program takes several seconds to initialize, and in that time, it reads a list of strings, representing the MAC/65 tokens, from DATA statements into the array TOKEN$. It also reads the corresponding decimal values of the tokens, which are used as indexes in building two arrays that hold the indexes for the starting (Array TS) and ending locations (Array TE) of each substring in the TOKEN$ array. Once initialized, it's a simple matter to print out the ATASCII value of any known token.

For example, to print out the text for token 11—the ".BYTE" directive—we could type the following line:

```
PRINT TOKEN$(TS(11),TE(11))
```

This, of course, will give us only one possible string, but it is easily modified by replacing the two numeric constants with variables. This was done in Line 1320 of the BASIC program to print out the token associated with the contents of the numeric variable A.

You might have noticed in Line 1390 that the routine used above to print out a token has a value of 128 added to the indexes, and, of course, there must be a very good reason for this. If you take a look at Tables 1 and 2, you'll see that quite a few of the token values used are common to both tables. MAC/65 doesn't mind this at all, since internally it uses two tables for doing substitutions. **The De-Tokenizer** program could have also used two sets of tables, but it was more to our advantage to employ only one.

If you remember from my little MAC/65 tutorial (the one I tricked you into reading), the token for a label is 128, plus the number of characters in the label, leaving all other tokens with a value less than 128. Hmm. . . If we leave the numbering of the instruction/directive tokens (Table 1) alone, and offset the numbering of the operand tokens (Table 2) by adding 128 to them, then they'll all fit into a 256-entry table. Yeah, yeah, that's the ticket! Then, to print an operand, we look at the second half of the look-up table (tokens 128-255), by adding 128 to the token value—just like the one being done in Line 1390.

Taking up space with source code

I wanted the output from **The MAC/65 De-Tokenizer** to be usable by owners of Atari's Assembler/Editor cartridge, so I elected not to try to convert the directives to those used by other 8-bit assemblers (like SynAssembler or Atari's Macro Assembler). It's up to the programmer to check for any incompatibility in this area.

The program does not output text in nice neat columns (remember, I told you this was a bare-bones program!), because it is largely a waste of precious disk space to do so. Most assembler programs won't care anyway, but if it bothers you that much, I'll be happy to let you modify my code.

Well, that's about it. I hope you're able to make as much use out of **The MAC/65 De-Tokenizer** as I had fun writing it (intense sarcasm here).

Charlie would like to thank Matthew J.W. Ratcliff, a heavy-duty proponent of MAC/65, for his aid in the testing of this program.

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

(Listing starts on next page)

Listing 1
BASIC listing

```
TQ 1000 REM MAC/65 TOKEN CONVERTER
LE 1010 REM (C) 1987 ANALOG COMPUTING
WL 1020 REM WRITTEN BY CHARLES BACHAND
IF 1030 REM
GC 1040 DIM T$(255),TE(255),HEX(15)
IQ 1050 DIM A$(40),TOKEN$(500)
TE 1060 GOSUB 1470:GOSUB 1660
OT 1070 TRAP 1070:? :? "RETURN for direct
   ory or name of":? " MAC/65 file";
PO 1080 INPUT A$:IF A$="" THEN GOSUB 1740
   :GOTO 1070
EL 1090 OPEN #1,4,0,A$
KA 1100 GET #1,A:GET #1,B
HZ 1110 IF A=254 AND A=B THEN 1130
BD 1120 ? "Not a MAC/65 File!":GOTO 1070
KJ 1130 GET #1,A:GET #1,B
PQ 1140 ? :? "File length = ";A+B*256+4
WT 1150 TRAP 1150:? :? "RETURN for direct
   ory or name of":? " OUTPUT file";
NO 1160 INPUT A$:IF A$="" THEN GOSUB 1740
   :GOTO 1150
GX 1170 OPEN #2,8,0,A$
IW 1180 REM
BJ 1190 REM PROCESS A LINE
IA 1200 REM
OZ 1210 TRAP 1440
YM 1220 GET #1,A:GET #1,B:GET #1,L
LZ 1230 L=L-3:PRINT #2;A+B*256;" ";
SO 1240 GET #1,A:L=L-1:IF A<>88 THEN 1270
WN 1250 FOR B=1 TO L:GET #1,A:PUT #2,A
MV 1260 NEXT B:PRINT #2:GOTO 1220
MS 1270 IF A<128 THEN 1320
BR 1280 FOR I=129 TO A:GET #1,A
NZ 1290 PUT #2,A:L=L-1:NEXT I
OJ 1300 IF L=0 THEN ? #2:GOTO 1220
XE 1310 GET #1,A:L=L-1
FX 1320 ? #2;" ";TOKEN$(T$(A),TE(A));" ";
XI 1330 IF A=0 THEN 1250
OV 1340 IF L=0 THEN ? #2:GOTO 1220
XQ 1350 GET #1,A:L=L-1
HB 1360 IF A>128 THEN C=A:FOR B=129 TO C:
   GET #1,A:PUT #2,A:L=L-1:NEXT B:GOTO 13
   40
BG 1370 IF A=7 THEN GET #1,A:GET #1,B:? #
   2;A+B*256;:L=L-2:GOTO 1340
TL 1380 IF A=8 THEN GET #1,A:? #2;A;:L=L-
   1:GOTO 1340
KD 1390 ? #2;TOKEN$(T$(A+128),TE(A+128));
   :IF A=6 THEN GET #1,A:GOSUB 1680:L=L-1
   :GOTO 1340
XP 1400 IF A=5 THEN GET #1,B:GET #1,A:L=L
   -2:GOSUB 1680:A=B:GOSUB 1680:GOTO 1340
MH 1410 IF A=59 THEN FOR B=1 TO L:GET #1,
   A:PUT #2,A:NEXT B:? #2:GOTO 1220
DI 1420 IF A=10 THEN GET #1,A:PUT #2,A:L=
   L-1
QP 1430 GOTO 1340
LL 1440 A=PEEK(195):IF A=136 THEN 1070
MY 1450 ? "ERROR #";A;" AT LINE ";PEEK(1
   86)+PEEK(187)*256:STOP
IW 1460 REM
IL 1470 REM SET-UP TOKEN TABLES
JC 1480 REM
PY 1490 ? :? " MAC/65 TOKEN CONVERTER"
KO 1500 ? :? "Initializing arrays, ";
IL 1510 ? "please wait...":C=1
MF 1520 READ TOKEN,A$:IF TOKEN<>-1 THEN G
   OSUB 1600:GOTO 1520
JN 1530 TOKEN=185:A$=",X":GOSUB 1600
JX 1540 TOKEN=184:A$=",Y":GOSUB 1600
GC 1550 TOKEN=7:A$=" ":GOSUB 1600
RT 1560 TOKEN=189:A$=",":GOSUB 1600
QX 1570 TOKEN=193:A$=CHR$(34):GOSUB 1600
SJ 1580 TOKEN=182:A$=",X)":GOSUB 1600
MD 1590 TOKEN=183:A$="),Y"
GQ 1600 T$(TOKEN)=C:TOKEN$(C)=A$
MM 1610 TE(TOKEN)=LEN(TOKEN$)
QM 1620 C=LEN(TOKEN$)+1:RETURN
IR 1630 REM
JP 1640 REM HEX CONVERSION
IX 1650 REM
SD 1660 FOR A=0 TO 15:READ A$
VG 1670 HEX(A)=ASC(A$):NEXT A:RETURN
WL 1680 C=INT(A/16):B=A-C*16
CZ 1690 PUT #2,HEX(C):PUT #2,HEX(B)
AN 1700 RETURN
IN 1710 REM
FS 1720 REM READ DISK DIRECTORY
IT 1730 REM
UC 1740 OPEN #3,6,0,"D:*.*":TRAP 1755
QV 1750 INPUT #3,A$:? A$:GOTO 1750
FY 1755 CLOSE #3:RETURN
JC 1760 REM
WC 1770 REM TOKEN TABLE
JI 1780 REM
HP 1790 DATA 79,ADC,77,AND,84,ASL,68,BCC
OG 1800 DATA 69,BCS,70,BEQ,71,BMI,72,BNE
EL 1810 DATA 73,BPL,74,BVC,75,BVS,42,BIT
KT 1820 DATA 43,BRK,44,CLC,45,CLD,46,CLI
WS 1830 DATA 47,CLV,82,CMP,40,CPX,41,CPY
LL 1840 DATA 34,DEC,48,DEX,49,DEY,78,EOR
YQ 1850 DATA 35,INC,50,INX,51,INY,33,JMP
OO 1860 DATA 32,JSR,81,LDA,36,LDX,37,LDY
MW 1870 DATA 86,LSR,52,NOP,76,ORA,53,PHA
LK 1880 DATA 54,PHP,55,PLA,56,PLP,85,ROL
TP 1890 DATA 87,ROR,83,SBC,59,SEC,60,SED
HY 1900 DATA 61,SEI,80,STA,38,STX,39,STY
SY 1910 DATA 62,TAX,63,TAY,64,TSX,65,TXA
KT 1920 DATA 66,TXS,67,TYA,57,RTI,58,RTS
LC 1930 DATA 21,BRA,90,DEA,91,INA,92,PHX
IR 1940 DATA 93,PHY,94,PLX,95,PLY,89,STZ
SU 1950 DATA 22,TRB,23,TSB,29,*=
QR 1960 DATA 14,.END,26,;,19,.ORG,30,=
IV 1970 DATA 20,.EQU,11,.BYTE,12,.SBYTE
TL 1980 DATA 25,.CBYTE,13,.DBYTE,9,.WORD
HG 1990 DATA 18,.DS,2,.ELSE,3,.ENDIF
PJ 2000 DATA 10,.ERROR,24,.FLOAT
CS 2010 DATA 1,.IF,17,.INCLUDE,27,.LOCAL
IV 2020 DATA 15,.OPT,8,.PAGE,28,.SET
AU 2030 DATA 0,ERROR - ,4,.MACRO,5,.ENDM
YP 2040 DATA 6,.TITLE,31,.=,16,.TAB
YK 2050 DATA 190,#,187, ,134,$,133,$
UG 2060 DATA 180, (,181, ),138,',159,[
PZ 2070 DATA 160,],146,+,149,/,148,*
GN 2080 DATA 167,\,150,&,164,!,165,^
FU 2090 DATA 152,=,156,>,157,<,158,-
NJ 2100 DATA 147,-,155,<>,154,>=,153,<=
WT 2110 DATA 179, .OR ,197,NO ,201
XA 2120 DATA LIST,178, .AND ,199
HO 2130 DATA ERR,177, .NOT ,200
YO 2140 DATA EJECT,176, .DEF ,198
YN 2150 DATA OBJ,175, .REF ,203,MLIST
BR 2160 DATA 204,CLIST,205,NUM,202,XREF
SX 2170 DATA 192,(,186,),139,%$,141,*
AY 2180 DATA 191,A,140,%,-1,XXX
CM 2190 DATA 0,1,2,3,4,5,6,7,8,9
KR 2200 DATA A,B,C,D,E,F
EV 2210 END
```

MICROMOD Turbobase

MICROMISER SOFTWARE
1635-A Holden Avenue
Orlando, FL 32809
1-800-451-4944
48K Disk $159.95

by Steve Panak

It's not often that companies send me their software, just begging to have it evaluated—especially small companies. (Perhaps I have somewhat of a bad reputation. I prefer to think I'm simply a very demanding consumer.) But such is the case here. What's more, the package came with a challenge: It demanded to be evaluated, not against other Atari software, but against PC/MS/DOS software. Well, so be it.

Surprisingly enough, **MICROMOD Turbobase** stacks up pretty well against its competition. It proclaims to be not just a database, but a fully integrated software package. It further boasts the largest and most efficient storage capacity of any Atari program, thus removing the need for disk swapping during normal operation (although you will have to switch disks when moving to a different module).

To accomplish these miracles, however, the program is very difficult to use. So difficult to use, in fact, that I found it a nearly insurmountable burden just to test its most basic features—and I am no novice. I use a number of PC/MS/DOS applications, and have never seen a program this hard to learn. It is closer in its degree of difficulty to learning a new language. But test it I did, mainly by stepping through the program's tutorial.

At the core of the program is a powerful database. What really threw me is that it seems to be structured—at least as far as the user is concerned—much differently than other DBs I've used. (Other DBs are menu driven to a much greater extent.) In **MICROMOD**, there are no simple prompts requesting the user to initialize new fields, no screens in which to enter new records.

But once I started in, I found the program stored information quite efficiently. Upon booting the package, you find yourself at the main menu, where you may begin entering records. The most important file is the NAMES file. It contains the names, addresses and phone numbers of all your contacts. Each is identified by a unique keyword, or abbreviated name, typically four letters in length. To enter these records, simply type *PUT*, then the keyword. At the succeeding prompts you'll enter the full name, address and phone number.

Once this file is complete, its information is used in the many other files of the program, saving storage space and automatically filling in information. Invoicing, inventory and payroll functions are all supported in the "Dated Records" program, which is basically a standard database manager. As you produce invoices, item descriptions can be automatically printed from other files, such as inventory. The payroll function will support all standard deductions, as well as print the checks for you.

Numerous reporting features are included, so that you can view your information in any manner you wish. Four simple preset formats are available, and you can create and store additional report formats containing any of the fields in your database. A general ledger feature keeps track of your financial data, and a simple word processor helps you get your message across.

The program is supplied on three double-sided disks, which are not copy protected. After about 30 days, you will be prompted to enter an authorization code, which can be obtained from MicroMiser. The program continues to work for about thirty days after you supply this prompt, so no lapse in service should occur.

For those interested in statistics, the program capacities are quite large, considering they operate on these machines. Depending on whether you have a single- or double-sided disk drive, you can store 5,000-10,000 general ledger entries, 1,500-3,000 addresses, 700-1,400 invoices, and up to 4,000 inventory items. While this might not be sufficient capacity for General Motors, it will satisfy most small business needs.

The manual is the largest I've ever seen for any Atari program. In fact, it's one of the largest manuals I've ever seen for *any* software. At over 400 pages—and weighing a couple of pounds—it is a lot to read through and lug around. Fortunately, a "cookbook" is provided to help the new user become familiar with the main features and commands. Indexing is not too bad, although it is still rather hard to find

the answers to most questions without reading quite a bit. On-line help is nonexistent, although the on-screen menus and prompts are sometimes useful. Customer support is good—only a phone call away—and updates crop up more often than any other program on the market.

The problem here is not the software, but the hardware it's designed to run on. Currently, PCs are the fastest and most popular business computers. I would find it very hard indeed to recommend that a small business use an Atari over a PC. The Atari just doesn't have a lot of memory, and expanding the memory on one is hard—and expensive.

The Atari is a great *home* computer. It plays games very well, and offers rudimentary word processing and database functions. But this program is simply too complex to store recipes or a Christmas card mailing list. Of course, if you're thinking of running a business through an Atari, and desire a powerful database program, **MICROMOD** is your best choice; and if you want an integrated package, it's your only choice. Just be prepared to spend a lot of time learning to use it.

Steve Panak is a Trust Attorney and a free-lance writer living in northeastern Ohio. He holds a B.S. in B.A. and a J.D. He currently oversees computer operations in his department, where he develops software to teach complex legal concepts. In his spare time, he enjoys computer games.

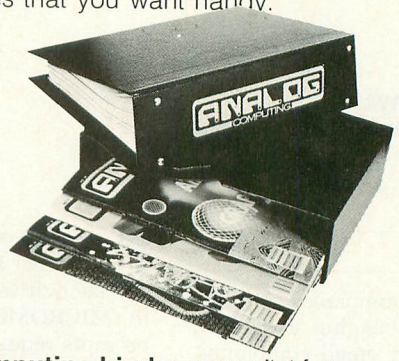
We Want Letters

The fate of *Video Game Digest*, **ANALOG Computing**'s magazine-within-a-magazine, is in your hands. If you want a monthly video game magazine, with reviews, news and insights into the fast-changing world of cartridge games, write and tell us so.

We'd like to hear from readers on every aspect of video gaming, and want to know what you'd like to see in *Video Game Digest*.

ULTIMATE STORAGE

Here's the perfect way to organize your **ANALOG Computing** library—sturdy, custom-made binders and files in deep blue leatherette with embossed silver lettering. Silver labels are included to index by volume and year. One binder or a box-style file is all you'll need to accommodate 12 issues (1 year) of **ANALOG Computing**—all the games, programs, tutorials and utilities that you want handy.



The ANALOG Computing binder opens flat for easy reading and reference. They're economically priced at only $9.95 each—3 binders for $27.95 or 6 binders for $52.95.

The ANALOG Computing file is attractive and compact, holding 12 issues for easy access. Files are available for only $7.95 each—3 files for $21.95 or 6 files for $39.95.

Add $1.00 per case/binder for postage and handling.
Outside U.S., add $2.50 per case/binder — U.S. funds only.

I enclose my check or money order in the amount of $_____.

Send me: ____ **ANALOG Computing** files
____ **ANALOG Computing** binders.

PLEASE PRINT.

Name: _____

Address: (No P.O. Boxes) _____

City: _____

State: _____ Zip Code: _____

Send your order to:

Jesse Jones Industries
DEPT. ACOM, 499 East Erie Ave., Philadelphia, PA 19134
Call Toll Free 1-800-972-5858 — 7 days, 24 hours

Charge orders only, minimum $15.00
PA residents, add 6% sales tax
Satisfaction guaranteed or money refunded.

VISA MasterCard

Breakers

by Rod Smith, Joe Vierra and William Mataga
BRODERBUND SOFTWARE, INC.
17 Paul Drive
San Rafael, CA 94903
All resolutions $44.95

by Steve Panak

If you've followed my rantings and ravings over the past few years, you probably know that my favorite type of game is interactive fiction. This could be due to my love of the written word, although others, less compassionate than myself, would attribute this affection (or is it an affliction?) to some depraved side of my nature. Regardless of the cause, I still love getting into the story.

Breakers from Broderbund is just the game I've been waiting for. A new entry into the ever-growing interactive fiction market, this game also defines new standards for ease of use. Although program design is a major consideration in any piece of software, the most important aspect of these games is the story. And in this department, **Breakers** doesn't disappoint.

The term ***Breakers*** refers to the galactic outcasts inhabiting the Nimbus Colony and the planet Borg below. These are the riffraff, criminals whose major occupation is smuggling. Their wares: slaves and narcotics. Although Borg is a mining planet, little ore has left the planet in years. The two leaders of the **Breakers**—both former Federation agents—have had a falling out, and are struggling for control of the operation. Rumor has it that another Federation agent has appeared on the scene. The main question is whether you feel up to the task ahead.

From this point, as in all works of interactive fiction, you're in control. **Breakers** features an advanced parser that understands over 1200 words, in both simple and complex sentences. You can easily converse with other characters, as well as move through rooms and examine objects. The **Breakers** universe unfolds in front of you, with events happening regardless of how you react. Special commands allow you to control the speed at which time passes or to save your position in the novel (the BOOKMARK command).

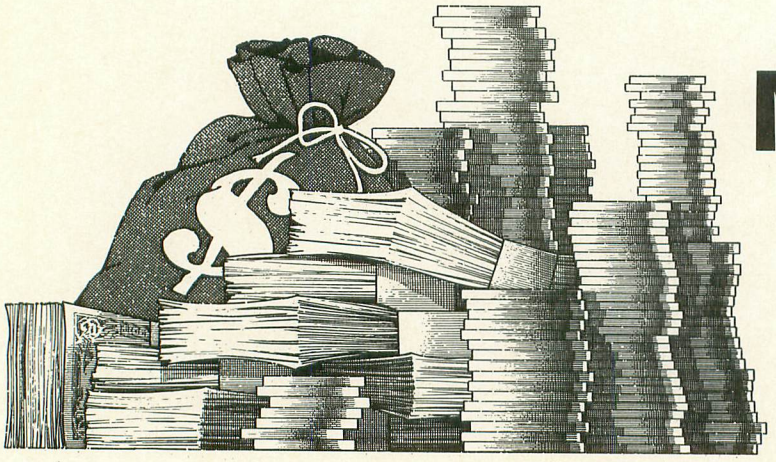
My favorite feature of this game is the fact that it doesn't take over the display. It leaves the desktop, with all its accessories intact, then adds a few of its own options. In playing text adventures, typically all your time is spent hunched over the keyboard typing in commands. In this game, however, pull-down menus free you from having to type and retype the most often used commands, allowing you to click on them instead. And you can easily add to these commands, and modify them throughout the game as your needs change. But, despite excellent program design, there is a major flaw.

Although I was able to issue commands rapidly using the various menus, all seemed for naught when I had to wait a lengthy amount of time for the response to most of my commands. The ST has over 500K, right? The disk, a max of 360? Then what's the problem? Most of the text should be able to reside in memory, allowing instant response. Aside from the frequent disk accesses, the program itself seemed to run slowly, sitting and thinking after each sentence input.

The documentation is good: a nice booklet, seventy pages in length, that provides background story, a partial list of commands, room for notes, and pages of nice illustrations. A separate reference card holds machine-specific instructions. The entire package was well thought out and put together. The prose rivals any similar game on the market, with vivid descriptions and lively characters.

Despite a few shortcomings, **Breakers** breaks new ground in software, especially in interactive fiction. Its story line is interesting, its characters engaging, and its puzzles challenging. Take a break from run-of-the-mill software; break away with **Breakers**.

Steve Panak is a Trust Attorney and a free-lance writer living in northeastern Ohio. He holds a B.S. and a J.D. He currently oversees computer operations in his department, where he develops software to teach complex legal concepts. In his spare time, he enjoys computer games.



Money Pouch

A coin counting game to help your children polish their mathematical skills

by Chuck Rosko

Money Pouch is a one-player educational game to help children become familiar with the concept of money. The game begins when Kiki, a baby kangaroo, is captured by the zookeeper. Kandy, Kiki's mother, must pay the zookeeper in order to free her baby. This is done by knocking out the correct combination of coins (pennies, nickles, dimes, quarters and half-dollars) to pay the zookeeper's fee.

Options

After booting up, in the upper left corner of the screen you'll see a 25 displayed. This is the greatest amount of money the zookeeper can ask for. If you choose 25, the zookeeper's price can vary from 1 cent to 25 cents. Changing this range can be accomplished by pressing the OPTION key. There are four different ranges to choose from.

You'll also see the word *EASY* displayed. This is the difficulty level, or how much time Kandy has to pay the fee. There are three difficulty levels, with *EASY* giving you the longest amount of time. Press SELECT to change the level and press the START key to begin playing.

Playing *Money Pouch*

Displayed at the top of the screen are your current score, the zookeeper's price and the bonus clock. Below these are five columns, each containing a different coin. Below the columns are five buckets, each labeled with the value of one of the coins. The order of the buckets is chosen randomly at the beginning of each round. Next to them is the waste bucket labeled 3. You have three chances to rescue Kiki. Underneath the waste bucket is Kandy's money total. You can use an unlimited number of coins.

Your goal is to free Kiki as quickly as possible. There's a bonus clock located in the upper right corner of the screen. This clock decreases in increments of 5 during each problem. The rate at which it decreases depends upon the difficulty level you've chosen. If the clock reaches zero, you'll lose one life. Once the zookeeper is paid, the points currently in the bonus clock are added to your score. Therefore, you'll earn more points if you use fewer coins and as little time as possible. The clock is reset to 50 for each new problem.

Your joystick moves Kandy left or right. To knock out a coin, push your joystick forward (north). The coin will then roll back and forth along the ramp above the buckets. (You can only knock out one coin at a time.) Now you must drop the coin into the bucket that matches the monetary value of that coin (i.e., nickle in "05").

Press the trigger button when the coin is over the correct bucket. If the correct coin is chosen, it will be added to Kandy's total. When Kandy's total *exactly* matches the zookeeper's fee, Kiki will be released. If you drop a coin in the wrong bucket, the bonus clock will decrease by 5 and the coin will pop back up so you can try again. Be careful—you'll lose a lot of time by trying buckets randomly.

If you don't want to use a coin that you've knocked out, drop it in the waste bucket. This way the coin will be ignored, and *not* added to your total. If your total exceeds the zookeeper's price, you'll lose one life.

Chuck Rosko is a microbiologist from Pittsburgh, Pennsylvania. His interests include his wife and son, hockey and writing educational programs.

(Listing starts on next page)

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

Listing 1
BASIC listing

```
WN 0 REM ********************************
ZA 1 REM *                              *
HK 2 REM *   MONEY POUCH (C) 1987       *
NS 3 REM *      BY CHUCK ROSKO          *
MD 4 REM *        FOR ANALOG            *
ZE 5 REM *                              *
WT 6 REM ********************************
ZJ 20 GRAPHICS 17:DIM XD(15),AA$(30),BB$(
   30),ZZ$(32),TT$(57),PR$(50),COL$(15),D
   EK(4)
JE 21 FOR X=5 TO 15:XD(X)=3:NEXT X:XD(7)=
   1:XD(11)=2:DEK(0)=1:DEK(1)=5:DEK(2)=10
   :DEK(3)=25:DEK(4)=50
UM 22 XD(14)=4:JJ=0:KK=1:LL=2:NN=3:XZ=200
   :PT=230:TP=29210:XYZ=235:ZL=10:Q=6:ST=
   16:PZA=210:POKE 559,JJ:GOTO 19000
UV 200 SOUND JJ,200,11,NN:GOSUB 246:RETUR
   N
LD 205 POSITION Q,20:? #6;"       ":POSIT
   ION Q,21:? #6;"       ":POSITION Q,22:
   ? #6;"       "
FN 206 POSITION Q,22:? #6;"       ":RETUR
   N
CB 210 SOUND JJ,Q,ZL,14:SOUND KK,8,ZL,14:
   GOSUB 246:RETURN
JG 215 FOR Z=14 TO JJ STEP -1:SOUND JJ,Z*
   13,ZL,Z:NEXT Z:RETURN
TC 220 FOR Z=7 TO 14:SOUND JJ,40*Z/NN,14-
   Z,14-Z:NEXT Z:RETURN
CG 230 A=USR(ADR(TT$),PAUSE):RETURN
QC 235 FOR D=KK TO ZL:SOUND JJ,ZL*D,ZL,ZL
   -D:NEXT D:RETURN
GC 240 TIME=LEVEL-INT((PEEK(20)+256*PEEK(
   19))/60):IF TIME<=JJ THEN 250
ZV 245 RETURN
VN 246 SOUND JJ,JJ,JJ,JJ:SOUND KK,JJ,JJ,J
   J:RETURN
HN 250 BNS=BNS-5:POSITION ST,KK:? #6;"  "
   :POSITION ST,KK:? #6;BNS:IF BNS=JJ THE
   N 4000
TJ 255 POKE 19,JJ:POKE 20,JJ:GOSUB XYZ:GO
   SUB XYZ:RETURN
JM 260 POKE 19,JJ:POKE 20,JJ
ZF 300 COR=INT(RND(JJ)*LIMIT)+KK:IF COR=1
   00 THEN POSITION 9,KK:? #6;"1.00":GOTO
    330
UW 310 POSITION 9,KK:? #6;"0":IF COR<ZL T
   HEN POSITION 11,KK:? #6;"0":POSITION 1
   2,KK:? #6;COR:GOTO 330
NC 320 POSITION 11,KK:? #6;COR
IP 330 X1=64:FOR KQ=1 TO 16:A=USR(MOVE,JJ
   ,PMB,BOD(JJ),X1,62-KQ,KQ):A=USR(MOVE,K
   K,PMB,TAIL(JJ),X1-8,62-KQ,KQ)
HK 332 NEXT KQ:Y1=46
FF 1000 J=STICK(JJ):GOSUB 240:GOSUB XZ
BD 1001 IF TIME=5 THEN POSITION Q,21:? #6
   ;"HELP "
HK 1002 IF TIME=NN THEN POSITION Q,21:? #
   6;"     "
AV 1030 PAUSE=ZL:ON XD(J) GOTO 1050,1100,
   1150,1200
LX 1050 X1=X1+12:IF X1>160 THEN X1=160:GO
   TO 1000
PM 1060 Y1=Y1-8:A=USR(MOVE,KK,PMB,TAIL(JJ
   ),X1-8,Y1,ST):A=USR(MOVE,JJ,PMB,BOD(JJ
   ),X1,Y1,ST):GOSUB 215
NF 1070 Y1=Y1+8:X1=X1+12:A=USR(MOVE,KK,PM
   B,TAIL(JJ),X1-8,Y1,ST):A=USR(MOVE,JJ,P
   MB,BOD(JJ),X1,Y1,ST):GOTO 1000
UB 1100 X1=X1-12:IF X1<64 THEN X1=64:GOTO
    1000
UJ 1110 Y1=Y1-8:A=USR(MOVE,KK,PMB,TAIL(KK
   ),X1+8,Y1,ST):A=USR(MOVE,JJ,PMB,BOD(KK
   ),X1,Y1,ST):GOSUB 215
UG 1120 Y1=Y1+8:X1=X1-12:A=USR(MOVE,KK,PM
   B,TAIL(KK),X1+8,Y1,ST):A=USR(MOVE,JJ,P
   MB,BOD(KK),X1,Y1,ST):GOTO 1000
PK 1150 A=USR(MOVE,KK,PMB,TAIL(JJ),X1-8,Y
   1,ST):A=USR(MOVE,JJ,PMB,BOD(JJ),X1,Y1,
   ST):GOTO 1000
LS 1200 POKE 706,72:IF X1=64 THEN COIN=KK
   :RESTORE 1291:GOTO 1230
DK 1205 POKE 706,14:IF X1=88 THEN COIN=5:
   RESTORE 1292:GOTO 1230
MI 1210 IF X1=112 THEN COIN=ZL:RESTORE 12
   93:GOTO 1230
FX 1215 IF X1=136 THEN COIN=25:RESTORE 12
   94:GOTO 1230
BG 1220 IF X1=160 THEN COIN=50:RESTORE 12
   95
YP 1230 FOR Y1=46 TO 26 STEP -4:A=USR(MOV
   E,KK,PMB,TAIL(JJ),X1-8,Y1,ST):A=USR(MO
   VE,JJ,PMB,BOD(JJ),X1,Y1,ST)
SF 1240 SOUND JJ,Y1*NN,ZL,4:NEXT Y1:GOSUB
    220:FOR Y1=26 TO 46 STEP 4:A=USR(MOVE
   ,KK,PMB,TAIL(JJ),X1-8,Y1,ST)
IC 1250 A=USR(MOVE,JJ,PMB,BOD(JJ),X1,Y1,S
   T):SOUND JJ,Y1*3,ZL,4:NEXT Y1:GOSUB 24
   6:Y1=46:X2=X1+8:C=JJ
DA 1260 FOR Y2=30 TO 63 STEP NN:A=USR(MOV
   E,LL,PMB,BALL(C),X2,Y2,Q):C=C+KK:IF C>
   NN THEN C=JJ
WX 1270 NEXT Y2:GOSUB PZA:Y2=63:PAUSE=5
DH 1280 READ X2,C:IF X2=-1 THEN GOSUB 210
   :GOTO 1300
ON 1290 A=USR(MOVE,2,PMB,BALL(C),X2,Y2,Q)
   :GOSUB PT:GOSUB XZ:GOTO 1280
RD 1291 DATA 80,1,88,2,96,3
KC 1292 DATA 104,0,112,1,120,2
XB 1293 DATA 128,3,136,0,144,1
ZU 1294 DATA 152,2,160,3,168,0
SB 1295 DATA 176,1,184,2,192,3,-1,-1
XS 1300 A=USR(MOVE,LL,PMB,BALL(NN),192,71
   ,Q):GOSUB PZA:RESTORE 2070
DX 2000 IF STRIG(JJ)=JJ THEN 2100
HZ 2020 IF X2=176 THEN POSITION Q,21:? #6
   ;"HURRY":A=USR(MOVE,NN,PMB,BAB,76,98,1
   1):GOTO 2040
CK 2030 IF X2=80 THEN POSITION Q,21:? #6;
   "     ":A=USR(MOVE,NN,PMB,BAB,60,98,11
   )
GW 2040 READ X2,C:IF X2=-1 THEN RESTORE 2
   070:GOTO 2040
BH 2060 A=USR(MOVE,LL,PMB,BALL(C),X2,71,Q
   ):PAUSE=5:GOSUB PT:GOSUB XZ:IF X2=56 O
   R X2=192 THEN GOSUB PZA:GOSUB 240
OM 2065 GOTO 2000
UA 2070 DATA 184,2,176,1,168,0,160,3,152,
   2,144,1,136,0,128,3,120,2,112,1,104,0,
   96,3,88,2,80,1,72,0,64,3,56,2
YG 2080 DATA 64,3,72,0,80,1,88,2,96,3,104
   ,0,112,1,120,2,128,3,136,0,144,1,152,2
   ,160,3,168,0,176,1,184,2,192,3,-1,-1
NG 2100 IF X2=64 THEN BKT=DEK(JJ):GOTO 21
   70
SM 2110 IF X2=88 THEN BKT=DEK(KK):GOTO 21
   70
CM 2120 IF X2=112 THEN BKT=DEK(LL):GOTO 2
   170
KH 2130 IF X2=136 THEN BKT=DEK(NN):GOTO 2
   170
UV 2140 IF X2=160 THEN BKT=DEK(4):GOTO 21
   70
JG 2150 IF X2=184 THEN BKT=0:GOTO 2170
```

```
OU 2160 GOTO 2020
HI 2170 POKE 623,4:COLOR 160:PLOT (X2/8)-
   6,15:A=USR(MOVE,LL,PMB,BALL(C),X2,82,Q
   )
JN 2180 GOSUB PZA:PAUSE=100:IF BKT=COIN T
   HEN 2300
JS 2190 IF BKT=0 THEN 2230
YQ 2210 POSITION Q,20:? #6;"wrong":POSITI
   ON Q,21:? #6;"bucket":POSITION Q,22:?
   #6;"MOM":GOSUB PT:GOSUB 250
DW 2220 A=USR(MOVE,LL,PMB,BALL(C),X2,75,Q
   ):GOSUB PZA:COLOR 134:PLOT (X2/8)-6,15
   :POKE 623,KK:GOSUB 205:GOTO 2000
RE 2230 POSITION Q,21:? #6;"OH MOM":GOSUB
    PT:COLOR 134:PLOT 17,15:POKE 623,KK:G
   OSUB 205
HI 2240 A=USR(MOVE,LL,PMB,BALL(C),20,75,Q
   ):GOSUB 240:GOTO 1000
TD 2300 COL$(11,11)="■":PAUSE=30:GOSUB PT
   :COL$(11,11)="y"
WR 2305 POSITION Q,21:? #6;"OH BOY ":A=US
   R(MOVE,LL,PMB,BALL(C),20,75,Q):FOR Z=G
   ES TO GES+COIN:DOL=INT(Z/100)
PQ 2310 POSITION 15,20:? #6;DOL:CENT=Z-(D
   OL*100)
SK 2311 IF CENT<ZL THEN POSITION 17,20:?
   #6;"0":POSITION 18,20:? #6;CENT:GOTO 2
   330
UQ 2320 POSITION 17,20:? #6;CENT
JP 2330 GOSUB PZA:NEXT Z:POSITION Q,21:?
   #6;"      ":COLOR 134:PLOT (X2/8)-6,15
   :POKE 623,KK:GES=GES+COIN
UR 2335 IF GES=COR THEN 6000
SU 2340 IF GES>COR THEN 4000
NK 2350 GOTO 1000
RC 4000 A=USR(MOVE,LL,PMB,BALL(0),26,70,Q
   ):COLOR 134:PLOT (X2/8)-6,15
EP 4005 GOSUB 205:POSITION Q,21:? #6;"BYE
    BYE":POKE 623,KK:FOR Y1=46 TO 150 STE
   P 8
AM 4006 A=USR(MOVE,KK,PMB,TAIL(JJ),X1-8,Y
   1,ST)
UT 4010 A=USR(MOVE,JJ,PMB,BOD(JJ),X1,Y1,S
   T):SOUND JJ,Y1,ZL,4:NEXT Y1:GOSUB 220:
   ROO=ROO-1:COLOR 16+ROO:PLOT 17,17
PL 4020 PAUSE=300:GOSUB 205:IF ROO=JJ THE
   N 5000
NN 4040 IF BNS=JJ THEN POSITION Q,20:? #6
   ;"TIME":POSITION Q,21:? #6;"RAN":POSIT
   ION Q,22:? #6;"OUT":GOSUB PT:GOTO 4090
II 4050 POSITION Q,20:? #6;"total":POSITI
   ON Q,21:? #6;"TOO":POSITION Q,22:? #6;
   "LARGE":GOSUB PT
LB 4090 POSITION Q,20:? #6;"TRY    ":POSIT
   ION Q,21:? #6;"AGAIN":POSITION Q,22:?
   #6;"MOMMY":GOSUB PT
AD 4100 GOSUB 205:POSITION ST,KK:? #6;"50
   ":POSITION 15,20:? #6;"0.00":BNS=50:PO
   KE 19,JJ:POKE 19,JJ:POKE 20,JJ:GES=0
PH 4101 GOTO 330
DD 5000 PAUSE=200:GOSUB PT:POSITION 5,13:
   ? #6;"GAME  OVER":GOSUB PT:COLOR 160:P
   LOT 0,1:DRAWTO 19,1:PLOT 0,2
SS 5001 DRAWTO 19,2
TQ 5010 GOTO 29010
IT 6000 PAUSE=100:GOSUB PT:M=JJ:PAUSE=20:
   POSITION Q,20:? #6;"MOMMY":POSITION Q,
   22:? #6;"DID IT":FOR Z=KK TO 7
DG 6010 COL$(4,4)=CHR$(104-104*M):COL$(12
   ,12)=COL$(4,4):SOUND JJ,81+M*40,ZL,ZL:
   GOSUB PT:M=-1*M+1:NEXT Z:GOSUB 246
YX 6020 GOSUB 205:GES=JJ:POSITION Q,19:?
   #6;"KIKI":FOR D=KK TO Q:A=USR(MOVE,JJ,
   PMB,BOD(KK),128,94-M*8,ST)
OL 6030 A=USR(MOVE,KK,PMB,TAIL(KK),136,94
   -M*8,ST):M=-1*M+1:GOSUB 215:GOSUB PT:N
   EXT D
US 6040 POSITION Q,19:? #6;"MOMMY":FOR D=
   1 TO 6:A=USR(MOVE,NN,PMB,BAB,112,98-M*
   Q,11):GOSUB 215:GOSUB PT:M=-1*M+1
ZD 6100 NEXT D:POSITION Q,19:? #6;"
   ":SC=SC+BNS:IF SC>9995 THEN POSITION K
   K,KK:? #6;"0    "
PW 6110 POSITION KK,KK:? #6;SC:GOSUB XYZ:
   BNS=50:POSITION ST,KK:? #6;"50":POSITI
   ON 15,20:? #6;"0.00"
JM 6120 M=JJ:PAUSE=30:FOR Z=KK TO ZL:COLO
   R 126+M:PLOT 9,19
GA 6140 GOSUB PT:M=-1*M+1:NEXT Z:PAUSE=20
   0:GOSUB PT:COLOR 160:PLOT 9,19
IH 6150 POSITION Q,20:? #6;"I'LL":POSITIO
   N Q,21:? #6;"SAVE":POSITION Q,22:? #6;
   "YOU"
KX 6200 A=USR(MOVE,NN,PMB,BAB,60,98,11):P
   AUSE=300:GOSUB PT:GOSUB 205:PAUSE=100:
   GOSUB PT:GOTO 260
VT 19000 REM __Assembly code
LI 19010 RESTORE 19015:FOR I=1 TO 32:READ
    A:ZZ$(I)=CHR$(A):NEXT I
DC 19015 DATA 104,104,133,204,104,133,203
   ,104,133,206,104,133,205,162,4,160,0
RH 19020 DATA 177,203,145,205,136,208,249
   ,230,204,230,206,202,208,240,96
HC 19030 POKE 106,PEEK(106)-5:GRAPHICS 17
   :START=(PEEK(106)+1)*256:POKE 752,1
GP 19035 A=USR(ADR(ZZ$),57344,START):A=US
   R(ADR(ZZ$),57344,START+512):RESTORE 25
   000
CQ 19040 FOR X=0 TO 504 STEP 8:FOR Y=0 TO
    7:READ Z:POKE X+Y+START,Z:NEXT Y:NEXT
    X
FI 19045 GRAPHICS 17:POKE 756,PEEK(106)+1
   :GOTO 25050
PU 25000 DATA 0,0,0,0,0,0,0,0,60,126,251,
   253,253,251,126,60,0,102,102,102,0,0,0
   ,0
IX 25002 DATA 4,4,4,4,4,4,4,4,32,32,32,32
   ,32,32,32,32,0,0,0,0,255,0,0
AL 25004 DATA 0,0,255,0,0,0,0,0,0,0,0,0,0
   ,7,4,4,4,4,7,0,0,0,0,0
LW 25006 DATA 0,0,0,0,0,224,32,32,32,32,2
   24,0,0,0,0,0,32,32,32,32,32,63,0,0
MD 25008 DATA 4,4,4,4,4,252,0,0,0,0,255,7
   3,146,255,0,0,0,0,0,0,0,24,24,0
ZI 25010 DATA 7,7,7,7,7,7,7,7,127,99,99,9
   9,99,99,127,0,56,24,24,24,62,62,62,0
QK 25012 DATA 127,3,3,127,96,96,127,0,126
   ,6,6,127,7,7,127,0,112,112,112,112,119
   ,127,7,0
CK 25014 DATA 127,96,96,127,3,3,127,0,124
   ,108,96,127,99,99,127,0,127,3,3,31,24,
   24,24,0
WG 25016 DATA 62,54,54,127,119,119,127,0,
   127,99,99,127,7,7,7,0,255,255,255,255,
   255,255,255,255
WW 25018 DATA 255,255,255,0,0,0,0,0,231,2
   31,231,231,231,231,231,231,231,231,231
   ,0,0,0,0,0
NO 25020 DATA 224,224,224,224,224,224,224
   ,224,0,36,126,100,126,38,126,36,0,8,12
   6,104,104,126,8,0
IR 25022 DATA 63,51,51,127,115,115,115,0,
   126,102,102,127,103,103,127,0,127,103,
   103,96,99,99,127,0
SA 25024 DATA 126,102,102,119,119,119,127
   ,0,127,96,96,127,112,112,127,0,127,96,
   96,127,112,112,112,0
JD 25026 DATA 127,99,96,111,103,103,127,0
   ,115,115,115,127,115,115,115,0,127,28,
   28,28,28,28,127,0
UY 25028 DATA 12,12,12,14,14,110,126,0,10
   2,102,108,127,103,103,103,0,48,48,48,1
   12,112,112,126,0
PL 25030 DATA 103,127,127,119,103,103,103
   ,0,103,119,127,111,103,103,103,0,127,9
```

```
   9,99,103,103,103,127,0
ZM 25032 DATA 127,99,99,127,112,112,112,0
   ,127,99,99,103,103,103,127,7,126,102,1
   02,127,119,119,119,0
GL 25034 DATA 127,96,127,3,115,115,127,0,
   127,28,28,28,28,28,28,0,103,103,103,10
   3,103,103,127,0
KV 25036 DATA 103,103,103,103,111,62,28,0
   ,103,103,103,111,127,127,103,0,115,115
   ,115,62,103,103,103,0
FZ 25038 DATA 103,103,103,127,28,28,28,0,
   224,224,224,0,0,0,0,0,7,7,7,0,0,0,0,0
LR 25040 DATA 36,36,36,36,36,36,36,36,0,6
   0,60,12,24,0,0,0,102,231,255,255,126,6
   0,24,0
XC 25042 DATA 0,36,126,126,60,24,0,0
UM 25050 TT$(1)="hh*♦♥P )> )/\ /"JPR/PX"
OG 25060 TT$(24)="N)/Pq*JPgh*♦♥P-- )/ /")"
ES 25070 TT$(46)="JPR/PxN)/Pq♦"
CW 26035 RESTORE 26036:FOR X=1 TO 15:READ
    Z:COL$(X,X)=CHR$(Z):NEXT X
QE 26036 DATA 140,90,27,104,0,88,182,73,1
   5,0,120,104,200,90,0
GN 26040 RESTORE 26041:FOR X=1 TO 46:READ
    Z:PR$(X,X)=CHR$(Z):NEXT X
CB 26041 DATA 72,138,72,152,72,162,0,173,
   11,212,201,7,240,3,174,214,89,160,0,18
   9,223,89,153,22,208,232,200
JZ 26042 DATA 192,5,208,244,142,214,89,10
   4,168,104,170,104,64,0,15,2,3,4,5
PP 26050 P=ADR(PR$):PHI=INT(P/256):PLO=(P
   -PHI*256):C=ADR(COL$):CHI=INT(C/256):C
   LO=(C-CHI*256)
LT 26060 PR$(21,21)=CHR$(CLO):PR$(22,22)=
   CHR$(CHI):PXHI=INT((P+41)/256):PXLO=(P
   +41)-(PXHI*256)
PE 26070 PR$(5T,5T)=CHR$(PXLO):PR$(17,17)
   =CHR$(PXHI):PR$(33,33)=CHR$(PXLO):PR$(
   34,34)=CHR$(PXHI)
KB 26080 POKE 512,PLO:POKE 513,PHI:POKE 5
   4286,192
KE 26090 DL=PEEK(560)+256*PEEK(561):POKE
   DL,112+128:POKE DL+7,134:POKE DL+20,13
   4
VY 28000 REM ********PM ROUTINE *****
GF 28010 DIM PMMOV$(100),BALL(4),TAIL(NN)
   ,BOD(NN),BOD0$(ST),BOD1$(ST),TAIL0$(ST
   ),TAIL1$(ST),BAB$(11),BAB(11)
SN 28015 DIM BALL0$(Q),BALL1$(Q),BALL2$(Q
   ),BALL3$(Q):MOVE=ADR(PMMOV$):BOD(JJ)=A
   DR(BOD0$):BOD(KK)=ADR(BOD1$)
LV 28017 TAIL(JJ)=ADR(TAIL0$):TAIL(KK)=AD
   R(TAIL1$):BALL(JJ)=ADR(BALL0$)
PJ 28019 BALL(KK)=ADR(BALL1$):BALL(LL)=AD
   R(BALL2$):BALL(NN)=ADR(BALL3$):BAB=ADR
   (BAB$):RESTORE 28030
GR 28020 FOR X=1 TO 100:READ Z:PMMOV$(X,X
   )=CHR$(Z):NEXT X:GOTO 28100
GQ 28030 DATA 216,104,104,133,213,104
   ,24,105,2,133,206,104,133,205,104,133,
   204,104,133,203,104,104,133,208
YH 28035 DATA 104,104,133,209,104,104,24,
   101,209,133,207,166,213,240,16,165,205
   ,24,105,128,133,205,165,206,105
HB 28040 DATA 0,133,206,202,208,240,160,0
   ,162,0,196,209,144,19,196,207,176,15,1
   32,212,138,168,177,203,164
GY 28045 DATA 212,145,205,232,169,0,240,4
   ,169,0,145,205,200,192,128,208,224,166
   ,213,165,208,157,0,208,96
NP 28100 BOD0$="♥@p\ \@♦x1deee♦8":TAIL0$=
   "┘ |♥┘ |♥♥♥♥├┘ ┘&<♥":BOD1$="┘ ;>: \→7'┘
   ┘┘/↑":TAIL1$="@e♥@e♥♥♥♥♥♥eeed<♥"
KT 28105 BAB$="p\ ▖ ◢←:8h ":BALL0$="<v♣v
```

ATARI USERS' GROUPS

NorthWest Phoenix Atari Connection (NWPAC)
SouthEast Vallet Atari Connection (SEVAC)
P.O. Box 36364, Phoenix, AZ 85067
Meetings; BBS; newsletter: *Nibbles & Bytes*
President (NWPAC) - Steve Marshall
President (SEVAC) - Tim Barr

Pasadena Area Local Atari Computer Enthusiasts (PALACE)
P.O. Box 1771, Arcadia, CA 91006
Meetings; BBS; newsletter

San Diego Atari Computer Enthusiasts (SDACE)
P.O. Box 203076, San Diego, CA 92120
Meetings; BBS; newsletter: *The I/O Connector*
President - David Delgadillo

New Orleans Atari Users' Group (NOAUG)
P.O. Box 73236, Metairie, LA 70033
Meetings; newsletter; President - Dave Porter

Twin Cities Atari Interest Group (TAIG)
P.O. Box 26128, Minneapolis, MN 55426
Meetings; newsletter

South Jersey Atari Computer Enthusiasts (SJACE)
1548-A Cedar St., Ft. Dix, NJ 08640
Meetings; newsletter: *Connect!*

Brooklyn Atari Society for Information and Communication (BASIC)
c/o Peter J. Fazio, 2724 East 23rd St., Brooklyn, NY 11235
Meetings; BBS; newsletter: *Dateline Atari*
President - Peter Fazio

Atari Computer Enthusiasts of Syracuse
c/o Vernon W. Smith, 101 Clarke St., Syracuse, NY 13210
Meetings; newsletter; President - Ken Benson

Tri-County Atari Users' Club (TCAUC)
4 Circle Dr., New Breman, OH 45869
Meetings; newsletter
8-Bit President - Alan Dock (419) 629-2438
ST SIG President - Pamela Rice Hahn (419) 394-7588

Rushmore Atari Computer Enthusiasts (RACE)
3711 Canyon Lake Dr. #202, Rapid City, SD 57702
Meetings; newsletter: *Race-Tracks*
President - Joe Moews (605) 348-6331

Peninsula Atari Computer Enthusiasts (PACE)
c/o Joe Cullen, 834 Churchhill Terrace, Hampton, VA 23666
Meetings; newsletter; President - Jeff Cleveland

Seattle-Puget Sound Atari Computer Enthusiasts (SPACE)
P.O. Box 110576, Tacoma, WA 98411-0576
Meetings; BBS; newsletter; President - Scott McGowen

The Toronto Atari Federation
5334 Yonge St., Toronto, Ontario, Canada M2N 6M2

Manitoba Atari Computer Club (MACC)
c/o Rick Singbeil, 483 Ferry Road
Winnipeg, Manitoba, Canada R3J 1W5
Meetings; newsletter; President - Bruce Campbell.

ATTENTION USERS' GROUPS

If you would like your organization to be listed here, send information (and newsletter, if appropriate) to **ANALOG Computing** Group Listing, L.F.P., Inc., 9171 Wilshire Blvd., Suite 300, Beverly Hills, Calif. 90210. **ANALOG Computing** is not responsible for errors in copy.

```
    (":BALL1$="(■I■f(":BALL2$="(n■■n(":BAL
    L3$="(f■■(("
BM 28200 PMBASE=INT((PEEK(145)+NN)/4)*4:P
    OKE 54279,PMBASE:PMB=PMBASE*256:POKE 5
    59,46:POKE 53277,NN:POKE 623,KK
IX 28210 POKE 704,28:POKE 705,28:POKE 707
    ,28
SM 29010 POSITION JJ,NN:? #6;"   %) %) %)
     %) %)   ";"    $  $  $  $  $   ";" #
    p$#n$#d$#q$#h$   "
YV 29020 POSITION JJ,Q:? #6;"  #e$#i$#i$#
    u$#a$   ";"  #n$#c$#m$#a$#1$   ";"  #n
    $#k$#e$#r$#v$   "
MF 29030 POSITION JJ,9:? #6;"  #y$#1$# $#
    t$#e$   ";"  # $#e$# $#e$# $   ";"'%,
    +, +, +,r+, +%%)"
DP 29040 POSITION JJ,12:? #6;"#
             $";"#---------------- $";"#
                    $"
IF 29060 POSITION JJ,15:? #6;"(&&&&&&&&&&
    &&&&&&&&&";"/::(::(::(::(::(:::)";"/01
    (05(10(25(50(:  :)"
KW 29070 POSITION JJ,18:? #6;"[;;=;;=;;=;
    ;=;;=;;;Z";"'%%%%)          '%%%%%)";"#
      $          #?0.00$"
SE 29075 COLOR 134:PLOT LL,15:PLOT 5,15:P
    LOT 8,15:PLOT 11,15:PLOT 14,15:PLOT 17
    ,15:COLOR 147:PLOT 17,17
VG 29080 POSITION JJ,21:? #6;"#     $
     (&&&&&&*";"#     $          total ";"(&&
    &&*"
RL 29090 COLOR 161:PLOT NN,4:COLOR 129:PL
    OT 6,4:PLOT 9,4:PLOT 12,4:PLOT 15,4:A=
    USR(MOVE,NN,PMB,BAB,60,98,11)
SD 29200 POSITION JJ,JJ:? #6;"25¢     mone
    y  pouch ":POSITION JJ,KK:? #6;"EASY":
    LIMIT=25:SC=JJ:GES=JJ:COIN=JJ
EP 29205 BNS=50:XO=JJ:LEV=JJ:LEVEL=15:CN=
    JJ:ROO=NN:RESTORE 29340
TJ 29210 READ D,B:IF D=-1 THEN RESTORE 29
    340:GOTO TP
XU 29220 IF D=-2 THEN POSITION Q,KK:? #6;
    "  for analog ":GOTO TP
CJ 29230 IF D=-3 THEN POSITION Q,KK:? #6;
    "by chuck rosko":GOTO TP
NK 29240 IF D=-4 THEN POSITION Q,KK:? #6;
    "copyright 1986":GOTO TP
WI 29250 IF PEEK(53279)=Q THEN GOSUB 246:
    GOTO 29280
SI 29255 IF PEEK(53279)=5 THEN GOSUB 246:
    GOTO 29330
JM 29260 IF PEEK(53279)=NN THEN GOSUB 246
    :GOTO 29310
VK 29270 SOUND JJ,D,ZL,ZL:SOUND KK,D+1,ZL
    ,ZL:A=USR(ADR(TT$),B):GOSUB 246:GOTO T
    P
NZ 29280 POSITION JJ,JJ:? #6;"score   Pri
    ce      ";" 0      ?  .      50  "
KI 29290 COLOR 7:PLOT 15,JJ:COLOR 5:PLOT
    16,JJ:PLOT 17,JJ:COLOR 9:PLOT 18,JJ:CO
    LOR 4:PLOT 18,KK:COLOR NN:PLOT 15,KK
YE 29300 COLOR Q:PLOT 17,LL:PLOT 5T,LL:CO
    LOR 10:PLOT 18,LL:COLOR 8:PLOT 15,LL
YZ 29301 FOR D=4 TO 0 STEP -1:Z=INT(RND(J
    J)*(D+KK)):TEMP=DEK(D):DEK(D)=DEK(Z):D
    EK(Z)=TEMP:NEXT D
TP 29302 POSITION JJ,17:? #6;"/  (  (  (
     (  ":FOR Z=JJ TO 4
HH 29303 IF DEK(Z)=1 THEN POSITION KK+Z*N
    N,17:? #6;"01":GOTO 29308
RY 29304 IF DEK(Z)=5 THEN POSITION KK+Z*N
    N,17:? #6;"05":GOTO 29308
BY 29305 IF DEK(Z)=10 THEN POSITION KK+Z*
    NN,17:? #6;"10":GOTO 29308
SF 29306 IF DEK(Z)=25 THEN POSITION KK+Z*
    NN,17:? #6;"25":GOTO 29308
MU 29307 IF DEK(Z)=50 THEN POSITION KK+Z*
    NN,17:? #6;"50":GOTO 29308
PD 29308 GOSUB XYZ:NEXT Z:GOTO 260
IY 29310 LIMIT=LIMIT+25:IF LIMIT>100 THEN
     LIMIT=25:POSITION JJ,JJ:? #6;"    "
TL 29320 POSITION JJ,JJ:? #6;LIMIT;"¢":PA
    USE=20:GOSUB PT:GOSUB XYZ:GOTO TP
JY 29330 LEV=LEV+KK:IF LEV>LL THEN LEV=JJ
VV 29331 GOTO 29332+LEV
VK 29332 LEVEL=15:POSITION JJ,KK:? #6;"EA
    SY":PAUSE=20:GOSUB PT:GOSUB XYZ:GOTO T
    P
NF 29333 LEVEL=ZL:POSITION JJ,KK:? #6;"MO
    D ":PAUSE=20:GOSUB PT:GOSUB XYZ:GOTO T
    P
QC 29334 LEVEL=5:POSITION JJ,KK:? #6;"HAR
    D":PAUSE=20:GOSUB PT:GOSUB XYZ:GOTO TP
OC 29340 DATA 121,30,121,10,121,10,121,30
    ,121,10,121,10,96,30,81,10,81,10,96,10
    ,96,10,121,30,-2,-2
WV 29350 DATA 108,30,108,10,108,10,108,30
    ,108,10,108,10,128,30,108,10,108,10,12
    8,10,162,30,-3,-3
HL 29360 DATA 121,10,121,10,121,10,121,10
    ,121,30,121,10,121,10,96,30,81,10,81,1
    0,96,10,96,10,121,30,-4,-4
PY 29370 DATA 108,30,108,10,108,10,162,10
    ,162,10,162,30,121,75,-1,-1
RE 30000 GOTO 30
```

●

BOOT UP TO BIG SAVINGS

SUBSCRIBE

☐ 1 Year.....$28..... Save $14
MCEYY

☐ 1 Year with Disk.....$105
DCEYY

Name

Address

City State Zip

Make checks payable to: L.F.P. Inc. Allow 4-6 weeks for delivery.

☐ Payment Enclosed ☐ Bill Me
☐ Charge My ☐ Visa ☐ MC
_____ Exp ____

Signature

FOREIGN—Add $7 per year
MONEY BACK if not delighted
Analog
P.O. Box 16927
N. Hollywood, CA 91615

Needlework Design

Brings creative impulses to the screen with ease

by Regena

Needlework Design is a set of programs written in low resolution ST BASIC. You can use these programs to design a needlework project, make changes on the screen, then handcraft the end result.

Included are three types of needlework designing sections. The Cross Stitch portion shows a grid on-screen; you use the mouse to place colored dots on the grid. After your pattern is complete, it may be printed out. The Doily pattern is similar to the Cross Stitch, except you design in only one color, with the design "reflected" on the four quadrants of the screen so the pattern is symmetrical. Quilt Squares lets you put together a patchwork quilt design. The basic pattern is then repeated so you can see how several squares would look together.

You can run the programs separately, or RUN NEEDLE (Listing 1) to get a menu screen, which will then chain to the program chosen.

Needlework Design (Listing 1) is a short main menu program. The full output window is used. The screen clears, and the title and three choices are printed. Lines 150-170 detect which key the user must press to make the choice (either a 1, 2 or 3.) The CHAIN command is used to load and run one of the three programs, entitled XSTITCH, DOILY and QUILT.

Cross Stitch

Cross Stitch (Listing 2, XSTITCH) is a graphics program to design counted cross stitch, needlepoint, or lace net darning. After the title and instruction screen clears, a grid appears. Use the mouse to move the mouse arrow to a square, then press the left mouse button to place a dot of color. You may change colors by positioning the mouse and pressing the right mouse button. Repeatedly pressing the right button will display all the colors. The left mouse button continues to draw in that color. At any time, you can press a SHIFT key to print the design. Different colors are represented by different letters of the alphabet.

Line 30 DIMensions the array D to keep track of what's in each grid position on the screen. Lines 40-180 clear the screen and print the title and instructions. Lines 190-210 initialize the D array. Lines 220-240 wait for the user to press the F1 function key to clear the instructions and start designing.

Line 250 clears the screen. Line 260 defines the drawing color to be light gray, then Lines 270-320 draw the grid on the screen using FOR-NEXT loops. Line 330 defines the starting color (red).

Lines 340-430 detect the mouse control. KB will equal 1 or 2 if the SHIFT key is pressed. MB will equal 0 if a mouse button is not pressed. Lines 390-410 change the mouse color when the right mouse button is pressed. The color number is incremented by 1 each time the button is pressed. After color number 15, the color number starts with 0 again.

Lines 420-450 determine the position of the mouse arrow if a mouse button is pressed. MX and MY are the coordinates. The arrow must be on the grid and cannot point to a line. Lines 460-490 define AA and BB for a row and column number in using the D array. The D array keeps track of the color number placed in the square. A and B are used as the center coordinates of the square chosen, and PCIRCLE A,B,2 draws the colored dot in the square.

Line 500 branches back to Line 360 to detect the next action with the mouse or keyboard.

Lines 510-580 contain the printing process when a SHIFT key is pressed. Line 560 checks each D element in order. If the color number is 0, a space is printed. If there's a color number in D, that number is added to 64 to get an ASCII

character number, which will print an alphabetic character. Each letter of the alphabet represents a different color in the design. Line 590 branches back to detecting the mouse after printing is completed.

Doily

DOILY (Listing 3) is similar to the Cross Stitch program, but uses only one color. A cross stitch doily is usually sewn in only one color and is symmetric in four quadrants (sometimes eight sections). This style is also appropriate for lace net darning. As you use the left mouse button to place dots on the grid, other dots are automatically drawn on the other quadrants, reflected on the center axes.

To erase a dot, simply position the mouse arrow on the dot and press the left button. The color drawn alternates between red and white. If you press the left mouse button on an empty square, a red dot will be drawn; if you press the button on a red dot, the dot will disappear.

Use the right mouse button to get a hard copy of the design. After the printing process you will have the same design to continue working on.

Line 30 DIMensions the D array to keep track of dot locations. Lines 40-150 clear the screen and print the title and instructions. Lines 160-180 initialize the D array. All positions are *0* to start with. When a dot of color is placed, that D element becomes a *1*. Lines 190-210 wait for the user to press the F1 function key to start designing.

Line 220 clears the screen. Line 230 defines the drawing color to be gray, and Lines 240-290 draw the grid. Lines 300-320 change the color and draw the center X- and Y-axis for reflection. Line 330 defines the drawing color as red.

Lines 340-420 detect the mouse control. Line 380 causes a branch to the printing routine if the right mouse button is pressed. MX and MY are the coordinates when the left mouse button is pressed. The mouse arrow must be on the grid and must not point to a line. Lines 430-470 place the dot on the grid. AA and BB use MX and MY to determine integer grid positions for the D array. A and B determine the center of the square so PCIRCLE A,B,2 can draw the dot. Line 450 uses the SGN function (sign) with the ABS function (absolute value) to switch the D element value of C between *0* and *1* or *1* and *0*. Line 460 then redefines D and the color.

Lines 480-530 draw the dots in the other quadrants, using coordinates A, B, A2 and B2. The corresponding D elements also need to be redefined. Line 540 branches back to the mouse detection.

Lines 550-620 contain the printing procedure for a hard copy. The D element will contain *1* if there's a dot and *0* if there isn't. The printer will print an *X* for a dot and a space for no dot. Line 630 branches back to the mouse detection so design can continue.

Quilt Squares

Quilt Squares (Listing 4, QUILT) lets you design a patchwork quilt on-screen before actually sewing all the little squares and triangles together. After you design one basic square, consisting of four small squares by four small squares, that pattern is repeated on the screen so you can see how several patterns will look together.

First choose three colors from the sixteen available. If you prefer a quilt with only two colors, simply choose one of the colors twice. After you press the F1 key to begin, the screen clears and the possible design squares are shown at the right side of the screen. The square may be all one color or any combination of triangles of two colors.

A larger basic set of 4x4 squares is shown on the left side of the screen. The squares are originally outlined in gray. As each one is highlighted in black, move the mouse arrow to one of the small designs and press the left mouse button. That pattern will then appear on the larger square. Continue the process for all sixteen squares.

After all sixteen squares have been designed, the computer will surround that original basic square with the quilt pattern repeated with three patterns by three patterns. You will now be able to see how your design will look as a larger patchwork quilt.

Now you can make changes. The original sixteen squares will be highlighted in black, one at a time. If you want to keep the square as is, press the right mouse button and it won't change. If you wish to change the pattern of one square, go to the small design squares and select a pattern with the left mouse button. The pattern selected will appear in the original basic pattern, then the computer will change the other eight corresponding squares in the repeated patterns. The sixteen original squares will continue to be highlighted one at a time if you want to keep making changes. You may keep the quilt on the screen as you actually cut and sew your real patchwork quilt, or press CTRL-C to stop the program.

Lines 30-40 DIMension variables used as arrays. C() is the three color numbers chosen. S$() is the patterns for the small design squares at the right side of the screen. TOP() and BOT() are the top and bottom colors for the triangles within the design squares. The variables starting with Q are used for the 4x4 basic design squares on the left side of the screen. QP$() is the pattern style, QT() is the top color and QB() is the bottom color. The Q variables are used when the pattern is repeated across the screen.

Lines 50-80 define a full window, clear the screen and print the title. Lines 90-260 receive the user's choices for three colors. INP(2) is used to detect a key press for a color number. If the color desired is color number 1, press *1* then RETURN; otherwise, you may press the color number without the return key. The color numbers are stored as C(1), C(2) and C(3).

Lines 270-320 print some instructions. Line 330 uses GOSUB SETUP to define the design patterns. Lines 340-350 wait for the function key F1 to be pressed to begin.

Line 360 clears the screen. Line 370 defines the drawing color to be light gray and Lines 380-440 draw the small squares for all the possible design squares. Lines 450-490 then draw diagonal lines in those squares consisting of two triangles. Lines 500-640 use the subroutine START to color in the squares. The data statements consist of X- and Y-coordinates for the fill commands using the three colors.

Line 650 redefines the color as light gray, then Lines 660-710 draw the main 4x4 square. Lines 720-870 are nested FOR-NEXT loops to design each of these sixteen squares.

The squares are defined by ROW and CLM, which vary from 1 to 4.

Line 740 determines graphic coordinates X1 and Y1, depending on ROW and CLM. Lines 750-760 then outline that "box" in black. Line 770 calls the subroutine to check the mouse. Line 780 determines if this is the first design phase or the changing phase. Line 790 calls the subroutine to color in the square, depending on the design chosen. Line 800 also checks which design phase; if the pattern already covers the screen, the subroutine REPEAT is called to change the corresponding eight squares in the repeated patterns. Line 810 changes the color back to gray to outline the box and Line 820 goes to the next box.

After the sixteen squares have been designed, Lines 830-870 repeat these sixteen squares across the screen in three main pattern blocks across and three main pattern blocks down, so you can see nine patterns together.

Lines 880-940 print the instructions for changing pattern squares by selecting a design pattern with the left mouse button or pressing the right mouse button for no change. GOTOXY is used to place the printing without messing up the quilt pattern on the screen. The variable FLAG is set to *0* to start the program, but now becomes *1* for the changing phase. Line 950 then branches to Line 720 to use the same programming lines in designing the sixteen squares.

Lines 960-1160 are the subroutine SQUARE, which is used to color in a larger square, depending on the small design square chosen. Lines 970-1030 erase a square during the "change" phase of designing. Dark gray is used to fill in the square and get rid of all previously drawn diagonal lines.

Line 1040 determines the pattern PAT$ of the particular square. *F* means the square is all one color. *RL* is a square of two triangles with the diagonal drawn from top right to bottom left. *LR* is a square of two triangles with the diagonal drawn from top left to bottom right. Line 1050 defines the top color T and the bottom color B. Lines 1060-1150 draw the appropriate diagonals, if necessary, and fill the colors. Line 1160 is RETURN.

Lines 1170-1360 are the subroutine CHECK, which checks the mouse control. Line 1200 determines if a mouse button is pressed. Line 1210 returns if you are in the changing design phase and the right mouse button is pressed to indicate no change. Line 1220 determines the coordinates of the arrow point when the left mouse button is pressed in the changing phase, or either button is pressed in the first design phase. The coordinates are MX and MY.

Lines 1230-1300 check the arrow position. The arrow point must be on or in a design square for the computer to respond. These lines make sure the arrow point is not in a white space around the small design square. If MX and MY are valid, Lines 1310 and 1320 determine integer coordinates YY and XX, indicating which row and column the design square is in. Lines 1330-1360 use YY and XX to determine the pattern style and colors, then return to the main program.

Lines 1370-1520 are the SETUP subroutine. Line 1380 defines *FLAG=0* for the first time through designing the sixteen squares. Lines 1390-1430 define the pattern styles, F, RL and LR, for the small design squares in five rows and three columns. Lines 1440-1510 define the top and bottom colors for each possible design square. Line 1520 returns to the main program.

Lines 1530-1570 are the subroutine START, which is used to fill in the colors of the small design squares.

Lines 1580-1630 are the subroutine BOX, which is used to outline a square in gray, dark gray or black.

Lines 1640-1710 are the subroutine REPEAT, which is used to repeat the basic sixteen-square pattern in three rows and three columns. Line 1720 ends the program.

Regena got her first home computer (a T1-99/4) for Christmas in 1980. Ideas for the hundreds of BASIC programs she's had published (for various computers) come from her six children. A regular columnist in COMPUTE!, *her latest book is* Elementary ST BASIC, *from COMPUTE! Publications, Inc.*

Listing 1
ST BASIC listing

```
10 REM NEEDLEWORK DESIGN
20 REM BY REGENA
30 FULLW 2:CLEARW 2
40 GOTOXY 8,2
50 PRINT "** NEEDLEWORK DESIGN **"
60 PRINT:PRINT
70 PRINT TAB(8);"CHOOSE:"
80 PRINT
90 PRINT TAB(8);"1  CROSS STITCH"
100 PRINT
110 PRINT TAB(8);"2  DOILY"
120 PRINT
130 PRINT TAB(8);"3  QUILT SQUARES"
140 PRINT:PRINT:PRINT
150 K=INP(2)
160 IF K<49 OR K>51 THEN 150
170 ON K-48 GOTO 180,200,220
180 PRINT "CROSS STITCH"
190 CHAIN "XSTITCH"
200 PRINT "DOILY"
210 CHAIN "DOILY"
220 PRINT "QUILT SQUARES"
230 CHAIN "QUILT"
240 END
```

ST CHECKSUM DATA

```
 10 data  269, 11, 516, 630, 284, 116,
 169, 175, 107, 140, 2417
 110 data  252, 146, 494, 956, 386, 20
9, 70, 81, 335, 109, 3038
 210 data  26, 211, 47, 789, 1073
```

Listing 2
ST BASIC listing

```
10 REM CROSS STITCH
20 REM BY REGENA
30 DIM D(73,41)
40 FULLW 2:CLEARW 2
50 GOTOXY 0,0
60 PRINT TAB(10);"** CROSS STITCH **"
70 PRINT:PRINT
80 PRINT " Draw by moving the mouse"
90 PRINT " to the desired square then"
```

```
100 PRINT " pressing the LEFT mouse bu
tton."
110 PRINT
120 PRINT " Change colors by pressing"
130 PRINT " the RIGHT mouse button."
140 PRINT:PRINT
150 PRINT " To print the design on a p
rinter,"
160 PRINT " press the SHIFT key."
170 PRINT:PRINT
180 PRINT " Press Control-C to stop."
190 FOR X=0 TO 73:FOR Y=0 TO 41
200 D(X,Y)=0
210 NEXT Y:NEXT X
220 PRINT:PRINT
230 PRINT " Press F1 to start."
240 R=INP(2):IF R<>187 THEN 240
250 CLEARW 2
260 COLOR 1,1,8,1,1
270 FOR X=4 TO 296 STEP 4
280 LINEF X,0,X,168
290 NEXT X
300 FOR Y=4 TO 168 STEP 4
310 LINEF 0,Y,296,Y
320 NEXT Y
330 C=2:COLOR 1,C,C,1,1
340 A#=GB
350 G2=PEEK(A#+12)
360 GEMSYS(79)
370 KB=PEEK(G2+8):IF KB=1 OR KB=2 THEN
 520
380 MB=PEEK(G2+6):IF MB=0 THEN 360
390 IF MB=1 THEN 420
400 C=C+1:IF C=16 THEN C=0
410 COLOR 1,C,C,1,1
420 MX=PEEK(G2+2):MY=PEEK(G2+4)-21
430 IF MX>295 OR MY>168 THEN 360
440 A=MX/4:B=MY/4
450 IF INT(A)=A OR INT(B)=B THEN 360
460 AA=INT(A):BB=INT(B)
470 A=AA*4+2:B=BB*4+2
480 D(AA,BB)=C
490 PCIRCLE A,B,2
500 GOTO 360
510 REM PRINTING
520 WIDTH LPRINT 75
530 LPRINT
540 FOR Y=0 TO 40
550 FOR X=0 TO 72
560 IF D(X,Y)=0 THEN LPRINT " "; ELSE
LPRINT CHR$(D(X,Y)+64);
570 NEXT X:LPRINT " "
580 NEXT Y
590 GOTO 360
600 END
```

ST CHECKSUM DATA

```
 10 data  565, 11, 617, 518, 604, 580,
 118, 915, 352, 441, 4721
 110 data  143, 135, 557, 54, 581, 0,
63, 972, 605, 310, 3420
 210 data  366, 50, 840, 589, 383, 43,
 997, 203, 322, 970, 4763
 310 data  205, 305, 627, 1, 37, 728,
998, 10, 337, 995, 4243
 410 data  79, 40, 719, 129, 241, 823,
 317, 691, 110, 403, 3552
 510 data  78, 420, 308, 995, 9, 417,
689, 327, 430, 785, 4458
```

Listing 3
ST BASIC listing

```
10 REM DOILY
20 REM BY REGENA
30 DIM D(73,41)
40 FULLW 2:CLEARW 2
50 GOTOXY 0,0
60 PRINT TAB(10);"** DOILY **"
70 PRINT:PRINT
80 PRINT " Draw by moving the mouse"
90 PRINT " to the desired square then"
100 PRINT " pressing the LEFT mouse bu
tton."
110 PRINT:PRINT
120 PRINT " To print the design on a p
rinter,"
130 PRINT " press the RIGHT mouse butt
on."
140 PRINT:PRINT
150 PRINT " Press Control-C to stop."
160 FOR X=0 TO 73:FOR Y=0 TO 41
170 D(X,Y)=0
180 NEXT Y:NEXT X
190 PRINT:PRINT:PRINT
200 PRINT " Press F1 to start."
210 R=INP(2):IF R<>187 THEN 210
220 CLEARW 2
230 COLOR 1,1,8,1,1
240 FOR X=4 TO 296 STEP 4
250 LINEF X,0,X,168
260 NEXT X
270 FOR Y=4 TO 168 STEP 4
280 LINEF 0,Y,296,Y
290 NEXT Y
300 COLOR 1,1,6,1,1
310 LINEF 0,84,296,84
320 LINEF 148,0,148,168
330 COLOR 1,2,2,1,1
340 A#=GB
350 G2=PEEK(A#+12)
360 GEMSYS(79)
370 MB=PEEK(G2+6):IF MB=0 THEN 360
380 IF MB=2 THEN 560
390 MX=PEEK(G2+2):MY=PEEK(G2+4)-21
400 IF MX>295 OR MY>168 THEN 360
410 A=MX/4:B=MY/4
420 IF INT(A)=A OR INT(B)=B THEN 360
430 AA=INT(A):BB=INT(B)
440 A=AA*4+2:B=BB*4+2
450 C=ABS(SGN(D(AA,BB))-1)
460 D(AA,BB)=C:COLOR 1,C*2,C*2
470 PCIRCLE A,B,2
480 AA2=73-AA:BB2=41-BB
490 A2=AA2*4+2:B2=BB2*4+2
500 PCIRCLE A2,B2,2
510 PCIRCLE A,B2,2
520 PCIRCLE A2,B,2
530 D(AA2,BB)=C:D(AA,BB2)=C:D(AA2,BB2)
=C
540 GOTO 360
550 REM PRINTING
560 WIDTH LPRINT 75
570 LPRINT
580 FOR Y=0 TO 40
590 FOR X=0 TO 72
600 IF D(X,Y)=1 THEN LPRINT "X"; ELSE
```

```
LPRINT " ";
610 NEXT X:LPRINT " "
620 NEXT Y
630 GOTO 360
640 END
```

●

ST CHECKSUM DATA

```
 10 data  532, 11, 617, 518, 604, 583,
 118, 915, 352, 441, 4691
 110 data  45, 572, 831, 54, 963, 596,
 329, 385, 971, 831, 5577
 210 data  577, 374, 34, 988, 194, 313
, 989, 224, 324, 25, 4042
 310 data  292, 509, 32, 1, 37, 728, 7
, 346, 59, 710, 2721
 410 data  120, 232, 814, 308, 955, 51
1, 104, 623, 692, 269, 4628
 510 data  132, 163, 620, 415, 90, 432
, 320, 7, 21, 927, 3127
 610 data  673, 311, 414, 797, 2195
```

●

Listing 4
ST BASIC listing

```
10 REM QUILT
20 REM BY REGENA
30 DIM C(3),S$(5,3),TOP(5,3),BOT(5,3)
40 DIM QP$(4,4),QT(4,4),QB(4,4)
50 FULLW 2:CLEARW 2
60 GOTOXY 0,0
70 PRINT TAB(10);"** QUILT **"
80 PRINT:PRINT
90 PRINT "Choose colors:"
100 FOR T=1 TO 3
110 PRINT:PRINT "COLOR";T
120 PRINT:PRINT " 0=White"
130 FOR CC=1 TO 15
140 COLOR CC:PRINT STR$(CC);
150 NEXT CC:PRINT
160 COLOR 1
170 K1=INP(2)
180 IF K1<48 OR K1>57 THEN 170
190 IF K1>49 THEN C(T)=K1-48:GOTO 240
200 K=INP(2)
210 IF K=13 THEN C(T)=K1-48:GOTO 240
220 IF K<48 OR K>53 THEN 200
230 C(T)=10*(K1-48)+K-48
240 PRINT:PRINT " COLOR NUMBER";C(T)
250 PRINT:PRINT
260 NEXT T
270 PRINT:PRINT:PRINT
280 PRINT "Choose pattern for square b
y"
290 PRINT "moving mouse arrow to desir
ed"
300 PRINT "small pattern and pressing"
310 PRINT "the left mouse button."
320 PRINT:PRINT:PRINT
330 GOSUB SETUP
340 PRINT "Press F1 to start."
350 K=INP(2):IF K<>187 THEN 350
360 CLEARW 2
370 COLOR 1,8,8,1,1
380 FOR A=240 TO 280 STEP 20
390 FOR B=10 TO 90 STEP 20
400 LINEF A,B,A+10,B
410 LINEF A,B,A,B+10
420 LINEF A+10,B,A+10,B+10
430 LINEF A,B+10,A+10,B+10
440 NEXT B,A
450 FOR A=240 TO 280 STEP 20
460 FOR B=30 TO 70 STEP 40
470 LINEF A,B+10,A+10,B
480 LINEF A,B+20,A+10,B+30
490 NEXT B,A
500 COLOR 1,C(1),C(1),1,1
510 RESTORE 520:GOSUB START
520 DATA 282,12,282,32,268,38
530 DATA 248,38,242,58,262,58
540 DATA 288,52,282,72,288,92
550 COLOR 1,C(2),C(2)
560 RESTORE 570:GOSUB START
570 DATA 262,12,262,32,288,38
580 DATA 268,52,282,58,248,78
590 DATA 262,72,268,92,242,98
600 COLOR 1,C(3),C(3)
610 RESTORE 620:GOSUB START
620 DATA 242,12,242,32,248,52
630 DATA 242,72,268,78,288,78
640 DATA 248,92,262,98,282,98
650 COLOR 1,8,8
660 FOR A=56 TO 112 STEP 14
670 LINEF 64,A,128,A
680 NEXT A
690 FOR A=64 TO 128 STEP 16
700 LINEF A,56,A,112
710 NEXT A
720 FOR ROW=1 TO 4
730 FOR CLM=1 TO 4
740 X1=48+16*CLM:Y1=42+14*ROW
750 BX=X1:BY=Y1
760 COLOR 1,1,1:GOSUB BOX
770 GOSUB CHECK
780 IF FLAG=1 AND MB=2 THEN 810
790 GOSUB SQUARE
800 IF FLAG=1 AND MB=1 THEN GOSUB REPE
AT
810 COLOR 1,8,8:GOSUB BOX
820 NEXT CLM,ROW
830 IF FLAG=1 THEN 880
840 FOR ROW=1 TO 4:FOR CLM=1 TO 4
850 X1=48+16*CLM:Y1=42+14*ROW
860 GOSUB REPEAT
870 NEXT CLM,ROW
880 REM CHANGE
890 GOTOXY 22,12:?"SELECT WITH"
900 GOTOXY 22,13:?"LEFT BUTTON"
910 GOTOXY 22,14:?"OR FOR NO"
920 GOTOXY 22,15:?"CHANGE PRESS"
930 GOTOXY 22,16:?"RIGHT BUTTON"
940 FLAG=1
950 GOTO 720
960 SQUARE:
970 IF FLAG=0 THEN 1040
980 COLOR 1,9,9:GOSUB BOX
990 LINEF BX,BY,BX+16,BY+14
1000 LINEF BX+16,BY,BX,BY+14
1010 FILL BX+2,BY+5:FILL BX+14,BY+6
1020 FILL BX+5,BY+2:FILL BX+5,BY+12
1030 COLOR 1,8,8:GOSUB BOX
1040 PAT$=QP$(ROW,CLM)
1050 T=QT(ROW,CLM):B=QB(ROW,CLM)
1060 IF PAT$<>"F" THEN 1090
1070 COLOR 1,T,T:FILL BX+2,BY+2
1080 GOTO 1160
1090 COLOR 1,8,8
1100 IF PAT$<>"RL" THEN 1130
1110 LINEF BX+16,BY,BX,BY+14
```

```
1120 GOTO 1140
1130 LINEF BX,BY,BX+16,BY+14
1140 COLOR 1,T,T:FILL BX+5,BY+2
1150 COLOR 1,B,B:FILL BX+5,BY+12
1160 RETURN
1170 CHECK:
1180 M#=GB:G2=PEEK(M#+12)
1190 GEMSYS(79)
1200 MB=PEEK(G2+6):IF MB=0 THEN 1190
1210 IF FLAG=1 AND MB=2 THEN RETURN
1220 MX=PEEK(G2+2):MY=PEEK(G2+4)-21
1230 IF MX<240 OR MX>290 THEN 1190
1240 IF MY<10 OR MY>100 THEN 1190
1250 IF MX>250 AND MX<260 THEN 1190
1260 IF MX>270 AND MX<280 THEN 1190
1270 IF MY>20 AND MY<30 THEN 1190
1280 IF MY>40 AND MY<50 THEN 1190
1290 IF MY>60 AND MY<70 THEN 1190
1300 IF MY>80 AND MY<90 THEN 1190
1310 YY=INT((MX-220)/20)
1320 XX=INT((MY+10)/20)
1330 PAT$=S$(XX,YY):QP$(ROW,CLM)=PAT$
1340 QT(ROW,CLM)=TOP(XX,YY)
1350 QB(ROW,CLM)=BOT(XX,YY)
1360 RETURN
1370 SETUP:
1380 FLAG=0
1390 RESTORE 1400
1400 DATA F,F,F,RL,RL,RL,LR,LR,LR
1410 DATA RL,RL,RL,LR,LR,LR
1420 FOR X=1 TO 5:FOR Y=1 TO 3
1430 READ S$(X,Y):NEXT Y,X
1440 FOR X=1 TO 5
1450 TOP(X,1)=C(3):TOP(X,2)=C(2):TOP(X
,3)=C(1)
1460 NEXT Y
1470 BOT(1,1)=C(3):BOT(1,2)=C(2):BOT(1
,3)=C(1)
1480 BOT(2,1)=C(1):BOT(2,2)=C(1):BOT(2
,3)=C(2)
1490 BOT(3,1)=C(1):BOT(3,2)=C(1):BOT(3
,3)=C(2)
1500 BOT(4,1)=C(2):BOT(4,2)=C(3):BOT(4
,3)=C(3)
1510 BOT(5,1)=C(2):BOT(5,2)=C(3):BOT(5
,3)=C(3)
1520 RETURN
1530 START:
1540 FOR T=1 TO 9
1550 READ X,Y:FILL X,Y
1560 NEXT T
1570 RETURN
1580 BOX:
1590 LINEF BX,BY,BX+16,BY
1600 LINEF BX+16,BY,BX+16,BY+14
1610 LINEF BX,BY,BX,BY+14
1620 LINEF BX,BY+14,BX+16,BY+14
1630 RETURN
1640 REPEAT:
1650 FOR XX=X1-64 TO X1+64 STEP 64
1660 FOR YY=Y1-56 TO Y1+56 STEP 56
1670 BX=XX:BY=YY
1680 COLOR 1,8,8:GOSUB BOX
1690 GOSUB SQUARE
1700 NEXT YY,XX
1710 RETURN
1720 END
●
```

ST CHECKSUM DATA

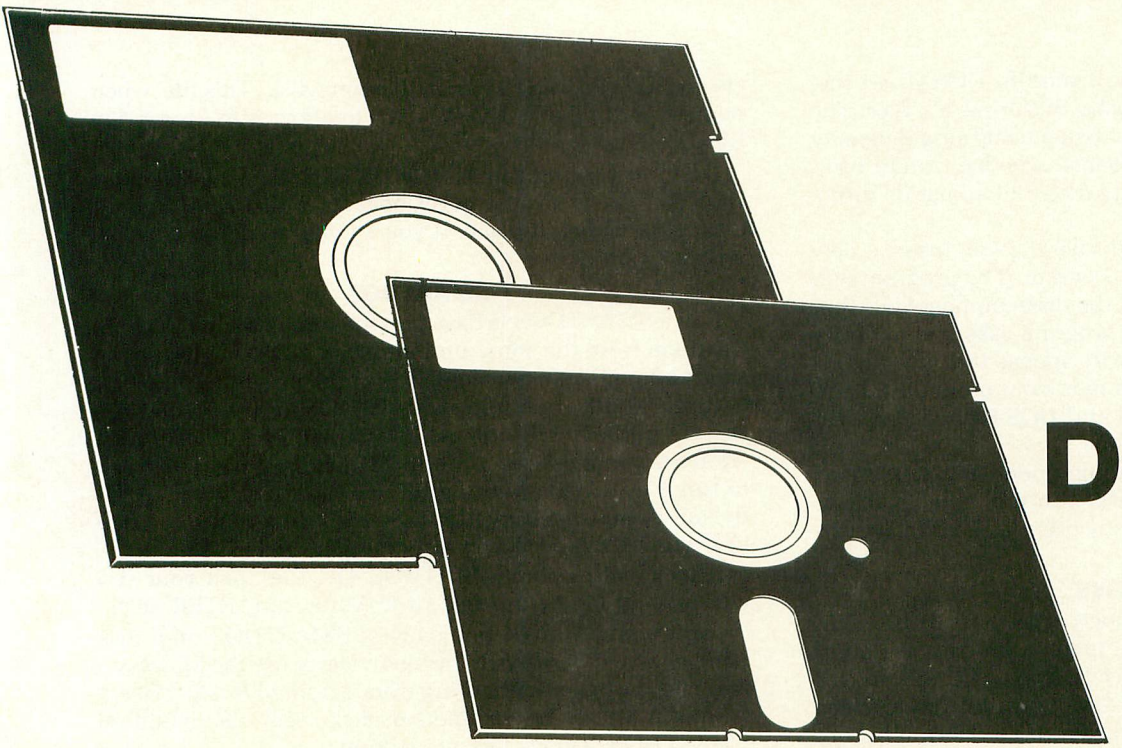
```
 10 data  572, 11, 13, 364, 520, 606,
614, 120, 5, 827, 3652
 110 data  236, 555, 4, 554, 412, 334,
 480, 499, 308, 373, 3755
 210 data  273, 198, 486, 731, 59, 305
, 967, 780, 350, 260, 4409
 310 data  478, 954, 31, 228, 556, 388
, 62, 77, 46, 293, 3113
 410 data  300, 869, 861, 464, 70, 43,
 588, 881, 479, 597, 5152
 510 data  636, 84, 89, 110, 239, 666,
 95, 132, 123, 230, 2404
 610 data  640, 52, 133, 124, 685, 983
, 311, 281, 8, 274, 3491
 710 data  262, 67, 19, 432, 943, 81,
917, 583, 151, 4, 3459
 810 data  89, 163, 511, 835, 437, 144
, 178, 790, 823, 910, 4880
 910 data  442, 921, 47, 170, 422, 477
, 646, 115, 177, 179, 3596
 1010 data  133, 208, 191, 687, 127, 9
78, 748, 562, 744, 242, 4620
 1110 data  183, 555, 185, 757, 816, 4
46, 301, 765, 806, 147, 4961
 1210 data  344, 170, 988, 649, 935, 9
42, 774, 781, 788, 788, 7159
 1310 data  703, 634, 622, 605, 564, 4
52, 390, 253, 29, 93, 4345
 1410 data  410, 520, 232, 30, 266, 39
5, 950, 950, 957, 965, 5675
 1510 data  972, 454, 370, 37, 779, 38
3, 459, 51, 921, 483, 4909
 1610 data  912, 481, 458, 473, 176, 1
97, 64, 214, 236, 927, 4138
 1710 data  459, 934, 1393
●
```


WHAT IS ST-CHECK?

Most ST BASIC program listings in this magazine are followed by a table of numbers appearing as data statements, called "ST CHECKSUM DATA." These numbers are to be used in conjunction with **ST-Check** (which appeared in **ANALOG Computing** issue 41).

ST-Check, written by Clayton Walnum, is designed to find and correct typing errors when readers are entering programs from the magazine. For those readers who would like copies of the article, you may send for back issue 41 of **ANALOG Computing** for $4.00.

ANALOG COMPUTING
P.O. Box 625, Holmes, PA 19045



DOS CD

Double your disk capacity

by Angelo Giambra

I was really excited when I purchased my new Astra 2001 disk drive. The first thing I did was begin to convert all my disks to double-density format, figuring I'd cut my disk usage in half.

Imagine my surprise when, while creating my very first double-density disk, I got an obscure disk error: *Error 169!* I checked to see if I had run out of room on the disk. No, I still had over 200 sectors left. As a last resort (naturally), I consulted the manual. Error 169 translation: Disk Directory Full.

Of course! You can only put 64 files on a disk. I'd never gotten this error before because it's practically impossible to get 64 files on a single-density disk, unless you have a lot of really small files.

128 vs. 256

Something seemed wrong here. Having written several utilities which modified DOS, I was pretty familiar with the File Management System (FMS). I knew that the 64-file limit was imposed because of the way the directory was implemented. The directory consists of eight disk sectors, beginning at sector 361. Each directory entry uses 16 bytes (see Figure 1), so in a 128-byte sector you can fit eight entries. Eight sectors times eight entries makes 64 files. Fine.

But double-density disks use 256-byte sectors. Wasn't it reasonable to assume that the 256-byte directory sectors on a double-density disk could fit 16 entries, hence 128 files per disk?

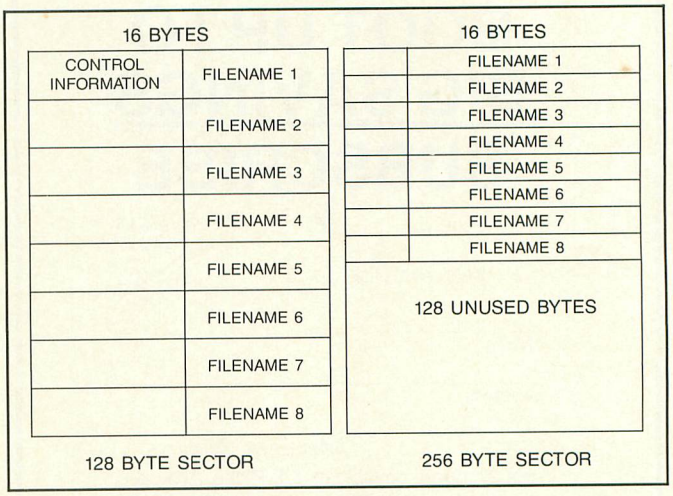


Figure 1

Wrong! Neither DOSXL, nor SMARTDOS, take advantage of the extra room in the directory. Believe it or not, on a double-density disk, half of every directory sector is unused. The FMS treats these sectors as if they were still single density.

The solution

Well, I decided, it was time for **DOS CD** (Capacity Doubler). There are several versions described in this article, depending on which DOS you use (DOSXL or SMARTDOS). I'll describe each one and tell you how to implement it.

The program in Listing 1 will modify DOSXL so that double-density disks will be capable of storing 128 files. The 64-file limit will still be in effect when using single-density disks; the modified FMS will sense whether you're writing to a single- or double-density disk and adjust its directory handling accordingly.

First, key in the program and save it. Now insert a new disk into any drive and run the program. The program will prompt: *WHICH DRIVE?* Key in the drive number you want **DOS CD** installed in. After the program asks you to verify the drive number, a warning will appear that the disk is about to be formatted. If everything is okay, press the START key. The disk will be formatted and **DOS CD** will be written out.

At this point, you will be running under **DOS CD**. It isn't necessary to reboot from the disk. You will now be able to store 128 files on this disk and any other disk you have formatted using **DOS CD**.

DOSXL.SYS

If you own one of OSS's supercartridges (BASIC XL, ACTION, MAC/65), you may be using a special version of DOSXL called DOSXL.SYS. This version of DOSXL moves itself up into the area of memory "underneath" the cartridge and frees up about 3 to 5K of memory. If you normally boot from a disk containing DOSXL.SYS when using your supercartridge, you may use the program in Listing 2 to create an AUTORUN.SYS file on the same disk. This file, when booted, will modify DOSXL.SYS to allow it to access 128 files.

(Note: If you put the AUTORUN.SYS file on a disk with DOSXL.SYS, you must never boot from that disk without a supercartridge in place. If you attempt to do so, the disk will not boot.)

SMARTDOS

Since SMARTDOS is based on the same FMS as DOSXL, it suffers from the same limitations of DOSXL. (It's not so smart after all!) No problem! Key in the program in Listing 3. Insert a disk containing SMARTDOS in drive 1 and run the program. It will create a file called SMARTDOS.AR1. (Note that your version of SMARTDOS must be configured to run *.AR1 upon booting.) If you have not modified the default options, you're all set. See your instruction manual if you're unsure about the configuration.

After creating the SMARTDOS.AR1 file, turn your system off and boot from this disk. Voilà! SMARTER DOS.

(Note: Save at least one copy of SMARTDOS on a disk with no SMARTDOS.AR1 file on it. Never use the Make System Files option after booting from modified SMARTDOS.) If you want to create another disk with the modified SMARTDOS, follow this procedure:

First, boot from the disk containing only SMARTDOS (no SMARTDOS.AR1). Choose the Make System Files option to create a new copy of SMARTDOS on another disk. Finally, copy SMARTDOS.AR1 to the new disk.

Also, the modification for SMARTDOS uses a small portion of page six—addresses $6CB to $6FF. Never run any software which writes over these addresses while running under modified SMARTDOS.

Compatibility

Which brings us to compatibility. If you boot from **DOS CD**, you'll be able to read all your single-density disks just fine. And you'll be able to write to them with no problem either. This is because **DOS CD** senses the density and treats single-density disks the same as regular DOS does.

But if you do a directory on your standard double-density disks, **DOS CD** will show only the first eight files. And those are the only files you'll be able to access. Why? Because a directory command causes FMS to search the directory sectors and print the filenames it finds there. FMS knows when it reaches the last entry in the directory because instead of a filename, it finds a binary zero in the status byte.

DOS CD begins searching the directory and prints the first eight files, then goes into the formerly unused second half of the directory sectors and finds—you guessed it—binary zeroes. So it stops.

Conversion

You may want to convert your present double-density disks so that **DOS CD** can access the remaining files. Once these disks are converted, **DOS CD** will also be able to store 128 files on these disks as well. But now you'll be in an opposite situation. Standard DOS will only be able to access the first eight files on a converted disk. Unlike **DOS CD**, standard DOS will be able to see all the files on the disk, but if you try to access anything beyond the first eight,

BOOT UP TO BIG SAVINGS

SUBSCRIBE

☐ 1 Year.....$28..... Save $14
MCEYY

☐ 1 Year with Disk.....$105
DCEYY

Name

Address

City State Zip

Make checks payable to: L.F.P. Inc. Allow 4-6 weeks for delivery.

☐ Payment Enclosed ☐ Bill Me
☐ Charge My ☐ Visa ☐ MC
_____ Exp ____

Signature
FOREIGN—Add $7 per year
MONEY BACK if not delighted
Analog
P.O. Box 16927
N. Hollywood, CA 91615

you'll get an Error 164 (or System Error in DOSXL). I'll explain why later.

When you run the program in Listing 4, it will create a machine language file on your disk called CONVERT. Before running CONVERT, I *strongly* recommend that you back up any disks you want to convert. If you made a mistake when typing in the program, you could really mess up your disks, and I don't want a lot of nasty mail!

Okay, all backed up? Let's proceed. Load the convert program using DOSXL's LOAD command. The screen will clear and you'll see the message: *PRESS START.*

Insert the disk you wish to convert in the drive designated as 1 and press START. As the conversion progresses, you'll see the name of each file the CONVERT program accesses. Following is an explanation of what's going on.

The hows and whys

All files on your disk are assigned a file number by the FMS when they are written to disk. This number simply corresponds to their position in the directory. The first file is assigned file number 0, the tenth file is file number 9, and so on.

Normally, the ninth file falls in the second directory sector since there are only eight entries per sector. In **DOS CD**, the ninth file falls into the second half of the first sector. The first file in the second directory sector, therefore, should be assigned file number 15.

The CONVERT utility reads each data sector of every file on your disk and reassigns a new file number based on the above.

Then, it does the following. If there were files in the first half of the current directory sector, CONVERT changes the eight status bytes in the second half of the directory to hex $80 and rewrites the directory sector. Why? Because hex $80 is the Deleted File status. This tricks **DOS CD** into thinking this is an entry for a file that has since been deleted. When you issue a directory command, deleted files are simply passed over, but the directory search is continued.

As you add files to the converted disk, **DOS CD** will use these directory positions to catalog the new files. That's how you'll get the extra 64 files on your converted disks.

Which explains why standard DOS gets an error when trying to access anything but the first eight files on a converted disk. Standard DOS computes what the file number should be for each file it accesses and compares it to what is actually written on each data sector. The computation is based on eight per directory sector, so a mismatch occurs when it encounters 16 entries per sector.

A challenge

It is possible to design even another version of **DOS CD**—one which will work on the XL machines which use the DOSXL.XL version of DOSXL.SYS. Like supercartridge DOSXL.SYS, this version boots up into the area above BASIC and frees up RAM in low memory. I wasn't able to implement this version of **DOS CD** since my version of DOSXL.SYS hangs the system whenever I try to use it on my XL. Either I have a bad copy, or there's a bug in this version. I'll leave the conversion to any hackers out there who want to accept a challenge.

If you store a lot of little files on disk, you're really going to like **DOS CD**. You'll be able to get your money's worth out of every disk!

Angelo Giambra is a specialist in Systems Software for General Electric in Largo, Florida. He has been an avid hobbyist since he bought his first 800 four years ago. He enjoys writing machine language utilities and extensions to the OS and DOS.

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

Gear Up Your Disk Drive For Big Savings!

Save $14 Off The Cover Price

ANALOG

☐ 1 Year.....$28.00 Save $14! MCEYY

☐ 1 Year with Disk.....$105.00 DCEYY

FOREIGN: Add $7 Per Year
Money Back If Not Delighted!

☐ Payment Enclosed ☐ Bill Me

☐ Charge My ☐ Visa ☐ MC

_____ Exp. __

Signature

Name

City State Zip

Make Checks Payable to: L.F.P. Inc. Allow 4-6 weeks for delivery of first issue.

Analog
P.O. Box 16927
N. Hollywood, CA 91615

Listing 1
BASIC listing

```
ZM 10 REM ***************
KP 20 REM *             *
LK 30 REM *  DOS CD     *
NP 40 REM *    by       *
QP 50 REM * A. GIAMBRA *
KT 60 REM *             *
ZS 70 REM ***************
BF 80 REM
MR 90 DIM I(4),AN$(1),FN$(6),D$(10)
YH 100 FN$="D"
YX 110 I(0)=5377
XF 120 I(1)=2952
YL 130 I(2)=2458
ZH 140 I(3)=3873
WX 150 I(4)=3480
KX 160 TRAP 210
OR 170 READ A:IF A=-1 THEN IX=IX+1:X=0:GO
   TO 170
KH 180 POKE I(IX)+X,A
XQ 190 X=X+1
OG 200 GOTO 170
ZF 210 GRAPHICS 18
IK 220 POSITION 1,3
DQ 230 ? #6;"ENTER DRIVE NUMBER"
PJ 240 OPEN #1,4,0,"K:"
NA 250 GET #1,CH
MQ 260 UNIT=VAL(CHR$(CH))
TB 270 FN$(2)=CHR$(CH)
LU 280 POSITION 3,6
AC 290 ? #6;"DRIVE NUMBER ";CHR$(CH)
LV 300 POSITION 3,7
CD 310 ? #6;"CORRECT? (Y/N)"
MV 320 GET #1,CH
JN 330 IF CHR$(CH)<>"Y" THEN CLOSE #1:GOT
   O 210
LH 340 CLOSE #1
ZO 350 GRAPHICS 18
KX 360 POSITION 5,3
UR 370 ? #6;"warning"
LF 380 POSITION 3,5
PW 390 ? #6;"disk will be"
LU 400 POSITION 4,6
QI 410 ? #6;"formatted"
NU 420 FOR I=1 TO 5
EY 430 SOUND 0,100,12,8
NO 440 POKE 711,72
IK 450 FOR X=1 TO 100:NEXT X
WA 460 SOUND 0,0,0,0
QS 470 POKE 711,68
IQ 480 FOR X=1 TO 100:NEXT X
GM 490 NEXT I
RN 500 ? #6:? #6;"PLEASE PRESS START"
WI 510 IF PEEK(53279)<>6 THEN 510
KB 520 FN$(3)=":"
ZM 530 GRAPHICS 18
KJ 540 POSITION 3,4
AW 550 ? #6;"FORMATTING ";FN$
QV 560 D$=FN$
GB 570 D$(3)=":DOS.SYS"
CM 580 FN$(4)="*.*"
IB 590 XIO 254,#1,0,0,FN$
JO 600 POSITION 2,4
OT 610 ? #6;"INSTALLING dos cd"
XN 620 POKE 7424,49
CZ 630 OPEN #1,8,0,D$
DY 640 DATA 172,254,18,136,240,6,160,240,
   169,144,208,4,160,112,169,16,141,58,15
   ,140,45,15,169,255
SE 650 DATA 141,2,19,76,38,15,32,41,21,17
   3,7,19,153,215,7,96,138,74,74,74,74,16
   8,96,172
NR 660 DATA 254,18,136,208,2,41,127,141,5
   ,19,96,-1
JK 670 DATA 32,41,21,185,215,7,72,172,254
   ,18,136,240,1,74,32,43,21,141,6,19,104
   ,10,10,10
UA 680 DATA 10,32,48,21,76,110,16,-1
KX 690 DATA 32,31,21,-1
BH 700 DATA 76,1,21,-1
EA 710 DATA 208,-1
```

Listing 2
BASIC listing

```
JA 10 REM ***************
HY 20 REM *  DOS CD FOR *
OT 30 REM *  DOSXL.SYS  *
LI 40 REM *     BY      *
SB 50 REM * A. GIAMBRA  *
JF 60 REM ***************
EC 70 DATA 0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,
   0,0,10,11,12,13,14,15
RU 80 DIM DAT$(96),HEX(22):FOR X=0 TO 22:
   READ N:HEX(X)=N:NEXT X:LINE=220:RESTOR
   E 220:TRAP 160:? "KCHECKING DATA"
QK 90 TOTAL=0:LINE=LINE+10:POSITION 2,2:?
    "LINE:";LINE:READ DAT$:IF LEN(DAT$)<>
   96 THEN 210
TO 100 DATLIN=PEEK(183)+PEEK(184)*256:IF
   DATLIN<>LINE THEN ? "LINE ";LINE;" MIS
   SING!":END
JP 110 FOR X=1 TO LEN(DAT$)-1 STEP 2:D1=A
   SC(DAT$(X,X))-48:D2=ASC(DAT$(X+1,X+1))
   -48:BYTE=HEX(D1)*16+HEX(D2)
ZS 120 IF PASS=2 THEN PUT #1,BYTE:NEXT X:
   READ CHKSUM:GOTO 90
DH 130 TOTAL=TOTAL+HEX(D1)+HEX(D2):NEXT X
GB 140 READ CHKSUM:IF TOTAL=CHKSUM THEN 9
   0
MW 150 GOTO 220
JW 160 IF PEEK(195)<>6 AND PEEK(195)<>5 T
   HEN 210
OR 170 IF PASS=0 THEN ? "KDATA STATMENTS
   CORRECT":? :? "INSERT DISK CONTAINING
   DOSXL.SYS":? "THEN PRESS ANY KEY"
GO 180 IF PEEK(764)=255 THEN 180
DL 190 IF PASS=0 THEN OPEN #1,8,0,"D1:AUT
   ORUN.SYS":PASS=2:LINE=220:RESTORE 220:
   TRAP 200:? "KCREATING FILE":GOTO 90
HY 200 CLOSE #1:END
JL 210 IF LEN(DAT$)=48 AND LINE=260 THEN
   TRAP 170:GOTO 100
MI 220 ? "BAD DATA: LINE ";LINE:END
UJ 230 DATA FFFF090A430AAC25AF88F006A0F0A
   990D004A070A9108D5CAB8C4FABA9FF8D29AF4
   C48AB20310AAD2EAF99000A608A4A,714
CG 240 DATA 4A4A4AA860AC25AF88D002297F8D2
   CAF60AAA7C8A720310AB9000A48AC25AF88F00
   14A20330A8D2DAF680A0A0A0A2038,628
KO 250 DATA 0A4C90ACBAA5BCA520270A43AB45A
   B4C090ABAA9BAA9D0440A640A200000A9528DE
   702A90A8DE80260A50C8D450AA50D,623
KV 260 DATA 8D460AA9448850CA90A850D60E702E
   802520AE002E102520A,268
```

Listing 3
BASIC listing

```
JA 10 REM ***************
HY 20 REM *  DOS CD FOR *
WC 30 REM *   SMARTDOS  *
LI 40 REM *     BY      *
SB 50 REM * A. GIAMBRA  *
JF 60 REM ***************
EC 70 DATA 0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,
   0,0,10,11,12,13,14,15
RU 80 DIM DAT$(96),HEX(22):FOR X=0 TO 22:
   READ N:HEX(X)=N:NEXT X:LINE=220:RESTOR
   E 220:TRAP 160:? "KCHECKING DATA"
```

```
QK 90 TOTAL=0:LINE=LINE+10:POSITION 2,2:?
   "LINE:";LINE:READ DAT$:IF LEN(DAT$)<>
   96 THEN 210
TO 100 DATLIN=PEEK(183)+PEEK(184)*256:IF
   DATLIN<>LINE THEN ? "LINE ";LINE;" MIS
   SING!":END
JP 110 FOR X=1 TO LEN(DAT$)-1 STEP 2:D1=A
   SC(DAT$(X,X))-48:D2=ASC(DAT$(X+1,X+1))
   -48:BYTE=HEX(D1)*16+HEX(D2)
ZS 120 IF PASS=2 THEN PUT #1,BYTE:NEXT X:
   READ CHKSUM:GOTO 90
DH 130 TOTAL=TOTAL+HEX(D1)+HEX(D2):NEXT X
GB 140 READ CHKSUM:IF TOTAL=CHKSUM THEN 9
   0
MW 150 GOTO 220
JW 160 IF PEEK(195)<>6 AND PEEK(195)<>5 T
   HEN 210
YL 170 IF PASS=0 THEN ? "}DATA STATMENTS
   CORRECT":? :? "INSERT DISK CONTAINING
   SMARTDOS":? "THEN PRESS ANY KEY"
GO 180 IF PEEK(764)=255 THEN 180
TY 190 IF PASS=0 THEN OPEN #1,8,0,"D1:SMA
   RTDOS.AR1":PASS=2:LINE=220:RESTORE 220
   :TRAP 200:? "}CREATING FILE":GOTO 90
HY 200 CLOSE #1:END
GU 210 IF LEN(DAT$)=46 AND LINE=250 THEN
   TRAP 170:GOTO 100
MI 220 ? "BAD DATA: LINE ";LINE:END
OT 230 DATA FFFFC506FF06ACFE1288F006A0F0A
   990D004A070A9108D3A0F8C2D0FA9FF8D02134
   C260F20ED06AD071399BC06608A4A,685
OW 240 DATA 4A4A4AA860ACFE1288D002297F8D0
   51360880BA60B20ED06B9BC0648ACFE1288F00
   14A20EF068D0613680A0A0A0A20F4,613
EB 250 DATA 064C6E109A099C0920E306210F230
   F4CC506980D980DD0,276
```

●

Listing 4
BASIC listing

```
JA 10 REM ****************
TY 20 REM *  DOSCD FILE *
OY 30 REM *  CONVERTER  *
LI 40 REM *     BY      *
SB 50 REM * A. GIAMBRA  *
JF 60 REM ****************
EC 70 DATA 0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,
   0,0,10,11,12,13,14,15
RU 80 DIM DAT$(96),HEX(22):FOR X=0 TO 22:
   READ N:HEX(X)=N:NEXT X:LINE=220:RESTOR
   E 220:TRAP 160:? "}CHECKING DATA"
QK 90 TOTAL=0:LINE=LINE+10:POSITION 2,2:?
   "LINE:";LINE:READ DAT$:IF LEN(DAT$)<>
   96 THEN 210
TO 100 DATLIN=PEEK(183)+PEEK(184)*256:IF
   DATLIN<>LINE THEN ? "LINE ";LINE;" MIS
   SING!":END
JP 110 FOR X=1 TO LEN(DAT$)-1 STEP 2:D1=A
   SC(DAT$(X,X))-48:D2=ASC(DAT$(X+1,X+1))
   -48:BYTE=HEX(D1)*16+HEX(D2)
ZS 120 IF PASS=2 THEN PUT #1,BYTE:NEXT X:
   READ CHKSUM:GOTO 90
DH 130 TOTAL=TOTAL+HEX(D1)+HEX(D2):NEXT X
GB 140 READ CHKSUM:IF TOTAL=CHKSUM THEN 9
   0
MW 150 GOTO 220
JW 160 IF PEEK(195)<>6 AND PEEK(195)<>5 T
   HEN 210
RT 170 IF PASS=0 THEN ? "}DATA STATMENTS
   CORRECT":? :? "PRESS ANY KEY"
GO 180 IF PEEK(764)=255 THEN 180
ZW 190 IF PASS=0 THEN OPEN #1,8,0,"D1:CON
   VERT.OBJ":PASS=2:LINE=220:RESTORE 220:
   TRAP 200:? "}CREATING FILE":GOTO 90
HY 200 CLOSE #1:END
KL 210 IF LEN(DAT$)=84 AND LINE=360 THEN
   TRAP 170:GOTO 100
MI 220 ? "BAD DATA: LINE ";LINE:END
WX 230 DATA FFFF0045374544495348204F4E5
   6455253494F4E5554494C49545950524553532
   0D3D4C1D2D4E3EFEEF6E5F2F3E9EF,665
UC 240 DATA EE20E9EE2020F0F2EFE7F2E5F3F33
   9453B45533A403C473748A210A9399D4403A94
   59D4503A9039D4203A9089D4A03A9,633
KW 250 DATA 029D4B032056E4A9028554A903855
   5A9008556A9009D4403A9459D4503A90F9D480
   3A9009D490320CE48A9079D4803A9,568
FW 260 DATA 0F9D4403A9459D4503A9038554A90
   6855520CE48A90B9D4803A9169D4403A9459D4
   503A9058554A904855520CE48AD1F,601
CD 270 DATA D0C906D0F9A9028554A9038555A92
   19D4403A9459D4503A90F9D480320CE48A9038
   554A9058555A9309D4403A9459D45,598
DH 280 DATA 03A9089D480320CE48A96985B3A90
   185B4A90085B085B285BD209148A5B38D0A03A
   5B48D0B03A93C8D0403A9458D0503,593
RH 290 DATA A952A0408D02038C030320B048A90
   085BAA4B2B93C452C3B45D00BC900D00CA9809
   93C45D038483349058 5BA20D748E6,565
IQ 300 DATA BD18A5B26910B00485B290D9A5BAF
   03FA90085BA85B2209148A5B38D0A03A5B48D0
   B03A93C8D0403A9458D0503A957A0,609
IO 310 DATA 808D02038C030320B04818E6B3D00
   2E6B4E6B0A5B0C90890034C8E484CFC476C0A0
   0A9038D3845A9018D0103A931A00F,564
GD 320 DATA 8D00038C0603A901A0008D09038C0
   803602059E41018CE38453013AD0203C952D00
   4A240D002A2808E03034CB04860A9,466
GG 330 DATA 0B9D42032056E460A210984818694
   19D4403A94569009D4503A90B9D4803A905855
   4A904855520CE4868A8C8C8C8B93C,586
SP 340 DATA 4585B5C8B93C4585B6209148A5B58
   D0A03A5B68D0B03A93C8D0403A9468D0503A95
   2A0408D02038C030320B048A5BD0A,574
GB 350 DATA 0A85B7AD3934497E494729034805B
   78D3947209148A5B58D0A03A5B68D0B03A93C8
   D0403A9468D0503A957A0808D0203,585
UO 360 DATA 8C030320B048A9085B785B86885B
   6F002E6B7AD3A4785B5F002E6B7A5B7F0034C0
   84960E002E1023C47,516
```

●

Listing 5
Assembly listing

```
          .OPT NO LIST
;*****************
;*               *
;*    DOS CD     *
;*      by       *
;*  A. Giambra   *
;*               *
;*****************
;
;DOS EQUATES
;
DRVTYP    =    $12FE    ;1=SINGLE 2=DOUBLE
BRANCH    =    $0F3A    ;BPL OR BCC
LENGTH    =    $0F2D    ;8 OR 16 ENTRIES
CURFCB    =    $1301    ;CURRENT FCB
DHOLES    =    $1302    ;DIRECTORY HOLE
CDIRD     =    $1305    ;DIR DISPLACEMENT
CDIRS     =    $1306    ;DIR SECTOR (1-8)
SFNUM     =    $1307    ;FILE NUMBER
DOSTAB    =    $07D7    ;DOS TABLE
ENTRDOS   =    $0F26    ;RE-ENRTY POINT
RDDIR     =    $106E    ;READ DIRECTORY
;
          *=   $1501
FIXDIR
          LDY  DRVTYP   ;GET DRIVE TYPE
          DEY           ;SINGLE DENSITY?
          BEQ  SNGLE    ;YEP
          LDY  #$F0     ;16 ENTRIES
          LDA  #$90     ;BCC
          BNE  EXITA
```

```
SNGLE
          LDY #$70     ;8 ENTRIES
          LDA #$10     ;BPL
EXITA
          STA BRANCH   ;MODIFY BRANCH
          STY LENGTH   ;MODIFY LENGTH
          LDA #$FF     ;BECAUSE WE
          STA DHOLES   ;DESTROYED THIS
          JMP ENTRDOS  ;NOW GET BACK
SAVESECT
          JSR SHIFT
          LDA SFNUM    ;GET FILE NUMBER
          STA DOSTAB,Y ;SAVE IT
          RTS          ;GET BACK
SHIFT
          TXA
          LSR A
SH1
          LSR A        ;MAKE IT AN INDEX
          LSR A        ;INTO OUR TABLE
          LSR A
          TAY
          RTS
MASK
          LDY DRVTYP   ;DRIVE TYPE
          DEY          ;SINGLE?
          BNE DOUBLE
          AND #$7F     ;MASK HIGH BIT
DOUBLE
          STA CDIRD    ;GOT DISPLACEMENT
          RTS
;
;WE MUST REPLACE THE RRDIR
;ROUTINE WITH OUR OWN
;THIS ROUTINE FIGURES OUT
;THE DIRECTORY SECTOR AND
;DISPLACEMENT USING THE
;FILE NUMBER
;
          *=  $0B88
          JSR SHIFT    ;GET TABLE INDEX
          LDA DOSTAB,Y ;LOAD FILE NO.
          PHA          ;SAVE IT
          LDY DRVTYP   ;GET DRIVE TYPE
          DEY          ;SINGLE DENSITY?
          BEQ SING     ;YEP?
          LSR A        ;SHIFT RIGHT
SING
          JSR SH1      ;SHIFT 3 TIMES
          STA CDIRS    ;GOT SECTOR
          PLA          ;RESTORE NUMBER
          ASL A        ;SHIFT LEFT
          ASL A        ;FOUR TIMES
          ASL A        ;TO EQUAL
          ASL A        ;DISPLACEMENT
          JSR MASK     ;CHECK DRIVE TYPE
          JMP RDDIR    ;GO TO DOS
          *=  $099A
          JSR SAVESECT ;INTERCEPT DOS
          *=  $0F21
          JMP FIXDIR   ;DITTO
          *=  $0D98
          .BYTE $D0    ;CHANGE BPL TO BNE
```

Listing 6
Assembly listing

```
          .OPT NO LIST
;*******************
;*                 *
;*  DOSXL.SYS      *
;*  CONVERSION     *
;*      by         *
;*  A. Giambra     *
;*                 *
;*******************
;
;DOS EQUATES
;
DRVTYP    =   $AF25    ;1=SINGLE 2=DOUBLE
BRANCH    =   $AB5C    ;BPL OR BCC
LENGTH    =   $AB4F    ;8 OR 16 ENTRIES
CURFCB    =   $AF28    ;CURRENT FCB
DHOLES    =   $AF29    ;DIRECTORY HOLE
CDIRD     =   $AF2C    ;DIR DISPLACEMENT
CDIRS     =   $AF2D    ;DIR SECTOR (1-8)
SFNUM     =   $AF2E    ;FILE NUMBER
ENTRDOS   =   $AB48    ;RE-ENRTY POINT
RDDIR     =   $AC90    ;READ DIRECTORY
DOSINI    =   $0C      ;DOS VECTOR
LOMEM     =   $02E7    ;LOW MEMORY
;
          *=  $0A00
DOSTAB    *=  *+9
FIXDIR
          LDY DRVTYP   ;GET DRIVE TYPE
          DEY          ;SINGLE DENSITY?
          BEQ SNGLE    ;YEP
          LDY #$F0     ;16 ENTRIES
          LDA #$90     ;BCC
          BNE EXITA
SNGLE
          LDY #$70     ;8 ENTRIES
          LDA #$10     ;BPL
EXITA
          STA BRANCH   ;MODIFY BRANCH
          STY LENGTH   ;MODIFY LENGTH
          LDA #$FF     ;BECAUSE WE
          STA DHOLES   ;DESTROYED THIS
          JMP ENTRDOS  ;NOW GET BACK
SAVESECT
          JSR SHIFT
          LDA SFNUM    ;GET FILE NUMBER
          STA DOSTAB,Y ;SAVE IT
          RTS          ;GET BACK
SHIFT
          TXA
          LSR A
SH1
          LSR A        ;MAKE IT AN INDEX
          LSR A        ;INTO OUR TABLE
          LSR A
          TAY
          RTS
MASK
          LDY DRVTYP   ;CHECK DRIVE TYPE
          DEY          ;SINGLE DRIVE?
          BNE DOUBLE
          AND #$7F     ;MASK HIGH BIT
DOUBLE
          STA CDIRD    ;GOT DISPLACEMENT
          RTS
SVE       =   *
;
;WE MUST REPLACE THE RRDIR
;ROUTINE WITH OUR OWN
;THIS ROUTINE FIGURES OUT
;THE DIRECTORY SECTOR AND
;DISPLACEMENT USING THE
;FILE NUMBER
;
          *=  $A7AA
          JSR SHIFT    ;GET TABLE INDEX
          LDA DOSTAB,Y ;LOAD FILE NO.
          PHA          ;SAVE IT
          LDY DRVTYP   ;GET DRIVE TYPE
          DEY          ;SINGLE DENSITY?
          BEQ SING     ;YEP?
          LSR A        ;SHIFT RIGHT
SING
          JSR SH1      ;SHIFT 3 TIMES
          STA CDIRS    ;GOT SECTOR
          PLA          ;RESTORE NUMBER
          ASL A        ;SHIFT LEFT
```

```
        ASL A          ;FOUR TIMES
        ASL A          ;TO EQUAL
        ASL A          ;DISPLACEMENT
        JSR MASK       ;GET DISPLACEMENT
        JMP RDDIR      ;GO TO DOS
        *=  $A5BA
        JSR SAVESECT ;INTERCEPT DOS
        *=  $AB43
        JMP FIXDIR   ;DITTO
        *=  $A9BA
        .BYTE $D0    ;CHANGE BPL TO BNE
        *=  $VE
ADDR
        JSR $00      ;INITIALIZE DOS
RESET
        LDA # <INIT ;REESTABLISH
        STA LOMEM    ;LOW MEMORY
        LDA # >INIT
        STA LOMEM+1
        RTS
INIT
        LDA DOSINI   ;POINT OUR VECTOR
        STA ADDR+1   ;TO DOS INIT CODE
        LDA DOSINI+1
        STA ADDR+2
        LDA # <ADDR ;POINT DOS INIT
        STA DOSINI   ;TO OUR CODE
        LDA # >ADDR
        STA DOSINI+1
        RTS
        *=  $02E7
        .WORD INIT
        *=  $02E0
        .WORD INIT
```

Listing 7
Assembly listing

```
        .OPT NO LIST
; SMARTDOS MODIFICATION
; by
; A. Giambra
;
;
;*******************
;DOS MODIFICATIONS*
;*******************
;
DRVTYP   =   $12FE    ;1=SINGLE 2=DOUBLE
BRANCH   =   $0F3A    ;BPL OR BCC
LENGTH   =   $0F2D    ;8 OR 16 ENTRIES
CURFCB   =   $1301    ;CURRENT FCB
DHOLES   =   $1302    ;DIRECTORY HOLE
CDIRD    =   $1305    ;DIR DISPLACEMENT
CDIRS    =   $1306    ;DIR SECTOR (1-8)
SFNUM    =   $1307    ;FILE NUMBER
ENTRDOS  =   $0F26    ;RE-ENRTY POINT
RDDIR    =   $106E    ;READ DIRECTORY
;
         *=  $06BC
DOSTAB   *=  *+9
FIXDIR
         LDY DRVTYP  ;GET DRIVE TYPE
         DEY         ;SINGLE DENSITY?
         BEQ SNGLE   ;YEP
         LDY #$F0    ;16 ENTRIES
         LDA #$90    ;BCC
         BNE EXITA
SNGLE
         LDY #$70    ;8 ENTRIES
         LDA #$10    ;BPL
EXITA
         STA BRANCH  ;MODIFY BRANCH
         STY LENGTH  ;MODIFY LENGTH
         LDA #$FF    ;BECAUSE WE
         STA DHOLES  ;DESTROYED THIS
         JMP ENTRDOS ;NOW GET BACK
SAVESECT
         JSR SHIFT
         LDA SFNUM   ;GET FILE NUMBER
         STA DOSTAB,Y ;SAVE IT
         RTS
SHIFT
         TXA
         LSR A
SH1
         LSR A       ;MAKE IT AN INDEX
         LSR A       ;INTO OUR TABLE
         LSR A
         TAY
         RTS
MASK
         LDY DRVTYP  ;DRIVE TYPE
         DEY         ;SINGLE?
         BNE DOUBLE
         AND #$7F    ;MASK HIGH BIT
DOUBLE
         STA CDIRD   ;GOT DISPLACEMENT
         RTS
;
;WE MUST REPLACE THE RRDIR
;ROUTINE WITH OUR OWN
;THIS ROUTINE FIGURES OUT
;THE DIRECTORY SECTOR AND
;DISPLACEMENT USING THE
;FILE NUMBER
;
         *=  $0B88
         JSR SHIFT   ;GET TABLE INDEX
         LDA DOSTAB,Y ;LOAD FILE NO.
         PHA         ;SAVE IT
         LDY DRVTYP  ;GET DRIVE TYPE
         DEY         ;SINGLE DENSITY?
         BEQ SING    ;YEP?
         LSR A       ;SHIFT RIGHT
SING
         JSR SH1     ;SHIFT 3 TIMES
         STA CDIRS   ;GOT SECTOR
         PLA         ;RESTORE NUMBER
         ASL A       ;SHIFT LEFT
         ASL A       ;FOUR TIMES
         ASL A       ;TO EQUAL
         ASL A       ;DISPLACEMENT
         JSR MASK    ;GET DISPLACEMENT
         JMP RDDIR   ;GO TO DOS
         *=  $099A
         JSR SAVESECT ;INTERCEPT DOS
         *=  $0F21
         JMP FIXDIR  ;DITTO

         .BYTE $D0   ;CHANGE BPL TO BNE
```

Listing 8
Assembly listing

```
         .OPT NO LIST
;
;  ********************
;  *      DISK        *
;  *   CONVERSION     *
;  *       by         *
;  *   A. GIAMBRA     *
;  ********************
;
;OS EQUATES
;
         *=  $0340
IOCB
ICHID    *=  *+1     ;DEVICE HANDLER
ICDNO    *=  *+1     ;DEVICE NUMBER
ICCOM    *=  *+1     ;I/O COMMAND
ICSTA    *=  *+1     ;I/O STATUS
ICBADR   *=  *+2     ;BUFFER ADDRESS
ICPUT    *=  *+2     ;DH PUT ROUTINE
```

```
ICBLEN   *=  *+2      ;BUFFER LENGTH
ICAUX1   *=  *+1      ;AUXILIARY BYTE
ICAUX2   *=  *+1      ;AUX 2
         *=  $0300    ;SIO ADDRESSES
DDEVIC   *=  *+1      ;DEVICE
DUNIT    *=  *+1      ;UNIT NO.
DCOMND   *=  *+1      ;COMMAND
DSTATS   *=  *+1      ;STATUS
DBUFLO   *=  *+2      ;BUFFER ADDRESS
DTIMLO   *=  *+1      ;TIMEOUT VALUE
DUNUSE   *=  *+1      ;NOT USED
DBYTLO   *=  *+2      ;NUMBER OF BYTES
DAUX1    *=  *+2      ;AUXILIARY BYTES
CIO      =   $E456    ;OS I/O ROUTINE
SIO      =   $E459    ;SERIAL I/O
DR       =   $52      ;READ A SECTOR
DW       =   $57      ;WRITE A SECTOR
OPN      =   $03      ;OPEN COMMAND
OUTPUT   =   $08      ;OPEN DIRECTION
PUT      =   $0B      ;PUT CHARACTERS
CONSOL   =   $D01F    ;START KEY
DOSVEC   =   $0A      ;DOS VECTOR
SECTORS  =   $B0      ;SECTOR COUNTER
INDEX    =   $B2      ;WORK INDEX
SN       =   $B3      ;SECTOR NUMBER
DSEC     =   $B5      ;DATA SECTOR
TEMP     =   $B7      ;WORK AREA
FLAG     =   $BA      ;WORK FLAG
DOCNO    =   $BD      ;FILE NUMBER
ROWCRS   =   $54      ;CURSOR ROW
COLCRS   =   $55      ;CURSOR COLUMN
;
         *=  $4500
START    .BYTE "DISK CONVERSION"
START1   .BYTE "UTILITY"
START2   .BYTE "PRESS START"
MSG      .BYTE "conversion in  "
MSG1     .BYTE "progress"
RETRY    *=  *+1
SCREEN   .BYTE "S:"
MASK     .BYTE $40
DIR      ;            DIRECTORY BUFFER
         *=  *+256
DATA     ;            DATA BUFFER
         *=  *+256
         .MACRO DSKIO
         JSR SETUP    ;SET UP REGISTERS
         LDA %1
         STA DAUX1    ;SET SECTOR ADDR
         LDA %2
         STA DAUX1+1
         LDA # <%3
         STA DBUFLO   ;BUFFER ADDRESS
         LDA # >%3
         STA DBUFLO+1
         LDA #%4
           .IF %4=DR
           LDY #$40
           .ELSE
           LDY #$80
           .ENDIF
         STA DCOMND   ;STORE COMMAND
         STY DSTATS   ;STORE DIRECTION
         JSR DOIO
         .ENDM
BEGIN
         LDX #$10     ;CHANNEL 1
         LDA # <SCREEN ;SET POINTER TO
         STA ICBADR,X ;DEVICE NAME
         LDA # >SCREEN
         STA ICBADR+1,X
         LDA #OPN     ;OPEN COMMAND
         STA ICCOM,X
         LDA #OUTPUT  ;OPEN DIRECTION
         STA ICAUX1,X
         LDA #2       ;GRAPHICS 2
         STA ICAUX2,X
         JSR CIO      ;DO I/O
         LDA #2       ;POSITION CURSOR
         STA ROWCRS
         LDA #3
         STA COLCRS
         LDA #0
         STA COLCRS+1
         LDA # <START ;POINT TO 1ST
         STA ICBADR,X ;MESSAGE LINE
         LDA # >START
         STA ICBADR+1,X
         LDA # <15    ;MESSAGE LENGTH
         STA ICBLEN,X
         LDA #0
         STA ICBLEN+1,X
         JSR PRINT    ;PRINT IT
         LDA # <7     ;MESSAGE LENGTH
         STA ICBLEN,X
         LDA # <START1 ;POINT TO 2ND
         STA ICBADR,X ;LINE OF MESSAGE
         LDA # >START1
         STA ICBADR+1,X
         LDA #3       ;POSITION CURSOR
         STA ROWCRS
         LDA #6
         STA COLCRS
         JSR PRINT    ;PRINT IT
         LDA # <11    ;MESSAGE LENGTH
         STA ICBLEN,X
         LDA # <START2 ;POINT TO LINE
         STA ICBADR,X ;3 OF MESSAGE
         LDA # >START2
         STA ICBADR+1,X
         LDA #5       ;POSITION CURSOR
         STA ROWCRS
         LDA #4
         STA COLCRS
         JSR PRINT    ;PRINT IT
WAIT
         LDA CONSOL   ;START PRESSED?
         CMP #6
         BNE WAIT     ;NO, HANG AROUND
         LDA #2       ;POSITION CURSOR
         STA ROWCRS
         LDA #3
         STA COLCRS
         LDA # <MSG   ;POINT TO MESSAGE
         STA ICBADR,X
         LDA # >MSG
         STA ICBADR+1,X
         LDA # <15    ;SET LENGTH
         STA ICBLEN,X
         JSR PRINT    ;PRINT IT
         LDA #3       ;POSITION CURSOR
         STA ROWCRS
         LDA #5
         STA COLCRS
         LDA # <MSG1 ;POINT TO NEXT
         STA ICBADR,X ;MESSAGE LINE
         LDA # >MSG1
         STA ICBADR+1,X
         LDA # <8     ;SET LENGTH
         STA ICBLEN,X
         JSR PRINT    ;PRINT IT
         LDA # <361   ;POINT TO 1ST
         STA SN       ;DIRECTORY SECTOR
         LDA # >361
         STA SN+1
         LDA #0       ;INIT WORK AREAS
         STA SECTORS
         STA INDEX
         STA DOCNO
READSECT
;READ DIRECTORY SECTOR
          DSKIO  SN,SN+1,DIR,DR
         LDA #0       ;ZERO THE FLAG
         STA FLAG
EXAMINE
         LDY INDEX    ;PNT TO STAT BYTE
```

```
        LDA DIR,Y
        BIT MASK     ;FILE IN USE?
        BNE DOFIX    ;YES,GO READ IT
        CMP #0       ;STATUS ZERO?
        BNE DONEXT   ;NO, SKIP IT
        LDA #$80     ;DELETED FLAG
        STA DIR,Y
        BNE DONEXT
DOFIX
        STA FLAG     ;MAKE FLAG NON-0
        JSR FIXIT    ;GO READ FILE
DONEXT
        INC DOCNO    ;INC FILE NO.
        CLC
        LDA INDEX    ;POINT TO NEXT
        ADC #$10     ;STATUS BYTE
        BCS RESET    ;DONE?
        STA INDEX    ;NOPE
        BCC EXAMINE
RESET
        LDA FLAG     ;ANYTHING IN DIR?
        BEQ FINISHED ;NO, DONE!
        LDA #0       ;ZERO THE FLAG
        STA FLAG     ;AND THE INDEX
        STA INDEX
;REWRITE THE SECTOR
         DSKIO  SN,SN+1,DIR,DW
RS1
        CLC
        INC SN       ;POINT TO NEXT
        BNE RS2      ;DIRECTORY SECTOR
        INC SN+1
RS2
        INC SECTORS ;INC SECTOR COUNT
        LDA SECTORS
        CMP #8       ;8 SECTORS READ?
        BCC RDSECT   ;NO,KEEP GOING
        JMP FINISHED
RDSECT
        JMP READSECT
FINISHED
        JMP (DOSVEC) ;GO TO DOS
SETUP
        LDA #3
        STA RETRY    ;SET RETRY COUNT
        LDA #1
        STA DUNIT    ;DRIVE 1
        LDA #$31
        LDY #15
        STA DDEVIC   ;SET DEVICE NO.
        STY DTIMLO   ;SET TIMEOUT VALUE
        LDA #1
        LDY #0
        STA DBYTLO+1 ;READ A 256
        STY DBYTLO   ;BYTE SECTOR
        RTS
DOIO
        JSR SIO      ;CALL SIO
        BPL XIT      ;WAS IT GOOD?
        DEC RETRY    ;NO,DEC RETRY
        BMI XIT      ;NO MORE RETRIES
        LDA DCOMND   ;GET COMMAND
        CMP #DR      ;WAS IT A READ
        BNE B1       ;NO
        LDX #$40     ;READ DIRECTION
        BNE B2
B1
        LDX #$80     ;WRITE DIRECTION
B2
        STX DSTATS   ;RESET STATUS
        JMP DOIO
XIT
        RTS
PRINT
        LDA #PUT     ;PUT BYTES
        STA ICCOM,X ;STORE IN COMMAND
        JSR CIO      ;DO I/O
        RTS
FIXIT
        LDX #$10     ;CHANNEL 1
        TYA
        PHA          ;SAVE Y
        CLC          ;PNT TO FILENAME
        ADC # <DIR+5
        STA ICBADR,X
        LDA # >DIR
        ADC #0
        STA ICBADR+1,X
        LDA # <11    ;SET NAME LENGTH
        STA ICBLEN,X
        LDA #5       ;POSITION CURSOR
        STA ROWCRS
        LDA #4
        STA COLCRS
        JSR PRINT    ;PRINT FILE NAME
        PLA
        TAY          ;RESTORE Y
        INY          ;ADD 3 TO Y
        INY
        INY
        LDA DIR,Y    ;GET BEGINNING
        STA DSEC     ;SECTOR
        INY
        LDA DIR,Y
        STA DSEC+1
READDATA
;READ A DATA SECTOR
        DSKIO  DSEC,DSEC+1,DATA,DR
        LDA DOCNO    ;GET FILE NO.
        ASL A        ;LEFT JUSTIFY IT
        ASL A
        STA TEMP     ;SAVE IT
        LDA DATA+253 ;GET OLD FILE NO.
        AND #$03     ;MASK OFF SECTOR
        PHA          ;HIGH BITS&SAVE IT
        ORA TEMP     ;COMBINE WITH FILE NO
        STA DATA+253 ;STORE IT
;WRITE DATA SECTOR
         DSKIO  DSEC,DSEC+1,DATA,DW
        LDA #0       ;ZERO TEMP
        STA TEMP
        STA TEMP+1
        PLA          ;GET SECTOR HIGH
        STA DSEC+1   ;STORE IT
        BEQ N3       ;WAS IT ZERO?
        INC TEMP     ;NO
N3
        LDA DATA+254 ;GET SECTOR LO
        STA DSEC     ;NEXT DATA SECTOR
        BEQ N4       ;WAS IT ZERO?
        INC TEMP
N4
        LDA TEMP     ;DATA SEC VALID?
        BEQ EXIT     ;NO,END OF FILE
        JMP READDATA
EXIT
        RTS
        *=  $02E0
        .WORD BEGIN
        .END
```

UTILITY

48K Disk

Busy Buddy Express

Put your BBS on hold when you need a break

by Matthew J.W. Ratcliff

When it comes to telecomputing, I spend a lot of time on the modem. It can be more than a little frustrating when I'm interrupted while on-line and get *timed out*. Most bulletin boards have an automatic timing feature; if you don't enter something within a minute or so, you're logged off. This prevents the BBS from getting tied up all night, in the event someone forgets to log off. Off course, it also tends to prevent users from taking a brief intermission while staying on line. What's really needed is a way to put the BBS "on hold."

I've caught myself typing SPACE and BACKSPACE many times, to prevent that timeout while reading a long message. I suppose I could buy a $300 programmable toy robot and train it to smack these two keys intermittently while I raid the fridge. Instead, I've created a more cost-effective "BBS space, backspace, intermittent typing hold on line unit" called **Busy Buddy Express**.

To create your copy of **Busy Buddy Express**, type Listing 1 using the "M/L Editor." **Busy Buddy** is a binary load file that hooks into the Atari's VBI (Vertical Blank Interrupt) vector. Once loaded, a title screen is displayed, along with a brief reminder of how the program functions, and control is then returned to DOS. **Busy Buddy** is a tiny program—less than 128 bytes of memory, it resides in the cassette buffer ($400-47F)—and is safe from most terminal program's special routines that generally reside in page 6 ($600-$6FF).

After performing a binary load from DOS, you can then load Express 3.0, by Keith Ledbetter. There are lots of other terminal programs available for your Atari, but **Busy Buddy** may not work with all of them. I talked to Keith about version 3.0, which is now available in both 850 and 1030 versions, and he assured me that he "steals" the VBI vector "legally." Many terminal programs employ VBI routines for clocks and other things, which simply hook in their own vectors and exit to the operating system when complete. Express 3.0 checks the current vector (which will be that for **Busy Buddy**), saves it as the exit vector, then hooks in its own.

Try **Busy Buddy** with other terminal programs. At the very worst, it won't work. If it doesn't, get Express 3.0. It's the best, in my opinion, and can be downloaded for free (it's a "share-ware" program) from many local BBSs and is in the telecommunications database on Delphi. The latest and greatest version of Express is always available for downloading from ICD's own BBS at 815-968-2229.

If you're on-line and need a break, simply press CTRL-SHIFT-INSERT (press the CONTROL and SHIFT buttons, then hit the insert, or greater than key). This will engage

Busy Buddy, and remind you that it's on by displaying an inverse letter *B* (for Busy) near the top left of your screen. Once every 3.5 seconds (approximately) a SPACE or BACK-SPACE character is alternately poked into the keyboard input register. The terminal program takes care of the rest, thinking you had typed those keys yourself. In case you should forget that **Busy Buddy** is keeping you connected, it will automatically cancel itself after 15 minutes. (This can be a real lifesaver if you forget you're on hold while connected long distance.) Note that **Busy Buddy** fools the terminal program, but not the operating system. Your computer's Attract Mode is not disabled by **Bud**.

To disengage **Busy Buddy**, simply press CTRL-SHIFT-CLEAR. An inverse letter *C* (for Canceled) will be displayed near the top left of the screen.

You may wish to automatically install **Busy Buddy** every time you run Express, but I wasn't successful in appending one file to the other, so both would automatically load and run. If you use SpartaDOS or DOS XL, you can create a batch file (such as STARTUP.BAT for Sparta, which is equivalent to an AUTORUN.SYS but can run multiple command files) to get the job done: BUSYBUD EXPRESS. You don't need to include the RS232 command, since Express automatically boots the 850 (or 1030) handler itself.

I tested **Busy Buddy** with Amodem Plus version 6.2, but it didn't work. I didn't test it with any other terminal software, because those are the only 8-bit terminal programs I've ever used regularly. If **Busy Buddy** doesn't work with your terminal software, I highly recommend Express 3.0. The price is right and you'll be hard put to find a more full-featured terminal program for the 8-bit Atari.

The next time you find you're in a marathon FOREM BBS message entry session and nature calls, simply call upon **Busy Buddy Express** to keep you on-line. It's a sure cure for those BBS timeout blues!

Listing 1
M/L Editor data

```
1000 DATA 255,255,0,64,251,64,173,36,2
,141,0,4,173,37,2,141,1688
1010 DATA 1,4,169,6,160,4,166,20,228,2
0,240,252,141,36,2,140,6219
1020 DATA 37,2,169,33,141,3,4,169,211,
141,2,4,169,0,141,4,1558
1030 DATA 4,170,169,11,141,66,3,173,22
,65,141,72,3,142,73,3,864
1040 DATA 169,72,141,68,3,169,64,141,6
9,3,32,86,228,96,125,127,4903
1050 DATA 160,194,245,243,249,160,194,
245,228,228,249,160,197,248,240,242,13
24
1060 DATA 229,243,243,160,155,127,66,8
9,32,77,97,116,42,82,97,116,4377
1070 DATA 32,45,32,102,114,111,109,32,
65,78,65,76,79,71,155,155,3769
1080 DATA 80,114,101,115,115,32,67,84,
82,76,45,83,72,73,70,84,1400
1090 DATA 45,73,78,83,69,82,84,32,116,
111,32,101,110,97,98,108,3232
1100 DATA 101,155,32,32,32,32,32,32,67
,84,82,76,45,83,72,73,9819
1110 DATA 70,84,45,67,76,69,65,82,32,3
2,116,111,32,100,105,115,2103
1120 DATA 97,98,108,101,155,87,104,101
,110,32,97,99,116,105,118,101,4903
1130 DATA 44,32,66,117,115,121,32,66,1
17,100,32,119,105,108,108,32,2799
1140 DATA 97,117,116,111,45,155,116,10
5,109,101,111,117,116,32,97,102,4729
1150 DATA 116,101,114,32,49,53,32,109,
105,110,117,116,101,115,155,116,5425
1160 DATA 111,32,252,64,22,65,112,114,
101,118,101,110,116,32,77,65,3214
1170 DATA 45,66,69,76,76,32,160,211,20
0,207,195,203,160,33,155,155,1036
1180 DATA 206,0,4,109,4,0,0,0,0,0,0,8,
72,173,4,4,5432
1190 DATA 208,23,173,252,2,201,247,208
,11,238,4,4,169,0,141,5,4543
1200 DATA 4,32,95,4,104,40,108,0,4,173
,252,2,201,246,208,11,7000
1210 DATA 169,0,141,4,4,32,95,4,76,31,
4,206,2,4,208,228,3087
1220 DATA 206,5,4,240,235,169,211,141,
2,4,173,3,4,141,252,2,5037
1230 DATA 201,52,240,8,169,52,141,3,4,
76,31,4,169,33,141,3,462
1240 DATA 4,76,31,4,152,72,160,40,169,
163,56,237,4,4,145,88,4439
1250 DATA 104,168,96,226,2,227,2,0,64,
0,0,0,0,0,0,0,4844
```

Listing 2
Assembly listing

```
1000 *SAVE#D:BUSYBUD.M65
1010 *ASM,#-,#D:BUSYBUD.COM
1020      .OPT OBJ
1030 * Busy Buddy XE by Mat*Rat
1040 * Ctrl-Shft-> BUSY ON
1050 * Ctrl-Shft-< BUSY OFF
1060 *
1070 FOURSEC = 211    ;ACTUALLY 3.5 SEC
1080 ; 3.5 SECONDS * 256 (MAXTIME)
1090 ; EQUALS ABOUT 15 MINUTES
1100 JIFFY = $14
1110 BUSYON = 247
1120 BUSYOFF = 246
1130 VVBLKD = $0224
1140 CH   =   $02FC
1150 SPACE = 33       ;INTERNAL CODE
1160 BS   =   52
1170 SAVMSC = $58     ; screen ptr
1180 * Operating system equates:
1190 CIO =    $E456
1200 ICCOM = $0342
1210 ICBAL = $0344
1220 ICBAH = $0345
1230 ICBLL = $0348
1240 ICBLH = $0349
1250 ICAX1 = $034A
1260 ICAX2 = $034B
1270 PUTBIN = $0B
1280 *
1290      .ORG $4000
1300 INIT LDA VVBLKD ; install
1310      STA VEXIT   ; Busy Buddy
1320      LDA VVBLKD+1 ; Express
1330      STA VEXIT+1 ; VBI routine
1340      LDA # <START
1350      LDY # >START
1360      LDX JIFFY
1370 HOLD CPX JIFFY   ; Sync up so
1380      BEQ HOLD    ; a VBI won't
1390      STA VVBLKD  ; crash the
1400      STY VVBLKD+1 ; installation
1410      LDA #SPACE  ; procedure
1420      STA BACKUP  ; Iniz variables
1430      LDA #FOURSEC ; dela timer
1440      STA TIMER   ; of 3.5 seconds
1450      LDA #0      ; Busy control
1460      STA BUSYCTL ; OFF for now
1470      TAX
1480      LDA #PUTBIN ; print title
1490      STA ICCOM   ; screen
1500      LDA LEN
1510      STA ICBLL
1520      STX ICBLH
1530      LDA # <TITLE
1540      STA ICBAL
1550      LDA # >TITLE
1560      STA ICBAH
1570      JSR CIO
1580      RTS           ; Back to DOS
1590 TITLE .BYTE "R Busy Buddy Expre"
1595      .BYTE "ss ",155
1600      .BYTE "BY Mat*Rat - from AN"
```

```
1605      .BYTE "ALOG",155,155
1610      .BYTE "Press CTRL-SHIFT-INSE"
1615      .BYTE "RT to enable",155
1620      .BYTE "      CTRL-SHIFT-CLEA"
1625      .BYTE "R  to disable",155
1630      .BYTE "When active, Busy Bud"
1635      .BYTE " will auto-",155
1640      .BYTE "timeout after 15 minu"
1645      .BYTE "tes",155
1650      .BYTE "to prevent MA-BELL "S"
1655      .BYTE "HOCK!",155,155
1660 LEN  .BYTE *-TITLE
1670 *
1680 * Busy Buddy VBI code
1690 * installed.
1700 * Actual program appears
1710 * below.
1720 *
1730      .ORG $0400
1740 VEXIT .WORD 0    ; VBI exit vector
1750 TIMER .BYTE 0    ; 3.5 Sec timer
1760 BACKUP .BYTE 0   ; Space or B5
1770 BUSYCTL .BYTE 0  ; control flg
1780 MAXTIM .BYTE 0   ; 15 min timer
1790 START PHP        ; VBI starts here
1800      PHA         ; save all stats
1810      LDA BUSYCTL ; and regs
1820      BNE TIMIT   ;ON
1830 TESTON LDA CH    ;Busy on request?
1840      CMP #BUSYON
1850      BNE EXIT    ; No, no change
1860      INC BUSYCTL ;OFF, TURN IT ON
1870      LDA #0
1880      STA MAXTIM  ; 256*3.5 = 15Min
1890      JSR BPR     ; Show ON char B
1900 EXIT PLA         ; Restore stats
1910      PLP         ; and reg & exit
1920      JMP (VEXIT)
1930 TIMIT LDA CH     ; Busy on
1940      CMP #BUSYOFF ; turn it off?
1950      BNE TIM1    ; No, time it out
1960 CANCEL LDA #0    ; yes, toggle ctl
1970      STA BUSYCTL
1980      JSR BPR     ; show Clear stat
1990      JMP EXIT
2000 TIM1 DEC TIMER   ; 15 minute time
2010      BNE EXIT    ; limit?
2020      DEC MAXTIM
2030      BEQ CANCEL  ; yes, cancel
2040      LDA #FOURSEC ; NO, reset
2050      STA TIMER   ; the timer
2060      LDA BACKUP  ; Send the SPACE
2070      STA CH      ; or B5 character
2080      CMP #B5     ; and set BACKUP
2090      BEQ PUTSP   ; variable for
2100      LDA #B5     ; next time.
2110      STA BACKUP
2120      JMP EXIT
2130 PUTSP LDA #SPACE
2140      STA BACKUP
2150      JMP EXIT
2160 BPR  TYA         ; Show a C or
2170      PHA         ; B status char
2180      LDY #40     ; near top left
2190      LDA #67-32+128
2200      SEC         ; of display
2210      SBC BUSYCTL ; by poking
2220      STA (SAVMSC),Y
2230      PLA         ; to screen RAM
2240      TAY
2250      RTS
2260      .ORG $02E0  ; iniz addr
2270      .WORD INIT  ; for LOAD
2280      .END
```


Get something Extra!

$8.95
PLUS $1.50 FOR POSTAGE AND HANDLING

An Atari 8-bit Extra
from ANALOG Computing

It's a book of some of the best articles and software listings submitted to **ANALOG Computing**—things we just couldn't fit in the monthly magazine pages.

Owners of Atari 8-bit computers will find the **Extra** a must. It gives you games, tutorials, utilities, applications, and more—material you'll want to keep.

When you're looking for the best in Atari—tutorials, games, reviews and programs—look for **ANALOG Computing**.

We're the magazine that always gives you something **Extra**.

Don't miss it!

Binary Load Pictures

A utility to convert pictures to binary load files

by Charles F. Johnson

When you want to share a picture file with a friend you always have to make sure that he or she has the correct program to load and display the picture. If you operate a BBS, you've probably had to deal with lots of questions about how to load a certain picture file. Finally, there's a program to resolve these problems. **Binary Load Pictures** (BLP, for short) will take a Micro Painter or Micro Illustrator picture file and convert it into a binary file that can be loaded with any DOS. (Anyone owning a disk drive has some kind of DOS.)

This program will handle either Micro Illustrator or Micro Painter files. It asks which type of picture you're converting, and then asks for the picture's filename and a new filename for the converted binary file. Type in the names with the D: drive specifier. (Remember to give different names so your original file remains intact.) I suggest using a filename extension of .OBJ for the converted file (this is an informal standard extension for a DOS binary load file). After you've entered the two filenames, sit back and let **BLP** go to work. In a short time, the binary file will be created using the filename you specified.

When you load this file from the DOS menu, the picture will be displayed. Then a short machine language routine waits for you to press a key. When you do, you'll be returned to DOS. That's all there is to it!

BLP can be used by machine language programmers to add a custom title screen to an already existing binary load program. Just use the DOS copy with append function, with your object filename as the source and the converted picture filename as destination. Next time you load this compound file, the title screen you've created with Micro Illustrator or Micro Painter will appear. After a keypress, the rest of the program file should load and run. The binary load picture file itself can also be named AUTORUN.SYS, for a title screen every time you boot up.

Listing 1 is MAKBIN.BAS, the BASIC program that converts the pictures. Listing 2 is the MAC/65 assembly language source code for the short program that handles displaying the picture. You don't have to type in Listing 2 to use MAKBIN.BAS—it's included for readers interested in 6502 assembly language. Take care when typing in the BASIC listing, and always save a copy of it before trying to run it. Have fun!

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

Listing 1
BASIC listing

```
XO 10 GRAPHICS 0:? :? "Just a sec..."
KT 20 DIM K$(1),TEMP$(15),IN$(15),OUT$(15
   ),BUF$(7680),HEADER$(7),H2$(7),COL(4)
YI 30 DIM POKE$(25),GR7PLUS$(77),INIT$(36
   )
AT 40 HEADER$="[XIG]♥├"
FU 50 RESTORE 50:FOR I=0 TO 4:READ BYTE:C
   OL(I)=BYTE:NEXT I:DATA 40,202,148,12,0
MH 60 IO=ADR("hhh[K]LV[d]"):BUF$="♥":BUF$(768
   0)=BUF$:BUF$(2)=BUF$:BUF=ADR(BUF$)
RI 70 RESTORE 530:FOR I=1 TO 25:READ BYTE
   :LET POKE$(I)=CHR$(BYTE):NEXT I
WJ 80 LET POKE=ADR(POKE$)
PC 90 RESTORE 550:FOR I=1 TO 77:READ BYTE
   :GR7PLUS$(I)=CHR$(BYTE):NEXT I
DR 100 GR7PLUS=ADR(GR7PLUS$)
IS 110 RESTORE 600:FOR I=1 TO 36:READ BYT
   E:INIT$(I)=CHR$(BYTE):NEXT I
XN 120 INIT=ADR(INIT$)
ME 130 RESTORE 630:FOR I=0 TO 237:READ BY
   TE:POKE 1536+I,BYTE:NEXT I
ZP 140 CLOSE #3:OPEN #3,4,0,"K:"
AW 150 GRAPHICS 0:? :? "  CONVERT PICTURE
   TO BINARY FILE ":? :? "by Charles F. J
   ohnson"
VI 160 ? :? "Which type of file do you wa
   nt":? "to convert?":? :? "[1]-Koala  [2]-M
   icroPainter  >";
BK 170 GET #3,K:K$=CHR$(K):IF K$<>"1" AND
    K$<>"2" THEN 170
PX 180 ? K$:KOALA=(K$="1")
YF 190 ? :? "Input Filename >";:INPUT #16
   ;IN$
XW 191 IF LEN(IN$)<3 THEN 195
WM 192 IF IN$(2,2)=":" OR IN$(3,3)=":" TH
   EN 200
ZZ 195 TEMP$=IN$:IN$="D1:":IN$(4)=TEMP$
WN 200 ? :? "Output Filename >";:INPUT #1
   6;OUT$
QD 201 IF LEN(OUT$)<3 THEN 205
FX 202 IF OUT$(2,2)=":" OR OUT$(3,3)=":"
   THEN 210
BC 205 TEMP$=OUT$:OUT$="D1:":OUT$(4)=TEMP
   $
YY 210 IF IN$=OUT$ THEN ? :? "INPUT AND O
   UTPUT NAMES MUST DIFFER!":GOTO 190
QO 220 OPEN #1,4,0,IN$:IF KOALA THEN GOSU
   B 360:GOTO 240
VG 230 GOSUB 470
IY 240 CLOSE #1:OPEN #2,8,0,OUT$
YC 250 RESTORE 740:FOR I=1 TO 284:READ BY
   TE:PUT #2,BYTE:NEXT I
ZO 260 Q=USR(POKE,866,11,868,SCREEN,872,7
   680):Q=USR(IO,32)
JC 270 PUT #2,240:PUT #2,6
LI 280 PUT #2,244:PUT #2,6
SU 290 FOR I=0 TO 4:PUT #2,COL(I):NEXT I
GF 300 PUT #2,224:PUT #2,2
GV 310 PUT #2,225:PUT #2,2
PY 320 PUT #2,0:PUT #2,96
XM 330 CLOSE #1:CLOSE #2
JC 340 GRAPHICS 0:? :? "Conversion comple
   te.":? :END
NL 350 REM  LOAD KOALA PICTURE
SB 360 FOR I=1 TO 7:GET #1,BYTE:H2$(I)=CH
   R$(BYTE):NEXT I
ZG 370 IF H2$<>HEADER$ THEN ? :? "ERROR!
   - ";IN$;" is not":? "a Koala picture f
   ile.":? :POP :END
XQ 380 GOSUB 510:GET #1,TYPE:FOR I=1 TO 5
   :GET #1,BYTE:NEXT I
YM 390 FOR I=0 TO 4:GET #1,BYTE:POKE 708+
```


Gear up Your Disk Drive For Big Savings!
Save $14 Off The Cover Price.

SUBSCRIBE TO

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS

ANALOG COMPUTING

☐ 1 Year.....$28.....Save $14
MCEYY

☐ 1 Year with Disk.....$105
DCEYY

Name

Address

City State Zip

Make checks payable to: L.F.P. Inc. Allow 4-6 weeks for delivery.

☐ Payment Enclosed ☐ Bill Me
☐ Charge My ☐ Visa ☐ MC
_____ Exp ____

Signature

FOREIGN—Add $7 per year
MONEY BACK if not delighted
Analog
P.O. Box 16927
N. Hollywood, CA 91615

```
   I,BYTE:COL(I)=BYTE:NEXT I
PQ 400 FOR I=1 TO 9:GET #1,BYTE:NEXT I
WE 410 POKE 850,7:IF TYPE THEN Q=USR(POKE
   ,852,BUF):GOTO 430
OM 420 Q=USR(POKE,852,SCREEN)
ON 430 Q=USR(POKE,856,7680):Q=USR(IO,16):
   CLOSE #1
LX 440 IF TYPE THEN Q=USR(INIT,BUF,TYPE)
ZK 450 RETURN
UU 460 REM LOAD MICROPAINTER PICTURE
CX 470 GOSUB 510:Q=USR(POKE,850,7,852,SCR
   EEN,856,7680):Q=USR(IO,16)
MF 480 GET #1,BYTE:COL(4)=BYTE:POKE 712,B
   YTE:FOR I=0 TO 2:GET #1,BYTE:COL(I)=BY
   TE:POKE 708+I,BYTE:NEXT I
ZS 490 RETURN
HN 500 REM SET UP GRAPHICS 0E SCREEN
RH 510 GRAPHICS 24:Q=USR(GR7PLUS):SCREEN=
   PEEK(88)+256*PEEK(89):RETURN
ZA 520 REM DATA FOR POKE ROUTINE
YZ 530 DATA 104,74,170,160,0,104,133,255,
   104,133,254,104,240,4,200,145,254,136,
   104,145,254,202,208,237,96
RU 540 REM DATA FOR GR7PLUS ROUTINE
ES 550 DATA 104,173,48,2,24,105,3,133,203
   ,173,49,2,105,0,133,204,160,0,177,203,
   201,79,208,21
SG 560 DATA 169,78,145,203,165,203,24,105
   ,2,133,203,165,204,105,0,133,204,169,0
   ,240,15,201,15,208
ER 570 DATA 6,169,14,145,203,208,5,201,65
   ,208,1,96,165,203,24,105,1,133,203,165
   ,204,105,0,133
JY 580 DATA 204,169,0,240,197
ET 590 REM DATA FOR INIT ROUTINE
YG 600 DATA 104,104,133,204,104,133,203,1
   04,168,104,133,214,165,88,133,205,133,
   215,133,217,165,89,133,206
KU 610 DATA 133,216,24,105,30,133,218,132
   ,208,76,0,6
UO 620 REM DATA FOR PAGE 6
NT 630 DATA 132,213,132,207,177,203,8,32,
   110,6,40,24,42,38,207,74,133,212,208,1
   4,177,203,133,213
FK 640 DATA 32,110,6,177,203,133,212,32,1
   10,6,165,207,240,34,177,203,32,110,6,1
   45,205,166,214,224
GS 650 DATA 2,208,6,32,215,6,76,60,6,32,1
   17,6,198,212,208,230,165,213,240,188,1
   98,213,16,222
CL 660 DATA 177,203,133,209,32,110,6,165,
   209,145,205,166,214,224,2,208,6,32,215
   ,6,76,98,6,32
DK 670 DATA 117,6,198,212,208,233,165,213
   ,240,150,198,213,16,225,230,203,208,2,
   230,204,96,24,165,205
KN 680 DATA 105,80,133,205,144,2,230,206,
   165,205,197,217,208,42,165,206,197,218
   ,208,36,165,208,208,33
MY 690 DATA 230,208,24,165,215,105,40,133
   ,205,165,216,105,0,133,206,24,165,217,
   105,40,133,217,144,2
NB 700 DATA 230,218,230,215,208,2,230,216
   ,96,198,208,165,215,133,205,133,217,16
   5,216,133,206,24,105,30
BF 710 DATA 133,218,24,165,88,105,40,170,
   165,89,105,0,228,205,208,224,197,206,2
   08,220,104,104,96,230
QQ 720 DATA 205,208,2,230,206,165,88,197,
   205,208,205,165,89,24,105,30,197,206,2
   08,196,240,230
ZR 730 REM DATA FOR 1ST PART OF FILE
EN 740 DATA 255,255,0,96,67,96,162,4,189,
   196,2,157,245,6,189,240,6,157,196,2,20
   2,16,241,173
LO 750 DATA 48,2,72,173,49,2,72,169,80,14
   1,48,2,169,96,141,49,2,169,255,141,252
   ,2,205,252
DT 760 DATA 2,240,251,141,252,2,162,4,189
   ,245,6,157,196,2,202,16,247,104,141,49
   ,2,104,141,48
MF 770 DATA 2,96,80,96,25,97,112,112,112,
   78,80,97,14,14,14,14,14,14,14,14,14,14
   ,14,14
OZ 780 DATA 14,14,14,14,14,14,14,14,14,14
   ,14,14,14,14,14,14,14,14,14,14,14,14,1
   4,14
PB 790 DATA 14,14,14,14,14,14,14,14,14,14
   ,14,14,14,14,14,14,14,14,14,14,14,14,1
   4,14
OK 800 DATA 14,14,14,14,14,14,14,14,14,14
   ,14,14,14,14,14,14,14,14,14,14,14,14,1
   4,14
TK 810 DATA 14,14,14,14,14,14,14,14,14,78
   ,0,112,14,14,14,14,14,14,14,14,14,14,1
   4,14
OO 820 DATA 14,14,14,14,14,14,14,14,14,14
   ,14,14,14,14,14,14,14,14,14,14,14,14,1
   4,14
OQ 830 DATA 14,14,14,14,14,14,14,14,14,14
   ,14,14,14,14,14,14,14,14,14,14,14,14,1
   4,14
OS 840 DATA 14,14,14,14,14,14,14,14,14,14
   ,14,14,14,14,14,14,14,14,14,14,14,14,1
   4,14
UN 850 DATA 14,14,14,14,14,14,14,14,14,14
   ,14,14,14,65,80,96,80,97,79,127
```

Listing 2
Assembly listing

```
1000      .OPT NO EJECT
1010 ;
1020 ; ROUTINE FOR MAKBIN.BAS
1030 ;
1040 ; By Charles F. Johnson
1050 ;
1060 ; Equates
1070 ;
1080 SDLSTL = $0230
1090 SDLSTH = $0231
1100 COLOR0 = $02C4
1110 CH   =   $02FC
1120 ;
1130 DLADDR = $6050
1140 ;
1150      *=  $06F0
1160 PICCOL .DS 5
1170 COLSAV .DS 5
1180 ;
1190      *=  $6000
1200      LDX #4
1210 COLORS
1220      LDA COLOR0,X ;Save current
1230      STA COLSAV,X ;colors
1240      LDA PICCOL,X ;Copy colors
1250      STA COLOR0,X ;from page six
1260      DEX
1270      BPL COLORS
1280 ;
1290      LDA SDLSTL  ;Save address
1300      PHA         ;of display list
1310      LDA SDLSTH
1320      PHA
1330 ;
1340      LDA # <DLADDR ;Install our
1350      STA SDLSTL  ;display list
1360      LDA # >DLADDR
1370      STA SDLSTH
1380 ;
1390      LDA #$FF    ;Clear key buffer
1400      STA CH
1410 ;
1420 WAIT
1430      CMP CH      ;Wait for
1440      BEQ WAIT    ;keypress
1450 ;
1460      STA CH      ;Cancel keypress
1470      LDX #4
1480 RESCOL
1490      LDA COLSAV,X ;Restore colors
1500      STA COLOR0,X
1510      DEX
1520      BPL RESCOL
1530 ;
1540      PLA         ;Restore
1550      STA SDLSTH  ;display list
1560      PLA
1570      STA SDLSTL
1580 ;
1590      RTS         ;Return
1600 ;
1610      .OPT NO LIST
```

GAME

48K Cassette or Disk

Cloudhopper

Fly the friendly skies — sort of.

by Greg Knauss

When the volcano erupted in your sleepy little town, it caused quite a stir—as any volcano in one of those square midwestern states would.

For weeks, the Knocks Volcano (as it came to be known) spat forth soot and molten rock. Fortunately, for some odd reason that we'll not get into here, the clouds that overhang the volcano absorbed the volcanic gook, thus saving the lower half of your town.

Several banks of clouds still pass peacefully over the now calmed volcano, but at their heart presently lies hardened rock. (Hey, you want reality, read *Newsweek*.)

Feeling adventuresome one day, you grabbed your faithful pogostick and invented the exhilirating art of *cloudhopping*. You went out one day when the fog was lying low and just climbed up. A few tentative hops and you were off...

Cloudhopping took the world by storm. *Sports Illustrated* ran a cover story. *Readers Digest* did a lengthy feature. The *National Enquirer* explained how you are actually an alien. Unfortunately, anyone else who tried this new thrill seemed to, er, die. The air currents that control the hardened clouds are very unpredicatable and the volcano is eager to swallow those who misnavigate them.

You've even invented a special suit to protect yourself if you should make such a silly mistake. The full-body jumpsuit will protect against two such slips, and each time it's exposed to the super-heat of the volcano it changes color to remind you how many dives you've taken: purple for none, blue for one, and green for two. (For those of you with a black-and-white reality, the suit gets lighter the more worn through it is.)

Then one day, while touching up your tan, you notice large red balloons hovering over the volcano. Tied to each was something. . .bricks. . .yellow. Gold! Someone must be stealing the gold from Fort Knox and tying it to balloons!

Suddenly, a black helicopter shoots between the clouds and snags a balloon. The crooks are trying to collect the loot!

You rush inside and grab your pogostick. You must rescue the gold for the United States. Or at least for yourself. . .

How to type (in three easy paragraphs)

Intrigued, or perhaps greedy? Stick the Action! cartridge in your computer, and type in Listing 1, using D:CHECK in Action! (issue 44) to check your work. I apologize for the long data lists in the program, but, hey, I don't have to type them in. . .

Save it! Not after you run it, not after you get a snack, *now!* Save it as "D:CLOUDHOP.ACT" on disk, or with the original name "C:" for cassette.

Now that **Cloudhopper** is saved (right?), go to the monitor, compile and run it.

How not to die

The game starts with a brazen title screen, for all to see.

Large, puffy cloud banks float peacefully back and forth. It is a pleasantly pastoral scene—except for the evil black helicopter and the hot lick of flames. SELECT will decrease the cloud density, and speed them and the helicopter up. Continued presses (five, for you picky people) will continue this process to an extreme whereupon the whole mess will return to the start. In technical talk, these are called "levels."

Once the thrill of flipping through all the levels has worn off, you can—wonder of wonders!—start the game. Can you guess how? That's right, press START! (Or use the trigger.)

You begin the game at the bottom of the screen, contentedly bobbing on a lower, middle cloud.

Suddenly a helicopter streaks by, only a few feet above your head. It banks, turns and heads for a second assault—this time on target!

A press on the trigger or a desperate push on the joystick will bounce your pogostick high enough to reach the next bank. Whether there are supporting clouds there or not is a different matter.

Left or right pushes on the stick, combined with a button press or upwardly diagonal shoves, will send your figure rocketing off at an angle.

The bottom-most layer of clouds is protected from the unsteady gusts of wind that rebound the other banks by the

lip of the volcano. This gives you a rather handy resting place, provided the 'copter doesn't draw a bead on you. However, below these nicely placed clouds lies the searing heat of the volcano; a slight slip will send you into it.

In case you're wondering what the helicopter does aside from look menacing, here are the instructions concerning that. . .The helicopter, when it touches you, will push you in the direction it is traveling. This is at the very least annoying and at the very most, deadly.

Now we get to the good stuff: the gold! To grab a brick of gold, just touch the balloon onto which it is fastened. The balloon will pop and the reward is yours.

Concerning points, the closer the balloon is to the flames, the more points it is worth. Point value also increases in higher levels.

You will not always be able to grab a balloon by just touching it though; sometimes your eyes will be, in the immortal words of Pink Floyd, "obscured by clouds," and it will require more than one wild stab to burst it.

When all the balloons are collected from the sky, you rocket out the top of the screen and, like magic, more appear.

In the unfortunate event of a dip in the volcano, press the trigger to bounce back onto the screen.

An oddity of nature should probably be mentioned here. After a while (1,000 points) of this silly bouncing, your jumpsuit will collect enough soot to act as an extra layer of insulation. This turns your suit black and gives you another chance to barbecue yourself.

After the last of the protecting cloth is burned from your body (when you take the plunge in a green suit), a press on the joystick button will return you to the title screen.

The SPACE BAR toggles the pause feature during play.

The end

So, valiant **Cloudhopper**, art thou prepared for thy sojourn above the very fires of hell? Gold be there. . .Go and conquer—'tis thine destiny.

Listing 1
Action! listing

```
; CLOUDHOPPER
; by Greg Knauss
; Copyright 1988 by ANALOG Computing
;
;       CHECKSUM DATA
;[18 D4 B3 2E 68 04 47 59
; 71 18 1B 42 6A 87 2F A5
; 43 14 66 25 C6 F3 5E 4D
; B6 7D 89 AC 89 8E D3 91
; EE 91 06 0F 15 B6 73 E9
; 1B 3C D3 B4 01 7B 6E 54
; 97 68 B8 C0 93 69 93 C7
; 31 ]

BYTE ST=[0],PM,Y,UP,X,S,P,OONS,ES,
 LIVES,CHX=[10],CHY=[40],FG
BYTE ARRAY CSPD(4),CX(4),CCNT(4)
INT X1,CX1=[1]
INT ARRAY CDIR(4)
CARD SC,J,I,CH,DRB,SCN,SCR=[0]

PROC DLIST()
[112 112 68 0 0 4 4 4 68 0 0 68 0 0
4 4 68 0 0 68 0 0 4 4 68 0 0 68 0 0
4 4 68 0 0 68 0 0 4 4 68 0 0 4
69 0 0 70 0 0 2 2 7 7 65 DLIST]
RETURN

PROC PLRVBI()
[162 3 189 244 6 240 89 56 221 240 6
240 83 141 254 6 106 141 255 6 142
253 6 24 169 0 109 253 6 24 109 252
6 133 204 133 206 189 240 6 133 203
173 254 6 133 205 189 248 6 170 232
46 255 6 144 16 168 177 203 145 205
169 0 145 203 136 202 208 244 76 87
6 160 0 177 203 145 205 169 0 145
203 200 202 208 244 174 253 6 173
254 6 157 240 6 189 236 6 240 48 133
203 24 138 141 253 6 109 235 6 133
204 24 173 253 6 109 252 6 133 206
189 240 6 133 205 189 248 6 170 160
0 177 203 145 205 200 202 208 248
174 253 6 169 0 157 236 6 202 48 3
76 2 6 76 98 228 0 0]
RETURN

PROC VBINIT()
[169 7 162 6 160 0 32 92 228 96]
RETURN

PROC CHSTORE()
[0 0 0 0 0 0 0 0
0 0 1 21 170 0 0 0
0 1 85 85 169 10 0 0
5 85 85 85 106 170 0 0
85 85 85 85 149 169 10 0
85 85 85 89 85 170 170 0
0 85 85 85 106 170 168 0
0 0 85 85 149 170 0 0
0 0 0 90 168 128 0 0
3 3 15 15 15 63 252 255
0 192 195 207 243 243 243 255
0 0 3 207 243 243 255 255
48 240 240 252 252 252 255 255
15 63 255 255 255 255 63 15
192 240 252 252 252 252 240 192
1 1 1 1 15 15 15 0
255 195 185 181 173 157 195 255
255 247 247 247 247 247 247 255
255 195 253 195 191 191 195 255
255 195 253 227 253 253 195 255
255 187 187 193 251 251 251 255
255 195 191 195 253 253 195 255
255 195 191 131 189 189 195 255
255 129 253 251 247 247 247 255
255 195 189 195 189 189 195 255
255 195 189 193 253 253 195 255
0 0 0 0 192 192 192 0
255 255 255 255 255 255 255 255
0 240 204 240 204 204 240 0
0 204 204 204 48 48 48 0
0 60 192 192 204 204 60 0
0 252 204 252 240 204 204 0
0 252 192 240 192 192 252 0
0 204 204 240 204 204 204 0
0 204 252 252 252 252 204 0
0 48 204 252 204 204 204 0
0 204 204 204 204 204 252 0
0 252 192 252 12 12 252 0
255 193 191 195 253 253 131 255
255 193 191 191 191 191 193 255
255 195 189 189 189 189 195 255
255 131 189 189 131 167 185 255
255 129 191 143 191 191 129 255
255 255 231 231 255 231 231 255
255 255 255 255 255 255 255 85
253 253 253 253 253 253 253 253
253 253 253 253 253 253 253 85
253 253 253 253 253 244 208 64
64 208 244 253 253 253 253 253
255 255 255 255 127 31 7 1
3 15 63 255 255 255 255 255
253 255 255 255 127 31 7 1
0 64 208 244 253 255 255 85
0 0 0 0 0 64 208 80]
RETURN

PROC GUYCLOTHES()
[56 60 0 0 36 126 255 90 126 60 60 60
```

```
126 126 24 0
28 60 0 0 36 126 255 90 126 60 60 60
126 126 24 0]
RETURN

PROC GUYFACE()
[0 0 40 60 24 0 0 129 126 24 24 24 24
126 24 0
0 0 20 60 24 0 0 129 126 24 24 24 24
126 24 0]
RETURN

PROC COPTER()
[240 16 56 77 143 222 124 48
30 16 56 77 143 222 124 48
15 8 28 178 241 123 62 12
120 8 28 178 241 123 62 12]
RETURN

PROC CLOUDS()
[5 6 7 6 7 6 5 4 5 6 7 8 0 0 0 1 2 3
 4 5 6 5 6 7 8 0 0 0 0 1 2 3 4 5 6 7
 6 7 8 0 0 1 2 3 4 5 4 5 6 7 6 7 2 8
 1 2 3 7 8 0 0 0 1 2 3 2 3 4 5 6 7 6
 7 8 1 8 0 0 0 0 1 2 3 7 8 0 0 1 2 3
 4 5 4 3 4 5 6 7 6 7 8 0 0 0 0 1 2 3
 4 5 6 7 8 0 0 0 1 2 3 4 5

 0 0 1 2 3 4 5 6 5 6 7 8 0 0 0 0 1 2
 3 4 5 7 8 0 0 0 0 1 2 3 4 5 4 5 6 7
 8 0 0 0 0 0 1 2 8 0 0 0 1 2 3 4 5 4
 6 7 8 0 0 0 0 0 0 0 0 1 2 3 4 5 6 7
 8 1 8 0 0 0 0 0 0 1 8 1 2 3 7 8 0 0
 0 0 0 0 0 1 2 3 4 5 6 5 4 5 6 7 8 0
 0 0 0 0 0 1 2 3 4 5 6 5

 0 0 0 0 0 1 2 3 4 6 7 8 0 0 0 0 0 1
 2 7 8 0 0 0 0 0 0 1 2 3 6 7 8 0 0 0
 1 2 3 4 5 4 5 6 5 6 7 8 0 0 0 0 0 1
 2 3 7 8 0 0 1 2 3 4 5 4 7 8 0 0 0 0
 0 1 8 0 0 0 0 1 2 3 4 6 7 8 1 8 1 2
 3 5 7 8 0 0 0 0 0 1 2 3 6 7 8 0 1 8
 0 0 0 1 2 3 4 5 6 5 6 7

 0 0 0 0 1 2 7 8 0 0 0 0 0 0 0 0 0 0
 1 8 0 0 0 0 0 0 0 0 0 1 7 8 0 0 0 0
 0 0 0 1 2 3 4 6 7 2 8 0 0 0 0 0 0 0
 1 2 3 6 7 8 1 2 7 8 0 0 0 0 0 0 1 2
 8 1 8 0 0 0 0 0 0 1 2 3 5 6 7 8 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 2 3 4 5 6
 5 6 7 8 0 0 0 1 2 8 0 0 0

 5 6 7 8 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 2 7 8 0 0 0 0 0 0 0 0 0 0 0 1 2 3
 4 5 6 5 4 5 6 7 8 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 2 3 4 5 4 5
 6 7 8 1 2 7 8 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 2 3 4 5 4 5 6 7 8 0 0

 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 1 2 3 4 3 4 5 6 7 6
 7 6 5 4 5 6 7 8 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0]

PROC SETCLOUDS()
FOR J=1 TO 120 DO
S=PEEK(CLOUDS+J+ST*120)
 POKE(J+SC+160,S)
 POKE(J+SC+640,S) OD
FOR J=1 TO 120 DO
S=PEEK(CLOUDS+J+(ST+1)*120)
 POKE(J+SC+400,S)
 POKE(J+SC+880,S) OD
 FOR J=1 TO 40 DO
  S=PEEK(CLOUDS+J+ST*120)
  POKE(SC+1120+J,S)
  I=RAND(4)+9 POKE(SC+1320+J,I)
  IF I=9 THEN I=13 FI
  POKE(SC+1280+J,I-1) OD
 RETURN

 PROC RNDCLD()
 CSPD(I)=RAND(3)+9-ST*2
 RETURN

 PROC MOVECLDS()
 FOR I=0 TO 3 DO
 CCNT(I)==+1
 WHILE CCNT(I)>CSPD(I) DO
 IF CX(I)=77 THEN CDIR(I)=-1 RNDCLD()
  FI
 IF CX(I)=0 THEN CDIR(I)=1 RNDCLD() FI
 CX(I)==+CDIR(I)
 POKEC(DLIST+I*8+9,SC+CX(I)+I*240+160)
 CCNT(I)=1
 OD OD
 RETURN

 PROC BALLOONS()
 P=0
 FOR I=5 TO 35 DO
 P=RAND(4)
 IF P=0 THEN J=1 FI
 IF P=1 THEN J=7 FI
 IF P=2 THEN J=13 FI
 IF P=3 THEN J=19 FI
 POSITION(I,J) PRINT("-.←↓Z ")
 I==+5 OD
 RETURN

 PROC PAUSE()
 SNDRST() POKE(764,255)
 WHILE PEEK(764)<>33 DO OD
 POKE(764,255)
 RETURN

 PROC PRNTSCR()
 POKEC(88,SCN)
 POSITION(12,0) PRINTC(SCR)
 POKEC(88,SC)
 RETURN

 PROC BURN()
 POKE(DLIST+45,SC+32*40+1)
 P=1 IF CX1=1 THEN P=17 FI
 POKE(1775,P)
 FOR I=1 TO 300 DO OD
 POKE(DLIST+45,SC+33*40+1)
 P=9 IF CX1=1 THEN P=25 FI
 POKE(1775,P)
 FOR I=1 TO 300 DO OD
 RETURN

 PROC CHMOVE()
 CHX==+CX1*((ST/2)+1)
 IF CHX<3 OR CHX>252 THEN CX1=-CX1
 CHY=RAND(5)*32+40 FI
 POKE(53251,CHX) POKE(1783,CHY)
 IF CHX<X THEN SOUND(1,190+(CHX MOD 4)
  *20,14,(CHX-X)/17-3) FI
 IF CHX>X THEN SOUND(1,190+(CHX MOD 4)
  *20,14,(X-CHX)/17-3) FI
 RETURN

 PROC BLOOGA()
 SOUND(0,0,0,0)
 I=3 J=(X-50)/4-1
 DO
 P=LOCATE(J,I)
 I==+1 IF I=24 THEN I=0 J==+1 FI
 UNTIL P=45 OD
 SOUND(0,50,10,10)
 POSITION(J,I-1) PRINT("   ←←←↓   ")
 OONS==+1 IF OONS=6 THEN E5=1 FI
 SCR==+10+I*(ST+1) PRNTSCR()
 UP=16 SOUND(0,0,0,0)
 RETURN

 PROC POS()
 POKE(53248,X) POKE(53249,X)
```

```
 POKE(53250,X)
POKE(1780,Y) POKE(1781,Y)
 POKE(1782,Y+16)
RETURN

PROC SETUP()
GRAPHICS(0) POKE(559,0) POKE(82,0)
 POKE(752,1) POKE(82,0)
SCN=PEEKC(88) SC=(PEEK(106)-16)*256
POKEC(DLIST+3,SC)
POKEC(DLIST+9,SC+4*40+1)
POKEC(DLIST+12,SC+7*40+1)
POKEC(DLIST+17,SC+10*40+1)
POKEC(DLIST+20,SC+13*40+1)
POKEC(DLIST+25,SC+16*40+1)
POKEC(DLIST+28,SC+19*40+1)
POKEC(DLIST+33,SC+22*40+1)
POKEC(DLIST+36,SC+25*40+1)
POKEC(DLIST+41,SC+28*40+1)
POKEC(DLIST+45,SC+32*40+1)
J=PEEKC(88) POKEC(DLIST+48,J)
POKEC(560,DLIST)
MOVEBLOCK(1536,PLRVBI,160)
PM=PEEK(106)-32 DRB=PM*256+1
ZERO(DRB+1024,1024) POKE(623,36)
POKE(1788,PM+4)
POKE(53277,3) POKE(54279,PM)
POKE(1771,PM)
POKE(1784,16) POKE(1785,16)
 POKE(1786,1) POKE(1787,8)
POKE(1772,17) POKE(1773,17)
 POKE(1774,1) POKE(1775,1)
POKE(53259,3)
MOVEBLOCK(DRB,GUYFACE,32)
MOVEBLOCK(DRB+256,GUYCLOTHES,32)
FOR I=0 TO 256 DO POKE(DRB+I+512,24)
 OD
MOVEBLOCK(DRB+256*3,COPTER,32)
VBINIT()
POKE(704,30) POKE(705,84) POKE(706,15)
POKE(707,2) POKE(708,15) POKE(709,8)
POKE(710,52) POKE(711,42)
POKE(712,7*16)
POSITION(0,0)
PRINTE("     FGHIJK         ")
PRINT("                    ")
CH=(PEEK(106)-24)*256
MOVEBLOCK(CH,CHSTORE,512)
POKE(756,CH/256)
ST=0
FOR I=0 TO 4 DO
 CDIR(I)=1
 CSPD(I)=RAND(3)+10-ST*2
 CX(I)=RAND(70)+1
 CCNT(I)=1 OD
ZERO(SC,2000) SETCLOUDS()
RETURN

PROC TITLE()
POKE(53278,0) POKE(77,0) POKEC(88,SC)
POSITION(0,7)
PRINTE
 ("          RLN M   RLP M M ;LP")
PRINTE
 ("          M   M   M M M M M M")
PRINTE
 ("          QLN LLN QLO QLO LLO")
POSITION(0,13)
PRINTE
 ("        M M RLP ;;P ;;P ;LN ;;P")
PRINTE
 ("        ;LM M M ;LO ;LO ;N  ;;O")
PRINTE
 ("        N N QLO N   N   LLN NSTU")
POSITION(14,20)
PRINT("(=  )?@) ABCDEE")
PRNTSCR() SETCLOUDS() MOVECLDS()
POKE(559,62) X=120 Y=169 POS()
DO
MOVECLDS() BURN() CHMOVE()
IF PEEK(53279)=5 THEN ST==+1
 IF ST=5 THEN ST=0 FI SETCLOUDS()
 FOR I=0 TO 3 DO RNDCLD() OD FI
UNTIL PEEK(53279)=6 OR STRIG(0)=0 OD
POKE(764,255)
ZERO(SC+7*40,120)
ZERO(SC+13*40,120)
ZERO(SC+20*40,40)
SCR=0 LIVES=3 UP=0 X1=0 OONS=0 FG=0
POKEC(88,SCN)
POSITION(12,0) PRINT("0    ")
POKEC(88,SC)
RETURN

PROC MAIN()
DO
SETUP() TITLE()
BALLOONS()
WHILE LIVES>0 DO
X1=0 X=120
DO
ES=0
WHILE ES=0 DO
POKE(77,0) MOVECLDS()
S=STICK(0)
P=PEEK(53254)
IF S=7 THEN POKE(1772,1) POKE(1773,1)
 FI
IF S=11 THEN POKE(1772,17)
 POKE(1773,17) FI
IF P=1 OR P=2 THEN
 IF S=7 THEN X1=1 FI
 IF S=11 THEN X1=-1 FI FI
IF PEEK(53263)=3 THEN X1=CX1 FI
IF X>200 THEN X1=-1 X==-1 FI
IF X<50 THEN X1=1 X==+1 FI
X==+X1
IF UP=0 THEN SOUND(0,0,0,0) FI
Y==+1
IF UP=0 AND (P=1 OR P=2) THEN UP=5
 X1=0
 IF S=14 OR S=10 OR S=6 OR STRIG(0)=0
  THEN UP=25 FI
 IF S=10 OR S=11 THEN X1=-1 FI
 IF S=6 OR S=7 THEN X1=1 FI FI
IF Y<5 THEN UP=0 FI
IF UP=0 AND (P=1 OR P=2) THEN
 UP=5 FI
IF UP>0 THEN UP==-1 Y==-2
 SOUND(0,UP*5+50,10,4) FI
IF UP>5 THEN Y==-1 FI
POS()
IF PEEK(53252)=4 AND Y<180
 THEN BLOOGA() FI
POKE(53278,0)
IF Y=180 AND UP=0 THEN ES=2
 FOR J=181 TO 240 DO MOVECLDS() Y==+1
 X==+X1 POS() CHMOVE() BURN() OD FI
BURN() CHMOVE()
IF PEEK(764)=33 THEN PAUSE() FI
IF SCR>999 AND FG=0 THEN FG=PEEK(705)
 POKE(705,2) LIVES==+1 FI
OD
IF OONS=6 THEN FOR J=4 TO Y/3 DO Y==-3
 POS() BURN() CHMOVE() MOVECLDS()
 SOUND(0,Y+50,10,4) OD
 SNDRST() X=120 Y=1 UP=0 X1=0 POS()
 ST==+1 IF ST=5 THEN ST=3 FI
 SETCLOUDS() BALLOONS() OONS=0 FI
UNTIL ES=2 OD
P=PEEK(705)
IF P=84 THEN S=118 FI
IF P=118 THEN S=184 FI
POKE(705,S)
IF FG>1 THEN POKE(705,FG) FG=1 FI
LIVES==-1
Y=240 UP=40
WHILE STRIG(0)=1 DO MOVECLDS() BURN()
CHMOVE()
POKE(764,255)
OD
X=0 POS()
OD
OD
●
```

GRAPHICS

48K Disk

The APAC System

The "Any Point, Any Color" system that gives you access to an 80 by 96 graphics mode with 256 colors

by Thomas Tanida

How would you like a graphics mode with 80 by 96 resolution? "Big deal," you say, since one doesn't even need an ST for that. How about if I said you could get this *and* the ability to put any of 256 colors *anywhere* on the screen, without resorting to display list interrupts or such tricks, in just 8K of memory? You *can* do this from your Atari 8-bit, just by using a function as simple as a BASIC PLOT statement.

The wonder behind all of this is something I call the **APAC System**, meaning "Any Point, Any Color." It gives you access to an 80 by 96 graphics mode with 256 colors. Just a single USR statement from BASIC (or a JSR from machine language), and this special graphics mode is set up. A few more USRs and you'll be doing PLOTs and DRAWTOs with hardly any effort.

Listing 1 is the object code for **APAC**. Type it in using the ANALOG M/L Editor (found elsewhere in this issue). After saving the object code to disk, you may load it from DOS as a binary file, or you may rename it AUTORUN.SYS and boot the disk containing it. It will install itself beginning at location 8960 ($2300) and move the OS MEMLO pointers up to protect itself. It is also RESET-proof.

Listing 2 (the longer BASIC program) is an Atari BASIC demonstration program for the **APAC System**. Type this program in and save it to disk. Run it, provided **APAC** has been loaded, and let it go for a few minutes. Press START to exit, OPTION to clear the screen. Then type in Listing 3, save it, and run it for another demonstration. You'll be impressed. For best effects, turn up the brightness and color levels of your television or monitor.

Listing 4 is a machine language kaleidoscope demonstration program. To use it, enter it using M/L Editor. From DOS, load the demo as a binary file and run at $6000. You must have **APAC** loaded in for the demo to work, since it accesses **APAC**. While the kaleidoscope demo runs, you may press the SPACE BAR to pause, ESCAPE to exit, or START to clear the screen.

Warning: If you type DOS from BASIC, any program you have in memory will be erased. Always save your program to disk before entering DOS. Also, if you are using Atari DOS 2.OS, you should create a MEM.SAV file if you want to keep **APAC** in memory, or load **APAC** from the DOS menu upon leaving DOS. **APAC** is erased when DUP.SYS is loaded.

Listings 5 and 6 are the MAC/65 source code for the **APAC System** and the kaleidoscope demo, respectively, and are included only for those people interested in assembly language programming.

How to use *APAC* with your own programs

With BASIC: Examine Line 30000 of either BASIC demonstration program. This reveals the following locations:

8960 — The PLOT routine
8963 — The DRAWTO routine
8966 — The EXIT APAC routine
8969 — The INIT APAC routine

INIT APAC — Usage: Q=USR(8969). This turns on the APAC graphics mode.

EXIT APAC — Usage: Q=USR(8966). This restores a Graphics 0 screen. This call to the APAC System is necessary to disable APAC because it re-

stores certain OS interrupt pointers. For more on this, see "How it works," below.

PLOT — Usage: Q=USR(8960,xpos,ypos,color). The "xpos" represents the x-position (0-79) and "ypos" represents the y-position (0-95). "Color" is any color (0-255), calculated by the function 16*color+luminance, which is exactly the same as if you were poking it into registers 704-712. The current color is always kept in location 203 ($CB).

DRAWTO — Usage: Q=USR(8963,xpos,ypos, color). The "xpos" represents the destination x-position, and "ypos" represents the destination y-position. "Color" is the same as above. You *must* specify the color. It will draw a line from the current coordinate position specified in the last PLOT or DRAWTO made (or 0,0 if you are just starting out). If you would like to use DRAWTO with the color last used and you don't know what that color is, you can use: Q=USR(8963,xpos,ypos,PEEK (203)). For all of the above functions, if you attempt to pass too many or too few parameters to them, the function will be aborted. Also, if you try to plot beyond the screen boundaries, the function will be aborted.

To use **APAC** from a machine language program, the key locations are: $230C (8972) = PLOT; $230F (8975) = DRAWTO; $2312 (8978) for EXIT APAC; and $2315 (8981) for INIT APAC. Make sure your program does not overlap locations $2300 through the value contained in MEMLO ($02E7-$02E8), or a system crash will most likely occur. The much used and abused page six is left untouched. Locations 203-222 ($CB-$DE) are altered by **APAC**. JSR to the locations given above as follows:

PLOT: acc = color; x-reg = x-pos; y-reg = y-pos.

DRAWTO: acc= color; x-reg= destination x-pos; y-reg= destination y-pos.

INIT and EXIT: All registers ignored and altered.

Note: Remember that the current color is always kept in location $CB (203). Also, from machine language there is no error checking for the PLOT and DRAWTO routines.

To use **APAC** with the Action! language, make the following declarations within your program:

```
PROC InitAPAC=$2315()
PROC ExitAPAC=$2312()
PROC PlotAPAC=$230C(BYTE xpos, ypos, col)
PROC DrawAPAC=$230F(BYTE xpos, ypos, col)
```

Call InitAPAC() to set up the **APAC** graphics mode, and ExitAPAC() to clear the **APAC** graphics mode and return to Graphics 0. Call PlotAPAC with three parameters to plot a point in the given color (col) at the given coordinates (xpos, ypos) and call DrawAPAC to draw a line from the last plotted point to the given coordinates (xpos, ypos) in the given color (col).

How it works

When I said you would need no display list interrupts, I fibbed. **APAC** uses DLIs on every line of the screen in a special way that lets you show any of 256 colors at any pixel. It essentially sets up a graphics mode with a line of GTIA Graphics mode 9, followed by a line of Graphics mode 11, followed, followed by 9, and so on. When this shows up on the screen, the luminance of Graphics 9 and color of Graphics 11 blend together. Since Graphics 9 has 16 luminances and Graphics 11 has 16 hues, you get 16 times 16(256)colors. The blending effect is what causes those horizontal blank lines most of the time; what appears to be the blank lines are really the Graphics 11 lines.

In addition to the DLIs, there is an immediate mode VBI which keeps the screen in sync. But to use **APAC**, all you have to do is call the INIT APAC, EXIT APAC, PLOT and DRAWTO routines correctly and **APAC** will do the rest. The DRAWTO routine uses the same vector algorithm published in Tom Hudson's "BASIC Training" (issue 18). I translated the BASIC listing into machine language and optimized the code greatly. (I suspect Tom had translated the BASIC code from machine language since it was a bit inefficient, but since his first name is Tom, he must be a good programmer.)

Maybe in the future, I'll see some really good applications for **APAC**: multi-colored, three dimensional spheres floating in space, or a fancy drawing program. Oh yes, there is yet *another* way to get 256 colors on the screen at any point, with 80 by 192 resolution and *no* blank lines, requiring 16K or memory and absolutely *no* DLIs (I promise!) Interested? I'll leave that up to you, the reader, to figure out how it might be one, but here's a hint: Use both Graphics 9 and 11, but in a different method than **APAC**. Will there be an APAC-II? We'll see. . .

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

Listing 1
M/L Editor listing

```
1000 DATA 255,255,253,34,248,35,76,0,6
4,76,24,35,76,136,35,76,1295
1010 DATA 57,36,76,149,36,76,43,35,76,
163,35,76,63,36,76,155,1734
1020 DATA 36,104,201,3,240,3,76,152,37
,104,104,170,104,104,168,104,6394
1030 DATA 104,32,139,37,133,203,134,85
,132,84,185,211,37,133,204,185,222
1040 DATA 51,38,133,205,138,74,168,177
,204,133,206,138,41,1,170,165,8937
1050 DATA 206,61,134,35,133,206,165,20
3,224,1,208,5,41,15,76,92,4329
1060 DATA 35,10,10,10,10,5,206,145,204
,24,165,204,105,40,133,204,7390
1070 DATA 144,2,230,205,177,204,61,134
,35,133,206,165,203,224,1,208,1345
1080 DATA 7,74,74,74,74,76,129,35,41,2
40,5,206,145,204,96,15,5479
1090 DATA 240,104,201,3,240,3,76,152,3
7,104,104,170,104,104,168,165,7644
1100 DATA 85,133,212,165,84,133,213,10
4,104,32,139,37,134,221,132,215,9773
1110 DATA 32,43,35,164,215,162,6,169,0
,149,213,202,208,251,166,221,3931
1120 DATA 228,212,240,20,144,7,138,229
,212,230,214,208,9,198,214,134,3433
1130 DATA 221,165,212,56,229,221,133,2
18,196,213,240,20,144,7,152,229,2375
1140 DATA 213,230,215,208,9,198,215,13
2,221,165,213,56,229,221,133,219,5308
1150 DATA 165,218,197,219,144,4,133,21
7,176,4,165,219,133,216,133,222,2996
1160 DATA 133,220,249,35,244,36,70,216
,70,217,165,222,240,55,165,216,3374
1170 DATA 24,101,218,133,216,197,220,1
44,11,229,220,133,216,165,212,24,2623
1180 DATA 101,214,133,212,165,217,24,1
01,219,133,217,197,220,144,11,229,2816
1190 DATA 220,133,217,165,213,24,101,2
15,133,213,166,212,164,213,32,45,634
1200 DATA 35,198,222,208,201,96,104,24
0,3,76,152,37,173,11,212,201,9060
1210 DATA 64,176,249,173,145,36,141,34
,2,173,146,36,141,35,2,169,4108
1220 DATA 64,141,14,212,162,0,169,12,1
```

```
41,66,3,32,86,228,162,0,3631
1230 DATA 142,75,3,169,3,141,66,3,169,
147,141,68,3,169,36,141,4113
1240 DATA 69,3,169,12,141,74,3,32,86,2
28,169,127,141,14,212,169,7646
1250 DATA 147,141,231,2,133,128,169,38
,141,232,2,133,129,96,0,0,3528
1260 DATA 69,58,104,240,3,76,152,37,16
9,0,141,47,2,141,14,212,3786
1270 DATA 133,84,133,85,133,86,141,203
,2,133,203,133,204,141,48,2,6657
1280 DATA 168,165,106,56,233,32,133,20
5,141,49,2,169,112,145,204,200,9803
1290 DATA 192,3,208,249,169,144,145,20
4,169,207,200,145,204,169,0,200,3213
1300 DATA 145,204,133,88,166,205,232,1
38,133,89,200,145,204,200,169,143,3694
1310 DATA 145,204,200,192,200,208,249,
160,102,169,207,145,204,169,0,200,3345
1320 DATA 145,204,245,36,210,37,165,10
6,56,233,16,200,145,204,160,200,1778
1330 DATA 169,65,145,204,200,169,0,145
,204,200,165,205,145,204,165,88,2789
1340 DATA 133,0,165,89,133,1,24,105,30
,133,221,160,0,152,145,0,4257
1350 DATA 136,208,251,230,1,165,1,197,
221,144,240,165,89,133,1,160,9796
1360 DATA 0,165,0,153,211,37,165,1,153
,51,38,24,165,0,105,80,2335
1370 DATA 133,0,144,2,230,1,200,192,96
,208,230,173,34,2,141,145,8490
1380 DATA 36,173,35,2,141,146,36,169,6
,162,37,160,161,32,92,228,6630
1390 DATA 169,179,141,0,2,169,37,141,1
,2,169,64,141,111,2,169,3528
1400 DATA 192,141,14,212,169,34,141,47
,2,96,32,255,255,169,121,133,9190
1410 DATA 12,133,2,169,37,133,13,133,3
,76,130,36,224,80,144,2,3381
1420 DATA 176,4,192,96,144,227,104,104
,96,170,240,5,104,104,202,208,636
1430 DATA 251,96,72,169,192,141,14,212
,169,179,141,0,2,141,203,2,6304
1440 DATA 104,76,95,228,72,169,195,141
,0,2,169,64,141,10,212,141,6816
1450 DATA 27,208,104,64,72,169,179,141
,0,2,169,192,141,10,212,141,7808
1460 DATA 27,208,104,64,0,64,24,64,165
,12,141,122,37,165,13,141,3397
1470 DATA 123,37,32,124,37,165,6,208,1
,96,169,0,133,8,76,0,949
1480 DATA 160,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,1640
```

Listing 2
BASIC listing

```
AR 1 REM * APAC DEMONSTRATION #1
NC 10 GRAPHICS 0:GOSUB 30000
YC 20 L1XP1=32:L1XP2=48:L1YP1=58:L1YP2=38
   :HUE1=0
KX 30 L2XP1=32:L2XP2=48:L2YP1=38:L2YP2=58
   :HUE2=136
QK 40 GOSUB 1000:L1XP1=L1XP1+RAND
5D 50 GOSUB 1000:L1XP2=L1XP2+RAND
5A 60 GOSUB 1000:L1YP1=L1YP1+RAND
TT 70 GOSUB 1000:L1YP2=L1YP2+RAND
LD 80 Q=USR(PLTAP,L1XP1,L1YP1,HUE1)
IM 90 Q=USR(DRW2AP,L1XP2,L1YP2,HUE1)
IK 100 L1XP1=L1XP1+(L1XP1(1)-(L1XP1)78)
LM 110 L1XP2=L1XP2+(L1XP2(1)-(L1XP2)78)
JV 120 L1YP1=L1YP1+(L1YP1(1)-(L1YP1)95)
MX 130 L1YP2=L1YP2+(L1YP2(1)-(L1YP2)95)
PQ 140 HUE1=HUE1+1:IF HUE1=256 THEN HUE1=
   0
OF 150 GOSUB 1000:L2XP1=L2XP1+RAND
QB 160 GOSUB 1000:L2XP2=L2XP2+RAND
PZ 170 GOSUB 1000:L2YP1=L2YP1+RAND
RV 180 GOSUB 1000:L2YP2=L2YP2+RAND
SE 190 Q=USR(PLTAP,L2XP1,L2YP1,HUE2)
QV 200 Q=USR(DRW2AP,L2XP2,L2YP2,HUE2)
LB 210 L2XP1=L2XP1+(L2XP1(1)-(L2XP1)78)
OD 220 L2XP2=L2XP2+(L2XP2(1)-(L2XP2)78)
MM 230 L2YP1=L2YP1+(L2YP1(1)-(L2YP1)95)
PO 240 L2YP2=L2YP2+(L2YP2(1)-(L2YP2)95)
SW 250 HUE2=HUE2+1:IF HUE2=256 THEN HUE2=
   0
EC 260 CONS=PEEK(53279)
EF 270 IF CONS=3 THEN Q=USR(INITGR)
LE 280 IF CONS=6 THEN Q=USR(EXITGR):END
QI 300 GOTO 40
XI 1000 RAND=INT(RND(1)*3)-1
AC 1010 RETURN
FZ 30000 INITGR=8969:PLTAP=8960:DRW2AP=89
   63:EXITGR=8966
C5 30005 Q=USR(INITGR)
DD 30010 RETURN
```

Listing 3
BASIC listing

```
KU 1 REM * APAC DEMONSTRATION #2 *
IQ 5 GOSUB 30000
QK 10 IF COL>256 THEN COL=COL-256
TZ 15 FOR Y=0 TO 79
EU 20 COL=COL+1
PK 30 Q=USR(PLTAP,0,Y,COL):Q=USR(DRW2AP,Y
   ,0,COL)
OG 40 NEXT Y
TL 50 FOR X=0 TO 79
EY 60 COL=COL+1
IR 70 Q=USR(PLTAP,X,79,COL):Q=USR(DRW2AP,
   79,X,COL)
OB 80 NEXT X
RQ 90 GOTO 10
FZ 30000 INITGR=8969:PLTAP=8960:DRW2AP=89
   63:EXITGR=8966
C5 30005 Q=USR(INITGR)
DD 30010 RETURN
```

Listing 4
M/L Editor data

```
1000 DATA 255,255,0,96,251,96,32,21,35
,162,39,134,141,134,142,232,7895
1010 DATA 134,144,169,0,162,8,149,132,
202,208,251,133,132,169,255,141,3314
1020 DATA 252,2,32,90,96,32,90,96,173,
10,210,41,31,208,3,32,2133
1030 DATA 126,96,32,144,96,230,132,173
,31,208,201,6,240,200,173,252,3377
1040 DATA 2,201,33,240,7,201,28,208,21
2,76,18,35,173,241,2,208,7871
1050 DATA 251,169,255,141,252,2,173,25
2,2,201,33,208,249,76,23,96,8536
1060 DATA 162,2,181,134,32,120,96,149,
134,202,208,246,32,166,96,162,287
1070 DATA 2,181,132,32,120,96,149,132,
202,208,246,76,166,96,73,255,1426
1080 DATA 24,105,1,96,162,4,173,10,210
,41,3,240,249,56,233,2,6587
1090 DATA 149,136,202,208,241,96,162,4
,181,132,24,117,136,201,40,144,7999
1100 DATA 5,56,233,40,176,247,149,132,
202,208,237,96,160,4,190,242,3052
1110 DATA 96,185,132,0,24,117,141,190,
248,96,149,150,136,208,239,32,709
1120 DATA 207,96,160,4,190,244,96,185,
132,0,24,117,141,190,250,96,9216
1130 DATA 149,150,136,208,239,165,132,
166,150,164,151,32,12,35,165,132,7524
1140 DATA 166,152,164,153,32,15,35,165
,132,166,151,164,152,32,12,35,4170
1150 DATA 165,132,166,150,164,153,76,1
5,35,0,0,1,1,0,0,0,5407
1160 DATA 2,1,252,96,254,96,3,0,2,0,0,
0,0,0,0,0,4189
```

Listing 5
Assembly listing

```
10 ********************************
20 * APAC SYSTEM, V1.1            *
30 * CREATED BY THOMAS TANIDA     *
40 * FIRST STARTED: 1/3/87        *
50 * LAST REVISED: 2/7/87         *
60 ********************************
70        *=  $22FD
80        .OPT NO EJECT
90        .TAB 8,12,30
0100 ********************************
0110 ZTEMP = $00
0120 COLSAV = 203
0130 YBYT =  204
0140 T8   =  206
0150 FX   =  212      ;212-222 USED
0160 FY   =  213      ; FOR DRAWTO
0170 XD   =  214
0180 YD   =  215
0190 XACC =  216
0200 YACC =  217
0210 DELTX = 218
0220 DELTY = 219
0230 EPOINT = 220
0240 TMP1 =  221
0250 COUNT = 222
0260 ********************************
0270 *      SYSTEM EQUATES          *
0280 ********************************
0290 CASINI = $02     ;2
0300 TRAMSZ = $06     ;6
0310 WARMST = $08     ;8
0320 DOSVEC = $0A     ;10
0330 DOSINI = $0C     ;12
0340 ROWCRS = $54     ;84
0350 COLCRS = $55     ;85
0360 SCREEN = $58     ;88
0370 RAMTOP = $6A     ;106
0380 LOMEM = $80      ;128
0390 VDSLST = $0200   ;512
0400 VVBLKI = $0222   ;546
0410 SDMCTL = $022F   ;559
0420 SDLSTL = $0230   ;560
0430 GPRIOR = $026F   ;623
0440 COLOR4 = $02CB   ;712
0450 MEMLO = $02E7    ;743
0460 ICCOM = $0342    ;834
0470 ICBADR = $0344   ;836
0480 ICBLEN = $0348   ;840
0490 ICAUX1 = $034A   ;842
0500 COLBK = $D01A    ;53274
0510 PRIOR = $D01B    ;53275
0520 WSYNC = $D40A    ;54282
0530 VCOUNT = $D40B   ;54283
0540 NMIEN = $D40E    ;54286
0550 CIOV =  $E456    ;58454
0560 SETVBV = $E45C   ;58460
0570 SYSVBV = $E45F   ;58463
0580 ********************************
0590 * (FOR BINARY LOADS)
0600      JMP SYSSET
0610 *
0620 * BASIC ENTRY POINTS:
0630      JMP BASPLT  ;PLOT
0640      JMP BASDRW2 ;DRAWTO
0650      JMP BASEXGT ;EXIT APAC
0660      JMP BASINGT ;INIT APAC
0670 * ML ENTRY POINTS:
0680      JMP PLT256  ;PLOT
0690      JMP DRT256  ;DRAWTO
0700      JMP EXITGT  ;EXIT APAC
0710      JMP INITGT  ;INIT APAC
0720 ********************************
0730 * BASIC ENTRY:
0740 * Q=USR(BASPLT,X,Y,COL)
0750 *
0760 * THIS SUBR PLOTS A POINT
0770 * ON THE APAC SCREEN AT THE
0780 * GIVEN X,Y POSITION USING THE
0790 * GIVEN COLOR (0-255).
0800 * REMEMBER: THERE ARE 192 LINES,
0810 * ALTERNATING LUM AND COL,
0820 * SO THE SCREEN LIMITS ARE
0830 * 80 HORIZONTAL AND 192/2=96
0840 * VERTICAL
0850 *
0860 * FIRST MAKE SURE THERE ARE 3
0870 * (NO MORE, NO LESS) ARGUMENTS
0880 *
0890 BASPLT
0900      PLA
0910      CMP #3
0920      BEQ GETPARM
0930      JMP TRAP      ;WRONG # OF ARGS
0940 GETPARM
0950      PLA           ;IGNORE HI-BYTE
0960      PLA
0970      TAX           ;X-POS
0980      PLA           ;IGNORE HI
0990      PLA
1000      TAY           ;Y-POS
1010      PLA
1020      PLA           ;COLOR
1030 * CHECK FOR ILLEGAL PLOT
1040      JSR BOUNDCK
1050 * ML ENTRY: A=COLOR
1060 *            X=X-POS
1070 *            Y=Y-POS
1080 PLT256
1090      STA COLSAV  ;SAVE COLOR
1100 PSAMCOL
1110      STX COLCRS  ;UPDATE CURSOR
1120      STY ROWCRS
1130 * GET LO-BYTE OF SCREEN MEM PNTR
1140      LDA SCRALO,Y
1150      STA YBYT
1160 * GET HI-BYTE OF SCREEN MEM PNTR
1170      LDA SCRAHI,Y
1180      STA YBYT+1
1190      TXA
1200      LSR A         ;A=A/2
1210      TAY
1220 * GET BYTE WITH PIXEL FROM THE
1230 * SCREEN CONTAINING THE LUM
1240      LDA (YBYT),Y
1250      STA T8        ;SAVE IT
1260      TXA
1270      AND #1        ;X=1=ODD X-POS
1280      TAX           ;X=0=EVEN
1290      LDA T8
1300 * MASK PREVIOUS PIXEL FROM SCRN
1310      AND BITABL,X
1320      STA T8
1330 * RETRIEVE COLOR
1340      LDA COLSAV
1350      CPX #1
1360      BNE LHNYB     ;X IS EVEN
1370 * THIS WORKS SINCE THE LUM
1380 * IS IN THE LOWER NYBBLE OF THE
1390 * COLOR, AND IF X IS ODD THEN
1400 * THE LOWER NYBBLE OF THE SCR
1410 * BYTE WILL CONTAIN THE PIXEL
1420      AND #$0F
1430      JMP POKLUM
1440 * MOVE LUM INTO HI-NYBBLE
1450 LHNYB
1460      ASL A
1470      ASL A
1480      ASL A
1490      ASL A
1500 * POKE THE LUM BACK INTO THE SCR
1510 POKLUM
1520      ORA T8
1530      STA (YBYT),Y
1540      CLC
1550      LDA YBYT
1560      ADC #40       ;ADD 40 FOR THE
1570      STA YBYT      ;NXT LINE (COLR)
1580      BCC GETCOLR
1590      INC YBYT+1
1600 * GETS THE BYTE CONTAINING THE
1610 * PIXEL WITH THE COLOR AND SAVE
1620 * IN T8
1630 GETCOLR
1640      LDA (YBYT),Y
1650      AND BITABL,X
1660      STA T8
1670      LDA COLSAV
1680      CPX #1
1690      BNE CHNYB
1700 * SINCE THE COLOR ITSELF IS IN
1710 * THE HI-NYBBLE OF THE GIVEN
```

```
1720 * COLOR, MOVE IT TO THE LOWER
1730 * NYBBLE SINCE THE X-POS IS ODD
1740      LSR A
1750      LSR A
1760      LSR A
1770      LSR A
1780      JMP POKCOL
1790 CHNYB
1800      AND #$F0     ;SINCE XPOS EVEN
1810 POKCOL
1820      ORA T8
1830 * PUT IT INTO THE SCREEN
1840      STA (YBYT),Y
1850      RTS
1860 * BIT MASKS TABLE
1870 BITABL .BYTE $0F,$F0
1880 ********************************
1890 * BASIC ENTRY:
1900 * Q=USR(BASDRW2,X,Y,COL)
1910 *
1920 * THIS SUBR DRAWS A LINE FROM
1930 * THE CURRENT CURSOR POSITION
1940 * TO THE GIVEN X,Y USING
1950 * THE GIVEN COLOR
1960 *
1970 * FIRST MAKE SURE THERE ARE 3
1980 * (NO MORE, NO LESS) ARGUMENTS
1990 *
2000 BASDRW2
2010      PLA
2020      CMP #3
2030      BEQ GETARGS
2040      JMP TRAP     ;WRONG # OF ARGS
2050 GETARGS
2060      PLA
2070      PLA
2080      TAX          ;DEST-X
2090      PLA
2100      PLA
2110      TAY          ;DEST-Y
2120      LDA COLCRS   ;CURSOR X-POS
2130      STA FX       ;FROM-X
2140      LDA ROWCRS   ;CURSOR Y-POS
2150      STA FY       ;FROM-Y
2160      PLA
2170      PLA
2180 * CHECK FOR ILLEGAL DRAWTO
2190      JSR BOUNDCK
2200 * ML ENTRY: A=COLOR
2210 *           X=DEST-X
2220 *           Y=DEST-Y
2230 DRT256
2240      STX TMP1     ;SAVE X
2250      STY YD       ;SAVE Y
2260 * PLOT THE DESTINATION PIXEL
2270      JSR PLT256
2280      LDY YD       ;RESTORE Y
2290 * LOOP TO SET XD,YD,XACC,YACC,
2300 * DELTX,DELTY TO 0
2310      LDX #6
2320      LDA #0
2330 POKZ0
2340      STA XD-1,X
2350      DEX
2360      BNE POKZ0
2370 * THE FOLLOWING ROUTINE IS BASED
2380 * ON THE VECTOR ROUTINE PRINTED
2390 * IN ANALOG, ISSUE #18, IN THE
2400 * "BASIC TRAINING" COLUMN BY
2410 * TOM HUDSON
2420 * INIT THE X VARIABLES
2430      LDX TMP1
2440      CPX FX
2450      BEQ SGNY
2460      BCC NEGXD
2470      TXA          ;DELTX=TX-FX
2480      SBC FX       ;CARRY WAS SET
2490      INC XD       ;XD=1
2500      BNE SAVDLX   ;ALWAYS
2510 NEGXD
2520      DEC XD       ;XD=-1 ($FF)
2530      STX TMP1
2540      LDA FX       ;DELTX=FX-TX
2550      SEC
2560      SBC TMP1
2570 SAVDLX
2580      STA DELTX
2590 * INIT THE Y VARIABLES
2600 SGNY
2610      CPY FY
2620      BEQ INTCNT
2630      BCC NEGY
2640      TYA          ;DELTY=YD-FY
2650      SBC FY       ;CARRY WAS SET
2660      INC YD       ;YD=1
2670      BNE SAVDLY   ;ALWAYS
2680 NEGY
2690      DEC YD       ;YD=-1 ($FF)
2700      STY TMP1
2710      LDA FY       ;DELTY=FY-TY
2720      SEC
2730      SBC TMP1
2740 SAVDLY
2750      STA DELTY
2760 INTCNT
2770      LDA DELTX
2780      CMP DELTY
2790      BCC XLTY
2800      STA YACC
2810      BCS INTEPT   ;ALWAYS
2820 * (X LESS THAN Y)
2830 XLTY
2840      LDA DELTY
2850      STA XACC
2860 INTEPT
2870      STA COUNT
2880      STA EPOINT
2890      LSR XACC     ;XACC=XACC/2
2900      LSR YACC     ;YACC=YACC/2
2910 * EXIT IF DESTINATION X,Y
2920 * SAME AS CURRENT X,Y
2930      LDA COUNT
2940      BEQ EXDRT
2950 *
2960 * THE MAIN LOOP!
2970 XCALC
2980      LDA XACC     ;CHANGE X
2990      CLC
3000      ADC DELTX
3010      STA XACC
3020      CMP EPOINT
3030      BCC YCALC
3040      SBC EPOINT   ;CARRY WAS SET
3050      STA XACC
3060      LDA FX
3070      CLC
3080      ADC XD
3090      STA FX
3100 YCALC
3110      LDA YACC     ;CHANGE Y
3120      CLC
3130      ADC DELTY
3140      STA YACC
3150      CMP EPOINT
3160      BCC PLIT
3170      SBC EPOINT   ;CARRY WAS SET
3180      STA YACC
3190      LDA FY
3200      CLC
3210      ADC YD
3220      STA FY
3230 * PLOT THE CALCULATED POINT
3240 PLIT
3250      LDX FX
3260      LDY FY
3270      JSR PSAMCOL ;PLOT SAME COLOR
3280      DEC COUNT
3290      BNE XCALC    ;DO MORE POINTS
3300 EXDRT
3310      RTS
3320 ********************************
3330 * BASIC ENTRY:
3340 * Q=USR(BASEXGT)
3350 *
3360 * THIS SUBR RESTORES THE SCREEN
3370 * TO GRAPHICS 0 AND RESETS OS
3380 * INTERRUPT POINTERS, ETC.
```

```
3390 * IT STILL PROTECTS APAC
3400 *
3410 BASEXGT
3420     PLA
3430     BEQ EXITGT  ;NO PARAMETERS!
3440     JMP TRAP
3450 * ML ENTRY:
3460 *  (REGISTERS IGNORED)
3470 EXITGT
3480     LDA VCOUNT  ;MAKE SURE
3490     CMP #64     ;WE AREN'T NEAR
3500     BCS EXITGT  ;A VBI
3510     LDA SAVVBI  ;RESTORE OS VBI
3520     STA VVBLKI
3530     LDA SAVVBI+1
3540     STA VVBLKI+1
3550 *
3560     LDA #64
3570     STA NMIEN   ;NO DLI'S
3580     LDX #$00    ;IOCB 0
3590     LDA #$0C    ;CLOSE
3600     STA ICCOM
3610     JSR CIOV    ;GO DO IT
3620 *
3630     LDX #$00    ;IOCB 0
3640     STX ICAUX1+1
3650     LDA #$03    ;OPEN
3660     STA ICCOM
3670 * POINT TO "E:", BELOW
3680     LDA # <SCRDEV
3690     STA ICBADR
3700     LDA # >SCRDEV
3710     STA ICBADR+1
3720     LDA #$0C    ;READ/WRITE
3730     STA ICAUX1
3740     JSR CIOV    ;GO DO IT
3750 *
3760     LDA #$7F
3770     STA NMIEN   ;NORML INTRUPTS
3780 * PROTECT APAC BY MOVING THE
3790 * BASIC AND OS POINTERS UP
3800 SETMEM
3810     LDA # <ENDAPAC ;LO-BYTE
3820     STA MEMLO
3830     STA LOMEM
3840     LDA # >ENDAPAC ;HI-BYTE
3850     STA MEMLO+1
3860     STA LOMEM+1
3870     RTS
3880 SAVVBI .BYTE 0,0
3890 SCRDEV .BYTE "E:"
3900 ********************************
3910 * BASIC ENTRY:
3920 * Q=USR(BASINGT)
3930 *
3940 * THIS SUBR SETS UP THE SPECIAL
3950 * APAC SCREEN WITH ALTERNATING
3960 * LINES OF LUMINANCE AND COLOR
3970 *
3980 BASINGT
3990     PLA
4000     BEQ INITGT  ;ZERO ARGS
4010     JMP TRAP
4020 * ML ENTRY:
4030 *  (REGISTERS IGNORED)
4040 INITGT
4050     LDA #0
4060     STA SDMCTL  ;DISABLE SCR
4070     STA NMIEN   ;NO INTERRUPTS
4080     STA ROWCRS  ;CURSOR=(0,0)
4090     STA COLCRS
4100     STA COLCRS+1
4110     STA COLOR4  ;BAKGRND=BLACK
4120     STA COLSAV  ;DEFAULT COL=0
4130     STA YBYT    ;MY POINTR TO DL
4140     STA SDLSTL  ;OS POINTR TO DL
4150     TAY         ;Y=0
4160     LDA RAMTOP  ;HI-BYTE OF
4170     SEC         ;DLIST ADR=
4180     SBC #32     ;  RAMTOP-32
4190     STA YBYT+1
4200     STA SDLSTL+1
4210 * CREATE THE DISPLAY LIST
4220 *
4230 * THE APAC DISPLAY LIST IS MUCH
4240 * THE SAME AS A GRAPHICS 8 DL
4250 * EXCEPT THERE ARE DLI'S ON
4260 * EVERY LINE, INCL THE LAST
4270 * BLANK SCAN LINE AT TOP OF THE
4280 * SCREEN
4290     LDA #112    ;CMD, 8 BLNK LNS
4300 NXTBNK
4310     STA (YBYT),Y
4320     INY
4330     CPY #3
4340     BNE NXTBNK
4350     LDA #$90    ;8 BLNKS+DLI
4360     STA (YBYT),Y
4370     LDA #$CF    ;MODE 15+DLI+LMS
4380     INY         ;Y=4
4390     STA (YBYT),Y
4400     LDA #0      ;LO-BYTE, SCRN
4410     INY         ;Y=5
4420     STA (YBYT),Y
4430     STA SCREEN
4440     LDX YBYT+1  ;HI-BYTE OF SCRN
4450     INX         ;IS RAMTOP-31
4460     TXA
4470     STA SCREEN+1
4480     INY         ;Y=6
4490     STA (YBYT),Y
4500     INY         ;Y=7
4510     LDA #$8F    ;ANTIC 15+DLI
4520 NXANTLN
4530     STA (YBYT),Y
4540     INY
4550     CPY #200
4560     BNE NXANTLN
4570     LDY #102    ;102TH BYTE, DL
4580     LDA #$CF    ;MODE 15+DLI+LMS
4590     STA (YBYT),Y
4600     LDA #0      ;ADR OF 2ND HALF
4610     INY         ;OF SCRN IS:
4620     STA (YBYT),Y
4630     LDA RAMTOP  ;(RAMTOP-16)*256
4640     SEC
4650     SBC #16
4660     INY         ;Y=104
4670     STA (YBYT),Y
4680     LDY #200
4690     LDA #65
4700     STA (YBYT),Y
4710     INY         ;Y=201
4720     LDA #0      ;LO-BYTE, ADR DL
4730     STA (YBYT),Y
4740     INY         ;Y=202
4750     LDA YBYT+1  ;HI-BYTE, ADR DL
4760     STA (YBYT),Y
4770 * CLEAR OUT SCREEN MEMORY
4780 CLRSCR
4790     LDA SCREEN
4800     STA ZTEMP
4810     LDA SCREEN+1
4820     STA ZTEMP+1
4830     CLC
4840     ADC #30     ;LAST PAGE OF
4850     STA TMP1    ;MEM TO CLEAR+1
4860 SPAGE
4870     LDY #0
4880     TYA         ;A=0
4890 ZERSB
4900     STA (ZTEMP),Y
4910     DEY
4920     BNE ZERSB
4930     INC ZTEMP+1
4940     LDA ZTEMP+1
4950     CMP TMP1    ;END?
4960     BCC SPAGE   ;NOPE
4970 * INITIALIZE THE TABLE
4980 * OF ADDRESSES THAT POINT TO THE
4990 * 96 APAC SCREEN LINES
5000     LDA SCREEN+1
5010     STA ZTEMP+1 ;HI-BYTE
5020     LDY #$00
5030 NXTLN80
5040     LDA ZTEMP   ;SAVE LO
5050     STA SCRALO,Y
5060     LDA ZTEMP+1 ;SAVE HI
5070     STA SCRAHI,Y
5080     CLC
5090     LDA ZTEMP
5100     ADC #80     ;80 BYTES PER
5110     STA ZTEMP   ;APAC LINE
5120     BCC NXTY
```

```
5130      INC ZTEMP+1
5140 NXTY
5150      INY
5160      CPY #96       ;96 LINES TO DO
5170      BNE NXTLN80
5180      LDA VVBLKI    ;SAVE THE OS VBI
5190      STA SAVVBI
5200      LDA VVBLKI+1
5210      STA SAVVBI+1
5220 * POINT TO THE APAC IMM VBI
5230      LDA #6        ;STAGE 1 VBI
5240      LDX # >IVBI   ;HI-BYTE
5250      LDY # <IVBI   ;LO-BYTE
5260      JSR SETVBV    ;SET IT
5270 * POINT TO THE FIRST DLI
5280      LDA # <DLI1
5290      STA VDSLST
5300      LDA # >DLI1
5310      STA VDSLST+1
5320      LDA #$40      ;GRAPHICS 9
5330      STA GPRIOR
5340      LDA #$C0      ;ALL INTERRUPTS
5350      STA NMIEN
5360      LDA #34
5370      STA SDMCTL    ;SCREEN ON
5380 LEAVE
5390      RTS
5400 ********************************
5410 * THE RESET HANDLER:
5420 * THE JSR $FFFF WILL POINT TO
5430 * DOS AFTER LOADING APAC
5440 * (SEE SYSSET, BELOW)
5450 *
5460 WRMSTRT
5470      JSR $FFFF
5480 * POINT THE OS RESET VECTORS
5490 * TO WRMSTRT
5500 SETVEC
5510      LDA # <WRMSTRT ;LO-BYTE
5520      STA DOSINI
5530      STA CASINI
5540      LDA # >WRMSTRT ;HI-BYTE
5550      STA DOSINI+1
5560      STA CASINI+1
5570 * GO SET THE LOMEM POINTERS
5580      JMP SETMEM
5590 ********************************
5600 * MAKE SURE X<80 AND Y<96
5610 *
5620 BOUNDCK
5630      CPX #80
5640      BCC CKYPOS
5650      BCS ERRBND    ;X>79, EXIT APAC
5660 CKYPOS
5670      CPY #96
5680      BCC LEAVE     ;ALL'S WELL
5690 ERRBND
5700      PLA           ;Y>95, SO PULL
5710      PLA           ;JSR TO BOUNDCK
5720      RTS           ;EXIT APAC
5730 ********************************
5740 * PULL OFF EXCESS ARGUMENTS
5750 *
5760 TRAP
5770      TAX
5780      BEQ EXTRAP    ;NO ARGS TO PULL
5790 DELARG
5800      PLA           ;PULL HI
5810      PLA           ;PULL LO
5820      DEX
5830      BNE DELARG    ;DO MORE ARGS
5840 EXTRAP
5850      RTS
5860 ********************************
5870 * APAC'S IMM MODE VBI KEEPS
5880 * EVERYTHING TIMED RIGHT
5890 *
5900 IVBI
5910      PHA           ;SAVE ACC.
5920      LDA #$C0      ;ENABLE
5930      STA NMIEN     ;ALL INTERRUPTS
5940      LDA # <DLI1   ;POINT TO
5950      STA VDSLST    ;FIRST DLI
5960      STA COLOR4    ;BACKGRND=BLACK
5970      PLA           ;RESTORE ACC.
5980      JMP SYSVBV    ;DO OS IMM VBI
5990 ********************************
6000 * THE DLI'S TOGGLE- I.E. THEY
6010 * APPEAR EVERY OTHER LINE,
6020 * ALTERNATING GRAPHICS 9 & 11
6030 *
6040 DLI1
6050      PHA           ;SAVE ACC.
6060      LDA # <DLI2
6070      STA VDSLST    ;POINT TO DLI2
6080      LDA #$40
6090      STA WSYNC
6100      STA PRIOR     ;GRAPHICS 9
6110      PLA           ;RESTORE ACC.
6120      RTI
6130 DLI2
6140      PHA           ;SAVE ACC.
6150      LDA # <DLI1
6160      STA VDSLST    ;POINT TO DLI1
6170      LDA #$C0
6180      STA WSYNC
6190      STA PRIOR     ;GRAPHICS 11
6200      PLA           ;RESTORE ACC.
6210      RTI
6220 ********************************
6230 SCRALO *= *+96
6240 SCRAHI *= *+96
6250 ENDAPAC = *
6260 ********************************
6270 * SYSSET INITIALIZES WRMSTRT
6280 * (POINTS A JSR THERE TO DOS)
6290 * ALSO JSR'S TO SETVEC
6300 * WHICH INITS APAC'S RESET TRAP
6310 * AND SELF-PROTECTION
6320 *
6330      *=  $4000
6340 SYSSET
6350      LDA DOSINI    ;MODIFY CODE
6360      STA WRMSTRT+1 ; IN SUBR
6370      LDA DOSINI+1  ; WARMSTRT
6380      STA WRMSTRT+2
6390      JSR SETVEC
6400      LDA TRAMSZ    ;IF 1, CART IN
6410      BNE GOCART
6420      RTS           ;(NO CARTRIDGE)
6430 GOCART
6440      LDA #0
6450      STA WARMST    ;DO A WARMSTART
6460      JMP $A000     ;TO CARTRIDGE
```

Listing 6
Assembly listing

```
10 ****************************
20 * APAC KAL DEMO, V2.0      *
30 * CREATED BY THOMAS TANIDA *
40 * FIRST STARTED: 1/31/87   *
50 * LAST REVISED: 2/7/87     *
60 ****************************
70        *=  $6000
80        .OPT NO EJECT
90        .TAB 8,12,30
0100 ********************************
0110 ZTEMP = $80
0120 DLE =   $82
0130 COLOR = $84
0140 COORDS = $85
0150 DLTX =  $89
0160 OFFX =  $8D
0170 PNTPOS = $96
0180 TEMP =  $A0
0190 STORE = $B0
0200 *
0210 * SYSTEM EQUATES
0220 *
0230 KEYDEL = $02F1   ;753
0240 KEYPRS = $02FC   ;754
0250 CONSOL = $D01F   ;53279
```

```
0260 RANDOM = $D20A  ;53770
0270 *
0280 * APAC SYS EQUATES
0290 *
0300 APACPLOT = $230C
0310 APACDRAW = $230F
0320 APACEXIT = $2312
0330 APACINIT = $2315
0340 *
0350 ********************************
0360 STARTUP
0370      JSR APACINIT ;SET UP APAC
0380      LDX #39      ;INIT VARS
0390      STX OFFX     ;OFFSETS
0400      STX OFFX+1
0410      INX
0420      STX OFFX+3
0430 *
0440      LDA #$00     ;ZERO OUT
0450      LDX #$08     ;$A5-$AC
0460 INT0
0470      STA COORDS-1,X
0480      DEX
0490      BNE INT0
0500      STA COLOR    ;1ST COLOR=BLACK
0510 LOOP
0520      LDA #$FF     ;CLEAR KEYBD REG
0530      STA KEYPRS
0540      JSR POSSET   ;LINE 1
0550      JSR POSSET   ;LINE 2
0560      LDA RANDOM   ;GET A RAND #
0570      AND #$1F     ;FROM 0-31
0580      BNE GOMOV    ;31 IN 32
0590      JSR DIRSET   ;1 IN 32
0600 GOMOV
0610      JSR MAKMOV
0620      INC COLOR    ;CHANGE COLOR
0630      LDA CONSOL   ;GET CONSOLE KEY
0640      CMP #6       ;START PRESSED?
0650      BEQ STARTUP  ;YEP- CLEAR SCR
0660      LDA KEYPRS   ;GET KEY
0670      CMP #33      ;SPACE PRESSED?
0680      BEQ PAUSE    ;YEP- PAUSE
0690      CMP #28      ;ESC PRESSED?
0700      BNE LOOP     ;NO- MAIN LOOP
0710      JMP APACEXIT
0720 ********************************
0730 PAUSE
0740      LDA KEYDEL   ;WAIT FOR KEY
0750      BNE PAUSE    ;TO BE RELEASED
0760      LDA #$FF     ;CLEAR KEYBD
0770      STA KEYPRS
0780 GETSPC
0790      LDA KEYPRS   ;GET KEYCODE
0800      CMP #33      ;SPACE?
0810      BNE GETSPC   ;WAIT FOR SPACE
0820      JMP LOOP     ;BACK TO MAIN
0830 ********************************
0840 POSSET
0850      LDX #2
0860 NXP2
0870      LDA COORDS+1,X ;ROTATE X2,Y2
0880      JSR SUB256
0890      STA COORDS+1,X
0900      DEX
0910      BNE NXP2
0920      JSR PNTSCR
0930 *
0940      LDX #2
0950 NXP1
0960      LDA COORDS-1,X ;ROTATE X1,Y1
0970      JSR SUB256
0980      STA COORDS-1,X
0990      DEX
1000      BNE NXP1
1010      JMP PNTSCR
1020 ********************************
1030 SUB256
1040      EOR #$FF     ;ACC=256-ACC
1050      CLC
1060      ADC #1
1070      RTS
1080 ********************************
1090 * DIRSET CHANGES THE DIRECTION
1100 * OF THE MOVEMENT OF LINES YOU
1110 * SEE.  OTHERWISE, THE KAL WOULD
1120 * BE REPETITIVE AND BORING.
1130 *
1140 DIRSET
1150      LDX #4
1160 GETPLM
1170      LDA RANDOM   ;GET A RANDOM #
1180      AND #$03     ;FROM 0-3
1190      BEQ GETPLM   ;BUT NOT 0
1200      SEC          ;MAKE IT -1,0,1
1210      SBC #2
1220      STA DLTX-1,X ;SAVE IT
1230      DEX          ;4 TIMES
1240      BNE GETPLM
1250      RTS
1260 ********************************
1270 MAKMOV
1280      LDX #4
1290 NXTPSX
1300      LDA COORDS-1,X ;GET COORD
1310      CLC
1320      ADC DLTX-1,X ;ADD -1,0 OR 1
1330 * ESSENTIALLY THESE NEXT FEW
1340 * LINES DO A "ACC MOD 40"
1350 * BY SUBTRACTING 40 EACH TIME
1360 * ACC>=40 UNTIL ACC<40
1370 *
1380 * THE PURPOSE IS TO MAKE SURE
1390 * THE COORDS DON'T GO OUTSIDE
1400 * SCREEN LIMITS (THERE ARE 4
1410 * QUADRANTS 40 BY 40)
1420 MOD40
1430      CMP #40
1440      BCC LT40     ;LESS THAN 40
1450      SEC
1460      SBC #40      ;A=A-40
1470      BCS MOD40    ;GO BACK
1480 LT40
1490      STA COORDS-1,X ;SAVE IT BACK
1500      DEX
1510      BNE NXTPSX   ;NEXT POSITION
1520      RTS
1530 ********************************
1540 PNTSCR
1550      LDY #4
1560 NXPERM
1570      LDX OFOFF-1,Y ;ALTERNATE
1580      LDA COORDS-1,Y ;THE COORDS
1590      CLC          ;PLOTTED
1600      ADC OFFX,X   ;TO REFLECT
1610      LDX PNTOFF-1,Y ;IN EACH
1620      STA PNTPOS,X ;PART OF THE
1630      DEY          ;SCREEN
1640      BNE NXPERM
1650      JSR PUTONSCR ;DRAW ON SCR
1660 *
1670      LDY #4       ;SAME AS ABOVE
1680 NXCOMB
1690      LDX OFOFF+1,Y ;SEE END OF
1700      LDA COORDS-1,Y ;KAL FOR THE
1710      CLC          ;REARRANGEMENT
1720      ADC OFFX,X   ;OF THE COORDS
1730      LDX PNTOFF+1,Y
1740      STA PNTPOS,X
1750      DEY
1760      BNE NXCOMB
1770 * (FALLS THROUGH TO BELOW)
1780 ********************************
1790 PUTONSCR
1800      LDA COLOR    ;GET COLOR
1810      LDX PNTPOS   ;X-POS
1820      LDY PNTPOS+1 ;Y-POS
1830      JSR APACPLOT ;PLOT X1,Y1
1840      LDA COLOR    ;GET COLOR
1850      LDX PNTPOS+2 ;DEST X-POS
1860      LDY PNTPOS+3 ;DEST Y-POS
1870      JSR APACDRAW ;DRAWTO X2,Y2
1880 *
1890      LDA COLOR    ;GET COLOR
1900      LDX PNTPOS+1 ;X-POS
1910      LDY PNTPOS+2 ;Y-POS
1920      JSR APACPLOT ;PLOT Y1,X2
1930      LDA COLOR    ;GET COLOR
1940      LDX PNTPOS   ;DEST X-POS
1950      LDY PNTPOS+3 ;DEST Y-POS
1960      JMP APACDRAW ;DRAWTO X1,Y2
1970 ********************************
1980 OFOFF .BYTE 0,0,1,1,0,0
1990 PNTOFF .BYTE 0,2,1,3,0,2
```

TUTORIAL

48K Cassette or Disk

Dealin' Demo

A machine language subroutine to create playing card graphics

by Eric Huffman

One day, while looking for an idea for a programming project, I came across a book which gave me not one idea but hundreds! The book, *The Official Rulebook of Playing Card Games*, was a veritable jackpot. I love card games and most of them can be easily coded in BASIC.

However, I hit a snag almost immediately. The play algorithms were rapidly done in BASIC, but the card graphics were painfully slow. "What I need," I said to myself, "is a machine language subroutine to handle the playing card graphics." I pulled out my Atari Macro Assembler.

About the program

The subroutine is stored in a string and called with the USR function. The routine will instantly place a card anywhere on a Graphics 8 screen. The card placed can be placed either face up or face down, or changed to the background color (for erasing purposes).

Dealin' Demo, found in Listing 1, will demo the subroutine in three ways. The first demo places cards randomly about the screen. The second demo fans the cards horizontally and shows the different spacings available and, finally, the third demo fans the deck in an arc, one suit at a time.

In any case, the routine is called with a:

J=USR(CS,Y,X,SIDE,SUIT,V)

Where:

CS	=	Subroutine start address
Y	=	Vertical position (0-160)
X	=	Horizontal position (0-37)
SIDE	=	Back, front or blank (0, 1 or 2)
SUIT	=	Heart, Club, Diamond or Spade (1, 2, 3 or 4)
V	=	Value (ASC("2-A"))

The program gets its image data from the Atari ROM, starting at location 57344. It does, however, vector through the Character Base Register (756). This will allow the user to load an alternate character set and use this set for his playing cards. (Anybody for Greek playing cards? Better yet, how about *Dungeons and Dragons* playing cards?)

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

Listing 1
BASIC listing

```
VI 10 REM PROGRAM: CARD.BAS
KU 11 REM A CARD DEALING SUBROUTINE.
VQ 12 REM ALLOWS A CARD TO BE DEALT
OY 13 REM ANYWHERE ON THE SCREEN.
BG 14 REM
WY 15 REM COPYRIGHT 1988
YD 16 REM BY ANALOG COMPUTING
BM 17 REM
PN 20 REM ALTER DLIST FOR TITLE SCREEN
IW 30 CLR :GRAPHICS 8:POKE 710,0:DLOC=PEE
   K(560)+PEEK(561)*256
CT 35 FOR I=60 TO 98:POKE DLOC+I,0:NEXT I
FY 40 POKE DLOC+4,0:POKE DLOC+5,7:POKE DL
   OC+101,6:POKE DLOC+100,0:POKE DLOC+102
   ,7
PV 45 FOR I=0 TO 2:POKE DLOC+103+I,2:POKE
    DLOC+I,0:NEXT I
NU 48 DIM TITLE$(12):TITLE$="DEALIN' DEMO
   "
ZG 50 FOR I=1 TO 12:POKE I+1579,ASC(TITLE
   $(I,I))-32:NEXT I
ZI 55 ? "       A CARD DEALING SUBROUTINE"
   :? "          BY ERIC HUFFMAN"
MS 90 GOSUB 2000:REM FOR SUBROUTINE SETUP
KY 270 REM RANDOM CARD DEMO
KW 280 FOR I=1 TO 100:J=USR(CS,Y,X,SIDE,S
   UIT,V)
FS 290 X=PEEK(53770)/7:Y=PEEK(53770)/1.56
   :SIDE=INT(RND(0)+0.5):SUIT=INT(RND(0)*
   4+1)
KW 292 V=ASC(NUM$(INT(RND(0)*13+1))):NEXT
    I
NU 320 REM LAYOUT 1 DEMO
```

```
QJ 324 GRAPHICS 8+16:POKE 709,15:POKE 710
   ,96:POKE 712,4
IT 325 FOR W=1 TO 3:FOR SIDE=0 TO 2
RM 330 FOR SUIT=1 TO 4:Y=SUIT*35-32:FOR N
   =1 TO 13
NC 331 V=ASC(NUM$(N)):X=N*W-W:Y=Y+2:J=USR
   (CS,Y,X,SIDE,SUIT,V):NEXT N:NEXT SUIT
PW 335 NEXT SIDE:NEXT W
IO 340 REM LAYOUT 2 DEMO
ZC 343 SIDE=1
IO 345 FOR SUIT=1 TO 4
DC 350 FOR N=1 TO 13:V=ASC(NUM$(N)):X=N*2
   +3+ABS(SUIT-2):Y=120-SQR(10000-(X*5-92
   )^2)+SUIT*20:J=USR(CS,Y,X,SIDE,SUIT,V)
HU 360 NEXT N:NEXT SUIT
EC 370 FOR I=0 TO 1000:NEXT I:END
UU 999 REM DATA FOR SUBROUTINE
OR 1000 DATA 104,169,0,133,77,165,88,133,
   203,165,89,133,204,104
QO 1010 DATA 104,240,17,168,24,165,203,10
   5,40,133,203,165,204,105,0,133,204,136
   ,208,240
LK 1020 DATA 24,104,104,101,203,133,203,1
   65,204,105,0,133,204,104,104,208,117,1
   04,104,104
PU 1030 DATA 104,162,16,160,3,169,50,145,
   203,136,169,51,145,203,136,169,179,145
   ,203,165
HQ 1040 DATA 203,24,105,80,133,203,165,20
   4,105,0,133,204,202,208,224,160,3,169,
   170,145
CC 1050 DATA 203,136,208,249,165,203,56,2
   33,40,133,203,165,204,233,0,133,204,16
   2,16,160
LB 1060 DATA 3,169,206,145,203,136,169,20
   4,145,203,136,169,140,145,203,165,203,
   56,233,80
PY 1070 DATA 133,203,165,204,233,0,133,20
   4,202,208,224,165,203,24,105,40,133,20
   3,165,204
VX 1080 DATA 105,0,133,204,160,3,169,170,
   145,203,136,208,249,96,56,233,1,240,32
   ,162
IB 1090 DATA 33,160,3,169,0,145,203,136,2
   08,249,165,203,24,105,40,133,203,165,2
   04,105
GQ 1100 DATA 0,133,204,202,208,231,104,10
   4,104,104,96,162,32,160,3,169,85,145,2
   03,136
WO 1110 DATA 208,249,165,203,24,105,40,13
   3,203,165,204,105,0,133,204,202,160,3,
   169,253
ME 1120 DATA 145,203,136,169,255,145,203,
   136,169,127,145,203,165,203,24,105,40,
   133
SC 1130 DATA 203,165,204,105,0,133,204,20
   2,208,224,160,3,169,85,145,203,136,208
KC 1140 DATA 249,169,0,133,206,133,208,13
   3,209,104,104,72,201,2,240,8,201,4,240
   ,4
CX 1150 DATA 169,85,133,209,104,56,233,1,
   10,10,10,10,201,48,208,3,24,105,11,24
FF 1160 DATA 105,64,24,10,38,206,10,38,20
   6,10,38,206,133,205,24,165,206,109,244
   ,2
WE 1170 DATA 133,206,104,104,56,233,32,24
   ,10,38,208,10,38,208,10,38,208,133,207
   ,24
EG 1180 DATA 165,208,109,244,2,133,208,56
   ,165,203,233,37,133,203,165,204,233,0,
   133,204
YE 1190 DATA 162,0,160,7,56,169,254,241,2
   07,10,24,105,1,5,209,129,203,56,165,20
   3
UJ 1200 DATA 233,40,133,203,165,204,233,0
   ,133,204,136,16,227,160,7,56,169,255,2
   41,205
CD 1210 DATA 10,10,24,105,1,5,209,129,203
   ,56,165,203,233,40,133,203,165,204,233
   ,0
PP 1220 DATA 133,204,136,16,226,24,165,20
   3,105,38,133,203,165,204,105,0,133,204
   ,160,7
EL 1230 DATA 56,169,127,241,205,5,209,129
   ,203,56,165,203,233,40,133,203,165,204
   ,233,0
SP 1240 DATA 133,204,136,16,231,160,7,56,
   169,255,241,207,74,5,209,129,203,56,16
   5,203
MI 1250 DATA 233,40,133,203,165,204,233,0
   ,133,204,136,16,230,96
BY 2000 REM SUBROUTINE SETUP
TU 2010 DIM CARD$(504),NUM$(13):CARD$=" "
   :CARD$(504)=CARD$:CARD$(2)=CARD$
CX 2020 CS=ADR(CARD$)
PB 2030 RESTORE 1000:FOR I=0 TO 503:READ
   D:POKE CS+I,D:NEXT I
IX 2040 REM J=USR(CS,Y,X,SIDE,SUIT,V)
ES 2050 REM CS=START OF SUBROUTINE
VI 2060 REM Y=VERTICAL POS. (0-160)
JH 2070 REM X=HOR. POS. (0-37)
UN 2080 REM SIDE=BACK,FRONT,BLANK(0,1,2)
QL 2090 REM SUIT=♥♣♦♠ (1,2,3,4)
EE 2100 REM V=VALUE (ASC("2-A"))
UN 2110 NUM$="23456789TJQKA"
XP 2120 GRAPHICS 8+16
NV 2130 POKE 709,15:POKE 710,96:POKE 712,
   4
AO 2140 RETURN
```

Listing 2
Assembly listing

```
0010 ;**********************************
0020 ; PROGRAM: CARD.SOR
0030 ;
0040 ; A RELOCATABLE ASSEMBLY LANGUAGE
0050 ; SUBROUTINE TO PLACE A CARD
0060 ; ANYWHERE ON A GR.8 SCREEN.  CARDS
0070 ; WILL BE 32 SCAN LINES HIGH AND 3
0080 ; BYTES WIDE
0090 ;
0100 ; BY: ERIC HUFFMAN
0110 ;     4330 NEW BEDFORD DRIVE
0120 ;     FT. COLLINS, CO 80525
0130 ;
0140 ;OPTIONS: FACE UP,BACK UP,& ERASE
0150 ;
0160 ;**********************************
0170 ; CARD TO SCREEN SUBROUTINE
0180 ;**********************************
0190 ;
0200 UL   EQU  203    ;UPPER LEFT POS
0210 SU   EQU  205    ;SUIT OF CARD
0220 VAL  EQU  207    ;VALUE OF CARD
0230 COL  EQU  209    ;COLOR OF CARD
0240 ;
0250      PLA         ;PULL ARG CNT
0260 ;
0270      LDA #0
0280      STA 77      ;RESET ATTRACT
0290 ;
0300      LDA 88      ;STORE SCRN UPPER
0310 ;                 CORNER IN UL
0320      STA UL
0330      LDA 89
0340      STA UL+1
0350 ;
0360 ;**********************************
0370 ; VERTICAL POSITION ADJUSTMENT
0380      PLA         ;PULL Y HIGH BYTE
0390 ;                ;DISCARD(=0)
0400      PLA         ;PULL Y
0410      BEQ DONEY   ;IF 0 SKIP
0420      ;           VERT ADJ
0430      TAY
0440 ;
0450 ; ADD 40 BYTES FOR EACH Y
0460 LOOPY CLC
0470      LDA UL
0480      ADC #40
0490      STA UL
0500      LDA UL+1
0510      ADC #0
0520      STA UL+1
0530      DEY
0540      BNE LOOPY   ; BUMP 40 UNTIL
0550      ;           Y=0
0560 ;
0570 ; HORIZONTAL POSITION ADJUSTMENT
0580 DONEY CLC
0590      PLA         ;PULL X HIGH BYTE
```

Dealin' Demo *continued*

```
0600 ;                  DISCARD(=0)
0610     PLA            ;PULL X LOW(0-37)
0620     ADC UL
0630     STA UL
0640     LDA UL+1
0650     ADC #0
0660     STA UL+1       ;X OFFSET ADDED
0670 ;
0680 ;**********************************
0690 ;
0700 ; DETERMINE OPTION (BACK,FRONT OR
0710 ; BLANK (0,1 OR 2)
0720     PLA            ;PULL SIDE HIGH,
0730 ;                   DISCARD (=0)
0740     PLA            ;PULL SIDE LOW
0750     BNE FB         ;IF NOT 0, SKIP
0760 ;                   TO FRONT/BLANK
0770 ;
0780 ;**********************************
0790 ;
0800 ; IF 0 THEN DISCARD UNUSED SUIT &
0810 ; VALUE AND DRAW BACK OF CARD
0820     PLA            ;PULL SUIT HIGH
0830     PLA            ;PULL SUIT
0840     PLA            ;PULL VALUE HIGH
0850     PLA            ;PULL VALUE
0860 ;
0870 ; PAINT BACK OF CARD
0880 ; FOR EASE OF ADDRESSING THE X
0890 ; REG HOLDS THE VERTICAL POSITION
0900 ; AND THE Y REG HOLDS THE HOR.
0910 ; (ONLY Y HAS INDEXED INDIRECT)
0920 ;
0930     LDX #16        ;16 PAIRS OF
0940     ;              SCAN LINE
0950 LX  LDY #3         ;3 BYTES WIDE
0960 ; BACK PATTERN IS 179,51,50 EVERY
0970 ; OTHER LINE GOING DOWN
0980     LDA #50
0990     STA (UL),Y
1000     DEY
1010     LDA #51
1020     STA (UL),Y
1030     DEY
1040     LDA #179
1050     STA (UL),Y
1060     LDA UL
1070     CLC
1080     ADC #80
1090     STA UL
1100     LDA UL+1
1110     ADC #0
1120     STA UL+1
1130     DEX
1140     BNE LX
1150 ; DRAW BOTTOM LINE
1160     LDY #3
1170 BT  LDA #170
1180     STA (UL),Y
1190     DEY
1200     BNE BT
1210 ; DRAW BACK PATTERN 140,204,206
1220 ; EVERY OTHER LINE GOING UP
1230     LDA UL
1240     SEC
1250     SBC #40
1260     STA UL
1270     LDA UL+1
1280     SBC #0
1290     STA UL+1
1300     LDX #16
1310 LX2 LDY #3
1320     LDA #206
1330     STA (UL),Y
1340     DEY
1350     LDA #204
1360     STA (UL),Y
1370     DEY
1380     LDA #140
1390     STA (UL),Y
1400     LDA UL
1410     SEC
1420     SBC #80
1430     STA UL
1440     LDA UL+1
1450     SBC #0
1460     STA UL+1
1470     DEX
1480     BNE LX2
1490     LDA UL
1500     CLC
1510     ADC #40
1520     STA UL
1530     LDA UL+1
1540     ADC #0
1550     STA UL+1
1560 ; DRAW TOP LINE
1570     LDY #3
1580 TP  LDA #170
1590     STA (UL),Y
1600     DEY
1610     BNE TP
1620     RTS            ; BACK DONE
1630 ;
1640 ;**********************************
1650 ;
1660 ; PAINT FRONT/BLANK
1670 ;
1680 FB  SEC
1690     SBC #1
1700     BEQ FRONT
1710 ;
1720 ;**********************************
1730 ;
1740 ; PAINT BLANK
1750 ; OK, OPTION MUST BE BLANK(ERASE)
1760 ; SO DRAW BACKGROUND COLOR
1770     LDX #33
1780 B2  LDY #3
1790 B1  LDA #0
1800     STA (UL),Y
1810     DEY
1820     BNE B1
1830     LDA UL
1840     CLC
1850     ADC #40
1860     STA UL
1870     LDA UL+1
1880     ADC #0
1890     STA UL+1
1900     DEX
1910     BNE B2
1920 ;PULL SUIT & VALUE, DISCARD
1930     PLA            ;PULL SUIT
1940     PLA
1950     PLA            ;PULL VALUE
1960     PLA
1970     RTS            ;ALL DONE WITH BL
ANK
1980 ;
1990 ;**********************************
2000 ;
2010 ; THIS SECTION PAINTS THE FRONT
2020 ; OF A CARD.
2030 ;
2040 FRONT LDX #32
2050     LDY #3
2060 ; DRAW TOP LINE
2070 TF  LDA #85
2080     STA (UL),Y
2090     DEY
2100     BNE TF
2110     LDA UL
2120     CLC
2130     ADC #40
2140     STA UL
2150     LDA UL+1
2160     ADC #0
2170     STA UL+1
2180     DEX
2190 ; DRAW WHITE FACE AND BORDER
2200 LX3 LDY #3
2210     LDA #253
2220     STA (UL),Y
2230     DEY
2240     LDA #255
```

```
2250      STA (UL),Y
2260      DEY
2270      LDA #127
2280      STA (UL),Y
2290      LDA UL
2300      CLC
2310      ADC #40
2320      STA UL
2330      LDA UL+1
2340      ADC #0
2350      STA UL+1
2360      DEX
2370      BNE LX3
2380 ; DRAW BOTTOM LINE
2390      LDY #3
2400 BF   LDA #85
2410      STA (UL),Y
2420      DEY
2430      BNE BF
2440 ; ZERO OUT SUIT, VALUE & COLOR
2450      LDA #0
2460      STA SU+1
2470      STA VAL+1
2480      STA COL
2490      PLA           ;PULL SUIT HIGH
2500      PLA           ;PULL SUIT (1-4)
2510 ; PUSH TO STACK FOR USE LATER
2520      PHA
2530 ; SET COLOR MASK - 0 FOR BLACK
2540 ;                  85 FOR RED
2550      CMP #2
2560      BEQ BLK
2570      CMP #4
2580      BEQ BLK
2590      LDA #85
2600      STA COL
2610 BLK  PLA           ;PULL SUIT AGAIN!
2620 ; FIND SUIT IN MEMORY
2630      SEC
2640      SBC #1
2650      ASL A
2660      ASL A
2670      ASL A
2680      ASL A
2690      CMP #48
2700      BNE SK1
2710      CLC
2720      ADC #11
2730 SK1  CLC
2740      ADC #64
2750      CLC
2760 ; NEXT MUST MULTIPLY BY 8
2770      ASL A
2780      ROL SU+1
2790      ASL A
2800      ROL SU+1
2810      ASL A
2820      ROL SU+1
2830      STA SU
2840      CLC
2850      LDA SU+1
2860      ADC 756       ;ADD CHAR. SET
2870      ;             START
2880      STA SU+1      ;SUIT MEM POS
2890      ;             STORED
2900 ; START ON VALUE
2910      PLA           ;VALUE HIGH
2920      PLA           ;VALUE LOW
2930      SEC
2940      SBC #32
2950      CLC           ;NEXT MUST X8
2960      ASL A
2970      ROL VAL+1
2980      ASL A
2990      ROL VAL+1
3000      ASL A
3010      ROL VAL+1
3020      STA VAL
3030      CLC
3040      LDA VAL+1
3050      ADC 756
3060      STA VAL+1     ;VALUE MEM.STORED
3070 ;
3080 ; * PUT EM ON THE SCREEN *
3090 ;
3100 ; START AT BOTTOM, WORK UP
3110      SEC
3120      LDA UL
3130      SBC #37
3140      STA UL
3150      LDA UL+1
3160      SBC #0
3170      STA UL+1
3180      LDX #0
3190 ;FIRST, THE VALUE ON RIGHT SIDE
3200      LDY #7
3210 VR   SEC
3220      LDA #254      ;CREATE INVERSE
3230      SBC (VAL),Y
3240      ASL A         ;FOR CLARITY
3250      CLC
3260      ADC #1
3270      ORA COL       ;COLOR MASK
3280      STA (UL,X)
3290      SEC
3300      LDA UL
3310      SBC #40
3320      STA UL
3330      LDA UL+1
3340      SBC #0
3350      STA UL+1
3360      DEY
3370      BPL VR
3380 ;SECOND, THE SUIT ON RIGHT SIDE
3390      LDY #7
3400 SR   SEC
3410      LDA #255      ;CREATE INVERSE
3420      SBC (SU),Y
3430      ASL A         ;FOR CLARITY
3440      ASL A         ;FOR CLARITY
3450      CLC
3460      ADC #1
3470      ORA COL       ;COLOR MASK
3480      STA (UL,X)
3490      SEC
3500      LDA UL
3510      SBC #40
3520      STA UL
3530      LDA UL+1
3540      SBC #0
3550      STA UL+1
3560      DEY
3570      BPL SR
3580      CLC
3590      LDA UL
3600      ADC #38
3610      STA UL
3620      LDA UL+1
3630      ADC #0
3640      STA UL+1
3650 ;THIRD, THE SUIT ON THE LEFT SIDE
3660      LDY #7
3670 SL   SEC
3680      LDA #127      ;CREATE INVERSE
3690      SBC (SU),Y
3700      ORA COL       ;COLOR MASK
3710      STA (UL,X)
3720      SEC
3730      LDA UL
3740      SBC #40
3750      STA UL
3760      LDA UL+1
3770      SBC #0
3780      STA UL+1
3790      DEY
3800      BPL SL
3810 ;LAST, THE VALUE ON THE LEFT SIDE
3820      LDY #7
3830 VL   SEC
3840      LDA #255      ;CREATE INVERSE
3850      SBC (VAL),Y
3860      LSR A         ;FOR CLARITY
3870      ORA COL       ;COLOR MASK
3880      STA (UL,X)
3890      SEC
3900      LDA UL
3910      SBC #40
3920      STA UL
3930      LDA UL+1
3940      SBC #0
3950      STA UL+1
3960      DEY
3970      BPL VL
3980      RTS           ;FRONT DONE ●
```

REGULAR FEATURE

Database Delphi

News and updates from the *ANALOG Computing* Atari Users' Group on Delphi

by Michael A. Banks (KZIN)

As usual, there's lots happening on Delphi in general, and in the Atari Users' Group in particular. In case you haven't logged on in a while, or might have missed something, we'll try to get you caught up on things here. And if you're not on-line. . .well, you really don't know what you're missing, so we're going to show a bit of it to you in a few lines.

First, let's take a a look at some recent enhancements to Delphi.

Downloading and uploading: Ymodem and a simplified database download menu

In addition to ASCII file transfer, Xmodem, windowed Xmodem, and Kermit file transfer protocols, Delphi now supports the increasingly popular Ymodem protocol for uploads and downloads. Ymodem is an extended version of Xmodem, using 1024-byte (1K) blocks, rather than Xmodem's 128-byte blocks—which adds noticeable speed to uploads and downloads!

To use Ymodem in the database areas, type *YM* at the *ACTION>* prompt—after you've set your terminal software to receive Ymodem, of course. (Notice to "Flash" terminal program users: Flash calls Ymodem "1 K-byte Xmodem.")

Alternatively, you can simply type *DOWN* (for "Download") at a database *ACTION>* prompt, and select Ymodem transfer from the menu that is displayed. This menu is a handy reminder of the various download protocols and the commands used to invoke them.

To use Ymodem in your personal workspace, simply type *YUP* or *YDO* (for "Ymodem UPload" or "Ymodem DOwnload," respectively) at the *WS>* prompt.

Database search enhancement

Delphi has streamlined its database search procedures. To find items meeting your criteria in any database in the Atari Users' Group, simply type *SEARCH* at a database name prompt, then enter a keyword. All items that don't match the keyword will be rendered temporarily "invisible" to you. Delphi will tell you how many items match your keyword, and you'll be presented with a temporary subset of the database that you can access with the usual database commands.

If the keyword you enter returns too many items, you can narrow the search by typing *NARROW*. This will prompt you for a keyword. After you enter it, Delphi will reduce the number of files "found" to include only those items that have both keywords.

Too few items? Use EXPAND to include all items that have either the first keyword or the new keyword. The NARROW and EXPAND commands can be used to specify virtually any number of keywords.

A quick look at Delphi groups and clubs—and The Atari Users' Group!

And now, for those of you who have been wondering just what Delphi looks like, here's a small sample—including the Atari Users' Group, the on-line service for readers of **ANALOG Computing** and **ST-Log**.

First, let's take a look at Delphi's "Groups and Clubs" menu (noting, of course, the number one position of our favorite group!):

Atari SIG	Music City
Antiques/Collectibles	OS9 On-Line
Apple][& ///	PC Compatibles/IBM SIG
Aviation SIG	Portable Place
Business Forum	Science Fiction Sig
Close Encounters	Starship Amiga
Color Computer	Tandy PC SIG
Commodore SIGs	TI Information Network
C*SIX	Theological Network
GameSig	Wang Users SIG
Macintosh ICONtact	Writers Group
Micro Artists (MANIAC)	HELP
Model Builders	EXIT

As you can see, there is a group or club for virtually any interest—computing or non-computing—on Delphi.

Now, let's take an all-too-brief look at the Atari Users' Group. (Type *ATARI* at the "Groups and Clubs" menu to get in.)

When you first enter the Atari area, you'll see **ANALOG**'s logo and banner. After the logo and banner and any new announcements, you'll see this menu:

ANALOG's ATARI SIG Menu:

Announcements	Request Free Upload
Conference	Set Preferences
Databases	Topic Descriptions
Entry Log	Who's Here
Forum (Messages)	Workspace
MAIL (Electronic)	Help
Member Directory	Exit
Poll	

ANALOG>What do you want to do?

The Atari menu is your gateway to many on-line activities, including real-time conference, databases packed with informative articles and useful programs, the "Forum" (a bulletin board where you can post questions and join in discussions on a wide variety of topics), and much more.

You may well be wondering what sort of information and programs are available in the databases. Here's a list of topics to whet your appetite:

Databases Available Menu:

General Interests	Reviews & News
Games & Entertainment	ST Programs
Telecommunications	Koala Pictures
Utilities	DEGAS Pictures
Toolbox for the ST	Current Issue
Sight & Sound	Home use
Education	Applications for the ST
Electronics & Science	

TOPIC>Which topic?

And—what the heck—here's a peek at a very small portion of the list of files available for download in just one of the database topics, "Games & Entertainment":

(Okay, so you're not on Delphi yet. That's alright: We've reserved a spot for you. Check elsewhere in these pages for information on a special Delphi signup offer.)

Atari database updates

Speaking of databases (and we were, a few lines back), fans of the Electronic MAG will be happy to know that MATRAT (Matthew J.W. Ratcliff) has finally found enough time to put together the fifth issue. Of special interest in this issue are bizarre techniques for crashing the ST. According to MATRAT, this information is important, "So we know how they

occur, and therefore, how to avoid them."

In the same issue, you find reviews of some classic 8-bit Atari cartridges by MATRAT, a report from 1BLAKE (Blake Arnold) on the Navarone Clock card for the ST, and more.

Files from the September and October, 1987 issues of **ST-Log** are in the "Current Issue" section of the database area. Included are Darek Mihocka's "ST Xformer" (this one turns your ST into an Atari 800), and the monochrome version of "Floyd the Droid on the Run." Eight-bitters will find files from the same issues of **ANALOG Computing** in the same section of the database.

Of special interest in the "Reviews and News" database is the verbatim text of ANALOG Publishing's Summer CES interview with Atari executives Neil Harris, Sam Tramiel and Jerry Brown.

C programmers and graphics fans will also find some special gems in the databases.

(Note: The programs presented in both **ANALOG Computing** and **ST-Log** magazines are posted in the relevant databases on or after the first of the month of each issue's cover date. While we permit free distribution of programs presented in both magazines, the text of program documentation and articles published in both magazines is copyrighted material and is definitely *not* public domain. This text may not be reproduced in any form [including distribution through BBSs or other telecommunication services] without written permission from the publishers of **ANALOG Computing** and **ST-Log.**)

Forum update: hot topics and yet another enhancement

Current hot topics of discussion in the Forum include OS-9 68K (and we couldn't help but notice a few pointed barbs regarding COCO OS-9 users), and ST pricing. Too, MATRAT managed to stir up more than a little interest with a message that brought to everyone's attention the fact that coverage of the ST and Amiga have been dropped from *Byte*—perhaps because they aren't drawing the advertising dollars that other products bring in, or because *Byte* readers have an IBM and Mac fixation. Who knows? Check into the Forum and add your comments!

Yet another useful command has been added to the retinue of Forum options. The command is TAG, and it's used to mark messages that you wish to view a second time for reference or reply. TAG works like this: Each time you read a message that you know you'll want to see again, type *TAG*. The message is marked, and will be displayed (along with any other marked messages) when you type *READ TAGGED*. If you tag messages and forget about them, Delphi will remind you when you leave Forum that there are tagged messages waiting, and give you the option of then reading them.

And why would you want to tag messages? Well, tagging messages is especially useful in selecting messages for download. Rather than opening your capture buffer and catching everything (and then having to sort through a few Ks worth of characters off-line), use TAG to mark messages of interest. After you've finished checking all of the new messages in the Forum, open your buffer, type *READ TAGGED*, and get only the messages you want.

TAG is also handy for marking messages for reply. Mark messages to which you wish to reply as you go through new messages, then go back with READ TAGGED to display those messages. This can save time and keep you from entering dumb messages. You won't have to stop to note message numbers, and by the time you enter your replies, you will have read everything relevant to the messages in question.

That's it for this month. We'll be back next issue with more news and tips. Until then, see you on-line!

```
DBASES:Gam> DIRECTORY

ROTO-WRENCH                    PROG    24-AUG   ANALOG2
LIFE IN THE FAST LANE          PROG    10-JUL   ANALOG2
DRAGONLORD DUNGEON EDITOR      PROG    10-JUL   ANALOG2
ROCKS!                         PROG     9-JUL   ANALOG2
AUSTRALIAN XAGON               PROG    17-JUN   ATARICOP
ANIMATED GRAPHIC STORYBOOK     PROG    16-JUN   ATARICOP
WALNUM'S VINTAGE ADVENTURES    PROG    15-JUN   ANALOG4
TV BINGO TRACKER               PROG    12-JUN   FREDBUSH
DRILLING FOR OIL               PROG    10-JUN   ATARICOP
CAVELORD FROM GERMANY          PROG    10-JUN   ATARICOP
FLOYD THE DROID GOES BLASTIN'  PROG     7-JUN   ANALOG2
FRENCH FORTUNE-WHEEL           PROG    24-MAY   NORMLEV
MIDAS MAZE                     PROG     1-MAY   ANALOG2
RAMBUG II                      PROG     1-MAY   ANALOG2
DEVIL'S DOORWAY                PROG     1-MAY   ANALOG2
LEATHER GOD.SOL                TEXT    21-APR   JEC
DECRYPTO                       PROG     7-APR   MENTAT
SLITHER                        PROG     6-MAR   ANALOG2
STARLANES                      PROG     6-MAR   ANALOG2
WYZLE                          PROG    25-FEB   ANALOG2
T.R.I.A.D.                     PROG     2-JAN   PHOEBUS
CHECKERS                       PROG    29-DEC   PHOEBUS
KRAZY KATTERPILLARS            PROG    24-DEC   ANALOG2
ZEVIOUS                        PROG    24-DEC   ATARICOP
SNOWBALL                       PROG    22-DEC   AJQ
AMAZING MAZE                   PROG    10-DEC   ANALOG2
FORTUNE-WHEEL                  PROG    10-DEC   ANALOG2
ESP TEST                       PROG    25-NOV   ATARICOP
COSMIC GLOB                    PROG    17-NOV   ANALOG2
MODEM CHESS                    PROG    17-NOV   ANALOG2
CASTEL QUEST                   PROG    29-OCT   ATARICOP
TOOTH PASTE                    DATA    29-OCT   ATARICOP
COMPUTER TESTS                 PROG    29-OCT   ATARICOP
THE GAME OF RATS               PROG    28-OCT   ANALOG2
DEATHZONE                      PROG    17-OCT   ANALOG2
SMACK                          PROG     4-OCT   RCURZON
WHEEL OF FORTUNE II            PROG    25-SEP   TUCKER
LAUNCH CODE                    PROG    15-SEP   ANALOG2
MOONLORD                       PROG    15-SEP   ANALOG2
COMMODORE KILL                 DATA     8-SEP   ATARICOP
WHEEL OF FORTUNE               PROG     5-SEP   ATARICOP
SOLAR FLARES. . .              TEXT     5-SEP   THUD
FUN WITH MACHINES              TEXT     5-SEP   THUD
More?No
```

Michael A. Banks is the author/designer of Gateway, a text and graphics adventure for the ST published by Pryority Software and Action Software.

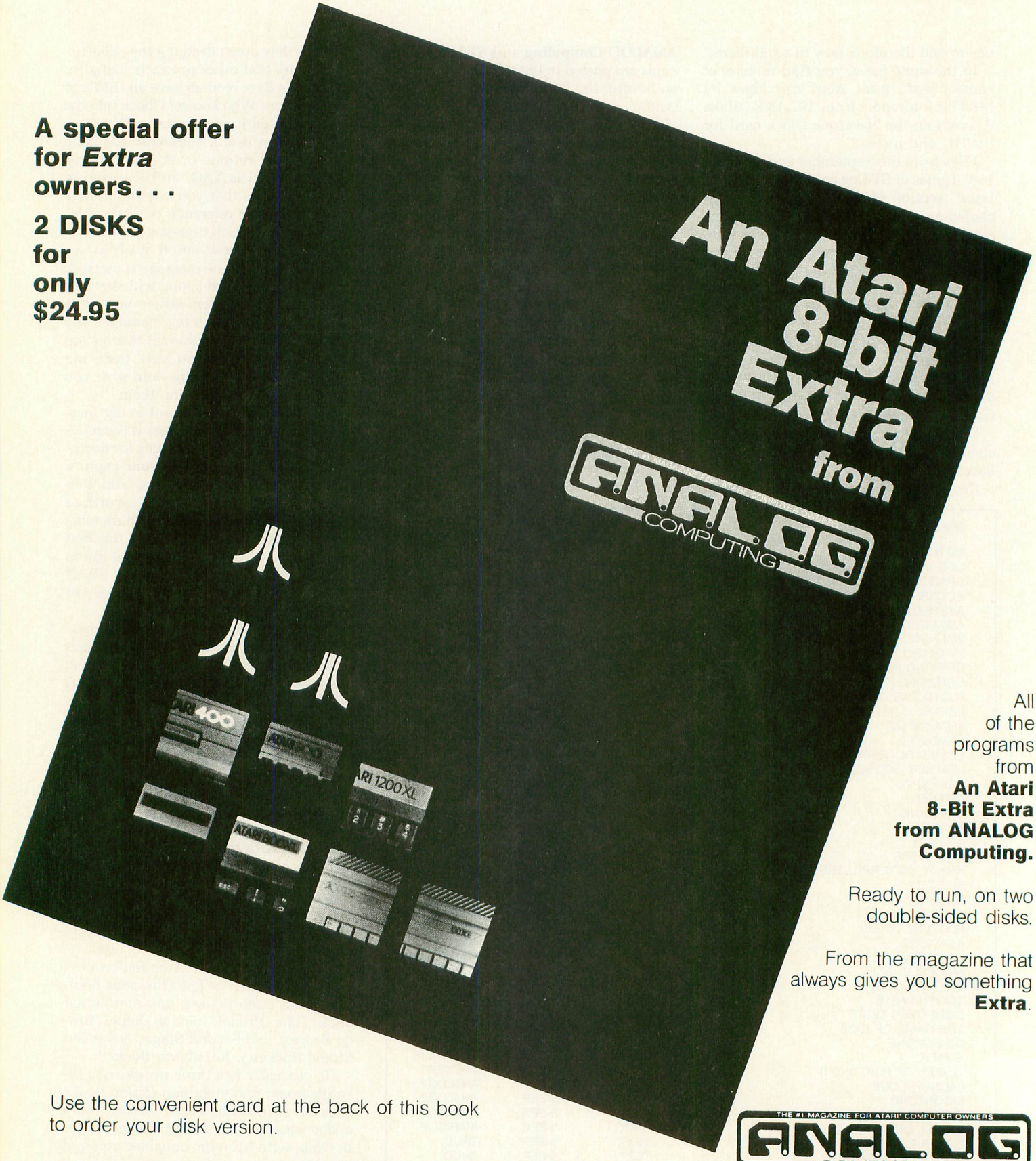
Banks also writes science fiction novels (among which is The Odysseus Solution, *from Baen Books), and non-fiction books (*The Official Guide to Delphi, *Brady Brooks, and* Second Stage: Advanced Model Rocketry, *Kalmbach Books).*

He currently has three novels, one juvenile book, and eight non-fiction books in print. A full-time writer for four years and a computer user for six, Banks resides in Ohio, with his wife, daughter, son and no cats.

Get the *Extra* on disk!

A special offer for *Extra* owners. . .

2 DISKS for only $24.95



All of the programs from **An Atari 8-Bit Extra from ANALOG Computing.**

Ready to run, on two double-sided disks.

From the magazine that always gives you something **Extra**.

Use the convenient card at the back of this book to order your disk version.

THE #1 MAGAZINE FOR ATARI® COMPUTER OWNERS
ANALOG COMPUTING

Send for it now!

Bits & Pieces

Leftover Zucchini — part 2

by Lee S. Brilliant, M.D.

Last time we began a long journey into the inner workings of the Atari SIO by looking at the interrupt system, and we hinted at finding other uses for your old 400/800 computers. This month we'll continue our trek by delving into the inner workings of POKEY, and seeing just what it does in terms of the serial port. So let's get started.

POKEY is a large-scale IC which combines several functions into one package. It has 16 registers in the address block $D200 to $D20F, and performs sound generation, paddle inputs, Serial I/O, keyboard scanning, random number generation, and IRQ interrupts. Two installments ago ("Sights and sounds"), we discussed the first nine registers, so we won't repeat this information, but we will refer to Figure 1 regarding the last seven registers.

Register	Address	Name	Read	Name	Write
9	D209	KBCODE	Raw keyboard code when moved to 764	STIMER	Starts POKEY timers when audio channels are timers
10	D20A	RANDOM	Random # 0-255	SKREST	Reset serial port bits 5-7 @ SKSTAT
11	D20B	———	NOT USED	POTGO	Start paddle read cycle
12	D20C		NOT USED		NOT USED
13	D20D	SERIN	Data read from serial port	SEROUT	Data write to serial port
14	D20E	IRQST	IRQ interrupt status	IRQEN	IRQ interrupt enable
15	D20F	SKSTAT	Serial port error status register	SKCTL	Serial port, pot & keyboard control register

Figure 1 — POKEY registers

Of these last seven registers, numbers 10, 13, 14 and 15 are of importance to Serial I/O.

One at a time

Data is sent serially by placing a byte into an 8-bit shift register, then applying clock pulses. A shift register is a line of 1-bit registers which "shift" their contents one position to the right with each clock pulse. The leftover bit is forced onto the output line with the wire assuming the logic state of the shifted bit. After eight clocks, the register is empty and the whole byte has been sent. POKEY's shift register output goes to pin 5 on the serial plug. Actually, the Atari uses a 10-bit register with the first (start) bit permanently set to 0; the last (stop) bit set to 1; and the data byte sandwiched in between.

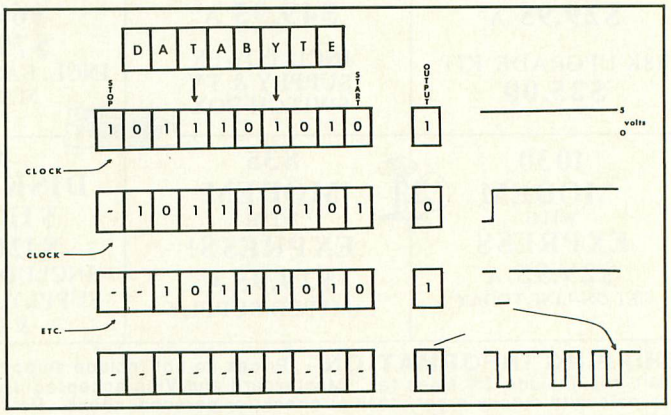


Figure 2 — Shift register

The similarity to the ASL, LSR, ROR and ROL commands is obvious.

The shift register is located within POKEY, so it makes sense that clock pulses are generated by the sound channels. While this provides flexibility in baud rates—and saves hardware—it prevents sound production and I/O from occurring at the same time. Channels 3 and 4 are arranged in 16-bit format and clocked at 1.79 MHz to yield a rate of about 19,200 baud or bits per second. The same frequency also shows up on the Clock Out pin of the serial plug and could be used for synchronous communications, but most Serial I/O is asynchronous.

You load the Serial Shift register by storing your data byte into the SEROUT register at $D20D (POKEY register 13). Assuming the clock is running, writing to SEROUT fills the shift register and starts serial transmission. Data is received in reverse order by another shift register with the input coming from pin 3 of the serial plug. The clock must be running when the Data In line goes to 0 with the start bit. The bit value of the Data In line is then passed to the first bit of the shift register with each clock pulse, while the rest of the data moves to the left. When the shift register is full, the byte is finally moved to SERIN, also at $D20D, where it can be read by the OS. The stop bit is always a 1 to leave the serial line ready for the next start bit.

Actually, it isn't quite that simple, because all of this reading, writing and shifting is regulated through the POKEY IRQ interrupt system. Three of POKEY's interrupts are used to regulate Serial I/O: two for transmission and one for reception. Writing to SEROUT fills an Internal Holding register, and when this byte is then moved to the shift register, POKEY signals that it wants more data by setting bit 4 of IRQST.

If the corresponding bit in IRQEN has been set, the interrupt is generated and the OS performs an IRQ, via the interrupt processor, which then routes through the VSEROR (Vectors for the SERial Output Requested) interrupt routine. This routine puts the next byte into SEROUT before the shift register sends the last bit of the previous byte, thus ensuring a smooth transmission of data. When the last data byte has been loaded into SEROUT, the next byte is a checksum. After the checksum is loaded into SEROUT, the VSEROC (Vectors for the SERial Output Complete) interrupt is enabled so that when the last bit of the checksum has been sent, this interrupt routine can be called to terminate transmission.

When data is being received, shifting begins when the Data In line goes to logic 0 with the start bit, and ends with

B&C ComputerVisions

3257 Kifer Road
Santa Clara, CA 95051
(408) 749-1003

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

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.

ATARI TRAK BALL $9.95 A SPICE UP THE ACTION IN YOUR ARCADE GAMES!	**ATARI SPACE AGE JOYSTICK** $5.00 A GUN TRIGGER ACTION!	**STANDARD ATARI JOYSTICK** $4.50 A STOCK UP ON A FEW SPARES	**REMOTE CONTROL JOYSTICKS (2)** $15.95 A REQUIRES 2600 POWER PACK FOR USE WITH 400/800/XL/XE - $5.00	**ATARI TOUCH TABLET** $39.95 A DON'T PUT OFF GETTING THIS HARD-TO-FIND ITEM
400 (16K) COMPUTER $29.95 A 48K UPGRADE KIT $25.00	**600XL (16K) COMPUTER** $49.95 A INCL. POWER SUPPLY & TV SWITCH BOX	**800 (48K) COMPUTER** $69.95 B $79.95 A INCL. BASIC CART & MANUAL	**NUMERIC KEY PAD** $7.95 A INCL. HANDLER DISK USE WITH THE BOOKKEEPER AND BASIC	**850 INTERFACE** $89.95 A LIMITED SUPPLY
1030 MODEM WITH **EXPRESS** $29.95 A GET ON-LINE TODAY!	**835 MODEM** WITH **EXPRESS!** $29.95 A LIMITED SUPPLY	**810 DISK DRIVE** $110.00 B $120.00 A INCLUDES POWER SUPPLY, I/O CABLE & DOS 2	**ATARI BOOKKEEPER** $14.95 - NO BOX ($19.95 WITH RECON KEYPAD) $24.95 - IN BOX ($29.95 WITH RECON KEYPAD)	**DISKETTES** AS LOW AS 20 CENTS 10 FOR $4.00 100 FOR $29.00 1000 FOR $200 MOST ARE UNNOTCHED WITH OLD SOFTWARE

SHIPPING INFORMATION - Prices do not include shipping and handling. Add $5.00 for small items 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

Circle #105 on reader service card.

the stop bit going to 1. At this point, the SERIN (SERial INput) interrupt is generated to call the OS to "read" the data byte before the next one appears. If too few or too many bits are received, or the start and stop bits are not right, error flags are set.

It does not register

The next POKEY register for us to explore is 14 at $D20E, with the split functions IRQST and IRQEN. Reading IRQST will give you the status of all POKEY IRQ requests and is normally only used by the interrupt processor to determine the cause of the IRQ interrupt. Writing to IRQEN will set the bits as follows:

Bit	Vector
5	Serial input data ready
4	Serial output data needed
3	Serial output transmission done

When any bit is set to *1*, then that interrupt request can be passed on to the processor. Conversely, a *0* blocks that interrupt from occurring as we saw in part 1. Finally, the last register at $D20F is labeled SKCTL/SKSTAT as follows:

Bit	Function
7	Serial data input framing error
6	Serial data input overrun error
5	Keyboard overrun (?)
4	Direct connection to Data In line
3	Shift key down
2	Last key still down
1	Serial Input Shift Register busy
0	Not Used

Reading gives the error status of the serial port in bits 5, 6 and 7; keyboard functions in bits 2 and 3; and some miscellaneous functions in the rest. We will only need the last 3 bits for our interface and the rest we will ignore. After each byte is received, the OS checks these bits. If an error occurred during serial input, you get the familiar Errors 140 or 142. All three error bits can be simultaneously reset by writing anything to SKREST($D20A), and this occurs after each reading of SKSTAT. Writing to SKCTL selects the function modes of POKEY's shift registers as illustrated below:

Bit	Function
7	Forces serial output to 0
6	Serial port parameter selections
5	Serial port parameter selections
4	Serial port parameter selections
3	Changes serial out from 1/0 logic to two-tone
2	Changes from normal to fast POT scan
1	Activates keyboard scanning
0	Enables keyboard "debounce" circuits

Now that you know about Serial I/O on the micro level, let us examine it on the macro level and look at the OS. Unfortunately, trying to understand the OS source code is like peeling an onion. Every time you remove one layer, you find another. If you keep at it, you eventually reach the core—and then nothing. So too, the I/O system is layered in levels and at the bottom appears to be nothing. Refer to the flowchart in Figure 3.

When you type the command LPRINT "HELLO" you set into motion a complex chain of machine routines. BASIC sets up IOCB 7 for OPEN. CIO then calls subroutines to set up IOCB7 for a write operation. The buffer pointers are set to the start location of the string and the number of bytes

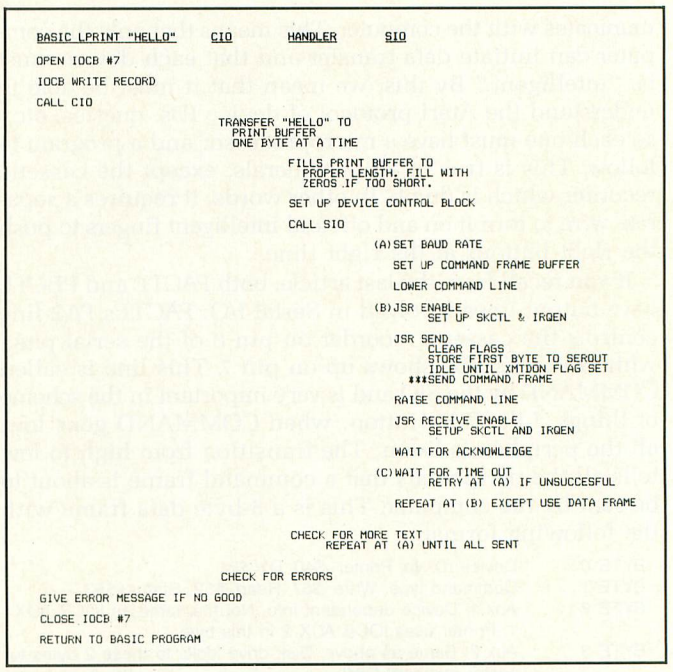


Figure 3 — CIO flowchart

to be sent. CIO then calls the printer handler, which moves blocks of data 40 bytes at a time to the print buffer. Then the printer handler invokes SIO to send the data. If there are less than 40 bytes to send, then zeroes are added to fill up the buffer.

To actually send the data, the handler sets up a block of RAM at $0300, which functions like the IOCB system, except it is used only by SIO. It contains the device ID number (printer is $40), the command type (write is $57), status byte, buffer location, size and AUX1 and AUX2 from the IOCB. SIO then uses subroutines to enable POKEY to send, and after a batch of extra JSRs, you wind up at the very bottom of SIO. What's found there is a short endless loop that can only be terminated by either of two flags being set. One is a Timeout flag and the other indicates that all the data was sent.

Timeouts are created by using the System VBlank Timer 1. While the OS idles in this loop, POKEY will continuously interrupt to increment the pointers and send the next data byte, via the shift register. When the last bit of the last byte is sent, VSEROC sets the Transmit Done flag and allows the OS to break out of the loop. The OS then finds its way back through all the layers of onion to your BASIC program where you first called LPRINT. Similar programming occurs to receive data. So the actual work of data transmission and reception is done exclusively by the interrupt routines.

Catching the right bus

Transferring data on the serial bus at 19,200 bits every second requires careful attention to a rigid protocol. Since there may be as many as ten different peripherals all sharing the same bus, each must have its own unique route number. Before any data can be transferred, the proper device must be queried to ensure that only the correct device com-

municates with the computer. This means that only the computer can initiate data transfer and that each device must be "intelligent." By this, we mean that it must be able to understand the Atari protocol of device IDs, queries, etc., so each one must have a microprocessor and a program to follow. This is true of all peripherals, except the cassette recorder which is dumb. In other words, it requires a separate wire to turn it on and off, and intelligent fingers to push the right buttons at the right time.

If you recall from the last article, both PACTL and PBCTL have output lines involved in Serial I/O: PACTL's PA2 line controls the cassette recorder on pin 8 of the serial plug, while PBCTL's PB2 shows up on pin 7. This line is called COMMAND by the SIO and is very important in the scheme of things. Like E.F. Hutton, when COMMAND goes low, all the peripherals listen. The transition from high to low tells all the peripherals that a command frame is about to be sent by the computer. This is a 5-byte data frame with the following format:

BYTE 0.....Device ID No. Printer=$40, D1=$31.
BYTE 1.....Command type. Write=$57, Read=$52, Status=$53.
BYTE 2.....Aux 1: Device dependent info. Not the same as IOCB AUX 1. Printer uses IOCB AUX 2 in this byte.
BYTE 3.....Aux 2: Same as above. Disk drive looks to these 2 bytes for sector number(LO/HI).
BYTE 4.....Checksum. Total of above bytes.

After this frame is sent, COMMAND returns to its logic 1 state. If the device, whose ID is in byte 0, is on-line, it will now send back a single byte containing the number $41, or the letter "A" for ACKnowledge. If no ACK is received, the SIO attempts to resend the command frame up to 13 times before it gives you an Error 138.

Let me digress for a moment. This whole transfer system makes a couple of assumptions. One is that the computer has a program plan on how to use a peripheral, and the other is that the peripheral knows that same plan. For example, printer frames are 40 bytes long, while disk drive and cassette frames are 128. Both of these handlers are built into the ROM, but peripherals know to use the proper buffer size. Handlers like the 850 interface must be uploaded either from disk or the peripheral itself. Meanwhile, back at the bus, if the computer is performing a "write," then the next event is the transmission of the data frame. If the computer is requesting a "read," then the peripheral sends another 1-byte frame containing a $43 or CoMPlete (CMP), followed by the data frame. If it was a write, when the peripheral finishes receiving the frame, it again sends an ACK, and if the frame is accepted, it sends a CMP. When a peripheral sends data, no other bytes follow the data frame. All this is summarized in Figure 4.

In our above example of LPRINT, IOCB 7 is opened by querying the printer, using the status routine in the printer handler. Status routines are basically READ procedures, but the frame sent back from the peripheral is not the same as a normal data frame. The printer sends back a 5-bit frame with the following format:

BYTE 0.....Success/failure flags. Usually=$80.
BYTE 1.....AUX2: Found in command frame byte 3.
BYTE 2.....Timeout value in seconds. Usually 28.
BYTE 3.....Not used in printer.
BYTE 4.....Checksum.

The disk drive has a different format for status, but again, both the peripheral and the handler know what to expect from each other. Once a satisfactory status frame is received back, the computer starts another cycle to send text to the printer, using command frame/40-byte data frame cycles, until all text is sent. The timeout value returned in byte 2 tells the handler how long to wait between an unsuccessful command or data frame and the next attempt to send. The printer returns 28 seconds and the handler allows only one retry. Try this: OPEN #1,8,0,"P:"

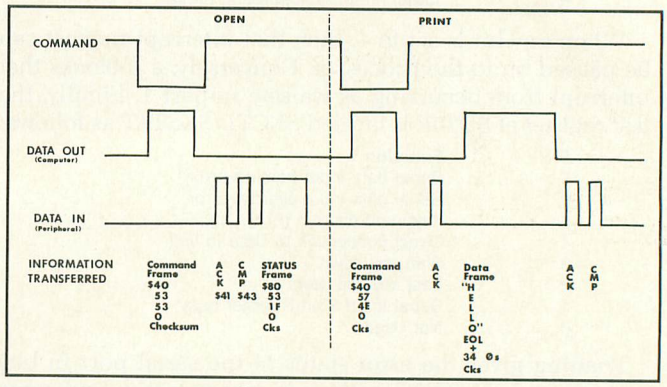


Figure 4 — Serial protocol

Now take your printer "off-line" with the power on and type: PRINT #1;"HELLO"

The computer will attempt to send, but will be unsuccessful, so it will wait 28 seconds and try again. You can hear this if your monitor volume is turned up high. If the printer still does not respond 28 seconds later, an Error 138 will be returned. Such a long time allows you to change paper or align forms, and so on.

For real Zucchini lovers only

Well, this article has been left in the oven too long—it's now hard and dry. The usual purpose of this series is to be useful and illustrative, at the same time providing a simple hands-on approach. I know it's been a long time waiting for your Atari Zucchini à-la-mode recipe, but I promise next month to knock your socks off—to reveal all—and show you how to turn your old Atari into a peripheral without any hardware modifications. Until then, keep reading. A

An Obstetrician-Gynecologist by day, Lee Brilliant, M.D. turns into a bug-eyed computer monster by night. He started on computers in August 1983 with a T1 99/4A and rapidly graduated to Atari. He's programmed Apple II, Commodore and IBM, but prefers his old 800. His favorite pastime is tearing computers apart to see how they tick. Of course, he uses a scalpel!

Q&A

Five things you always wanted to know about video games.

by The Game Doctor

Video games are back, and with them come questions. Old questions, new questions; everybody has questions. These are five of the most frequently asked video game queries and, just for the heck of it, answers are included.

Q: Are video games addictive?

A: Video games are not "addictive." Vulnerable people are apt to become emotionally dependent on a wide range of enjoyable experiences. Movies, comic books, TV and sports have all, at various times, been vilified as "addictive," because of the compulsive behavior exhibited by their devotees. Video games, which provide high levels of stimulation, feedback and interaction, are an especially attractive form of entertainment for those seeking escape from conventional reality.

Video games should be enjoyed as a component in a balanced lifestyle. It should be remembered, however, that the child who spends a great deal of time playing video games is at least engaging in an interactive activity, and not passively vegetating in front of a TV set.

Q: Are video games detrimental to your health?

A: A plethora of physical ills have been ascribed to video games, but no evidence exists to support any of these theories.

Staring for hours at a large, brightly colored video display probably won't improve anyone's vision, but it doesn't appear to be harmful, either. Remember, even folks who spend their entire working day in front of a VDT do not qualify for "hazardous duty" benefits.

The one area where personal experience indicates that gaming *can* cause physical pain regards joystick controllers. A controller that doesn't fit comfortably in one's hand can cause cramps and circulation problems.

Q: Do video games hurt TVs?

A: This largely mythical notion stems from the fact that some of the earliest Pong-type home game systems did, in fact, leave "scars" on TV tubes.

Games which featured fixed white areas (like the perforated "net," which ran vertically down the center of most early game screens) often burned themselves into the picture tube, leaving an imprint that remained, even when the game was switched off.

Since those primitive times, game systems routinely employ color-switching technology to eliminate this problem.

Large-screen TVs, however, sometimes have systems not compatible with video games, so always consult the manufacturer before wiring a video game system to such a set.

Q: Are video game cartridges interchangeable among systems?

A: Alas, no. Although the Atari 7800 also plays 2600-format games, video game software is definitely *non*-compatible. Any potential damage is headed off, however, by the individual design of the cartridge casings, which do not permit even accidental mis-insertion.

Q: Which video game system is the best?

A: The answer to this one is simple: what are you looking for in a video game system?

In terms of cost, the three major systems (Atari 7800, NES, Sega) are all pretty similar.

The Atari, by virtue of its 2600 compatibility, has the largest library of available software, but its 7800 catalog is still rather light on titles. The NES has *plenty* of available software. Nintendo alone will produce thirty-six titles this year, while third-party developers—like Activision, Konami, CapCom, Data East and Absolute Entertainment—will produce another thirty-six. Sega, meanwhile, is still the new kid on the block, with only twenty-four cartridges available, but more promised.

The 7800 is strongest in the arena of arcade shoot-'em-ups, with its knockout versions of Joust, Robotron:2084 and of Pole Position II. Better still, any Atari-compatible joystick will work with this system, providing a selection of controllers at a range of price points.

The NES is strong in several areas, enabling the system to handle medium-action arcade games (Super Mario Bros.), sports (Baseball, Tennis), and even fairly sophisticated action-adventures (The Legend of Zelda)—with equal facility. The NES also boasts a cornucopia of peripherals, from the dubious game-playing robot to a light gun (the Zapper).

The Sega System has the most sizzle of the big three. It is sleekly styled, as are the peripherals. (Nintendo's Zapper looks like something Dirty Harry would be comfortable with, while Sega's Light Phaser would seem more at home tucked through Han Solo's waistband.) It also boasts the most eye-popping graphics, a first-rate library (Choplifter, TransBot, Wrestling, Rambo, etc.) and a hot 3-D peripheral.

Sega has also upped the ante in the memory sweepstakes with its new 2-meg carts, and seems determined to go head-to-head with Nintendo.

In other words, each system has its own strengths; viability is determined solely by the needs of the consumer.

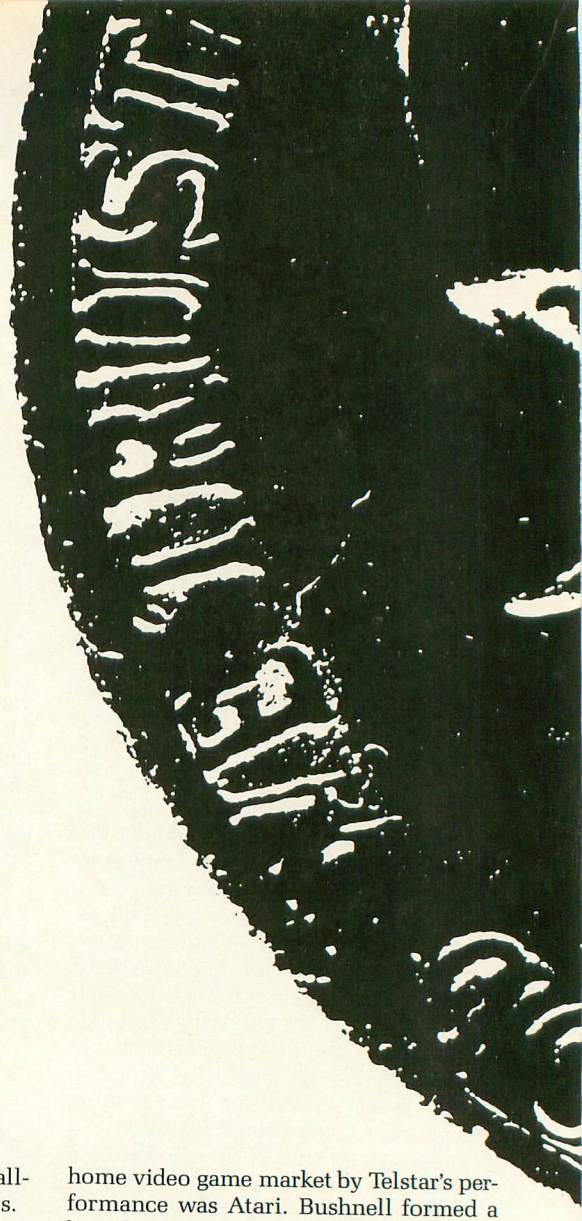
Ask The Game Doctor

Got a question about video game hardware or software? *Video Game Digest* has the man with all the answers, the Game Doctor. The celebrated software sawbones has hung his shingle and opened his office in these pages.

So, if you have a question for the megabyte medico, just send it to: Game Doctor, *Video Game Digest*, c/o **ANALOG Computing**, P.O. Box 23, Worcester, MA 01603.

VIDEO GAME DIGEST

The history of video gaming



Part 2: The Golden Age dawns

by Arnie Katz and Joyce Worley

(In Chapter 1 of our story, we left a deliriously happy Nolan Bushnell counting the proceeds from the first day of operation for his video game coin-op Pong. . . .)

Pong's opening day success was a preview of things to come. Bushnell formed a new company, dubbed Atari, to manufacture and distribute coin-operated Pong machines. (For latecomers, the name derives from a term used in the ancient Japanese strategy game of Go, which is the equivalent of "check" in Chess.)

After Pong, Bushnell continued to address the amusement center audience. He marketed a series of driving games, sports contests and tank battles. Though many were successful, Atari didn't advance the state of the art significantly until the release of Breakout. This venerable wall-bashing game pioneered a play-mechanic which is still highly popular today.

A major technological breakthrough, the invention of the LSI (Large Scale Integration) chip, had the minor side-effect of moving home video gaming to its next stage. The General Instruments AY38500 chip, for example, could carry enough program instructions to play four ball-and-paddle or two target video games.

If the founding of Atari is monument to the entrepreneurial spirit, then the entry of Coleco into the video game field is a tribute to the positive power of corporate drift. The company's name is a contraction of "Connecticut Leather Company." Coleco went from leather goods to toys through a fortuitous string of expansion moves which began with a deal to produce toy gun holsters with a Tom Mix license tie-in. Coleco dove into the above-ground swimming pool market after World War II—and ran right into the boom in the construction of suburban housing developments.

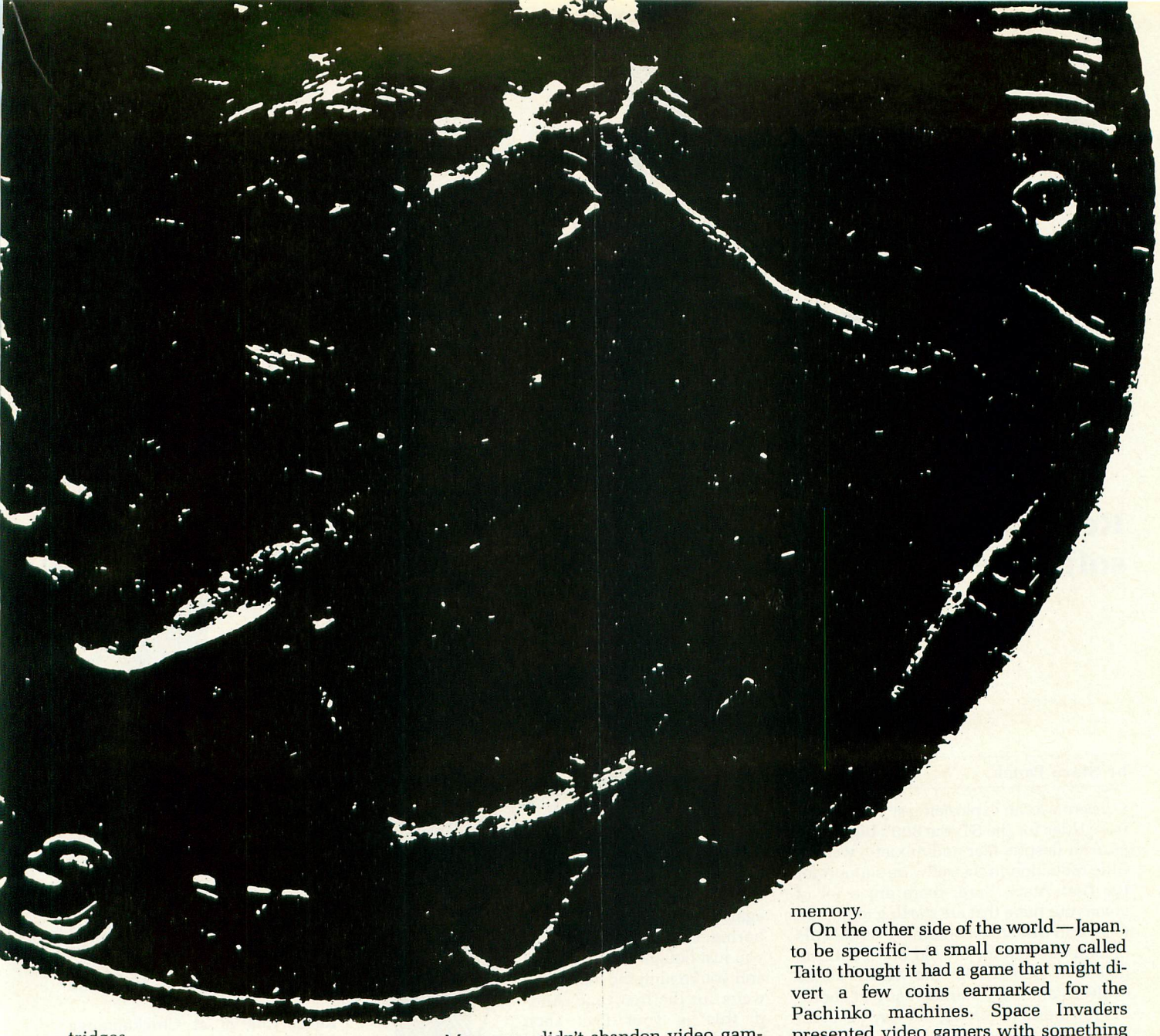
Good fortune continued to smile on Coleco when it became General Instrument's biggest customer for the AY38500 in 1975. Coleco's Telstar Arcade, the first "dedicated chip" home video game system, rocked the toy business. By Christmas of the following year, more than 75 other manufacturers had issued similar video game units. No one could have imagined how quickly all of these players would become obsolete.

Among companies pulled into the home video game market by Telstar's performance was Atari. Bushnell formed a brand new division to sell home systems. He inked a deal with Sears—a landmark agreement guaranteeing nationwide distribution for Atari video games—then secured needed working capital by selling Atari to Warner Communications for $28 million in 1976. He remained in charge as Chairman of the Board.

The glut of all-too-similar hard-wired video game units quickly exceeded the saturation point. Consumers could barely tell one system from another, a condition that wasn't helped by many rapid changes in the manufacturers' product lines. To many, it seemed like a new and better game appeared about every two weeks.

And yet, the games were basically all the same: endless variations on the ball-and-paddle format. Boredom swept the land. The backs of closets began to fill up.

Fairchild Electronics thought it had an idea to chase away the boredom. It introduced the first programmable video game system, the Channel F, in 1976. Fairchild transferred the game programs from the console to interchangeable game car-



tridges.

RCA joined Fairchild in the programmable video game market with its Studio II. The use of black and white graphics was merely the worst of its many flaws. No one greatly mourned its passing.

That wonderful year, 1978

Coin-op giant Bally tested the home market with the Bally Professional Arcade. This small gaming computer, which made its debut during the second half of 1977, boasted by far the best sound and graphics of any home system and played an astonishing roster of games. Bally's lack of familiarity with the home market probably had more to do with the Professional Arcade's lack of success than any intrinsic weakness in the unit. It was too expensive, and its distribution was concentrated on computer stores rather than department, leisure electronics, and discount outlets.

Magnavox didn't abandon video gaming when Odyssey failed to catch the public fancy. The company made a series of dedicated video game consoles during the mid-1970s, culminating in the programmable Odyssey^2 in 1978.

Atari's entry in the programmable sweepstakes was the Video Computer System (VCS), the ancestor of today's 2600. It presented home versions of some of Atari's top arcade programs like Combat and almost instantly leapt to the front of the pack in sales and popularity.

Had one Atari faction had its way, the VCS would have had a memory limit of 2K. Since no game could ever use up more memory than this, they reasoned, why load up the VCS with an unnecessary frill? Caution carried the day, however, It was at least possible that cartridges would some day require more RAM. Soon, software designers would thank Atari for the decision to go with a 4K memory.

On the other side of the world—Japan, to be specific—a small company called Taito thought it had a game that might divert a few coins earmarked for the Pachinko machines. Space Invaders presented video gamers with something entirely new: a shooting game with animated targets.

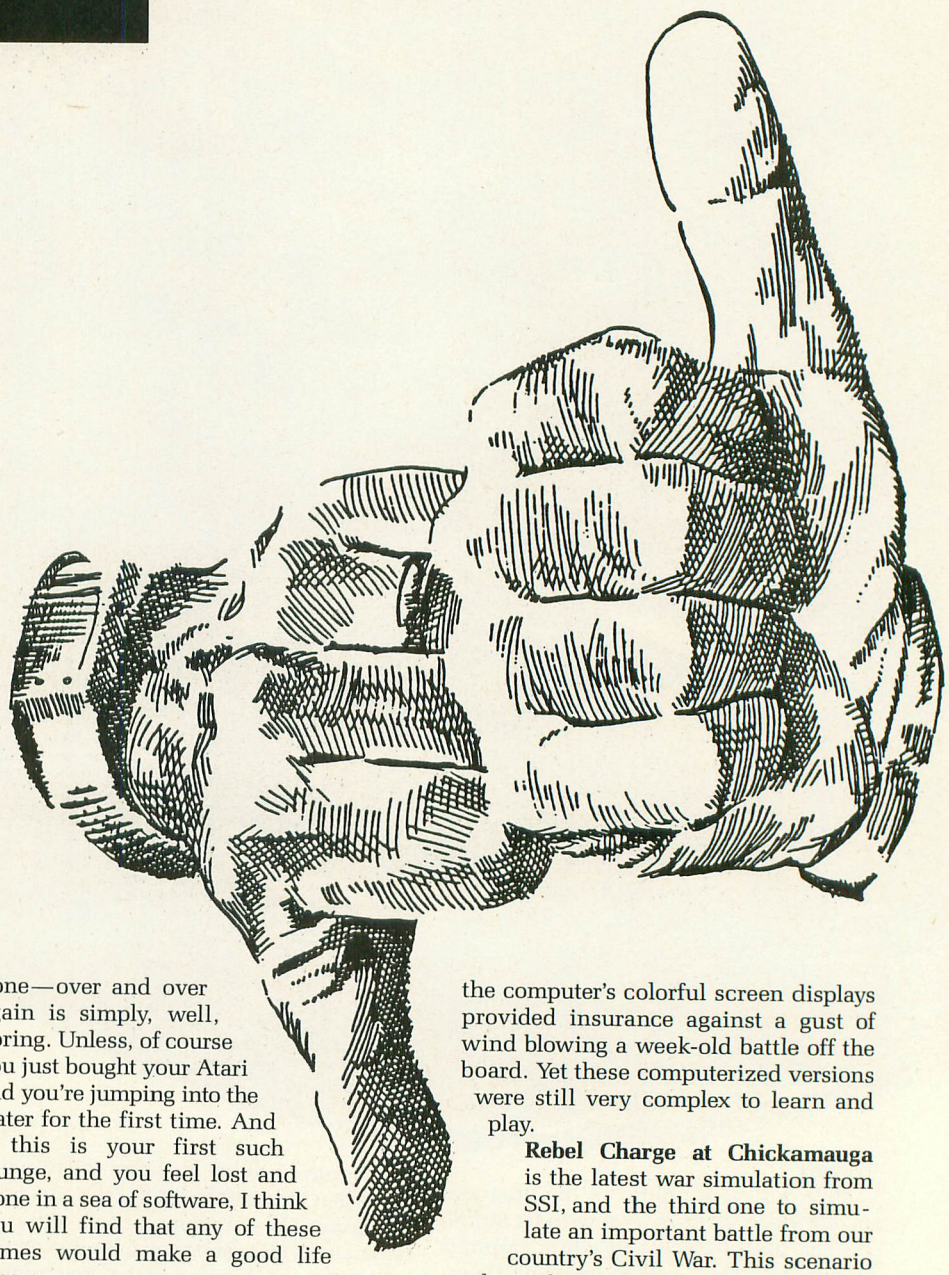
It didn't take Space Invaders six months to shatter the dominance of Pachinko. It would soon prove an even bigger hit in Occidental family amusement centers.

The Mattel Intellivision reached stores in 1980 and was an immediate hit, despite its relatively high price. Although the unit had some drawbacks, like shoddy controllers and slow movement for on-screen objects, it quickly displaced the Odyssey^2 as the major competitor to the Atari VCS.

Can George Plimpton help Mattel dethrone Atari? Why is Jim Levy having all those lunches with programmers? And what is that wocka-wocka sound? The answers to these and other burning questions will be found in "The History of Video Games Part III: The Golden Age" in next month's **Video Game Digest.**

Panak strikes!

Reviews of the latest software



by Steve Panak

There was an explosion last year in software titles for the ST, the 800's big brother. And despite fear and anxiety, we saw little reduction in the software support for the first Atari. Sure, there are a lot of games out there that are starting to require 64K, and diskettes containing mouthfuls of data too rich for my old 800's delicate digestive system. But the XL/XE series evolved, and now stands ready to shoulder this burden, paving the way for more and better games.

Last year we were treated to a variety of fine 8-bit software. But by the same token, there were a few disappointments as well—the biggest being the lack of earth-shattering new releases. Don't get me wrong here; there were a lot of great game simulations from SSI and GDW, plus a continuum of high-quality text adventures from Infocom.

I might be tempted to reach the conclusion that all genres have been covered, all stones turned, every iota of creativity spent. But an addictive little ditty by the name of Trailblazer, which I reviewed last time, showed that not to be the case. I think it's just a little harder to come up with something really new these days. The well is not dry, it simply takes a little more elbow grease to get it primed.

If there's one thing I'd like to see in the future it would be more innovation. Getting the same thing—however well done—over and over again is simply, well, boring. Unless, of course you just bought your Atari and you're jumping into the water for the first time. And if this is your first such plunge, and you feel lost and alone in a sea of software, I think you will find that any of these games would make a good life preserver.

Rebel Charge at Chickamauga

by Chuck Kroegel
SSI
1046 North Rengstorff Avenue
Mountain View, CA 94043
48K Disk $49.95

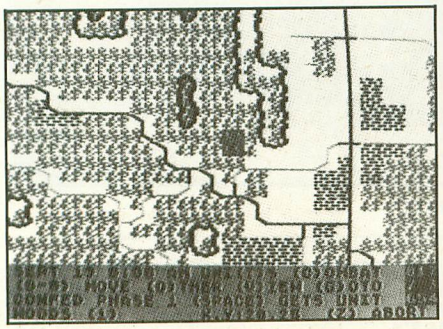
In the beginning, war simulations were played on large maps, with hundreds of small, cardboard squares representing the infantry and artillery battalions, while dice and complex charts decided life and death on the battlefield. Then the computer took over, with its ability to quickly calculate values, thus relieving much of the monotony of the game. Moreover, the computer's colorful screen displays provided insurance against a gust of wind blowing a week-old battle off the board. Yet these computerized versions were still very complex to learn and play.

Rebel Charge at Chickamauga is the latest war simulation from SSI, and the third one to simulate an important battle from our country's Civil War. This scenario explores the Confederates' last major attempt to take control of a split nation. And, by combining extraordinary realism and three degrees of difficulty with simple control and intuitive commands, SSI creates a game that all war-gamers will want on their side.

It's the fall of 1863. The Union, having scored victories at Gettysburg and Vicksburg, seems invincible. Still, the Confederates were not known to give up easily, so the two opposing forces find themselves clashing at Chickamauga Creek. Although this battle may not be the most famous of the war, this computer-recreation makes you feel as though you're there. And who knows, you might just change history.

The battle is recreated in 13 two-hour

turns, requiring up to 40 hours to complete. The game uses a refined version of the system last seen in Gettysburg: The Turning Point, which allows incredible control of the individual units. The myriad of commands are easily learned and implemented. In fact, this latest product of SSI's evolution is the easiest to play that I've yet seen.



Rebel Charge at Chickamauga

The simple game system contains two phases: operation and combat. The former lets you examine and give orders to your troops, while the latter gives you the power to resolve the various conflicts that arise. Using the keyboard, you move the cursor around the map, examining and ordering your units. Tapping the space bar picks up a unit (or the first unit, in the event you have stacked multiple units on one square), and displays various attributes, such as size, firepower, morale and efficiency. At this point, the command menu appears and you are able to move or aim the unit. A key is provided to cycle you to the next unit on the map awaiting instructions, insuring that no one is left without orders.

Operation points, which are awarded during each operation phase, determine how far a unit can move and how much it can fire. Efficiency, fatigue and morale points modify the play, hence decreasing accuracy and slowing movement. Intermediate and advanced versions of the game add even more realism, taking into account the leaders' strengths and their movement among units. It's staggering how many factors are taken into account. For instance, combat results are modified depending on what type of terrain the troops are in, their formation and location, and even their past success.

Once all the orders have been given, you move to the combat phase, in which the computer resolves all pending conflicts. The map scrolls to display each unit as it is engaged in combat, while a text window at the bottom of the screen flashes statistics, such as unit involved and number of men killed.

Once the battles have been resolved, and the computer has made its move, you begin the next turn, starting the cycle over again. This continues until 13 turns have passed—or until one side is clearly the winner.

Sound, which can be turned off if desired, enhances the play. You can hear the artillery shells falling and the rifles firing. The graphics are also quite good. Most players should have very little difficulty differentiating between their many unit icons, as each has very specific markings.

The manual is superbly designed and written. A tutorial gets the impatient player, as well as the novice, right into the thick of the action, while another section details the rules for the intermediate and advanced games. A generous amount of charts, maps and tables keep you apprised of modifiers, troop location and the chain of command. Six pages of background provide a nice history primer on this important battle. But, despite all of these kudos, I do have a couple of complaints.

First of all, I truly hate using the numbers 1 through 8 to control movement. Even with the on-screen help window, it's difficult to remember which numbers correspond to each of the eight compass headings, especially since the numbers don't exist in keypad form. It would be nice to have the option of using a joystick to control cursor movement. Also, there are too many disk swaps at the end of each turn. I guess that's the price you have to pay for having a game crammed into 48K.

Overall, **Rebel Charge at Chickamauga** is a simulation well done, from a company that knows how to get the job done. If you're into this type of game, you will not be disappointed, although the high price could wound your wallet rather severely.

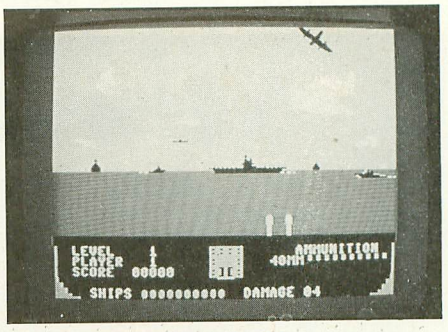
Triple Pack

by Bruce and Roger Carver
ACCESS SOFTWARE, INC.
#A 2561 South 1560 West
Woods Cross, UT 84087
48K Disk $19.95

While the packaging is new, the three games contained in **Triple Pack** are old. When sold separately, I felt that these three games offered little. However, the idea of selling them as a package at a special price, changed my feelings.

The two disks contain the games Beach-Head I and II, and Raid Over Moscow. Since all of these games have been examined in depth in the past (issues 30, 49 and 51), I'll not go into great detail on them here. Suffice it to say that each follows the same basic framework: You engage in a series of battles on your way to a final confrontation. These battles consist of a number of different screens, and sometimes feature different combat methods.

In Beach-Head I, you move through six sequences in which you search out the enemy, attack their island, and finally destroy their fortress. In Beach-Head II, it seems that the evil dictator escaped your prior attack, and is ready for a second go at it. In four sequences, you attempt to rescue hostages captured in previous battles, ending with a final showdown against the "Dragon" himself. Beach-Head I and II are very similar. The main difference is that in II you control individual troops to a greater extent.



Beach-Head I

In Raid Over Moscow (my favorite of the three), you are the squadron commander of the U.S. Defense Space Station, with a virtual suicide mission: You must stop a nuclear attack. You have to knock out launch sites, then penetrate Moscow and destroy the defense center. In seven play sequences, you determine the location of the latest missile launch, then destroy it. In the final chapters of this saga, you storm the defense center's reactor, wiping out the defensive troops and wasting the maintenance robots. Without its coolant, the core becomes unstable and detonates. The most successful agents not only complete this suicide mission, but survive it as well. This game provides plenty of opportunity for strategy, as you try to decide which targets need your immediate attention, and which can wait.

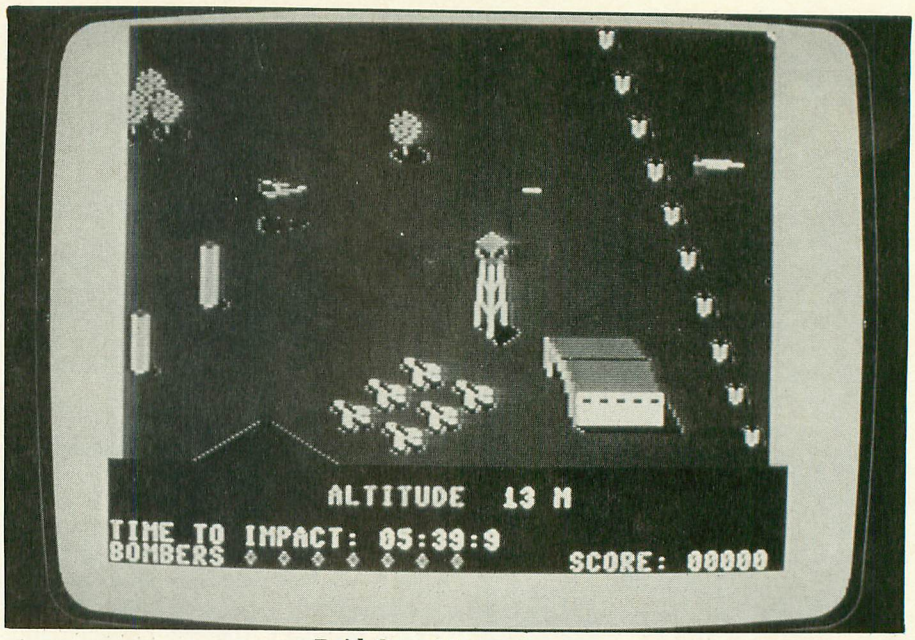
Both Beach-Head I and II can be played by one or two players, one taking the offense and one the defense, and each al-

lows multiple skill levels. Moscow similarly allows selection of one of three skill levels, however, only one may play the game at any given time. The graphics in each of these games is fair to good, and nearly identical—at least in style. Nothing spectacular, their resolution and detail are similar to that found in the vast majority of 8-bit gameware. I had little trouble controlling any of these games, and the action is undeniably fast, especially when those tracer bullets start flying.

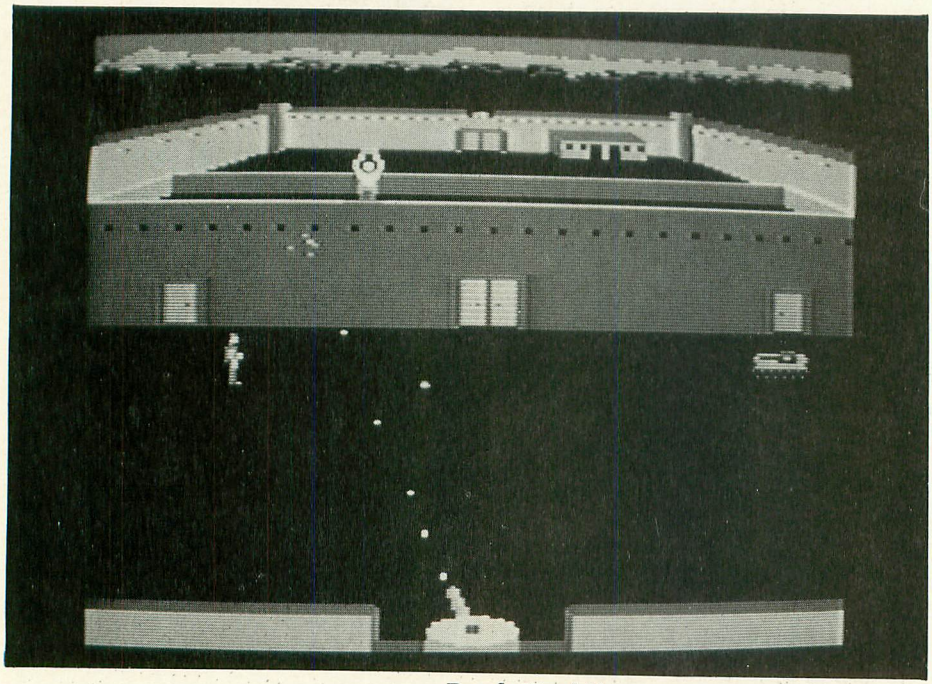
Each program has its own simple black-and-white instruction manual. These are, for all intents and purposes, reprints of the manuals packaged with the original versions of these games. A small cardboard slip provides quick loading instructions applicable to all three. This setup is fine, although I'd rather see all documentation gathered in one place. The trade-off of inconvenience against the low price, however, is tolerable.

The repackaging of these three games into a bargain trio brings the price down to about $7 a game, which is more than fair, considering that each of these games sold separately for $39.95 a year ago. **Triple Pack** packs a triple punch that might just be worth a look.

That's a wrap for this month. Last year, we saw a couple of great games, several good games, and a lot of really mediocre ones. You can rest assured that the above fall right about in the middle. While they break no new ground, they do provide good, strong performance that will surely be appreciated by Atari newcomers.



Raid Over Moscow



Beach Head-II

ANALOG's new customer service number is (818) 760-8983. You may also write to:

ANALOG Computing
P.O. Box 16927
North Hollywood, CA 91615

ST-Log
P.O. Box 16928
North Hollywood, CA 91615

The above phone number and addresses are for subscription matters only.

SUPER DATA BASE 1 – 2 – 3
(more than a data base)
Only
$59.95 *
Will generate all the custom software you will ever need.

1. Create Professional Data Screens.
2. Run Custom Reports.
3. Link Everything Together With Custom One Key Menus.

• **Easy Manual and Tutorials** •

For IBM PC, PC Jr., XT
Atari (except ST) Apple II, IIE
Commodore 64
Hard Disk compatible - 36,000 Files
• ***High Quality*** •
Adventure Games **Only $ 19.95 ea.** *

I. M. P. SOFTWARE
Rt. 1 Box 362
Ozark, Mo. 65721
Call your local dealer or us at
(417) 485-6398
* Add $2.00 shipping and handling
Missouri residents add 5.25% sales tax

Circle #106 on reader service card.

New! 3-D Graphic WW11 Sea Game

Now play "Destroyer Warship" on your XL/XE Computer System! It is a WWII arcade type Sea Battle against Japan! You are captain and protect Midway Island and the Shipping Lanes! Its a hard war if you can take it! Play it! Be a hero and play "Destroyer Warship"!! * Now Only! $14.95 + 2.95 shipping & handling. (Continental U.S.).
* Requires XL/XE, joysticks & 1050 drive.
64K

New! "Lem Landing Simulator & Fastbudget". Land the LEM on the planets or figures your budget! (Order Yours Today!)
* LEM Landing Simulator & Fastbudget
* Now Only $9.95 + $2.95 shipping & handling. (Continental U.S.).
* Requires XL/XE, joystick & 1050 drive.
64K
* Cashier's Check or Money Order only!
* Michigan Residents add 4% Sales Tax.

Send To: IntelaSoft
P.O. Box 102
St. Charles, MI 48655

Prices are subject to change without notice. Due to low prices all sales are final.

Circle #107 on reader service card.

Working with the Atari ST

by David Lawrence and Mark England
SUNSHINE BOOKS
12-13 Little Newport Street
London WC2H 7PP, U.K.
134 pages — $14.95 (7.95 pounds sterling)

by John W. Little

Working with the Atari ST (***WAST*** for brevity) is intended for those inexperienced with their new ST, but who *have* gotten it set up and running.

The book's sixteen chapters do an adequate job (sometimes more than adequate) of describing the hardware and system, and instructing us in the use of the GEM desktop. ***WAST*** has both a table of contents (two!) and an index. I feel it is among the best books out for the new ST user.

In fact, even a person who's been using the machine for some time may find, on reading this book, that his or her early education was incomplete; I found the section on configuring the desktop invaluable, and I have yet to see a comparable treatment of installing and using applications.

Excepting typos, none of which are critical, there are few errors in ***WAST***. Those I did find are a result of the fact that it was written when the ST was very new—or perhaps not even released yet.

Chapters 1 through 7 describe the hardware and operating system. For the most part, ***WAST*** gives us, in layman's terms: the 68000 and its memory, comparing it with other common CPU chips; the graphics system, touching lightly on the multi-plane color concept; the sound system, with a short discussion of the Programmable Sound Generator and MIDI; the peripheral ports and devices that use them (mouse, floppy, etc.); GEM, with sections on VDI and AES; and TOS, comparing it with CP/M.

In these chapters, a predictable error of "early-ST" books is found. The ST is said to have interfaces for three different kinds of monitors: composite video, RGB, or monochrome (page 13). The statement that the ST can handle only three accessories dates the book as "pre-ROM TOS."

With the general description out of the way, Chapter 8 gets down to the "how-to" of using the ST, covering mouse techniques (selecting, dragging, double-clicking), window techniques (moving, sizing and selecting the window itself, as well as using the scroll bars and sliders to move the window contents), and file manipulations (copying, deleting). It's eleven pages of good, solid information that is basic to using the desktop.

I feel Chapter 9, though brief, is one of the most important in the book. It tells how to configure the desktop so that it is comfortable and convenient for the individual user. It also instructs us in how to save that configuration so it will be the same the next time we boot. That may seem elementary to experienced users, but learning to control the environment is crucial to making the ST a joy to use.

Chapter 10 continues on "environment control," with drop-down menus. Each option of each menu is discussed in ample detail, including the "original" desk accessories: VT52 emulator; control panel, with which the user can change some parameters, such as screen color and key and mouse-button sensitivity; RS232 configuration; and printer installation.

For those not familiar with the concept of pathnames, Chapter 11 is a tutorial on the use of folders, with an exercise to work while reading the chapter. It also explains the dangers of deleting folders without ascertaining their contents.

Chapter 12 is the only thorough discussion I've seen on the installation and use of ST applications. I had often seen the menu option "Install Application," without knowing how much it exemplified the power of GEM. In my opinion, the information here and in Chapter 9 are worth the price of the whole book.

Chapter 13 describes some of the basic types of applications available for the ST (word processors, databases, etc.) and again gives away its early publication by espousing the virtues of GEM Draw, GEM Paint, and GEM Write, which were not released for the ST. Ignoring that flaw, it does a good job of informing the neophyte on the different types of applications available, and what features might be considered when purchasing them.

Chapters 14 and 15 introduce Logo and BASIC, respectively. The purpose is simply to give the reader a taste of the languages, including a sample program or two for each language.

Finally, Chapter 16 is a discussion of an alternative to GEM, the BOS operating system, by Business Operating Software. It covers some of the features of BOS and makes note of substantial differences between it and GEM, without attempting to picture one or the other as superior.

With the few exceptions noted, this book is old hat to someone who's been using the ST for any length of time. For the new user, though, this one gets my vote as the book to have, both for its overall view of the ST and its specific instruction on actually using the machine.

John W. Little started computing on an 800 about four years ago, and does most of his programming on the 8-bits in assembly. He likes to use the joystick ports for experimenting with real-world-to-computer I/O. He bought one of the earliest STs and fiddles around with C and 68000 assembly.

When you want to talk Atari

XL/XE HARDWARE

INTERFACES

ICD
P:R Connection 58.99
Printer Connection 39.99
Supra
1150 . . . . . 38.99
1151 (1200 XL) 39.99
Xetec
Graphix Interface 38.99
Atari
850 Interface 109.00

COMPUTERS

Atari 130XE $135

Atari
65 XE 99.99

XL/XE ENHANCEMENTS

Axion 32K Mem. Board (400/800) . 19.99
Atari 80 Column Card 84.99

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 . . . . . 209.00
Supra
2400 Baud XL, XE 169.00
2400 Baud ST 169.00

MONITORS

Magnavox
CM8502 13" Comp. & Cables . . . 169.00

ST HARDWARE

520 ST FM RGB/COLOR $789

Includes: 520 ST FM with 3½" drive built-in, mouse, power supply and 1224 color monitor.

1040 RGB/Color System 899.00
1040 Monchrome System 769.00
1040 Computer (no monitor) . . . 639.00

520ST FM Monochrome System (Includes: 520 ST, internal drive, modulator, mouse, Basic and monochrome monitor) . . . . . Call

SM124 Monochrome Monitor . . 159.00
SM1224 Color Monitor 329.00

DRIVES

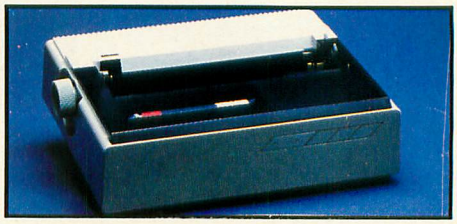
Atari
XF551 Drive (XL/XE) 189.00
AA314 DS/DD Disk (ST) 199.00
AA354 SS/DD Disk (ST) 119.00

Atari SHD204 20 Meg for ST $559

I.B.
5.25 ST Drive 229.00
Indus
GTS 100 3½" Drive (ST) 199.00
GT Drive (XL/XE) 179.00
Supra
20 Meg Hard Drive (XL/XE) 639.00
20 Meg Hard Drive (ST) 539.00
30 Meg Hard Drive (ST) 689.00

PRINTERS

Atari
1020 XL/XE Plotter 31.99



Atari XDM 121 Letter Quality XL/XE $149

XM-M801 XL/XE Dot Matrix 189.00
XM-M804 ST Dot Matrix 179.00

Brother
M-1109 100 cps Dot Matrix 199.00
M-1409 180 cps Dot Matrix 309.00
HR-20 22 cps Daisywheel 339.00

Citizen
120D 120 cps Dot Matrix 149.00
180D 180 cps Dot Matrix 169.00
Premier-35 35 cps Daisywheel . 479.00

Epson
LX-800 150 cps, 80 col. 179.00
Hi-80 4 pen plotter 249.00
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
EX-800 300 cps, 80 col. . . . . Call

NEC
P2200 pinwriter 24-wire 379.00
P660 pinwriter 24-wire 459.00
P760 pinwriter 132 col. 669.00

Okidata
Okimate 20 color printer 129.00
ML-182 120 cps, 80 column 229.00
ML-192 + 200 cps, 80 column . . 359.00
ML-193 + 200 cps, 132 col. 469.00

Panasonic
KX-P1080i 144 cps, 80 col. 189.00
KX-P1091i 194 cps, 80 col. 199.00

Star Micronics
NX-1000 140 cps, 80 column . . . 169.00
NX-15 120 cps, 132 column 319.00

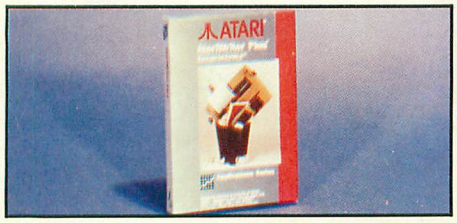
Toshiba
P321-SL 216 cps, 24-wire 539.00

COMPUTER MAIL ORDER

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

XL/XE SOFTWARE

Access
Leaderboard Golf .....(pk.) 15.99
Tournament Disk 13.99
Accolade
Hardball 19.99
Atari
Filemanager 11.99
Music Painter 11.99



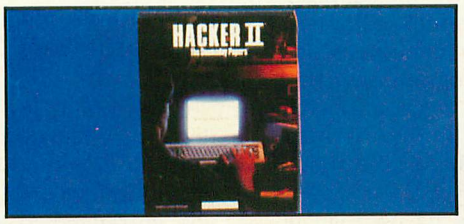
Atariwriter Plus $35.99

Atari Program Exchange
Misc. Programs (cassettes) ....at 1.99
Batteries Included
Home Pack 18.99
Broderbund
Printshop 25.99
Karateka 12.99
Cygnus
Starfleet I 32.99
Datasoft
221 Baker St. 21.99
Electronic Arts
Touchdown Football 11.99
Pinball Construction 9.99
Microprose
Top Gunner 15.99
F-15 Eagle Strike 21.99
Silent Service 21.99
Origin Systems
Ultima 4 .....36.99
Roklyn SPECIAL
Gorf/Wizard of Wor/AT Deluxe/
Anti-Sub/Journey to Planet ..(ea.) 3.99
Strategic Simulations
Gemstone Warrior 11.99
Sublogic
Flight Simulator II 31.99
Scenery FL, NC, SC 14.99
X-Lent
Typesetter 22.99
Printshop Interface 21.99

ACCESSORIES

Maxell
MD1-M SS/DD 5¼" 7.99
MD2-DM DS/DD 5¼" 8.99
MF-1DDM SS/DD 3½" 12.49
MF2-DDM DS/DD 3½" 18.49
Sony
MD1D SS/DD 5¼" 7.99
MD2D DS/DD 5¼" 9.49
MFD-1DD SS/DD 3½" 12.49
MFD-2DD DS/DD 3½" 19.49
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 (XL/XE)
Sparta DOS Construction Set ...28.99
US Doubler/Sparta DOS 47.99
Real Time Clock 48.99
Rambo XL 28.00
Multi I/O Board 256K 169.00
Multi I/O Board 1 Meg 299.00

ST SOFTWARE



Activision
Hacker II Doomsday $29.99

Access
Leaderboard Golf 22.99
Antic
Stereo CAD 3-D 54.99
Atari
Algebra I Vol II GRD 7-9 .....16.99
Avant Garde
PC Ditto 69.99
Batteries Included
Degas Elite 39.99

ST SOFTWARE

Cygnus
Starfleet I 33.99
DAC
Payroll 44.99
Electronic Arts
Gridiron Fooball/Auto Duel .(ea.) 30.99
Firebird
Guild of Thieves 25.99
Infocom
Beyond Zork 37.99
Michtron
Major Motion 25.99
Microprose
Silent Service 24.99
F-15 Strike Eagle 24.99
Miles Software
ST Wars 28.99
Mark Williams
C 119.00
Paradox
Wanderer (3D) 25.99
War Zone/Fireblaster 26.99
Psygnosis
Barbarian/Deep Space .....(ea.) 26.99
Strategic Simulations
Rings of Zilfin 23.99
Sublogic
Flight Simulator II 32.99
Timeworks
Swiftcalc/Wordwriter .....(ea.) 47.99
Partner ST 41.99

DAC
Easy Accounting $59.99

Unison World
Printmaster Plus 25.99
Word Perfect Corp
Word Perfect 4.1 .....239.00

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

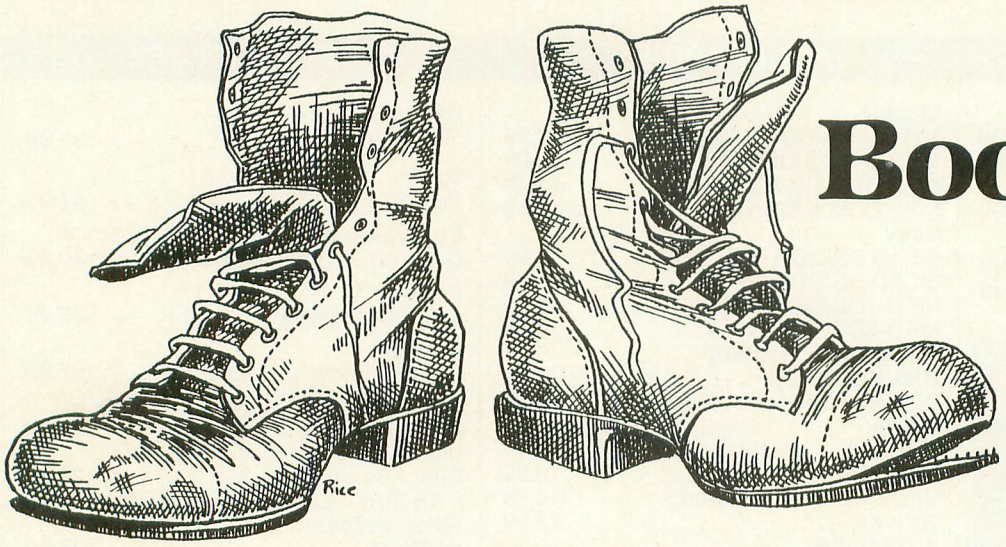
Educational, Governmental and Corporate Organizations call toll-free 1-800-221-4283

CMO. 477 East Third Street, Dept. B7, Williamsport, PA 17701

ALL MAJOR CREDIT CARDS ACCEPTED

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.

Circle #108 on reader service card.



Boot Camp

RAMdisk file copier

by Karl E. Wiegers

Recently, while I was talking to a regular **Boot Camp** reader, he made the interesting observation that the logical process of going from program idea to final product really isn't discussed much in the public literature. You can read scads of articles about this language or that, programming tips from experts, and even books on the best ways to design complex software systems. However, very little is written about the thought processes other programmers go through in developing an idea.

Assembly language programming poses its own set of questions about the most effective sequence of program development steps. This is because you have to do a lot of the work in assembly (such as memory allocation) that the computer handles in high-level languages.

This month's **Boot Camp** will be a little different. I'll try to pass on to you the thoughts that flit through my mind, and the resulting activities, as I develop an assembly program. Along the way, we'll actually generate a useful product.

The first thing I do when starting a MAC/65 session with my 130XE is set up a RAMdisk and copy my MACRO.LIB and SUBS.LIB files to it. Then I'm ready to roll. I've often wanted a program to handle this chore for me, and, by the end of this discussion, we'll all have one in hand. The sample program reads a disk file called RAMDISK.FIL that contains a list of the files you want to place in the RAMdisk. Each of those files is copied in turn, and the MAC/65 editor magically appears afterward. You can name this file AUTORUN.SYS, so the whole process takes place automatically on booting.

This program uses several macros from earlier columns, as well as a few new ones. If you don't have a macro assembler, you'll need to modify the code to expand the macros out by hand and make any other changes specific to your assembler. If you don't have an Atari 130XE with the RAMdisk, but have a second physical disk drive, you can change the references in the program from D8: to D2:. If you only have a 64K or smaller computer with a single drive, please bear with us and keep reading, since you may encounter some other useful information.

Getting started

Quite often, the hardest part of writing a program is deciding what sort of program to write. Each month I have to think of something that at least *some* of you will find useful, informative, interesting and amusing enough to keep you entranced to the very last word. Your own program ideas are probably more goal oriented . . . How can I write the world's best word processor program? How can I dazzle my friends with some graphics displays? How can I save twenty bucks and write my own checkbook balancing program? How can I make $300,000 quickly in the software business? This month's idea came from my desire to find more ways to have the computer work for *me*, instead of the other way around.

I tend to think about my programming projects for quite a while before actually sitting down at the keyboard. This bit of wisdom leads to a discussion of the classic steps involved in software development (or any other problem solving exercise): analysis, design, programming, testing, debugging, release on an unsuspecting public. These steps tend to flow from one into another, and all too often they become hopelessly interwoven.

Analysis consists of making sure you understand the problem. A lot of people blithely skip this step and proceed with only a vague notion of what they're trying to accomplish. If you aren't sure what you're trying to do, how will you know when you're done? I always feel better if I know just where I'm heading before I write any code.

Design involves coming up with a solution to the problem. This is the most challenging part of the project, con-

ceptually, and one of the most critical. In the years I've spent as a professional software developer in the Eastman Kodak Research Laboratories, I've learned that every minute you spend on design is worthwhile. For smaller projects like the RAMdisk copier program, I do much of the design in my head—over and over and over again—until I think I've figured out every angle. Then I put it on paper and see all the things I missed.

There are many techniques—all useful, but none perfect—for working out a design on paper. The traditional method is flowcharting. While flowcharting is still useful for figuring out the details of logic flow, it's severely limited when dealing with any but the smallest programs. Many new methods for representing a system design have appeared in the last few years; I'm still looking into them and trying different approaches. All of them deal with the basic pieces of the puzzle: input (what data are we going to process?); a process (what are we going to do to the data?); and output (where do we put the results?).

Let's think about this in the context of today's program, and I'll try to reconstruct some of my thought processes in a coherent fashion.

Analysis and design

First, we'll deal with the problem analysis. I want to write a program that will copy a specified group of files from the boot disk to a RAMdisk. The program should run automatically upon booting. Of course, the RAMdisk must be set up before this program can run. After completion, control of my Atari should be passed either to the cartridge (probably MAC/65) or the DOS menu, if no cartridge is present.

Now some thoughts on the system design, pretty much in the order in which I first had them. I need to have a list of the filenames to be copied. This list could be in a file on the boot disk. Let's call it RAMDISK.FIL. The program will end after the last record in RAMDISK.FIL has been processed.

I could read the file in its entirety into memory, then copy it to the RAMdisk. This would require a lot of RAM for a big file. Alternatively, I could read it in little chunks, copying each chunk to the RAMdisk after reading it. The last might be shorter than earlier chunks. This method won't require much RAM, since I can make the chunks as big or small as I like. I could even copy it 1 byte at a time. I decided to use the second method, of 255-byte chunks.

(So far, the problem analysis and design are really independent of the language and computer that will be used for the program. In fact, this analysis could apply just as well to a bunch of monks copying some documents by hand. We can describe this level of system design as being very abstract. Now, I'll become more concrete and continue the design at a more detailed level.)

I'll write this program on my Atari 130XE in assembly language, using MAC/65. I'll need one IOCB (Input/Output Control Block) for reading the RAMDISK.FIL file, another IOCB for reading the file being copied, and a third for the output file on the RAMdisk. I'll process chunks of files 255 bytes long at a time. The program will stop executing when an End-Of-File marker (EOF, for short) is read from RAMDISK.FIL. I'll set this program up to autorun upon loading from disk. If the program file is named AUTORUN.SYS, it will load and execute upon booting.

How about memory allocation? I'll begin this program at address $5000, just because I always do. I need to reserve a block of RAM 255 bytes long to hold each chunk of file as I read it (a "buffer"). I also need to reserve space to hold the name of the file being processed (which I just read from RAMDISK.FIL), and space for the output name (the same as input, except with a D8: drive specification), at 16 bytes each. (If I were writing a graphics program, I'd also decide how much RAM I needed for display list, character sets or whatever, and what addresses I'd use.)

Now, what sort of things might go wrong that I must teach my program to handle? Maybe the user's computer doesn't have a RAMdisk setup. Maybe the RAMDISK.FIL file is missing. Maybe the files listed in RAMDISK.FIL aren't on the boot disk. In the first two cases, the program will simply stop executing. For the third situation, I'll copy any files I find and print a message for any that aren't found. I'll use any macros I have lying around to save my typing in code wherever possible.

Picture perfect?

Whew! Did you follow all that? Maybe a diagram would clarify things somewhat. In Figure 1, I've drawn what's called an "action diagram." This is a simple way to sketch out the detailed logical structure of a program, as an alternative to a flowchart. I'm sure you could also draw a flowchart for a program this short, but let's move into the 1980s.

```
+--- COPY FILES TO RAM DISK
|
|   Open D1:RAMDISK.FIL
|
|   +--- If error
|   |
|   |   Print error message
|   |
|<--+---+---Escape
|   |
|   +--- Else
|   |
|   |   +--- Loop 1
|   |   |
|   |   |   Read a record from D1:RAMDISK.FIL
|   |   |
|   |   |   +--- If EOF
|   |   |   |
|   |   |   |   Close D1:RAMDISK.FIL
|   |   |   |
|<--+---+---+---+---Escape
|   |   |   |
|   |   |   +--- Else
|   |   |   |
|   |   |   |   Set up output filename
|   |   |   |
|   |   |   |   +--- Loop 2
|   |   |   |   |
|   |   |   |   |   Open input file
|   |   |   |   |   Read a chunk
|   |   |   |   |
|   |   |   |   |   +--- If not EOF
|   |   |   |   |   |
|   |   |   |   |   |   Write chunk to output file
|   |   |   |   |   |
|   |   |   |   |   +--- Else if EOF
|   |   |   |   |   |
|   |   |   |   |   |   Get number of bytes read
|   |   |   |   |   |   Write chunk to output file
|   |   |   |   |   |   Close input and output files
|   |   |   |<--+---+---+---+---Escape
|   |   |   |   |   |
|   |   |   |   |   +--- End if
|   |   |   |   |
|   |   |   |   +--- End Loop 2
|   |   |   |
|   |   |   +--- End if
|   |   |
|   |   +--- End Loop 1
|   |
|   +--- End if
|
+--- End COPY FILES TO RAM DISK
```

Figure 1

The brackets in the action diagram enclose logical blocks of operation. The IF/ELSE/ENDIF structures show conditional actions that depend on various situations. The brackets marked with a double line (equal signs) at the top indicate a repetitive (looping) action. The horizontal arrows labeled "Escape" show ways to exit from the inner bracketed actions when certain conditions (usually errors) are encountered. Sometimes we can use error conditions to our advantage, such as when an EOF condition error means we're done with a particular operation.

After all this design work, I'm ready to begin writing a program. Again, let me stress that the more time you spend going over and over the design steps, the less aggravation you'll encounter when coding, testing and debugging. I guarantee you'll think of new angles, potential pitfalls, and so on, each time you iterate through the design process. Believe me, it's a lot easier to modify an action diagram than rewrite a segment of your program because of a logic flaw or functional omission.

Toolbox

While working through the program design, I also think about what tools I'll need to implement it. For a graphics program, I ponder such heavy questions as whether I need a vertical blank interrupt routine to move players, display list interrupts to make the screen look nice, a mixed-mode display list (build your own or modify the default?), or some redefined characters. I think about the macros and subroutines I have available that might do the job, and any others I'll have to create.

For the file copier program, I need my OPEN macro to open some IOCBs, and, of course, the CLOSE macro when I'm finished with them. The PRINT macro would be nice for putting error messages on the screen. I can use INPUT (discussed in issue 58) to read the entries in RAMDISK.FIL. MOVE would be handy for copying the input filename to the output filename space, so I can change the drive designation to D8:.

Recall that I decided to read a chunk of the input file containing 255 characters and store it in a buffer. INPUT won't work for this, since INPUT expects to see a carriage return (end-of-line character, EOL, $9B) at the end of the string being read. Similarly, PRINT adds an EOL to the output string, so that's no good for writing to the output file. I guess I'll have to write some new macros that don't involve carriage returns. These are the PUT and GET macros found in Listing 1.

New macros

Please merge Listing 1 with your existing MACRO.LIB file, using the line numbers shown. The equates at the top of Listing 1 define the error number for an end-of-file condition (EOF, $88) and the CIO command bytes for PUTCHAR and GETCHAR (put and get a string of characters, respectively) operations. RUNAD ($02E0) is a magic address for making an object code program run automatically upon loading, as we'll see a little later.

The PUT macro sends a string of bytes out to a specified IOCB. PUT takes three parameters. The first is the IOCB number to use, and the second is the address of the buffer where the data to be output resides. Parameter 3 is an optional number of bytes to be output. If %3 is less than 256 it's assumed to be a value. If greater, it's assumed to be the address of a pair of bytes containing the number of bytes to be output. If %3 is missing, only 1 byte is output.

PUT is pretty straightforward. When I wrote PUT and GET, I found they had several statements in common, so I collected those into a separate macro called PGSETUP (Lines 6780-7070 of Listing 1). PUT and GET pass two parameters to PGSETUP, which sets up the IOCB buffer address and length for the specific operation being performed.

GET is the converse of PUT. The command value of GETCHAR ($07) is used in Line 6650, rather than the PUTCHAR ($0B) of Line 6360. GET and PUT both treat EOL like any other character. However, GET does recognize the end-of-file marker (CTRL-3), which terminates the input operation and returns an error status of $88 (decimal 136).

There's one more macro in Listing 1, OPENA (Lines 7110-7360). If you look back to the OPEN macro (issue 55), you'll see that it expected parameter 4 to be a literal character string containing the name of the unit or file to open. However, in this program I'll be opening some file whose name I just read from the RAMDISK.FIL file. Hence, I need a version of an OPEN macro that can accept an address as a parameter. OPENA corrects this limitation of my original OPEN macro.

Programming, at last!

If you're like me, you'll have to restrain yourself from the temptation to dive right in and start spewing out code as soon as you think you're ready. The catch is that you probably aren't ready as soon as you think you are. However, at this point I felt I understood the problem clearly and had a design in hand that was good enough to implement. Hence, I wrote the program you see in Listing 2. Type in Listing 2 and assemble it into a file called BC59.OBJ. Once you've tested BC59.OBJ enough to convince yourself that it works (just load it from the DOS menu with option *L*), you can safely rename it to AUTORUN.SYS, so it executes upon booting.

Okay, I admit it. My program didn't look like Listing 2 when I first wrote it. I went through the usual series of testing, changing and retesting sections of code. Once I found the line I'd inadvertently deleted during editing, things went pretty smoothly. I also admit that, while coding, I found a few things I'd overlooked during the design phase, so I squeezed them into my action diagram. Incidentally, the action diagram (or other graphic representation of the program logic) will be mighty useful when you look at a program again a few months after writing it and want to figure out how in the world it works. Comments help too.

The BYTESREAD equate in Line 200 of Listing 2 is kind of interesting. Recall that the last chunk of data we read from the input file being copied may be shorter than 255 bytes, since GET stops whenever an EOF is encountered. We'll want to know just how many bytes were read in that final chunk, so we can PUT exactly the right number out to the copy of that file on the RAMdisk. The other equate, INBUFF, says we want to use the 255 bytes, beginning at address $6000, as our buffer for data read from the input file.

The first thing our action diagram says to do is try to open

the RAMDISK.FIL file. Lines 360-370 set the stage by clearing the decimal mode for arithmetic (not critical in this program, but a good practice) and clearing the display screen using the CLS subroutine from last time. Lines 380-480 attempt to open D1:RAMDISK.FIL for input using IOCB 2. If successful, we branch to label FOUNDIT and proceed. Otherwise, Lines 400-480 print an appropriate message on the screen and jump to the end of the program at label EXIT.

Recall that CIO errors leave their calling card in the Y-register. In Lines 400-410, I'm storing this value temporarily on the program stack, so I can write the first part of the error message (*MISSING*). The error number from the OPEN is retrieved in Lines 430-440. If I didn't stash the Y-register contents like this, Y would otherwise contain the error status from the PUT in Line 420, which is probably 1 and therefore not very useful! Our old subroutine, STATUSERR, tries to figure out the problem with the OPEN and prints an appropriate message. I could have used PRINT for the *MISSING* message, but that would have caused a carriage return before STATUSERR told us what happened—strictly a cosmetic decision, but appearances are important.

The action diagram next says we should read a record from RAMDISK.FIL, as in Line 580. Lines 600-690 handle any error, which should just be an end-of-file condition. This isn't really an error; it just indicates that the program's job is complete. In any case, we close IOCB 2 and exit from the program. Remember to always close IOCBs you open.

The filename read from RAMDISK.FIL is stored at address FNAMEIN. Sixteen bytes were reserved for this purpose in Line 1450. In Lines 770-800, we copy that filename to the FNAMEOUT address reserved in Line 1460, and change the drive identifier to D8:. Note that this code segment assumes the records in RAMDISK.FIL are in the form D1:FILENAME.EXT, one filename per record; more about that later.

Line 810 prints the filename on-screen, so we can keep the user informed as to what's going on. I'm a big believer in not making the user guess whether or not the computer is paying any attention to him. In Lines 820-870, we open the input file and handle any problems. This time we jump back to get the next filename, even if the present one caused some error.

The next section, at FOUNDINP, attempts to open an output file on the RAMdisk. This section will trap for situations such as no RAMdisk existing, or a full RAMdisk, or. . . Notice how many lines of code I've devoted to error handling? This is pretty typical for my programs. Just when you think you've covered every possible thing that might go wrong, either the user or the computer will get even more creatively nasty.

The actual copy routine in Lines 1080-1130 is ridiculously simple. It just alternates between reading a 255-byte chunk from the input file and writing it to the output file. When the EOF is detected, control branches to the FINISH routine at Line 1210. Now we find out how many bytes were actually read during the final GET operation, by checking the IOCB 3 buffer length, and put that number of bytes out to IOCB 4. This completes the copy process, so both IOCBs are closed. A message confirming that the copy was successful is printed, and Line 1310 jumps back to read another record from RAMDISK.FIL. What could be simpler?

Autorunning

If you want a binary (object code) program to run automatically upon loading, end it with an RTS (Return From Subroutine) instruction, as in Line 1380. This will return control to the environment from which the program was loaded, usually DOS. Setting up a file to "load-and-go" is very simple. Address RUNAD ($02E0) just has to be loaded with the address at which execution is to begin. In this program, like all of mine, the magic address is $5000. Note that, in Line 290 of Listing 2, I thoughtfully put the label START right at the beginning of the program. Then, at the very end of the program (Lines 1580-1590), I set the program counter to RUNAD and state that the 2-byte address defined by START is to be loaded into RUNAD. Very simple, very easy.

Now, you can assemble Listing 2 and save the object code under filename AUTORUN.SYS. If you aren't using a RAMdisk to house your MACRO.LIB and SUBS.LIB files, change the drive numbers in Lines 180 and 1520. Each time you boot from that disk, the .LIB files will be copied onto the RAMdisk.

But wait! How do we get the RAMDISK.FIL file? The easiest method is to use the "Copy" function from the DOS menu. Select option C for copy, and copy from device E: (the screen editor) to device D1:RAMDISK.FIL. Each line you type at the cursor and end with a RETURN will now end up in the RAMDISK.FIL file. Enter the complete filespecs for the files you want to copy to the RAMdisk, in the form *D1:MACRO.LIB*. Press RETURN after each filespec. When you're done, press CTRL-3 to create the end-of-file character, and RAMDISK.FIL is created. To verify the contents, simply copy from D1:RAMDISK.FIL to E:. Of course, you could also use any text editor that produces straight ATASCII text to create the RAMDISK.FIL entries.

Wrap-up

I hope you found "A Trip Through Karl's Brain" to be informative. Everyone writes programs a little differently, but my approach seems to get the job done. I have some other ideas for useful programs that I might present in future issues. Remember, ask not what you can do for your computer. Ask, rather, what your computer can do for you.

Despite having a Ph.D. in organic chemistry, Karl Wiegers earns a living writing applications software for photographic research at Eastman Kodak Company, mostly on an IBM mainframe. He is also interested in educational applications of Atari 8-bit, Atari ST and Apple II computers.

Listing 1
Assembly listing

```
0165 EOF =    $88
0195 GETCHAR = $07
0205 PUTCHAR = $0B
0265 RUNAD = $02E0
6180 ;
6190 ;**********************************
6200 ;
6210 ;PUT macro
6220 ;
```

```
;Usage:  PUT IOCB,address,length
;
;'IOCB' is the IOCB number to use
;'address' is a label or buffer
;address where the output data is
;'length' is the number of bytes
;to be output-if missing then =1
;
    .MACRO PUT
    .IF %0<2 .OR %0>3
      .ERROR "Error in PUT"
      .ELSE
      LDX #%1*16
      LDA #PUTCHAR
      STA ICCOM,X
      .IF %0=2
         PGSETUP  %2,1
        .ELSE
         PGSETUP  %2,%3
        .ENDIF
      JSR CIOV
      .ENDIF
    .ENDM
;
;*******************************
;
;GET macro
;
;Usage:  GET IOCB,address,length
;
;'IOCB' is the IOCB number to use
;'address' is a label or buffer
;address where the input data
;should go
;'length' is the number of bytes
;to be input-if missing then =1
;
    .MACRO GET
    .IF %0<2 .OR %0>3
      .ERROR "Error in GET"
      .ELSE
      LDX #%1*16
      LDA #GETCHAR
      STA ICCOM,X
      .IF %0=2
         PGSETUP  %2,1
        .ELSE
         PGSETUP  %2,%3
        .ENDIF
      JSR CIOV
      .ENDIF
    .ENDM
;
;*******************************
;
;PGSETUP macro
;
;Usage:  PGSETUP address,length
;
;'address' is I/O buffer address
;'length' is number of bytes for
;PUT or GET operation (value<256
;or address)
;
    .MACRO PGSETUP
    .IF %0<>2
      .ERROR "Error in PGSETUP"
      .ELSE
      LDA # <%1
      STA ICBAL,X
      LDA # >%1
      STA ICBAH,X
      .IF %2<256
        LDA #%2
        STA ICBLL,X
        LDA #0
        STA ICBLH,X
        .ELSE
        LDA %2
        STA ICBLL,X
        LDA %2+1
        STA ICBLH,X
        .ENDIF
      .ENDIF
    .ENDM
;
;*******************************
;
;OPENA macro
;
;Usage:  OPENA IOCB,ax1,ax2,add
;
;'IOCB' is IOCB number to use
;'ax1' is task number
;'ax2' is the 2nd auxiliary byte
;'add' is the address of the
;device name to be opened
;
    .MACRO OPENA
    .IF %0<>4
      .ERROR "Error in OPENA"
      .ELSE
      LDX #%1*16
      LDA #%2
      STA ICAX1,X
      LDA #%3
      STA ICAX2,X
      LDA # <%4
      STA ICBAL,X
      LDA # >%4
      STA ICBAH,X
      JSR OPENIOCB
      .ENDIF
    .ENDM
```

Listing 2
Assembly listing

```
;Program to copy a list of files
;whose names are in a file named
;D1:RAMDISK.FIL from drive D1:
;to RAM disk drive D8:
;
;by Karl E. Wiegers
;
    .OPT OBJ,NO LIST
    .INCLUDE #D8:MACRO.LIB
;
BYTESREAD = $4FFE
INBUFF = $6000
;
;*******************************
;   PROGRAM BEGINS HERE
;*******************************
;
    *=  $5000
;
START
;
;-----------------------------
;look for D1:RAMDISK.FIL; print
;error message if not found
;-----------------------------
;
    CLD
    JSR CLS
     OPEN  2,4,0,"D1:RAMDISK.FIL"
    BPL FOUNDIT
    TYA
    PHA
     PUT  0,MISSING,15
    PLA
    TAY
    JSR STATUSERR
     CLOSE  2
    JMP EXIT
MISSING .BYTE "D1:RAMDISK.FIL "
```

```
0490 ;
0500 ;-----------------------------
0510 ;read a record from RAMDISK.FIL;
0520 ;if EOF is reached, program is
0530 ;complete; print message if
0540 ;some other error crops up
0550 ;-----------------------------
0560 ;
0570 FOUNDIT
0580       INPUT  2,FNAMEIN
0590      BPL NOTEOF
0600      CPY #EOF
0610      BNE OTHERERR
0620       CLOSE  2
0630      JMP EXIT
0640 OTHERERR
0650       PRINT  UNKNOWNERR
0660       CLOSE  2
0670      JMP EXIT
0680 UNKNOWNERR .BYTE "Unknown error"
0690      .BYTE " on RAMDISK.FIL",EOL
0700 ;
0710 ;-----------------------------
0720 ;build the output file name,
0730 ;open input file, handle error
0740 ;-----------------------------
0750 ;
0760 NOTEOF
0770       MOVE  FNAMEIN,FNAMEOUT,16
0780      LDX #1
0790      LDA #56      ;ATASCII '8'
0800      STA FNAMEOUT,X
0810       PRINT  FNAMEIN
0820       OPENA  3,4,0,FNAMEIN
0830      BPL FOUNDINP
0840      JSR STATUSERR
0850       CLOSE  3
0860       CLOSE  4
0870      JMP FOUNDIT
0880 ;
0890 ;-----------------------------
0900 ;open output file, check for
0910 ;error with ramdisk
0920 ;-----------------------------
0930 ;
0940 FOUNDINP
0950       OPENA  4,8,0,FNAMEOUT
0960      BPL DOCOPY
0970       CLOSE  2
0980       CLOSE  3
0990       PRINT  RAMDERROR
1000      JMP EXIT
1010 RAMDERROR .BYTE "Problem with"
1020      .BYTE " the ramdisk...",EOL
1030 ;
1040 ;-----------------------------
1050 ;copy file in blocks of 255 bytes
1060 ;-----------------------------
1070 ;
1080 DOCOPY
1090       GET  3,INBUFF,255
1100      BMI FINISH
1110       PUT  4,INBUFF,255
1120      CLC
1130      BCC DOCOPY
1140 ;
1150 ;-----------------------------
1160 ;write the remaining number of
1170 ;input bytes, close files, go
1180 ;get the next input filename
1190 ;-----------------------------
1200 ;
1210 FINISH
1220      LDX #$30
1230      LDA ICBLL,X
1240      STA BYTESREAD
1250      LDA ICBLH,X
1260      STA BYTESREAD+1
1270       PUT  4,INBUFF,BYTESREAD
1280       CLOSE  3
1290       CLOSE  4
1300       PRINT  OKAY
1310      JMP FOUNDIT
1320 OKAY .BYTE "Copied okay",EOL
1330 ;
1340 ;-----------------------------
1350 ;RTS lets this be AUTORUN.SYS
1360 ;-----------------------------
1370 ;
1380 EXIT RTS
1390 ;
1400 ;-----------------------------
1410 ;space for input & output
1420 ;filenames
1430 ;-----------------------------
1440 ;
1450 FNAMEIN      16
1460 FNAMEOUT      16
1470 ;
1480 ;-----------------------------
1490 ;don't forget the subroutines!
1500 ;-----------------------------
1510 ;
1520      .INCLUDE #D8:SUBS.LIB
1530 ;
1540 ;-----------------------------
1550 ;set up for autorun on loading
1560 ;-----------------------------
1570 ;
1580      *=  RUNAD
1590      .WORD START
```

Attention Programmers!

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

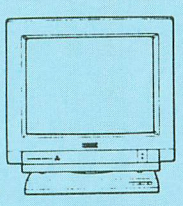
All submissions for publication, both program listings and text, should be provided in printed and magnetic form. Typed or printed copy of text is mandatory and should be in upper and lower case with double spacing. By submitting articles to **ANALOG Computing**, authors acknowledge that such materials,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

For those of you who are sincerely interested in the rules and regulations for publication, we've taken this opportunity to print our guidelines for authors. See page 128 of this book for everything you'll need to know.

Send your programs and articles to:
Editor, **ANALOG Computing**
P.O. Box 23, Worcester, MA 01603.

MAGNAVOX

EGA MONITOR CM9043



- Full compatibility with the IBM Enhanced Graphics Adapter Card, or equivalent
- High contrast 14" CRT (13" visual)
- Glare-reducing etched glass tube face
- 64 color capability in Enhanced Graphics (EGA) mode
- 16 color capability in Color Graphics (CGA) mode
- Up to 640 dots resolution (horizontal)
- Up to 350 lines resolution (vertical, Enhanced Graphics mode)
- Text mode display function in green or amber

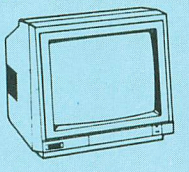
$339

EGA Card – $125

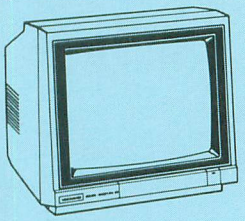
8CM515

New

$269.95



Cable 515 Magnavox to Amiga Computer – $13.95



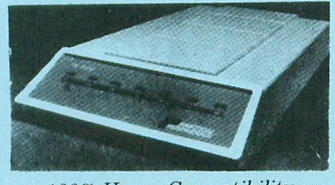
$184.95

CM8502 Color Monitor 40

• Accepts composite video input • 330 dots resolution (horizontal) • 350 lines resolution (vertical) • Green text display switch • Variable sharpness control

Modems

2400 Baud External	139.95
2400 Baud Internal	139.95
1200 Baud External	79.95
1200 Baud Internal	64.95



100% Hayes Compatibility

Hayes is a Registered Trademark of Hayes

- **Full manufacturer warranty.**
- **Personal check 3 weeks clearance.**
- **Return authorization required.**
- **Prices show 3% cash discount.**
- **Compatibility not guaranteed.**
- **COD accepted.**

APO, FPO, International: add $5 plus 3% priority. No sales tax outside N.Y. Prices/availability subject to change.

Star micronics

NEW NX1000

NLQ – 36 CPS

DRAFT – 144 CPS

BEST PRICE CALL

NX1000 Rainbow Color Printer – Call

NICKEL CITY ELECTRONICS

Orders

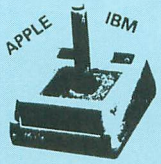
1-800-634-1870

Tech Support

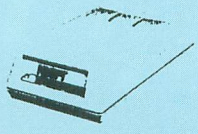
1-716-684-7350

P.O. Box 1025, Buffalo, NY 14225

SUPER SPECIAL OF THE MONTH



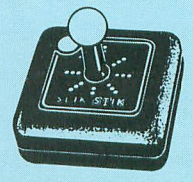
KC3
APPLE IBM
$16.99



KRAFT
MICROMOUSE IBM
$39.95

Cables

IBM Parallel Cable 6'	5.99
IBM Parallel Cable 10'	7.99
M-M RS232 Cable 6'	7.95
M-M RS232 Cable 10'	8.49
M-F RS232 Cable 6'	7.95
M-F RS232 Cable 10'	8.49
IBM Modem Cable 6'	5.99
IBM Modem Cable 10'	7.99
IBM at Modem Cable	5.99
M-M Centronics 6'	11.95
M-F Centronics	11.95
M-M Centronics 10'	12.95
IBM Keyboard Extension	4.99
Monitor Extension-DB9 M-F	11.95
Monitor Cable-DB9 M-M	11.95
IBM Color RGB Cable-DB9 Cable to 8 Pin Din	12.95
Mac' to Imagewriter	9.99
Mac' to Imagewriter II	9.99
Mac' to Hayse	9.99



Accessories

A/B DB9 Switch Box	32.95
ABCD DB9 Switch Box	49.95
A/B Telephone Switch Box RJ11	34.95
Serial Cross-Over Switch Box	39.95
Parallel Cross-Over Switch Box	39.95
M-M DB9 Gender Changer	5.95
F-F DB9 Gender Changer	5.95
A/B Parallel Switch Box	22.95
A/B Serial Switch Box	22.95
A/B/C/D Parallel Switch Box	39.95
A/B/C/D Serial Switch Box	39.95
6 Outlet AC Surge Pr	9.95
B-109 Parallel Card	29.95
B-106 Serial Card	26.95
Male/Male Gender Chg	5.95
Feml/Feml Gender Chg	5.95
Winner 200	11.99
Winner 104	4.99
Winner 770 Com/Atari	11.95
Winner 909 Apple/IBM	23.95
Winner 220 Joystick	13.95
500XJ Atari/Comm	14.95
Competition Pro 5000	14.95
Kraft Apple/IBM KC 3	24.95
Mazemaster Atari/C64	14.95
#31 Apple Mouse	32.95
Starmaster Joystick	5.49
Joystick Ast Ada. Kit	27.95
Kraft Ace Joystick	5.49
Kraft Premium 2 IBM	26.95
Apple Mach 3 Stick	12.95
Apple Kraft Stick	8.95
Mouse Joystick	12.95
Competition Pro	15.95
Suncom Slik Stiks	5.99
Suncom Econo Stik	4.99
Suncom Tac 2	9.99
Suncom Tac 3	11.99
Suncom Tac 5	13.95
Suncom Starfighter	17.85
Icon Controller	15.99
Tac 1 +	24.95
Starfighter IBM	17.85
Starfighter C64/Atar	8.95
Terminator Joystick	19.95

$3.99

Disk Label Kit

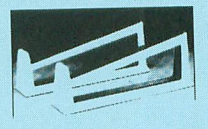
MEMOREX

UNIVERSAL PRINTER STAND

MEMOREX

Printer Support System

$6.99



- Two-Piece Stand Saves Space
- Tilt-In Better View
- Supports Printers and Portable Computers

PAPER

#0150
Ink Jet
500 Count
$14.99

#0200
4 Color - 200 Sheets
$ 6.99

#0024
2400 Sheets - 20 lb.
$22.95

DISKETTES

3.5	SS/DD	13.99
3.5	DS/DD	16.99
5.25	SS/DD	5.99
5.25	DS/DD	8.99
5.25	DS/HD	16.99

20# 250 SHEETS

$4.99 12 + **$3.99**

#0035
Thermal Transfer
500 Ct.
$12.99

Circle #109 on reader service card.

Refresh Your Memory

And Keep Your Cool.

Introducing the ST Hard Drive System from ICD that refreshes your memory better than any other ST hard drive around. *No problem.*

It's the drive that not only looks cool, but stays cool too. All because of a built-in fan that knows exactly how to beat the heat and maintain a calm, cool and collected environment . . . even in *your* most heated situations. *No sweat.*

And, it's the hard drive that takes a refreshing approach to aesthetic case design as well. See for yourself. It's easy on the space, fitting perfectly under the monitor. And it's easy on the eyes, tailored to look great in the company of your Atari ST. With adjustable front legs, your monitor gets the lift it needs for comfortable viewing. *No strain.*

Despite a sleek and compact exterior, the ICD ST Hard Drive System is packed full of overwhelming enhancements. Like an internal clock that tags each file with up-to-the-minute time and date information. Not to mention expansion capabilities that welcome the connection of up to six SCSI devices and daisy-chaining Atari's DMA Bus (ACSI). It's available in more memory capacities than you can imagine. With storage ranging from 20 megabyte systems up to 280 megabytes. And, there's dual drives too, that double your protection and double your confidence. *No stress.*

So, the next time you think about a hard drive for your Atari ST, think about the countless ways we can refresh your memory. It's the only drive worth remembering. Because it's from ICD. *No wonder.*

For further product information, please call or write for our catalog today.

ICD
1220 Rock Street
Rockford, IL 61101-1437
(815)968-2228
MODEM: (815)968-2229
FAX: (815)968-6888

Atari ST is a trademark of Atari Corporation.
Circle #110 on reader service card.